Electronic Colloquium on Computational Complexity, Report No. 100 (2023)

Learning in Pessiland via Inductive Inference

Shuichi Hirahara[*] Mikito Nanashima[†]

Abstract

Pessiland is one of Impagliazzo's five possible worlds in which NP is hard on average, yet no one-way function exists. This world is considered the most pessimistic because it offers neither algorithmic nor cryptographic benefits.

In this paper, we develop a unified framework for constructing strong learning algorithms under the non-existence of a one-way function, indicating a positive aspect of Pessiland. Using our framework, we improve the learning algorithm for *adaptively changing distributions*, which was introduced by Naor and Rothblum (ICML'06). Although the previous learner assumes the knowledge of underlying distributions, our learner is *universal*, i.e., does not assume any knowledge on distributions, and has better sample complexity. We also employ our framework to construct a strong agnostic learner with *optimal sample complexity*, which improves the previous PAC learner of Blum, Furst, Kearns, and Lipton (Crypto'93). Our learning algorithms are *worst-case* algorithms that run in exponential time with respect to computational depth, and as a by-product, we present the first characterization of the existence of a one-way function by the *worst-case* hardness of some promise problem in AM. As a corollary of our results, we establish the robustness of average-case learning, that is, the equivalence among various average-case learning tasks, such as (strong and weak) agnostic learning, learning adaptively changing distributions with respect to arbitrary unknown distributions, and weak learning with membership queries with respect to the uniform distribution.

Our framework is based on the theory of Solomonoff's inductive inference and the universal extrapolation algorithm of Impagliazzo and Levin (FOCS'90). Conceptually, the framework demonstrates that Pessiland is, in fact, a wonderland for machine learning in which various learning tasks can be efficiently solved by the generic algorithm of universal extrapolation.

[*]National Institute of Informatics, Japan. `s_hirahara@nii.ac.jp`
[†]Tokyo Institute of Technology, Japan. `nanashima@c.titech.ac.jp`

ISSN 1433-8092

Contents

1 Introduction

One of the major open questions in theoretical computer science is to base the security of a cryptographic primitive on the hardness of NP. In an influential paper of Impagliazzo [Imp95], he clearly addressed the gap between the hardness of NP and the existence of a cryptographic primitive, by proposing the notion of five possible worlds. *Pessiland* is one of the five possible worlds in which NP is hard to solve on average but there exists no one-way function (OWF)—a cryptographic primitive whose existence is often considered to be a minimal assumption for complexity-based cryptography [IL89]. This is the most pessimistic possibility of our world: In Pessiland, there is no generic heuristic algorithm that solves NP on average, yet no complexity-based cryptography is possible, which would have devastating impacts on our society. Unfortunately, so far researchers have not succeeded in excluding Pessiland from the five possible worlds, and our world could correspond to Pessiland at present. Towards closing the gap between the (average-case) hardness of NP and the existence of a secure cryptographic primitive, it is important to investigate the following question.

What kind of computational tasks can be efficiently solved in Pessiland?

Note that the security of a one-way function can be based on the hardness of any tasks that can be solved in Pessiland. The major open problem of ruling out Pessiland is equivalent to proving that there exists a heuristic algorithm that solves NP on average under the non-existence of a one-way function.

One of the most general tasks that can be achieved in Pessiland is learning *adaptively changing distributions* (ACDs), which was introduced by Naor and Rothblum [NR06]. ACDs model the setting in which two parties, Alice and Bob, interact with each other, and a learner, Eve, tries to impersonate Bob so that Alice cannot distinguish Bob's true message from Eve's imitation. Alice and Bob have some shared secret information at the beginning of the protocol, but Eve does not have access to the secret information and can just observe the communication between Alice and Bob. During the communication, Alice and Bob can *adaptively* change their internal states, which makes the learning task highly nontrivial even in a resource-unbounded setting. Specifically, the learning task for an ACD $(\mathcal{G}, D)$ is formalized as follows. (i) At the initial step, a hidden initial state $s := s_0 \in \{0,1\}^{\mathsf{poly}(n)}$ is selected according to a samplable distribution $\mathcal{G}$ (s_0 corresponds to the secret information shared by Alice and Bob); (ii) at each i-th stage, a sample x^i is generated by a polynomial-time sampler D as $(x^i, s') := D(s; r)$, where r is a hidden random seed, and D updates its internal state s to s' (D corresponds to the algorithms of Alice and Bob); (iii) a learner is given a stream of the samples $x^1, x^2, \ldots$, and the task of the learner is to choose a stage i (which we call a *prediction stage*) and to approximately simulate the conditional distribution of the next outcome x^i given the initial state s_0 and the past stream $x^1, \ldots, x^{i-1}$ without observing x^i. The main result of [NR06] is that for *each* ACD $(\mathcal{G}, D)$, there exists a polynomial-time algorithm $L_{\mathcal{G},D}$ that learns the ACD $(\mathcal{G}, D)$ if there is no one-way function (and vice versa). Naor and Rothblum [NR06, Section 1.3] argued "several (seemingly innocuous) modifications to the definition of learning ACDs would make the learning task too hard or impossible," suggesting that their learning model might be the most general learning setting which cannot be extended further.

The learning algorithm for ACDs is useful for investigating minimal assumptions required for the existence of cryptographic primitives. The original motivation of Naor and Rothblum [NR06] was to show the necessity of a one-way function for constructing online memory checking algorithms [NR09]. Similarly, Naor and Yogev [NY19] considered the question of constructing a Bloom filter in an adversarial setting, i.e., a space-efficient randomized data structure such that no efficient adversary can find a false positive given black-box access to the data structure. By using the learner for ACDs, they showed that for any "nontrivial" Bloom filter, there exists an adversary that finds a false positive under the non-existence of a one-way function; that is, the existence of a one-way function is necessary for constructing a Bloom filter with non-trivial space complexity in the adversarial setting. However, there remains an interesting loophole in the impossibility results of [NR09; NY19] because the learner $L_{\mathcal{G},D}$ of [NR06] assumes the knowledge of the ACD $(\mathcal{G}, D)$. By hiding the source code of a Bloom filter from the adversary, it might be possible to construct a non-trivial Bloom filter without using a one-way function.

2 Our Results

In this paper, we develop a unified framework for constructing efficient learners in various learning models under the non-existence of a one-way function. We use the framework to construct an improved learner for ACDs and an agnostic learner in Pessiland. The framework is based on the inductive inference of Solomonoff [Sol64a; Sol64b], which is more general than the task of learning ACDs. In Solomonoff's inductive inference, we assume that an infinite sequence $x_1, x_2, x_3, \ldots$ of symbols is generated from some unknown computable distribution. The theory of inductive inference [Sol64a; Sol64b; MF98; Hut05; LV19] shows that there is an inefficient learner that predicts the next symbol x_i given the previous symbols $x_1, \ldots, x_{i-1}$ for most choices of i. Under the non-existence of a one-way function, we prove that there exists a polynomial-time algorithm for a time-bounded analogue of Solomonoff's inductive inference. In fact, as early as 1990, Impagliazzo and Levin [IL90] suggested that this is possible, and called their (unspecified) algorithm *universal extrapolation*; unfortunately, details are not given in their paper, and the relationship among universal extrapolation and other learning models was unclear. Surprisingly, we prove that universal extrapolation improves, generalizes, and unifies previous results developed over the last three decades in the literature. Moreover, our agnostic learner achieves an optimal sample complexity up to a constant factor. Conceptually, our results suggest that Pessiland is, in fact, a wonderland for machine learning in which various learning tasks can be efficiently solved by the generic algorithm of universal extrapolation, indicating a positive aspect of Pessiland.[1] Philosophical implications of our results will be discussed in Section 4.1. We proceed to describe our results in detail.

2.1 A Unified Framework of Learning in Pessiland

As a corollary of our general framework, we improve the result of [NR06].

Theorem 2.1 (informal; see Theorem 9.7 for a formal statement)**.** *There exists no infinitely-often one-way function*[2] *if and only if there exists an efficient randomized algorithm that ϵ-closely learns every* unknown *ACD with an s-bit initial state with sample complexity $O(s \cdot \epsilon^{-2}\delta^{-1})$ with probability*

[1] In this sense, the name of Pessiland becomes a bit misleading. Following that Algorithmica is a wonderland for algorithms and Heuristica is a wonderland for heuristics, we could give Pessiland an alternative name "Learnabilica". See Section 4.1.

[2] In this paper, we mainly discuss the relationships between learnability for all large enough example sizes and all parameters (i.e., accuracy and confidence) and OWF with infinitely often security (i.e., the security holds for infinitely many seed lengths) to focus on algorithmic aspects. Note that our results also hold for OWF with sufficiently large security (i.e., the security holds for any sufficiently large seed length) by considering the learnability on infinitely many example sizes with arbitrarily small parameters fixed beforehand.

$1-\delta$. *Moreover, the learner chooses a prediction stage* $i \sim \{1, \dots, O(s \cdot \epsilon^{-2}\delta^{-1})\}$ *uniformly at random.*

Our learning algorithm for ACDs improves [NR06] in the following points. (i) The sample complexity of [NR06] is $O(s \cdot \epsilon^{-4}\delta^{-2})$, which is improved to $O(s \cdot \epsilon^{-2}\delta^{-1})$. (ii) Our learner chooses a prediction stage i uniformly at random and works at a $(1-\delta)$-fraction of i's, whereas the learner of [NR06] adaptively chooses the prediction stage.[3] (iii) Our learner is *universal*, i.e., it does not depend on the description of ACDs. In particular, by replacing the leaner for ACDs in [NY19] with Theorem 2.1, we eliminate the loophole in the result of [NY19].

Corollary 2.2 (informal; see Section 9.2.3 for details). *If there exists no one-way function, then there exists a* universal *adversary* A *such that for any source code* B *generated in time* t*, the adversary* A *finds a false positive for the non-trivial Bloom filter specified by* B *in time* $\mathsf{poly}(t)$.

In other words, to construct a non-trivial Bloom filter B in Pessiland, a legitimate user must spend (super-polynomially) more time than the adversary A.

Another consequence of our unified framework is a characterization of the existence of a one-way function by *agnostic learning* [KSS94], which generalizes PAC learning. In agnostic learning, a learner is given samples of the form (x, b) (we call $x \in \{0,1\}^n$ and $b \in \{0,1\}^{\leq n}$ an *example* and a *label* of x, respectively) that are selected identically and independently according to an unknown distribution. The goal of the learner for a target class $\mathscr{C}$ of functions is to output a good hypothesis that approximates (within an additive accuracy parameter ϵ) the best function $f \in \mathscr{C}$ that approximates the labels.[4] More precisely, for a given accuracy parameter ϵ and a distribution $\mathcal{D}$ over samples, an agnostic learner outputs a hypothesis h satisfying $\Pr_{(x,b)\sim\mathcal{D}}[h(x) \neq b] \leq \mathsf{opt}_{\mathscr{C}}(\mathcal{D}) + \epsilon$, where $\mathsf{opt}_{\mathscr{C}}(\mathcal{D}) = \min_{f \in \mathscr{C}} \Pr_{(x,b)\sim\mathcal{D}}[f(x) \neq b]$.

Theorem 2.3 (informal; see Theorem 10.6 for details). *For any (multi-output) target class* $\mathscr{C}$ *that contains polynomial-size circuits, the following are equivalent.*

1. *There exists no infinitely-often one-way function.*
2. $\mathscr{C}$ *is efficiently agnostic learnable on average when samples are independently and identically drawn from an unknown sampling algorithm with* $s(n) = \mathsf{poly}(n)$*-bit secret advice that is selected according to another unknown samplable distribution. Furthermore, the sample complexity is* $O(s(n)/\epsilon^2)$ *for any accuracy parameter* $\epsilon \in (0,1]$ *and the description length of hypotheses output by the learner is* $O(n \cdot s(n)/\epsilon^2)$.

The sample complexity $O(s(n)/\epsilon^2)$ (i.e., the number of samples required for learning) is optimal up to a constant factor; see Appendix A for a proof. Moreover, our algorithm learns the target class $\mathscr{C} = \{f : \{0,1\}^n \to \{0,1\}^n\}$ of *all the (possibly not efficiently computable) functions*— as long as samples are drawn from efficiently samplable distributions. Note that the same learning task is provably impossible in a worst-case setting because there exists a function that cannot be approximated by any efficiently computable function.

[3]This results falsify the following incorrect statement in [NR06]: "An algorithm for learning ACDs must use its knowledge of the ACD $(\mathcal{G}, D)$ in order to decide, on-the-fly, at what round it outputs its hypothesis."

[4]Strictly speaking, we focus on agnostic learning for the 0-1 loss (i.e., the prediction loss in [KSS94]) in this work. Note that our learner can be generalized to the case of general (polynomial-time computable) loss functions if the number of labels is polynomially bounded (see also Sections 3.3 and 10.3).

Theorem 2.3 generalizes the work of Blum, Furst, Kearns, and Lipton [BFKL93], which characterizes the existence of a one-way function by the hardness of an average-case variant of PAC learning, in the following points: (i) the learning model is generalized from PAC learning to agnostic learning; (ii) the label is generalized from binary to polynomial length; and (iii) the applicable settings of distributions on samples are broadened. Specifically, the work [BFKL93] only considered the case in which a target function is selected independently of the example distribution, while our learner works on average for randomly selected $\mathsf{P/poly}$-samplable joint distributions over samples. Note that the previous learner in [BFKL93] does not work in the general case of joint distributions unless Pessiland is ruled out. This is because (i) the learner in [BFKL93] is *proper*, i.e., it outputs a hypothesis in the same class of target functions, and (ii) the NP-hardness result of proper learning [PV88] implies that every distributional NP problem is reducible to proper learning 2-term DNFs in our average-case setting. We bypass this difficulty by considering improper learning.

More generally, our framework shows that any learning task which admits a "cheating learner" can be solved in Pessiland. Details will be discussed in Section 3.3.

As a consequence of our framework, we show that various average-case learning tasks are, in fact, equivalent to each other, thereby establishing the robustness of average-case learning.

Theorem 2.4 (informal; see Theorems 9.7 and 10.6 for details). *The following are equivalent:*

1. *The non-existence of infinitely-often one-way function;*

2. **(Learning ACDs.)** *Every unknown ACD is learnable in polynomial time.*

3. **(Distributional learning.)** $\mathsf{P/poly}$-*samplable distributions are distributionally learnable in polynomial time on average under every unknown sampling algorithm for the target distribution the learner attempts to statistically approximate.*

4. **(Agnostic learning.)** *The class of all (polynomial-length multi-output) functions is agnostically learnable in polynomial time on average under an unknown sampling algorithm with secret advice that is selected according to another unknown samplable distribution.*

5. **(Weak learning with membership queries.)** $\mathsf{P/poly}$ *is weakly PAC learnable in polynomial time with membership queries under a uniform example distribution and a samplable distribution (fixed beforehand) over target functions.*

Here, *distributional learning* in Item 3, introduced in [KMRRSS94], is a special case of learning ACDs, where the internal state never changes, i.e., samples are drawn independently and identically from an unknown distribution $\mathcal{D}$, and the task of the learner is to find a hypothesis that statistically simulates $\mathcal{D}$. The equivalence between Items 1 and 5 follows from the well-known relationship between a pseudorandom function generator [GGM86] and hardness of weak learning with membership queries [Val84].

In Theorem 2.4, we can observe many surprising phenomena in average-case learning, such as (i) a reduction from agnostic learning to PAC learning, (ii) a reduction from learning under general distributions to learning under the uniform distribution, (iii) boosting of accuracy, (iv) a reduction from learning distributions to learning binary classification, (v) a reduction from learning with membership queries to learning with random samples. Note that in the time-unbounded (i.e., statistical) setting, such a robustness of learnability is well known as the fundamental theorem of statistical learning [cf. SB14, Section 6], where the learnability is characterized by VC dimension.

To the best of our knowledge, the robustness of learnability was not much known in a time-bounded (i.e., computational) setting.[5] Theorem 2.4 establishes a computational variant of the fundamental theorem through the non-existence of a one-way function.

2.2 Worst-Case Learning in Pessiland

Although we stated our results in terms of average-case learning, our learning algorithms, in fact, have a *worst-case* guarantee based on the notion called *computational depth* [AFMV06; AF09]. This worst-case guarantee is important for obtaining Corollary 2.2. To introduce the notion of computational depth, we first review the notion of *time-bounded universal a priori probability*, which was used by Impagliazzo and Levin [IL90]. We fix an efficient universal Turing machine U arbitrarily. For every $t \in \mathbb{N}$, we use the notation U^t to refer to the execution of U in t steps[6]. Then, U gives rise to the following important distribution.

Definition 2.5 (Universal probability [IL90])**.** *For each $t \in \mathbb{N}$, the t-time-bounded universal distribution Q^t is defined as the distribution of the output of $U^t(r)$ for a uniformly random seed $r \sim \{0,1\}^t$. For a string $x \in \{0,1\}^*$, we define the* t-time-bounded universal a priori probability *of x as*

$$\mathrm{Q}^t(x) := \Pr_{r \sim \{0,1\}^t}\left[U^t(r) = x\right].$$

We also define $\mathrm{q}^t(x) := -\log \mathrm{Q}^t(x)$. (When $\mathrm{Q}^t(x) = 0$, we regard $\mathrm{q}^t(x)$ as ∞.)

In Appendix B, we observe that this notion is essentially equivalent to the notion of probabilistic Kolmogorov complexity, which was recently introduced by Goldberg, Kabanets, Lu, and Oliveira [GKLO22].

Now, we introduce the notion of computational depth.[7]

Definition 2.6 (Computational depth)**.** *For every $t \in \mathbb{N}$ and $x \in \{0,1\}^*$, we define the t-time-bounded computational depth $\mathrm{cd}^t(x)$ of x as*

$$\mathrm{cd}^t(x) := \mathrm{q}^t(x) - \mathrm{K}(x),$$

where $\mathrm{K}(x)$ denotes the Kolmogorov complexity of x, that is, $\min\{|d| \mid U(d) = x\}$.

This notion has a beautiful interpretation in terms of a conditional probability:

$$\mathrm{cd}^t(x) \approx -\log \Pr_{r \sim \{0,1\}^t}\left[U^t(r) = x \;\middle|\; U(r) = x\right].$$

That is, $2^{-\mathrm{cd}^t(x)}$ is approximately equal to the probability that the universal Turing machine halts in time t over a uniformly random program $r \sim \{0,1\}^t$, conditioned on the event that U prints x.

One of the fundamental properties of computational depth is that no randomized efficient algorithm can generate a computationally deep string with all but negligible probability.

[5] An important exception is boosting [e.g., Sch90].

[6] More precisely, we assume that each Turing machine has a write-only output tape, and we do not take into account whether it halts when we consider U^t.

[7] In the original definition [AFMV06], the computational depth is defined as $\mathrm{cd}^t := \mathrm{K}^t(x) - \mathrm{K}(x)$. In this work, we employ q^t instead of K^t, which gives essentially the same quantity with ones used in [Ben88; LOZ22].

Lemma 2.7 (see Lemma 6.14 for a proof)**.** *There exists a polynomial τ such that for every $p(n)$-time samplable distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$, every $n \in \mathbb{N}$, every $t \geq \tau(p(n))$, and every $k \in \mathbb{N}$,*

$$\Pr_{x\sim\mathcal{D}_n}\left[\mathrm{cd}^t(x) > k\right] \leq 2^{-k+\log t+O(1)}.$$

This indicates that most strings that can be produced efficiently have a logarithmically small computational depth. More properties of computational depth can be found in Section 6.3.

The running time of our learning algorithms is proportional to an exponential in the computational depth. For example, as long as the initial state s_0 of an ACD has shallow computational depth, our learning algorithm runs efficiently in the *worst case.*

Theorem 2.8 (informal; a worst-case variant of Theorem 2.1)**.** *There exists no infinitely-often one-way function if and only if there exists an efficient randomized algorithm L that ϵ-closely learns every unknown ACD with* every *s-bit initial state s_0 with sample complexity $O(s \cdot \epsilon^{-2}\delta^{-1})$ in time* $\mathsf{poly}(2^{\mathrm{cd}^t(s_0)}, \epsilon^{-1}, \delta^{-1})$ *with probability $1-\delta$ over the internal randomness of L and the ACD.*

Note that this is a *worst-case* learning algorithm that works for every initial state $s_0 \in \{0,1\}^s$. Theorem 2.1 and Corollary 2.2 are corollaries of Theorem 2.8 since no efficient algorithm can generate computationally deep strings. We mention that Antunes and Fortnow [AF09] showed that the running time of an arbitrary average-case algorithm for solving a *decidable problem* on input x can be characterized by $2^{O(\mathrm{cd}^t(x)+\log t)}$. However, it is unclear whether a similar characterization can be directly applicable to Theorems 2.1 and 2.8.

We can also state the result of agnostic learning in terms of computational depth.

Theorem 2.9 (informal; a worst-case variant of Theorem 2.3)**.** *For any (multi-output) target class $\mathscr{C}$ that contains polynomial-size circuits, the following are equivalent:*

1. *There exists no infinitely-often one-way function.*
2. *$\mathscr{C}$ is agnostic learnable with accuracy ϵ in time* $\mathsf{poly}(2^{\mathrm{cd}^t(z)}, \epsilon^{-1})$*, when $O(|z|/\epsilon^2)$ samples are identically and independently selected according to an unknown distribution samplable with secret advice $z \in \{0,1\}^*$.*

2.3 Worst-Case Characterization of One-Way Functions

As a by-product of our proof techniques, we obtain the first characterization of the existence of a one-way function (which has an *average-case* security requirement) by the *worst-case* intractability of some promise problem. The characterization contributes to a line of research on meta-complexity, which we briefly review below.

Impagliazzo and Levin [IL90] not only foreshadowed the significance of universal extrapolation, but also *stated* the first characterization of a one-way function based on some average-case hardness of a computational problem.

Theorem 2.10 ([IL90, "Proposition" 1])**.** *The following are equivalent.*

1. *There exists no infinitely-often one-way function.*

2. *There exists a randomized polynomial-time algorithm M such that, for every polynomial samplable distribution $\mathcal{D}$, there exists a polynomial t_0 such that for all large $n \in \mathbb{N}$, for every integer $t \geq t_0(n)$, for every $\delta^{-1} \in \mathbb{N}$,*

$$\Pr_{\substack{x \sim \mathcal{D}_n \\ M}} \left[\mathrm{Q}^t(x) \cdot (1-\delta) \leq M(x, 1^t, 1^{\delta^{-1}}) \leq \mathrm{Q}^t(x) \cdot (1+\delta) \right] \geq 1 - \delta.$$

Unfortunately, the proof of Theorem 2.10 is omitted from the paper [IL90]. As a consequence, Theorem 2.10 is not appreciated in the recent line of research on meta-complexity. It is recently that Liu and Pass [LP20] established the characterization of the existence of a one-way function by the average-case hardness of time-bounded Kolmogorov complexity *with respect to the uniform distribution*, which spawned many subsequent works. Since $\mathrm{q}^t(x)$ and the time-bounded Kolmogorov complexity $\mathrm{K}^t(x)$ are approximately equal under a standard derandomization hypothesis (which follows from Appendix B and [GKLO22]), the main difference between [IL90] and [LP20] is that the former considers every efficiently samplable distribution, whereas the latter considers the uniform distribution. Inspired by [LP20], Ilango, Ren, and Santhanam [IRS22] (among other results) presented the characterization of the existence of a one-way function by the average-case hardness of resource-unbounded Kolmogorov complexity with respect to *every efficiently samplable distribution.* This result was instrumental to obtaining the equivalence between the non-existence of a one-way function and average-case symmetry of information [HILNO23].

We observe that the result of [IRS22] is, perhaps surprisingly,[8] a corollary of Theorem 2.10. Specifically, since any output of a randomized efficient algorithm has shallow computational depth (Lemma 2.7), we have $\mathrm{q}^t(x) \approx \mathrm{K}(x)$ with high probability over a random x drawn from any unknown efficiently samplable distribution.[9] Thus, the approximation algorithm of Theorem 2.10 also approximates $\mathrm{K}(x)$ on average. We also present the "first written" proof of Theorem 2.10, which can be used to construct the universal extrapolation algorithm; see Section 7 for the proof of Theorem 2.10.

Moreover, using the notion of computational depth, we characterize the existence of a one-way function by the *worst-case* hardness of some promise problem $\Pi \in \mathsf{pr\text{-}AM}$, which asks to approximate $\mathrm{K}(x)$ for every computationally shallow input x.

Theorem 2.11. *There exists a constant c such that the following are equivalent.*

1. *There exists no infinitely-often one-way function.*

2. *The following promise problem $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}}) \in \mathsf{pr\text{-}AM}$ is in $\mathsf{pr\text{-}BPP}$.*

$$\Pi_{\mathrm{YES}} = \left\{ (x, 1^s, 1^t) \;\middle|\; \mathrm{K}(x) \leq s \textit{ and } \mathrm{cd}^t(x) \leq \log t \right\},$$
$$\Pi_{\mathrm{NO}} = \left\{ (x, 1^s, 1^t) \;\middle|\; \mathrm{K}(x) > s + c\log t \textit{ and } \mathrm{cd}^t(x) \leq \log t \right\}.$$

Recently, Hirahara [Hir23] characterized the existence of a one-way function by NP-hardness of distributional Kolmogorov complexity (under randomized reductions) and the worst-case intractability of NP. He left open the question of characterizing the existence of a one-way function

[8] In [IRS22, page 1580], the authors discussed the reason why their results were not shown in the 1990s. We observe that one of their results follows from Theorem 2.10.

[9] It is worth mentioning that the algorithm obtained from Theorem 2.10 is *universal*, i.e., it does not depend on a input distribution, whereas the algorithm of [IRS22] depends on an input distribution.

by the worst-case intractability of some natural problem. Theorem 2.11 complements this question, by giving a somewhat artificial problem whose worst-case intractability characterizes the existence of a one-way function.

3 Proof Techniques

Our main contributions are to implement the ideas of using "universal extrapolation" suggested by Impagliazzo and Levin [IL90] and to identify the connection between universal extrapolation and various learning algorithms. To this end, we first need to give a formal definition of universal extrapolation, which was not formally defined in the original paper [IL90]. In Section 3.1, we define universal extrapolation as an algorithm that predicts a next symbol according to the time-bounded universal distribution. In Section 3.2, we employ the theory of inductive inference to show that the universal extrapolation algorithm indeed predicts a next symbol with respect to any efficiently samplable distribution. Although the ideas of Sections 3.1 and 3.2 were suggested by Impagliazzo and Levin [IL90], prior to our work, the relationship between the universal extrapolation algorithm and other learning algorithms was unclear. We observe that the framework of Solomonoff's inductive inference is more general than the framework of learning ACDs, which enables us to complete the proofs of Theorems 2.1 and 2.8. Extending these ideas to agnostic learning is highly non-trivial. In Section 3.3, we introduce a general framework for translating the universal extrapolation algorithm into various learning algorithms, such as PAC learners, learners for ACDs, and agnostic learners. It is worth emphasizing that this framework enables us to construct learning algorithms in various settings, which were previously constructed using ad hock approaches in [NR06; BFKL93; Nan21a]. To obtain the optimal sample complexity of agnostic learning in Theorem 2.3, we develop a different framework based on the theory of universal prediction in Section 3.4.

3.1 Universal Extrapolation

To state universal extrapolation formally, we introduce relevant notations. For every distribution $\mathcal{D}$ over $\{0,1\}^*$, every $x \in \{0,1\}^*$, and $k \in \mathbb{N}$, we use the notation $\mathsf{Next}_k(\mathcal{D}; x)$ to refer to the conditional distribution of the k-bit prefix of a subsequent string of x selected according to $\mathcal{D}$.[10] For example, if $\mathcal{D}$ is a uniform distribution over $\{0,1\}^{\leq n} := \cup_{i\leq n}\{0,1\}^i$, then for every $x \in \{0,1\}^{\leq n}$ and every $k \in \mathbb{N}$, $\mathsf{Next}_k(\mathcal{D}; x)$ is a uniform distribution over $\{0,1\}^{\leq \min\{k, n-|x|\}}$. For any distributions $\mathcal{D}$ and $\mathcal{E}$, let $\mathsf{L}_1(\mathcal{D}, \mathcal{E})$ denote the total variation distance between $\mathcal{D}$ and $\mathcal{E}$. For every distribution $\mathcal{D}$ over strings and every $x \in \{0,1\}^*$, we define $\mathcal{D}(x) = \Pr_{x'\sim\mathcal{D}}[x' = x]$. In particular, $\mathrm{Q}^t(x)$ is the probability that x is drawn from the t-time-bounded universal distribution Q^t, which is consistent with Definition 2.5.

We formulate universal extrapolation as a randomized algorithm UE that, for a given $k \in \mathbb{N}$ and a computationally shallow string x, extrapolates the next k bits subsequent to x under the time-bounded universal distribution Q^t. We construct UE under the non-existence of one-way functions.

Theorem 3.1 (Universal extrapolation)**.** *If there exists no infinitely-often one-way function, then there exists a randomized polynomial-time algorithm* UE *such that for all* $k, t, \epsilon^{-1}, \alpha \in \mathbb{N}$ *and all*

[10]If x does not match any prefix in the support of $\mathcal{D}$, we regard $\mathsf{Next}_k(\mathcal{D}; x)$ as the distribution of the empty symbol.

$x \in \{0,1\}^*$ *with* $\mathrm{cd}^t(x) := \mathrm{q}^t(x) - \mathrm{K}(x) \le \alpha$,

$$\mathsf{L}_1\left(\mathsf{UE}(x; 1^{\langle k,t,\epsilon^{-1},2^\alpha\rangle}), \mathsf{Next}_k(\mathrm{Q}^t; x)\right) \le \epsilon.$$

Although Theorem 3.1 is stated using the notion of computational depth, it is essentially equivalent to saying that UE is an efficient heuristic algorithm that extrapolates $\mathsf{Next}_k(\mathrm{Q}^t; x)$ on average with respect to every efficiently samplable distribution [AF09].[11] The fact that UE is an efficient heuristic algorithm follows from the fact that no randomized efficient algorithm can produce computationally deep strings (Lemma 2.7).

In this work, we present two different proofs of Theorem 3.1. The first one is based on an inverter for a distributional one-way function [IL89], and the second one is based on the universal approximation algorithm of the universal a priori probability (Theorem 2.10). The former is simpler in the sense that it extrapolates the next bits simultaneously, and it is analogous to the extrapolation algorithms that have been considered in several works [Ost91; OW93; NR06; ABX08; Xia10; NY19]. The latter extrapolates next k bits inductively one-by-one, and this appears to be the original intention in [IL90].

3.2 Time-Bounded Universal Inductive Inference

Since the universal extrapolation algorithm UE predicts the next k bits subsequent to an input x according to the *time-bounded universal distribution* Q^t, it is not clear from the definition whether it predicts the next k bits according to an *unknown efficiently samplable distribution.* The theory of Solomonoff's inductive inference suggests that UE indeed extrapolates the next k bits for every unknown efficiently samplable distribution. Let $\mathrm{KL}(\mathcal{D}||\mathcal{E})$ represent the KL divergence between two distributions $\mathcal{D}$ and $\mathcal{E}$.

Lemma 3.2 (see also [Hut05; LV19])**.** *There exists a polynomial τ such that for every distribution $\mathcal{D}$ over $y^1, \dots, y^m \in \{0,1\}^*$, if $\mathcal{D}$ has a t_D-time sampler described by d bits, then for every $t, q \in \mathbb{N}$ with $t \ge \tau(d, t_D)$,*

$$\mathbb{E}\left[\mathrm{KL}\left(\mathsf{Next}_{|y^i|}(\mathcal{D}; y^1 \cdots y^{i-1}) \mid\mid \mathsf{Next}_{|y^i|}(\mathrm{Q}^t; y^1 \cdots y^{i-1})\right)\right] \le \frac{O(d)}{m},$$

where the expectation is taken over $i \sim [m]$ and $y^1, \dots, y^m \sim \mathcal{D}$.

Lemma 3.2 shows that the distribution of the next symbol y^i given $y^1, \dots, y^{i-1}$ according to an unknown distribution $\mathcal{D}$ is close to that of the next symbol y^i given $y^1, \dots, y^{i-1}$ according to the universal distribution Q^t if m is sufficiently large. Lemma 3.2 can be easily proved by the chain rule for KL divergence (Lemma 6.3), and the only property of Q^t used in the proof is that Q^t *dominates* every efficiently samplable distribution $\mathcal{D}$; that is, for each $y^1, \dots, y^m$,

$$\mathrm{Q}^t(y^1, \dots, y^m) \ge 2^{-O(d)} \cdot \mathcal{D}(y^1, \dots, y^m).$$

This domination property holds because the prefix of a random seed to U^t in the sampling process of Q^t matches the d-bit description of a sampling algorithm for $\mathcal{D}$ with probability at least $2^{-O(d)}$, in

[11] Note, however, that a similar equivalence may not be easily shown for the learning task for ACDs (Theorem 2.1 versus Theorem 2.8). Using the notion of computational depth makes our final results slightly more general.

which case Q^t is statistically identical to $\mathcal{D}$. A self-contained proof for (an extension of) Lemma 3.2 can be found in Section 9.1.

Since the distribution $\mathsf{Next}_{|y^i|}(\mathrm{Q}^t; y^1 \cdots y^{i-1})$ can be approximated by UE (Theorem 3.1), Lemma 3.2 implies that UE also approximates the distribution $\mathsf{Next}_{|y^i|}(\mathcal{D}; y^1 \cdots y^{i-1})$, that is, the next symbol with respect to an unknown distribution $\mathcal{D}$. In more detail, if we choose $m := O(d/(\delta\epsilon^2))$, then by Markov's inequality, Lemma 3.2 implies that with probability at least $1 - \frac{O(d)}{m\epsilon^2} \geq 1 - \delta$ over a choice of i and $y^1, \ldots, y^m$,

$$\mathrm{KL}\left(\mathsf{Next}_{|y^i|}(\mathcal{D}; y^1 \cdots y^{i-1}) \mid\mid \mathsf{Next}_{|y^i|}(\mathrm{Q}^t; y^1 \cdots y^{i-1})\right) \leq \epsilon^2.$$

By Pinsker's inequality (Fact 6.1), this implies

$$\mathsf{L}_1\left(\mathsf{Next}_{|y^i|}(\mathcal{D}; y^1 \cdots y^{i-1}), \mathsf{Next}_{|y^i|}(\mathrm{Q}^t; y^1 \cdots y^{i-1})\right) \leq \epsilon. \tag{1}$$

We are ready to complete a proof sketch of Theorems 2.1 and 2.8. Learning ACDs $(\mathcal{G}, D)$ can be regarded as a special case of Solomonoff's inductive inference in which the symbols $y^1, \ldots, y^m$ are generated from an ACD $(\mathcal{G}, D)$. By Eq. (1), to ϵ-closely learn ACDs with probability at least $1-\delta$, it suffices to run UE on input $y^i, \ldots, y^{i-1}$ for a random choice of $i \sim [m]$, where $m := O(d/(\delta\epsilon^2))$ and d is the length of the initial state of $(\mathcal{G}, D)$. It is worth emphasizing that the overall proofs are simple, yet the sample complexity is improved from $O(d/(\delta^2\epsilon^4))$ [NR06] to $O(d/(\delta\epsilon^2))$.

3.3 Translating Universal Extrapolation into Learning

Although our learner for ACDs can be easily obtained from the universal extrapolation algorithm UE, it is highly non-trivial to construct an agnostic learner from UE. In this section, we develop a general framework for translating UE into learning algorithms in various settings.

Generalizing the previous setting of Solomonoff's inductive inference, we consider the following setting of online learning. At each stage $i \in \mathbb{N}$, a learner first observes a string $x^i \in \{0,1\}^*$, which represents auxiliary information for the i-th prediction. A typical goal of the learner is to predict the next symbol y^i from the previous observations $x^1, y^1, \ldots, x^{i-1}, y^{i-1}, x^i$. At the end of stage i, the learner observes the outcome y^i, and proceeds to the next stage.

This general setting captures many learning problems, including PAC learning, agnostic learning, and learning ACDs. In learning ACDs, each x^i is an empty string ε, and each y^i is the i-th sample drawn from an ACD. In PAC learning, each x^i is an example independently and identically drawn from an example distribution $\mathcal{D}$, and each y^i is $f(x^i)$, where f is a target function. Agnostic learning is also captured in the same way, except that each y^i is also selected according to some distribution determined by x^i.

The same bound with Lemma 3.2 can be proved for the generalized online learning, and this enables us to show that UE also solves PAC learning. One may be tempted to conjecture that UE also solves the task of agnostic learning by just predicting the next symbol y^i. However, *the same approach fails for agnostic learning,* as the following simple example shows. Consider the distribution $\mathcal{D}$ over samples (x, b), where x is the empty string ε and b is 1 with probability $\frac{2}{3}$ and 0 with probability $\frac{1}{3}$. An optimal hypothesis for this distribution $\mathcal{D}$ is the constant-1 function ($h \equiv 1$), which achieves $\Pr_{(x,b)\sim\mathcal{D}}[h(x) = b] = \frac{2}{3}$. However, if UE is run on input $z = (\varepsilon, b^1, \varepsilon, b^2, \ldots, \varepsilon)$, where (ε, b^i) is an independent sample from $\mathcal{D}$, the output of UE is close to the distribution of b, which achieves $\Pr_{(x,b)\sim\mathcal{D}}[\mathsf{UE}(z) = b] \approx \left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2 = \frac{5}{9} \ll \frac{2}{3}$. The issue is that the universal extrapolation algorithm is designed for predicting the next symbol, which differs from the goal of agnostic learning.[12]

To fix this issue, we introduce a general framework that incorporates the goal of each learning task. The goal is specified by the notion of *cheating learner.*[13] Unlike the standard learner, a cheating learner is given access to the oracle that returns, for each access, an independent sample drawn from the distribution of the next symbol y^i conditioned on the previous observations $x^1, y^1, \dots, x^{i-1}, y^{i-1}, x^i$. Note that the oracle access is so powerful that most learning tasks become trivial. For example, cheating learners for PAC learning and learning ACDs can just output the sample returned from the oracle. The construction of a cheating learner for agnostic learning with an accuracy parameter ϵ is as follows. The cheating learner invokes the oracle $q := O(1/\epsilon^2)$ times to obtain independent samples $y_1, \dots, y_q \in \{0,1\}$ and outputs the majority of $y_1, \dots, y_q$.[14] This cheating learner achieves the goal of agnostic learning because for any example x, the output of the cheating learner achieves $\max_{y^* \in \{0,1\}} \Pr_{(x',y')}[y' = y^* | x' = x]$ up to an additive error ϵ.

To obtain an agnostic learner in the standard learning model, we need to eliminate the oracle access from the cheating learner. The next theorem shows that, by replacing the oracle access with UE, a cheating learner can be transformed into a standard learner.

Theorem 3.3 (informal; see Theorem 9.1 for details)**.** *Let L_{cheat} be a randomized cheating learner of query complexity q. Then, there exists a learner L such that for every polynomial-time samplable distribution $\mathcal{D} = \{\mathcal{D}_z\}_{z \in \{0,1\}^*}$, where each $\mathcal{D}_z$ is over streams $x^1, y^1, \dots, x^m, y^m$ (for $m := m(z)$), every $z \in \{0,1\}^*$, every sufficiently large $t \in \mathbb{N}$, every $\delta^{-1}, \lambda^{-1}, \alpha \in \mathbb{N}$, if $m \geq O(|z| \cdot q \cdot \delta^{-1}\lambda^{-2})$ and $\mathrm{cd}^t(z) \leq \alpha$, then*

$$\Pr\left[\mathsf{L}_1\left(L(x^1, y^1 \dots, x^{i-1}, y^{i-1}, x^i; 1^{\langle t, 2^\alpha, \delta^{-1}, \lambda^{-1}\rangle}), L^{\mathcal{E}}_{\mathrm{cheat}}\right) \leq \lambda\right] \geq 1 - \delta,$$

where the probability is taken over $i \sim [m]$ and $x^1, y^1 \dots, x^m, y^m \sim \mathcal{D}_z$, and $\mathcal{E}$ is an oracle that returns a sample drawn from the conditional distribution of y^i given $x^1, y^1 \dots, x^{i-1}, y^{i-1}, x^i$ with respect to $\mathcal{D}_z$. Here, the learner $L(x^1, y^1 \dots, x^i; 1^a)$ is defined as $L_{\mathrm{cheat}}^{\mathsf{UE}(x^1, y^1, \dots, x^i; 1^{a'})}$ for some parameters a and a'.

This theorem provides a unified framework for constructing learners in various learning models in Pessiland. Any cheating learner, which is often easy to construct, can be implemented as an efficient learning algorithm that uses the universal extrapolation algorithm UE.

The proof of Theorem 3.3 is similar to the construction of our universal learner for ACDs in Section 3.2. As in Lemma 3.2, even if q samples are drawn from the distributions, the KL divergence between $\mathcal{E} := \mathsf{Next}_{|y_i|}(\mathcal{D}_z; x^1, y^1, \dots, x^i)$ and $\mathsf{Next}_{|y_i|}(\mathrm{Q}^t; x^1, y^1, \dots, x^i)$ is at most $q \cdot \frac{O(d)}{m}$ (see Lemma 9.2 for the formal statement). By Theorem 3.1, the latter distribution can be approximated by UE. Thus, $L^{\mathcal{E}}_{\mathrm{cheat}}$ is statistically close to $L^{\mathsf{UE}}_{\mathrm{cheat}} =: L$.

Although Theorem 3.3 enables the statistical simulation of every polynomial-time cheating learner by using UE for a sufficiently large m, the sample complexity m depends on the number of queries q. This results in suboptimal sample complexity in the case of agnostic learning. Specifically, the cheating agnostic learner with an accuracy parameter $\epsilon \in (0,1)$ makes $q = O(\epsilon^{-2})$ queries to

[12]This example indicates that our framework Theorem 9.1 is a *strict* generalization of the framework of [NR06].

[13]The term "cheating" is due to the fact that the learner can freely read the future through the oracle without making any prediction from the past.

[14]For simplicity, we consider agnostic learning for binary-output functions.

the oracle. In addition, we need to set the parameter $\lambda := O(\epsilon)$ so that the learner simulates the cheating learner within an approximation error $O(\epsilon)$. Furthermore, we need to set $\delta := O(\epsilon)$ to bound the failure probability of the simulation above by $O(\epsilon)$ over the choice of an example. As a result, the learner obtained by Theorem 3.3 requires $O(q \cdot \lambda^{-2} \cdot \delta^{-1}) = O(\epsilon^{-5})$ samples, which is much worse than the optimal dependence ϵ^{-2} in Theorem 2.3; see Corollary 9.14 for details.

3.4 Improving Sample Complexity in Agnostic Learning

In order to improve the sample complexity of the agnostic learner, we present another framework. The main reason why the bound obtained from Theorem 3.3 is loose for agnostic learning is that the setting is too general. The main idea of improving the sample complexity is to restrict ourselves to a setting in which a cheating learner tries to minimize a certain loss function. In such a setting, we obtain the sample complexity m that does not depend on the query complexity of a cheating learner, which enables us to construct the agnostic learner with the optimal sample complexity (Theorem 2.3).

In more detail, we consider the following specific task of cheating learners. Let $b \in \mathbb{N}$ be the length of each label y^i. At stage i, a learner is given a stream $x^{<i} := x^1, y^1 \dots, x^{i-1}, y^{i-1}$ and x^i and chooses an action α from a set $\mathcal{A}$ of actions. For a loss function $l \colon \mathcal{A} \times \{0,1\}^b \to [0, C]$, where $C > 0$ is a constant, the goal of the learner is to minimize the expected loss $\mathbb{E}_{x^i, y^i}[l(\alpha, y^i)]$. For instance, agnostic learning is captured as the case where the action set $\mathcal{A}$ is the set $\{0,1\}^b$ of labels, and the loss function $l \colon \{0,1\}^b \times \{0,1\}^b \to [0,1]$ is the 0-1 loss function defined as

$$l(\alpha, y) = \begin{cases} 0 & \text{if } \alpha = y, \\ 1 & \text{if } \alpha \neq y. \end{cases}$$

In this case, the expected loss is

$$\mathop{\mathbb{E}}_{x^i, y^i}[l(\alpha, y^i)] = \Pr_{x^i, y^i}[\alpha \neq y^i],$$

whose minimization is the goal of agnostic learning.

Now, we state our framework for a cheating learner that tries to minimize an expected loss. The following theorem shows that any cheating learner can be transformed into a learner that achieves an approximately minimum expected loss by replacing the oracle with UE.

Theorem 3.4 (informal; see Theorem 10.2 for a formal statement)**.** *Let $b \in \mathbb{N}$. Let $\mathcal{A}$ be a set of actions, and let $l \colon \mathcal{A} \times \{0,1\}^b \to [0, C]$ be a loss function. For every cheating learner L_{cheat} that outputs an action in $\mathcal{A}$, there exists a learner L such that for every polynomial-time samplable distribution $\mathcal{D} = \{\mathcal{D}_z\}_{z \in \{0,1\}^*}$, where each $\mathcal{D}_z$ is over streams $x^1, y^1, \dots, x^m, y^m$, for every $z \in \{0,1\}^*$, every sufficiently large $t \in \mathbb{N}$, and every $\epsilon^{-1}, \delta^{-1}, \alpha \in \mathbb{N}$, if $m \geq O(|z| \cdot C^2 \cdot \epsilon^{-2}\delta^{-2})$ and $\mathrm{cd}^t(z) \leq \alpha$, then*

$$\Pr_{i, x^{<i}}\left[\mathop{\mathbb{E}}_{x^i, y^i, L}\left[l(L(x^{<i}, x^i; 1^{\langle b, t, 2^\alpha, \epsilon^{-1}, \delta^{-1}\rangle}), y^i)\right] \leq \mathop{\mathbb{E}}_{x^i}\left[\min_{\alpha \in \mathcal{A}} \mathop{\mathbb{E}}_{y^i}[l(\alpha, y^i)]\right] + 2\Delta + \epsilon\right] \geq 1 - \delta,$$

where $x^{<i} := x^1, y^1 \dots, x^{i-1}, y^{i-1}$ is selected according to $\mathcal{D}_z$; x^i, y^i are selected according to the conditional distribution of the i-th observation given $x^{<i}$ under $\mathcal{D}_z$, and

$$\Delta := \sup_{\mathcal{O}: \text{distribution over } \{0,1\}^b} \left(\mathop{\mathbb{E}}_{\substack{L^{\mathcal{O}}_{\text{cheat}} \\ y \sim \mathcal{O}}}[l(L^{\mathcal{O}}_{\text{cheat}}, y)] - \min_{\alpha \in \mathcal{A}} \mathop{\mathbb{E}}_{y \sim \mathcal{O}}[l(\alpha, y)] \right).$$

Here, the learner $L(x^{<i}, x^i; 1^a)$ is defined as $L_{\mathrm{cheat}}^{\mathsf{UE}(x^{<i}, x^i; 1^{a'})}$ for some parameters a and a'.

We often consider a situation in which $\Delta \approx 0$, i.e., the cheating learner outputs $\alpha \in \mathcal{A}$ that approximately minimizes $\mathbb{E}_{y\sim\mathcal{O}}[l(\alpha, y)]$, in which case we have $\mathbb{E}\big[l(L_{\mathrm{cheat}}^{\mathcal{O}}, y)\big] \approx \min_{\alpha\in\mathcal{A}} \mathbb{E}_{y\sim\mathcal{O}}[l(\alpha, y)]$ for every oracle $\mathcal{O}$. In this case, Theorem 10.2 shows that with high probability over i and $x^{<i}$,

$$\mathop{\mathbb{E}}_{x^i, y^i, L}\left[l(\alpha_L, y^i)\right] \lesssim \mathop{\mathbb{E}}_{x^i}\left[\min_{\alpha\in\mathcal{A}} \mathop{\mathbb{E}}_{y^i}[l(\alpha, y^i)]\right],$$

where α_L is the choice of the action taken by the learner L. This inequality means that the learner L achieves the best expected loss up to a small additive error. Moreover, the sample complexity m does not depend on the number of queries of the cheating learner.

Now, we can obtain an agnostic learner with optimal sample complexity as follows. Let L_{cheat} be the cheating learner for agnostic learning with an accuracy parameter $\epsilon > 0$ (that is constructed in Section 3.3). This cheating learner minimizes the expected 0-1 loss up to an additive error ϵ, and thus we have $\Delta \le \epsilon$. Applying Theorem 10.2 to L_{cheat}, we obtain an efficient learner $L := L_{\mathrm{cheat}}^{\mathsf{UE}}$ that achieves the best expected 0-1 loss up to an additive error $O(\epsilon)$ with sample complexity $m = O(|z| \cdot \epsilon^{-2})$ (for a constant $\delta > 0$), which completes a proof sketch of Theorem 2.3.

The proof of Theorem 3.4 is based on the theory of universal prediction [MF98; Hut05]. In a nutshell, the theory shows that the minimization of an expected loss under the *time-unbounded* universal distribution Q^∞ yields the minimization of an expected loss under any computable distribution, which is dominated by Q^∞. We apply the theory to the time-bounded universal distribution Q^t. However, this presents a technical challenge because the theory of universal prediction deals with *time-unbounded* settings. The main issue is that there may not exist a polynomial-time algorithm that minimizes an expected loss under Q^t. We close this computational-statistical gap by assuming that $\Delta \approx 0$ in Theorem 3.4, that is, a cheating learner efficiently minimizes the expected loss. See Section 10 for details.

4 Discussion

4.1 "Learnabilica": An Alternative Name of Pessiland

Our results suggest that Pessiland is a wonderland for machine learning. In order to highlight the positive aspect of Pessiland, we put forward an alternative name of Pessiland—"Learnabilica". This is a world in which NP is hard on average but various learning tasks can be efficiently solved on average. As shown in Theorem 2.4, Learnabilica is equivalent to Pessiland. In this section, we discuss the property of Learnabilica and compare it with Heuristica.

Recall that Heuristica [Imp95] is a hypothetical world in which NP is hard in the worst case but easy on average. This is a wonderland for heuristics because there exists an efficient heuristic algorithm that solves every problem in NP on average. In Heuristica, it is hard to find a hard instance on which heuristic algorithms for NP fail. This can be explained using the notion of computational depth. In Heuristica, there exists a heuristic algorithm that solves any NP problem on input x in time $2^{O(\mathrm{cd}^t(x) + \log t)}$ [AF09; Hir21; LOZ22]. Thus, to generate a hard instance for the heuristic algorithm, one needs to generate a computationally deep string, which is very difficult: By Lemma 2.7, any efficient randomized algorithm cannot generate computationally deep strings except for a negligible probability. In fact, the randomized algorithm fails to generate computationally deep strings unless a coin flip sequence given to the algorithm happens to be computationally shallow. This follows from the *slow growth law of computational depth* (Theorem 6.13), which states that no efficient algorithm can rapidly increase computational depth. Thus, in Heuristica, NP is easy for all practical purposes. Note that NP contains many important problems in practice, such as optimization problems and circuit minimization.

Similarly, in Learnabilica, there exists a learning algorithm that solves various learning tasks in exponential time with respect to computational depth. Just as in the case of Heuristica, it is very difficult to find a hard instance on which the learning algorithm fails. In Learnabilica, various learning tasks can be solved efficiently. By designing cheating learners and using our frameworks, one can obtain many learning algorithms. For example, we can design learning algorithms that perform universal top-k prediction and universal likelihood estimation, as we show in Sections 9.3 and 9.4, respectively. Here, top-k prediction refers to the very natural task of predicting the next outcome by producing the top k most likely candidates with the estimated likelihood for a given $k \in \mathbb{N}$, e.g., in the weather forecast, $\{(\text{sunny}, 0.8), (\text{cloudy}, 0.15), (\text{rainy}, 0.02)\}$ when $k = 3$. Top-k prediction is a common task in recommendation systems. Likelihood estimation refers to the task of estimating the probability that a given label is observed as the next outcome within an additive error, e.g., the probability of "rainy" in the weather forecast. Thus, in Learnabilica, weather forecasts are extremely precise, recommendation systems are highly effective, and many machine learning tasks that are important in practice can be easily solved.

It is an interesting research question whether an efficient "universal artificial intelligence" can be constructed in Learnabilica. The definition of a universal artificial intelligence was proposed by Hutter [Hut05]; however, the universal artificial intelligence involves solving optimization problems, which may not be solvable in Learnabilica.

4.2 Limitations

Our learning algorithms run in exponential time in the computational depth of secret information. It is natural to ask whether the running time can be improved to polynomial time in the worst case under the non-existence of a one-way function. Unfortunately, this is not known even under the stronger assumption that NP is easy on average (i.e., $\mathsf{DistNP} \subseteq \mathsf{HeurP}$): A learning algorithm that runs in polynomial time can be used to break the security of a cryptographic primitive called auxiliary-input one-way function [ABX08; Nan21a]. However, Hirahara and Nanashima [HN22] showed that there exists no relativizing proof for the implication from the average-case easiness of NP to the non-existence of auxiliary-input one-way functions. Since all the proofs in this work are relativizing, improving the running time of our learning algorithms requires fundamentally new ideas.

One could argue that our universal learning algorithm for ACDs is inferior to the learning algorithm for *specific* ACDs $(\mathcal{G}, D)$ of [NR06] because the running time of our algorithm is proportional to an exponential in the computational depth of the description of $(\mathcal{G}, D)$, which is a constant but could be a huge constant. This is an unfair comparison because the universal learner is not given the knowledge about the distribution. It is easy to incorporate prior knowledge in our universal learning algorithm, which improves the running time mildly; see Appendix D for details. We emphasize that the exponential dependence in computational depth is only relevant to the time complexity of our learning algorithms, but not to the sample complexity. For example, the sample complexity of Theorem 2.8 does not depend on computational depth.

4.3 Future Research Directions

One of the most important future research directions is excluding Pessiland by improving the learnability result. Recall that, in this work, we generalized the learnability results previously shown in [BFKL93; NR06; Nan21a] to the more general average-case setting; e.g., from distributions separated into an example distribution and a distribution over functions [BFKL93; Nan21a] to joint distributions over samples. Towards the goal an important future research direction is to achieve the same task under some "restriction" on hypotheses, which makes learning problems closer to NP-hard problems. Recently, Hirahara [Hir22] proved the NP-hardness of MINLT (i.e., learning by hypotheses with the *minimum* description); thus, proving the feasibility of MINLT in the average-case setting in Theorem 2.3 under the non-existence of OWF implies *excluding Pessiland*. This approach appears hopeful at present in the sense that the proof presented in [Hir22] is non-relativizing, and such a breakthrough does not contradict our current knowledge. Generally, the description length of hypotheses our learner produces significantly depends on the sample complexity. Therefore, as in Theorem 3.4, improving the sample complexity (depending on the minimum description length of the consistent hypothesis in this case) can be one approach for excluding Pessiland from learning theory. Another approach is to consider restricted average-case settings at first and to generalize the result. According to this approach, in Appendix C, we show that MINLT is feasible on average in the restricted average-case setting studied in [BFKL93] under an additional derandomization assumption. We remark that extending Appendix C to the average-case setting in Theorem 2.3 implies excluding Pessiland.

Another future research direction is to show reductions similar to Theorem 2.4 for more restricted classes such as NC^1 and $\mathsf{AC}^0[p]$ instead of $\mathsf{P/poly}$. In computational learning theory, several novel learners have been developed for restrictive classes such as DNFs and $\mathsf{AC}^0[p]$, but almost all the learners require membership queries and work only under the uniform example distribution [LMN93; KM93; Jac97; CIKK16; CIKK17]. By showing a result similar to Theorem 2.4 for such restrictive classes, we can convert the previous (somewhat theoretical) learners into more capable and practical learners.

Other future research direction is to establish the robust learning theory for the case where samples are selected an unknown $\mathsf{P/poly}$-samplable distribution (i.e., worst-case with respect to $\mathsf{P/poly}$-samplable distributions) under the non-existence of *auxiliary-input* one-way functions.

Open Question 4.1. Can we show the equivalence of the following by a relativizing proof? Or, is there any oracle separation (i.e., the barrier against relativizing proofs)?

- The non-existence of auxiliary-input one-way function;
- Weak learning for $\mathsf{P/poly}$ with membership queries under all unknown $\mathsf{P/poly}$-samplable distributions over samples.
- Agnostic learning under all unknown $\mathsf{P/poly}$-samplable distributions over samples.
- Distributional learning for all unknown $\mathsf{P/poly}$-samplable distributions.

Either result, i.e., the equivalence or the oracle separation, will provide important knowledge. The former will yield new clear insights into the dichotomy between learning theory and cryptography along with this work; the latter will show that the robustness of learning in Theorem 2.4 is indeed a unique property of average-case learning. We remark that Xiao [Xia10] presented a related oracle separation between weak learning and distributional learning, but they considered not efficiently samplable distributions (over examples) in weak learning.

5 Related Work

Valiant [Val84] observed that the existence of pseudorandom functions (which is equivalent to the existence of one-way functions [GGM86; HILL99]) implies the hardness of learning for $\mathsf{P/poly}$. A line of subsequent work proved the cryptographic hardness of learning for several natural subclasses [KV94; AK95; Kha93; KS09; DV21]. Compared to our work, these works investigated the opposite direction, i.e., from learning to the non-existence of cryptographic primitives.

Several learnability results have been shown under some related algorithmic assumptions. As mentioned in the introduction, Impagliazzo and Levin [IL90] initiated the study on efficient learning under the non-existence of OWF, and subsequent works [BFKL93; NR06; NY19] investigated the capability of learning in more standard models. Oliveira and Santhanam [OS17] presented the characterization of the existence of exponentially secure one-way functions based on the exponential hardness of PAC learning with membership queries on the uniform example distribution in the *non-uniform* setting, which is incomparable with our result. Nanashima [Nan21a] considered another average-case variant of PAC learning, where a target function is selected according to a fixed samplable distribution but examples are selected according to unknown (possibly not efficiently samplable) distributions as in the original PAC learning model. He showed that PAC leaning is feasible in such a model under the stronger assumption that there is no auxiliary-input one-way function. Although his idea can be applied in the case of infinitely-often one-way functions, the learner only works in the same setting as [BFKL93], which is improved by this work in the same sense as discussed in Section 2. A line of study [CIKK16; CIKK17; BCKRS22] presented distribution-specific efficient learners under the stronger assumptions of the existence of natural proofs. Li and Vitányi [LV89] developed a universal PAC learner on simple distributions that contain $\mathsf{P/poly}$-computable distributions under the stronger assumption that every NP problem is easy on average. Hirahara and Nanashima [HN21] also developed a universal agnostic learner that works on all unknown $\mathsf{P/poly}$-samplable distributions over examples under the stronger assumption that every NP problem is easy on average with zero error. Nanashima [Nan20] showed that PAC learning $\mathsf{P/poly}$ is feasible under the non-existence of a cryptographic primitive called an auxiliary-input local hitting set generator, which has a significantly weaker security condition than OWF. In another line of research, several cryptographic primitives were constructed based on the hardness of learning linear functions with random noise (e.g. [Reg09; DP12]). In this context, our result can be regarded as the construction of a one-way function whose security is based on far more general assumptions of the average-case hardness of learning.

Our result has a similarity to the well-known Yao's next-bit generator theorem [Yao82] in the following sense. The next-bit generator theorem shows that, we can translate a distinguisher (from random strings) into a next-bit predictor P for an efficiently computable function that expands secret information (selected uniformly at random). Here, P weakly simulates the next bit of the outcome on average over the choice of the secret information and a position of the simulated bit. By contrast, through Theorem 3.3 for the trivial one-query cheating learner, we can translate a distributional inverter into a universal *next-block* predictor such that for every function that expands secret information (selected according to some samplable distribution) as blocks, the generator strongly simulates the distribution of the next block of the outcome on average over the choice of the secret information and a position of the simulated block. Vadhan and Zheng [VZ12] obtained a related equivalence result between conditional pseudo-entropy and KL-hardness by using the uniform min-max theorem [VZ13], but their work appears to have no direct connection to learnability discussed in this work, such as learning ACDs and agnostic learning.

Klivans, Lee, and Wan [KLW10] introduced another formulation of agnostic learning on average, where they assumed that a target function f is selected according to some samplable distribution and ask a learner to learn all target functions that are close to f with high probability over the choice of f. Unfortunately, our learning algorithm does not work under this formulation because every case of adversarial noise must be considered regardless of samplability. Nevertheless, our learner works as long as the unknown adversary that determines noise is selected according to a uniform and efficient sampling mechanism. This computational assumption appears natural under the strong form of the Church–Turing thesis.

Several novel techniques for learning natural classes (e.g., decision trees and DNFs) on average have developed in PAC learning model [JS05; Sel08; Sel09; JLSW11; AC15] and in the agnostic learning model with membership queries [KKMS08; GKK08; KK09; KLW10]. These results are unconditional but work on only some specific distributions, particularly in many cases, the uniform distribution over examples and target functions in the class.

A recent line of research [LP20; LP21c; LP21b; RS21; ACMTV21; IRS22; LP22] characterized the existence of OWF by the average-case meta-complexity, i.e., the average-case hardness of computing the minimum description length of a given string. Our results indicated that, through the existence of OWF, the feasibility of various learning problems (as in Theorem 2.4) is also characterized by the average-case meta-complexity.

Recently, Hopkins, Kane, Lovett, and Mahajan [HKLM22] established a robust theory of learning in statistical settings, which includes the equivalence between PAC learning and agnostic learning (where they considered the worst-case setting with respect to distributions as in the original models). Their idea relies on unbounded computational resources and appears not to hold when we consider polynomial-time learners.

Organization of this Paper

The remainder of this paper is organized as follows. In Section 6, we introduce some additional basic notions for our formal arguments. In Section 7, we present the full proof of the proposition in [IL90] for universal estimation of probability, whose formal proof was not found in previous work. We also discuss the relationships between [IL90] and meta-complexity in Section 7. In Section 8, we present our formulation of universal extrapolation and present the formal proof. In Section 9, we introduce the general framework for translating universal extrapolation into learning algorithms and present the formal statement and the proof of Theorem 3.3. Note that Theorem 2.1 is also proved in Section 8 as an application. In Section 10, we introduce the framework for translating universal extrapolation into learning algorithms for minimizing the expected loss and present the formal statement and the proof of Theorem 3.4. Note that Theorem 2.3 is also proved in Section 10 as an application.

6 Preliminaries

All logarithms are base 2 unless specified otherwise. We use ε to represent an empty symbol. We distinguish ε from ϵ and often use ϵ for an accuracy parameter.

Let $\langle , \rangle$ be a (standard) paring function that maps $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$. For each $k \in \mathbb{N}$ and $n_1, \dots, n_k \in \mathbb{N}$, we use the notation $\langle n_1, \dots, n_k \rangle$ to represent $\langle n_1, \langle n_2, \langle \cdots, \langle n_{k-1}, n_k \rangle \rangle \rangle \rangle \in \mathbb{N}$. For each $k \in \mathbb{N}$ and $x_1, \dots, x_k \in \{0,1\}^*$, we abuse the notation $\langle x_1, \dots, x_k \rangle$ to represent the (standard) binary encoding for the k-tuple $(x_1, \dots, x_k)$. For every $k, k' \in \mathbb{N}$ with $k \le k'$, let $[k : k']$ denote a set $\{k, k+1, \dots, k'\}$. For each $k \in \mathbb{N}$, let $[k] := [1 : k] = \{1, \dots, k\}$.

For every $x, y \in \{0,1\}^*$, let $x \circ y$ denote the concatenation of x and y. For readability, we may omit $\circ$ from $x \circ y$. For each $x \in \{0,1\}^n$ and each $i \in [n]$, we let x_i denote the i-th bit of x. For every $x \in \{0,1\}^n$ and every $S = \{i_1, \dots, i_k\} \subseteq [n]$ (where $i_1 < \cdots < i_k$), we let x_S denote $x_{i_1} \circ \cdots \circ x_{i_k}$; in particular, $x_{[k]} = x_1 \circ \cdots \circ x_k$ and $x_{[k:k']} = x_k \circ \cdots \circ x_{k'}$ for each $k \le k' \le n$. For convenience, we define $x_{[k:k']}$ for any $k, k' \in \mathbb{N}$ with $k \le k'$ as $x_{[k:k'] \cap [|x|]}$; e.g., $01101_{[3:7]} = 01101_{[3:5]} = 101$ and $011_{[10:20]} = \varepsilon$.

For every function $f \colon \mathcal{X} \to \mathcal{Y}$ and every $y \in \mathrm{Im} f := \{f(x) : x \in \mathcal{X}\}$, let $f^{-1}(y) := \{x \in \mathcal{X} : f(x) = y\}$.

For each $n \in \mathbb{N}$, we let U_n denote the uniform distribution over $\{0,1\}^n$ or a random variable selected uniformly at random from $\{0,1\}^n$ in context. For any distribution $\mathcal{D}$, we use the notation $x \sim \mathcal{D}$ to refer to the sampling of x according to D. For any finite set S, we use the notation $x \sim S$ to refer to the uniform sampling of x from S. For each distribution $\mathcal{D}$ and each $x \in \{0,1\}^*$, let $\mathcal{D}(x) = \Pr_{y \sim D}[y = x]$. Furthermore, we let $\mathcal{D}^*(x)$ denote the probability that a string whose prefix matches x is selected according to $\mathcal{D}$, i.e., $\mathcal{D}^*(x) = \Pr_{y \sim \mathcal{D}}[y \text{ begins with } x]$. For simplicity, we may identify a distribution $\mathcal{D}$ with a random variable drawn from $\mathcal{D}$.

For each $t \colon \mathbb{N} \to \mathbb{N}$, we say that a family $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$ of distributions (on binary strings) is $t(n)$-time samplable if there exists a $t(n)$-time deterministic algorithm D (called a sampling algorithm or a sampler for $\mathcal{D}$) such that, for each $n \in \mathbb{N}$, the distribution of $D(1^n, U_{t(n)})$ is statistically identical to $\mathcal{D}_n$. We say that a family $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$ of distributions is (polynomial-time) samplable if $\mathcal{D}$ is $p(n)$-samplable for some polynomial $p(n)$. Let PSamp denote a set of all samplable distributions. For each $t \colon \mathbb{N} \to \mathbb{N}$, we say that a family $\mathcal{D} = \{\mathcal{D}_z\}_{z \in \{0,1\}^*}$ of distributions (on binary strings) is $t(|z|)$-time samplable if there exists a deterministic algorithm D such that, for each $z \in \{0,1\}^*$, the distribution of $D(z, U_{t(|z|)})$ is statistically identical to $\mathcal{D}_z$, and $D(z, \text{-})$ halts in time $t(|z|)$. For every $t(n)$-time samplable distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$, we use the notation $d(\mathcal{D})$ to refer to the description length of the sampler for $\mathcal{D}$ (with respect to the universal Turing machine U); i.e., there exists a $t(n)$-time algorithm D of description length $d(\mathcal{D})$ such that each $\mathcal{D}_n$ is statistically identical to $D(1^n, r)$ for $r \sim \{0,1\}^{t(n)}$. We use the same notation $d(\mathcal{D})$ for samplable distributions $\mathcal{D} = \{\mathcal{D}_z\}_{z \in \{0,1\}^*}$ indexed by z.

For every randomized algorithm A using $s(n)$ random bits on an n-bit input, we use the notation $A(x; r)$ to refer to the execution of $A(x)$ with a random tape r for each $x \in \{0,1\}^n$ and $r \in \{0,1\}^{s(n)}$. In this paper, we assume basic knowledge of probability theory, including the union bound, Markov's inequality, Jensen's inequality, and Hoeffding's inequality. In addition, we employ the following useful lemma.

Fact 6.1 (Pinsker's inequality)**.** *For any distributions $\mathcal{X}$ and $\mathcal{Y}$,*

$$\mathsf{L}_1(\mathcal{X}, \mathcal{Y}) \le \sqrt{\frac{1}{2} \mathrm{KL}(\mathcal{X} || \mathcal{Y})}.$$

We introduce conditional KL divergence and the chain rule for KL divergence.

Definition 6.2 (Conditional KL divergence)**.** *For random variables* $(\mathcal{X}, \mathcal{X}')$ *and* $(\mathcal{Y}, \mathcal{Y}')$*, the conditional KL divergence from* $\mathcal{X}'|\mathcal{X}$ *to* $\mathcal{Y}'|\mathcal{Y}$ *is defined as*

$$\mathrm{KL}((\mathcal{X}'|\mathcal{X})||(\mathcal{Y}'|\mathcal{Y})) = \mathop{\mathbb{E}}_{(x,x')\sim(\mathcal{X},\mathcal{X}')}\left[\log\frac{\Pr[\mathcal{X}'=x'|\mathcal{X}=x]}{\Pr[\mathcal{Y}'=x'|\mathcal{Y}=x]}\right].$$

Lemma 6.3 (Chain rule for KL divergence [cf. CT06, Theorem 2.5.3])**.** *For any random variables* $(\mathcal{X}, \mathcal{X}')$ *and* $(\mathcal{Y}, \mathcal{Y}')$*, it holds that*

$$\mathrm{KL}(\mathcal{X}, \mathcal{X}'||\mathcal{Y}, \mathcal{Y}') = \mathrm{KL}(\mathcal{X}||\mathcal{Y}) + \mathrm{KL}((\mathcal{X}'|\mathcal{X})||(\mathcal{Y}'|\mathcal{Y})).$$

In particular, for any $m \in \mathbb{N}$ *and any random variables* $(\mathcal{X}^1, \dots, \mathcal{X}^m)$ *and* $(\mathcal{Y}^1, \dots, \mathcal{Y}^m)$*,*

$$\mathrm{KL}(\mathcal{X}^1, \dots, \mathcal{X}^m||\mathcal{Y}^1, \dots, \mathcal{Y}^m) = \sum_{i=1}^{m}\mathrm{KL}((\mathcal{X}^i|\mathcal{X}^1, \dots, \mathcal{X}^{i-1})||(\mathcal{Y}^i|\mathcal{Y}^1, \dots, \mathcal{Y}^{i-1})).$$

We also employ the following useful fact.

Fact 6.4. *For any distributions* $\mathcal{X}$ *and* $\mathcal{Y}$ *on a discrete domain* D *and every randomized function* f *of domain* D*, if* $\mathrm{KL}(\mathcal{X}||\mathcal{Y}) < \infty$*, then* $\mathrm{KL}(f(\mathcal{X})||f(\mathcal{Y})) \le \mathrm{KL}(\mathcal{X}||\mathcal{Y})$*.*

Proof. In cases of deterministic f, we verify $\mathrm{KL}(\mathcal{X}||\mathcal{Y}) - \mathrm{KL}(f(\mathcal{X})||f(\mathcal{Y})) \ge 0$ as follows:

$$\begin{aligned}
\mathrm{KL}(\mathcal{X}||\mathcal{Y}) - \mathrm{KL}(f(\mathcal{X})||f(\mathcal{Y})) &= \sum_{x\in D}\mathcal{X}(x)\log\frac{\mathcal{X}(x)}{\mathcal{Y}(x)} - \sum_{a\in\mathrm{Im}f}\Pr[f(\mathcal{X})=a]\log\frac{\Pr[f(\mathcal{X})=a]}{\Pr[f(\mathcal{Y})=a]} \\
&= \sum_{x\in D}\mathcal{X}(x)\log\frac{\mathcal{X}(x)}{\mathcal{Y}(x)} - \sum_{a\in\mathrm{Im}f}\sum_{x\in D}\mathcal{X}(x)\mathbb{1}\{f(x)=a\}\log\frac{\Pr[f(\mathcal{X})=a]}{\Pr[f(\mathcal{Y})=a]} \\
&= \sum_{x\in D}\mathcal{X}(x)\log\frac{\mathcal{X}(x)}{\mathcal{Y}(x)} - \sum_{x\in D}\mathcal{X}(x)\sum_{a\in\mathrm{Im}f}\mathbb{1}\{f(x)=a\}\log\frac{\Pr[f(\mathcal{X})=a]}{\Pr[f(\mathcal{Y})=a]} \\
&= \sum_{x\in D}\mathcal{X}(x)\left(\log\frac{\mathcal{X}(x)}{\mathcal{Y}(x)} - \log\frac{\Pr[f(\mathcal{X})=f(x)]}{\Pr[f(\mathcal{Y})=f(x)]}\right) \\
&= \mathop{\mathbb{E}}_{x\sim\mathcal{X}}\left[\log\frac{\Pr_{\mathcal{X}}[\mathcal{X}=x|f(\mathcal{X})=f(x)]}{\Pr_{\mathcal{Y}}[\mathcal{Y}=x|f(\mathcal{Y})=f(x)]}\right].
\end{aligned}$$

For each $x \in D$, let $\mathcal{X}'_{f(x)}$ (resp. $\mathcal{Y}'_{f(x)}$) be the conditional distribution of $\mathcal{X}$ (resp. $\mathcal{Y}$) given the event that $f(\mathcal{X}) = f(x)$ (resp. $f(\mathcal{Y}) = f(x)$). By regarding that a sampling $x \sim \mathcal{X}$ is performed as $x' \sim \mathcal{X}$ and $x \sim \mathcal{X}'_{f(x')}$, we have

$$\begin{aligned}
\mathrm{KL}(\mathcal{X}||\mathcal{Y}) - \mathrm{KL}(f(\mathcal{X})||f(\mathcal{Y})) &= \mathop{\mathbb{E}}_{x'\sim\mathcal{X}}\left[\mathop{\mathbb{E}}_{x\sim\mathcal{X}'_{f(x')}}\left[\log\frac{\Pr_{\mathcal{X}}[\mathcal{X}=x|f(\mathcal{X})=f(x)]}{\Pr_{\mathcal{Y}}[\mathcal{Y}=x|f(\mathcal{Y})=f(x)]}\right]\right] \\
&= \mathop{\mathbb{E}}_{x'\sim\mathcal{X}}\left[\mathrm{KL}(\mathcal{X}'_{f(x')}||\mathcal{Y}'_{f(x')})\right] \\
&\ge 0,
\end{aligned}$$

where the inequality follows from the non-negativity of the KL divergence.

To extend the above to the cases of randomized f, we apply the chain rule for KL divergence (Lemma 6.3). Let $\mathcal{R}$ be a distribution over randomness for f. Then,

$$\mathrm{KL}(f(\mathcal{X};\mathcal{R})||f(\mathcal{Y};\mathcal{R})) = \mathrm{KL}(\mathcal{R}||\mathcal{R}) + \mathop{\mathbb{E}}_{r\sim\mathcal{R}}[\mathrm{KL}(f(\mathcal{X};r)||f(\mathcal{Y};r))] \leq \mathop{\mathbb{E}}_{r\sim\mathcal{R}}[\mathrm{KL}(\mathcal{X}||\mathcal{Y})] = \mathrm{KL}(\mathcal{X}||\mathcal{Y}),$$

where the inequality follows from the deterministic cases since each $f(;r)$ can be regarded as a deterministic function. □

We introduce an important notion of Kolmogorov complexity.

Definition 6.5 (Kolmogorov complexity)**.** *For every $t \in \mathbb{N}$ and every $x, z \in \{0,1\}^*$, we define the t-time-bounded Kolmogorov complexity of x given z as $\mathrm{K}^t(x|z) = \min_{p\in\{0,1\}^*}\{|p| : U^t(p,z) = x\}$. We also define the (time-unbounded) Kolmogorov complexity of x given z as $\mathrm{K}(x|z) = \lim_{t\to\infty} \mathrm{K}^t(x|z)$. We omit the description "$|z$" if z is the empty string, i.e., $\mathrm{K}^t(x) = \min_{p\in\{0,1\}^*}\{|p| : U^t(p) = x\}$.*

6.1 Infinitely-Often One-Way Function

We present the formal definition of infinitely-often one-way functions.

Definition 6.6 (Infinitely-often one-way function)**.** *A polynomial-time-computable family $f = \{f_n \colon \{0,1\}^{s(n)} \to \{0,1\}^{t(n)}\}_{n\in\mathbb{N}}$ of functions is said to be an* infinitely-often one-way function *if for every randomized polynomial-time algorithm A and every polynomial p, for infinitely many $n \in \mathbb{N}$,*

$$\Pr_{x\sim\{0,1\}^{s(n)},A}[A(f_n(x), 1^n) \in f_n^{-1}(f_n(x))] \leq 1/p(n),$$

where the probability is taken over the internal randomness of A and $x \sim \{0,1\}^{s(n)}$.

We also introduce infinitely-often pseudorandom functions.

Definition 6.7 (Infinitely-often pseudorandom function)**.** *A polynomial-time-computable family $f = \{f_n \colon \{0,1\}^{s(n)} \times \{0,1\}^n \to \{0,1\}^n\}_{n\in\mathbb{N}}$ is said to be an infinitely-often pseudorandom function if for every randomized polynomial-time oracle machine $A^?$ and every polynomial p, for infinitely many $n \in \mathbb{N}$,*

$$\left|\Pr_{A,r\sim\{0,1\}^{s(n)}}\left[A^{f_n(r,\cdot)}(1^n) = 1\right] - \Pr_{A,\phi_n\sim F_n}\left[A^{\phi_n(\cdot)}(1^n) = 1\right]\right| < 1/p(n),$$

where F_n is a set of all functions that maps n-bit strings to n-bit strings.

In words, an infinitely-often pseudorandom function is an efficiently computable function indistinguishable from uniformly selected random functions for efficient adversaries as long as the random seed r is hidden.

The following is a well-known result in cryptography.

Theorem 6.8 ([GGM86; HILL99])**.** *An infinitely-often one-way function exists if and only if an infinitely-often pseudorandom function exists.*

Note that our results also hold for OWF with sufficiently large security (i.e., the security holds for any sufficiently large seed length) by considering the learnability on infinitely many sample sizes n with arbitrarily small parameters fixed beforehand as polynomial-time computable functions in n. This follows from the observation that, for each sample size n, the number of relevant security parameters is at most $\mathsf{poly}(n)$ in our reductions, and all of them are efficiently computable and bounded by a polynomial in n and the parameters (when the parameters are efficiently computable from n) and owing to the standard combining trick (cf. [NR06, Lemma 4.1]).

6.2 Domination Property

We introduce the important *domination* property of the time-bounded universal distribution Q^t.

Lemma 6.9 (domination)**.** *For every distribution $\mathcal{D}$ that has a t_D-time sampler of description length d, there exists $t_0 \in \mathbb{N}$ such that for every $x \in \{0,1\}^*$ and for every $t \in \mathbb{N}$ with $t \geq t_0$, it holds that $\mathrm{Q}^t(x) \geq \mathcal{D}(x)/2^{O(d)}$. Furthermore, $t_0 = \tau_{\mathrm{dom}}(d, t_D)$ for a universal polynomial τ_{dom} (that depends on U).*

Proof. Let $M \in \{0,1\}^d$ be the binary encoding of the sampler for $\mathcal{D}$. If τ_{dom} is sufficiently large (depending only on U), then any $t \geq \tau_{\mathrm{dom}}(d, t_D)$ is sufficiently large that (i) U^t simulates M and (ii) $\langle M, x\rangle$ (where x is a seed for M) is encoded by t bits. Therefore, if the prefix of random seed r to $U^t(r)$ corresponds to $\langle M, \text{-}\rangle$, then the conditional distribution of Q^t is statistically equivalent to $\mathcal{D}$. The lemma holds because the condition is satisfied with probability at least $2^{-O(d)}$. □

It is not hard to verify that if a distribution $\mathcal{X}$ is dominated by another distribution $\tilde{\mathcal{X}}$, then $\tilde{\mathcal{X}}$ approximates $\mathcal{X}$ with a constant KL divergence.

Proposition 6.10. *Let $\mathcal{X}$ and $\tilde{\mathcal{X}}$ be distributions over a discrete set. If there exists an absolute constant $d > 0$ such that $\tilde{\mathcal{X}}(x) \geq \mathcal{X}(x)/2^d$ for all $x \in \mathrm{supp}(\mathcal{X})$, then $\mathrm{KL}(\mathcal{X}||\tilde{\mathcal{X}}) \leq d$.*

Proof. The proposition is verified as follows:

$$\mathrm{KL}(\mathcal{X}||\tilde{\mathcal{X}}) = \mathop{\mathbb{E}}_{x\sim\mathcal{X}}\left[\log\frac{\mathcal{X}(x)}{\tilde{\mathcal{X}}(x)}\right] \leq \mathop{\mathbb{E}}_{x\sim\mathcal{X}}\left[\log 2^d\right] = d.$$

□

6.3 Basic Properties of q^t and Computational Depth

Recall that for every $t \in \mathbb{N}$ and every string $x \in \{0,1\}^*$,

$$\mathrm{q}^t(x) := -\log \mathrm{Q}^t(x) = -\log \Pr_{d\sim\{0,1\}^t}\left[U^t(d) = x\right].$$

In the resource-unbounded case, it is known that $\lim_{t\to\infty} \mathrm{q}^t(x)$ is equal to the Kolmogorov complexity of x up to an additive constant [LV19].

We observe the upper bound on q^t, which follows from the domination property.

Proposition 6.11 (Implicit in [IL90])**.** *There exists a polynomial τ such that for every $t_D(n)$-time samplable distribution $\mathcal{D}$, where $t_D(n) \geq n$, every $n \in \mathbb{N}$, every $t \geq \tau(d(\mathcal{D}), t_D(n))$, and every $x \in \mathrm{supp}(\mathcal{D}_n)$,*

$$\mathrm{q}^t(x) \leq -\log \mathcal{D}_n(x) + \log t + O(\mathrm{K}^t(\mathcal{D})) \leq -\log \mathcal{D}_n(x) + \log t + O(d(\mathcal{D})),$$

where $\mathrm{K}^t(\mathcal{D})$ represents the t-time-bounded Kolmogorov complexity of the sampler of $\mathcal{D}$.

Proof. Let S be a randomized polynomial-time algorithm that, on input 1^n, outputs a string distributed according to $\mathcal{D}_n$. Since $S(1^n;\text{-})$ is described by $d_n := O(d(\mathcal{D})+\log n)$ bits, by Lemma 6.9, for every $n \in \mathbb{N}$, every $t \geq \tau_{\mathsf{dom}}(d_n, t_D(n))$, and every $x \in \operatorname{supp}(\mathcal{D}_n)$,

$$2^{-\mathrm{q}^t(x)} = \mathrm{Q}^t(x) \geq 2^{-C\log n} \cdot 2^{-C\cdot d(\mathcal{D})} \cdot \mathcal{D}_n(x),$$

for some absolute constant $C > 0$. By selecting a polynomial τ as

$$\tau(d(\mathcal{D}), t_D(n)) = \max\{\tau_{\mathsf{dom}}(d_n, t_D(n)), t_D(n)^C\},$$

we have that for every $t \geq \tau(d(\mathcal{D}), t_D(n))$,

$$\begin{aligned}\mathrm{q}^t(x) &\leq -\log \mathcal{D}_n(x) + C \cdot d(\mathcal{D}) + C\log n \\ &\leq -\log \mathcal{D}_n(x) + C \cdot d(\mathcal{D}) + C\log t_D(n) \\ &\leq -\log \mathcal{D}_n(x) + C \cdot d(\mathcal{D}) + \log t.\end{aligned}$$

□

We also note that $\mathrm{K}(x)$ is a lower bound for $\mathrm{q}^t(x)$.

Fact 6.12 (e.g., in [LV19, Chapter 4])**.** *For every $x \in \{0,1\}^*$ and every $t \in \mathbb{N}$,*

$$\mathrm{K}(x) \leq \mathrm{q}^t(x) + O(\log t).$$

Recall that for every $t \in \mathbb{N}$ and $x \in \{0,1\}^*$, the t-time-bounded computational depth $\mathrm{cd}^t(x)$ of x is defined as

$$\mathrm{cd}^t(x) := \mathrm{q}^t(x) - \mathrm{K}(x).$$

We use the following theorem.

Theorem 6.13 (Slow growth low [Ben88; AFPS12; Hir23])**.** *There exist polynomials τ, τ' such that for every $t_f(n)$-time computable function $f\colon \{0,1\}^* \to \{0,1\}^*$, every $x \in \{0,1\}^*$, and every $t \geq \tau'(|x|, |f|)$,*

$$\mathrm{cd}^{\tau(t, t_f(|x|))}(f(x)) \leq \mathrm{cd}^t(x) + \log t + O(|f|),$$

where $|f|$ represents the description size of f (as a Turing machine with respect to $U^{t_f(n)}$).

A worst-case algorithm that works in exponential time in the computational depth of input also works on average under every samplable distributions because of the following fact that no efficient algorithm can produce strings with high computational depth for most strings x produced by efficient algorithms [AFMV06; AF09; Hir21].

Lemma 6.14. *There exists a polynomial τ such that for every $t_D(n)$-time samplable distribution $\mathcal{D}$, where $t_D(n) \geq n$, every $n \in \mathbb{N}$, every $t \geq \tau(d(\mathcal{D}), t_D(n))$, and every $k \in \mathbb{N}$,*

$$\Pr_{x \sim \mathcal{D}_n}\left[\mathrm{cd}^t(x) > k\right] = \Pr_{x \sim \mathcal{D}_n}\left[\mathrm{q}^t(x) - \mathrm{K}(x) > k\right] \leq 2^{-k+\log t + O(d(\mathcal{D}))}.$$

Particularly, by letting $k = \log \delta^{-1} + 2\log t$ for $\delta^{-1} \in \mathbb{N}$, we have for all $n \geq 2^{O(d(\mathcal{D}))}$ and all $t \geq \tau(d(\mathcal{D}), t_D(n))$,

$$\Pr_{x \sim \mathcal{D}_n}\left[\mathrm{cd}^t(x) \leq \log \delta^{-1} + 2\log t\right] \geq 1 - \delta.$$

Proof. Let τ be the polynomial in Proposition 6.11. Without loss of generality, we assume that $\tau(d(\mathcal{D}), t_D(n)) \geq 2t_D(n)$.

By Proposition 6.11, for every $n, t \in \mathbb{N}$ with $t^{1/2} \geq \tau(d(\mathcal{D}), t_D(n))$ and every $x \in \mathrm{supp}(\mathcal{D}_n)$,

$$2^{\mathrm{q}^t(x) - 2^{-1}\log t - O(d(\mathcal{D}))} \leq 2^{\mathrm{q}^{t^{1/2}}(x) - 2^{-1}\log t - O(d(\mathcal{D}))} \leq \frac{1}{\mathcal{D}_n(x)}.$$

Since $\mathcal{D}_n$ is sampled in time $t_D(n)$, we can assume that $|x| \leq t_D(n)$ for every $x \in \mathrm{supp}(\mathcal{D}_n)$.

For every $n, t \in \mathbb{N}$ with $t^{1/2} \geq \tau(d(\mathcal{D}), t_D(n))$ $(\geq 2t_D(n))$,

$$\begin{aligned}
\mathop{\mathbb{E}}_{x\sim\mathcal{D}_n}\left[2^{-\mathrm{K}(x)+\mathrm{q}^t(x)-2^{-1}\log t - O(d(\mathcal{D}))}\right] \leq \mathop{\mathbb{E}}_{x\sim\mathcal{D}_n}\left[\frac{2^{-\mathrm{K}(x)}}{\mathcal{D}_n(x)}\right] &= \sum_{x\in\mathrm{supp}(\mathcal{D}_n)} \mathcal{D}_n(x)\frac{2^{-\mathrm{K}(x)}}{\mathcal{D}_n(x)} \\
&= \sum_{x\in\mathrm{supp}(\mathcal{D}_n)} 2^{-\mathrm{K}(x)} \\
&\leq \sum_{i=1}^{2t_D(n)} \sum_{\substack{x\in\{0,1\}^*:\\ |x|\leq t_D(n), \mathrm{K}(x)=i}} 2^{-i} \\
&\leq \sum_{i=1}^{2t_D(n)} 2^i \cdot 2^{-i} \\
&= 2t_D(n) \leq t^{1/2}.
\end{aligned}$$

This implies the lemma as follows:

$$\begin{aligned}
\Pr_{x\sim\mathcal{D}_n}[\mathrm{q}^t(x) - \mathrm{K}(x) > k] &= \Pr_{x\sim\mathcal{D}_n}[2^{\mathrm{q}^t(x)-\mathrm{K}(x)-2^{-1}\log t - O(d(\mathcal{D}))} > 2^{k-2^{-1}\log t - O(d(\mathcal{D}))}] \\
&\leq \frac{\mathbb{E}_{x\sim\mathcal{D}_n}\left[2^{-\mathrm{K}(x)+\mathrm{q}^{t(n)}(x)-2^{-1}\log t(n) - O(d(\mathcal{D}))}\right]}{2^{k-2^{-1}\log t - O(d(\mathcal{D}))}} \\
&\leq 2^{2^{-1}\log t} \cdot 2^{-k+2^{-1}\log t + O(d(\mathcal{D}))} \\
&= 2^{-k+\log t + O(d(\mathcal{D}))},
\end{aligned}$$

where the first inequality follows from Markov's inequality. The lemma is obtained by regarding τ^2 as the polynomial τ in the statement. □

We also use the following lemma that shows the time-bounded computational depth of a sample drawn from a samplable distribution $\mathcal{D} = \{\mathcal{D}_z\}_{z\in\{0,1\}^*}$ is not much different from the time-bounded computational depth of the nonuniform advice z. Particularly when z represents computationally shallow secret information used for generating samples, the sample set is also computationally shallow with high probability.

Lemma 6.15. *There exist polynomials τ, τ' such that for every $t_D(|z|)$-samplable distribution $\mathcal{D} = \{\mathcal{D}_z\}_{z\in\{0,1\}^*}$, where $t_D(|z|) \geq |z|$, every $z \in \{0,1\}^*$, every $i \in \mathbb{N}$, and every $t \geq \tau'(d(\mathcal{D}), t_D(|z|))$,*

$$\Pr_{x\sim\mathcal{D}_z}[\mathrm{cd}^{\tau(t)}(x_{[i]}) \leq \mathrm{cd}^t(z) + k] \geq 1 - 2^{-k+\log t + O(d(\mathcal{D}))}.$$

Particularly, by letting $k = \log \delta^{-1} + 2\log t$ for $\delta^{-1} \in \mathbb{N}$, we have for all $|z| \geq 2^{O(d(\mathcal{D}))}$, all $i \in \mathbb{N}$, and all $t \geq \tau'(d(\mathcal{D}), t_D(|z|))$,

$$\Pr_{x \sim \mathcal{D}_z}[\mathrm{cd}^{\tau(t)}(x_{[i]}) \leq \mathrm{cd}^t(z) + \log \delta^{-1} + 2\log t] \geq 1 - \delta.$$

Proof. Let D be the $t_D(|z|)$-time sampler for $\mathcal{D}$ described by $d(\mathcal{D})$ bits; i.e., each $\mathcal{D}_z$ is statistically equivalent to $D(z;r)$ for $r \sim \{0,1\}^{t_D(|z|)}$.

By Theorem 6.13, there exist universal polynomials τ, τ' such that for every $t \geq \tau'(d(\mathcal{D}), t_D(|z|))$ (recall that $t_D(|z|) \geq |z|$) and every $i \leq t_D(|z|)$,

$$\mathrm{cd}^{\tau(t)}(D(z;r)_{[i]}) \leq \mathrm{cd}^{2t}(z,r) + 2^{-1}\log t + O(d(\mathcal{D})) \tag{2}$$

We will prove that there exists a universal polynomial τ'' such that for every $t \geq \tau''(t_D(|z|))$, every $k' \in \mathbb{N}$ and every $i \leq t_D(|z|)$,

$$\Pr_{r \sim \{0,1\}^{t_D(|z|)}}[\mathrm{cd}^{2t}(z,r) \leq \mathrm{cd}^t(z) + k'] \geq 1 - 2^{-k' + 2^{-1}\log t}. \tag{3}$$

Note that Eq. (3) trivially holds for $k' \leq 0$ because $2^{-k'+2^{-1}\log t} \geq 1$ in the case. For any $k \in \mathbb{N}$, by taking $k' = k - 2^{-1}\log t - O(d(\mathcal{D}))$,

$$\Pr_{r \sim \{0,1\}^{t_D(|z|)}}[\mathrm{cd}^{2t}(z,r) + 2^{-1}\log t + O(d(\mathcal{D})) \leq \mathrm{cd}^t(z) + k] \geq 1 - 2^{-k+\log t + O(d(\mathcal{D}))}. \tag{4}$$

Eq. (2) and Eq. (4) imply that for every $t \geq \max\{\tau'(d(\mathcal{D}), t_D(|z|)), \tau''(t_D(|z|))\}$, every $i \in \mathbb{N}$, and every $k \in \mathbb{N}$,

$$\begin{aligned}&\Pr_{x \sim \mathcal{D}_z}[\mathrm{cd}^{\tau(t)}(x_{[i]}) \leq \mathrm{cd}^t(z) + k] \\ &\quad = \Pr_{r \sim \{0,1\}^{t_D(|z|)}}[\mathrm{cd}^{\tau(t)}(D(z;r)_{[i]}) \leq \mathrm{cd}^t(z) + k] \geq 1 - 2^{-k+\log t + O(d(\mathcal{D}))},\end{aligned}$$

where we use the fact that $x_{[i]} = x$ in the case of $i > t_D(|z|) \geq |x|$.

In the remainder we prove Eq. (3). By selecting sufficiently large polynomial τ''', we have that for every $z, r \in \{0,1\}^*$, and $t \geq \tau'''(|z|, |r|)$

$$\mathrm{Q}^{2t}(z,r) \geq \mathrm{Q}^t(z) 2^{-|r|-C}$$

for some absolute constant C because there exists a $2t$-time $(|r| + O(1))$-size program that prints (r', r) for $r' \sim \mathrm{Q}^t$ and embedded r. Therefore, we have

$$\mathrm{q}^{2t}(z,r) \leq \mathrm{q}^t(z) + |r| + C.$$

By contrast, we apply the Symmetry of Information [ZL70] to obtain that

$$\mathrm{K}(z,r) \geq \mathrm{K}(z) + \mathrm{K}(r|z) - C' \log |z||r|$$

for some absolute constant C'.

Let $\tau''(t_D(|z|)) = \max\{\tau'''(t_D(|z|), t_D(|z|)), 4 \cdot 2^{2C} t_D(|z|)^{4C'}\}$. Then, from the two inequalities above, for any $z \in \{0,1\}^*$, any $r \in \{0,1\}^{t_D(|z|)}$, and any $t \geq \tau''(t_D(|z|))$,

$$\begin{aligned}\mathrm{cd}^{2t}(z,r) &= \mathrm{q}^{2t}(z,r) - \mathrm{K}(z,r) \\ &\leq \mathrm{q}^t(z) + |r| + C - (\mathrm{K}(z) + \mathrm{K}(r|z) - C' \log |z||r|) \\ &\leq \mathrm{cd}^t(z) + |r| + 2^{-1} \log t - 1 - \mathrm{K}(r|z),\end{aligned}$$

where the last inequality follows from

$$2^{-1}\log t \geq 2^{-1} \log \tau''(t_D(|z|)) \geq 2^{-1} \log(4 \cdot 2^{2C} t_D(|z|)^{4C'}) = C + C' \log t_D(|z|)^2 + 1 \geq C + C' \log |z||r| + 1.$$

Therefore, for every $k' \in \mathbb{N}$,

$$\begin{aligned}\Pr_{r \sim \{0,1\}^{t_D(|z|)}}[\mathrm{cd}^{2t}(z,r) \leq \mathrm{cd}^t(z) + k'] &\geq \Pr_{r \sim \{0,1\}^{t_D(|z|)}}[\mathrm{K}(r|z) \geq |r| - k' + 2^{-1}\log t - 1] \\ &\geq 1 - \frac{\sum_{i=0}^{|r|-k'+2^{-1}\log t - 1} 2^i}{2^{|r|}} \geq 1 - 2^{-k' + 2^{-1}\log t}.\end{aligned}$$

□

7 Estimating Universal Probability and Kolmogorov Complexity

In this section, we formally prove the following theorem. Note that Theorem 2.10 corresponds to Item 1 ⇔ Item 3.

Theorem 7.1. *The following are equivalent.*

1. *There exists no infinitely-often one-way function.*
2. *There exists a randomized polynomial-time algorithm M such that for all $t, \alpha, \delta^{-1} \in \mathbb{N}$ and all $x \in \{0,1\}^*$ with $\mathrm{cd}^t(x) \leq \alpha$,*

$$\Pr_M\left[\mathrm{Q}^t(x) \cdot (1-\delta) \leq M(x, 1^{\langle t, \delta^{-1}, 2^\alpha \rangle}) \leq \mathrm{Q}^t(x) \cdot (1+\delta)\right] \geq 1 - \delta.$$

3. *There exists a randomized polynomial-time algorithm M such that for every $\mathcal{D} \in$ PSAMP, there exists a polynomial t_0 such that for all large $n \in \mathbb{N}$, for every integer $t \geq t_0(n)$, for every $\delta^{-1} \in \mathbb{N}$,*

$$\Pr_{M, x \sim \mathcal{D}_n}\left[\mathrm{Q}^t(x) \cdot (1-\delta) \leq M(x, 1^{\langle t, \delta^{-1} \rangle}) \leq \mathrm{Q}^t(x) \cdot (1+\delta)\right] \geq 1 - \delta.$$

Furthermore, we show that Theorem 7.1 yields a universal algorithm that approximates the resource-unbounded Kolmogorov complexity of a string chosen from *unknown* samplable distributions. This result implies the recent result of [IRS22] that constructed an efficient algorithm $M_{\mathcal{D}}$ that approximates the Kolmogorov complexity of x drawn from any fixed samplable distribution $\mathcal{D}$. In addition, we show the existence of a promise problem in AM whose *worst-case* hardness characterizes the existence of one-way functions.

Theorem 7.2. *The following are equivalent.*

1. *There exists no infinitely-often one-way function.*
2. *There exists a randomized polynomial-time algorithm M such that, for all $n, \delta^{-1}, t, \alpha \in \mathbb{N}$ and all $x \in \{0,1\}^n$ with $\mathrm{cd}^t(x) \leq \alpha$,*
$$\Pr_M \left[\mathrm{K}(x) - \alpha - O(\log t) \leq M(x, 1^{\langle t, \delta^{-1}, 2^\alpha \rangle}) \leq \mathrm{K}(x) \right] \geq 1 - \delta.$$
3. *There exists a randomized polynomial-time algorithm M such that, for every $\mathcal{D} \in \mathrm{PSAMP}$, there exists a polynomial t_0 such that for all large $n \in \mathbb{N}$, for every integer $t \geq t_0(n)$, for every $\delta^{-1} \in \mathbb{N}$,*
$$\Pr_{\substack{x \sim \mathcal{D}_n \\ M}} \left[\mathrm{K}(x) - \log(1/\delta) - O(\log t) \leq M(x, 1^t, 1^{\delta^{-1}}) \leq \mathrm{K}(x) \right] \geq 1 - \delta.$$
4. *The following promise problem $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}}) \in \mathsf{AM}$ is in prBPP:*
$$\Pi_{\mathrm{YES}} = \{(x, 1^s, 1^t) : \mathrm{K}(x) \leq s \ \wedge \ \mathrm{cd}^t(x) \leq \log t\}$$
$$\Pi_{\mathrm{NO}} = \{(x, 1^s, 1^t) : \mathrm{K}(x) > s + c \log t \ \wedge \ \mathrm{cd}^t(x) \leq \log t\},$$
where $c \geq 1$ is a universal constant.

The proof of Theorem 7.1 relies on another result of [IL89; IL90], which enables us to estimate the probability $\mathcal{D}(x)$ for a string x drawn from a *known* distribution $\mathcal{D}$. We prove this in Section 7.1. In Section 7.2, we apply this to the time-bounded universal distribution, which yields a proof of Theorem 7.1. Finally, we complete a proof of Theorem 7.2 in Section 7.3.

The proof of Theorem 7.1 (1 $\Rightarrow$ 2) also yields the same theorem for $\mathrm{Q}^{t,*}(x)$ instead of $\mathrm{Q}^t(x)$ by replacing the universal Turing machine U^t with the truncated universal Turing machine that outputs the prefix of U^t. Recall that $\mathrm{Q}^{t,*}(x)$ is the probability that the prefix of a sample $y \sim \mathrm{Q}^t$ matches x. We will use the universal approximation for $\mathrm{Q}^{t,*}$ in one of the proofs of the universal extrapolation theorem in Section 8.

Theorem 7.3. *If there is no infinitely-often one-way function, then there exists a randomized polynomial-time algorithm M such that for all $t, \alpha, \delta^{-1} \in \mathbb{N}$ and all $x \in \{0,1\}^*$ with $\mathrm{cd}^t(x) \leq \alpha$,*
$$\Pr_M \left[\mathrm{Q}^{t,*}(x) \cdot (1 - \delta) \leq M(x, 1^{\langle t, \delta^{-1}, 2^\alpha \rangle}) \leq \mathrm{Q}^{t,*}(x) \cdot (1 + \delta) \right] \geq 1 - \delta.$$

7.1 Estimating the Probability with respect to Known Distributions

We first apply a standard hardness amplification technique to obtain an inverter that takes an additional parameter $\delta^{-1} \in \mathbb{N}$ and successfully inverts a one-way function with probability $1 - \delta$.

Proposition 7.4. *If there exists no infinitely-often one-way function, then for every polynomial-time-computable family $f = \left\{ f_n \colon \{0,1\}^{s(n)} \to \{0,1\}^{t(n)} \right\}_{n \in \mathbb{N}}$, there exists a randomized polynomial-time algorithm A such that for every $n \in \mathbb{N}$ and every $\delta^{-1} \in \mathbb{N}$,*
$$\Pr_{x \sim \{0,1\}^{s(n)}, A} [A(f_n(x); 1^n, 1^{\delta^{-1}}) \notin f_n^{-1}(f_n(x))] \leq \delta.$$

Proof Sketch. We define a new family $f' = \left\{f_{\langle n,k\rangle}\right\}_{\langle n,k\rangle \in \mathbb{N}}$ so that

$$f'_{\langle n,k\rangle}(x_1,\dots,x_k) = (f(x_1),\dots,f(x_k))$$

for every $x_1,\dots,x_k \in \{0,1\}^{s(n)}$, where $\langle\text{-},\text{-}\rangle \colon \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ is a bijection. Yao's amplification theorem [Yao82; Gol01] shows that inverting f_n with probability $1-\delta$ reduces to the task of inverting $f_{\langle n,k\rangle}$ for some $k = \mathsf{poly}(n,\delta^{-1})$. $\square$

We now show that there exists an algorithm that approximates $\mathcal{D}_n(x)$ for a string x drawn from $\mathcal{D}_n$.

Lemma 7.5 ([IL89; IL90; Imp92])**.** *Assume that there exists no infinitely-often one-way function. Then, for every $\mathcal{D} \in \mathrm{PSAMP}$, there exists a randomized polynomial-time algorithm A such that for all large $n \in \mathbb{N}$ and all large $\delta^{-1} \in \mathbb{N}$,*

$$\Pr_{\substack{x\sim\mathcal{D}_n \\ A}}\left[(1-\delta)\cdot\mathcal{D}_n(x) \le A(x,1^n,1^{\delta^{-1}}) \le (1+\delta)\cdot\mathcal{D}_n(x)\right] \ge 1-\delta.$$

Although a proof of this result appeared to be known to researchers in the 1990s (e.g., [OW93]), we are not aware of any published proof. In fact, recent papers [Nan21b; IRS21] provide a weaker algorithm that approximates $\mathcal{D}_n(x)$ within some constant factor (instead of a $(1+\delta)$-factor for an arbitrary small $\delta > 0$), which was further used in [LP21a]. Although such a weak algorithm was sufficient for most applications in the previous studies, it is insufficient for our purpose of showing the universal extrapolation theorem. To the best of our knowledge, the following is the first written proof of Lemma 7.5. The proof closely follows the proof of the construction of a distributional inverter presented in the PhD thesis of Impagliazzo [Imp92].

Proof of Lemma 7.5. Let f be the sampler of $\mathcal{D}$. That is, $f = \{f_n : \{0,1\}^n \to \{0,1\}^n\}_{n\in\mathbb{N}}$ and the distribution of $f_n(x)$ over $x \sim \{0,1\}^n$ is identical to $\mathcal{D}_n$. (Without loss of generality, we assume that the input length of f_n and the output length are identical.)

Let $h_\ell \colon \{0,1\}^n \to \{0,1\}^\ell$ be a pairwise independent hash. Consider the family $g = \left\{g_{\langle n,\ell\rangle}\right\}_{n,\ell\in\mathbb{N}}$ of functions defined as follows: $g_{\langle n,\ell\rangle}(x,h_\ell) := (f_n(x), h_\ell(x), h_\ell)$. Here, we identify h_ℓ with the random bits used to generate h_ℓ. For simplicity, in the following, we omit the subscripts of g because n and ℓ are clear from the context.

Since g is computable in polynomial time, the assumption implies that there exists a randomized polynomial-time algorithm I such that

$$\Pr_{x,h_\ell,I}\left[I(g(x,h_\ell);1^{\langle n,\ell\rangle},1^{\epsilon^{-1}}) \in g^{-1}(g(x,h_\ell))\right] \ge 1-\epsilon. \tag{5}$$

Here, the probability is taken over all x, h_ℓ, and the internal randomness of I. For notational simplicity, we omit I from the subscript of probabilities in the following. We also often omit the auxiliary input $(1^{\langle n,\ell\rangle},1^{\epsilon^{-1}})$ from the input to I.

Using this inverter I, we present the definition of the algorithm A: The algorithm A takes $(y,1^n,1^{\delta^{-1}})$ as input and sets $\epsilon := 1/\mathsf{poly}(n/\delta)$ for some polynomial poly to be chosen later. Then, for each $\ell = n+\log(1/\delta),\dots,0$ in decreasing order, A estimates

$$v_\ell := \Pr_{\substack{r\sim\{0,1\}^\ell \\ h_\ell}}\left[I(y,r,h;1^{\langle n,\ell\rangle},1^{\epsilon^{-1}}) \in g^{-1}(y,r,h)\right]$$

by randomly sampling r and h several times. Let $\widetilde{v_\ell}$ be the estimate of v_ℓ computed by A. If $\widetilde{v_\ell} \le \delta$, the algorithm A continues to the next loop with ℓ reduced by 1. If $\widetilde{v_\ell} > \delta$, the algorithm A outputs $\widetilde{v_\ell} \cdot 2^{\ell-n}$ and halts.

Below, we prove that the output of A approximates $\mathcal{D}_n(y) = 2^{-n} \cdot \left|f_n^{-1}(y)\right|$ with probability at least $1-\delta$.

By Hoeffding's inequality, the algorithm A can compute $\widetilde{v_\ell}$ such that $|\widetilde{v_\ell} - v_\ell| \le \epsilon$ in time $\mathsf{poly}(n/\epsilon)$ with probability at least $1 - 2^{-n}$ over the internal randomness of A. In the following, we may assume that $\widetilde{v_\ell}$ satisfies $|\widetilde{v_\ell} - v_\ell| \le \epsilon$. We may also assume that $\delta \ge 2^{-n}$, as otherwise a brute-force search can be used to compute $\left|f_n^{-1}(y)\right|$ in time $2^{O(n)}$.

Fix an input $y \in \mathrm{image}(f_n)$. Let $X \subseteq \{0,1\}^n$ denote $f_n^{-1}(y)$. Let $\ell \in \mathbb{N}$ be the last integer in the algorithm A on input y.

Claim 7.6. *Assume that A halts with ℓ. Then,*

$$\widetilde{v_\ell} - \epsilon \le |X| \cdot 2^{-\ell}.$$

In particular,

$$\delta - \epsilon \le |X| \cdot 2^{-\ell}.$$

Proof. Since $|\widetilde{v_\ell} - v_\ell| \le \epsilon$, it suffices to show $v_\ell \le |X| \cdot 2^{-\ell}$. Observe that I successfully inverts g on (y, r, h_ℓ) only if (y, r, h_ℓ) is in the image of g, in which case $r = h_\ell(x)$ for some $x \in f^{-1}(y) = X$. Since $r \sim \{0,1\}^\ell$, by a union bound, the probability v_ℓ that I successfully inverts g on (y, r, h_ℓ) is at most $|X| \cdot 2^{-\ell}$. The "in particular" part follows from the fact that $\widetilde{v_\ell} > \delta$ when A halts. $\diamond$

Claim 7.7. *With probability at least $1 - \sqrt{\epsilon} \cdot 2n$ over a choice of $y := f_n(x)$ and $x \sim \{0,1\}^n$, it holds that for every ℓ,*

$$|X| \cdot 2^{-\ell} \cdot \left(1 - |X| \cdot 2^{-\ell}\right) \cdot \left(1 - \sqrt{\epsilon}\right) \le \widetilde{v_\ell} + \epsilon.$$

Proof. For notational simplicity, we omit the subscript ℓ of h_ℓ. Fix $y \in \mathrm{image}(f_n)$. For a hash function h, let S_h denote the set of all the strings $x \in X$ such that $h(x) \ne h(x')$ for every $x' \in X \setminus \{x\}$. Let $h(S_h)$ denote the image of S_h under h. For every $x \in X$, by a union bound, we have $\Pr_h[x \notin S_h] \le |X| \cdot 2^{-\ell}$. In particular, we obtain

$$\mathop{\mathbb{E}}_h[|S_h|] \ge |X| \cdot (1 - |X| \cdot 2^{-\ell}). \tag{6}$$

Let $h(S_h)$ denote the image of S_h under h. Under the event that $r \in h(S_h)$, the random variable (r, h) is identical to the distribution of $(h(x), h)$ over $x \sim S_h$. Thus, we obtain

$$\begin{aligned} v_\ell &= \Pr_{r,h}\left[I(y, r, h; 1^{\langle n,\ell \rangle}, 1^{\epsilon^{-1}}) \in g^{-1}(y, r, h)\right] \\ &\ge \Pr[r \in h(S_h)] \cdot \Pr_{\substack{h \\ x \sim S_h}}\left[I(f(x), h(x), h) \in g^{-1}(f(x), h(x), h)\right]. \end{aligned}$$

For the first factor, we have

$$\Pr[r \in h(S_h)] = 2^{-\ell} \cdot \mathop{\mathbb{E}}_h[|h(S_h)|] \ge 2^{-\ell} \cdot |X| \cdot (1 - |X| \cdot 2^{-\ell}),$$

where the last inequality follows from Eq. (6). The second factor can be bounded from below by

$$\begin{aligned}&\Pr_{\substack{h\\x\sim X}}\left[I(f(x),h(x),h)\in g^{-1}(f(x),h(x),h)\right]\\=&\Pr_{\substack{h\\x\sim\{0,1\}^n}}\left[I(f(x),h(x),h)\in g^{-1}(f(x),h(x),h)\;\middle|\;y=f(x)\right]\end{aligned}$$

because $S_h\subseteq X$.

Finally, we consider the random variable y distributed according to $f_n(x)$ over $x\sim\{0,1\}^n$. By Markov's inequality and Eq. (5), with probability at least $1-\sqrt{\epsilon}$ over $y\sim f_n(U_n)$, it holds that

$$\Pr_{\substack{h\\x\sim\{0,1\}^n}}\left[I(f(x),h(x),h)\in g^{-1}(f(x),h(x),h)\;\middle|\;y=f(x)\right]\geq 1-\sqrt{\epsilon}.$$

By taking a union bound over all $\ell\leq n+\log(1/\delta)\leq 2n$, with probability at least $1-\sqrt{\epsilon}\cdot 2n$ over $y\sim f_n(U_n)$, the same event occurs, in which case we have

$$v\geq 2^{-\ell}\cdot|X|\cdot(1-|X|\cdot 2^{-\ell})\cdot(1-\sqrt{\epsilon}).$$

The claim follows from $v_\ell\leq\widetilde{v_\ell}+\epsilon$. ◇

Let ℓ be the last integer in the algorithm A on input y. Since A did not halt in all the preceding loops, we have $\widetilde{v}_{\ell'}\leq\delta$ for every $\ell'>\ell$. By Claim 7.7, we obtain

$$|X|\cdot 2^{-\ell'}\cdot\left(1-|X|\cdot 2^{-\ell'}\right)\cdot\left(1-\sqrt{\epsilon}\right)\leq\widetilde{v}_{\ell'}+\epsilon\leq\delta+\epsilon.$$

Let $\gamma:=|X|\cdot 2^{-\ell}$. Then, we have

$$2^{-k}\gamma\cdot(1-2^{-k}\gamma)\cdot(1-\sqrt{\epsilon})\leq\delta+\epsilon$$

for every integer $k\geq 1$. In particular, we obtain $\gamma\leq 4\delta$ for sufficiently small δ and ϵ. (Otherwise, we can select k so that $2^{-k}\gamma\in[2\delta,4\delta]$, which is a contradiction.) We also have $\delta-\epsilon<\gamma$ from Claim 7.6. We choose a small ϵ so that $\epsilon\ll\delta^2$. Then, $\sqrt{\epsilon}\ll\delta$ and $\epsilon\ll\gamma\delta$.

By Claim 7.7 and $\gamma\leq 4\delta$, we obtain

$$\gamma\cdot(1-O(\delta))\leq\gamma\cdot(1-4\delta)\cdot(1-\sqrt{\epsilon})-\epsilon\leq\widetilde{v_\ell}.$$

By Claim 7.6, we have

$$\widetilde{v_\ell}\leq\gamma+\epsilon\leq\gamma\cdot(1+\delta).$$

Recall that the output of A is $\widetilde{v_\ell}\cdot 2^{\ell-n}$. We conclude that

$$|X|\cdot 2^{-n}\cdot(1-O(\delta))\leq\widetilde{v_\ell}\cdot 2^{\ell-n}\leq|X|\cdot 2^{-n}\cdot(1+\delta).$$

□

7.2 Universal Approximation of Q^{t} and $\mathrm{Q}^{\mathrm{t},*}$

We now apply Lemma 7.5 to the time-bounded universal distribution and obtain the "first written" proof of the theorem of Impagliazzo and Levin [IL90].

Proof of Theorem 7.1. We only prove Item 1 ⇒ Item 2. Note that Item 2 ⇒ Item 3 follows from Lemma 6.14. Consider the time-bounded universal distribution $\mathcal{M} = \{\mathcal{M}_t\}_{t\in\mathbb{N}}$ defined as follows: $\mathcal{M}_t$ is the distribution of $U^t(d)$ over a random choice of $d \sim \{0,1\}^t$. Observe $\mathcal{M} \in \text{PS}{\scriptstyle\text{AMP}}$. By applying Lemma 7.5 to $\mathcal{M}$, we obtain a randomized polynomial-time algorithm A such that for all $t \in \mathbb{N}$ and all $\delta^{-1} \in \mathbb{N}$,

$$\Pr_{\substack{x\sim\mathcal{M}_t\\ A}}\left[(1-\delta)\cdot\mathcal{M}_t(x) \le A(x, 1^t, 1^{\delta^{-1}}) \le (1+\delta)\cdot\mathcal{M}_t(x)\right] \ge 1-\delta.$$

Define $M(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle}) = A(x, 1^t, 1^{\delta^{-1}(|x|t\delta^{-1}\alpha)^c 2^\alpha})$, where c is a sufficiently large constant we will choose later. Then, we have

$$\Pr_{\substack{x\sim\mathcal{M}_t\\ M}}\left[(1-\delta)\cdot\mathrm{Q}^t(x) \le M(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle}) \le (1+\delta)\cdot\mathrm{Q}^t(x)\right] \ge 1-\delta 2^{-\alpha}(|x|t\delta^{-1}\alpha)^{-c},$$

where we used the fact that $\delta^2 2^{-\alpha}(|x|t\delta^{-1}\alpha)^{-c} < \delta$ and $\mathcal{M}_t(x) = \mathrm{Q}^t(x)$. By Markov's inequality,

$$\Pr_{x\sim\mathcal{M}_t}\left[\Pr_M\left[(1-\delta)\cdot\mathrm{Q}^t(x) \le M(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle}) \le (1+\delta)\cdot\mathrm{Q}^t(x)\right] < 1-\delta\right] < 2^{-\alpha}(|x|t\delta^{-1}\alpha)^{-c}.$$

We prove the theorem by contraposition. For each $n \in \mathbb{N}$, let $E^n_\alpha \subseteq \{0,1\}^n$ be the subset of $x \in \{0,1\}^n$ satisfying

$$\Pr_M\left[(1-\delta)\cdot\mathrm{Q}^t(x) \le M(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle}) \le (1+\delta)\cdot\mathrm{Q}^t(x)\right] < 1-\delta;$$

i.e., E^n_α is the error set for M given the parameter α. It suffices to show $\mathrm{cd}^t(x) > \alpha$ for every $x \in E^n_\alpha$.

By the choice of the confidence parameter for A, it holds that $\sum_{x\in E^n_\alpha}\mathrm{Q}^t(x) \le (nt\delta^{-1}\alpha)^{-c}2^{-\alpha}$. Thus, we have $\sum_{x\in E^n_\alpha}\mathrm{Q}^t(x)(nt\delta^{-1}\alpha)^c 2^\alpha \le 1$.

Since the error set E^n_α is decidable (with given $M, t, \alpha, \delta^{-1}, c$ and unbounded computational resources) by trying all random strings for M, we can define a distribution family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ such that each $\mathcal{D}_n$ is (time-unconditionally) computable and $\mathcal{D}_n(x) \ge \mathrm{Q}^t(x)(nt\delta^{-1}\alpha)^c 2^\alpha$ for every $x \in E^n_\alpha$. By a coding theorem, this implies that

$$\mathrm{K}(x) \le -\log\mathrm{Q}^t(x) - c\log(nt\delta^{-1}\alpha) - \alpha + O(\log(nt\delta^{-1}\alpha) + \log c) < -\log\mathrm{Q}^t(x) - \alpha$$

by selecting a sufficiently large c.

Therefore, for every $n \in \mathbb{N}$ and every $x \in E^n_\alpha$,

$$\mathrm{cd}^t(x) = \mathrm{q}^t(x) - \mathrm{K}(x) = -\log\mathrm{Q}^t(x) - \mathrm{K}(x) > \alpha.$$

□

We can also show Theorem 7.3 by replacing $\mathcal{M} = \{\mathcal{M}_t\}_{t\in\mathbb{N}}$ in the proof above with $\mathcal{M} = \{\mathcal{M}_{\langle t,i\rangle}\}_{t,i\in\mathbb{N}}$, where each $\mathcal{M}_{\langle t,i\rangle}$ is the distribution of $U^t(d)_{[i]}$ over $d \sim \{0,1\}^t$, and defining M as $M(x; 1^{\langle t,\delta^{-1},2^\alpha\rangle}) := A(x, 1^{\langle t,|x|\rangle}, 1^{\delta^{-1}|x|^c 2^\alpha})$. For completeness, we prove Theorem 7.3 formally below.

Proof of Theorem 7.3. Consider the time-bounded universal distribution $\mathcal{M} = \left\{\mathcal{M}_{\langle t,i\rangle}\right\}_{t,i\in\mathbb{N}}$ defined as follows: $\mathcal{M}_{\langle t,i\rangle}$ is the distribution of $U^t(d)_{[i]}$ over a random choice of $d \sim \{0,1\}^t$. Observe $\mathcal{M} \in \text{PSAMP}$. By applying Lemma 7.5 to $\mathcal{M}$, we obtain a randomized polynomial-time algorithm A such that for all $t, i \in \mathbb{N}$ and all $\delta^{-1} \in \mathbb{N}$,

$$\Pr_{\substack{x\sim\mathcal{M}_{\langle t,i\rangle}\\ A}}\left[(1-\delta)\cdot\mathcal{M}_{\langle t,i\rangle}(x) \le A(x, 1^{\langle t,i\rangle}, 1^{\delta^{-1}}) \le (1+\delta)\cdot\mathcal{M}_{\langle t,i\rangle}(x)\right] \ge 1-\delta.$$

Define $M(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle}) = A(x, 1^{\langle t,|x|\rangle}, 1^{\delta^{-1}(|x|t\delta^{-1}\alpha)^c 2^\alpha})$, where c is a sufficiently large constant we will choose later. Then, we have

$$\Pr_{\substack{x\sim\mathcal{M}_{\langle t,|x|\rangle}\\ M}}\left[(1-\delta)\cdot\mathrm{Q}^{t,*}(x) \le M(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle}) \le (1+\delta)\cdot\mathrm{Q}^{t,*}(x)\right] \ge 1-\delta 2^{-\alpha}(|x|t\delta^{-1}\alpha)^{-c},$$

where we used the fact that $\delta^2 2^{-\alpha}(|x|t\delta^{-1}\alpha)^{-c} < \delta$ and $\mathcal{M}_{\langle t,|x|\rangle}(x) = \mathrm{Q}^{t,*}(x)$. By Markov's inequality,

$$\Pr_{x\sim\mathcal{M}_{\langle t,|x|\rangle}}\left[\Pr_M\left[(1-\delta)\cdot\mathrm{Q}^{t,*}(x) \le M(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle}) \le (1+\delta)\cdot\mathrm{Q}^{t,*}(x)\right] < 1-\delta\right] < 2^{-\alpha}(|x|t\delta^{-1}\alpha)^{-c}.$$

We prove the theorem by contraposition. For each $n \in \mathbb{N}$, let $E_\alpha^n \subseteq \{0,1\}^n$ be the subset of $x \in \{0,1\}^n$ satisfying

$$\Pr_M\left[(1-\delta)\cdot\mathrm{Q}^{t,*}(x) \le M(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle}) \le (1+\delta)\cdot\mathrm{Q}^{t,*}(x)\right] < 1-\delta;$$

i.e., E_α^n is the error set for M given the parameter α. It suffices to show $\mathrm{cd}^t(x) > \alpha$ for every $x \in E_\alpha^n$.

By the choice of the confidence parameter for A, it holds that $\sum_{x\in E_\alpha^n}\mathrm{Q}^{t,*}(x) \le (nt\delta^{-1}\alpha)^{-c}2^{-\alpha}$. Thus, we have $\sum_{x\in E_\alpha^n}\mathrm{Q}^{t,*}(x)(nt\delta^{-1}\alpha)^c 2^\alpha \le 1$.

Since the error set E_α^n is decidable (with given $M, t, \alpha, \delta^{-1}, c$ and unbounded computational resources) by trying all random strings for M, we can define a distribution family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ such that each $\mathcal{D}_n$ is (time-unconditionally) computable and $\mathcal{D}_n(x) \ge \mathrm{Q}^{t,*}(x)(nt\delta^{-1}\alpha)^c 2^\alpha$ for every $x \in E_\alpha^n$. This implies that

$$\begin{aligned}
\mathrm{K}(x) &\le -\log\mathrm{Q}^{t,*}(x) - c\log(nt\delta^{-1}\alpha) - \alpha + O(\log(nt\delta^{-1}\alpha) + \log c)\\
&\le -\log\mathrm{Q}^{t}(x) - c\log(nt\delta^{-1}\alpha) - \alpha + O(\log(nt\delta^{-1}\alpha) + \log c)\\
&< -\log\mathrm{Q}^{t}(x) - \alpha
\end{aligned}$$

by selecting a sufficiently large c, where the second inequality follows from $\mathrm{Q}^{t,*}(x) \ge \mathrm{Q}^t(x)$.

Therefore, for every $n \in \mathbb{N}$ and every $x \in E_\alpha^n$,

$$\mathrm{cd}^t(x) = \mathrm{q}^t(x) - \mathrm{K}(x) = -\log\mathrm{Q}^t(x) - \mathrm{K}(x) > \alpha.$$

□

7.3 Applications to Meta-Complexity Theoretic Characterizations of OWFs

We now use Theorem 7.1 to estimate the resource-unbounded Kolmogorov complexity of a string drawn from any unknown distribution.

With the ingredients developed so far, we now prove the main result of this section.

Proof of Theorem 7.2. Item 2 $\Rightarrow$ Item 1, Item 3 $\Rightarrow$ Item 1, and Item 4 $\Rightarrow$ Item 1 can be easily proved using [HILL99] (see [IRS22]) and Lemma 6.14 showing that most strings drawn from a samplable distribution is logarithmically small.

Item 1 $\Rightarrow$ Item 2. Using Theorem 7.1, let M be the algorithm of Item 2. We define an algorithm M' so that $M'(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle}) := -\log M(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle})$. By the property of M, for every $x \in \{0,1\}^*$ with $\mathrm{cd}^t(x) \le \alpha$, with probability at least $1-\delta$ over the randomness for M,

$$\mathrm{q}^t(x) - 1 \le -\log M(x, 1^t, 1^{\delta^{-1}}) \le \mathrm{q}^t(x) + 1.$$

In addition, for every $x \in \{0,1\}^*$ with $\mathrm{cd}^t(x) \le \alpha$,

$$\mathrm{q}^t(x) = \mathrm{K}(x) + \mathrm{cd}^t(x) \le \mathrm{K}(x) + \alpha.$$

By Fact 6.12,

$$\mathrm{K}(x) \le \mathrm{q}^t(x) + O(\log t).$$

Therefore, it holds that

$$\Pr_{M'}\left[\mathrm{K}(x) - O(\log t) - 1 \le M'(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle}) \le \mathrm{K}(x) + \alpha + 1\right] \ge 1-\delta.$$

Item 2 follows from subtracting $\alpha+1$ from the output of M'.

Item 2 $\Rightarrow$ Item 3. Let M be the efficient randomized algorithm in Item 2. We define an algorithm M' so that $M'(x, 1^t, 1^{\delta^{-1}}) = M(x, 1^{\langle t, 2\delta^{-1}, 2\delta^{-1}t^2\rangle})$. By Lemma 6.14, for every $\mathcal{D} \in \mathrm{PS{\small AMP}}$, there exists a polynomial t_0 such that for every sufficiently large $n \in \mathbb{N}$, every $t \ge t_0(n)$, and every $\delta^{-1} \in \mathbb{N}$,

$$\Pr_{x\sim\mathcal{D}_n}[\mathrm{cd}^t(x) \le \log 2\delta^{-1} + 2\log t] \ge 1 - \delta/2.$$

Whenever $\mathrm{cd}^t(x) \le \log 2\delta^{-1} + 2\log t$ holds,

$$\Pr_M[\mathrm{K}(x) - \log 2\delta^{-1} - 2\log t - O(\log t) \le M(x, 1^{\langle t,2\delta^{-1},2\delta^{-1}t^2\rangle}) \le \mathrm{K}(x)] \ge 1 - \delta/2.$$

Recall that

$$\log 2\delta^{-1} + 2\log t + O(\log t) \le \log\delta^{-1} + O(\log t).$$

Therefore, by the union bound,

$$\Pr_{M'}[\mathrm{K}(x) - \log\delta^{-1} - O(\log t) \le M'(x, 1^t, 1^{\delta^{-1}}) \le \mathrm{K}(x)] \ge 1-\delta.$$

Item 2 $\Rightarrow$ Item 4. Let M be the efficient randomized algorithm in Item 2. Let c' be a constant larger than the multiplicative constant in the O-notation in Item 2, i.e., for all $n, \delta^{-1}, t, \alpha \in \mathbb{N}$ and all $x \in \{0,1\}^n$ with $\mathrm{cd}^t(x) \le \alpha$,

$$\Pr_M\left[\mathrm{K}(x) - \alpha - c'\log t \le M(x, 1^{\langle t,\delta^{-1},2^\alpha\rangle}) \le \mathrm{K}(x)\right] \ge 1-\delta.$$

According to Fact 6.12, we can take sufficiently large c' so that $\mathrm{K}(x) \le \mathrm{q}^t(x) + c'\log t - 1$ for every x and t.

We determine the constant c in Item 4 as $c = c' + 1$.

Claim 7.8. *The following promise problem $\Pi = (\Pi_{\text{Yes}}, \Pi_{\text{No}})$ is in* AM*:*

$$\Pi_{\text{Yes}} = \{(x, 1^s, 1^t) : \mathrm{K}(x) \le s \ \wedge\ \mathrm{cd}^t(x) \le \log t\}$$
$$\Pi_{\text{No}} = \{(x, 1^s, 1^t) : \mathrm{K}(x) > s + c\log t \ \wedge\ \mathrm{cd}^t(x) \le \log t\}$$

We defer the proof of the following claim. Now, we prove that Item 2 implies $\Pi \in \mathsf{prBPP}$.

We define an algorithm $M'(x, 1^s, 1^t)$ as

$$M'(x, 1^s, 1^t) = 1 \iff M(x, 1^{\langle t,4,t\rangle}) \le s.$$

We show that

$$(x, 1^s, 1^t) \in \Pi_{\text{Yes}} \Longrightarrow \Pr[M'(x, 1^s, 1^t) = 1] \ge 3/4$$
$$(x, 1^s, 1^t) \in \Pi_{\text{No}} \Longrightarrow \Pr[M'(x, 1^s, 1^t) = 0] \ge 3/4.$$

If $(x, 1^s, 1^t) \in \Pi_{\text{Yes}}$, then $\mathrm{cd}^t(x) \le \log t$, and with probability at least $1 - 1/4 = 3/4$ over the choice of the randomness for M,

$$M(x, 1^{\langle t,4,t\rangle}) \le \mathrm{K}(x) \le s.$$

Therefore, $M'(x, 1^s, 1^t)$ outputs 1 in this case.

If $(x, 1^s, 1^t) \in \Pi_{\text{No}}$, then $\mathrm{cd}^t(x) \le \log t$, and with probability at least $1 - 1/4 = 3/4$ over the choice of the randomness for M,

$$M(x, 1^{\langle t,4,t\rangle}) \ge \mathrm{K}(x) - \log t - c'\log t > s + c\log t - (c'+1)\log t = s,$$

and $M'(x, 1^s, 1^t)$ outputs 0 in this case.

$\Pi \in \mathsf{AM}$ follows from the lower bound protocol [GS86; BT06] as follows.

Proof of Claim 7.8. Based on the lower bound protocol [cf. BT06, Lemma 2.6], the following language $\Gamma = (\Gamma_{\text{Yes}}, \Gamma_{\text{No}})$ is contained in AM.

$$\Gamma_{\text{Yes}} = \{(x, 1^s, 1^t) : \log|\{r \in \{0,1\}^t \ : U^t(r) = x\}| \ge s\}$$
$$\Gamma_{\text{No}} = \{(x, 1^s, 1^t) : \log|\{r \in \{0,1\}^t : U^t(r) = x\}| \le s - 1\}.$$

Since AM is closed under Karp reductions, it suffices to verify that

$$(x, 1^s, 1^t) \in \Pi_{\text{Yes}} \Longrightarrow (x, 1^{t-s-\log t}, 1^t) \in \Gamma_{\text{Yes}}$$
$$(x, 1^s, 1^t) \in \Pi_{\text{No}} \Longrightarrow (x, 1^{t-s-\log t}, 1^t) \in \Gamma_{\text{No}}.$$

If $(x, 1^s, 1^t) \in \Pi_{\text{Yes}}$, then

$$\mathrm{q}^t(x) = \mathrm{K}(x) + \mathrm{cd}^t(x) \le s + \log t.$$

Since $\mathrm{q}^t(x) = -\log \mathrm{Q}^t = t - \log|\{r \in \{0,1\}^t \ : U^t(r) = x\}|$, we have

$$\log|\{r \in \{0,1\}^t \ : U^t(r) = x\}| \ge t - s - \log t,$$

and $(x, 1^{t-s-\log t}, 1^t) \in \Gamma_{\text{Yes}}$.

By contrast, if $(x, 1^s, 1^t) \in \Pi_{\text{No}}$, then

$$\mathrm{q}^t(x) + c'\log t - 1 \ge \mathrm{K}(x) > s + c\log t.$$

Recall that $c = c' + 1$. Therefore, we have

$$\log|\{r \in \{0,1\}^t \ : U^t(r) = x\}| < t - s - \log t,$$

and $(x, 1^{t-s-\log t}, 1^t) \in \Gamma_{\text{No}}$. ◇

□

8 Universal Extrapolation

In this section, we formulate universal extrapolation and present the formal proof.

Theorem 8.1 (Universal Extrapolation)**.** *If there exists no infinitely-often one-way function, then there exists a randomized polynomial-time algorithm* UE *such that for all* $k, t, \epsilon^{-1}, \alpha \in \mathbb{N}$ *and all* $x \in \{0,1\}^*$ *with* $\mathrm{cd}^t(x) \leq \alpha$*,*

$$\mathsf{L}_1\left(\mathsf{UE}(x; 1^{\langle k,t,\epsilon^{-1},2^{\alpha}\rangle}), \mathsf{Next}_k(\mathrm{Q}^t, x)\right) \leq \epsilon.$$

We present two proofs of Theorem 8.1 with different constructions of UE. The first one is based on a distributional inverter [IL89]. The second one is based on the approximation of the universal a priori probability in Section 7, which appeared to be the original intention of [IL90].

8.1 Proof by Distributional Inverter

We present the proof of the universal extrapolation theorem based on distributional inverters. In the proof, we use the following theorem, which is well-known as the equivalence between the existence of one-way functions and the existence of distributional one-way functions.

Theorem 8.2 ([IL89; Imp92])**.** *The following are equivalent:*

1. *There exists no infinitely-often one-way function.*
2. *For every polynomial-time-computable family* $f = \{f_n \colon \{0,1\}^{s(n)} \to \{0,1\}^{t(n)}\}_{n\in\mathbb{N}}$*, there exists a randomized polynomial-time algorithm* A *such that for every* $n, \epsilon^{-1}, \delta^{-1} \in \mathbb{N}$*,*

$$\Pr_{y \sim f_n(U_{s(n)})}\left[\mathsf{L}_1\left(A(y; 1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}}), \mathsf{UnifInv}_{f_n}(y)\right) \leq \epsilon\right] \geq 1 - \delta,$$

where $\mathsf{UnifInv}_f(y)$ *is a random variable selected according to the uniform distribution over* $f^{-1}(y)$ *for each* $y \in \mathrm{Im} f$*.*

Theorem 8.2 implies distributional inverting under Q^t in worst-case exponential time in the time-bounded computational depth of input.

Lemma 8.3. *If there exists no infinitely-often one-way function, then there exist a randomized polynomial-time algorithm* A' *and a polynomial* τ *such that for all* $t, \epsilon^{-1}, \alpha \in \mathbb{N}$ *and all* $x \in \{0,1\}^*$ *with* $\mathrm{cd}^t(x) \leq \alpha$*,*

$$\mathsf{L}_1\left(A'(x; 1^{\langle t,\epsilon^{-1},1^{2^{\alpha}}\rangle}), \mathsf{UnifInv}_{U^t_{[|x|]}}(x)\right) \leq \epsilon,$$

where $U^t_{[n]}$ *denotes the universal Turing machine whose output is truncated to the first* n *bits; i.e.,* $U^t_{[n]}(s) = U^t(s)_{[n]}$ *for each* $s \in \{0,1\}^t$*.*

Proof. Let A be the randomized algorithm obtained from Theorem 8.2 (1⇒2) for a polynomial-time-computable family f defined as follows: for every $n, t \in \mathbb{N}$ and $s \in \{0,1\}^t$, $f_{\langle n,t\rangle}(s) = U^t_{[n]}(s)$.

For every input, A' is defined as

$$A'(x; 1^{\langle t,\epsilon^{-1},2^{\alpha}\rangle}) = A(x; 1^{\langle n,t\rangle}, 1^{\epsilon^{-1}}, 1^{(nt\epsilon^{-1}\alpha)^c 2^{\alpha}}),$$

where $n = |x|$ and c is a sufficiently large constant.

We prove the lemma by contraposition. For each $n \in \mathbb{N}$, let $E_\alpha^n \subseteq \{0,1\}^n$ be the subset of $x \in \{0,1\}^n$ satisfying

$$\mathsf{L}_1\left(A'(x; 1^{\langle t, \epsilon^{-1}, 2^\alpha\rangle}), \mathsf{UnifInv}_{U^t_{[n]}}(x)\right) > \epsilon;$$

i.e., E_α^n is the error set for A given the parameter α. We show $\mathrm{cd}^t(x) > \alpha$ for every $x \in E_\alpha^n$.

For each $x \in E_\alpha^n$, let $\delta(x) := \Pr_s[f_{\langle n,t\rangle}(s) = x]$. We have

$$\delta(x) = \Pr_s[f_{\langle n,t\rangle}(s) = x] = \Pr_s[U^t_{[|x|]}(s) = x] \geq \Pr_s[U^t(s) = x] = \mathrm{Q}^t(x).$$

For every $x \in E_\alpha^n$,

$$\mathsf{L}_1\left(A(x; 1^{\langle n,t\rangle}, 1^{\epsilon^{-1}}, 1^{n2^\alpha}), \mathsf{UnifInv}_{U^t_{[n]}}(x)\right) = \mathsf{L}_1\left(A(x; 1^{\langle n,t\rangle}, 1^{\epsilon^{-1}}, 1^{n2^\alpha}), \mathsf{UnifInv}_{f_{\langle n,t\rangle}}(x)\right) > \epsilon$$

By the choice of the confidence parameter for A, it holds that $\sum_{x \in E_\alpha^n} \delta(x) \leq (nt\epsilon^{-1}\alpha)^{-c}2^{-\alpha}$. Thus, we have $\sum_{x \in E_\alpha^n} \delta(x)(nt\epsilon^{-1}\alpha)^c 2^\alpha \leq 1$.

Since the error set E_α^n is decidable (with given $M, t, \epsilon^{-1}, \alpha, c$ and unbounded computational resources), we can define a distribution family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ such that each $\mathcal{D}_n$ is (time-unconditionally) computable and $\mathcal{D}_n(x) \geq \delta(x)(nt\epsilon^{-1}\alpha)^c 2^\alpha$ for every $x \in E_\alpha^n$. This implies that

$$\begin{aligned}
\mathrm{K}(x) &\leq -\log \mathcal{D}_n(x) + O(\log(nt\epsilon^{-1}\alpha) + \log c) \\
&\leq -\log \delta(n) - c\log(nt\epsilon^{-1}\alpha) - \alpha + O(\log(nt\epsilon^{-1}\alpha) + \log c) \\
&< -\log \delta(n) - \alpha \\
&\leq -\log \mathrm{Q}^t(x) - \alpha
\end{aligned}$$

by selecting a sufficiently large c.

Therefore, for every $n \in \mathbb{N}$ and every $x \in E_\alpha^n$,

$$\mathrm{cd}^t(x) = \mathrm{q}^t(x) - \mathrm{K}(x) = -\log \mathrm{Q}^t(x) - \mathrm{K}(x) > \alpha.$$

□

Now, we derive Theorem 8.1 from Lemma 8.3.

Proof of Theorem 8.1. Let A' be the randomized algorithm in Lemma 8.3. Then, the algorithm UE is constructed as

$$\mathsf{UE}(x; 1^{\langle k,t,\epsilon^{-1},2^\alpha\rangle}) = U^t(A'(x; 1^{\langle t,\epsilon^{-1},2^\alpha\rangle}))_{[|x|+1:|x|+k]}.$$

The correctness is verified as follows: for every $x \in \{0,1\}^*$,

$$\begin{aligned}
&\mathsf{L}_1\left(\mathsf{UE}(x; 1^{\langle k,t,\epsilon^{-1},2^\alpha\rangle}), \mathsf{Next}_k(\mathrm{Q}^t, x)\right) \\
&\quad = \mathsf{L}_1\left(U^t(A'(x; 1^{\langle t,\epsilon^{-1},2^\alpha\rangle}))_{[|x|+1:|x|+k]}, \mathsf{Next}_k(\mathrm{Q}^t, x)\right) \\
&\quad = \mathsf{L}_1\left(U^t(A'(x; 1^{\langle t,\epsilon^{-1},2^\alpha\rangle}))_{[|x|+1:|x|+k]}, U^t(\mathsf{UnifInv}_{U^t_{[|x|]}}(x))_{[|x|+1:|x|+k]}\right) \\
&\quad \leq \mathsf{L}_1\left(A'(x; 1^{\langle t,\epsilon^{-1},2^\alpha\rangle}), \mathsf{UnifInv}_{U^t_{[|x|]}}(x)\right).
\end{aligned}$$

By Lemma 8.3, for all $t, \epsilon^{-1}, \alpha \in \mathbb{N}$ and all $x \in \{0,1\}^*$ with $\mathrm{cd}^t(x) \le \alpha$,

$$\mathsf{L}_1\left(\mathsf{UE}(x; 1^{\langle k,t,\epsilon^{-1},2^\alpha\rangle}), \mathsf{Next}_k(\mathrm{Q}^t, x)\right) \le \mathsf{L}_1\left(A'(x; 1^{\langle t,\epsilon^{-1},2^\alpha\rangle}), \mathsf{UnifInv}_{U^t_{[|x|]}}(x)\right) \le \epsilon.$$

$\square$

8.2 Proof by Estimating Universal a Priori Probability

We present another proof of Theorem 8.1, where we construct another extrapolation algorithm that predicts the next k bits one-by-one according to the approximated likelihood of the next 1 bit.

Proof of Theorem 8.1. Let $\mathcal{D} = \{\mathcal{D}_{t,i}\}_{t,i\in\mathbb{N}}$ be a distribution family, where each $\mathcal{D}_{t,i}$ is a distribution of $x_{[i]} \circ b$ for $x \sim \mathrm{Q}^t$ and $b \sim \{0,1,\varepsilon\}$. Note that $\mathcal{D}$ is $\mathsf{poly}(t,i)$-time samplable.

Under the nonexistence of an infinitely-often one-way function, from Theorem 7.3 and Lemma 6.14, we have an algorithm M such that for any given parameters $1^{\langle t,i\rangle}, 1^{\delta^{-1}}$,

$$\Pr_{x\sim\mathcal{D}_{\langle t,i\rangle}, M}\left[\mathrm{Q}^{t,*}(x)\cdot(1-\delta) \le M(x; 1^{\langle t,i\rangle}, 1^{\delta^{-1}}) \le \mathrm{Q}^{t,*}(x)\cdot(1+\delta)\right] \ge 1-\delta.$$

We construct the algorithm UE based on M as follows. On input $x, 1^{\langle k,t,\epsilon^{-1},2^\alpha\rangle}$, the algorithm UE samples $y_j \in \{0,1,\varepsilon\}$ inductively in $j \in [k]$ according to the following procedure: if $y_{j-1} = \varepsilon$ and $j \ge 2$, then $y_j = \varepsilon$; otherwise,

$$y_j = \begin{cases} 0 & \text{with probability } \min\{p_0/p_\varepsilon, 1\} \\ 1 & \text{with probability } \min\{p_1/p_\varepsilon, 1\} \\ \varepsilon & \text{with probability } \max\{(p_\varepsilon - p_0 - p_1)/p_\varepsilon, 0\} \end{cases}$$

where

$$\begin{aligned} p_0 &= M(x \circ y_1 \circ \cdots \circ y_{j-1} \circ 0; 1^{\langle t,|x|+j-1\rangle}, 1^{\delta^{-1}}) \\ p_1 &= M(x \circ y_1 \circ \cdots \circ y_{j-1} \circ 1; 1^{\langle t,|x|+j-1\rangle}, 1^{\delta^{-1}}) \\ p_\varepsilon &= M(x \circ y_1 \circ \cdots \circ y_{j-1}; 1^{\langle t,|x|+j-1\rangle}, 1^{\delta^{-1}}) \end{aligned}$$

for $\delta = \min\{\epsilon'/(4k^2+\epsilon'), \epsilon'/6k\}$. Here, $\epsilon' = \epsilon/(6\cdot 2^\alpha(|x|t\epsilon^{-1}\alpha)^c)$, and c is a sufficiently large constant specified later. Then, UE outputs $y_1 \circ\cdots\circ y_k$. If $p_\varepsilon = 0$ at some stage, UE outputs "error" and halts.

We verify the correctness of UE via the following three steps. First, we assume the ideal case in which M can output the exact value of $\mathrm{Q}^{t,*}(x)$ for any given x regardless of its computational depth. Second, we take the multiplicative approximation error $(1\pm\delta)$ into account. Finally, we take the confidence error and the computational depth into account. Below, we omit the parameters of UE and M for readability.

Suppose that M can output the exact value of $\mathrm{Q}^{t,*}(x)$ for a given x with probability 1 over the choice of x and randomness for M. Then, by induction in j, we can easily verify that if $y_{j-1} \ne \varepsilon$, then for each $b \in \{0,1,\varepsilon\}$,

$$\Pr[y_j = b | y_1\cdots y_{j-1}] = \Pr[\mathsf{Next}_1(\mathrm{Q}^t, xy_1\cdots y_{j-1}) = b].$$

Thus, for every $y^* \in \{0,1\}^k$,

$$\begin{aligned}\Pr[\mathsf{UE}(x) \text{ outputs } y^*] &= \prod_{j=1}^{k} \Pr[y_j = y_j^* | y_{[j-1]} = y_{[j-1]}^*] \\ &= \prod_{j=1}^{k} \Pr[\mathsf{Next}_1(\mathrm{Q}^t, xy_1^* \cdots y_{j-1}^*) = y_j^*] \\ &= \Pr[\mathsf{Next}_k(\mathrm{Q}^t, x) = y^*],\end{aligned}$$

and for every $y^* \in \{0,1\}^{k'}$ with $k' < k$,

$$\begin{aligned}\Pr[\mathsf{UE}(x) \text{ outputs } y^*] &= \prod_{j=1}^{k'} \Pr[y_j = y_j^* | y_{[j-1]} = y_{[j-1]}^*] \cdot \Pr[y_{k'+1} = \varepsilon | y_{[k']} = y_{[k']}^*] \\ &= \prod_{j=1}^{k'} \Pr[\mathsf{Next}_1(\mathrm{Q}^t, xy_1^* \cdots y_{j-1}^*) = y_j^*] \cdot \Pr[\mathsf{Next}_1(\mathrm{Q}^t, xy_1^* \cdots y_{k'}^*) = \varepsilon] \\ &= \Pr[\mathsf{Next}_k(\mathrm{Q}^t, x) = y^*].\end{aligned}$$

Therefore, the distribution of $\mathsf{UE}(x)$ is statistically equivalent to $\mathsf{Next}_k(\mathrm{Q}^t, x)$.

Next, we take the approximation error into account; i.e., we assume that $M(x)$ outputs a value of $p \in [\mathrm{Q}^{t,*}(x)(1 \pm \delta)]$ for any given x.

Fix $j \in [k]$ and $x \in \{0,1\}^*$ arbitrarily. Notice that $p_b \in [\mathrm{Q}^{t,*}(xy_1 \cdots y_{j-1}b)(1 \pm \delta)]$ for each $b \in \{0,1,\varepsilon\}$. Let $\mathcal{D}_j$ denote the distribution of y_j given $y_1 \cdots y_{j-1}$. Then, the following claim holds:

Claim 8.4.

$$\mathsf{L}_1(\mathcal{D}_j, \mathsf{Next}_1(\mathrm{Q}^t, xy_1 \cdots y_{j-1})) \le \frac{\epsilon'}{2k^2}$$

We defer the proof of Claim 8.4 to the end of the proof because it has no technical novelty.

By Claim 8.4, we have

$$\begin{aligned}&\mathsf{L}_1(\mathcal{D}_1 \cdots \mathcal{D}_j, \mathsf{Next}_j(\mathrm{Q}^t, x)) \\ &\quad \le \mathsf{L}_1(\mathcal{D}_1 \cdots \mathcal{D}_j, \mathcal{D}_1 \cdots \mathcal{D}_{j-1}\mathsf{Next}_1(\mathrm{Q}^t, x \circ \mathcal{D}_1 \cdots \mathcal{D}_{j-1})) \\ &\qquad\qquad + \mathsf{L}_1(\mathcal{D}_1 \cdots \mathcal{D}_{j-1}\mathsf{Next}_1(\mathrm{Q}^t, x \circ \mathcal{D}_1 \cdots \mathcal{D}_{j-1}), \mathsf{Next}_j(\mathrm{Q}^t, x)) \\ &\quad \le \frac{\epsilon'}{2k^2} + \mathsf{L}_1(\mathcal{D}_1 \cdots \mathcal{D}_{j-1}\mathsf{Next}_1(\mathrm{Q}^t, x \circ \mathcal{D}_1 \cdots \mathcal{D}_{j-1}), \mathsf{Next}_{j-1}(\mathrm{Q}^t, x)\mathsf{Next}_1(\mathrm{Q}^t, x \circ \mathsf{Next}_{j-1}(\mathrm{Q}^t, x))) \\ &\quad \le \frac{\epsilon'}{2k^2} + \mathsf{L}_1(\mathcal{D}_1 \cdots \mathcal{D}_{j-1}, \mathsf{Next}_{j-1}(\mathrm{Q}^t, x)).\end{aligned}$$

By induction in $j \in [k]$, the total variation distance between $\mathsf{Next}_j(\mathrm{Q}^t, x)$ and the distribution $\mathcal{D}_1 \cdots \mathcal{D}_j$ of $y_1 \cdots y_j$ is at most $j \cdot \frac{\epsilon'}{2k^2}$. By letting $j = k$, the total variation distance between $\mathsf{Next}_k(\mathrm{Q}^t, x)$ and $\mathsf{UE}(x)$ is at most $\epsilon'/2k \le \epsilon/2$.

Finally, we take the confidence error into account. Namely, we assume that for each $x \in \{0,1\}^*$ and for each $j \in \mathbb{N}$ with $j \le k$,

$$\Pr_{xy_1 \cdots y_{j-1} \sim \mathrm{Q}^t_{[|x|+j-1]}, b \sim \{0,1,\varepsilon\}, M} \left[M(xy_1 \cdots y_{j-1}b) \text{ fails to output } (1 \pm \delta)\mathrm{Q}^{t,*}(xy_1 \cdots y_{j-1}b)\right] < \delta \le \frac{\epsilon'}{6k}.$$

Therefore, for every $b \in \{0, 1, \varepsilon\}$,

$$\Pr_{xy_1\cdots y_{j-1}\sim \mathrm{Q}^t_{[|x|+j-1]}, M}\left[M(xy_1\cdots y_{j-1}b) \text{ fails to output } (1\pm\delta)\mathrm{Q}^{t,*}(xy_1\cdots y_{j-1}b)\right] < 3\delta \le \frac{\epsilon'}{2k}.$$

Recall that, as long as M succeeds in outputting $(1\pm\delta)\mathrm{Q}^{t,*}(xy_1\cdots y_{j'-1}b)$ for each step $j' < j$, the distribution of $xy_1\cdots y_j$ is $\epsilon'/2k$-close to $\mathrm{Q}^t_{[i+j]}$. By the union bound, the probability that M fails at some stage $j \in [k]$ is bounded above by

$$\left(\frac{\epsilon'}{2k} + \frac{\epsilon'}{2k}\right)\cdot 3k = 3\epsilon' = \frac{\epsilon}{2}\cdot\frac{1}{2^\alpha(|x|t\epsilon^{-1}\alpha)^c}.$$

By Markov's inequality,

$$\Pr_{x\sim\mathrm{Q}^t_{[|x|]}}\left[\Pr_M\left[M \text{ fails at some stage in executing } \mathsf{UE}(x)\right] \le \epsilon/2\right] \ge 1 - \frac{1}{2^\alpha(|x|t\epsilon^{-1}\alpha)^c}.$$

For every choice of x that satisfies the event above, (i) if M does not fail, the total variation distance between $\mathsf{UE}(x)$ and $\mathsf{Next}_k(\mathrm{Q}^t, x)$ is at most $\epsilon/2$, and (ii) the probability that M fails at some stage is at most $\epsilon/2$. Thus, we conclude that the total variation distance between $\mathsf{UE}(x)$ and $\mathsf{Next}_k(\mathrm{Q}^t, x)$ is at most $\epsilon/2 + \epsilon/2 = \epsilon$ for such x, i.e.,

$$\Pr_{x\sim\mathrm{Q}^t_{[|x|]}}\left[\mathsf{L}_1(\mathsf{UE}(x), \mathsf{Next}_k(\mathrm{Q}^t, x)) \le \epsilon\right] \ge 1 - \frac{1}{2^\alpha(|x|t\epsilon^{-1}\alpha)^c}.$$

Recall that $\mathrm{Q}^t_{[|x|]}$ is the same distribution as the distribution of $f_{\langle |x|, t\rangle}(U_t)$ in the proof of Theorem 8.3. By selecting sufficiently large c and applying the same argument as Theorem 8.3, we have that for every x with $\mathrm{cd}^t(x) \le \alpha$,

$$\mathsf{L}_1(\mathsf{UE}(x), \mathsf{Next}_k(\mathrm{Q}^t, x)) \le \epsilon.$$

Thus, the remaining is the proof of Claim 8.4.

Proof of Claim 8.4. For each $b \in \{0, 1, \varepsilon\}$, let $p_b^* \in [0, 1]$ be $p_b^* = \mathrm{Q}^{t,*}(xy_1\cdots y_{j-1}b)$. Recall that $p_b \in [p_b^*(1\pm\delta)]$. Therefore,

$$\begin{aligned}
\mathsf{L}_1(\mathcal{D}_i, \mathsf{Next}_1(\mathrm{Q}^t, xy_1\cdots y_{j-1})) &\le \frac{1}{2}\sum_{b\in\{0,1\}}\left|\frac{p_b}{p_\varepsilon} - \frac{p_b^*}{p_\varepsilon^*}\right| + \frac{1}{2}\left|\frac{p_\varepsilon - p_0 - p_1}{p_\varepsilon} - \frac{p_\varepsilon^* - p_0^* - p_1^*}{p_\varepsilon^*}\right| \\
&= \frac{1}{2}\sum_{b\in\{0,1\}}\left|\frac{p_b}{p_\varepsilon} - \frac{p_b^*}{p_\varepsilon^*}\right| + \frac{1}{2}\left|\frac{p_0^* + p_1^*}{p_\varepsilon^*} - \frac{p_0 + p_1}{p_\varepsilon}\right| \\
&\le \sum_{b\in\{0,1\}}\left|\frac{p_b}{p_\varepsilon} - \frac{p_b^*}{p_\varepsilon^*}\right|.
\end{aligned}$$

For each $b \in \{0, 1\}$, if $\frac{p_b}{p_\varepsilon} - \frac{p_b^*}{p_\varepsilon^*} \ge 0$, then

$$\left|\frac{p_b}{p_\varepsilon} - \frac{p_b^*}{p_\varepsilon^*}\right| = \frac{p_b}{p_\varepsilon} - \frac{p_b^*}{p_\varepsilon^*} \le \frac{(1+\delta)p_b^*}{(1-\delta)p_\varepsilon^*} - \frac{p_b^*}{p_\varepsilon^*} = \frac{p_b^*}{p_\varepsilon^*}\cdot\frac{2\delta}{1-\delta};$$

otherwise,

$$\left|\frac{p_b}{p_\varepsilon} - \frac{p_b^*}{p_\varepsilon^*}\right| = \frac{p_b^*}{p_\varepsilon^*} - \frac{p_b}{p_\varepsilon} \leq \frac{p_b^*}{p_\varepsilon^*} - \frac{p_b^*(1-\delta)}{p_\varepsilon^*(1+\delta)} = \frac{p_b^*}{p_\varepsilon^*} \cdot \frac{2\delta}{1+\delta} \leq \frac{p_b^*}{p_\varepsilon^*} \cdot \frac{2\delta}{1-\delta}.$$

In any case,

$$\left|\frac{p_b}{p_\varepsilon} - \frac{p_b^*}{p_\varepsilon^*}\right| \leq \frac{p_b^*}{p_\varepsilon^*} \cdot \frac{2\delta}{1-\delta}.$$

Therefore,

$$\begin{aligned}
\mathsf{L}_1(\mathcal{D}_i, \mathsf{Next}_1(\mathrm{Q}^t, xy_1 \cdots y_{j-1})) &\leq \sum_{b \in \{0,1\}} \left|\frac{p_b}{p_\varepsilon} - \frac{p_b^*}{p_\varepsilon^*}\right| \\
&\leq \sum_{b \in \{0,1\}} \frac{p_b^*}{p_\varepsilon^*} \cdot \frac{2\delta}{1-\delta} \\
&= \frac{2\delta}{1-\delta} \cdot \frac{p_0^* + p_1^*}{p_\varepsilon^*} \\
&\leq \frac{2\delta}{1-\delta} \\
&\leq \frac{\epsilon}{2k^2},
\end{aligned}$$

where the last inequality is obtained by rearranging $\delta \leq \epsilon/(4k^2 + \epsilon)$. □

□

9 Distributional Learning in Pessiland

In this section, we consider the online learning framework introduced in Section 3.3. Recall that, in the framework, a learner first observes advice information $x^i \in \{0,1\}^*$ and then obtains $y^i \in \{0,1\}^*$ (we call y^i the i-th label) at stage $i \in \mathbb{N}$, where each data may be correlated with the previous streams. The task of the learner at stage i is, for a given advice string x_i, to predict the next outcome y_i.

We introduce several notions to discuss the learning framework above more formally. In Section 9, we use $a \in \mathbb{N} \cup \{0\}$ and $b \in \mathbb{N}$ to represent the size of each observation (x^i, y^i) as $|x^i| = a$ and $|y^i| = b$, and we use $m \in \mathbb{N}$ to represent the total number of stages. Our results hold in more general cases where $|x^i|$ and $|y^i|$ vary, by the same proof.

For every offline stream $x \in \{0,1\}^*$, a stream $x^1, y^1, \ldots, x^m, y^m$ of data for online learning is determined as follows: for each $i \in \mathbb{N}$,

$$x^i = x_{[(a+b)(i-1)+1:(a+b)(i-1)+a]} \text{ and } y^i = y_{[(a+b)(i-1)+a+1:(a+b)i]}.$$

Note that $x = x^1y^1 \circ \cdots \circ x^m y^m$, and x^i and y^i can be empty when $|x| < m(a+b)$. For each $i \in [m]$, we let $x^{<i} := x^1y^1 \circ \cdots \circ x^{i-1}y^{i-1} = x_{[(a+b)(i-1)]}$. Furthermore, for every distribution $\mathcal{D}$ on (offline) binary strings, we let $\mathcal{D}^{<i}$ represent a distribution of $x^{<i}$ for $x \sim \mathcal{D}$ and let $\mathcal{D}^{i,x^{<i}}$ represent the conditional distribution of the i-th advice string given $x^{<i}$, i.e., $\mathcal{D}^{i,x^{<i}} \equiv \mathsf{Next}_a(\mathcal{D}, x^{<i})$.

When the offline string x is selected according to some distribution $\mathcal{D}_z$ indexed by $z \in \{0,1\}^*$, the conditional distribution of the i-th label given $(x^{<i}, x^i)$ is $\mathsf{Next}_b(\mathcal{D}_z, x^{<i}x^i)$. When $\mathcal{D}_z$ and $x^{<i}$ are clear in context, we use the notation Label_i^{z,x^i} to refer to $\mathsf{Next}_b(\mathcal{D}_z, x^{<i}x^i)$.

Recall that a cheating learner is a learner that can freely observe the labels in the future through the additional oracle access to Label_i^{z,x^i}. The key insight for translating UE into learning algorithms in the framework above is that we can replace the oracle access to Label_i^{z,x^i} with UE in the average-case setting. This is stated as the following meta-theorem.

Theorem 9.1. *Suppose that* UE *in Theorem 8.1 exists. Then, for every oracle machine (i.e., a cheating learner)* $L_{cheat}^?$ *of polynomial-time computable query complexity* $q(\cdot)$*, there exist a randomized algorithm* L *and a polynomial* m_0 *satisfying the following: for every* $t_D(|z|)$*-time samplable family* $\mathcal{D} = \{\mathcal{D}_z\}_{z\in\{0,1\}^*}$*, where each* $\mathcal{D}_z$ *is over binary strings, every* $a \in \mathbb{N}\cup\{0\}$*, every* $s, b, t, \delta^{-1}, \lambda^{-1}, \alpha \in \mathbb{N}$ *with* $t \geq \max\{d(\mathcal{D}), t_D(s)\}$*, every* $z \in \{0,1\}^s$ *with* $\mathrm{cd}^t(z) \leq \alpha$*, every auxiliary input* $w \in \{0,1\}^*$*, and every* $m \geq m_0(d(\mathcal{D}), s, q(w), \delta^{-1}, \lambda^{-1})$*,*

$$\Pr_{i,x^{<i},x^i}\left[\mathsf{L}_1\left(L(x^{<i}, x^i, w; 1^{\langle s,b,t,2^\alpha,\delta^{-1},\lambda^{-1}\rangle}), L_{cheat}^{\mathsf{Label}_i^{z,x^i}}(w)\right) \leq \lambda\right] \geq 1-\delta,$$

where $i \sim [m]$*,* $x^{<i} \sim \mathcal{D}_z^{<i}$*, and* $x^i \sim \mathcal{D}_z^{i,x^{<i}}$*.*

Furthermore, $m_0(d(\mathcal{D}), s, q, \delta^{-1}, \lambda^{-1}) = O((d(\mathcal{D}) + s)\cdot q \cdot \delta^{-1}\lambda^{-2})$*, and* L *halts in polynomial time in the input length and the running time of* L_{cheat}*.*

The proof of Theorem 9.1 is presented in Section 9.1. Theorem 9.1 is sufficient for simulating any polynomial-time cheating learners with polynomial-time overhead. We present distributional learning and learning ACDs (i.e., Theorem 2.1) in Section 9.2. In addition, we present applications to two natural learning tasks (i) finding top-k most possible labels and (ii) estimating likelihood of given labels in Section 9.3 and 9.4, respectively.

9.1 Time-Bounded Universal Inductive Inference

First, we show the following key lemma, which is an extension of Lemma 3.2.

Lemma 9.2. *For every distribution* $\mathcal{D}$ *over binary strings such that* $\mathcal{D}$ *has a* t_D*-time sampler described by* d *bits, and for every* $a \in \mathbb{N}\cup\{0\}$*, every* $t, q, b, m \in \mathbb{N}$ *with* $t \geq \tau_{\mathrm{dom}}(d, t_D)$*, and every* q*-query (possibility not efficiently computable) randomized oracle machine* I*,*

$$\mathop{\mathbb{E}}_{i\sim[m],x^{<i}\sim\mathcal{D}^{<i},x^i\sim\mathcal{D}^{i,x^{<i}}}\left[\mathrm{KL}\left(I^{\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)}||I^{\mathsf{Next}_b(\mathrm{Q}^t,x^{<i}x^i)}\right)\right] \leq q\cdot\frac{O(d)}{m},$$

where the hidden constant in $O(d)$ *depends on only the universal Turing machine.*

Proof. Fix $a \in \mathbb{N}\cup\{0\}$ and $b, m, q \in \mathbb{N}$ arbitrarily. For each $x \in \mathrm{supp}(\mathcal{D})$, we use the notations $x^1, y^1, \ldots, x^m, y^m$ and $x^{<i}$ as defined at the beginning of Section 9. In addition, we introduce random variables $X_D^1, Y_D^1, \ldots, X_D^m, Y_D^m$ and $X_{\mathrm{Q}^t}^1, Y_{\mathrm{Q}^t}^1, \ldots, X_{\mathrm{Q}^t}^m, Y_{\mathrm{Q}^t}^m$ as follows: for each $i \in [m]$,

$$\begin{aligned}X_D^i &:= \mathcal{D}_{[(a+b)(i-1)+1:(a+b)(i-1)+a]}\\ Y_D^i &:= \mathcal{D}_{[(a+b)(i-1)+a+1:(a+b)i]}\\ X_{\mathrm{Q}^t}^i &:= \mathrm{Q}^t_{[(a+b)(i-1)+1:(a+b)(i-1)+a]}\\ Y_{\mathrm{Q}^t}^i &:= \mathrm{Q}^t_{[(a+b)(i-1)+a+1:(a+b)i]}.\end{aligned}$$

In words, X_D^i and Y_D^i (resp. $X_{\mathrm{Q}^t}^i$ and $Y_{\mathrm{Q}^t}^i$) represent the i-th advice string and the i-th label drawn from $\mathcal{D}$ (resp. Q^t), respectively. Note that

$$\begin{aligned}\mathcal{D}_{[(a+b)m]} &= X_D^1 \circ Y_D^1 \circ \cdots \circ X_D^m \circ Y_D^m \\ \mathrm{Q}^t_{[(a+b)m]} &= X_{\mathrm{Q}^t}^1 \circ Y_{\mathrm{Q}^t}^1 \circ \cdots \circ X_{\mathrm{Q}^t}^m \circ Y_{\mathrm{Q}^t}^m.\end{aligned}$$

For every q-query randomized oracle machine I and every distribution $\mathcal{O}$, the distribution of $I^{\mathcal{O}}$ is considered as $I(a_1, \dots, a_q)$ for $a_1, \dots, a_q \sim \mathcal{O}$, where we regard I as a randomized function. Thus, for any distributions $\mathcal{O}$ and $\mathcal{O}'$ with $\mathrm{KL}(\mathcal{O}||\mathcal{O}') < \infty$, we have

$$\mathrm{KL}(I^{\mathcal{O}}||I^{\mathcal{O}'}) \leq \mathrm{KL}(I(\mathcal{O}_1, \dots, \mathcal{O}_q)||I(\mathcal{O}'_1, \dots, \mathcal{O}'_q)) \leq \mathrm{KL}(\mathcal{O}_1, \dots, \mathcal{O}_q||\mathcal{O}'_1, \dots, \mathcal{O}'_q) = q \cdot \mathrm{KL}(\mathcal{O}||\mathcal{O}'),$$

where each $\mathcal{O}_i$ (resp. $\mathcal{O}'_i$) is an independent random variable drawn from $\mathcal{O}$ (resp. $\mathcal{O}'$), and the second inequality follows from Fact 6.4.

If $t \geq \tau_{\mathrm{dom}}(d, t_D)$, then $\mathrm{Q}^t(x) \geq \mathcal{D}(x)/2^{O(d)}$ for every $x \in \{0,1\}^*$ by Lemma 6.9; thus, for every $x \in \{0,1\}^*$, $\mathrm{KL}(\mathsf{Next}_b(\mathcal{D}, x)||\mathsf{Next}_b(\mathrm{Q}^t, x)) < \infty$.

Therefore, for each $i \in [m]$,

$$\begin{aligned}&\mathop{\mathbb{E}}_{x^{<i} \sim \mathcal{D}^{<i}, x^i \sim \mathcal{D}^{i, x^{<i}}}\left[\mathrm{KL}\left(I^{\mathsf{Next}_b(\mathcal{D}, x^{<i}x^i)}||I^{\mathsf{Next}_b(\mathrm{Q}^t, x^{<i}x^i)}\right)\right] \\ &\quad= q \cdot \mathop{\mathbb{E}}_{x^{<i} \sim \mathcal{D}^{<i}, x^i \sim \mathcal{D}^{i, x^{<i}}}\left[\mathrm{KL}\left(\mathsf{Next}_b(\mathcal{D}, x^{<i}x^i)||\mathsf{Next}_b(\mathrm{Q}^t, x^{<i}x^i)\right)\right] \\ &\quad= q \cdot \mathrm{KL}\left(\left(Y_D^i|X_D^1, Y_D^1, \dots, Y_D^{i-1}, X_D^i\right)\middle|\middle|\left(Y_{\mathrm{Q}^t}^i|X_{\mathrm{Q}^t}^1, Y_{\mathrm{Q}^t}^1, \dots, Y_{\mathrm{Q}^t}^{i-1}, X_{\mathrm{Q}^t}^i\right)\right). \qquad (7)\end{aligned}$$

Thus, we obtain that for every $t \geq \tau_{\mathrm{dom}}(d, t_D)$,

$$\begin{aligned}&\mathop{\mathbb{E}}_{i \sim [m], x^{<i} \sim \mathcal{D}^{<i}, x^i \sim \mathcal{D}^{i, x^{<i}}}\left[\mathrm{KL}\left(I^{\mathsf{Next}_b(\mathcal{D}, x^{<i}x^i)}||I^{\mathsf{Next}_b(\mathrm{Q}^t, x^{<i}x^i)}\right)\right] \\ &\quad= \frac{1}{m}\sum_{i=1}^m \mathop{\mathbb{E}}_{x^{<i} \sim \mathcal{D}^{<i}, x^i \sim \mathcal{D}^{i, x^{<i}}}\left[\mathrm{KL}\left(I^{\mathsf{Next}_b(\mathcal{D}, x^{<i}x^i)}||I^{\mathsf{Next}_b(\mathrm{Q}^t, x^{<i}x^i)}\right)\right] \\ &\quad= \frac{q}{m}\sum_{i=1}^m \mathrm{KL}\left(\left(Y_D^i|X_D^1, Y_D^1, \dots, Y_D^{i-1}, X_D^i\right)\middle|\middle|\left(Y_{\mathrm{Q}^t}^i|X_{\mathrm{Q}^t}^1, Y_{\mathrm{Q}^t}^1, \dots, Y_{\mathrm{Q}^t}^{i-1}, X_{\mathrm{Q}^t}^i\right)\right) \\ &\quad= \frac{q}{m}\Bigg(\mathrm{KL}\left(X_D^1, Y_D^1, \dots, X_D^m, Y_D^m\middle|\middle|X_{\mathrm{Q}^t}^1, Y_{\mathrm{Q}^t}^1, \dots, X_{\mathrm{Q}^t}^m, Y_{\mathrm{Q}^t}^m\right) \\ &\qquad\qquad - \sum_{i=1}^m \mathrm{KL}\left(\left(X_D^i|X_D^1, Y_D^1, \dots, X_D^{i-1}, Y_D^{i-1}\right)\middle|\middle|\left(X_{\mathrm{Q}^t}^i|X_{\mathrm{Q}^t}^1, Y_{\mathrm{Q}^t}^1, \dots, X_{\mathrm{Q}^t}^{i-1}, Y_{\mathrm{Q}^t}^{i-1}\right)\right)\Bigg) \\ &\quad\leq \frac{q}{m} \cdot \mathrm{KL}\left(X_D^1, Y_D^1, \dots, X_D^m, Y_D^m\middle|\middle|X_{\mathrm{Q}^t}^1, Y_{\mathrm{Q}^t}^1, \dots, X_{\mathrm{Q}^t}^m, Y_{\mathrm{Q}^t}^m\right) \\ &\quad\leq \frac{q}{m} \cdot \mathrm{KL}\left(\mathcal{D}||\mathrm{Q}^t\right) \\ &\quad\leq \frac{q}{m} \cdot O(d),\end{aligned}$$

where the second equality follows from Eq. (7), the third equality follows from the chain rule for the KL divergence (Lemma 6.3), the first inequality follows from the non-negativity of the KL divergence, the second inequality follows from Fact 6.4, and the last inequality follows from $t \geq \tau_{\mathrm{dom}}(d, t_D)$), Lemma 6.9, and Proposition 6.10. □

Theorem 9.1 follows from Theorem 8.1 and Lemma 9.2.

Proof of Theorem 9.1. Let UE be the universal extrapolation algorithm in Theorem 8.1. Let τ and τ' be the polynomials in lemma 6.15. Let $L^{?}_{cheat}$ be an arbitrary oracle machine (a cheating learner) of polynomial-time computable query complexity $q := q(w)$. Let $\mathcal{D} = \{\mathcal{D}_z\}_{z\in\{0,1\}^*}$ be a $t_D(|z|)$-time samplable distribution family.

We construct the learner L that executes L_{cheat}, where the query access is simulated by UE. The formal construction is as follows: On input $x^{<i}, x^i, w, 1^{\langle s,b,t,1^\alpha,\delta^{-1},\lambda^{-1}\rangle}$, the learner L executes the cheating learner $L^{?}_{cheat}(w)$, where L answers each query access to Label_i^{z,x^i} by

$$\mathsf{ans} \leftarrow \mathsf{UE}(x^{<i}x^i; 1^{\langle b,t',\lambda'^{-1},1^{\alpha'}\rangle})$$

for $t' = \max\{\tau(\tau'(t,t)), \tau_1(t,s,t)\}$, $\lambda' = \lambda/(2q(w))$, and $\alpha' = \alpha + \log 2\delta^{-1} + 2\log t'$ where L' uses fresh randomness to execute UE for each access, and τ_1 is a sufficiently large polynomial specified later. It is easy to verify that L halts in polynomial time in the input length and the running time of $L_{cheat}(w)$. Below, we show the correctness of L. For readability, we omit parameters for L and UE.

By Lemma 9.2, there exists a polynomial τ_1 such that for every $a \in \mathbb{N}\cup\{0\}$, every $s,t,m,b \in \mathbb{N}$, every $z \in \{0,1\}^s$ with $t \geq \tau_1(d(\mathcal{D}), s, t_D(s))$, and every $w \in \{0,1\}^*$,

$$\mathop{\mathbb{E}}_{i\sim[m],x^{<i}\sim\mathcal{D}_z^{<i},x^i\sim\mathcal{D}_z^{i,x^{<i}}}\left[\mathrm{KL}(L_{cheat}^{\mathsf{Label}_i^{z,x^i}}(w)||L_{cheat}^{\mathsf{Next}_b(\mathsf{Q}^t,x^{<i}x^i)}(w))\right] \leq \frac{q\cdot c'(s+d(\mathcal{D}))}{m}$$

for some universal constant $c' > 0$.

Let $m_0 := \frac{q\cdot 4c'(s+d(\mathcal{D}))}{\lambda^2\delta} = O(\frac{q\cdot(s+d(\mathcal{D}))}{\lambda^2\delta})$. Then, for every $m \geq m_0$, $z \in \{0,1\}^s$, and $w \in \{0,1\}^*$, we have

$$\mathop{\mathbb{E}}_{i,x^{<i},x^i}\left[\mathrm{KL}(L_{cheat}^{\mathsf{Label}_i^{z,x^i}}(w)||L_{cheat}^{\mathsf{Next}_b(\mathsf{Q}^t,x^{<i}x^i)}(w))\right] \leq \frac{q\cdot c'(s+d(\mathcal{D}))}{m} \leq \frac{\lambda^2\delta}{4}.$$

By the non-negativity of KL divergence and Markov's inequality,

$$\Pr_{i,x^{<i},x^i}\left[\mathrm{KL}(L_{cheat}^{\mathsf{Label}_i^{z,x^i}}(w)||L_{cheat}^{\mathsf{Next}_b(\mathsf{Q}^t,x^{<i}x^i)}(w)) > \frac{\lambda^2}{2}\right] < \frac{\delta}{2}.$$

By Pinsker's inequality (Fact 6.1),

$$\Pr_{i,x^{<i},x^i}\left[\mathsf{L}_1(L_{cheat}^{\mathsf{Label}_i^{z,x^i}}(w), L_{cheat}^{\mathsf{Next}_b(\mathsf{Q}^t,x^{<i}x^i)}(w)) \leq \frac{\lambda}{2}\right]$$
$$\geq \Pr_{i,x^{<i},x^i}\left[\mathrm{KL}(L_{cheat}^{\mathsf{Label}_i^{z,x^i}}(w)||L_{cheat}^{\mathsf{Next}_b(\mathsf{Q}^t,x^{<i}x^i)}(w)) \leq \frac{\lambda^2}{2}\right] \geq 1 - \frac{\delta}{2}. \quad (8)$$

We remark that the above holds for any $a \in \mathbb{N}\cup\{0\}$, any $s,b,t,m,\delta^{-1},\lambda^{-1} \in \mathbb{N}$, any $z \in \{0,1\}^s$, and any $w \in \{0,1\}^*$ satisfying $t \geq \tau_1(d(\mathcal{D}), s, t_D(s))$ and $m \geq m_0$.

For all $a \in \mathbb{N}\cup\{0\}$, $s,m,t,b,\delta^{-1},\lambda^{-1},\alpha \in \mathbb{N}$, $i \in [m]$, and $z,w \in \{0,1\}^*$ with $m \geq m_0$, $\mathrm{cd}^t(z) \leq \alpha$, and $t \geq \max\{d(\mathcal{D}), t_D(s)\}$, we have $\tau'(t,t) \geq \tau'(d(\mathcal{D}), t_D(s))$. By Lemma 6.15,

$$\Pr_{x^{<i}\sim\mathcal{D}_z^{<i},x^i\sim\mathcal{D}_z^{i,x^{<i}}}[\mathrm{cd}^{\tau(\tau'(t,t))}(x^{<i}x^i) \leq \mathrm{cd}^{\tau'(t,t)}(z) + \log 2\delta^{-1} + 2\log\tau'(t,t)] \geq 1 - \frac{\delta}{2}.$$

In this case, by the choices of t' and α',

$$\begin{aligned}
\mathrm{cd}^{t'}(x^{<i}x^i) &\le \mathrm{cd}^{\tau(\tau'(t,t))}(x^{<i}x^i)\\
&\le \mathrm{cd}^{\tau'(t,t)}(z) + \log 2\delta^{-1} + 2\log \tau'(t,t)\\
&\le \mathrm{cd}^{t}(z) + \log 2\delta^{-1} + 2\log t'\\
&\le \alpha + \log 2\delta^{-1} + 2\log t' = \alpha'.
\end{aligned}$$

Therefore, by Theorem 8.1,

$$\Pr_{x^{<i}\sim\mathcal{D}_z^{<i},\, x^i\sim\mathcal{D}_z^{i,x^{<i}}}\left[\mathsf{L}_1\left(\mathsf{UE}(x^{<i}x^i;1^{\langle b,t',\lambda'^{-1},1^{\alpha'}\rangle}),\mathsf{Next}_b(\mathrm{Q}^{t'},x^{<i}x^i)\right)\le\frac{\lambda}{2q}\right]\ge 1-\frac{\delta}{2}. \tag{9}$$

Recall that (i) L simulates the oracle access by $\mathsf{UE}(x^{<i}x^i)$, and (ii) L_{cheat} accesses the oracle at most q times. Thus, for every $i, x^{<i}, x^i$ satisfying the event in Eq. (9) ,

$$\mathsf{L}_1(L(x^{<i},x^i,w),L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'},x^{<i}x^i)}(w)) = \mathsf{L}_1(L_{cheat}^{\mathsf{UE}(x^{<i}x^i)}(w),L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'},x^{<i}x^i)}(w)) \le q\cdot\frac{\lambda}{2q}=\frac{\lambda}{2}.$$

Because $t'\ge\tau_1(t,s,t)\ge\tau_1(d(\mathcal{D}),s,t_D(s))$, by Eq. (8) and Eq. (9) and the union bound,

$$\mathsf{L}_1(L_{cheat}^{\mathsf{Label}_i^{z,x^i}}(w),L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'},x^{<i}x^i)}(w))\le\frac{\lambda}{2}$$

and

$$\mathsf{L}_1(L(x^{<i},x^i,w),L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'},x^{<i}x^i)}(w))\le\frac{\lambda}{2}$$

hold with probability at least $1-\delta$ over the choice of $i\sim[m], x^{<i}\sim\mathcal{D}_z^{<i}$ and $x^i\sim\mathcal{D}_z^{i,x^{<i}}$. In this case, we have

$$\begin{aligned}
&\mathsf{L}_1\left(L(x^{<i},x^i,w),L_{cheat}^{\mathsf{Label}_i^{z,x^i}}(w)\right)\\
&\quad\le \mathsf{L}_1\left(L(x^{<i},x^i,w),L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'},x^{<i}x^i)}(w)\right)+\mathsf{L}_1\left(L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'},x^{<i}x^i)}(w),L_{cheat}^{\mathsf{Label}_i^{z,x^i}}(w)\right)\\
&\quad\le\frac{\lambda}{2}+\frac{\lambda}{2}=\lambda.
\end{aligned}$$

Thus, we conclude that

$$\Pr_{i,x^{<i},x^i}\left[\mathsf{L}_1\left(L(x^{<i},x^i,w),L_{cheat}^{\mathsf{Label}_i^{z,x^i}}(w)\right)\le\lambda\right]\ge 1-\delta.$$

□

9.2 Universal Distributional Learning and Universal Learning ACDs

In this section, we consider the problems of learning unknown distributions from samples, which was first studied by Kearns, Mansour, Ron, Rubinfeld, Schapire, and Sellie [KMRRSS94] (see also [Xia10]).

9.2.1 Definitions of Learning Models

First, we formally introduce the learning models, i.e., distributional learning and learning ACDs, where the latter was introduced by Naor and Rothblum [NR06].

Distributional Learning. We define a sampler of sample size n as a multi-output algorithm that is given a random seed as input and outputs an n-bit string. For convenience, we identify a sampler $S\colon \{0,1\}^\ell \to \{0,1\}^n$ with a distribution of $S(r)$ for $r \sim \{0,1\}^\ell$. For each sampler S, we define an example oracle EX_S as the oracle that returns $x \sim S$ for each access. For simplicity, we define the time complexity of sampler as a function in the sample size n instead of the seed length ℓ. For any $t, s \in \mathbb{N}$, we say that a sampler S of sample size n is t/s-time computable if there exists a program $\Pi_S \in \{0,1\}^{\le s}$ such that $U^t(\Pi_S, r) = S(r_{[\ell]})$ for each seed $r \sim \{0,1\}^t$. We also define the t'-time-bounded computational depth of a t/s-time computable sampler S as $\min_{\Pi_S} \mathrm{cd}^{t'}(\Pi_S)$, where the minimum is taken over programs $\Pi_S \in \{0,1\}^{\le s}$ satisfying $U^t(\Pi_S, r) = S(r_{[\ell]})$ for each seed $r \sim \{0,1\}^t$ (such Π_S exists since S is t/s-time computable).

A distributional learner for t/s-time samplable distributions is given oracle access to EX_S for an unknown t/s-time computable sampler S and attempts to construct a sampler that statistically simulates S.

Definition 9.3 (Distributional learning)**.** *Let $\mathcal{S}$ be a class of samplers. We say that $\mathcal{S}$ is distributionally learnable in polynomial time if there exists a polynomial-time randomized oracle machine (i.e., learner) L such that for every sufficiently large $n \in \mathbb{N}$, every $\epsilon^{-1}, \delta^{-1} \in \mathbb{N}$ and every sampler $S_n \in \mathcal{S}$ of sample size n, the algorithm L satisfies*

$$\Pr_{\mathsf{EX}_{S_n}, L}\left[L^{\mathsf{EX}_{S_n}}(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}}) \textit{ outputs a circuit } h \textit{ s.t. } \mathsf{L}_1(S_n, h(r)) \le \epsilon\right] \ge 1 - \delta,$$

where r represents a uniformly random seed for h. We also define the sample complexity $m(n, \epsilon, \delta)$ as the upper bound of the number of oracle accesses $L(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$ requires.

Note that the learner L does not know the target sampler S_n, except for the prior knowledge of the class $\mathcal{S}$ (i.e., modeling of environment).

We also consider the average-case variant of distributional learning. We define *a distribution on samplers* as a family $\mathcal{G} = \{\mathcal{G}_n\}_{n\in\mathbb{N}}$ of distributions, where each $\mathcal{G}_n$ is a distribution on descriptions of samplers of sample size n. For every distribution $\mathcal{G}$ on $t(n)/s(n)$-time computable samplers and every $n \in \mathbb{N}$, we use the notation $\mathcal{G}_n$ to refer to the n-th distribution in $\mathcal{G}$, i.e., the distribution on (at most $s(n)$-bit) descriptions of a $t(n)/s(n)$-time sampler of sample size n.

Definition 9.4 (Distributional learning on average)**.** *Let $\mathcal{C}$ be a class of distributions on the class $\mathcal{S}$ of samplers. We say that $\mathcal{S}$ is distributionally learnable in polynomial time on average under $\mathcal{C}$ if there exists a polynomial-time randomized oracle machine (i.e., learner) L such that for every distribution $\mathcal{G} \in \mathcal{C}$ (note that $\mathcal{G}$ is a distribution on samplers), every sufficiently large $n \in \mathbb{N}$, and every $\epsilon^{-1}, \delta^{-1} \in \mathbb{N}$, the algorithm L satisfies*

$$\Pr_{S_n\sim\mathcal{G}_n, \mathsf{EX}_{S_n}, L}\left[L^{\mathsf{EX}_{S_n}}(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}}) \textit{ outputs a circuit } h \textit{ s.t. } \mathsf{L}_1(S_n, h(r)) \le \epsilon\right] \ge 1 - \delta,$$

where r represents a uniformly random seed for h. We also define the sample complexity $m(n, \epsilon, \delta)$ as the upper bound of the number of oracle accesses $L(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$ requires.

Note that the learner L knows neither the target sampler S_n nor the underlying distribution $\mathcal{G}$.

Learning Adaptively Changing Distributions. Next, we introduce learning ACDs first studied in [NR06].

An ACD (adaptively changing distribution) is a randomized Turing machine D satisfying the following syntax: For every sample size $n \in \mathbb{N}$,

1. D takes two inputs 1^n and $\sigma \in \{0,1\}^*$, where σ is called an internal state and initialized by some initial state $s_0 \in \{0,1\}^*$. (In the original definition in [NR06], s_0 is also selected according to a samplable distribution. In this section, we apply a more general definition.)

2. For any $\sigma \in \{0,1\}^*$, the algorithm $D(1^n, \sigma)$ randomly generates a sample $x \in \{0,1\}^*$ and a next state $s' \in \{0,1\}^*$ (x and s' can be correlated).

Then, any ACD D determines an example oracle $\mathsf{EX}_{n,s_0,D}$ for each sample size $n \in \mathbb{N}$ and each initial state s_0, as follows:

1. $\mathsf{EX}_{n,s_0,D}$ has a hidden internal state σ, which is initialized by s_0.

2. For each query access (without input), $\mathsf{EX}_{n,s_0,D}$ generates $(x, s') \leftarrow A(1^n, \sigma)$ and returns x as a sample. Then, $\mathsf{EX}_{n,s_0,D}$ updates the internal state σ as $\sigma := s'$.

For any functions $s(n)$ and $t(n)$, we say that an ACD D is $t(n)$-time samplable and has an $s(n)$-bit initial state if for every $n \in \mathbb{N}$ and every initial state $\sigma_0 \in \{0,1\}^{\leq s(n)}$, for every possible state σ in the execution with initial state σ_0, $D(1^n, \sigma)$ halts in $t(n)$ time (i.e., $\sigma \in \{0,1\}^{\leq t(n)}$).

In learning ACD D, a learner has query access to $\mathsf{EX}_{n,s_0,D}$ for a given parameter 1^n, where s_0 is a hidden initial state. The goal of the learner is to select some stage $i \in \mathbb{N}$ and, after observing the first i samples $x^1, \ldots, x^i$ from $\mathsf{EX}_{n,s_0,D}$, to statistically simulate the conditional distribution of the next sample x^{i+1} given the initial state s_0 and $x^1, \ldots, x^i$. For convenience, we use the notation $D_i^{s_0}(x^1, \ldots, x^i)$ to refer to the conditional distribution that the learner attempts to simulate at stage i.

Definition 9.5 (Learning ACDs). *Let $s(n)$ and $t(n)$ be polynomials. Let $\mathcal{S} = \{\mathcal{S}_n\}_{n\in\mathbb{N}}$, where $\mathcal{S}_n \subseteq \{0,1\}^{\leq s(n)}$, be a subset of initial states. We say that $t(n)$-time samplable ACDs of $s(n)$-bit initial state in $\mathcal{S}$ are learnable in polynomial time if there exists a randomized polynomial-time algorithm L such that for every $t(n)$-time samplable ACD D of $s(n)$-bit initial state, every sufficiently large $n \in \mathbb{N}$, every $\epsilon^{-1}, \delta^{-1} \in \mathbb{N}$, and every $s_0 \in \mathcal{S}_n$, the algorithm L satisfies the following with probability at least $1 - \delta$ over the choice of samples from $\mathsf{EX}_{n,s_0,D}$ and randomness for L:*

1. *$L^{\mathsf{EX}_{n,s_0,D}}(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$ obtains samples $x^1, x^2, \ldots,$ from $\mathsf{EX}_{n,s_0,D}$.*

2. *After obtaining i samples $x^1, \ldots, x^i$ (where i is selected by L), $L^{\mathsf{EX}_{n,s_0,D}}(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$ outputs some circuit h as a hypothesis without additional access to $\mathsf{EX}_{n,s_0,D}$.*

3. *The hypothesis h satisfies $\mathsf{L}_1(D_i^{s_0}(x^1, \ldots, x^i), h(r)) \leq \epsilon$, where r represents a uniformly random seed for h.*

We define the sample complexity $m(n, \epsilon, \delta)$ as the upper bound of the number of oracle accesses by $L(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$.

Particularly we focus on the case in which the time-bound computational depth of the initial state is bounded.

Naor and Rothblum [NR06] considered the average-case setting where the initial state is drawn some efficiently computable randomized machine G and a learner knows (G, D). In this work, we also extend their result to the more general case in which a learner does not know (G, D). When (G, D) is known, then the task of learning ACDs can be regarded as learning the initial state s_0. However, when (G, D) is unknown, the task appears to be far more complex. In particular, even if the learner *knows* the initial state s_0, this does not mean that the learner can immediately simulate $D_i^{s_0}(x^1, \ldots, x^i)$.

Definition 9.6 (Universal average-case learning ACDs)**.** *Let $s(n), t(n)$ and $t'(n)$ be polynomials. We say that $t(n)$-time samplable ACDs of $s(n)$-bit initial state are learnable in polynomial time on average under $t'(n)$-time samplable distributions if there exists a randomized polynomial-time algorithm L such that for every $t'(n)$-time samplable distribution $\mathcal{G}$ over $\{0,1\}^{\leq s(n)}$ and every $t(n)$-time samplable ACD D of $s(n)$-bit initial state, every sufficiently large $n \in \mathbb{N}$, and every $\epsilon^{-1}, \delta^{-1} \in \mathbb{N}$, the algorithm L satisfies the following with probability at least $1 - \delta$ over the choice of $s_0 \sim \mathcal{G}_n$, samples from $\mathsf{EX}_{n,s_0,D}$, and randomness for L:*

1. *$L^{\mathsf{EX}_{n,s_0,D}}(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$ obtains samples $x^1, x^2, \ldots,$ from $\mathsf{EX}_{n,s_0,D}$.*
2. *After obtaining i samples $x^1, \ldots, x^i$ (where i is selected by L), $L^{\mathsf{EX}_{n,s_0,D}}(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$ outputs some circuit h as a hypothesis without additional access to $\mathsf{EX}_{n,s_0,D}$.*
3. *The hypothesis h satisfies $\mathsf{L}_1(D_i^{s_0}(x^1, \ldots, x^i), h(r)) \leq \epsilon$, where r represents a uniformly random seed for h.*

We define the sample complexity $m(n, \epsilon, \delta)$ as the upper bound of the number of oracle accesses by $L(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$.

Note that average-case distributional learning in Definition 9.4 is a special case of universal average-case learning ACDs in Definition 9.6, where an initial state s_0 is a target sampler, a generator G of ACD is a sampling algorithm for selecting the target sampler, and a sampling algorithm D of the ACD does not change the internal state (i.e., always outputs $\sigma = s_0$ for a given current state $\sigma = s_0$).

9.2.2 Main Result

Now, we show the following learnability result as an application of Theorem 9.1. Note that Item 2 and Item 3 of Theorem 2.4 correspond to Item 5 and Item 3, respectively.

Theorem 9.7. *The following are equivalent:*

1. *There is no infinitely-often one-way function.*
2. (Learning computationally shallow distributions) *For all polynomials $s(n), t(n), t'(n)$ and all $\alpha(n) = O(\log n)$, $t(n)/s(n)$-time samplable distributions (i.e., $t(n)/s(n)$-time computable samplers) of $t'(n)$-time-bounded computational depth $\alpha(n)$ are distributionally learnable in polynomial time with sample complexity $O((s(n) + n) \cdot \epsilon^{-2}\delta^{-1})$.*

3. (Universal Average-Case Distributional Learning) *For all polynomials $s(n), t(n)$, and $t'(n)$, $t(n)/s(n)$-time samplable distributions are distributionally learnable in polynomial time* on average *under (unknown) $t'(n)$-time samplable distributions with sample complexity $O((s(n)+n) \cdot \epsilon^{-2}\delta^{-1})$.*

4. (Learning ACDs of computationally shallow initial states) *For all polynomials $s(n), t(n), t'(n)$ and all $\alpha(n) = O(\log n)$, there exists a polynomial-time randomized algorithm such that $t(n)$-time samplable ACDs of $s(n)$-bit initial state whose $t'(n)$-time-bounded computational depth is at most $\alpha(n)$ are learnable in polynomial time with sample complexity $O((s(n)+n) \cdot \epsilon^{-2}\delta^{-1})$.*

5. (Universal Learning ACDs) *For all polynomials $s(n), t(n), t'(n)$, $t(n)$-time samplable ACDs of $s(n)$-bit initial state are learnable in polynomial time on average under $t'(n)$-time samplable distributions with sample complexity $O((s(n)+n) \cdot \epsilon^{-2}\delta^{-1})$.*

6. *There exists a polynomial-time randomized algorithm L that takes input $(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$ and additional meta-parameters $1^{\langle s,t,t',2^\alpha \rangle}$ such that for all functions $s(n), t(n), t'(n)$ and $\alpha(n)$, the algorithm $L(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}}; 1^{\langle s(n),t(n),t'(n),2^{\alpha(n)} \rangle})$ learns every $t(n)$-time samplable ACD of $s(n)$-bit initial state whose $t'(n)$-time-bounded computational depth is at most $d(n)$ with sample complexity $O((s(n)+n) \cdot \epsilon^{-2}\delta^{-1})$*

We remark that the implication (item 1 $\Rightarrow$ item 5) strengthens the main result of [NR06] in the sense that our learner works for unknown distributions. In addition, the dependence of parameters $\epsilon^{-2}\delta^{-1}$ in sample complexity is improved from $\epsilon^{-4}\delta^{-2}$ in [NR06].

When the learner additionally obtains the upper bound d on the description length of the ACD (and the underlying distribution of initial states in item 5), the query complexity $O((s(n)+n) \cdot \epsilon^{-2}\delta^{-1})$ can be improved to $O((s(n)+d) \cdot \epsilon^{-2}\delta^{-1})$. We will elaborate on this point after proving Theorem 9.7.

Proof of Theorem 9.7. The main part of the proof is to prove (item 1 $\Rightarrow$ item 6). First, we prove other implications.

(item 6 $\Rightarrow$ item 2) and (item 6 $\Rightarrow$ item 4) are trivial by the statements. (item 6 $\Rightarrow$ item 3) and (item 6 $\Rightarrow$ item 5) also follow from Lemma 6.14 and a basic probabilistic argument based on the union bound.

(item 2 $\Rightarrow$ item 1) follows from the observation in [KMRRSS94, Theorem 17]. An efficient distributional learner can distinguish any infinitely-often pseudorandom function $f = \{f_n : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^n\}$ from a truly random function ϕ_n by distributionally learning the distribution of $x \circ f_n(r,x)$ for $x \sim \{0,1\}^n$, because the distribution of $x \circ \phi_n(x)$ cannot be statistically approximated by polynomial-size circuits with high probability. This matches our average-case framework, as the secret seed r for f is selected uniformly at random. We can also prove (item 3 $\Rightarrow$ item 1), (item 4 $\Rightarrow$ item 1), and (item 5 $\Rightarrow$ item 1) base on the same proof because the time-bounded computational depth of the random seed r is logarithmically small with high probability.

Now, we derive (item 1 $\Rightarrow$ item 6) from Theorems 8.1 and 9.1. By the non-existence of infinitely-often one-way functions and Theorem 8.1, there exists the universal extrapolation algorithm UE. Therefore, we can apply Theorem 9.1.

We consider the trivial 1-query cheating learner $L^?_{cheat}$ that directly outputs a sample x obtained from the oracle. We apply Theorem 9.1 for $L^?_{cheat}$ and obtain a learner L' that simulates $L^?_{cheat}$ as in Theorem 9.1. Note that L' halts in polynomial time in the length of its input.

We construct the learner L that is taken meta-parameters $1^{\langle s(n),t(n),t'(n),2^{\alpha(n)}\rangle}$ and learns $t(n)$-time samplable ACDs of $s(n)$-bit initial state s_0 with $\mathrm{cd}^{t'(n)} \leq \alpha(n)$ as follows: On input $1^n, 1^{\epsilon^{-1}}$ and $1^{\delta^{-1}}$, the learner L selects $i \sim [m_0]$, where $m_0 := m_0(n, s(n), 1, \epsilon^{-1}, \delta^{-1}) = O((s(n)+n)\delta\epsilon^{-2})$ represents the sample complexity as in Theorem 9.1. Then, L obtains i samples $x^1, \dots, x^i$ from the example oracle $\mathsf{EX}_{n,s_0,D}$ and outputs a hypothesis h that takes a random seed r for executing L' as input and outputs

$$L'(x^1 \circ \dots \circ x^i, \varepsilon, \varepsilon; 1^{\langle s(n),n,\tau,2^{\alpha(n)},\delta^{-1},\epsilon^{-1}\rangle}; r),$$

where $\tau = O(m_0 t(n) + n + t'(n))$ is specified later. It is easy to verify that L halts in polynomial time in $n, \epsilon^{-1}, \delta^{-1}, s(n), t(n), t'(n)$ and $2^{\alpha(n)}$, and the query complexity is $m_0 = O((s(n)+n)\epsilon^{-2}\delta)$.

We verify that L learns all $t(n)$-time samplable ACDs D of $s(n)$-bit initial state s_0 with $\mathrm{cd}^{t'(n)} \leq \alpha(n)$. The ACD D determines a distribution family $\mathcal{D} = \{\mathcal{D}_{s_0}\}_{s_0 \in \{0,1\}^{\leq s(n)}}$, where $\mathcal{D}_{s_0}$ is a distribution of an *infinitely* long string $x^1x^2x^3\cdots$, where each x^i is the i-th sample generated by $D(1^n,\text{-})$ with initial state s_0. Then, for every $n, \epsilon^{-1}, \delta^{-1} \in \mathbb{N}$ and every initial state $s_0 \in \{0,1\}^{\leq s}$, the $m_0 n$-bit prefix of $\mathcal{D}_{s_0}$ is samplable in $\tau' = O(m_0 \cdot t(n))$ time. By taking $\tau = \max\{\tau', n, t'(n)\} = O(m_0 t(n) + n + t'(n))$, we also have $\tau \geq d(\mathcal{D})$ for any sufficiently large $n \geq d(\mathcal{D})$. By Theorem 9.1, for every sufficiently large $n \in \mathbb{N}$ with $n \geq d(\mathcal{D})$, every $\epsilon^{-1}, \delta^{-1}$ and every $z \in \{0,1\}^{\leq s(n)}$ with $(\mathrm{cd}^{\tau}(z) \leq)\mathrm{cd}^{t'}(z) \leq \alpha(n)$ (note that we select $a = 0$ and $b = n$),

$$\begin{aligned}
&\Pr_{L,x^1,\dots,x^i}\left[\mathsf{L}_1(h(r), D_i^{s_0}(x^1,\dots,x^i)) \leq \epsilon\right] \\
&\qquad = \Pr_{i,x^1,\dots,x^i}\left[\mathsf{L}_1(L'(x^1 \circ \dots \circ x^i, \varepsilon, \varepsilon; 1^{\langle s(n),n,\tau,2^{\alpha(n)},\delta^{-1},\epsilon^{-1}\rangle}), \mathsf{Label}_i^{s_0,\varepsilon}) \leq \epsilon\right] \geq 1-\delta,
\end{aligned}$$

where $i \sim [m_0]$, and $x^1, \dots, x^i$ are samples generated by $D(1^n,\text{-})$ with initial state s_0. Therefore, L' satisfies the requirements in Definition 9.6. $\square$

As mentioned before the proof, when the learner additionally obtains the upper bound d on the description length of the ACD and the underlying distribution of initial states (if any), the query complexity $O((s(n)+n)\cdot\epsilon^{-2}\delta^{-1})$ can be improved to $O((s(n)+d)\cdot\epsilon^{-2}\delta^{-1})$. This follows from observations below on the proofs of Theorems 8.1, 9.1 and 9.7:

In the proof of Theorem 8.1, we can use the time-bounded universal distribution given security parameter as an advice string for the universal Turing machine instead of the time-bounded universal distribution. It yields the universal extrapolation algorithm under the situation where the security parameter is given as a conditional string. The proof of Theorem 9.1 holds even for such a conditional case in which an advise string is given to the universal Turing machine. In the proof of Theorem 9.7, we can apply this version of universal extrapolation given a security parameter (which corresponds to 1^n in learning ACDs), and it yields the improved sample complexity $O((s(n)+d)\cdot\epsilon^{-2}\delta^{-1})$ because the whole sampling process (i.e., $\mathcal{D}$ in the proof of Theorem 9.7 and the sampler for the initial state) is described by $O(d)$ bits when 1^n is given.

9.2.3 Necessity of OWF for Nontrivial Bloom Filters with Hidden Codes

We state the necessity of one-way functions for constructing a nontrivial Bloom filter in environments where the source code of the Bloom filter is hidden from the efficient adversaries. The result follows from the work by Naor and Yogev [NY19], who first studied Bloom filters in adversarial environments. The only difference is to apply Theorem 9.7 for unknown ACDs instead of average-case learners for known ACDs [NR06].

Only in this subsection, we employ the standard sufficiently large security for one-way functions, and the corresponding learnability result (Theorem 9.7 Item 3) holds for infinitely many n and for an accuracy parameter $\epsilon = 1/p(n)$ and a confidence parameter $\delta = 1/q(n)$, where p, q is arbitrary large polynomials fixed beforehand (see Section 6.1).

First, we introduce Bloom filters in adversarial environments, introduced in [NY19].

Definition 9.8 (Bloom filter with an unsteady representation)**.** *Let $U = \{0,1\}^u$ be a universe. For $n \in \mathbb{N}$ and $\epsilon \in [0,1]$, an (n,ϵ)-Bloom filter of m-bit memory with an unsteady representation is a pair of randomized polynomial-time algorithms $B = (B_1, B_2)$ satisfying the following syntax: B_1 is given a set $S \subseteq U$ of size n and outputs a representation of a (filtering) rule $M_0 \in \{0,1\}^m$, and B_2 is given a rule M_i and query x and then outputs a new representation of a rule M_{i+1} and a response $y \in \{0,1\}$ (reject/accept) to the query x. Here, B determines an interface $\mathsf{BF}(\cdot)$ as follows: $\mathsf{BF}(\cdot)$ has an internal memory M, which is initialized with M_0. On input x, $\mathsf{BF}(\cdot)$ acts as follows:*

The interface $\mathsf{BF}(x)$**:**

1. *$(M', y) \leftarrow B_2(M, x)$.*
2. *Update the internal memory as $M := M'$.*
3. *Output y.*

We also use the notation $\mathsf{BF}(\cdot; x^1, \ldots, x^t)$ to refer to the interface $\mathsf{BF}(x)$ after performing the queries $x^1, \ldots, x^t$; i.e.,

$\mathsf{BF}(x; x^1, \ldots, x^t)$**:**

1. *For each $i = 1$ to t, execute $\mathsf{BF}(x^i)$.*
2. *Output $\mathsf{BF}(x)$.*

Then, B satisfies the following completeness and soundness for any given set S:

1. Completeness: *For any $x \in S$, any $t \in \mathbb{N}$, and any sequence of queries $x^1, \ldots, x^t$, we have $\Pr_B[\mathsf{BF}(x; x^1, \ldots, x^t) = 1] = 1$.*
2. Soundness: *For any $x \notin S$, any $t \in \mathbb{N}$, and any sequence of queries $x^1, \ldots, x^t$, we have $\Pr_B[\mathsf{BF}(x; x^1, \ldots, x^t) = 1] \leq \epsilon$.*

An instance $x \notin S$ that makes BF output 1 is called a false positive*. Note that false positives can change according to past queries.*

When the amount m the memory is larger than $O(nu)$, then $(n,0)$-Bloom filter is trivially achievable by storing the whole set S. Thus, we focus on the case in which m is not large enough to store the whole set S. Particularly when u is enough large so that $u \geq 2\log n + \log \epsilon^{-1}$, it is known that the parameters n, m and ϵ must satisfy $\epsilon \geq 2^{-m/n}$, and this is tight [CFGMW78]. Therefore, $\epsilon_0 := 2^{-m/n}$ is called *minimum error* in [NY19].

The term *unsteady representation* is based on the property that Bloom filter in the definition above can change its filtering rule adaptively during the execution. For example, in the case of filtering out spam mails, we can think the filtering rules as a representation of a white list, where the soundness requires that the probability that some spam mail x is accepted and passes the filtering soft is at most ϵ. The unsteadiness enables to update the filtering rule adaptively according the received emails.

An adversary for a Bloom filter attempts to find a false positive during the interaction with a Bloom filter. The security of Bloom filters in such an adversarial environment is formally defined by the security game as follows.

Definition 9.9 (Adversarial resiliency)**.** *Let $B = (B_1, B_2)$ be an (n, ϵ)-Bloom filter. For $q \in \mathbb{N}$, we say that B is q-adversarial resilient if for every randomized polynomial-time adversary $A = (A_1, A_2)$ and for all large enough security parameter $\lambda \in \mathbb{N}$, it holds that*

$$\Pr_{A,B}[\mathsf{Challenge}_{A,q}(\lambda) = 1] \leq \epsilon,$$

where $\mathsf{Challenge}_{A,q}(\lambda)$ is the outcome of the following process:

$\mathsf{Challenge}_{A,q}(\lambda)$**:**

1. *$S \leftarrow A_1(1^\lambda, 1^{\langle n,u \rangle})$, and if $|S| > n$, output 0.*
2. *$M_0 \leftarrow B_1(S, 1^\lambda, 1^{\langle n,u \rangle})$ and initialize BF with M_0.*
3. *$x^* \leftarrow A_2^{\mathsf{BF}}(1^\lambda, 1^{\langle n,u \rangle})$ where A_2 performs at most q adaptive queries $x^1, \dots, x^q$.*
4. *If $x^* \notin S \cup \{x^1, \dots, x^q\}$ and $\mathsf{BF}(x^*) = 1$, then output 1 (i.e., success in finding a false positive); otherwise, output 0.*

We say that an adversary A breaks the security of B with q queries if A satisfies the condition above.

Note that A_2 does not take S as input, which enhances the necessity result of one-way functions than the case in which A_2 takes S.

When the amount m of the memory is sufficiently large so that negligible ϵ can be accomplished, any polynomial-time adversary has only negligible chance in finding any false positive, and constructing adversarial resilient Bloom filters is trivial. Therefore, we focus our attention on the case of non-negligible minimum error, as in [NY19].

Definition 9.10. *We say that $m := m(n)$ is nontrivial memory complexity if $\epsilon_0 = 2^{-m(n)/n} \geq 1/\mathsf{poly}(n)$; equivalently, $m = O(n \log n)$.*

Naor and Yogev [NY19] proved the following theorem that shows the necessity of one-way functions for non-trivial Bloom filters.

Theorem 9.11 ([NY19])**.** *Let $\epsilon \in (0, 1)$, $m := m(n)$ be a nontrivial memory complexity function, and $U = \{0, 1\}^u$ be an universe for $u := u(n) = \omega(\log m\epsilon_0^3)$. If there exists no one-way function, then for any large enough $n \in \mathbb{N}$, there exists a constant C such that every (n, ϵ)-Bloom filter of m-bit memory is not q-resilient for any $q = Cm\epsilon_0^3$.*

The theorem above shows that for each (n,ϵ)-Bloom filter B, there exists an adversary A_B that wins the security game for $q = O(m\epsilon_0^3)$. By observing the proof in [NR06; NY19], the adversary A_B heavily relies on B; so it remained the possibility that we can construct a Bloom filter in their setting *without* one-way function by hiding B from adversaries. As a corollary to Theorem 9.7, we exclude this possibility by generalizing the adversary to the universal one that works for every (n,ϵ)-Bloom filter B that has small computational depth.

Corollary 9.12. *Let $\epsilon \in (0,1)$, $m := m(n)$ be a nontrivial memory complexity function, and $U = \{0,1\}^u$ be an universe for $u := u(n) = \omega(\log m\epsilon_0^3)$. Let $\tau(n,\lambda)$ be an arbitrary polynomial. If there exists no one-way function, then there exists a polynomial-time adversary A such that for every $t,d,\alpha \in \mathbb{N}$ and for every large enough $n \in \mathbb{N}$, the adversary $A(;1^{\langle d,t,2^\alpha\rangle})$ breaks all $\tau(n,\lambda)$-time computable (n,ϵ)-Bloom filter B of m-bit memory, description size at most d, and t-time-bounded computational depth at most α with $q = O((m+d)\epsilon_0^3)$ queries, where λ is a security parameter.*

The informal statement of Corollary 2.2 follows from the fact that any source code B generated in time t only has t'-time-bounded computational depth at most $\log t + O(1)$ for $t' = \mathsf{poly}(t)$ based on Theorem 6.13.

Proof sketch. The proof follows [NY19]; so here we briefly review their proof and highlight how the universality of learning ACDs are inherited in the security game for breaking the security of Bloom filters.

First, we describe the adversary $A = (A_1, A_2)$ for a Bloom filter $B = (B_1, B_2)$. At the initial step A_1 selects a uniformly random subset S of size $n \in \mathbb{N}$. Then we consider the following learning ACD problem:

1. $B_1(S, 1^\lambda, 1^{\langle n,u\rangle})$ outputs an initial filtering rule M_0, and then we regard M_0 as the initial state in learning ACDs.

2. A_2 executes a learner L for ACDs with an accuracy parameter $\epsilon = \Theta(\epsilon_0)$ and a sufficiently small constant confidence parameter δ, where we regard the outcome of $k = \Theta(\epsilon_0^{-1})$ consecutive transactions $(x^1,\dots,x^k,b_1,\dots,b^k)$, where $x^i \sim \{0,1\}^u$ and $b^i \leftarrow \mathsf{BF}(x^i;x^1,\dots,x^{i-1})$ for each i, as one sample for each step.

3. If L halts and outputs a hypothesis h at some step, then A_2 selects $x^1,\dots,x^k \sim \{0,1\}^u$ and executes h repeatedly $\ell = \Theta(\epsilon_0^{-1})$ times to obtain samples $b^{1,1},\dots,b^{k,1},\dots,b^{1,\ell},\dots,b^{k,\ell} \in \{0,1\}$ under the condition that the first half sample of h is $x^1,\dots,x^k$.[15]

4. If there exists $i \in [k]$ such that $b^{i,j} = 1$ for all $j \in [\ell]$, then A_2 queries $x^1,\dots,x^{i-1}$ to BF in this order, and outputs $x^* := x^i$ as a false positive.

As a crucial lemma, Naor and Yogev [NY19] showed that the outcome x^* of the above adversary A satisfies $\mathsf{BF}(x^*) = 1$. When the size of universe is large enough than the query complexity of L and $\log n$, this must not be queried previously and not be in S with high probability; thus, x^* is a false positive with high probability.

Now, we use the universal learner for ACDs with sample complexity $O((d+s)\epsilon^2\delta^{-1}) = O((d+s)\epsilon_0^2)$ as L above. Note that if the description of a Bloom filter has t-time-bounded computational

[15]Technically, we usually need distributional inverting for conditional sampling from the hypothesis h [cf. Xia10], and this point seems not to be discussed explicitly in [NY19]. In our case, we can obtain the hypothesis that statistically simulates the conditional distribution under the conditional string directly from Theorem 9.1.

depth at most α, then with probability $1-\delta'$ (where $\delta' > 0$ is an arbitrarily small constant) over the choice of S, an initial filtering rule M_0 has t'-time-bounded computational depth at most $\alpha + \log t' + O(1)$ for $t' = \mathsf{poly}(\tau(n,\lambda))$ by Lemma 6.15 (where the constant term only depends on the universal Turing machine). Therefore, A above works for such a Bloom filter with infinitely many λ for every sufficiently large $n \in \mathbb{N}$. For each sample in learning ACDs, we need to invoke BF repeatedly k times. Therefore, we obtain the universal adversary with query complexity $q = k \cdot O((d+m)\epsilon_0^2) = O((d+m)\epsilon_0^3)$ and time complexity $\mathsf{poly}(q, u, 2^\alpha, t') = \mathsf{poly}(2^\alpha, \lambda, n, m, d, u)$. $\square$

9.3 Universal Top-k Prediction

In this section, we consider a natural task of predicting the next outcome by producing the top k most likely candidates with the estimated likelihood for a given $k \in \mathbb{N}$, e.g., {(sunny, 0.8), (cloudy, 0.15), (rainy, 0.02)} in the weather forecast when $k = 3$. We show that this learning task is feasible on average under the non-existence of OWF.

Corollary 9.13 (Universal top-k prediction)**.** *If there is no infinitely-often one-way function, then there exist a polynomial-time randomized algorithm L and a polynomial m_L such that for every $t_D(|z|)$-time samplable family $\mathcal{D} = \{\mathcal{D}_z\}_{z\in\{0,1\}^*}$, where each $\mathcal{D}_z$ is over binary strings, every sufficiently large $s \in \mathbb{N}$, every $z \in \{0,1\}^s$, and every $a, b, k, m, t, \epsilon^{-1}, \delta^{-1}, \lambda^{-1}, \alpha \in \mathbb{N}$ with $t \geq t_D(s)$, $k \leq 2^b$, $\mathrm{cd}^t(z) \leq \alpha$, and $m \geq m_L(s, b, \epsilon^{-1}, \delta^{-1}, \lambda^{-1})$, the following holds with probability at least $1-\delta$ over the choice of $i \sim [m]$ and $x^{<i} \sim \mathcal{D}_z^{<i}$. With probability at least $1-\epsilon$ over the choice of $x^i \sim \mathcal{D}_z^{i,x^{<i}}$ and the randomness for L, the learner L satisfies the following:*

- *$L(x^{<i}, x^i, 1^k; 1^{\langle s,b,t,2^\alpha,\epsilon^{-1},\delta^{-1},\lambda^{-1}\rangle})$ outputs $(y_1, p_1), \ldots, (y_k, p_k) \in \{0,1\}^b \times [0,1]$.*
- *Let $P = \{p_1^*, \ldots, p_{2^b}^*\}$ be an ordered multi-set defined as $P = \{\mathsf{Label}_i^{z,x^i}(y) : y \in \{0,1\}^b\}$ and $p_j^* \geq p_{j+1}^*$ for every $j \in [2^b - 1]$ (i.e., P is a ranking of probabilities of the next labels). Then, for each $j \in [k]$,*
$$\left|p_j - p_j^*\right| \leq \lambda \text{ and } \left|p_j - \mathsf{Label}_i^{z,x^i}(y_j)\right| \leq \lambda.$$

Furthermore, $m_L(s, b, \epsilon^{-1}, \delta^{-1}, \lambda^{-1}) = O(sb\lambda^{-2}\epsilon^{-3}(\log\epsilon^{-1})\delta^{-1})$.

Particularly, top-1 prediction corresponds to agnostic learning for 0-1 loss, where the sample complexity is worse than the result in Section 10.

Corollary 9.14. *If there is no infinitely-often one-way function, then there exist a polynomial-time randomized algorithm L and a polynomial m_L such that for every $t_D(|z|)$-time samplable family $\mathcal{D} = \{\mathcal{D}_z\}_{z\in\{0,1\}^*}$, where each $\mathcal{D}_z$ is over samples in $\{0,1\}^n \times \{0,1\}^b$ for $n := \mathsf{poly}(|z|)$ and $b := \mathsf{poly}(|z|)$, every sufficiently large $s \in \mathbb{N}$, every $z \in \{0,1\}^s$, and every $m, t, \epsilon^{-1}, \delta^{-1}, \alpha \in \mathbb{N}$ with $m \geq m_L(s, b, \epsilon^{-1}, \delta^{-1})$, $t \geq m \cdot t_D(s)$, and $\mathrm{cd}^t(z) \leq \alpha$,*

$$\Pr\left[\Pr_{(x,y)\sim\mathcal{D}_z}[L(x^1, y^1, \ldots, x^i, y^i, x; 1^{\langle s,t,2^\alpha,\epsilon^{-1},\delta^{-1}\rangle}) = y] \leq \min_{f:\{0,1\}^n\to\{0,1\}^b} \Pr_{(x,y)\sim\mathcal{D}_z}[f(x) = y] + \epsilon\right] \geq 1-\delta,$$

where the outer probability is taken over $i \sim [m]$, $(x^1, y^1), \ldots, (x^i, y^i) \sim \mathcal{D}_z$. Furthermore, $m_L(s, b, \epsilon^{-1}, \delta^{-1}) = O(sb\epsilon^{-5}(\log\epsilon^{-1})\delta^{-1})$.

Proof of Corollary 9.13. We construct a cheating learner $L^{?}_{cheat}$ such that for every $b, k, \lambda^{-1}, \epsilon^{-1} \in \mathbb{N}$ with $k \leq 2^b$ and every distribution Label over $\{0,1\}^b$, the learner $L^{\mathsf{Label}}_{cheat}$ satisfies the following with probability at least $1 - \epsilon/2$ over the choice of samples according to Label:

- $L^{\mathsf{Label}}_{cheat}(1^k, 1^{\langle b, \lambda^{-1}, \epsilon^{-1}\rangle})$ outputs $(y_1, p_1), \ldots, (y_k, p_k) \in \{0,1\}^b \times [0,1]$.
- Let $P = \{p^*_1, \ldots, p^*_{2^b}\}$ be an ordered multi-set defined as $P = \{\mathsf{Label}(y) : y \in \{0,1\}^b\}$ and $p^*_j \geq p^*_{j+1}$ for every $j \in [2^b - 1]$. Then, for each $j \in [k]$,
$$\left|p_j - p^*_j\right| \leq \lambda \text{ and } |p_j - \mathsf{Label}(y_j)| \leq \lambda.$$

Furthermore, the query complexity of L_{cheat} is at most $q = O(\lambda^{-2} b \log \epsilon^{-1})$.

According to Theorem 9.1, we obtain an algorithm L' that simulates L_{cheat} with UE. The learner L is defined as

$$L(x^{<i}, x^i, 1^k; 1^{\langle s,b,t,2^\alpha,\epsilon^{-1},\delta^{-1},\lambda^{-1}\rangle}) = L'(x^{<i}, x^i, 1^k, 1^{\langle b,\lambda^{-1},\epsilon^{-1}\rangle}; 1^{\langle s,b,t,2^\alpha,4\epsilon^{-1}\delta^{-1},4\epsilon^{-1}\rangle}),$$

and the sample complexity function is

$$m_L(s, b, \epsilon^{-1}, \delta^{-1}, \lambda^{-1}) = O(sq\epsilon^{-1}\delta^{-1}\epsilon^{-2}) = O(sb\lambda^{-2}\epsilon^{-3}(\log \epsilon^{-1})\delta^{-1})$$

for sufficiently large $s \geq d(\mathcal{D})$.

The correctness of L is verified as follows. Without loss of generality, we assume that $t_D(s) \geq s$. For every sufficiently large $s \geq d(\mathcal{D})$, we have $t \geq t_D(s) \geq d(\mathcal{D})$. Therefore, by Theorem 9.1, for every sufficiently large $s \in \mathbb{N}$ and every $a, b, k, m, t, \epsilon^{-1}, \delta^{-1}, \lambda^{-1}, \alpha \in \mathbb{N}$ satisfying the assumptions, with probability at least $1 - \delta\epsilon/4$ over the choice of $i \sim [m], x^{<i} \sim \mathcal{D}^{<i}_z, x^i \sim \mathcal{D}^{i,x^{<i}}_z$, with probability at least $1 - \epsilon/2 - \epsilon/4 = 1 - 3\epsilon/4$ over the choice of randomness for L', the output $(y_1, p_1), \ldots, (y_k, p_k)$ of L' (i.e., L) satisfies the same property as L_{cheat}. By the simple probabilistic argument (i.e., Markov's inequality and the Union bound) with probability at least $1 - \delta$ over the choice of $i \sim [m], x^{<i} \sim \mathcal{D}^{<i}_z, x^i \sim \mathcal{D}^{i,x^{<i}}_z$, and with probability at least $1 - 3\epsilon/4 - \epsilon/4 = 1 - \epsilon$ over the choice of $x^i \sim \mathcal{D}^{i,x^{<i}}_z$ and randomness for L, the same event occurs.

The remainder of the proof involves the construction of L_{cheat}, which follows from the standard empirical estimation. On input $1^k, 1^{\langle b,\lambda^{-1},\epsilon^{-1}\rangle}$, the learner L_{cheat} obtains $q := 8\lambda^{-2} b \ln 4\epsilon^{-1}$ samples $z_1, \ldots, z_q \in \{0,1\}^b$ from Label and counts $m_y := |\{i \in [q] : z^i = y\}|$ for every $y \in \{0,1\}^b$. Let $\tilde{y}_1, \ldots, \tilde{y}_{2^b} \in \{0,1\}^b$ be the ordering of $\{0,1\}^b$ based on the largeness of m_y, i.e., $m_{\tilde{y}_j} \geq m_{\tilde{y}_{j+1}}$ for each $j \in [2^b - 1]$. Then, L_{cheat} outputs k pairs $(\tilde{y}_1, m_{\tilde{y}_1}/q), \ldots, (\tilde{y}_k, m_{\tilde{y}_k}/q)$.

We verify the correctness of L_{cheat}. For each $j \in [2^b]$, let $p_j = m_{\tilde{y}_j}/q$. Note that $p_1 \geq p_2 \geq \cdots \geq p_{2^b}$. For each $y \in \{0,1\}^b$, by Hoeffding's inequality, it holds that $m_y/q \in [\mathsf{Label}(y) \pm \lambda/4]$ with probability at least $1 - 2e^{-2q(\lambda/4)^2} \geq 1 - e^{-b} \cdot \epsilon/2 \geq 1 - 2^{-b} \cdot \epsilon/2$. By the union bound, every $y \in \{0,1\}^b$ satisfies $m_y/q \in [\mathsf{Label}(y) \pm \lambda/4]$ with probability at least $1 - \epsilon/2$. We assume that this event occurs. Then, for every $j \in [k]$, it trivially holds that $|p_j - \mathsf{Label}(\tilde{y}_j)| \leq \lambda/4 \leq \lambda$. We also show that $|p_j - p^*_j| \leq \lambda$ (this indeed holds for any $j \in [2^b]$) as follows: Let $y^*_1, \ldots, y^*_{2^b}$ be the ordering of $\{0,1\}^b$ such that $p^*_j = \mathsf{Label}(y^*_j)$ for each j, where we break ties arbitrarily. Let $\ell \in [2^b]$ be an index such that $\tilde{y}_\ell = y^*_1$. Then, for any $j < \ell$, it holds that $\mathsf{Label}(y^*_1) - \mathsf{Label}(\tilde{y}_j) \leq \lambda/2$; otherwise,

$$p_\ell \geq \mathsf{Label}(\tilde{y}_\ell) - \lambda/4 = \mathsf{Label}(y^*_1) - \lambda/4 > \mathsf{Label}(\tilde{y}_j) + \lambda/2 - \lambda/4 \geq (p_j - \lambda/4) + \lambda/4 = p_j.$$

This implies that $|p_j - p_j^*| \le \lambda$ for every $j \le \ell$ because (i) there are at least ℓ elements (including y_1^*) whose outcome probability according to Label is at least $\mathsf{Label}(y_1^*) - \lambda/2$; (ii) thus, $p_j^* \ge \mathsf{Label}(y_1^*) - \lambda/2$ for every $j \le \ell$, and (iii) it holds that, for every $j \le \ell$,

$$p_j \ge p_\ell \ge \mathsf{Label}(\tilde{y}_\ell) - \lambda/4 = \mathsf{Label}(y_1^*) - \lambda/4 \ge \mathsf{Label}(y_j^*) - \lambda/4 = p_j^* - \lambda/4 \ge p_j^* - \lambda$$

and

$$p_j \le \mathsf{Label}(\tilde{y}_j) + \lambda/4 \le \mathsf{Label}(y_1^*) + \lambda/4 \le p_j^* + \lambda/2 + \lambda/4 \le p_j^* + \lambda.$$

Next, let $\ell'(> \ell)$ be an index such that $\mathsf{Label}(\tilde{y}_{\ell'}) = \max_{j>\ell} \mathsf{Label}(\tilde{y}_j)$. By the same argument as above, we can show that $|p_j - p_j^*| \le \lambda$ for every $j \in \mathbb{N}$ with $\ell + 1 \le j \le \ell'$. We continue this argument until we run out of elements and obtain $|p_j - p_j^*| \le \lambda$ for every $j \in [2^b]$. $\square$

Proof of Corollary 9.14 (sketch). Let L' be the learner in Corollary 9.13. The agnostic learner L given a sample set $X := \big((x^1, y^1), \ldots, (x^{i-1}, y^{i-1}), x\big)$ and a parameter $1^{\langle s,t,2^\alpha,\epsilon^{-1},\delta^{-1}\rangle}$ executes

$$L'(X, 1^k; 1^{\langle s,b,Ct,2^\alpha,2\epsilon^{-1},\delta^{-1},2\epsilon^{-1}\rangle})$$

for $k = 1$ (i.e., top-1 prediction) and some absolute constant C. It it easy to verify that the required sample complexity is $m = O(sb\epsilon^{-5}(\log \epsilon^{-1})\delta^{-1})$ by Corollary 9.13.

For every $t_D(|z|)$-time samplable distribution $\mathcal{D} = \{\mathcal{D}_z\}$, a distribution of m samples drawn from $\mathcal{D}_z$ is $(C \cdot m \cdot t_D(|z|))$-time samplable by selecting a large enough constant C. Thus, for every $t \ge m \cdot t_D(|z|)$, the learner L' is executed validly and satisfies the following with probability at least $1-\delta$ over the choice of i and $(x^1, y^1), \ldots, (x^{i-1}, y^{i-1})$: with probability $1-\epsilon/2$ over the choice of an example x, the learner L' finds a label $\tilde{y}$ whose outcome probability is the same as the maximum one within additive error $\epsilon/2$. The probability that $\tilde{y}$ corresponds to the actual next label is equal to the optimal probability (by the most frequent label) within additive error $\epsilon/2$. Therefore, the probability that L' correctly predicts the next label is equal to the optimal probability within $1 \cdot \epsilon/2 + \epsilon/2 \cdot 1 = \epsilon$ in expectation over the choice of a sample (x, b). $\square$

9.4 Universal Likelihood Estimation

In this section, we consider a natural task of estimating the probability that a given label is observed as the next outcome within an additive error, e.g., the probability of "rainy" in the weather forecast. We show that this learning task is feasible on average under the non-existence of OWF.

Corollary 9.15 (Universal likelihood estimation)**.** *If there is no infinitely-often one-way function, then there exist a polynomial-time randomized algorithm L and a polynomial m_L such that for every $t_D(|z|)$-time samplable family $\mathcal{D} = \{\mathcal{D}_z\}_{z\in\{0,1\}^*}$, where each $\mathcal{D}_z$ is over binary strings, for all sufficiently large $s \in \mathbb{N}$, all $z \in \{0,1\}^s$, all $a, b, k, m, t, \epsilon^{-1}, \delta^{-1}, \lambda^{-1} \in \mathbb{N}$ with $t \ge t_D(s)$, $\mathrm{cd}^t(z) \le \alpha$ and $m \ge m_L(s, k, \epsilon^{-1}, \delta^{-1}, \lambda^{-1})$, and all $y_1, \ldots, y_k \in \{0,1\}^b$, the following holds with probability at least $1-\delta$ over the choice of $i \sim [m], x^{<i} \sim \mathcal{D}_z^{<i}$. With probability at least $1-\epsilon$ over the choice of $x^i \sim \mathcal{D}_z^{i,x^{<i}}$ and randomness for L, the learner L satisfies the following:*

$L(x^{<i}, x^i, y_1, \ldots, y_k; 1^{\langle s,t,2^\alpha,\epsilon^{-1},\delta^{-1},\lambda^{-1}\rangle})$ outputs $p_1 \ldots, p_k \in [0,1]$ satisfying, for each $j \in [k]$,

$$\left| p_j - \mathsf{Label}_i^{z,x^i}(y_i) \right| \le \lambda.$$

Furthermore, $m_L(s, k, \epsilon^{-1}, \delta^{-1}, \lambda^{-1}) = O(s\lambda^{-2}\epsilon^{-3}(\log \epsilon^{-1})\delta^{-1} \log k)$.

Proof. The outline of the proof is similar to that for the proof of Corollary 9.13. First, we construct a cheating learner $L^{?}_{cheat}$ such that for every $b, k, \lambda^{-1} \in \mathbb{N}$, every $y_1, \dots, y_k \in \{0,1\}^b$, and every distribution Label over $\{0,1\}^b$, the learner $L^{\mathsf{Label}}_{cheat}$ satisfies the following with probability at least $1 - \epsilon/2$ over the choice of samples drawn from Label:

> $L^{\mathsf{Label}}_{cheat}(y_1, \dots, y_k, 1^{\langle \lambda^{-1}, \epsilon^{-1}\rangle})$ outputs $p_1 \dots, p_k \in [0,1]$ satisfying $|p_j - \mathsf{Label}(y_i)| \le \lambda$ for each $j \in [k]$.

Furthermore, the query complexity of L_{cheat} is at most $q = O(\lambda^{-2} \log(k\epsilon^{-1}))$.

Then, by Theorem 9.1, we obtain a learner L' that simulates L_{cheat} by UE and construct the learner L defined as

$$L(x^{<i}, x^i, y_1, \dots, y_k; 1^{\langle s,t,2^\alpha,\epsilon^{-1},\delta^{-1},\lambda^{-1}\rangle}) = L'(x^{<i}, x^i, y_1, \dots, y_k, 1^{\langle \lambda^{-1},\epsilon^{-1}\rangle}; 1^{\langle s,b,t,2^\alpha,4\epsilon^{-1}\delta^{-1},4\epsilon^{-1}\rangle}).$$

The sample complexity function is

$$m_L(s, k, \epsilon^{-1}, \delta^{-1}, \lambda^{-1}) = O(sq\epsilon^{-1}\delta^{-1}\epsilon^{-2}) = O(s\lambda^{-2}\epsilon^{-3}(\log \epsilon^{-1})\delta^{-1} \log k)$$

for every sufficiently large $s \ge d(\mathcal{D})$. The correctness of L is verified in the same way as Corollary 9.13. Thus, we only present the construction of L_{cheat}.

The cheating learner L_{cheat} is constructed according to the standard empirical estimation. On input $y_1, \dots, y_k$ and $\langle \lambda^{-1}, \epsilon^{-1}\rangle$, the learner L_{cheat} obtains $q := 2^{-1}\lambda^{-2}\ln(4k\epsilon^{-1})$ samples $z_1, \dots, z_q \in \{0,1\}^b$ from Label and counts $m_j := |\{i \in [q] : z_i = y_j\}|$ for each $j \in [k]$. Then, L_{cheat} outputs $p_j = m_j/q$ for each $j \in [k]$.

The correctness of L_{cheat} is verified as follows. By Hoeffding's inequality, it holds that $m_j/q \in [\mathsf{Label}(y_j) \pm \lambda]$ with probability at least $1 - 2e^{-2q\lambda^2} \ge 1 - \epsilon/(2k)$. By the union bound, it holds that $p_j = m_j/q \in [\mathsf{Label}(y_j) \pm \lambda]$ for all $j \in [k]$ with probability at least $1 - \epsilon/2$. □

10 Agnostic Learning in Pessiland

In this section, we focus on minimizing the expected loss in the online learning framework presented in Section 9. We show that if the value of the loss function is bounded above by $c > 0$, then we can obtain the lower bound of the required number of stages as a function in c instead of the number of queries. As an application, we obtain a universal agnostic learner having better sample complexity than Corollary 9.14 obtained from Theorem 9.1.

As in Section 9, we use $a \in \mathbb{N} \cup \{0\}$ and $b, m \in \mathbb{N}$ to refer to the size of advice string, the size of each label, and the total number of stages (i.e., sample complexity), respectively. The learning framework and notations are identical to those presented in Section 9, and the only differences are the following: (i) a learner is given a past stream $x^{<i}$ and advice information x^i and selects an action $\alpha_{x^{<i},x_i}$ from a set $\mathcal{A}$ of actions and (ii) the goal of the learner is to minimize the expected loss with respect to a loss function $l \colon \mathcal{A} \times \{0,1\}^b \to \mathbb{R}_{\ge 0}$; i.e., the learner attempts to minimize

$$\mathop{\mathbb{E}}_{x^i,y^i}[l(\alpha_{x^{<i},x_i}, y^i)],$$

where $x^i \sim \mathcal{D}_z^{i,x^{<i}}$ and $y^i \sim \mathsf{Label}_i^{z,x^i}$.

First, we present the meta-theorem for minimizing the expected loss, which yields better sample complexity when the cheating learner requires polynomially many queries for minimizing the expected loss with respect to a loss function bounded above by a small value.

Definition 10.1 (Action set and bounded loss function)**.** *An action set $\mathcal{A} = \{\mathcal{A}_{w,b}\}_{w \in \{0,1\}^*, b \in \mathbb{N}}$ is defined as a family of subsets, where $\mathcal{A}_{w,b} \subseteq \{0,1\}^*$. For a function $c\colon \{0,1\}^* \times \mathbb{N} \to \mathbb{R}_{\geq 0}$, a loss function $l\colon \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}_{\geq 0}$ is said to be c-bounded (with respect to $\mathcal{A}$) if for every $w \in \{0,1\}^*$, every $b \in \mathbb{N}$, every $\alpha \in \mathcal{A}_{w,b}$, and every $y \in \{0,1\}^{\leq b}$, it holds that $l(\alpha, y) \leq c(w,b)$.*

Theorem 10.2. *Let $\mathcal{A} = \{\mathcal{A}_{w,b}\}_{w \in \{0,1\}^*, b \in \mathbb{N}}$ be an action set, and let $l\colon \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}_{\geq 0}$ be a c-bounded loss function (with respect to $\mathcal{A}$) for a polynomial-time computable function $c\colon \{0,1\}^* \times \mathbb{N} \to \mathbb{R}_{\geq 0}$.*

Suppose that UE *in Theorem 8.1 exists. Then, for every oracle machine (cheating learner) $L^?_{cheat}$ that outputs an action in a set $\mathcal{A}_{w,b}$ with polynomial-time computable query complexity $q(w)$ (where w denotes an input for $L^?_{cheat}$, and b denotes the length of each label), there exist a polynomial m_0 and a randomized algorithm L that outputs an action in the same set $\mathcal{A}_{w,b}$ satisfying the following: for every $t_D(|z|)$-time samplable family $\mathcal{D} = \{\mathcal{D}_z\}_{z \in \{0,1\}^*}$, where each $\mathcal{D}_z$ is over binary strings, every $s, a, b, t, \epsilon^{-1}, \delta^{-1}, \alpha \in \mathbb{N}$ with $t \geq \max\{d(\mathcal{D}), t_D(s)\}$, every $z \in \{0,1\}^*$ with $\mathrm{cd}^t(z) \leq \alpha$, every auxiliary input $w \in \{0,1\}^*$, and every $m \geq m_0(d(\mathcal{D}), s, c(w,b), \epsilon^{-1}, \delta^{-1})$,*

$$\Pr_{i, x^{<i}}\left[\mathop{\mathbb{E}}_{x^i, y^i, L}\left[l(L(x^{<i}, x^i, w; 1^{\langle s,b,t,2^\alpha,\epsilon^{-1},\delta^{-1}\rangle}), y^i)\right] \leq \mathop{\mathbb{E}}_{x^i}\left[\min_{\alpha \in \mathcal{A}_{w,b}} \mathop{\mathbb{E}}_{y^i}[l(\alpha, y^i)]\right] + 2\Delta_{L_{cheat}}(w,b) + \epsilon\right] \geq 1-\delta,$$

where $i \sim [m]$, $x^{<i} \sim \mathcal{D}_z^{<i}$, $x^i \sim \mathcal{D}_z^{i,x^{<i}}$, $y^i \sim \mathsf{Label}_i^{z,x^i}$, and

$$\Delta_{L_{cheat}}(w,b) := \sup_{\mathcal{O}: \text{distribution over } \{0,1\}^{\leq b}} \left(\mathop{\mathbb{E}}_{\mathcal{O}, y \sim \mathcal{O}}[l(L^{\mathcal{O}}_{cheat}(w), y)] - \min_{\alpha \in \mathcal{A}_{w,b}} \mathop{\mathbb{E}}_{y \sim \mathcal{O}}[l(\alpha, y)]\right).$$

Furthermore, $m_0(d(\mathcal{D}), s, c, \epsilon^{-1}, \delta^{-1}) = O((s + d(\mathcal{D})) \cdot c^2 \cdot \epsilon^{-2}\delta^{-2})$, and L halts in polynomial time in the input length and the running time of L_{cheat}.

The proof is given in Section 10.1. In Section 10.2, we apply the meta-theorem for the 0-1 loss function and obtain Theorem 2.3. In Section 10.3, we consider agnostic learning with respect to general loss functions in the case where the number of labels is polynomially bounded.

10.1 Time-Bounded Universal Prediction

First, we show the following key lemma.

Lemma 10.3. *Let $b \in \mathbb{N}$. Let $\mathcal{A} \subseteq \{0,1\}^*$, and let $l\colon \mathcal{A} \times \{0,1\}^{\leq b} \to \mathbb{R}_{\geq 0}$ be a loss function satisfying that there exists $C > 0$ such that $l(\alpha, y) \leq C$ for every $\alpha \in \mathcal{A}$ and $y \in \{0,1\}^{\leq b}$.*

For every distribution $\mathcal{D}$ on $\{0,1\}^*$ such that $\mathcal{D}$ has a t_D-time sampler described by d bits, and for every $t, a, b, m \in \mathbb{N}$ with $t \geq \tau_{\mathrm{dom}}(d, t_D)$,

$$\mathop{\mathbb{E}}_{i, x^{<i}, x^i}\left[\sum_{y \in \{0,1\}^{\leq b}} \left|\mathsf{Next}_b(\mathcal{D}, x^{<i}x^i)(y) - \mathsf{Next}_b(\mathrm{Q}^t, x^{<i}x^i)(y)\right| \cdot \max_{\alpha \in \mathcal{A}} l(\alpha, y)\right] \leq C \cdot \sqrt{\frac{O(d)}{m}},$$

and for every oracle machine $I^?$ that outputs a string in $\mathcal{A}$,

$$\mathop{\mathbb{E}}_{i,x^{<i},x^i}\left[\left|\mathop{\mathbb{E}}_{y\sim\mathsf{Next}_b(\mathrm{Q}^t,x^{<i}x^i)}[l(I^{\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)},y)]-\mathop{\mathbb{E}}_{y\sim\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)}[l(I^{\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)},y)]\right|\right]\leq C\cdot\sqrt{\frac{O(d)}{m}},$$

where $i\sim[m], x^{<i}\sim\mathcal{D}^{<i}, x^i\sim\mathcal{D}^{i,x^{<i}}$, and the hidden constant in $O(d)$ depends on only the universal Turing machine.

Proof. The first claim is verified as follows:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{i,x^{<i},x^i}\left[\sum_{y\in\{0,1\}^{\leq b}}\left|\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)(y)-\mathsf{Next}_b(\mathrm{Q}^t,x^{<i}x^i)(y)\right|\cdot\max_{\alpha\in\mathcal{A}}l(\alpha,y)\right]\\
&\leq C\cdot\mathop{\mathbb{E}}_{i,x^{<i},x^i}\left[\sum_{y\in\{0,1\}^{\leq b}}\left|\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)(y)-\mathsf{Next}_b(\mathrm{Q}^t,x^{<i}x^i)(y)\right|\right]\\
&=2C\cdot\mathop{\mathbb{E}}_{i,x^{<i},x^i}\left[\mathsf{L}_1\left(\mathsf{Next}_b(\mathcal{D},x^{<i}x^i),\mathsf{Next}_b(\mathrm{Q}^t,x^{<i}x^i)\right)\right]\\
&\leq 2C\cdot\mathop{\mathbb{E}}_{i,x^{<i},x^i}\left[\sqrt{2^{-1}\cdot\mathrm{KL}\left(\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)||\mathsf{Next}_b(\mathrm{Q}^t,x^{<i}x^i)\right)}\right]\\
&\leq\sqrt{2}C\cdot\sqrt{\mathop{\mathbb{E}}_{i,x^{<i},x^i}\left[\mathrm{KL}\left(\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)||\mathsf{Next}_b(\mathrm{Q}^t,x^{<i}x^i)\right)\right]}\\
&\leq\sqrt{2}C\cdot\sqrt{\frac{O(d)}{m}}=C\cdot\sqrt{\frac{O(d)}{m}},
\end{aligned}$$

where the first inequality holds by $l(\alpha,y)\leq C$ for every $\alpha\in\mathcal{A}$ and $y\in\{0,1\}^{\leq b}$, the second inequality follows from Pinsker's inequality (Fact 6.1), the third inequality follows from Jensen's inequality, and the last inequality follows from Lemma 9.2 for a trivial 1-query algorithm I that outputs a sample obtained from the oracle.

The second claim is verified as follows:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{i,x^{<i},x^i}\left[\left|\mathop{\mathbb{E}}_{y\sim\mathsf{Next}_b(\mathrm{Q}^t,x^{<i}x^i)}[l(I^{\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)},y)]-\mathop{\mathbb{E}}_{y\sim\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)}[l(I^{\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)},y)]\right|\right]\\
&=\mathop{\mathbb{E}}_{i,x^{<i},x^i}\left[\left|\sum_{y\in\{0,1\}^{\leq b}}\left(\mathsf{Next}_b(\mathrm{Q}^t,x^{<i}x^i)(y)-\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)(y)\right)\mathbb{E}[l(I^{\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)},y)]\right|\right]\\
&\leq\mathop{\mathbb{E}}_{i,x^{<i},x^i}\left[\sum_{y\in\{0,1\}^{\leq b}}\left|\mathsf{Next}_b(\mathrm{Q}^t,x^{<i}x^i)(y)-\mathsf{Next}_b(\mathcal{D},x^{<i}x^i)(y)\right|\cdot\max_{\alpha\in\mathcal{A}}l(\alpha,y)\right]\\
&\leq C\cdot\sqrt{\frac{O(d)}{m}},
\end{aligned}$$

where the last inequality follows from the first claim. □

Now, we derive Theorem 10.2 from Theorem 8.1 and Lemma 10.3, which is a time-bounded variant of the theory of universal prediction presented in [MF98].

Proof of Theorem 10.2. Let UE be the universal extrapolation algorithm in Theorem 8.1. Let τ and τ' be the polynomials shown in Lemma 6.15. We consider a $t_D(|z|)$-time samplable distribution family $\mathcal{D} = \{\mathcal{D}_z\}_{z\in\mathbb{N}}$.

As the proof of Theorem 9.1, we construct a learner L that executes L_{cheat}, where the query access to Label is simulated by UE. On input $x^{<i}, x^i, w, 1^{\langle s,b,t,2^\alpha,\epsilon^{-1},\delta^{-1}\rangle}$, the learner L executes the cheating learner $L^{?}_{cheat}(w)$, where L answers each query to Label_i^{z,x^i} by

$$\mathsf{ans} \leftarrow \mathsf{UE}(x^{<i}x^i; 1^{\langle b,t',\epsilon'^{-1},2^{\alpha'}\rangle})$$

for $t' = \max\{\tau(\tau'(t,t)), \tau_1(t,s,t)\}$, $C := c(w,b)$, $\epsilon' = \epsilon/(4Cq(w))$, and $\alpha' = \alpha\log(8C\epsilon^{-1}\delta^{-1}) + 2\log t'$, where L uses fresh randomness for each query access, and τ_1 is a polynomial specified later. It is easy to verify that L halts in polynomial time in the input length and the running time of L_{cheat}. Below, we show the correctness of L. For readability, we omit the parameters for L and UE.

For the correctness, we evaluate the following expectation for $i \sim [m], x^{<i} \sim \mathcal{D}_z^{<i}$:

$$\mathop{\mathbb{E}}_{x^i,y^i,L}\left[l(L(x^{<i},x^i,w),y^i)\right],$$

where $x^i \sim \mathcal{D}_z^{i,x^{<i}}$, $y^i \sim \mathsf{Label}_i^{z,x^i}$.

For every $z, i, x^{<i}$, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x^i,y^i,L}\left[l(L(x^{<i},x^i,w),y^i)\right] \\
&\quad= \mathop{\mathbb{E}}_{x^i,L}\left[\sum_{y^i\in\{0,1\}^{\le b}} \mathsf{Next}_b(\mathcal{D}_z, x^{<i}x^i)(y^i)\cdot l(L(x^{<i},x^i,w),y^i)\right] \\
&\quad\le \mathop{\mathbb{E}}_{x^i,L}\left[\sum_{y^i\in\{0,1\}^{\le b}} (\mathsf{Next}_b(\mathrm{Q}^t, x^{<i}x^i)(y) + |\mathsf{Next}_b(\mathcal{D}_z, x^{<i}x^i)(y^i) - \mathsf{Next}_b(\mathrm{Q}^t, x^{<i}x^i)(y)|)\cdot l(L(x^{<i},x^i,w),y^i)\right] \\
&\quad= S_1 + S_2,
\end{aligned}$$

where

$$\begin{aligned}
S_1 &:= \mathop{\mathbb{E}}_{x^i,L}\left[\sum_{y^i\in\{0,1\}^{\le b}} \mathsf{Next}_b(\mathrm{Q}^t, x^{<i}x^i)(y^i)\cdot l(L(x^{<i},x^i,w),y^i)\right] \\
&= \mathop{\mathbb{E}}_{x^i,L}\left[\mathop{\mathbb{E}}_{y^i\sim\mathsf{Next}_b(\mathrm{Q}^t,x^{<i}x^i)}[l(L(x^{<i},x^i,w),y^i)]\right] \\
S_2 &:= \mathop{\mathbb{E}}_{x^i,L}\left[\sum_{y^i\in\{0,1\}^{\le b}} |\mathsf{Next}_b(\mathcal{D}_z, x^{<i}x^i)(y^i) - \mathsf{Next}_b(\mathrm{Q}^t, x^{<i}x^i)(y^i)|\cdot l(L(x^{<i},x^i,w),y^i)\right] \\
&\le \mathop{\mathbb{E}}_{x^i}\left[\sum_{y^i\in\{0,1\}^{\le b}} |\mathsf{Next}_b(\mathcal{D}_z, x^{<i}x^i)(y^i) - \mathsf{Next}_b(\mathrm{Q}^t, x^{<i}x^i)(y^i)|\cdot \max_{\alpha\in\mathcal{A}_{w,b}} l(\alpha,y^i)\right].
\end{aligned}$$

First, we show the upper bound on S_2. By Lemma 10.3, there exists a polynomial τ_1 such that for every $s, t, m, a, b \in \mathbb{N}$, $z \in \{0,1\}^s$, and $w \in \{0,1\}^*$ with $t \geq \tau_1(d(\mathcal{D}), s, t_D(s))$,

$$\mathop{\mathbb{E}}_{i,x^{<i}}[S_2] \leq \mathop{\mathbb{E}}_{i,x^{<i},x^i}\left[\sum_{y^i \in \{0,1\}^{\leq b}} |\mathsf{Next}_b(\mathcal{D}_z, x^{<i}x^i)(y^i) - \mathsf{Next}_b(\mathrm{Q}^t, x^{<i}x^i)(y^i)| \cdot \max_{\alpha \in \mathcal{A}_{w,b}} l(\alpha, y^i)\right]$$
$$\leq C \cdot \sqrt{\frac{c'(s + d(\mathcal{D}))}{m}} = \sqrt{\frac{c'C^2(s + d(\mathcal{D}))}{m}}$$

for some universal constant $c' > 0$.

Let $m_0 := \frac{256c'C^2(s+d(\mathcal{D}))}{\epsilon^2\delta^2} = O(\frac{C^2 \cdot (s+d(\mathcal{D}))}{\epsilon^2\delta^2})$. Then, for every $m \geq m_0$, we have

$$\mathop{\mathbb{E}}_{i,x^{<i}}[S_2] \leq \sqrt{\frac{c'C^2(s + d(\mathcal{D}))}{m}} \leq \sqrt{\frac{c'C^2(s + d(\mathcal{D}))}{m_0}} = \frac{\epsilon\delta}{16}.$$

It is easy to verify that S_2 is always non-negative. Thus, by Markov's inequality,

$$\Pr_{i,x^{<i}}\left[S_2 \leq \frac{\epsilon}{4}\right] \geq 1 - \frac{\delta}{4}. \tag{10}$$

Note that the above holds for any $s, t, m, \epsilon^{-1}, \delta^{-1}, a, b \in \mathbb{N}$, any $z \in \{0,1\}^s$, and any $w \in \{0,1\}^*$ satisfying $t \geq \tau_1(d(\mathcal{D}), s, t_D(s))$ and $m \geq m_0$.

Next, we show the upper bound on S_1. For readability, we omit "$, x^{<i}x^i$" from $\mathsf{Next}_b(\mathrm{Q}^{t'}, x^{<i}x^i)$ and $\mathsf{Next}_b(\mathcal{D}, x^{<i}x^i)$ and write them as $\mathsf{Next}_b(\mathrm{Q}^{t'})$ and $\mathsf{Next}_b(\mathcal{D})$, respectively.

For each $x^i \in \mathrm{supp}(\mathcal{D}_z^{i,x^{<i}})$, we define E_{x^i} and S'_{1,x^i} as follows:

$$E_{x^i} := \mathop{\mathbb{E}}_{L, y^i \sim \mathsf{Next}_b(\mathrm{Q}^{t'})}[l(L(x^{<i}, x^i, w), y^i) - l(L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'})}(w), y^i)]$$
$$S'_{1,x^i} := \left|\mathop{\mathbb{E}}_{y^i \sim \mathsf{Next}_b(\mathrm{Q}^{t'})}[l(L_{cheat}^{\mathsf{Next}_b(\mathcal{D}_z)}(w), y^i)] - \mathop{\mathbb{E}}_{y^i \sim \mathsf{Next}_b(\mathcal{D}_z)}[l(L_{cheat}^{\mathsf{Next}_b(\mathcal{D}_z)}(w), y^i)]\right|.$$

Then, we have

$$\begin{aligned}
S_1 &= \mathop{\mathbb{E}}_{x^i, L, y^i \sim \mathsf{Next}_b(\mathrm{Q}^{t'})}[l(L(x^{<i}, x^i, w), y^i)] \\
&= \mathop{\mathbb{E}}_{x^i, y^i \sim \mathsf{Next}_b(\mathrm{Q}^{t'})}[l(L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'})}(w), y^i)] + \mathop{\mathbb{E}}_{x^i}[E_{x^i}] \\
&\leq \mathop{\mathbb{E}}_{x^i}\left[\min_{\alpha \in \mathcal{A}_{w,b}} \mathop{\mathbb{E}}_{y^i \sim \mathsf{Next}_b(\mathrm{Q}^{t'})}[l(\alpha, y^i)]\right] + \Delta_{L_{cheat}}(w, b) + \mathop{\mathbb{E}}_{x^i}[E_{x^i}] \\
&\leq \mathop{\mathbb{E}}_{x^i}\left[\mathop{\mathbb{E}}_{y^i \sim \mathsf{Next}_b(\mathrm{Q}^{t'})}[l(L_{cheat}^{\mathsf{Next}_b(\mathcal{D}_z)}(w), y^i)]\right] + \Delta_{L_{cheat}}(w, b) + \mathop{\mathbb{E}}_{x^i}[E_{x^i}] \\
&\leq \mathop{\mathbb{E}}_{x^i}\left[S'_{1,x^i}\right] + \mathop{\mathbb{E}}_{x^i}\left[\mathop{\mathbb{E}}_{y^i \sim \mathsf{Next}_b(\mathcal{D}_z)}[l(L_{cheat}^{\mathsf{Next}_b(\mathcal{D}_z)}(w), y^i)]\right] + \Delta_{L_{cheat}}(w, b) + \mathop{\mathbb{E}}_{x^i}[E_{x^i}] \\
&\leq \mathop{\mathbb{E}}_{x^i}\left[S'_{1,x^i}\right] + \mathop{\mathbb{E}}_{x^i}\left[\min_{\alpha \in \mathcal{A}_{w,b}} \mathop{\mathbb{E}}_{y^i \sim \mathsf{Next}_b(\mathcal{D}_z)}[l(\alpha, y^i)]\right] + 2\Delta_{L_{cheat}}(w, b) + \mathop{\mathbb{E}}_{x^i}[E_{x^i}] \\
&= \mathop{\mathbb{E}}_{x^i}\left[\min_{\alpha \in \mathcal{A}_{w,b}} \mathop{\mathbb{E}}_{y^i \sim \mathsf{Label}_i^{z,x^i}}[l(\alpha, y^i)]\right] + 2\Delta_{L_{cheat}}(w, b) + \mathop{\mathbb{E}}_{x^i}\left[S'_{1,x^i}\right] + \mathop{\mathbb{E}}_{x^i}[E_{x^i}],
\end{aligned}$$

where the first and last inequalities follow from the definition of $\Delta_{L_{cheat}}(w,b)$.

For all $s, m, t, \epsilon^{-1}, \delta^{-1}, a, b \in \mathbb{N}$, $i \in [m]$, and $w \in \{0,1\}^*$ with $m \ge m_0$ and $t \ge \max\{d(\mathcal{D}), t_D(s)\}$, we have $\tau'(t,t) \ge \tau'(d(\mathcal{D}), t_D(s))$. By Lemma 6.15,

$$\Pr_{x^{<i}\sim\mathcal{D}_z^{\le i}, x^i\sim\mathcal{D}_z^{i,x^{<i}}}[\mathrm{cd}^{\tau(\tau'(t,t))}(x^{<i}x^i) \le \mathrm{cd}^{\tau'(t,t)}(z) + \log 8C\epsilon^{-1}\delta^{-1} + 2\log\tau'(t,t)] \ge 1 - \frac{\epsilon\delta}{8C}.$$

In this case,

$$\begin{aligned}\mathrm{cd}^{t'}(x^{<i}x^i) &\le \mathrm{cd}^{\tau(\tau'(t,t))}(x^{<i}x^i)\\ &\le \mathrm{cd}^{\tau'(t,t)}(z) + \log 8C\epsilon^{-1}\delta^{-1} + 2\log\tau'(t,t)\\ &\le \mathrm{cd}^{t}(z) + \log 2\delta^{-1} + 2\log t'\\ &\le \alpha + \log 8C\epsilon^{-1}\delta^{-1} + 2\log t' = \alpha'.\end{aligned}$$

Therefore, by Theorem 8.1,

$$\Pr_{x^{<i}\sim\mathcal{D}_z^{\le i}, x^i\sim\mathcal{D}_z^{i,x^{<i}}}\left[\mathsf{L}_1\left(\mathsf{UE}(x^{<i}x^i; 1^{\langle b,t',\epsilon'^{-1},1^{\alpha'}\rangle}), \mathsf{Next}_b(\mathrm{Q}^{t'}, x^{<i}x^i)\right) \le \frac{\epsilon}{4Cq(w)}\right] \ge 1 - \frac{\epsilon\delta}{8C}.$$

By Markov's inequality,

$$\Pr_{z,x^{<i}}\left[\Pr_{x^i}\left[\mathsf{L}_1\left(\mathsf{UE}(x^{<i}x^i), \mathsf{Next}_b(\mathrm{Q}^{t'})\right) \le \frac{\epsilon}{4Cq(w)}\right] \ge 1 - \frac{\epsilon}{4C}\right] \ge 1 - \frac{\delta}{2}. \tag{11}$$

Recall that (i) L simulates the oracle access for executing L_{cheat} by $\mathsf{UE}(x^{<i}x^i)$, and (ii) L_{cheat} accesses the oracle at most $q(w)$ times. Thus, for every $w \in \{0,1\}^*$ and every $z, x^{<i}, x^i$ satisfying the event in inequality (11), it holds that

$$\mathsf{L}_1(L(x^{<i}, x^i, w), L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'})}(w)) = \mathsf{L}_1(L_{cheat}^{\mathsf{UE}(x^{<i}x^i)}(w), L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'})}(w)) \le q(w)\cdot\frac{\epsilon}{4Cq(w)} = \frac{\epsilon}{4C}$$

and

$$\begin{aligned}E_{x^i} &= \mathop{\mathbb{E}}_{L, y^i\sim\mathsf{Next}_b(\mathrm{Q}^{t'})}[l(L(x^{<i}, x^i, w), y^i) - l(L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'})}(w), y^i)]\\ &\le \max_{y\in\{0,1\}^{\le b}} \mathop{\mathbb{E}}_{L,\mathsf{Next}_b(\mathrm{Q}^{t'})}[l(L(x^{<i}, x^i, w), y) - l(L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'})}(w), y)]\\ &\le \max_{y\in\{0,1\}^{\le b}} \sum_{\alpha\in\mathcal{A}_{w,b}} \left(\Pr[L(x^{<i}, x^i, w) = \alpha] - \Pr[L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'})}(w) = \alpha]\right)\cdot l(\alpha, y)\\ &\le \max_{\alpha'\in\mathcal{A}_{w,b}, y\in\{0,1\}^{\le b}} l(\alpha', y)\cdot\sum_{\alpha\in\mathcal{A}_{w,b}}\left(\Pr[L(x^{<i}, x^i, w) = \alpha] - \Pr[L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'})}(w) = \alpha]\right)\\ &\le C\cdot\frac{\epsilon}{4C} = \frac{\epsilon}{4}.\end{aligned}$$

Thus, for every $z, x^{<i}$ satisfying the event in inequality (11), we have

$$\begin{aligned}\mathop{\mathbb{E}}_{x^i}[E_{x^i}] &= \mathop{\mathbb{E}}_{x^i, L, y^i \sim \mathsf{Next}_b(\mathrm{Q}^{t'})}[l(L(x^{<i}, x^i, w), y^i) - l(L_{cheat}^{\mathsf{Next}_b(\mathrm{Q}^{t'})}(w), y^i)] \\ &\le 1 \cdot \frac{\epsilon}{4} + \frac{\epsilon}{4C} \max_{\alpha, \alpha' \in \mathcal{A}_{w,b}, y \in \{0,1\}^{\le b}} \left(l(\alpha, y) - l(\alpha', y)\right) \\ &\le \frac{\epsilon}{4} + \frac{\epsilon}{4C} \cdot C = \frac{\epsilon}{2}.\end{aligned}$$

We also evaluate S'_{1,x^i} in the same way as S_2. By Lemma 10.3, for every $s, t, m, a, b \in \mathbb{N}$, $z \in \{0,1\}^s$, and $w \in \{0,1\}^*$ with $t \ge \tau_1(d(\mathcal{D}), s, t_D(s))$ and $m \ge m_0$,

$$\mathop{\mathbb{E}}_{i, x^{<i}}[\mathop{\mathbb{E}}_{x^i}[S'_{1,x^i}]] \le \sqrt{\frac{c' C^2 (s + d(\mathcal{D}))}{m}} \le \sqrt{\frac{c' C^2 (s + d(\mathcal{D}))}{m_0}} = \frac{\epsilon\delta}{16}.$$

Because S'_{1,x^i} is always non-negative, by Markov's inequality,

$$\Pr_{i, x^{<i}}\left[\mathop{\mathbb{E}}_{x^i}[S'_{1,x^i}] \le \frac{\epsilon}{4}\right] \ge 1 - \frac{\delta}{4}. \tag{12}$$

Because $t' \ge \tau_1(t, s, t) \ge \tau_1(d(\mathcal{D}), s, t_D(s))$ for $t \ge \max\{d(\mathcal{D}), t_D(s)\}$, by inequalities (10), (11), and (12) and the union bound, for every $w \in \{0,1\}^*$, it holds that (i) $\mathbb{E}_{x^i}[E_{x^i}] \le \epsilon/2$, (ii) $\mathbb{E}_{x^i}[S'_{1,x^i}] \le \epsilon/4$, and (iii) $S_2 \le \epsilon/4$ with probability at least $1 - \delta$ over the choice of $i \sim [m]$ and $x^{<i} \sim \mathcal{D}_z^{<i}$. In this case, we have

$$\begin{aligned}\mathop{\mathbb{E}}_{x^i, y^i, L}\left[l(L(x^{<i}, x^i, w), y^i)\right] &\le S_1 + S_2 \\ &\le \mathop{\mathbb{E}}_{x^i}\left[\min_{\alpha \in \mathcal{A}_{w,b}} \mathop{\mathbb{E}}_{y^i \sim \mathsf{Label}_i^{z, x^i}}[l(\alpha, y^i)]\right] + 2\Delta_{L_{cheat}}(w, b) + \mathop{\mathbb{E}}_{x^i}\left[S'_{1,x^i}\right] + \mathop{\mathbb{E}}_{x^i}[E_{x^i}] + S_2 \\ &\le \mathop{\mathbb{E}}_{x^i}\left[\min_{\alpha \in \mathcal{A}_{w,b}} \mathop{\mathbb{E}}_{y^i \sim \mathsf{Label}_i^{z, x^i}}[l(\alpha, y^i)]\right] + 2\Delta_{L_{cheat}}(w, b) + \frac{\epsilon}{2} + \frac{\epsilon}{4} + \frac{\epsilon}{4} \\ &= \mathop{\mathbb{E}}_{x^i}\left[\min_{\alpha \in \mathcal{A}_{w,b}} \mathop{\mathbb{E}}_{y^i \sim \mathsf{Label}_i^{z, x^i}}[l(\alpha, y^i)]\right] + 2\Delta_{L_{cheat}}(w, b) + \epsilon.\end{aligned}$$

Hence, we conclude that

$$\Pr_{i, x^{<i}}\left[\mathop{\mathbb{E}}_{x^i, y^i, L}\left[l(L(x^{<i}, x^i, w), y^i)\right] \le \mathop{\mathbb{E}}_{x^i}\left[\min_{\alpha \in \mathcal{A}_{w,b}} \mathop{\mathbb{E}}_{y^i \sim \mathsf{Label}_i^{z, x^i}}[l(\alpha, y^i)]\right] + 2\Delta_{L_{cheat}}(w, b) + \epsilon\right] \ge 1 - \delta.$$

□

10.2 Universal Agnostic Learning

We introduce the agnostic learning model and show that universal average-case agnostic learning is feasible under the non-existence of one-way functions.

In this section, we define a sampler of example size n as a multi-output circuit that outputs a pair $(x, y) \in \{0,1\}^n \times \{0,1\}^{\mathsf{poly}(n)}$ (we call x an example and y a label of x). For any sampler $S\colon \{0,1\}^\ell \to \{0,1\}^n \times \{0,1\}^{\mathsf{poly}(n)}$, we use $S^{(1)}$ (resp. $S^{(2)}$) to refer to the circuit that produces the first (resp. second) half element of S, i.e., $S(r) = (S^{(1)}(r), S^{(2)}(r))$, for each seed $r \in \{0,1\}^\ell$. For convenience, we may identify a sampler $S\colon \{0,1\}^\ell \to \{0,1\}^n \times \{0,1\}^{\mathsf{poly}(n)}$ with a distribution of $S(r)$, where $r \sim \{0,1\}^\ell$. For each sampler S, we define an example oracle EX_S as the oracle that returns $(x, y) \sim S$ for each access. For simplicity, we define the time complexity of sampler as a function in the example size n instead of the seed length ℓ. For any $t, s \in \mathbb{N}$, we say that a sampler S of example size n is t/s-time computable if there exists a program $\Pi_S \in \{0,1\}^{\leq s}$ such that $U^t(\Pi_S, r) = S(r_{[\ell]})$ for each seed $r \in \{0,1\}^t$. Additionally if $\mathrm{cd}^{t'}(\Pi_S) \leq \alpha$, we say that the t-time-bounded computational depth of S is at most α.

In the original agnostic learning model presented in [KSS94], a learner for a concept class $\mathscr{C}$ is given access to EX_S for an unknown sampler S, and the task is to approximate the best function in $\mathscr{C}$ that approximates the label under S (more generally, the best function in $\mathscr{C}$ that minimizes the expected loss for some loss function) for all samplers S (i.e., in the worst case with regard to S).

Definition 10.4 (Agnostic learning)**.** *Let $b\colon \mathbb{N} \to \mathbb{N}$ be the size of each label. Let $\mathscr{C}$ be a concept class defined as a subset of $\{f\colon \{0,1\}^n \to \{0,1\}^{b(n)} : n \in \mathbb{N}\}$. Let $l\colon \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}_{\geq 0}$ be a loss function. Let $\mathcal{S} = \{\mathcal{S}_n\}_{n\in\mathbb{N}}$ be a class of samplers, where each $\mathcal{S}_n$ is a subset of samplers over $\{0,1\}^n \times \{0,1\}^{b(n)}$.*

We say that a randomized oracle L, which is referred to as an agnostic learner, agnostically learns $\mathscr{C}$ on $\mathcal{S}$ for a loss function l if for every sufficiently large $n \in \mathbb{N}$, every $\epsilon^{-1}, \delta^{-1} \in \mathbb{N}$, and every (unknown) target sampler $S \in \mathcal{S}_n$, the learner $L(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$ is given access to EX_S and outputs a circuit $h\colon \{0,1\}^n \to \{0,1\}^{b(n)}$ such that

$$\mathop{\mathbb{E}}_{(x,y)\sim S}[l(h(x), y)] \leq \mathsf{opt}_{\mathscr{C}}(S) + \epsilon,$$

where

$$\mathsf{opt}_{\mathscr{C}}(S) = \min_{f\in\mathscr{C}} \mathop{\mathbb{E}}_{(x,y)\sim S}[l(f(x), y)]$$

with probability at least $1 - \delta$ over the choice of EX_S and randomness for L. We define the sample complexity $m(n, \epsilon, \delta)$ of L as the upper bound on the number of query accesses by $L(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$ for each $n, \epsilon^{-1}, \delta^{-1} \in \mathbb{N}$.

We also introduce the average-case variant of agnostic learning. We define a distribution on samplers as a family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ of distributions, where $\mathcal{D}_n$ is a distribution on descriptions of a sampler of example size n. For every distribution $\mathcal{D}$ on samplers and every $n \in \mathbb{N}$, we use the notation $\mathcal{D}_n$ to refer to the n-th distribution in $\mathcal{D}$, i.e., the distribution on descriptions of samplers of example size n. Then, we formally define the average-case variant of agnostic learning as follows.

Definition 10.5 (Agnostic learning on average)**.** *Let $b\colon \mathbb{N} \to \mathbb{N}$ be the size of each label. Let $\mathscr{C}$ be a concept class defined as a subset of $\{f\colon \{0,1\}^n \to \{0,1\}^{b(n)} : n \in \mathbb{N}\}$ and $\mathcal{D}$ be a distribution on samplers S over $\{0,1\}^n \times \{0,1\}^{b(n)}$ for the example size n. Let $l\colon \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}_{\geq 0}$ be a loss function.*

We say that a randomized oracle L, which is referred to as an agnostic learner, agnostically learns $\mathscr{C}$ on average under $\mathcal{D}$ for a loss function l if for every sufficiently large $n \in \mathbb{N}$ and every

$\epsilon^{-1}, \delta^{-1} \in \mathbb{N}$, the learner $L(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$ is given access to EX_S, where S is an unknown target sampler selected according to $\mathcal{D}_n$, and outputs a circuit $h\colon \{0,1\}^n \to \{0,1\}^{b(n)}$ such that

$$\mathop{\mathbb{E}}_{(x,y)\sim S}[l(h(x), y)] \leq \mathsf{opt}_{\mathscr{C}}(S) + \epsilon$$

with probability at least $1-\delta$ over the choice of $S \sim \mathcal{D}_n$, EX_S, and randomness for L. We define the sample complexity $m(n,\epsilon,\delta)$ of L as the upper bound on the number of query accesses by $L(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}})$ for each $n, \epsilon^{-1}, \delta^{-1} \in \mathbb{N}$.

Let $\mathscr{D}$ be a class of distributions on samplers. We say that $\mathscr{C}$ is agnostic learnable in polynomial time on average under $\mathscr{D}$ for a loss function l if there exists a polynomial-time agnostic learner that agnostically learns $\mathscr{C}$ on average under $\mathcal{D}$ for every (unknown) $\mathcal{D} \in \mathscr{D}$.

If we do not specify the loss function, we always assume the 0-1 loss function l defined as

$$l(\tilde{y}, y) = \begin{cases} 1 & \text{if } \tilde{y} \neq y \\ 0 & \text{if } \tilde{y} = y. \end{cases}$$

In this case, the requirement for the hypothesis h in Definition 10.5 is simply expressed as follows:

$$\Pr_{(x,y)\sim S}[h(x) \neq y] \leq \mathsf{opt}_{\mathscr{C}}(S) + \epsilon = \min_{f \in \mathscr{C}} \Pr_{(x,y)\sim S}[f(x) \neq y] + \epsilon.$$

Now, we show the feasibility of universal agnostic learning from Theorem 10.2, which is a formal statement of Theorem 2.3. Note that Item 4 of Theorem 2.4 corresponds to Item 2.

Theorem 10.6. *The following are equivalent:*

1. *There is no infinitely-often one-way function.*
2. *For all polynomials $b(n), s(n), t(n)$, and $t'(n)$, the class $\mathcal{F} = \{f\colon \{0,1\}^n \to \{0,1\}^{b(n)} : n \in \mathbb{N}\}$ is agnostically learnable in polynomial time on average under (unknown) $t'(n)$-time samplable distributions over $t(n)/s(n)$-time computable samplers with sample complexity $m(n,\epsilon,\delta) = O(s(n)\epsilon^{-2}\log\delta^{-1})$.*
3. *There exists a polynomial-time randomized algorithm L that is given input $1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}}$ and an additional meta-parameter $1^{\langle b,s,t,t',2^\alpha\rangle}$ such that for all functions $b(n), s(n), t(n), t'(n), \alpha(n)$, the algorithm $L(1^n, 1^{\epsilon^{-1}}, 1^{\delta^{-1}}; 1^{\langle b(n),s(n),t(n),t'(n),2^{\alpha(n)}\rangle})$ agnostically learns $\mathcal{F} = \{f\colon \{0,1\}^n \to \{0,1\}^{b(n)} : n \in \mathbb{N}\}$ on $t(n)/s(n)$-time computable samplers whose $t'(n)$-time-bounded computational depth is at most $\alpha(n)$ in polynomial time in $n, \epsilon^{-1}, \delta^{-1}, b(n), s(n), t(n), t'(n), 2^{\alpha(n)}$ with sample complexity $m(n,\epsilon,\delta) = O(s(n)\epsilon^{-2}\log\delta^{-1})$.*

The sample complexity above is optimal when δ is constant, as discussed in Appendix A.

Proof. The implication item 2 $\Rightarrow$ item 1 follows from [GGM86; HILL99] and the observation in [Val84]. The implication item 3 $\Rightarrow$ item 2 follows from Lemma 6.14 and a basic probabilistic argument based on the union bound. Thus, we only show the implication item 1 $\Rightarrow$ item 3.

Suppose that there is no infinitely-often one-way function (item 1). Then, there exists the universal extrapolation algorithm UE in Theorem 8.1.

First, we construct an agnostic learner as a cheating learner $L^?_{cheat}$ as follows: On input $1^{\epsilon^{-1}}, 1^b$ (where $\epsilon^{-1}, b \in \mathbb{N}$) and given access to distribution Label over $\{0,1\}^{\leq b}$, the learner $L^?_{cheat}$ obtains

$q := (96)^2(b+1)\epsilon^{-2}\ln(192\epsilon^{-1})$ samples $y^1, \ldots, y^q$ from Label and outputs the most frequently sampled label $\tilde{y} \in \{0,1\}^{\le b}$, i.e., $\tilde{y} = y^{\tilde{i}}$ for $\tilde{i} = \arg\max_{i\in[q]} |\{j \in [q] : y^i = y^j\}|$. Trivially, $L^?_{cheat}$ halts in $\mathsf{poly}(\epsilon^{-1}, b)$ time.

We apply Theorem 10.2 for $L^?_{cheat}$ to obtain the learner L' that simulates $L^?_{cheat}$. We can show that L' satisfies the following property:

Claim 10.7. *There exists a polynomial $m_0(n, \epsilon^{-1}, \delta^{-1}) = O(s(n)\epsilon^{-2}\delta^{-2})$ such that for every $t(n)/s(n)$-time computable sampler whose $t'(n)$-time-bounded computational depth is at most $\alpha(n)$, every sufficiently large $n \in \mathbb{N}$, and every $\epsilon^{-1}, \delta^{-1} \in \mathbb{N}$,*

$$\Pr_{i,(x^1,y^1),\ldots,(x^{i-1},y^{i-1})}\left[\Pr_{(x,y)\sim S, L'}[L'(x^{<i}, x, 1^{\epsilon^{-1}}, 1^{b(n)}; 1^{\langle s(n), b(n), \tau, 2^{\alpha(n)}, 16\epsilon^{-1}, \delta^{-1}\rangle}) \neq y] \le \mathsf{opt}_{\mathcal{F}}(S) + \frac{\epsilon}{8}\right] \ge 1-\delta,$$

where $m = m_0(n, \epsilon^{-1}, \delta^{-1})$, $i \sim [m]$, $(x^1, y^1), \ldots, (x^{i-1}, y^{i-1}) \sim S$, $x^{<i} = x^1y^1 \cdots x^{i-1}y^{i-1}$, and $\tau = O(t'(n) + t(n)m)$.

First, we assume Claim 10.7 and show Theorem 10.6.

We construct an agnostic learner L for a fixed confidence error $\delta = 1/4$ with sample complexity $O(s(n)\epsilon^{-2})$. To reduce the confidence error $1/4$ to the arbitrary $\delta \in (0,1]$ given as a parameter, it suffices to repeat L $O(\log \delta^{-1})$ times with the accuracy error $\epsilon/2$ and output the best hypothesis by empirically estimating the accuracy error for each hypothesis within the approximation error $\pm\epsilon/2$ (see [HKLW88]). The time and sample complexity is affected only by the multiplicative factor $O(\log \delta^{-1})$ (note that the empirical estimation only requires additional $O(\epsilon^{-2}\log\delta^{-1})$ samples).

The construction of L is as follows: On input $1^n, 1^{\epsilon^{-1}}$, a meta-parameter $1^{\langle b(n), s(n), t(n), t'(n), 2^{\alpha(n)}\rangle}$, and given access to EX_S, where S is an unknown $t(n)/s(n)$-time computable sampler whose example size is n and $t'(n)$-time-bounded computational depth is at most $\alpha(n)$, the learner L selects $i \sim [m]$, where $m = m_0(n, \epsilon^{-1}, 8)$ and m_0 is the polynomial in Claim 10.7, and obtains $i-1$ samples $(x^1, y^1), \ldots, (x^{i-1}, y^{i-1})$ from EX_S. Then, L selects a sufficiently long random string r and outputs a circuit (i.e., hypothesis) h_r that is taken $x \in \{0,1\}^n$ as input and outputs

$$y = L'(x^1y^1 \cdots x^{i-1}y^{i-1}, x, \langle 1^{\epsilon^{-1}}, 1^{b(n)}\rangle; 1^{\langle s(n), b(n), \tau, 2^{\alpha(n)}, 16\epsilon^{-1}, 8\rangle}; r),$$

where $\tau = O(t'(n) + t(n)m)$ as indicated in Claim 10.7.

It is not hard to verify that L halts in $\mathsf{poly}(n, \epsilon^{-1}, s(n), b(n), t(n), t'(n), 2^{\alpha(n)})$ time. In addition, the sample complexity is $m_0(n, \epsilon^{-1}, 8) = O(s(n)\epsilon^{-2})$. We also verify the correctness of L as follows. By Claim 10.7, with probability at least $7/8$ over the choice of $i, (x^1, y^1), \ldots, (x^{i-1}, y^{i-1})$, it holds that

$$0 \le \Pr_{(x,y)\sim S, r}[h_r(x) \neq y] - \mathsf{opt}_{\mathcal{F}}(S) \le \epsilon/8,$$

where the non-negativity follows from the definition of $\mathsf{opt}_{\mathcal{F}}(S)$. Thus, by Markov's inequality,

$$\Pr_r\left[\Pr_{(x,y)\sim S}[h_r(x) \neq y] - \mathsf{opt}_{\mathcal{F}}(S) \le \epsilon\right] \ge 7/8.$$

By the union bound, with probability at least 1- 1/8 -1/8= 3/4 over the choice of randomness for L and samples drawn from EX_S, the learner L outputs a hypothesis h_r that satisfies

$$\Pr_{(x,y)\sim S}[h_r(x) \neq y] \le \mathsf{opt}_{\mathcal{F}}(S) + \epsilon.$$

In the remainder, we show Claim 10.7.

Proof of Claim 10.7. First, we analyze the performance of the cheating learner $L^{?}_{cheat}$. Let Label be an arbitrary distribution over $\{0,1\}^{\le b}$. Let y^* be the label mostly generated according to Label, i.e., $y^* := \arg\max_{y\in\{0,1\}^{\le b}} \mathsf{Label}(y)$ (breaking ties arbitrarily). Recall that $L^{\mathsf{Label}}_{cheat}$ collects $q := (96)^2(b+1)\epsilon^{-2}\ln(192\epsilon^{-1})$ samples $y^1,\dots,y^q$. By Hoeffding's inequality, for each $y \in \{0,1\}^{\le b}$, the estimated outcome probability $\tilde{p}_y = |\{i \in [q] : y^i = y\}|/q$ satisfies $\mathsf{Label}(y) - \epsilon/96 \le \tilde{p}_y \le \mathsf{Label}(y) + \epsilon/96$ with probability at least $1 - 2e^{2q(\epsilon/96)^2} \ge 1 - (\epsilon/96)\cdot 2^{-(b+1)}$. By the union bound, $\mathsf{Label}(y) - \epsilon/96 \le \tilde{p}_y \le \mathsf{Label}(y) + \epsilon/96$ holds for all $y \in \{0,1\}^{\le b}$ with probability at least $1 - \epsilon/96$. Under this event, the probability that the output $\tilde{y}$ of $L^{\mathsf{Label}}_{cheat}$ corresponds to $y \sim \mathsf{Label}$ is at least

$$\mathsf{Label}(\tilde{y}) \ge \tilde{p}_{\tilde{y}} - \epsilon/96 \ge \tilde{p}_{y^*} - \epsilon/96 \ge \mathsf{Label}(y^*) - \epsilon/48.$$

In this case, we have

$$\begin{aligned}\Pr_{\mathsf{Label},y\sim\mathsf{Label}}[L^{\mathsf{Label}}_{cheat}(1^\epsilon,1^b) \ne y | \forall \tilde{p}_y \in \mathsf{Label}(y) \pm \epsilon/96] &\le \Pr_{y\sim\mathsf{Label}}[y \ne y^*] + \epsilon/48\\ &= \min_{\tilde{y}\in\{0,1\}^{\le b}} \Pr_{y\sim\mathsf{Label}}[y \ne \tilde{y}] + \epsilon/48.\end{aligned}$$

Therefore, for every $\epsilon^{-1}, b \in \mathbb{N}$ and every Label over $\{0,1\}^{\le b}$,

$$\Pr_{\mathsf{Label},y\sim\mathsf{Label}}[L^{\mathsf{Label}}_{cheat}(1^\epsilon,1^b) \ne y] = \min_{\tilde{y}\in\{0,1\}^{\le b}} \Pr_{y\sim\mathsf{Label}}[y\ne\tilde{y}] + \epsilon/48 + \epsilon/96 \le \min_{\tilde{y}\in\{0,1\}^{\le b}} \Pr_{y\sim\mathsf{Label}}[y\ne\tilde{y}] + \epsilon/32.$$

Thus, for every $\epsilon^{-1}, b \in \mathbb{N}$,

$$\Delta_{L_{cheat}}(\langle 1^\epsilon, 1^b\rangle, b) := \sup_{\mathsf{Label}} \left(\Pr_{\mathsf{Label},y\sim\mathsf{Label}}[L^{\mathsf{Label}}_{cheat}(1^\epsilon,1^b) \ne y] - \min_{\tilde{y}\in\{0,1\}^{\le b}} \Pr_{y\sim\mathsf{Label}}[y\ne\tilde{y}] \right) \le \epsilon/32.$$

Now, we analyze the performance of L'. Let $\mathcal{D}$ be an arbitrary $t(n)/s(n)$-time computable sampler described by a program Π_S whose $t'(n)$-time-bounded computational depth is at most $\alpha(n)$. Let m_0 be the polynomial in Theorem 10.2.

We define a distribution family $\mathcal{E}_t = \{\mathcal{E}_{t,z}\}_{z\in\{0,1\}^*}$, where $\mathcal{E}_{t,z}$ is a distribution of an *infinitely* long string $x^1x^2x^3\cdots$. Here, for each $i \in \mathbb{N}$, $x^i \sim U^t(s, r^i)$ for a uniformly random seed $r^i \sim \{0,1\}^t$ (note that if s is a description of a t-time-computable sampler, each x^i corresponds to a sample). Then, for every $n, \epsilon^{-1}, \delta^{-1} \in \mathbb{N}$ and $m_0 := m_0(C\log t, s(n), 1, 16\epsilon^{-1}, \delta^{-1})$, where $C > 0$ is a universal constant, the prefix $x^1\cdots x^{m_0}$ of $\mathcal{D}_{t(n),\Pi_S}$ is samplable in $O(m_0\cdot t(n))$ time. Therefore, by Theorem 10.2, for every sufficiently large $n \in \mathbb{N}$ and every $\epsilon^{-1}, \delta^{-1}$ (note that we select $a = n$ and $b = b(n)$) and for sufficiently large $\tau = \max\{t'(n), O(m_0\cdot t(n))\}$,

$$\Pr_{i,\{(x^j,y^j)\}_{j=1}^{i-1}}\left[\Pr_{(x,y)\sim S,L'}\left[L'(x^{<i},x,1^\epsilon,1^{b(n)};1^N) \ne y\right] \le \mathop{\mathbb{E}}_{x\sim S^{(1)}}\left[\min_{y^*} \Pr_{y\sim S^{(2)}_{|x}}[y^*=y]\right] + 2\Delta_{L_{cheat}} + \frac{\epsilon}{16}\right] \ge 1-\delta,$$

where $i \sim [m_0]$; $(x^1,y^1),\dots,(x^{i-1},y^{i-1}) \sim S$; $N = \langle n, s(n), b(n), \tau, 16\epsilon^{-1}, \delta^{-1}\rangle$; $x^{<i} = x^1y^1\cdots x^{i-1}y^{i-1}$; $S^{(2)}_{|x}$ is a conditional distribution of $S^{(2)}$ given $x \sim S^{(1)}$, and $\Delta_{L_{cheat}} = \Delta_{L_{cheat}}(\langle 1^\epsilon,1^b\rangle, b) \le \epsilon/32$. It is easy to verify that

$$\mathop{\mathbb{E}}_{x\sim S^{(1)}}\left[\min_{y^*\in\{0,1\}^{<b(n)}} \Pr_{y\sim S^{(2)}_{|x}}[y^*=y]\right] = \mathsf{opt}_{\mathcal{F}}(S).$$

Thus, we conclude that

$$\Pr_{S,i,\{(x^j,y^j)\}_{j=1}^{i-1}}\left[\Pr_{(x,y)\sim S,L'}\left[L'(x^{<i},x,1^{\epsilon},1^{b(n)};1^N)\neq y\right]\leq \mathsf{opt}_{\mathcal{F}}(S)+\epsilon/8\right]\geq 1-\delta.$$

The sample complexity m_0 is bounded above by

$$m_0(C\log t(n),s(n),1,16\epsilon^{-1},\delta^{-1})=O((s(n)+\log t(n))\epsilon^{-2}\delta^{-2})=O(s(n)\epsilon^{-2}\delta^{-2}),$$

where we assume that $\log t(n)\leq s(n)$; otherwise, a learner can try all possible $t(n)/s(n)$-time computable samplers in $t(n)\cdot 2^{O(s(n))}\leq \mathsf{poly}(t(n))$ time and output the best hypothesis through the standard empirical estimation of the accuracy error. $\diamond$

$\square$

10.3 Universal Agnostic Learning for General Loss

In this section, we consider agnostic learning for general polynomial-time computable loss functions.

Theorem 10.8. *The following are equivalent:*

1. *There is no infinitely-often one-way function.*
2. *For every $b(n)=O(\log n)$, every polynomials $s(n),t(n),t'(n)$, and $c(n)$, and every polynomial-time computable loss function l with $l(\tilde{y},y)\leq c(n)$ for each $n\in\mathbb{N}$ and each $\tilde{y},y\in\{0,1\}^{\leq b(n)}$, the class $\mathcal{F}=\{f\colon\{0,1\}^n\to\{0,1\}^{b(n)}:n\in\mathbb{N}\}$ is agnostically learnable for the loss function l in polynomial time on average under (unknown) $t'(n)$-time samplable distributions over $t(n)/s(n)$-time computable samplers with sample complexity $m(n,\epsilon,\delta)=O(s(n)c(n)^2\epsilon^{-2}\log\delta^{-1})$.*
3. *For every $b(n)=O(\log n)$, every polynomial $c(n)$, and every polynomial-time computable loss function l with $l(\tilde{y},y)\leq c(n)$ for each $n\in\mathbb{N}$ and each $\tilde{y},y\in\{0,1\}^{\leq b(n)}$, there exists a polynomial-time randomized algorithm L that is given input $1^n,1^{\epsilon^{-1}},1^{\delta^{-1}}$ and an additional meta-parameter $1^{\langle s,t,t',2^{\alpha}\rangle}$ such that for all functions $s(n),t(n),t'(n),\alpha(n)$, the algorithm $L(1^n,1^{\epsilon^{-1}},1^{\delta^{-1}};1^{\langle s(n),t(n),t'(n),2^{\alpha(n)}\rangle})$ agnostically learns $\mathcal{F}=\{f\colon\{0,1\}^n\to\{0,1\}^{b(n)}:n\in\mathbb{N}\}$ for the loss function l on $t(n)/s(n)$-time computable samplers whose $t'(n)$-time-bounded computational depth is at most $\alpha(n)$ in polynomial time in $n,\epsilon^{-1},\delta^{-1},b(n),s(n),t(n),t'(n),2^{\alpha(n)}$ with sample complexity $m(n,\epsilon,\delta)=O(s(n)c(n)^2\epsilon^{-2}\log\delta^{-1})$.*

Proof. The implication item 2 $\Rightarrow$ item 1 follows from [GGM86; HILL99] and the observation in [Val84]. The implication item 3 $\Rightarrow$ item 2 follows from Lemma 6.14 and a basic probabilistic argument based on the union bound. Thus, we only show the implication item 1 $\Rightarrow$ item 2.

The outline of the proof is similar to that for Theorem 10.6. The only difference is the construction of the cheating learner $L^{?}_{cheat}$.

Let $l\colon\{0,1\}^*\times\{0,1\}^*\to\mathbb{R}_{\geq 0}$ be an arbitrary polynomial-time computable loss function such that $l(\tilde{y},y)\leq c(n)$ for each $n\in\mathbb{N}$ and each $\tilde{y},y\in\{0,1\}^{\leq b(n)}$, where $b(n)=O(\log n)$ and $c(n)=\mathsf{poly}(n)$. We construct a cheating learner $L^{?}_{cheat}$ for minimizing the expected loss with respect to l with additive error $\epsilon/32$ for all distributions Label over $\{0,1\}^{\leq b(n)}$.

On input $1^n,1^{\epsilon^{-1}}$ and given access to a distribution Label over $\{0,1\}^{\leq b(n)}$, the learner L_{cheat} obtains $q:=(64c(n)2^{b(n)+1})^2(b(n)+1)\epsilon^{-2}\ln(128\epsilon^{-1}c(n))$ samples $y^1,\ldots y^q$ from Label and computes

$\tilde{p}_y = |\{j \in [q] : y^j = y\}|/q$ for each $y \in \{0,1\}^{\leq b(n)}$ (note that it takes only $O(q2^{b(n)}) = \mathsf{poly}(n, \epsilon^{-1})$ times). These estimated probabilities determine an estimated distribution $\tilde{Y}$ over $\{0,1\}^{\leq b(n)}$, where each y is drawn from $\tilde{Y}$ with probability $\tilde{p}_y$. The learner L_{cheat} minimizes the expected loss under $\tilde{Y}$, i.e., L_{cheat} computes $\tilde{l}_\alpha = \mathbb{E}_{y\sim\tilde{Y}}[l(\alpha, y)]$ for each $\alpha \in \{0,1\}^{\leq b(n)}$ and outputs $\tilde{\alpha} = \arg\min_{\alpha\in\{0,1\}^{\leq b(n)}} \tilde{l}_\alpha$ (note that it takes only $O(2^{b(n)}) = \mathsf{poly}(n)$ times).

We show that for every $n, \epsilon^{-1} \in \mathbb{N}$ and every distribution Label over $\{0,1\}^{\leq b(n)}$,

$$\mathop{\mathbb{E}}_{\mathsf{Label}, y\sim\mathsf{Label}}[l(L_{cheat}^{\mathsf{Label}}(1^n, 1^{\epsilon^{-1}}), y)] \leq \min_{\alpha\in\{0,1\}^{\leq b}} \mathop{\mathbb{E}}_{y\sim\mathsf{Label}}[l(\alpha, y)] + \epsilon/32. \tag{13}$$

This implies Theorem 10.8 by the same argument as Theorem 10.6; i.e., we apply Theorem 10.2 for L_{cheat} to obtain the learner L' that simulates L_{cheat}, and then we construct an agnostic learner L (with fixed confidence error $\delta = 1/4$) that selects $i \sim [m]$ and randomness r, where $m = O(s(n)c(n)^2\epsilon^{-2})$ indicated in Theorem 10.2, collects $i-1$ samples $(x^1, y^1), \ldots, (x^{i-1}, y^{i-1})$ from EX_S, and outputs h_r that takes $x \in \{0,1\}^n$ as input and outputs

$$y = L'(x^1y^1 \cdots x^{i-1}y^{i-1}, x, 1^n, 1^{\epsilon^{-1}}; 1^{\langle n, s(n), b(n), \tau, 32\epsilon^{-1}, 8\rangle}; r),$$

for $\tau = O(t'(n) + t(n)m)$. The correctness of L holds in the same way as Theorem 10.6 if inequality (13) holds.

We verify inequality (13). Let $\alpha^* = \arg\min_{\alpha\in\{0,1\}^{\leq b(n)}} \mathbb{E}_{y\sim\mathsf{Label}}[l(\alpha, y)]$. Recall that $L_{cheat}^{\mathsf{Label}}$ collects $q := (64c(n)2^{b(n)+1})^2(b(n)+1)\epsilon^{-2}\ln(128\epsilon^{-1}c(n))$ samples $y^1, \ldots, y^q$. By Hoeffding's inequality, for each $y \in \{0,1\}^{\leq b(n)}$, the estimated outcome probability $\tilde{p}_y = |\{i \in [q] : y^i = y\}|/q$ satisfies $\mathsf{Label}(y) - \epsilon/(64c(n)2^{b(n)+1}) \leq \tilde{p}_y \leq \mathsf{Label}(y) + \epsilon/(64c(n)2^{b(n)+1})$ with probability at least $1 - 2e^{2q(\epsilon/64c(n)2^{b(n)+1})^2} \geq 1 - (\epsilon/64c(n)) \cdot 2^{-(b(n)+1)}$. By the union bound, $\mathsf{Label}(y) - \epsilon/(64c(n)2^{b(n)+1}) \leq \tilde{p}_y \leq \mathsf{Label}(y) + \epsilon/(64c(n)2^{b(n)+1})$ holds for all $y \in \{0,1\}^{\leq b(n)}$ with probability at least $1 - \epsilon/64c(n)$. Under this event, the output $\tilde{\alpha} \in \{0,1\}^{\leq b(n)}$ of $L_{cheat}^{\mathsf{Label}}$ satisfies

$$\begin{aligned}
\mathop{\mathbb{E}}_{y\sim\tilde{Y}}[l(\tilde{\alpha}, y)] \leq \mathop{\mathbb{E}}_{y\sim\tilde{Y}}[l(\alpha^*, y)] &= \sum_{y\in\{0,1\}^{\leq b(n)}} \tilde{p}_y \cdot l(\alpha^*, y) \\
&\leq \sum_{y\in\{0,1\}^{\leq b(n)}} (\mathsf{Label}(y) + \frac{\epsilon}{64c(n)2^{b(n)+1}}) \cdot l(\alpha^*, y) \\
&\leq \sum_{y\in\{0,1\}^{\leq b(n)}} \mathsf{Label}(y) l(\alpha^*, y) + \frac{\epsilon \cdot |\{0,1\}^{\leq b(n)}|}{64c(n)2^{b(n)+1}} \cdot \max_{y\in\{0,1\}^{\leq b(n)}} l(\alpha^*, y) \\
&\leq \mathop{\mathbb{E}}_{y\sim\mathsf{Label}}[l(\alpha^*, y)] + \frac{\epsilon}{64} \\
&= \min_{\alpha\in\{0,1\}^{\leq b(n)}} \mathop{\mathbb{E}}_{y\sim\mathsf{Label}}[l(\alpha, y)] + \frac{\epsilon}{64}.
\end{aligned}$$

Let B be the event that there exists $y \in \{0,1\}^{\leq b(n)}$ such that $\mathsf{Label}(y) - \epsilon/(64c(n)2^{b(n)+1}) \leq \tilde{p}_y \leq$

$\mathsf{Label}(y) + \epsilon/(64c(n)2^{b(n)+1})$ does not hold. Then, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathsf{Label},y\sim\mathsf{Label}}[l(L_{cheat}^{\mathsf{Label}}(1^n, 1^{\epsilon^{-1}}), y)] &\leq \Pr[\neg B] \cdot \left(\min_{\alpha\in\{0,1\}^{\leq b(n)}} \mathop{\mathbb{E}}_{y\sim\mathsf{Label}}[l(\alpha, y)] + \frac{\epsilon}{64} \right) + \Pr[B] \cdot \max_{\alpha,y\in\{0,1\}^{\leq b(n)}} l(\alpha, y) \\
&\leq 1 \cdot \left(\min_{\alpha\in\{0,1\}^{\leq b(n)}} \mathop{\mathbb{E}}_{y\sim\mathsf{Label}}[l(\alpha, y)] + \frac{\epsilon}{64} \right) + \frac{\epsilon}{64c(n)} \cdot c(n) \\
&= \min_{\alpha\in\{0,1\}^{\leq b(n)}} \mathop{\mathbb{E}}_{y\sim\mathsf{Label}}[l(\alpha, y)] + \frac{\epsilon}{32}.
\end{aligned}$$

□

References

[ABX08] Benny Applebaum, Boaz Barak, and David Xiao. "On Basing Lower-Bounds for Learning on Worst-Case Assumptions". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2008, pp. 211–220. DOI: 10.1109/FOCS.2008.35.

[AC15] Dana Angluin and Dongqu Chen. "Learning a Random DFA from Uniform Strings and State Information". In: *Algorithmic Learning Theory - 26th International Conference, ALT 2015, Banff, AB, Canada, October 4-6, 2015, Proceedings*. Ed. by Kamalika Chaudhuri, Claudio Gentile, and Sandra Zilles. Vol. 9355. Lecture Notes in Computer Science. Springer, 2015, pp. 119–133. DOI: 10.1007/978-3-319-24486-0_8. URL: https://doi.org/10.1007/978-3-319-24486-0%5C_8.

[ACMTV21] Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. "One-Way Functions and a Conditional Variant of MKTP". In: *Proceedings of the Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. 2021, 7:1–7:19. DOI: 10.4230/LIPIcs.FSTTCS.2021.7.

[AF09] Luis Filipe Coelho Antunes and Lance Fortnow. "Worst-Case Running Times for Average-Case Algorithms". In: *Proceedings of the Conference on Computational Complexity (CCC)*. 2009, pp. 298–303. DOI: 10.1109/CCC.2009.12.

[AFMV06] Luis Antunes, Lance Fortnow, Dieter van Melkebeek, and N. V. Vinodchandran. "Computational depth: Concept and applications". In: *Theor. Comput. Sci.* 354.3 (2006), pp. 391–404. DOI: 10.1016/j.tcs.2005.11.033.

[AFPS12] Luis Filipe Coelho Antunes, Lance Fortnow, Alexandre Pinto, and Andre Souto. "Low-Depth Witnesses are Easy to Find". In: *Comput. Complex.* 21.3 (2012), pp. 479–497. DOI: 10.1007/s00037-011-0025-1.

[AGMMM18] Eric Allender, Joshua A. Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. "Minimum Circuit Size, Graph Isomorphism, and Related Problems". In: *SIAM J. Comput.* 47.4 (2018), pp. 1339–1372. DOI: 10.1137/17M1157970.

[AK95] Dana Angluin and Michael Kharitonov. "When Won't Membership Queries Help?" In: *J. Comput. Syst. Sci.* 50.2 (1995), pp. 336–355. DOI: 10.1006/jcss.1995.1026.

[BCKRS22] Eric Binnendyk, Marco Carmosino, Antonina Kolokolova, Ramyaa Ramyaa, and Manuel Sabin. "Learning with distributional inverters". In: *The 33rd International Conference of Algorithmic Learning Theory (ALT2022)*. Proceedings of Machine Learning Research. PMLR, 2022.

[Ben88] C. H. Bennett. "Logical Depth and Physical Complexity". In: *The universal Turing machine, a half century survey* (1988), pp. 227–257.

[BFKL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. "Cryptographic Primitives Based on Hard Learning Problems". In: *Proceedings of the International Cryptology Conference (CRYPTO)*. 1993, pp. 278–291. DOI: 10.1007/3-540-48329-2_24.

[BT06] Andrej Bogdanov and Luca Trevisan. "On Worst-Case to Average-Case Reductions for NP Problems". In: *SIAM J. Comput.* 36.4 (2006), pp. 1119–1159. DOI: 10.1137/S0097539705446974.

[CFGMW78] Larry Carter, Robert W. Floyd, John Gill, George Markowsky, and Mark N. Wegman. "Exact and Approximate Membership Testers". In: *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*. Ed. by Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho. ACM, 1978, pp. 59–65. DOI: 10.1145/800133.804332. URL: https://doi.org/10.1145/800133.804332.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. "Learning Algorithms from Natural Proofs". In: *Proceedings of the Conference on Computational Complexity (CCC)*. 2016, 10:1–10:24. DOI: 10.4230/LIPIcs.CCC.2016.10.

[CIKK17] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. "Agnostic Learning from Tolerant Natural Proofs". In: *Proceedings of the Approximation, Randomization, and Combinatorial Optimization (APPROX/RANDOM)*. 2017, 35:1–35:19. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2017.35.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)* Wiley, 2006. ISBN: 978-0-471-24195-9.

[DP12] Ivan Damgård and Sunoo Park. "Is Public-Key Encryption Based on LPN Practical?" In: *IACR Cryptol. ePrint Arch.* (2012), p. 699. URL: http://eprint.iacr.org/2012/699.

[DV21] Amit Daniely and Gal Vardi. "From Local Pseudorandom Generators to Hardness of Learning". In: *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*. 2021, pp. 1358–1394.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions". In: *J. ACM* 33.4 (1986), pp. 792–807. DOI: 10.1145/6490.6503.

[GKK08] Parikshit Gopalan, Adam Tauman Kalai, and Adam R. Klivans. "Agnostically learning decision trees". In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*. Ed. by Cynthia Dwork. ACM, 2008, pp. 527–536. DOI: 10.1145/1374376.1374451. URL: https://doi.org/10.1145/1374376.1374451.

[GKLO22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor Carboni Oliveira. "Probabilistic Kolmogorov Complexity with Applications to Average-Case Complexity". In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Ed. by Shachar Lovett. Vol. 234. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 16:1–16:60. DOI: 10.4230/LIPIcs.CCC.2022.16. URL: https://doi.org/10.4230/LIPIcs.CCC.2022.16.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. ISBN: 0-521-79172-3. DOI: 10.1017/CBO9780511546891.

[GS86] Shafi Goldwasser and Michael Sipser. "Private Coins versus Public Coins in Interactive Proof Systems". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 1986, pp. 59–68. DOI: 10.1145/12130.12137.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function". In: *SIAM J. Comput.* 28.4 (1999), pp. 1364–1396. DOI: 10.1137/S0097539793244708.

[HILNO23] Shuichi Hirahara, Rahul Ilango, Zhenjian Lu, Mikito Nanashima, and Igor Carboni Oliveira. "A Duality Between One-Way Functions and Average-Case Symmetry of Information". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2023.

[Hir21] Shuichi Hirahara. "Average-case hardness of NP from exponential worst-case hardness assumptions". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2021, pp. 292–302. DOI: 10.1145/3406325.3451065.

[Hir22] Shuichi Hirahara. "NP-Hardness of Learning Programs and Partial MCSP". In: *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*. 2022.

[Hir23] Shuichi Hirahara. "Capturing One-Way Functions via NP-Hardness of Meta-Complexity". In: *55th Annual ACM Symposium on Theory of Computing, STOC*. 2023.

[HKLM22] Max Hopkins, Daniel M. Kane, Shachar Lovett, and Gaurav Mahajan. "Realizable Learning is All You Need". In: *Conference on Learning Theory, 2-5 July 2022, London, UK*. Ed. by Po-Ling Loh and Maxim Raginsky. Vol. 178. Proceedings of Machine Learning Research. PMLR, 2022, pp. 3015–3069. URL: https://proceedings.mlr.press/v178/hopkins22a.html.

[HKLW88] David Haussler, Michael J. Kearns, Nick Littlestone, and Manfred K. Warmuth. "Equivalence of Models for Polynomial Learnability". In: *Proceedings of the First Annual Workshop on Computational Learning Theory, COLT '88, Cambridge, MA, USA, August 3-5, 1988*. Ed. by David Haussler and Leonard Pitt. ACM/MIT, 1988, pp. 42–55. URL: http://dl.acm.org/citation.cfm?id=93040.

[HN21] Shuichi Hirahara and Mikito Nanashima. "On Worst-Case Learning in Relativized Heuristica". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 751–758. DOI: 10.1109/FOCS52979.2021.00078.

[HN22] Shuichi Hirahara and Mikito Nanashima. “Finding Errorless Pessiland in Error-Prone Heuristica”. In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Ed. by Shachar Lovett. Vol. 234. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 25:1–25:28. DOI: 10.4230/LIPIcs.CCC.2022.25. URL: https://doi.org/10.4230/LIPIcs.CCC.2022.25.

[Hut05] Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Berlin: Springer, 2005, 300 pages. ISBN: 3-540-22139-5. DOI: 10.1007/b138233. URL: http://www.hutter1.net/ai/uaibook.htm.

[IL89] Russell Impagliazzo and Michael Luby. “One-way Functions are Essential for Complexity Based Cryptography (Extended Abstract)”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 1989, pp. 230–235. DOI: 10.1109/SFCS.1989.63483.

[IL90] Russell Impagliazzo and Leonid A. Levin. “No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 1990, pp. 812–821. DOI: 10.1109/FSCS.1990.89604.

[Imp92] Russell Impagliazzo. “Pseudo-random generators for cryptography and for randomized algorithms”. PhD thesis. University of California, Berkeley, 1992.

[Imp95] Russell Impagliazzo. “A Personal View of Average-Case Complexity”. In: *Proceedings of the Structure in Complexity Theory Conference*. 1995, pp. 134–147. DOI: 10.1109/SCT.1995.514853.

[IRS21] Rahul Ilango, Hanlin Ren, and Rahul Santhanam. “Hardness on any Samplable Distribution Suffices: New Characterizations of One-Way Functions by Meta-Complexity”. In: *Electron. Colloquium Comput. Complex.* 28 (2021), p. 82.

[IRS22] Rahul Ilango, Hanlin Ren, and Rahul Santhanam. “Robustness of average-case meta-complexity via pseudorandomness”. In: *STOC ’22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. Ed. by Stefano Leonardi and Anupam Gupta. ACM, 2022, pp. 1575–1583. DOI: 10.1145/3519935.3520051. URL: https://doi.org/10.1145/3519935.3520051.

[Jac97] Jeffrey C. Jackson. “An Efficient Membership-Query Algorithm for Learning DNF with Respect to the Uniform Distribution”. In: *J. Comput. Syst. Sci.* 55.3 (1997), pp. 414–440. DOI: 10.1006/jcss.1997.1533.

[JLSW11] Jeffrey C. Jackson, Homin K. Lee, Rocco A. Servedio, and Andrew Wan. “Learning random monotone DNF”. In: *Discret. Appl. Math.* 159.5 (2011), pp. 259–271. DOI: 10.1016/j.dam.2010.08.022. URL: https://doi.org/10.1016/j.dam.2010.08.022.

[JS05] Jeffrey C. Jackson and Rocco A. Servedio. “Learning Random Log-Depth Decision Trees under Uniform Distribution”. In: *SIAM J. Comput.* 34.5 (2005), pp. 1107–1128. DOI: 10.1137/S0097539704444555. URL: https://doi.org/10.1137/S0097539704444555.

[Kha93] Michael Kharitonov. "Cryptographic hardness of distribution-specific learning". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 1993, pp. 372–381. DOI: 10.1145/167088.167197.

[KK09] Adam Kalai and Varun Kanade. "Potential-Based Agnostic Boosting". In: *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*. Ed. by Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams, and Aron Culotta. Curran Associates, Inc., 2009, pp. 880–888. URL: https://proceedings.neurips.cc/paper/2009/hash/13f9896df61279c928f19721878fac41-Abstract.html.

[KKMS08] Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. "Agnostically Learning Halfspaces". In: *SIAM J. Comput.* 37.6 (2008), pp. 1777–1805. DOI: 10.1137/060649057. URL: https://doi.org/10.1137/060649057.

[KLW10] Adam R. Klivans, Homin K. Lee, and Andrew Wan. "Mansour's Conjecture is True for Random DNF Formulas". In: *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*. Ed. by Adam Tauman Kalai and Mehryar Mohri. Omnipress, 2010, pp. 368–380. URL: http://colt2010.haifa.il.ibm.com/papers/COLT2010proceedings.pdf%5C#page=376.

[KM93] Eyal Kushilevitz and Yishay Mansour. "Learning Decision Trees Using the Fourier Spectrum". In: *SIAM J. Comput.* 22.6 (1993), pp. 1331–1348. DOI: 10.1137/0222080. URL: https://doi.org/10.1137/0222080.

[KMRRSS94] Michael J. Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. "On the learnability of discrete distributions". In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*. Ed. by Frank Thomson Leighton and Michael T. Goodrich. ACM, 1994, pp. 273–282. DOI: 10.1145/195058.195155. URL: https://doi.org/10.1145/195058.195155.

[Ko91] Ker-I Ko. "On the Complexity of Learning Minimum Time-Bounded Turing Machines". In: *SIAM J. Comput.* 20.5 (1991), pp. 962–986. DOI: 10.1137/0220059.

[KS09] Adam R. Klivans and Alexander A. Sherstov. "Cryptographic hardness for learning intersections of halfspaces". In: *J. Comput. Syst. Sci.* 75.1 (2009), pp. 2–12. DOI: 10.1016/j.jcss.2008.07.008. URL: https://doi.org/10.1016/j.jcss.2008.07.008.

[KSS94] Michael J. Kearns, Robert E. Schapire, and Linda Sellie. "Toward Efficient Agnostic Learning". In: *Mach. Learn.* 17.2-3 (1994), pp. 115–141. DOI: 10.1007/BF00993468. URL: https://doi.org/10.1007/BF00993468.

[KV94] Michael J. Kearns and Leslie G. Valiant. "Cryptographic Limitations on Learning Boolean Formulae and Finite Automata". In: *J. ACM* 41.1 (1994), pp. 67–95. DOI: 10.1145/174644.174647.

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. "Constant Depth Circuits, Fourier Transform, and Learnability". In: *J. ACM* 40.3 (1993), pp. 607–620. DOI: 10.1145/174130.174138.

[LO22] Zhenjian Lu and Igor Carboni Oliveira. “Theory and Applications of Probabilistic Kolmogorov Complexity”. In: *Bull. EATCS* 137 (2022). URL: http://bulletin.eatcs.org/index.php/beatcs/article/view/700.

[LOZ22] Zhenjian Lu, Igor Carboni Oliveira, and Marius Zimand. “Optimal Coding Theorems in Time-Bounded Kolmogorov Complexity”. In: *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*. 2022, 92:1–92:14. DOI: 10.4230/LIPIcs.ICALP.2022.92.

[LP20] Yanyi Liu and Rafael Pass. “On One-way Functions and Kolmogorov Complexity”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 1243–1254.

[LP21a] Yanyi Liu and Rafael Pass. “A Note on One-way Functions and Sparse Languages”. In: *Electron. Colloquium Comput. Complex.* (2021), p. 92.

[LP21b] Yanyi Liu and Rafael Pass. “Cryptography from sublinear-time average-case hardness of time-bounded Kolmogorov complexity”. In: *STOC ’21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*. Ed. by Samir Khuller and Virginia Vassilevska Williams. ACM, 2021, pp. 722–735. DOI: 10.1145/3406325.3451121. URL: https://doi.org/10.1145/3406325.3451121.

[LP21c] Yanyi Liu and Rafael Pass. “On the Possibility of Basing Cryptography on EXP$\neq$ BPP”. In: *Proceedings of the International Cryptology Conference (CRYPTO)*. 2021, pp. 11–40. DOI: 10.1007/978-3-030-84242-0_2.

[LP22] Yanyi Liu and Rafael Pass. “On One-Way Functions from NP-Complete Problems”. In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Ed. by Shachar Lovett. Vol. 234. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 36:1–36:24. DOI: 10.4230/LIPIcs.CCC.2022.36. URL: https://doi.org/10.4230/LIPIcs.CCC.2022.36.

[LV19] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition*. Texts in Computer Science. Springer, 2019. ISBN: 978-3-030-11297-4. DOI: 10.1007/978-3-030-11298-1.

[LV89] Ming Li and Paul M. B. Vitányi. “A Theory of Learning Simple Concepts Under Simple Distributions and Average Case Complexity for the Universal Distribution (Extended Abstract)”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 1989, pp. 34–39. DOI: 10.1109/SFCS.1989.63452.

[MF98] Neri Merhav and Meir Feder. “Universal Prediction”. In: *IEEE Trans. Inf. Theory* 44.6 (1998), pp. 2124–2147. DOI: 10.1109/18.720534. URL: https://doi.org/10.1109/18.720534.

[Nan20] Mikito Nanashima. “Extending Learnability to Auxiliary-Input Cryptographic Primitives and Meta-PAC Learning”. In: *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*. 2020, pp. 2998–3029.

[Nan21a] Mikito Nanashima. "A Theory of Heuristic Learnability". In: *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*. Ed. by Mikhail Belkin and Samory Kpotufe. Vol. 134. Proceedings of Machine Learning Research. PMLR, 2021, pp. 3483–3525. URL: http://proceedings.mlr.press/v134/nanashima21a.html.

[Nan21b] Mikito Nanashima. "On Basing Auxiliary-Input Cryptography on NP-Hardness via Nonadaptive Black-Box Reductions". In: *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*. 2021, 29:1–29:15. DOI: 10.4230/LIPIcs.ITCS.2021.29.

[NR06] Moni Naor and Guy N. Rothblum. "Learning to impersonate". In: *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*. Ed. by William W. Cohen and Andrew W. Moore. Vol. 148. ACM International Conference Proceeding Series. ACM, 2006, pp. 649–656. DOI: 10.1145/1143844.1143926. URL: https://doi.org/10.1145/1143844.1143926.

[NR09] Moni Naor and Guy N. Rothblum. "The complexity of online memory checking". In: *J. ACM* 56.1 (2009), 2:1–2:46. DOI: 10.1145/1462153.1462155.

[NY19] Moni Naor and Eylon Yogev. "Bloom Filters in Adversarial Environments". In: *ACM Trans. Algorithms* 15.3 (2019), 35:1–35:30. DOI: 10.1145/3306193. URL: https://doi.org/10.1145/3306193.

[OS17] Igor Carboni Oliveira and Rahul Santhanam. "Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness". In: *Proceedings of the Computational Complexity Conference (CCC)*. 2017, 18:1–18:49. DOI: 10.4230/LIPIcs.CCC.2017.18.

[Ost91] Rafail Ostrovsky. "One-Way Functions, Hard on Average Problems, and Statistical Zero-Knowledge Proofs". In: *Proceedings of the Structure in Complexity Theory Conference*. 1991, pp. 133–138. DOI: 10.1109/SCT.1991.160253.

[OW93] Rafail Ostrovsky and Avi Wigderson. "One-Way Fuctions are Essential for Non-Trivial Zero-Knowledge". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 1993, pp. 3–17. DOI: 10.1109/ISTCS.1993.253489.

[PV88] Leonard Pitt and Leslie G. Valiant. "Computational limitations on learning from examples". In: *J. ACM* 35.4 (1988), pp. 965–984. DOI: 10.1145/48014.63140.

[Reg09] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *J. ACM* 56.6 (2009), 34:1–34:40. DOI: 10.1145/1568318.1568324.

[RS21] Hanlin Ren and Rahul Santhanam. "Hardness of KT Characterizes Parallel Cryptography". In: *Proceedings of the Computational Complexity Conference (CCC)*. 2021, 35:1–35:58. DOI: 10.4230/LIPIcs.CCC.2021.35.

[SB14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014. ISBN: 978-1-10-705713-5. URL: http://www.cambridge.org/de/academic/subjects/computer-science/pattern-recognition-and-machine-learning/understanding-machine-learning-theory-algorithms.

[Sch90] Robert E. Schapire. “The Strength of Weak Learnability”. In: *Mach. Learn.* 5 (1990), pp. 197–227. DOI: 10.1007/BF00116037.

[Sel08] Linda Sellie. “Learning Random Monotone DNF Under the Uniform Distribution”. In: *21st Annual Conference on Learning Theory - COLT 2008, Helsinki, Finland, July 9-12, 2008*. Ed. by Rocco A. Servedio and Tong Zhang. Omnipress, 2008, pp. 181–192. URL: http://colt2008.cs.helsinki.fi/papers/76-Sellie.pdf.

[Sel09] Linda Sellie. “Exact learning of random DNF over the uniform distribution”. In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*. Ed. by Michael Mitzenmacher. ACM, 2009, pp. 45–54. DOI: 10.1145/1536414.1536424. URL: https://doi.org/10.1145/1536414.1536424.

[Sol64a] Ray J. Solomonoff. “A Formal Theory of Inductive Inference. Part I”. In: *Inf. Control.* 7.1 (1964), pp. 1–22. DOI: 10.1016/S0019-9958(64)90223-2.

[Sol64b] Ray J. Solomonoff. “A Formal Theory of Inductive Inference. Part II”. In: *Inf. Control.* 7.2 (1964), pp. 224–254. DOI: 10.1016/S0019-9958(64)90131-7.

[SU05] Ronen Shaltiel and Christopher Umans. “Simple extractors for all min-entropies and a new pseudorandom generator”. In: *J. ACM* 52.2 (2005), pp. 172–216. DOI: 10.1145/1059513.1059516.

[Val84] Leslie G. Valiant. “A Theory of the Learnable”. In: *Commun. ACM* 27.11 (1984), pp. 1134–1142. DOI: 10.1145/1968.1972.

[VZ12] Salil P. Vadhan and Colin Jia Zheng. “Characterizing pseudoentropy and simplifying pseudorandom generator constructions”. In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2012, pp. 817–836. DOI: 10.1145/2213977.2214051.

[VZ13] Salil P. Vadhan and Colin Jia Zheng. “A Uniform Min-Max Theorem with Applications in Cryptography”. In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*. 2013, pp. 93–110. DOI: 10.1007/978-3-642-40041-4_6.

[Xia10] David Xiao. “Learning to Create is as Hard as Learning to Appreciate”. In: *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*. Ed. by Adam Tauman Kalai and Mehryar Mohri. Omnipress, 2010, pp. 516–528. URL: http://colt2010.haifa.il.ibm.com/papers/COLT2010proceedings.pdf%5C#page=524.

[Yao82] Andrew Chi-Chih Yao. “Theory and Applications of Trapdoor Functions (Extended Abstract)”. In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 1982, pp. 80–91. DOI: 10.1109/SFCS.1982.45.

[ZL70] Alexander K Zvonkin and Leonid A Levin. “The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms”. In: *Russian Mathematical Surveys* 25.6 (1970), pp. 83–124.

A Lower Bound on Sample Complexity in Agnostic Learning on Average

In this section, we show that the dependence of s and ϵ^{-1} on the sample complexity of our agnostic learner (in Theorems 2.3 and 10.6) is optimal, particularly for learning parities on average with noise over unit vectors. The proof is essentially same as that of the fundamental theorem of statistical learning (cf. [SB14, Sections 6 and 28]), except that we consider a natural average-case analogue of shattered sets.

We review the problem of learning parities with noise (LPN). For every $\alpha \in \{0,1\}^n$, let $\chi_\alpha \colon \{0,1\}^n \to \{0,1\}$ denote a parity function defined as $\chi_\alpha(x) = \langle x, \alpha\rangle_{\mathbb{F}_2}$, where $\langle,\rangle_{\mathbb{F}_2}$ represents the inner product on $\mathbb{F}_2$. For each $n \in \mathbb{N}$, $\alpha \in \{0,1\}^n$, $\gamma \in [0,1/2]$, and $S \subseteq \{0,1\}^n$, we define a distribution $\mathsf{LPN}^S_{\alpha,\gamma}$ over samples in $\{0,1\}^n \times \{0,1\}$ as the distribution of $(x, \chi_\alpha(x) \oplus \xi)$ for $x \sim S$ and $\xi \sim \mathsf{Ber}(1/2-\gamma)$, where $\mathsf{Ber}(1/2-\gamma)$ represents the Bernoulli distribution with parameter $1/2-\gamma$, i.e., $\xi = 1$ (resp. $\xi = 0$) with probability $1/2-\gamma$ (resp. $1/2+\gamma$). It is easily verified that, for every $\gamma \in [0,1/2]$ and every $S \subseteq \{0,1\}^n$,

$$\mathsf{opt}_{\mathcal{F}}(\mathsf{LPN}^S_{\alpha,\gamma}) = \min_{f \colon \{0,1\}^n \to \{0,1\}} \Pr_{(x,b)\sim \mathsf{LPN}^S_{\alpha,\gamma}}[f(x) \neq b] = \frac{1}{2} - \gamma.$$

We mainly focus on the specific subset $S^e_n = \{e^1, \ldots, e^{\lfloor n/2 \rfloor}\} \subseteq \{0,1\}^n$, where each $e^j \in \{0,1\}^n$ is the j-th unit vector, i.e., $e^j_i = 1$ iff $i = j$. For every $c \in \mathbb{N}$, the distribution $\mathsf{LPN}^{S^e_{|\alpha|}}_{\alpha,2^{-c}}$ is samplable in time $O(|\alpha| \cdot c)$ when α is given as advice.

We show the following matching lower bound on sample complexity for LPN over S^e_n.

Theorem A.1. *Suppose that a (possibly not efficient) agnostic learner L satisfies that for every sufficiently large $n \in \mathbb{N}$, every $\epsilon^{-1} \in \mathbb{N}$, and every $c \in \mathbb{N}$ with $c \leq \log^2 n$,*

$$\Pr_{\alpha\sim\{0,1\}^n, \mathsf{LPN}^{S^e_n}_{\alpha,2^{-c}}}\left[L^{\mathsf{LPN}^{S^e_n}_{\alpha,2^{-c}}}(1^n, 1^{\epsilon^{-1}}) \text{ outputs } h \text{ s.t. } \Pr_{(x,b)\sim\mathsf{LPN}^{S^e_n}_{\alpha,2^{-c}}}[h(x) \neq b] \leq \left(\frac{1}{2} - 2^{-c}\right) + \epsilon\right] \geq \frac{7}{8}.$$

Then, the sample complexity m_L of L must satisfy $m_L(n,\epsilon) = \Omega(n\epsilon^{-2})$ (note that the secret information is represented by $s := |\alpha| = n$ bits).

Theorem A.1 is obtained in the same way as the fundamental theorem of statistical learning, where we use the simple observation that parity functions shatter S^e_n *on average* over the choice of parity functions (instead of the argument of the VC dimension). Here, we only present the proof outline. For the detailed argument, we refer the reader to the proof in [SB14, Section 28.2.2].

Proof sketch. Suppose that the sample complexity $m_L(n,\epsilon)$ satisfies $m_L(n,\epsilon) < 8\lfloor n/2 \rfloor \epsilon^{-2}$. Below, we derive a contradiction to show $m_L(n,\epsilon) \geq 8\lfloor n/2\rfloor\epsilon^{-2} = \Omega(n\epsilon^{-2})$. For readability, we omit the superscript S^e_n from $\mathsf{LPN}^{S^e_n}_{\alpha,\rho/2}$ in this proof.

Fix sufficiently large $n \in \mathbb{N}$ and $k \in \mathbb{N}$ with $k \in (\log 8\sqrt{2}, \log^2 n + 2)$ arbitrarily. Let $m = \lfloor n/2 \rfloor$. Let $\epsilon = 2^{-k}$ and $\rho = 8\epsilon$ (note that $\rho < 1/\sqrt{2}$). Let $c = -\log\rho = k-3$. Since $c + 1 \leq \log^2 n$, the learner L satisfies

$$\Pr_{\alpha\sim\{0,1\}^n, \mathsf{LPN}_{\alpha,\rho/2}}\left[L^{\mathsf{LPN}_{\alpha,\rho/2}}(1^n, 1^{\epsilon^{-1}}) \text{ outputs } h \text{ s.t. } \Pr_{(x,b)\sim\mathsf{LPN}_{\alpha,\rho/2}}[h(x) \neq b] \leq (1+\rho)/2 + \epsilon\right] \geq \frac{7}{8},$$

and $L(1^n, 1^{\epsilon^{-1}})$ makes only at most $8m\epsilon^{-2}$ queries. Without loss of generality, we assume that $L(1^n, 1^{\epsilon^{-1}})$ makes *exactly* $M := 8m\epsilon^{-2}$ queries, and M samples $S_\alpha := \{(x^i, \chi_\alpha(x^i) \oplus \xi^i)\}_{i=1}^M$ are given as auxiliary input, where $\alpha \sim \{0,1\}^n$, $x^i \sim S_n^e$, and $\xi^i \sim \mathsf{Ber}((1+\rho)/2)$ for each i.

For each $\alpha \in \{0,1\}^n$, let $\mathsf{opt}_\alpha := \mathsf{opt}_{\mathcal{F}}(\mathsf{LPN}_{\alpha,\rho/2}) = (1-\rho)/2$. For each hypothesis $h\colon \{0,1\}^n \to \{0,1\}$, let $\ell_\alpha(h)$ be the error probability of h, i.e.,

$$\ell_\alpha(h) := \Pr_{(x,b)\sim \mathsf{LPN}_{\alpha,\rho/2}}[h(x) \neq b] = \frac{1+\rho}{2} \cdot \frac{|\{i \in [m] : h(e^i) \neq \alpha_i\}|}{m} + \frac{1-\rho}{2} \cdot \frac{|\{i \in [m] : h(e^i) = \alpha_i\}|}{m}.$$

Therefore,

$$\ell_\alpha(h) - \mathsf{opt}_\alpha = \rho \cdot \frac{|\{i \in [m] : h(e^i) \neq \alpha_i\}|}{m}. \tag{14}$$

Let $L(S_\alpha)$ denote the hypothesis produced by $L(1^n, 1^{\epsilon^{-1}})$ given a sample set S_α. Then, we have

$$\Pr_{\alpha, S_\alpha}[\ell_\alpha(L(S_\alpha)) - \mathsf{opt}_\alpha > \epsilon] \leq 1/8.$$

Now, we consider another learning algorithm L^* that is given S_α and produces a hypothesis $L^*(S_\alpha)\colon S_n^e \to \{0,1\}$ such that, for each $e^i \in S_n^e$, the value of $L^*(S_\alpha)(e^i)$ is the majority among the labels of e^i in the sample set S_α (breaking ties arbitrarily). Then, L^* is the optimal learner (cf. [SB14, Lemma 28.1]), i.e.,

$$\Pr_{\alpha, S_\alpha}[\ell_\alpha(L^*(S_\alpha)) - \mathsf{opt}_\alpha > \epsilon] \leq \Pr_{\alpha, S_\alpha}[\ell_\alpha(L(S_\alpha)) - \mathsf{opt}_\alpha > \epsilon] \leq 1/8. \tag{15}$$

Furthermore,

$$\begin{aligned}\mathop{\mathbb{E}}_{\alpha, S_\alpha}[\ell_\alpha(L^*(S_\alpha)) - \mathsf{opt}_\alpha] &= \frac{\rho}{m} \mathop{\mathbb{E}}_{\alpha, S_\alpha}\left[|\{i \in [m] : L^*(S_\alpha)(e^i) \neq \alpha_i\}|\right] \\ &= \frac{\rho}{m} \sum_{i=1}^m \Pr_{\alpha, S_\alpha}[L^*(S_\alpha)(e^i) \neq \alpha_i].\end{aligned}$$

By Slud's inequality and careful calculations (see [SB14, Section 28.2.2]), the right-hand side is bounded below as

$$\frac{\rho}{m} \sum_{i=1}^m \Pr_{\alpha, S_\alpha}[L^*(S_\alpha)(e^i) \neq \alpha_i] \geq \frac{\rho}{2}\left(1 - \sqrt{2\rho^2 M/m}\right),$$

where we use the fact that $\rho < 1/\sqrt{2}$.

Since $M < 8m\epsilon^{-2} = m/(8\rho^2)$,

$$\mathop{\mathbb{E}}_{\alpha, S_\alpha}[\ell_\alpha(L^*(S_\alpha)) - \mathsf{opt}_\alpha] \geq \frac{\rho}{2}\left(1 - \sqrt{2\rho^2 M/m}\right) > \frac{\rho}{2}\left(1 - \frac{\rho}{2}\right) = \frac{\rho}{2} - \frac{\rho^2}{4} \geq \frac{\rho}{4} = 2\epsilon.$$

By equation (14), it holds that $\ell_\alpha(L^*(S_\alpha)) - \mathsf{opt}_\alpha \leq \rho$ for every α and S_α. Therefore, we have

$$\Pr_{\alpha, S_\alpha}[\ell_\alpha(L^*(S_\alpha)) - \mathsf{opt}_\alpha > \epsilon] > \frac{1}{8}; \tag{16}$$

otherwise,

$$\mathop{\mathbb{E}}_{\alpha, S_\alpha}[\ell_\alpha(L^*(S_\alpha)) - \mathsf{opt}_\alpha] \leq \epsilon \cdot 1 + \frac{1}{8} \cdot \rho = 2\epsilon.$$

Inequality (16) contradicts inequality (15). $\square$

B Universal Distribution and Probabilistic Kolmogorov Complexity

In this section, we show that q^t (in Definition 2.5) is equivalent to the time-bounded probabilistic Kolmogorov complexity pK^t up to an additive logarithmic factor and a polynomial overhead of the time-bound. The latter notion was recently studied by Goldberg, Kabanets, Lu, and Oliveira [GKLO22] in the context of meta-complexity. Further background on probabilistic Kolmogorov complexity can be found in [LO22].

For future work, we consider a general case in which an auxiliary advice string is given. First, we extend the definition of q^t to such a case.

Definition B.1 (implicit in [IL90])**.** *For every $t \in \mathbb{N}$ and every $z \in \{0,1\}^*$, we define the t-time-bounded universal distribution Q^t_z given z as the distribution of $U^t(r;z)$ for $r \sim \{0,1\}^t$, where the universal Turing machine U is given query access to each bit of z.*

For every $t \in \mathbb{N}$ and every $x, z \in \{0,1\}^*$, we define $\mathrm{q}^t(x|z)$ as

$$\mathrm{q}^t(x|z) = -\log \mathrm{Q}^t_z(x).$$

(If $\mathrm{Q}^t_z(x) = 0$, then we regard $\mathrm{q}^t(x|z)$ as ∞.)

Note that $\mathrm{q}^t(x|\varepsilon)$ is equal to $\mathrm{q}^t(x)$ in Definition 2.5.

The time-bounded probabilistic Kolmogorov complexity pK^t is defined as follows.

Definition B.2 (Probabilistic Kolmogorov complexity [GKLO22])**.** *For every $t \in \mathbb{N}$ and every $x, z \in \{0,1\}^*$, we define the t-time-bounded Kolmogorov complexity of x given z as*

$$\mathrm{pK}^t(x|z) = \min\left\{ k \in \mathbb{N} : \Pr_{r \sim \{0,1\}^t}\left[\exists \pi \in \{0,1\}^k \text{ s.t. } U^t(\pi, r; z) = x\right] \geq 2/3 \right\},$$

where the universal Turing machine U is given query access to each bit of z. (If there is no such $p \in \{0,1\}^*$, then we regard $\mathrm{pK}^t(x|z)$ as ∞ for convenience.)

Below, we only consider the case in which $\mathrm{q}^t(x|z) < \infty$ and $\mathrm{pK}^t(x|z) < \infty$.

The equivalence between q^t and pK^t is stated as follows. Note that the second statement follows from the optimal coding theorem for pK^t proved by Lu, Oliveira, and Zimand [LOZ22].

Proposition B.3. *For every $t \in \mathbb{N}$ and every $x, z \in \{0,1\}^*$,*

$$\mathrm{q}^{O(t)}(x|z) \leq \mathrm{pK}^t(x|z) + O(\log t).$$

Theorem B.4 ([LOZ22])**.** *For every $t \in \mathbb{N}$ and every $x, z \in \{0,1\}^*$,*

$$\mathrm{pK}^{p(t)}(x|z) \leq \mathrm{q}^t(x|z) + \log p(t),$$

where p is a universal polynomial that depends on only U.

Proof of Proposition B.3. By the definition of $\mathrm{pK}^t(x|z)$, there exist at least $(2/3) \cdot 2^t$ random seeds $r \in \{0,1\}^t$ that have a program $\pi_r \in \{0,1\}^{\mathrm{pK}^t(x|z)}$ such that $U^t(\pi_r, r; z) = x$. Without loss of generality, we can assume that $\mathrm{pK}^t(x|z) \leq t$. By selecting sufficiently large $t' = O(t)$, the probability that the prefix of a random seed $r' \sim \{0,1\}^{t'}$ corresponds to the program $\langle \pi_r, r \rangle$ is at least $2^{-\mathrm{pK}^t(x|z) - O(\log t) - t}$ (by the standard encoding). Therefore, we obtain $\mathrm{Q}^{t'}_z(x) \geq (2/3) \cdot 2^t \cdot 2^{-\mathrm{pK}^t(x|z) - O(\log t) - t} = 2^{-\mathrm{pK}^t(x|z) - O(\log t)}$ and $\mathrm{q}^{t'}(x|z) = -\log \mathrm{Q}^{t'}_z(x) \leq \mathrm{pK}^t(x|z) + O(\log t)$. $\square$

Theorem B.4 follows from the observation that the proof of the optimal coding theorem for pK^t in [LOZ22, Theorem 5] holds even with additional access to z.

C MINLT under Separated Distributions in Pessiland

In this section, we show that the search version of MINLT is efficiently solvable on average in the restricted setting studied in [BFKL93] under the non-existence of OWF and the standard derandomization assumption (see Theorem C.7).

First, we review the problem MINLT. We introduce additional notions. In this section, we may call a distribution over $\{0,1\}^n \times \{0,1\}$ a *sampler*. For any $m \in \mathbb{N}$ and any distribution family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{D}_n$ is a distribution on samplers over $\{0,1\}^n \times \{0,1\}$, we let $\mathcal{D}^m = \{\mathcal{D}_n^m\}_{n\in\mathbb{N}}$ denote a distribution family over sample sets such that each $\mathcal{D}_n^m$ is the distribution of a sample set $\{(x^1,b^1),\dots,(x^m,b^m)\}$, where $(x^i,b^i) \sim S$ for each $i \in [m]$ and $S \sim \mathcal{D}_n$ (note that the sampler S is selected only once). We also use the notation $(\boldsymbol{x},\boldsymbol{b}) \sim \mathcal{D}_n^m$ to indicate that $\boldsymbol{x} = (x^1,\dots,x^m)$ and $\boldsymbol{b} = (b^1,\dots,b^m)$ for $\{(x^1,b^1),\dots,(x^m,b^m)\} \sim \mathcal{D}_n^m$.

We define LT-complexity of sample sets.

Definition C.1 (LT-complexity [Ko91])**.** *Let $X = \{(x^1,b^1),\dots,(x^m,b^m)\}$ be a sample set, where $x^i \in \{0,1\}^*$ and $b^i \in \{0,1\}$ for each $i \in [m]$. For every $t \in \mathbb{N}$, the t-time-bounded LT-complexity of X is denoted by $\mathrm{LT}^t(X)$ and defined as*

$$\mathrm{LT}^t(X) := \min\{|\Pi| : \Pi \in \{0,1\}^* \textit{such that } U^t(\Pi, x^i) = b^i \textit{ for all } i \in [m]\}.$$

Now, we define MINLT as a problem that asks LT-complexity of a given sample set.

Definition C.2 (MINLT [Ko91])**.** *For every $t \in \mathbb{N}$, we define the language* $\mathrm{MINLT}[t]$ *as follows:*

$$\mathrm{MINLT}[t] = \left\{(X, 1^s) : n, m \in \mathbb{N}, X \in (\{0,1\}^n \times \{0,1\})^m, \mathrm{LT}^t(X) \le s\right\}.$$

We define the search version of $\mathrm{MINLT}[t]$ *as a problem that asks, for a given $X = \{(x^1,b^1),\dots,(x^m,b^m)\}$, where $x^i \in \{0,1\}^n$ and $b^i \in \{0,1\}$ for each $i \in [m]$, to find $\Pi \in \{0,1\}^*$ such that $|\Pi| = \mathrm{LT}^t(X)$ and $U^t(\Pi, x^i) = b^i$ for all $i \in [m]$.*

First, we present a meta-theorem. Roughly speaking, the meta-theorem shows that if there exists no infinitely-often one-way function, then we can construct an algorithm that solves $\mathrm{MINLT}[\tau]$ on average under any samplable distribution on samplers that satisfies a time-bounded LT-complexity analogue of the coding theorem.

Lemma C.3. *If there exists no infinitely-often one-way function, then for every constant $C > 0$, there exists a randomized algorithm L such that for every polynomials $t(n), \sigma(n)$, every (unknown) $t(n)$-samplable distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ on samplers, every sufficiently large $n \in \mathbb{N}$, and every $m, \delta^{-1}, \tau \in \mathbb{N}$, the algorithm $L(\text{-}; 1^{\langle n,m,\delta^{-1},t(n),\sigma(n),\tau\rangle})$ solves the search version of* $\mathrm{MINLT}[\tau]$ *on average under $\mathcal{D}^m$ with error probability at most δ as long as $\mathcal{D}$ satisfies the following: for all $\gamma^{-1} \in \mathbb{N}$,*

$$\Pr_{X=(\boldsymbol{x},\boldsymbol{b})\sim\mathcal{D}_n^m}\left[\mathrm{LT}^\tau(X) \le \min\{-\log \mathcal{D}_n^m(\boldsymbol{b}|\boldsymbol{x}) + C(\log\tau + \log\gamma^{-1}), \sigma(n)\}\right] \ge 1 - \gamma,$$

where $\mathcal{D}_n^m(\boldsymbol{b}|\boldsymbol{x}) = \Pr_{(\boldsymbol{x}',\boldsymbol{b}')\sim\mathcal{D}_n^m}[\boldsymbol{b}' = \boldsymbol{b}|\boldsymbol{x}' = \boldsymbol{x}]$.

Proof. Let $t_U(n)$ be a simulation overhead function of the universal Turing machine U; i.e., for every $t \in \mathbb{N}$, every Turing machine M, and every input $x \in \{0,1\}^*$, if $M(x)$ halts in t time, then $U^{t_U(t)}(M, x) = M(x)$.

We define a polynomial-time-computable family $f = \{f_n \colon \{0,1\}^{\mathsf{poly}(n)} \to \{0,1\}^{\mathsf{poly}(n)}\}$ as

$$f_{\langle n,m,t,\sigma,\tau\rangle}(i, j, \Pi_f, \Pi_e, s^0, s^1, \ldots, s^m) = (i, x^1, U^\tau((\Pi_f)_{[i]}, x^1), \ldots, x^m, U^\tau((\Pi_f)_{[i]}, x^m)),$$

where $i \in [\sigma]$, $j \in [n]$, $\Pi_f, \Pi_e \in \{0,1\}^n$, $s^0, s^1, \ldots, s^m \in \{0,1\}^{t_U(t)}$, and $x^1, \ldots x^m$ are determined as follows: for each $k \in [m]$,

$$S = U^{t_U(t)}((\Pi_e)_{[j]}, \langle 1^n, s^0\rangle)$$
$$x^k = \begin{cases} \mathsf{CircEval}(S, s^k) & \text{if } S \text{ is a description of a circuit of output length } n \\ 0^n & \text{otherwise,} \end{cases}$$

where $\mathsf{CircEval}$ is the standard circuit evaluation algorithm that runs in polynomial time. Without loss of generality, we assume that the circuit size of S above is at most $t_U(t)$ because its description is printed in $t_U(t)$ time.

Since we execute U in polynomial time in above, f is a polynomial-time-computable family. We assume that the input to f_n is given as a concatenated string. For each $n, m, t, \sigma, \tau \in \mathbb{N}$, let $r(n, m, t, \sigma, \tau)$ be the input size of $f_{\langle n,m,t,\sigma,\tau\rangle}$, i.e., $r(n, m, t, \sigma, \tau) = \lceil \log \sigma \rceil + \lceil \log n \rceil + 2n + t_U(t)(m + 1)$.

By Proposition 7.4 and the assumption that there exists no infinitely-often one-way function, there exists a randomized polynomial-time algorithm A such that for every $n, m, t, \sigma, \tau \in \mathbb{N}$ and every $\delta^{-1} \in \mathbb{N}$,

$$\Pr\left[A(f_{\langle n,m,t,\sigma,\tau\rangle}(U_{r(n,m,t,\sigma,\tau)}); 1^{\langle n,m,t,\sigma,\tau\rangle}, 1^{\delta^{-1}}) \notin f^{-1}_{\langle n,m,t,\sigma,\tau\rangle}(f_{\langle n,m,t,\sigma,\tau\rangle}(U_{r(n,m,t,\sigma,\tau)}))\right] \le \delta.$$

We construct a randomized algorithm L in the theorem from A. For given parameters $n, m, \delta^{-1}, t, \sigma, \tau \in \mathbb{N}$ and a sample set $X = \{(x^i, b^i)\}_{i=1}^m$, the algorithm L executes

$$A(i, x^1, b^1, \ldots, x^m, b^m; 1^{\langle n,m,t,\sigma,\tau\rangle}, 1^{\delta'^{-1}})$$

for each $i \in [\sigma]$ and for δ' defined as

$$\delta' = \frac{1}{2} \cdot \frac{(\delta/2)^C}{n^2\sigma\tau^C}\frac{\delta}{2},$$

where $C > 0$ is the constant in the theorem. For each i, if A returns some inverse element $X^i = (i, j^i, \Pi_f^i, \Pi_e^i, s^{i,0}, s^{i,1}, \ldots, s^{i,m})$, then L checks whether $f(X^i) = (i, x^1, b^1, \ldots, x^m, b^m)$. Let $i^* \in [\sigma]$ be the minimum integer i for which A succeeds in inverting (if not, A returns $\perp$ and halts). Then, L outputs $(\Pi_f^{i^*})_{[i^*]}$ as a hypothesis. It is not hard to verify that L halts in time $\mathsf{poly}(n, m, \delta^{-1}, t, \sigma, \tau)$.

To show the correctness of L, we first prove the following claim.

Claim C.4. *For any polynomials $t(n), \sigma(n)$, any $t(n)$-samplable distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ on samplers, any sufficiently large $n \in \mathbb{N}$, and any $m, \tau \in \mathbb{N}$, if $\mathcal{D}$ satisfies that for all $\gamma^{-1} \in \mathbb{N}$,*

$$\Pr_{X=(\boldsymbol{x},\boldsymbol{b})\sim\mathcal{D}_n^m}\left[\mathrm{LT}^\tau(X) \le \min\{-\log \mathcal{D}_n^m(\boldsymbol{b}|\boldsymbol{x}) + C(\log\tau + \log\gamma^{-1}), \sigma(n)\}\right] \ge 1 - \gamma,$$

then it holds that, for every $\gamma^{-1} \in \mathbb{N}$,

$$\Pr_{X=\{(x^i,b^i)\}_{i=1}^m\sim\mathcal{D}_n^m}\left[\Pr_{U_r}\left[f(U_r) = (\mathrm{LT}^\tau(X), x^1, b^1, \ldots, x^m, b^m)\right] \ge \frac{\gamma^C}{n^2\sigma(n)\tau^C} \cdot \mathcal{D}_n^m(X)\right] \ge 1 - \gamma,$$

where $r := r(n, m, t(n), \sigma(n), \tau)$.

Proof. We define a $t(n)$-samplable distribution $\mathcal{D}' = \{\mathcal{D}'_n\}_{n\in\mathbb{N}}$ such that each $\mathcal{D}'_n$ is a distribution of the first half of $S \sim \mathcal{D}_n$ (i.e., the sub-circuit that produces examples). Let M be the deterministic $t(n)$-time sampling algorithm for $\mathcal{D}'$, and let $d = |M|$.

We only consider a sufficiently large $n \in \mathbb{N}$ so that $n \geq 2^d$. Then, with probability $n^{-1} \cdot 2^{-d} \geq n^{-2}$ over the choice of $j \in [n]$ and $\Pi_e \in \{0,1\}^n$, the program $(\Pi_e)_{[j]}$ (in the input of f) corresponds to the description of M. Under this condition, the distribution of S in the computation of f is statistically identical to $\mathcal{D}'_n$, and for each $m \in \mathbb{N}$ and $(\boldsymbol{x}, \boldsymbol{b}) \in \mathrm{supp}(\mathcal{D}^m_n)$, the probability that $\boldsymbol{x}$ is sampled according to $\mathcal{D}^m_n$ (we denote this probability by $\mathcal{D}^m_n(\boldsymbol{x})$) is equivalent to the conditional probability that $\boldsymbol{x} = (x^1, \ldots, x^m)$ holds for $(i, x^1, b^1, \ldots, x^m, b^m) \sim f(U_r)$. Thus, for each $m \in \mathbb{N}$ and each $(\boldsymbol{x}, \boldsymbol{b}) \in \mathrm{supp}(\mathcal{D}^m_n)$,

$$\Pr_{U_r}\left[\boldsymbol{x} = (x^1, \ldots, x^m) \text{ for } f(U_r) = (i, x^1, b^1, \ldots, x^m, b^m)\right] \geq \frac{\mathcal{D}^m_n(\boldsymbol{x})}{n^2}.$$

Fix $\gamma^{-1} \in \mathbb{N}$ and $X = (\boldsymbol{x}, \boldsymbol{b}) \in \mathrm{supp}(\mathcal{D}^m_n)$ satisfying the following condition arbitrarily:

$$\mathrm{LT}^\tau(X) \leq \min\{-\log \mathcal{D}^m_n(\boldsymbol{b}|\boldsymbol{x}) + C(\log\tau + \log\gamma^{-1}), \sigma(n)\} \tag{17}$$

Over the choice of $i \in [\sigma(n)]$ (in the input of f), the event that $i = \mathrm{LT}^\tau(X)$ $(\leq \sigma(n))$ occurs with probability $\sigma(n)^{-1}$. We consider the event $E_{\boldsymbol{x}}$ that $i = \mathrm{LT}^\tau(X)$ and $\boldsymbol{x} = (x^1, \ldots, x^m)$ holds for $f(U_r) = (i, x^1, b^1, \ldots, x^m, b^m)$. Then, we have

$$\Pr_{f(U_r)}[E_{\boldsymbol{x}}] \geq \frac{\mathcal{D}^m_n(\boldsymbol{x})}{n^2\sigma(n)}.$$

Under the condition that $E_{\boldsymbol{x}}$ occurs, the probability that $(\Pi_f)_{[i]}$ (note that $i = \mathrm{LT}^\tau(X)$) corresponds to the program Π^*_X satisfying $|\Pi^*_X| = \mathrm{LT}^\tau(X)$ and $b^i = U^\tau(\Pi^*_X, x^i)$ for each $i \in [m]$ is

$$\begin{aligned}\Pr_{\Pi_f}\left[(\Pi_f)_{[i]} = \Pi^*_X\right] &= 2^{-\mathrm{LT}^\tau(X)} \\ &\geq 2^{\log \mathcal{D}^m_n(\boldsymbol{b}|\boldsymbol{x}) - C(\log\tau + \log\gamma^{-1})} \\ &= \frac{\gamma^C \cdot \mathcal{D}^m_n(\boldsymbol{b}|\boldsymbol{x})}{\tau^C},\end{aligned}$$

where the inequality follows from (17).

Thus, for each $X = (\boldsymbol{x}, \boldsymbol{b}) \in \mathrm{supp}(\mathcal{D}^m_n)$ (let $\boldsymbol{x} = (x^1, \ldots, x^m)$ and $\boldsymbol{b} = (b^1, \ldots, b^m)$), if $(\boldsymbol{x}, \boldsymbol{b})$ satisfies (17), then we have

$$\begin{aligned}\Pr_{U_r}\left[f(U_r) = (\mathrm{LT}^\tau(X), x^1, b^1, \ldots, x^m, b^m)\right] &\geq \Pr_{\Pi_f}\left[(\Pi_f)_{[\mathrm{LT}^\tau(X)]} = \Pi^*_X\right] \cdot \Pr_{f(U_r)}[E_{\boldsymbol{x}}] \\ &\geq \frac{\gamma^C \cdot \mathcal{D}^m_n(\boldsymbol{b}|\boldsymbol{x})}{\tau^C} \cdot \frac{\mathcal{D}^m_n(\boldsymbol{x})}{n^2\sigma(n)} \\ &= \frac{\gamma^C}{n^2\sigma(n)\tau^C} \cdot \mathcal{D}^m_n(X).\end{aligned}$$

Since $\Pr_{(\boldsymbol{x},\boldsymbol{b})\sim\mathcal{D}^m_n}[(\boldsymbol{x}, \boldsymbol{b}) \text{ satisfies } (17)] \geq 1 - \gamma$ follows from the assumption, the claim holds. $\square$

Now, we show the correctness of $L(\text{-}; 1^{\langle n,m,\delta^{-1},t(n),\sigma(n),\tau\rangle})$. Suppose that the error probability of L is grater than δ. Then, we derive a contradiction. For readability, we omit the parameters of L below.

By the assumption on $\mathcal{D}$ in the theorem and Claim C.4, we have

$$\Pr_{X=\{(x^i,b^i)\}_{i=1}^m\sim\mathcal{D}_n^m}\left[\Pr_{U_r}\left[f(U_r)=(\mathrm{LT}^\tau(X),x^1,b^1,\ldots,x^m,b^m)\right]\geq\frac{(\delta/2)^C}{n^2\sigma(n)\tau^C}\cdot\mathcal{D}_n^m(X)\right]\geq 1-\delta/2.$$

Let GoodSamp $\subseteq \mathrm{supp}(\mathcal{D}_n^m)$ be the set of samples $X\in\mathrm{supp}(\mathcal{D}_n^m)$ satisfying the event in the probability above, i.e., $\Pr_{X\sim\mathcal{D}_n^m}[X\in\mathrm{GoodSamp}]\geq 1-\delta/2$, and for each $X=\{(x^i,b^i)\}_{i=1}^m\in$ GoodSamp,

$$\Pr_{U_r}\left[f(U_r)=(\mathrm{LT}^\tau(X),x^1,b^1,\ldots,x^m,b^m)\right]\geq\frac{(\delta/2)^C}{n^2\sigma(n)\tau^C}\cdot\mathcal{D}_n^m(X).$$

For each $X=\{(x^i,b^i)\}_{i=1}^m\in\mathrm{supp}(\mathcal{D}_n^m)$, we let F_X denote the event that $L(X)$ fails in finding a program $\Pi\in\{0,1\}^*$ such that $|\Pi|=\mathrm{LT}^\tau(X)$ and $b^i=U^\tau(\Pi,x^i)$ for each $i\in[m]$. Note that F_X occurs only if $A(\mathrm{LT}^\tau(X),x^1,b^1,\ldots,x^m,b^m)$ fails in finding an inverse element.

Under the assumption that the error probability of L is greater than δ, we have

$$\begin{aligned}\Pr_{X\sim\mathcal{D}_n^m,L}[F_X\wedge X\in\mathrm{GoodSamp}]&\geq\Pr_{X\sim\mathcal{D}_n^m,L}[F_X]-\Pr_{X\sim\mathcal{D}_n^m}[X\notin\mathrm{GoodSamp}]\\&\geq\delta-\delta/2=\delta/2.\end{aligned}$$

Furthermore, we obtain

$$\begin{aligned}\delta/2&\leq\Pr_{X\sim\mathcal{D}_n^m,L}[F_X\wedge X\in\mathrm{GoodSamp}]\\&=\sum_{X\in\mathrm{GoodSamp}}\mathcal{D}_n^m(X)\cdot\Pr_L[F_X]\\&\leq\sum_{X=\{(x^i,b^i)\}_{i=1}^m\in\mathrm{GoodSamp}}\mathcal{D}_n^m(X)\cdot\Pr_A[A(\mathrm{LT}^\tau(X),x^1,b^1,\ldots,x^m,b^m)\text{ fails in inverting}]\\&\leq\sum_{X=\{(x^i,b^i)\}_{i=1}^m\in\mathrm{GoodSamp}}\frac{n^2\sigma(n)\tau^C}{(\delta/2)^C}\Pr_{U_r}\left[f(U_r)=(\mathrm{LT}^\tau(X),x^1,b^1,\ldots,x^m,b^m)\right]\\&\qquad\cdot\Pr_A[A(\mathrm{LT}^\tau(X),x^1,b^1,\ldots,x^m,b^m)\text{ fails in inverting}]\\&=\frac{n^2\sigma(n)\tau^C}{(\delta/2)^C}\Pr_{A,U_r}[A(f(U_r))\text{ fails in inverting }\wedge\exists X=\{(x^i,b^i)\}_{i=1}^m\in\mathrm{GoodSamp}\\&\qquad\text{s.t. }f(U_r)=(\mathrm{LT}^\tau(X),x^1,b^1,\ldots,x^m,b^m)]\\&\leq\frac{n^2\sigma(n)\tau^C}{(\delta/2)^C}\Pr_{A,U_r}[A(f(U_r))\text{ fails in inverting}].\end{aligned}$$

Therefore, the failure probability of A is at least

$$\frac{\delta}{2}\cdot\frac{(\delta/2)^C}{n^2\sigma(n)\tau^C}=2\delta'.$$

This is a contradiction because the failure probability of A is at most δ' by the choice of the parameter for A in L. □

Next, we show that the coding theorem in Lemma C.3 holds if a distribution on samplers is separated into two distributions $\mathcal{E}$ and $\mathcal{F}$ over examples and a target function (as the model studied in [BFKL93]) under the following derandomization assumption for nondeterministic circuits.

Hypothesis C.5 (Pseudorandom generator against nondeterministic circuits)**.** *There exists a constant $C_0 > 0$ such that for every polynomial m, there exists a $\mathsf{poly}(m(n))$-time computable pseudorandom generator $G = \{G_n\}_{n\in\mathbb{N}}$, where $G_n\colon \{0,1\}^{C_0 \log m(n)} \to \{0,1\}^{m(n)}$ that $1/m(n)$-fools $m(n)$-size nondeterministic circuits, i.e., for every $m(n)$-size nondeterministic circuit C and every $n \in \mathbb{N}$;*

$$\left| \Pr_{z\sim\{0,1\}^{C_0 \log m(n)}}[C(G(z)) = 1] - \Pr_{w\sim\{0,1\}^{m(n)}}[C(w) = 1] \right| \le \frac{1}{m(n)}.$$

For example, Hypothesis C.5 holds if E requires exponential-size (single-valued) nondeterministic circuits almost everywhere [SU05].

For all distribution families $\mathcal{E} = \{\mathcal{E}_n\}_{n\in\mathbb{N}}$ and $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{E}_n$ is over $\{0,1\}^n$ and each $\mathcal{F}_n$ is over (binary representations of) functions in a concept class $\mathscr{C}_n \subseteq \{f\colon \{0,1\}^n \to \{0,1\}\}$, we use the notation $\mathcal{D}_{\mathcal{E},\mathcal{F}}$ to refer to the following distribution of samplers: for each $n \in \mathbb{N}$, (i) select $f \sim \mathcal{F}_n$, and (ii) output a sampler $S_{\mathcal{E},f}$ that generates $(x, f(x))$ for $x \sim \mathcal{E}_n$.

Lemma C.6. *If Hypothesis C.5 holds, then for every polynomial-time evaluatable concept class $\mathscr{C} = \{\mathscr{C}_n\}_{n\in\mathbb{N}}$, where $\mathscr{C}_n \subseteq \{f\colon \{0,1\}^n \to \{0,1\}\}$, there exist a polynomial τ_0 and a constant $C > 0$ such that for every polynomial $t(n)$, every $t(n)$-time samplable distribution $\mathcal{E}$ over examples, every $t(n)$-time samplable distribution $\mathcal{F}$ over $\mathscr{C}$, every sufficiently large $n \in \mathbb{N}$, every $m, \tau \in \mathbb{N}$ with $\tau \ge \tau_0(n, m, t(n))$, and every $X = (\boldsymbol{x}, \boldsymbol{b}) \in \mathrm{supp}((\mathcal{D}_{\mathcal{E},\mathcal{F}})_n^m)$, it holds that*

$$\mathrm{LT}^\tau(X) \le -\log(\mathcal{D}_{\mathcal{E},\mathcal{F}})_n^m(\boldsymbol{b}|\boldsymbol{x}) + C\log\tau.$$

Particularly, $\mathcal{D}_{\mathcal{E},\mathcal{F}}$ satisfies the condition in Theorem C.3 for every $\tau \ge \tau_0(n, m, t(n))$ when $\sigma(n)$ in Theorem C.3 is the upper bound on the length of binary representation for class $\mathscr{C}_n$.

Proof. The proof essentially appeared in [AGMMM18; Hir21]; the only difference is that we consider the time-bounded LT-complexity.

For readability, we omit the subscript $\mathcal{E}, \mathcal{F}$ of $\mathcal{D}_{\mathcal{E},\mathcal{F}}$. Let F be the $t(n)$-time sampling algorithm for $\mathcal{F}$.

Fix $X = (\boldsymbol{x}, \boldsymbol{b}) \sim \mathcal{D}_n^m$ arbitrarily. Let $\boldsymbol{x} = (x^1, \dots, x^m)$, $\boldsymbol{b} = (b^1, \dots, b^m)$, and $p = \mathcal{D}_n^m(\boldsymbol{b}|\boldsymbol{x})$. Then, we have

$$\begin{aligned} p = \mathcal{D}_n^m(\boldsymbol{b}|\boldsymbol{x}) &= \Pr_{f\sim\mathcal{F}_n}[f(x^i) = b^i \text{ for each } i \in [m]] \\ &= \Pr_{r\sim\{0,1\}^{t(n)}}[f = F(1^n, r) \text{ and } f(x^i) = b^i \text{ for each } i \in [m]]. \end{aligned}$$

Let $R \subseteq \{0,1\}^{t(n)}$ be a set of random strings $r \in \{0,1\}^{t(n)}$ such that $f = F(1^n, r)$ and $f(x^i) = b^i$ for each $i \in [m]$. Then, we have $p = |R| \cdot 2^{-t(n)}$.

Let $s = \lceil -\log p \rceil$ and $\ell = t(n) - s - 2$. We consider the pairwise-independent hash family $\mathcal{H} = \{h_{U,v} : \{0,1\}^{t(n)} \to \{0,1\}^\ell\}_{(U,v)}$, where $U \in \mathbb{F}_2^{\ell\times t(n)}$, $v \in \mathbb{F}_2^\ell$, and $h_{U,v}(r) = U \cdot r + v$ (where we identify $\{0,1\}$ with $\mathbb{F}_2$).

By Chebyshev's inequality, with probability at least $1/4$ over the choice of (U, v), there exists an element $r_{U,v} \in R \cap h_{U,v}^{-1}(0^\ell)$, and $|h_{U,v}^{-1}(0^\ell)| \le 2^{s+3}$ holds (cf. [AGMMM18, Claim 4.2.1]).

Furthermore, through Gaussian elimination, any $r \in h_{U,v}^{-1}(0^\ell)$ is reconstructed from (U, v) and $\log|h_{U,v}^{-1}(0^\ell)| \leq s + 3$ additional bits of information (i.e., index in $h_{U,v}^{-1}(0^\ell)$) in $\mathsf{poly}(t(n))$ time (cf. [AGMMM18, Claim 4.2.2]). Particularly, $r_{U,v} \in R \cap h_{U,v}^{-1}(0^\ell)$ is also reconstructed from (U, v) and some $w_{U,v} \in \{0,1\}^{\leq s+3}$.

Now, we consider the following nondeterministic algorithm A. On input $z \in \{0,1\}^{\ell(t(n)+1)}$ and auxiliary advice $\boldsymbol{x} = (x^1, \dots, x^m)$ and $\boldsymbol{b} = (b^1, \dots, b^m)$, the algorithm A regards z as a tuple (U, v), where $U \in \mathbb{F}_2^{\ell \times t(n)}$ and $v \in \mathbb{F}_2^\ell$, and nondeterministically guesses $w_z \in \{0,1\}^{\leq s+3}$ such that the reconstruction algorithm specified above generates a function $f\colon \{0,1\}^n \to \{0,1\}$ for which $f(x^i) = b^i$ holds for each $i \in [m]$. If there exists such a w_z, then A accepts z. By the argument above, the following holds:

$$\Pr_{z \sim \{0,1\}^{\ell(t(n)+1)}}[A(z; \boldsymbol{x}, \boldsymbol{b}) = 1] \geq 1/4.$$

By the standard way to translate a Turing machine into a circuit, we obtain a nondeterministic circuit $\tilde{A}$ of size $\tau(n, m, t(n))$ that corresponds to A, where τ is a polynomial determined by the time complexity for evaluating $\mathscr{C}$ (because A executes the evaluation algorithm for $\mathscr{C}$). Let $G\colon \{0,1\}^{C_0 \log \tau(n,m,t(n))} \to \{0,1\}^{\tau(n,m,t(n))}$ be the pseudorandom generator in Hypothesis C.5 for circuit size $\tau(n, m, t(n))$. Then, there must exist a string $z' \in \{0,1\}^{C_0 \log \tau(n,m,t(n))}$ such that $\tilde{A}(G(z')) = 1$; otherwise,

$$\Pr_{z \sim \{0,1\}^{\tau(n,m,t(n))}}[\tilde{A}(z) = 1] - \Pr_{z' \sim \{0,1\}^{C_0 \log \tau(n,m,t(n))}}[\tilde{A}(G(z')) = 1] \geq 1/4 - 0 = 1/4,$$

which contradicts the fact that G is a pseudorandom generator.

Since $\tilde{A}(G(z')) = 1$, there exists a witness $w \in \{0,1\}^{\leq s+3}$ such that the reconstruction algorithm above generates $f\colon \{0,1\}^n \to \{0,1\}$ satisfying that $f(x^i) = b^i$ holds for each $i \in [m]$ from $G(z')$ (regarded as a seed for the hash function) and index w. We remark that this consistent function f is uniformly constructed from G, z', w in $\tau'(n, m, t(n))$ time, where τ' is a polynomial determined by the time-complexity of evaluating $\mathscr{C}$ and computing G.

Thus, there exists a polynomial τ'' (determined by the time-complexity of evaluating $\mathscr{C}$ and computing G, and simulation overhead for U) such that

$$\begin{aligned}
\mathrm{LT}^{\tau''(n,m,t(n))}(X) &\leq |G| + |z'| + |w| + O(1) \\
&\leq s + O(\log \tau''(n, m, t(n))) \\
&\leq -\log \mathcal{D}_n^m(\boldsymbol{b}|\boldsymbol{x}) + C_1 \log \tau''(n, m, t(n))
\end{aligned}$$

for some absolute constant $C_1 > 0$. $\square$

Lemmas C.3 and C.6 immediately imply the following learnability result.

Theorem C.7. *Let $\mathscr{C} = \{\mathscr{C}_n\}_{n \in \mathbb{N}}$ be a polynomial-time evaluatable class, where $\mathscr{C}_n \subseteq \{f\colon \{0,1\}^n \to \{0,1\}\}$, such that the length of binary representation for $\mathscr{C}_n$ is at most $\ell(n)$ for some polynomial ℓ.*

Under the non-existence of infinitely-often one-way functions and Hypothesis C.5, there exist a polynomial-time randomized algorithm L and a polynomial τ_0 such that for every polynomial $t(n)$, every unknown $t(n)$-time samplable distribution $\mathcal{E}$ over examples, every unknown $t(n)$-time samplable distribution $\mathcal{F}$ over $\mathscr{C}$, every sufficiently large $n \in \mathbb{N}$, and every $m, \delta^{-1}, \tau \in \mathbb{N}$ with $\tau \geq \tau_0(n, m, t(n))$, the algorithm $L(\text{-}; 1^{\langle n,m,\delta^{-1},t(n),\ell(n),\tau \rangle})$ solves the search version of MINLT$[\tau]$ *on average under $\mathcal{D}_{\mathcal{E},\mathcal{F}}^m$ with error probability at most δ.*

D Usage of Pre-Knowledge

In this section, we discuss that the time-complexity of our learner can be improved when pre-knowledge is available, where we consider the pre-knowledge given as $z = M^1, \dots, M^\kappa$ for an absolute constant κ and descriptions $M^1, \dots, M^\kappa$ of samplers that select secret information.

We introduce some notions. For any $t \in \mathbb{N}$ and any $z \in \{0,1\}^*$, we define the t-time bounded distribution given z, denoted by Q_z^t, as the distribution of $U^t(r, z)$ for $r \sim \{0,1\}^t$. We use the notation Q_z^t to refer to the probability that x is sampled from Q_z^t. We also define the conditional variants of q^t and computational depth as $\mathrm{q}^t(x|z) = -\log \mathrm{Q}_z^t(x)$ and $\mathrm{cd}^t(x|z) := \mathrm{q}^t(x|z) - \mathrm{K}(x|z)$.

For every fixed advice z, we can use $\mathrm{cd}^t(x|z)$ instead of $\mathrm{cd}^t(x)$ in universal extrapolation.

Theorem D.1 (Universal Extrapolation with Fixed Advice)**.** *If there exists no infinitely-often one-way function, then for every advice string $z \in \{0,1\}^*$, there exists a randomized polynomial-time algorithm UE_z such that for all $k, t, \epsilon^{-1}, \alpha, d \in \mathbb{N}$ and all $x \in \{0,1\}^*$ with $\mathrm{cd}^t(x|z) \le \alpha$,*

$$\mathsf{L}_1\left(\mathsf{UE}_z(x; 1^{\langle k,t,\epsilon^{-1},2^\alpha\rangle}), \mathsf{Next}_k(\mathrm{Q}_z^t, x)\right) \le \epsilon.$$

Proof. The proof is the same as that of Theorem 8.1, except we always invoke $U^t(\text{-}, z)$ instead of U^t. □

For every fixed advice z, we can use Q_z^t instead of Q^t in Lemma 9.2.

Lemma D.2. *For every advice $z \in \{0,1\}^*$ and every distribution $\mathcal{D}$ over binary strings such that $\mathcal{D}$ has a t_D-time sampler described by d bits, and for every $a \in \mathbb{N} \cup \{0\}$, every $t, q, b, m \in \mathbb{N}$ with $t \ge \tau_{\mathrm{dom}}(d, t_D)$, and every q-query (possibility not efficiently computable) randomized oracle machine I,*

$$\mathop{\mathbb{E}}_{i \sim [m], x^{<i} \sim \mathcal{D}^{<i}, x^i \sim \mathcal{D}^{i, x^{<i}}}\left[\mathrm{KL}\left(I^{\mathsf{Next}_b(\mathcal{D}, x^{<i}x^i)} || I^{\mathsf{Next}_b(\mathrm{Q}_z^t, x^{<i}x^i)}\right)\right] \le q \cdot \frac{O(d)}{m},$$

where the hidden constant in $O(d)$ depends on only the universal Turing machine.

Proof. In the same way as the proof of Lemma 9.2, we obtain

$$\mathbb{E}\left[\mathrm{KL}\left(I^{\mathsf{Next}_b(\mathcal{D}, x^{<i}x^i)} || I^{\mathsf{Next}_b(\mathrm{Q}_z^t, x^{<i}x^i)}\right)\right] \le \frac{q}{m} \sum_{x \in \mathrm{supp}(\mathcal{D})} \mathcal{D}(x) \log \frac{\mathcal{D}(x)}{\mathrm{Q}_z^t(x)}.$$

The above implies the lemma because the domination property $\mathrm{Q}_z^t(x) \ge 2^{-O(d(\mathcal{D}))}\mathcal{D}(x)$ holds by considering the case where the prefix of the random seed for U^t corresponds to the sampler of $\mathcal{D}$. □

By combining Theorem D.1 with Lemma D.2 as in Section 9.2, we obtain the following learner for ACDs with better time complexity under the pre-knowledge.

Theorem D.3. *Let $s(n)$ be a polynomial and $\mathcal{S}$ be a finite subset of pairs $(\mathcal{G}, D)$, where D is a polynomial-time samplable ACD of $s(n)$-bit initial state and $\mathcal{G}$ is samplable distribution over $s(n)$-bit initial states. If there exists no infinitely-often one-way function, then there exists a learning algorithm $L_\mathcal{S}$ such that for every $(\mathcal{G}, D) \in \mathcal{S}$, the learner $L_\mathcal{S}$ learns D under $\mathcal{G}$ in $\mathsf{poly}(n, |\mathcal{S}|, \epsilon^{-1}, \delta^{-1})$ time with sample complexity $O(s(n) \cdot \epsilon^{-2}\delta^{-1})$.*

Proof. For sufficiently large t and for every $(\mathcal{G}, D) \in \mathcal{S}$,

$$\mathrm{Q}^t_{\mathcal{S}}(\mathcal{G}, D) \geq 2^{-O(\log |\mathcal{S}|)}$$

because there exists a program that has an index $i \in [|\mathcal{S}|]$ and outputs the indicated i-th distribution in $\mathcal{S}$. Therefore, $\mathrm{q}^t(\mathcal{G}, D|\mathcal{S}) \leq O(\log |\mathcal{S}|)$.

Based on the same proof as that of Theorem 9.7, the theorem follows from Theorem D.1, Lemma D.2, and the fact that for sufficiently large t and for every $(\mathcal{G}, D) \in \mathcal{S}$,

$$\mathrm{cd}^t(\mathcal{G}, D|\mathcal{S}) < \mathrm{q}^t(\mathcal{G}, D|\mathcal{S}) \leq O(\log |\mathcal{S}|).$$

$\square$

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il