

Learning, Compression, and Meta-Complexity

Halley Goldberg*

Department of Computer Science
University of Oxford

Mikito Nanashima[†]

School of Computing
Institute of Science Tokyo

Igor C. Oliveira[‡]

Department of Computer Science
University of Warwick

October 8, 2026

Abstract

The idea that learning is closely tied to compression has long served as both a powerful principle and, at times, an informal analogy across machine learning and learning theory. Here we focus on settings in which that intuition can be made mathematically precise, surveying formal connections among learning algorithms, data compression, and computational complexity theory, with an emphasis on recent progress and current frontiers. We describe how techniques from complexity theory can guide the design of both supervised and unsupervised learning algorithms with provable guarantees, and conversely, how learning algorithms can yield new insights and results in complexity theory. A central theme is the role of data compression as a bridge between learning and complexity, especially through an area called meta-complexity, which studies the computational difficulty of compression problems and time-bounded variants of Kolmogorov complexity.

*Email: halley.goldberg@cs.ox.ac.uk

[†]Email: nanashima@comp.isct.ac.jp

[‡]Email: igor.oliveira@warwick.ac.uk

Contents

1	Introduction	4
1.1	Overview	4
1.2	Organisation	7
2	Preliminaries	9
2.1	Basic Notation	9
2.2	Probability Theory and Information Theory	9
2.3	Learning Frameworks	10
2.3.1	Sequence Extrapolation	10
2.3.2	Learning Distributions	11
2.3.3	PAC Learning	11
2.3.4	Agnostic Learning	12
2.3.5	Heuristic Learnability	13
2.4	Theoretical Computer Science	13
2.4.1	Cryptography	14
2.4.2	Kolmogorov Complexity	16
2.4.3	Circuit Complexity	18
2.4.4	Meta-Complexity	20
2.4.5	Natural Properties	21
2.4.6	Pseudorandomness	21
3	Classical Connections Between Learning and Compression	23
3.1	Model Compression	23
3.2	Sample Compression	25
4	Complexity-Theoretic Foundations of Learning and Compression	28
4.1	Learning Paradises	28
4.2	Basic Relations	28
4.3	Impagliazzo's Worlds and Meta-Complexity	31
4.4	Auxiliary Results	33
4.5	White-Box Learning and One-Way Functions	34
4.5.1	White-Box Learning Frameworks	34
4.5.2	White-Box Learning and Breaking Auxiliary-Input One-Way Functions	35
5	Learning via Meta-Complexity and Algorithmic Information Theory	39
5.1	Detecting Structure: Supervised Learning via Distinguishers	39
5.1.1	From Distinguishers to Predictors	39
5.1.2	Direct Product and Random Examples	40
5.1.3	Nisan-Wigderson and Membership Queries	45
5.2	Solomonoff Induction: Unsupervised Learning via Kolmogorov Complexity	47
5.2.1	The Classical Theory	48
5.2.2	The Time-Bounded Setting	49
5.3	Equivalences Between Learning and Meta-Complexity	52
5.4	Learning Speedups: Reducing the Running Time via Meta-Complexity	55

6	Learning vs Compression via Shannon's Information Theory	57
6.1	Starting from Shannon's Information Theory	57
6.1.1	Universal Coding and Kolmogorov Complexity as Its Ultimate Goal	58
6.1.2	Universal Coding and Learning	58
6.2	Computational Learning and Meta-Complexity	63
6.2.1	Learning to Compression via Arithmetic Coding	64
6.2.2	Compression to Learning over Polynomially Many Messages	66
6.2.3	Compression to Learning over General Messages	69
6.2.4	Language Modeling	72
7	Meta-Complexity and Computational Hardness from Learning	74
7.1	Deterministic Learners	74
7.2	Randomized and Quantum Learners	76
7.3	Unconditional Meta-Complexity Lower Bounds via Learning	78
8	Related Work	80

1 Introduction

1.1 Overview

We describe connections between learning, compression, and meta-complexity (see Figure 1). We start off with a brief introduction to each of these themes, emphasising some important conceptual points along the way.

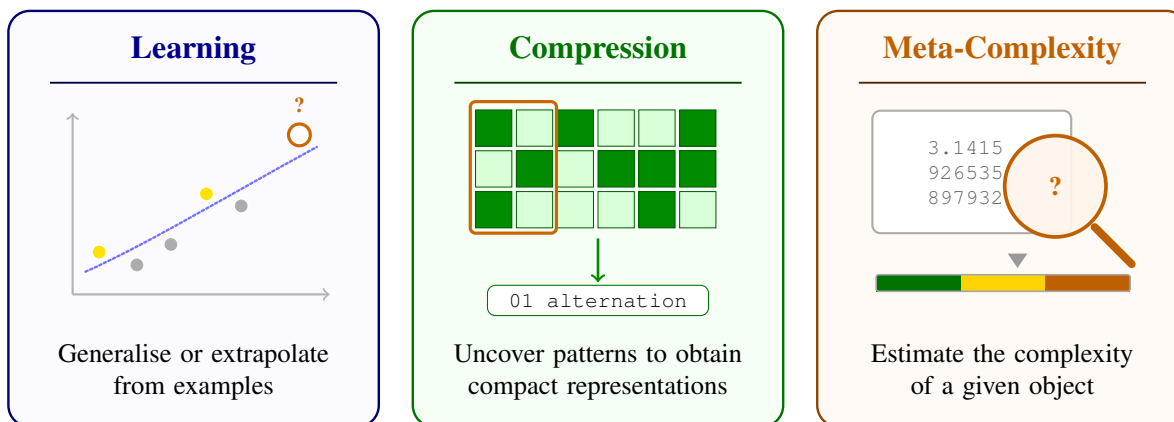


Figure 1: Three perspectives on structure in data: learning, compression, and meta-complexity. We investigate correspondences among them through several lenses, including Shannon’s information theory, modern variants of (time-bounded) Kolmogorov complexity, and the theory of computational pseudorandomness. Interestingly, these connections can be used both to design new learning algorithms and to prove that certain compression problems admit no efficient algorithms.

Learning and Compression. At a high level, learning and compression are two ways of asking whether observed data admit a concise explanation:

- A *learning algorithm* uses examples to construct a model that makes useful predictions on new data.
- A *compression algorithm* represents the data using fewer bits, together with a procedure that recovers the original data.

The outputs are different, but both tasks succeed by identifying *structure* or *regularity*: a pattern that permits prediction also makes some outcomes more likely than others, and likely outcomes can be assigned shorter descriptions. Conversely, a short description is evidence that the object is not completely unstructured and can therefore be a useful starting point for prediction.

This connection takes different forms in supervised and unsupervised learning. In *supervised learning*, examples arrive with labels, and the learner seeks a hypothesis that predicts the label of a fresh example. Here compression may mean inferring from a large labelled sample a short consistent hypothesis, as in Occam’s Razor, or retaining a small subsample from which a good hypothesis can be reconstructed. In *unsupervised learning*, there are no labels. The learner may instead produce an approximate distribution that generates the observations, or predict the next part of a sequence from its prefix. Here the relationship with *source coding* is especially direct: if a probabilistic model assigns probability p to an outcome, then the outcome can be encoded using roughly $\log(1/p)$ bits. Thus accurate probabilistic predictions lead to

short descriptions of likely data, while valid code lengths can conversely be interpreted as probability assignments. This correspondence connects distribution learning, sequence prediction, and source coding.

One of the main goals of this survey is to determine precisely how far this correspondence extends. We formulate the relevant computational tasks and present reductions showing when a learner can be transformed into a compressor, and when compression can be used to construct a learner.

Two Theories of Information. We use two complementary notions of *information*. In *Shannon’s information theory*, a message is drawn from a *probability distribution*, and entropy measures the optimal expected description length when coding repeated outputs of that source [Sha48] (see [CT91]). This is a statistical and distributional theory: the information content of an outcome is determined by its probability under the chosen source. In contrast, in *algorithmic information theory*, the Kolmogorov complexity of an *individual string* is the length of its shortest generating program [Kol65] (see [LV19]). No source distribution needs to be specified, and the theory specifies the information content of each string.

These two theories are related, but not interchangeable. For instance, *source coding*, mentioned above, connects probability distributions with description length: a distribution with a short sampler gives short algorithmic descriptions to its typical outputs. Nevertheless, compressing individual strings in this way may be computationally expensive and may require access to the code of the sampling procedure; moreover, a string with a short program need not be well compressed by a particular sampler and the associated source coding procedure.

A major focus of this survey is on connections between learning and compression under *computational efficiency constraints*. Consider for instance Kolmogorov complexity. It describes the shortest possible program, but it places no bound on how long that program may take to produce its output. Given a string, estimating its Kolmogorov complexity is *uncomputable* in general. Moreover, central properties of the theory do not automatically survive once efficiency is imposed. For example, *symmetry of information* holds for Kolmogorov complexity up to a small additive loss, whereas its time-bounded analogues do not hold in general (see, e.g., [HIL⁺23, HMO26]). Classical Kolmogorov complexity, while philosophically relevant, is not by itself an adequate foundation for computational learning. To study *efficient* learning through compression, we need both a theory of *time-bounded* descriptions and a theory of the *computational complexity* of finding or recognizing them.

Meta-Complexity. These considerations lead naturally to *meta-complexity*, the study of the computational complexity of problems that ask about the complexity of input objects. Canonical examples ask whether a string has a short program that generates it within a prescribed time bound, or whether the truth table of a Boolean function can be computed by a small Boolean circuit. Such problems formalise the task of detecting whether an object is compressible or structured; see [Sim23] for further context. The central question is no longer merely whether short descriptions exist, but whether they can be recognised or found efficiently.

The complexity of these compression problems is intertwined with some of the most fundamental unresolved questions in theoretical computer science. In the terminology of Impagliazzo’s five worlds [Imp95], we do not yet know which computational world we inhabit: whether worst-case hard problems remain hard on typical inputs, whether such hardness supports secure cryptography, and how broadly efficient algorithms can exploit structure. Meta-complexity, through the study of the complexity of compression problems, turns out to provide a suitable framework in which to investigate this tension between tractability on typical inputs and cryptographic hardness. In turn, this will allow us to compare the feasibility of different learning tasks

and their relation to Impagliazzo’s computational worlds.

There is, however, an important mismatch between learning and the standard formulation of compression problems in meta-complexity. Most canonical problems in meta-complexity are *decision problems*. For example, given a string x , a time bound t , and a description length s , the Minimum Time-Bounded Kolmogorov Complexity problem (MINKT) asks whether some program of size at most s produces x in at most t steps. The corresponding *search problem* asks for such a program. Learning is inherently a search task: a learner must output a hypothesis, sampler, or predictor, rather than merely certify that a concise representation exists. Thus, in principle, an efficient algorithm that merely recognises compressible data does not automatically yield a corresponding search algorithm. Interestingly, many results surveyed here bridge this gap indirectly: a decision procedure separating structured objects from random ones is sufficient to construct a learning algorithm.

Having introduced the main themes, we elaborate on the scope of the survey. We first revisit classical connections in which compression yields generalisation in supervised learning. After this, we compare learning tasks of different strengths, from universal sequence extrapolation and distribution learning to more traditional forms of PAC learning, and relate this hierarchy to average-case complexity, meta-complexity, and cryptography. Next, we explain how algorithms for detecting objects of low description complexity can produce learning algorithms, both through reconstruction procedures in supervised learning and through time-bounded variants of Solomonoff induction in unsupervised learning. This perspective also reveals novel phenomena such as generic learning speedups and, in certain settings, equivalences between efficient learning and the easiness of an associated meta-complexity problem. Finally, we discuss an unexpected converse direction, showing how sufficiently powerful learning algorithms can yield computational hardness results and new lower bounds in meta-complexity. For a more detailed description of each section of the survey, we refer to [Section 1.2](#) below.

We place particular emphasis on recent developments in these rapidly evolving areas. These include new connections among learning, time-bounded Kolmogorov complexity, meta-complexity, cryptography, average-case complexity, and universal sequence extrapolation. Our aim is not only to state these results, but also to explain the underlying ideas and, where appropriate, provide proofs or detailed proof sketches. The emerging theory remains far from complete, so open problems form an integral part of the survey.

Intended Audience and Required Background. The primary audience includes students and researchers interested in computational complexity, computational learning, and theoretical computer science more broadly. We aim to make the main results and proof ideas accessible without assuming prior expertise in every area involved. The preliminaries review the necessary concepts and definitions, and provide pointers to standard textbooks that cover the required material in more detail.

We also hope that researchers and students in machine learning will find the survey useful as an entry point to the complexity-theoretic foundations of learning and compression. We warn that some arguments might omit standard background or routine intermediate steps. Readers may find it helpful to use their preferred AI assistant as an interactive companion for unpacking definitions, reconstructing such steps, or translating between the terminology of theoretical computer science and intuitions from machine learning.

We stress that the selection of topics necessarily reflects the authors’ background and perspective. In particular, we do not attempt to provide a comprehensive account of learning and compression from the perspective of statistics, information theory, or theoretical machine learning. We refer the reader to [Section 8](#) for some additional pointers and complementary perspectives.

Finally, to present a general theory, much of this survey considers expressive learning problems involving unrestricted polynomial-size Boolean circuits or all efficiently samplable example distributions. As explained later in the text, these learning problems are intractable under cryptographic conjectures. However, many ideas and techniques surveyed here also apply to more restricted learning tasks, and such tasks are more likely to admit efficient algorithms. It is possible, therefore, that some results can inform practical methods or analyses, and it would be interesting to see further work along these lines. In fact, going beyond the formal connections covered here, the compression perspective has found numerous concrete applications in both classical and contemporary machine learning. A comprehensive treatment of the rich empirical literature on learning and compression lies beyond the scope of this survey, although we provide some pointers in [Section 8](#).

1.2 Organisation

The remainder of the survey is organised as follows.

- **Section 2: Preliminaries.** We establish the notation and background used throughout the survey. After reviewing the required probability and information-theoretic notions, we define the supervised and unsupervised learning frameworks considered later, including sequence extrapolation, distribution learning, PAC learning, agnostic learning, and heuristic learning. We also recall the necessary concepts from cryptography, circuit complexity, Kolmogorov complexity, meta-complexity, and natural properties.
- **Section 3: Classical Connections Between Learning and Compression.** We present two fundamental roles of compression in supervised learning. Model compression formalises the Occam principle that a short hypothesis fitting the observed labels is unlikely to generalise poorly, while sample compression retains a small portion of the labelled sample from which a hypothesis can be reconstructed. These results illustrate, without computational efficiency considerations, how the amount of information retained from a sample controls generalisation.
- **Section 4: Complexity-Theoretic Foundations of Learning and Compression.** We compare five increasingly weak “learning paradises”, ranging from universal sequence extrapolation and distribution learning to heuristic PAC learning, and establish reductions among them. We then place these learning tasks within Impagliazzo’s worlds, relating their feasibility to average-case complexity, meta-complexity, and cryptographic hardness. Finally, we study white-box learning, where the learner receives the code of a sampler generating its labelled examples, and connect this additional access to the inversion of one-way functions.
- **Section 5: Learning via Meta-Complexity and Algorithmic Information Theory.** We explain how algorithms that distinguish low complexity objects from random ones can be combined with reconstruction procedures of pseudorandom generators to obtain supervised learning algorithms. For unsupervised learning, [Section 5.2](#) revisits Solomonoff induction and asks when prediction based on shortest programs can be implemented efficiently using estimates of time-bounded Kolmogorov complexity. We also present equivalences between learning and meta-complexity in [Section 5.3](#), and a technique to boost the running time of learning algorithms in [Section 5.4](#).
- **Section 6: Learning vs Compression via Shannon’s Information Theory.** We develop the correspondence among source coding, universal coding, and distribution learning, first at the information-theoretic level and then under computational efficiency constraints. The discussion covers arithmetic

coding, next-bit prediction, source coding theorems for time-bounded Kolmogorov complexity, conditional sampling, and symmetry of information. We conclude by relating this framework to autoregressive language modelling, where next-token probabilities determine predictions and code lengths.

- **Section 7: Meta-Complexity and Computational Hardness from Learning.** We study the converse direction, explaining how nontrivial deterministic, randomized, and quantum learners can imply circuit size lower bounds, a major challenge in computational complexity theory. We then show how this paradigm can establish unconditional lower bounds for meta-complexity problems. In particular, [Section 7.3](#) describes a general template in which an assumed algorithm for measuring compressibility produces a learner whose complexity-theoretic consequences contradict the original easiness assumption.
- **Section 8: Related Work.** We provide pointers to related connections and research directions outside the survey’s main scope, including further interactions among learning, compression, algorithmic information theory, complexity theory, and cryptography. We also provide links to the empirical literature on learning and compression and to related practical aspects of machine learning.

Acknowledgements. We thank Shuichi Hirahara for helpful discussions on deriving meta-complexity lower bounds from deterministic learning, which led to Item (ii) of [Theorem 7.1](#). We also thank Valentine Kabanets for helpful feedback on an earlier draft of this survey.

Use of AI Tools. We used AI tools to search for relevant literature, assist with the preparation of diagrams, identify typographical errors, check proofs for correctness, and make textual edits to improve the exposition.

Funding Information. This work was supported in part by the UKRI Frontier Research Guarantee Grant EP/Y007999/1 and the Centre for Discrete Mathematics and its Applications (DIMAP) at the University of Warwick. The authors would also like to thank the Isaac Newton Institute for Mathematical Sciences, Cambridge, for support and hospitality during the programme Logical Foundations of Complexity Theory (LFC), where part of the work on this paper was undertaken. The LFC programme was supported by EPSRC grant EP/Z000580/1.

2 Preliminaries

2.1 Basic Notation

For $m \geq 1$, we let $[m] = \{1, \dots, m\}$.

We denote the empty string by ϵ , and the length of a string x by $|x|$. For strings $x \in \{0, 1\}^n$ and $y \in \{0, 1\}^m$, their concatenation is written as $xy \in \{0, 1\}^{n+m}$.

Given a string $z \in \{0, 1\}^n$ and a set $S \subseteq [n]$, we let $z|_S \in \{0, 1\}^{|S|}$ denote the string obtained by restricting z to the coordinates in S .

When considering algorithms, PPT denotes probabilistic polynomial time.

2.2 Probability Theory and Information Theory

If $x \sim D$, then x is sampled from the distribution D . For a set S , $x \sim S$ denotes a uniformly random element from S . For any a in the support of D , $D(a)$ denotes its probability. We write $\mathcal{U}_\ell$ for the uniform distribution over $\{0, 1\}^\ell$. We use $\Delta_{\text{TV}}(P, Q)$ to denote the statistical distance of two distributions with the same support. We will also use KL-divergence and its relation to statistical distance, as reviewed below.

Definition 2.1 (KL Divergence). Let P and Q be distributions on the same finite set Ω . The *Kullback–Leibler divergence* from P to Q is defined by

$$\text{KL}(P\|Q) \triangleq \mathbb{E}_{x \sim P} \left[\log \frac{P(x)}{Q(x)} \right] = \sum_{x \in \Omega} P(x) \log \frac{P(x)}{Q(x)}.$$

Lemma 2.2 (Pinsker’s Inequality). Let P and Q be distributions on the same finite set. Then

$$\Delta_{\text{TV}}(P, Q)^2 \leq \frac{1}{2} \text{KL}(P\|Q).$$

Definition 2.3 (Conditional KL Divergence). Let (P_X, P_Y) and (Q_X, Q_Y) be random pairs on the same finite product set $\mathcal{X} \times \mathcal{Y}$, and suppose that $\text{Support}(P_X) \subseteq \text{Support}(Q_X)$. The *conditional Kullback–Leibler divergence* from P_Y conditioned on P_X to Q_Y conditioned on Q_X is defined by

$$\text{KL}(P_Y \mid P_X \| Q_Y \mid Q_X) \triangleq \mathbb{E}_{x \sim P_X} [\text{KL}(P_Y \mid P_X = x \| Q_Y \mid Q_X = x)].$$

Lemma 2.4 (Chain Rule for KL Divergence). Let $(P_1, \dots, P_m)$ and $(Q_1, \dots, Q_m)$ be random vectors on the same finite product set $\mathcal{X}_1 \times \dots \times \mathcal{X}_m$, and suppose that $\text{Support}(P_1, \dots, P_m) \subseteq \text{Support}(Q_1, \dots, Q_m)$. For every $i \in [m]$, write

$$P_{< i} \triangleq (P_1, \dots, P_{i-1}) \quad \text{and} \quad Q_{< i} \triangleq (Q_1, \dots, Q_{i-1}),$$

where $P_{< 1}$ and $Q_{< 1}$ denote the empty tuple. Then

$$\begin{aligned} \text{KL}((P_1, \dots, P_m) \| (Q_1, \dots, Q_m)) &= \sum_{i=1}^m \text{KL}(P_i \mid P_{< i} \| Q_i \mid Q_{< i}) \\ &= m \cdot \mathbb{E}_{i \sim [m]} [\text{KL}(P_i \mid P_{< i} \| Q_i \mid Q_{< i})]. \end{aligned}$$

We use the following definition of entropy.

Definition 2.5 (Entropy). For a probability distribution $\mathcal{D}$ over a finite set, define the entropy of $\mathcal{D}$ as

$$\text{H}(\mathcal{D}) \triangleq \sum_{x \in \text{Support } \mathcal{D}} \mathcal{D}(x) \log \frac{1}{\mathcal{D}(x)}.$$

For more information about information theory and its applications, see [CT91, PW25].

2.3 Learning Frameworks

In *supervised learning*, the learner receives examples together with the desired answers, usually labels produced by an unknown target function, and the goal is to predict the correct label on new examples. In *unsupervised learning*, the learner receives samples without such labels and must instead capture some structure of the source that generated them.

In the learning models introduced below, sequence extrapolation and distribution learning are unsupervised, since the input consists only of samples from an unknown process or distribution. On the other hand, PAC learning and heuristic learning are supervised, since the learner is given labelled examples of the form $(x, f(x))$ and aims to predict the label of future examples.

2.3.1 Sequence Extrapolation

A common way to learn from data is to ask what should come next. After observing a prefix of a sequence, a learner tries to reproduce the distribution of the next block without being told the process that generated the sequence. This viewpoint captures settings ranging from sequential prediction to universal induction: the past is useful precisely when it reveals regularity in an unknown source. Here we make that intuition computational by assuming only that the sequence is produced by a short randomized program running within a prescribed time bound. Since the source itself may be random, the learner is not expected to guess the next block exactly. Instead, it should make a probabilistic prediction close to the source's true conditional distribution.

Fix a *sequence length* $m \in \mathbb{N}$, a *block length* $n \in \mathbb{N}$, an *error bound* $\varepsilon > 0$, a *confidence value* $\delta > 0$, a *description length* $s \in \mathbb{N}$, and a time bound $t \in \mathbb{N}$.

We consider *randomized* Turing machines M of description length at most s (according to a fixed universal randomized machine U^r). We will be interested in the random variable $U^r(M, \epsilon, 1^t)$ encoding the output of M on the empty string ϵ when U^r runs over the input pair (M, ϵ) for t steps. If this simulation does not produce an output, we set the output to $0^{n \cdot m}$. For convenience, for a choice of n and m , if the output x of $U^r(M, \epsilon, 1^t)$ is of length less than $n \cdot m$, we add 0's to it and consider instead the string $x' = x0^{nm-|x|}$. Therefore, given n and m , we will view $x \sim U^r(M, \epsilon, 1^t)$ as a *random variable* $(x_1, \dots, x_m)$ distributed over $\{0, 1\}^{n \cdot m}$. For simplicity, we might omit the empty string ϵ .

We consider a task called $(m, n, s, t, \varepsilon, \delta)$ -*extrapolation*. In this computational task, we consider a randomized algorithm L such that, for every (unknown) choice of M of length at most s and index i , given the first $i - 1$ blocks $(x_1, \dots, x_{i-1})$ of $x \sim U^r(M, 1^t)$ (denoted by $x_{<i}$), the goal is for $L(x_{<i}) \sim \{0, 1\}^n$ to be distributed according to $(\mathcal{X}_i \mid x_{<i})$, where for a given choice of n, m, M, t , the random variable $(\mathcal{X}_i \mid x_{<i})$ denotes the conditional distribution of the i -th block x_i given preceding blocks $x_1, \dots, x_{i-1} \in \{0, 1\}^n$. Formally, we have the following definition.

Definition 2.6 (Sequence Extrapolation). We say that a randomized algorithm L *solves* $(m, n, s, t, \varepsilon, \delta)$ -*extrapolation* if for every choice of $M \in \{0, 1\}^{\leq s}$ the following holds:

$$\Pr_{i \sim [m], x \sim U^r(M, 1^t)} \left[\Delta_{\text{TV}}(L(x_{<i}), (\mathcal{X}_i \mid x_{<i})) \leq \varepsilon \right] \geq 1 - \delta.$$

We assume that the parameters $(m, n, s, t, \varepsilon^{-1}, \delta^{-1})$ are also given to L in unary when considering uniform algorithms and parameter sequences.

For broader treatments of sequential prediction and algorithmic approaches to extrapolation, see [CL06, HQC24].

2.3.2 Learning Distributions

Sometimes the goal is not merely to predict the next observation, but to construct a model of the source itself. Given independent samples from an unknown distribution, a distribution learner seeks a new sampling procedure whose outputs are statistically indistinguishable from those of the original source, up to a prescribed error. Such a model captures enough of the source's structure to generate fresh representative data, even though it need not reveal how the source was originally implemented.

Formally, we consider the task of learning an unknown discrete distribution D from a fixed class $\mathcal{D}$ of distributions [KMR⁺94]. The learner has access to i.i.d. samples from D , and the goal is to output a circuit $C: \{0, 1\}^T \rightarrow \{0, 1\}^n$ that samples a distribution that is ε -close to D .

Definition 2.7 (Learning Distributions [KMR⁺94]). Let $\mathcal{D} = \{\mathcal{D}_n\}_{n \geq 1}$ be a class of distributions, where each $\mathcal{D}_n$ is a set of distributions supported over $\{0, 1\}^n$. The class $\mathcal{D}$ is said to be learnable in time $T(n, \varepsilon, \delta)$ and with sample complexity $m(n, \varepsilon, \delta)$ if there is a randomized algorithm A such that, for every $n, D \in \mathcal{D}_n$, $\varepsilon > 0, \delta > 0$, and $m = m(n, \varepsilon, \delta)$, $A(1^n, 1^{\varepsilon^{-1}}, 1^{\delta^{-1}}, \cdot)$ runs in time at most $T = T(n, \varepsilon, \delta)$ and

$$\Pr_{A, \vec{x} \sim D^{(m)}} \left[C \leftarrow A(1^n, 1^{\varepsilon^{-1}}, 1^{\delta^{-1}}, \vec{x}) \text{ and } \Delta_{TV}(C(\mathcal{U}_T), D) \leq \varepsilon \right] \geq 1 - \delta, \quad (1)$$

where $C(\mathcal{U}_T)$ is the distribution sampled by C , and $\vec{x} \sim D^{(m)}$ denotes a tuple of m i.i.d. samples from D .

For convenience, when Equation (1) holds for a distribution D , we say that A ε -learns D with probability at least $1 - \delta$.

For functions $s, t: \mathbb{N} \rightarrow \mathbb{N}$, we can consider the class $\mathcal{D}^{s,t} = \{\mathcal{D}_n^{s(n),t(n)}\}_{n \geq 1}$ of distributions, where each $\mathcal{D}_n^{s',t'}$ consists of the set of all distributions D supported over $\{0, 1\}^{n'}$ sampled by a randomized machine M that runs in time at most t' and has description length at most s' .

Similarly, we can consider the distribution learning problem for $\mathcal{D}_n^{s,t}$ when $n, s, t \in \mathbb{N}$ are *input parameters* provided to the learning algorithm $A(1^n, 1^{\varepsilon^{-1}}, 1^{\delta^{-1}}, 1^t, 1^s, \cdot)$.

2.3.3 PAC Learning

PAC (Probably Approximately Correct) learning formalises the familiar supervised task of learning a rule from labelled examples. An unknown target concept assigns a label to each input, and the learner observes examples drawn from a distribution that determines which inputs matter. From sufficiently many such examples, the learner should produce a hypothesis that makes few mistakes on a fresh input from the same distribution. The name highlights the two allowances that make the goal realistic: the hypothesis need only be approximately correct, and the learner need only succeed with high probability. In other words, PAC learning asks for reliable generalisation to the inputs that are likely to be encountered.

Formally, in the PAC learning framework, one considers an unknown function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ for some $n \in \mathbb{N}$. Such a function is referred to as a *concept*. A *concept class* $\mathcal{C}$ refers to a set of concepts, with $\mathcal{C}_n$ denoting $\mathcal{C} \cap \{f: \{0, 1\}^n \rightarrow \{0, 1\}\}$ for a fixed $n \in \mathbb{N}$. The PAC learnability of a concept class is defined with respect to a class $\mathcal{D}$ of probability distribution ensembles $D = \{D_n\}_{n \in \mathbb{N}}$, where each D_n is supported over $\{0, 1\}^n$.

PAC learners are provided a *labelled example oracle* $\text{EX}(f, D_n)$, with each query returning an independently sampled pair $(x, f(x))$, where $x \sim D_n$.

Definition 2.8 (PAC Learning [Val84]; see [Ser25] for a survey). Let $\mathcal{C}$ be a concept class, and let $\mathcal{D}$ be a class of distributions. An algorithm A PAC learns $\mathcal{C}$ over $\mathcal{D}$ if, for all $D \in \mathcal{D}$, all sufficiently large $n \in \mathbb{N}$, all $\delta, \varepsilon \in (0, 1)$, and all $f \in \mathcal{C}_n$,

$$\Pr_{A; \text{EX}(f, D_n)} \left[A^{\text{EX}(f, D_n)}(1^n, 1^{\varepsilon^{-1}}, \delta^{-1}) \text{ outputs } h \text{ such that } \Pr_{x \sim D_n} [h(x) = f(x)] \geq 1 - \varepsilon \right] \geq 1 - \delta.$$

For $s : \mathbb{N} \rightarrow \mathbb{N}$, we can consider the class $\mathcal{C}^s = \{\mathcal{C}_n^{s(n)}\}_{n \in \mathbb{N}}$ of concepts, where each $\mathcal{C}_n^{s(n)}$ is the subset of $\mathcal{C}_n$ with representations of size $s(n)$. For example, if $\mathcal{C} = \text{P/poly}$, then $\mathcal{C}^s = \text{SIZE}[s]$, the set of functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$ computable by circuits of size $s(n)$. As in the previous section, we define $\mathcal{D}^{a,t}$ to be the class of distributions samplable by a t -time machine with description length a . To make these parameters explicit, s, t, a may be given as inputs to the learning algorithm.

One may also consider the model of PAC learning with *membership queries*, which is exactly the same as the definition above, except the learner A is additionally provided an f -oracle.

A learner that always outputs a hypothesis h belonging to $\mathcal{C}$ is called *proper*. Otherwise, the learner is *improper*.

If ε is not given as an input but the hypothesis output by A satisfies $\Pr_{x \sim D_n} [h(x) = f(x)] \geq 1/2 + \gamma(n)$ with probability $1 - \delta$ for some $\gamma(n) = \text{poly}^{-1}(n)$, then A is referred to as a *weak learner*.

For textbook introductions to computational learning theory and the PAC framework, see [KV94a, SSBD14].

2.3.4 Agnostic Learning

The PAC model assumes that every observed label is generated by some target concept from the class under consideration. In practice, labels may be noisy, internally inconsistent, or governed by a relationship that the chosen class cannot represent. *Agnostic learning* removes this assumption and asks what can still be learned from an arbitrary joint distribution over inputs and labels. Since nearly perfect prediction may then be impossible, success is measured relative to the best concept available in the class. The learner must perform almost as well as that benchmark without knowing either the source of the labels or the benchmark concept in advance. This formulation preserves a meaningful learning objective while allowing both noise and model misspecification.

In technical terms, one cannot assume that the given examples are labelled according to a function f belonging to the relevant concept class $\mathcal{C}$. In fact, one cannot assume that the labels correspond to a fixed function at all. In particular, we now consider a class $\mathcal{E}$ of joint distribution ensembles $E = \{E_n\}_{n \in \mathbb{N}}$, where each E_n is supported over $\{0, 1\}^n \times \{0, 1\}$. The example oracle $\text{EX}(E_n)$ simply returns an independent random sample from E_n each time it is queried.

Now, there is clearly no hope in general for an efficient learner to produce a hypothesis with high correctness probability over E (i.e., $\Pr_{(x,b) \sim E_n} [h(x) = b] \approx 1$).¹ Therefore, as alluded to above, we measure the success of an agnostic learner relative to the concept class $\mathcal{C}$ in question, asking for a hypothesis that does “almost as well” as the best concept in $\mathcal{C}$. To that end, for $n \in \mathbb{N}$, define

$$\text{opt}_n(\mathcal{C}, E) \triangleq \max_{c \in \mathcal{C}_n} \left\{ \Pr_{(x,b) \sim E_n} [c(x) = b] \right\}.$$

¹For example, consider $E_n = (\mathcal{U}_n, \mathcal{U})$. This joint distribution can be sampled efficiently, but no fixed circuit has correctness probability greater than $1/2$.

Definition 2.9 (Agnostic Learning [KSS94, Hau92]). Let $\mathcal{C}$ be a concept class, and let $\mathcal{E}$ be a class of joint distributions. An algorithm A *agnostically learns* $\mathcal{C}$ over $\mathcal{E}$ if, for all $E \in \mathcal{E}$, all sufficiently large $n \in \mathbb{N}$, and all $\delta, \varepsilon \in (0, 1)$,

$$\Pr_{A; \text{EX}(E_n)} \left[A^{\text{EX}(E_n)}(1^n, 1^{\varepsilon^{-1}}, \delta^{-1}) \text{ outputs } h \text{ such that } \Pr_{(x,b) \sim E_n} [h(x) = b] \geq \text{opt}_n(\mathcal{C}, E) - \varepsilon \right] \geq 1 - \delta.$$

In this context, if we say that an algorithm agnostically learns over the distribution class $\mathcal{D}^{a,t} = \{\mathcal{D}_m^{a(m),t(m)}\}_{m \geq 1}$ as defined above, we interpret each $\mathcal{D}_m^{a(m),t(m)}$ as a set of joint example distributions supported over $\{0, 1\}^{m-1} \times \{0, 1\}$.

For a textbook treatment of agnostic learning, see [SSBD14].

2.3.5 Heuristic Learnability

Standard supervised learning guarantees usually require the learner to succeed for every target concept in the class. In complexity theory and cryptography, however, target functions often arise from a randomized generation procedure, just as the inputs on which they are evaluated arise from their own sampling process. *Heuristic learnability* reflects this situation by measuring prediction success on average over both the chosen target and the chosen input. The learner may therefore fail on exceptional targets or examples, provided that these failures are sufficiently rare in the overall experiment. Even a small advantage over random guessing can be significant in this setting, making the model especially useful for expressing connections among prediction, average-case hardness, and cryptographic security.

Let $\mathcal{C}$ be a concept class, with each $f \in \mathcal{C}_n$ having a representation σ_f of size at most polynomial in n . Let $P = \{P_n\}_{n \in \mathbb{N}}$ be a distribution ensemble, with each P_n supported over representations σ_f of target functions $f \in \mathcal{C}_n$ such that f can be evaluated in polynomial time given σ_f . Let $D = \{D_n\}_{n \in \mathbb{N}}$ be another distribution ensemble, with each D_n supported over example strings $x \in \{0, 1\}^n$.

Definition 2.10 ([BFL93]). For any $\varepsilon : \mathbb{N} \rightarrow (0, 1/2)$, we say that $\mathcal{C}$ is ε -predictable on average over P and D if there exists an algorithm A such that for all sufficiently large $n \in \mathbb{N}$,

$$\Pr_{\substack{A; \sigma_f \sim P_n; \\ \text{EX}(f, D_n); \\ x' \sim D_n}} \left[A^{\text{EX}(f, D_n)}(x') = f(x') \right] \geq 1 - \varepsilon(n).$$

We say that $\mathcal{C}$ is *strongly predictable on average* over P and D if it is $(1/q(\cdot))$ -predictable on average for any polynomial q . We say it is *weakly predictable on average* over P and D if, for some polynomial q' , it is $(1/2 - 1/q'(\cdot))$ -predictable on average. We mention that weak predictability for the case of $\mathcal{C} = \text{P/poly}$ and all $P, D \in \text{PSAMP}$ implies strong predictability for such $\mathcal{C}$, P , and D [BFL93].

For $t : \mathbb{N} \rightarrow \mathbb{N}$, let $\text{SAMP}[t]$ denote the class of distributions samplable uniformly in time at most t . In the case of a heuristic learner A working on some fixed concept class $\mathcal{C}^s$, $P \in \text{SAMP}[t_1]$, and $D \in \text{SAMP}[t_2]$, the parameters s, t_1, t_2 may be given as additional inputs to A .

For further background on heuristic learnability and related notions, see [Nan21].

2.4 Theoretical Computer Science

We will require several notions from cryptography, complexity theory, Kolmogorov complexity, pseudorandomness, and meta-complexity. For general background in computational complexity, including randomized computation, nonuniform computation, cryptography, and circuit complexity, see [AB09].

2.4.1 Cryptography

Infinitely-Often Cryptography. For the purposes of cryptography, it is important for security to hold *almost everywhere*. That is, one would hope for a cryptographic primitive that cannot be broken at any (but perhaps finitely many) input lengths. Likewise, for the purposes of learning and positive algorithmic tasks more broadly, one would hope for algorithms to be successful at all but finitely many input lengths. While it is reasonable to ask for this property in both cases, it poses a problem if one would like to investigate dualities between the possibility of secure cryptography and the hardness of tasks such as learning. In particular, if the hardness of learning at some fixed input length translates into the hardness of breaking a cryptographic primitive at a related input length, then the impossibility of learning in the sense described above would only yield security at infinitely many input lengths.

For this reason, to aid in a better understanding of connections between algorithmic hardness and cryptography, we consider cryptography that may only be secure *infinitely often*.

A *pseudorandom function family* is a family of keyed polynomial-time computable functions that look like a random oracle to any efficient observer. Here, we state the definition of an infinitely-often variant.

Definition 2.11 (Infinitely-Often Pseudorandom Function). An *infinitely-often pseudorandom function*, denoted ioPRF, is a polynomial-time computable function family $f = \{f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}\}_{n \in \mathbb{N}}$ such that, for any inverse polynomial $\delta : \mathbb{N} \rightarrow [0, 1]$ and PPT machine A , there exist infinitely many $n \in \mathbb{N}$ such that

$$\left| \Pr_{A; u \sim \{0, 1\}^n} \left[A^{f(u, -)}(1^n) = 1 \right] - \Pr_{A; R_n} \left[A^{R_n(-)}(1^n) = 1 \right] \right| < \delta(n),$$

where $R_n : \{0, 1\}^n \rightarrow \{0, 1\}$ is a uniformly random function over the set of all functions with these parameters.

Auxiliary-Input Cryptography. We also consider variants of cryptographic primitives that are allowed to make use of an additional publicly available non-uniform input string. Since this kind of cryptography generalizes the standard setting in which no such auxiliary string is given, its existence is a weaker assumption.

As with infinitely-often cryptography, auxiliary-input cryptography can arise naturally when investigating connections between secure cryptography and the hardness of learning. In particular, a specific hard instance of learning might translate to a related description of a secure cryptographic primitive. In such settings, one may only obtain auxiliary-input cryptography rather than standard (uniform) cryptography.

An *auxiliary-input function family*

$$f = \{f_z : \{0, 1\}^{n(|z|)} \rightarrow \{0, 1\}^{\ell(|z|)}\}_{z \in \{0, 1\}^*}$$

is said to be *polynomial-time computable* if there is a polynomial-time machine F such that for all $z \in \{0, 1\}^*$ and $x \in \{0, 1\}^{n(|z|)}$, $F(z, x) = f_z(x)$.

A one-way function is a function that may be computed efficiently but cannot be inverted efficiently. This is a basic building block of many other objects in private-key cryptography. Here, we state the definition of the auxiliary-input variant.

Definition 2.12 (Auxiliary-Input One-Way Function [OW93]). An *auxiliary-input one-way function*, denoted AIOWF, is a polynomial-time computable auxiliary-input function family f such that, for any inverse polynomial δ and PPT machine A , there exist infinitely many $z \in \{0, 1\}^*$ such that

$$\Pr_{A; x \sim \mathcal{U}_{n(|z|)}} \left[A(z, f_z(x)) \in f_z^{-1}(f_z(x)) \right] < \delta(|z|).$$

Impagliazzo and Luby [IL89] proved that, if all candidate one-way functions can be inverted in the sense of finding *some* pre-image of a given output, then they can be inverted in the stronger sense of finding *nearly uniformly random* pre-images. Thus, the existence of one-way functions is equivalent to the existence of *distributional one-way functions*, which are secure against adversaries finding nearly uniformly random pre-images. More specifically, for a uniformly random input x and a poly-time computable function f , no adversary A on input $f(x)$ returns x' such that $(x', f(x))$ is close in statistical distance to $(x, f(x))$. Below, the auxiliary-input generalization is stated.

Definition 2.13 (Distributional Auxiliary-Input One-Way Function). A *distributional auxiliary-input one-way function*, denoted distAIOWF , is a polynomial-time computable auxiliary-input function family $f = \{f_z : \{0, 1\}^{n(|z|)} \rightarrow \{0, 1\}^{\ell(|z|)}\}_{z \in \{0, 1\}^*}$ such that, for some inverse polynomial δ , for every PPT machine A , there exist infinitely many $z \in \{0, 1\}^*$ such that

$$\Delta((A(z, f_z(x)), f_z(x)), (x, f_z(x))) > \delta(|z|),$$

where $x \sim \mathcal{U}_{n(|z|)}$, and Δ denotes statistical (total variation) distance.

A *pseudorandom generator* is a deterministic machine that maps a short uniformly random *seed* to a longer output string such that the distribution on outputs is indistinguishable from uniformly random to efficient observers.

Definition 2.14 (Auxiliary-Input Pseudorandom Generator). An *auxiliary-input pseudorandom generator*, denoted AIPRG , is a polynomial-time computable auxiliary-input function family $G = \{G_z : \{0, 1\}^{n(|z|)} \rightarrow \{0, 1\}^{\ell(|z|)}\}_{z \in \{0, 1\}^*}$ such that $\ell(|z|) > n(|z|)$, and for any inverse polynomial δ and PPT machine A , there exist infinitely many $z \in \{0, 1\}^*$ such that

$$\left| \Pr_{A; v \sim \mathcal{U}_{n(|z|)}} [A(z, G_z(v)) = 1] - \Pr_{A; u \sim \mathcal{U}_{\ell(|z|)}} [A(z, u) = 1] \right| \leq \delta(|z|).$$

We use the statement $\exists \text{AIPRG}$ to denote that the above exists for any choice of $\ell(|z|) = n(|z|)^{O(1)}$.

Definition 2.15 (Auxiliary-Input Pseudorandom Function). An *auxiliary-input pseudorandom function*, denoted AIPRF , is a polynomial-time computable auxiliary-input function family $f = \{f_z : \{0, 1\}^{n(|z|)} \times \{0, 1\}^{n(|z|)} \rightarrow \{0, 1\}\}_{z \in \{0, 1\}^*}$ such that, for any inverse polynomial δ and PPT machine A , there exist infinitely many $z \in \{0, 1\}^*$ such that

$$\left| \Pr_{A; u \sim \{0, 1\}^{n(|z|)}} [A^{f_z(u, -)}(z) = 1] - \Pr_{A; R_{|z|}} [A^{R_{|z|}(-)}(z) = 1] \right| < \delta(|z|),$$

where $R_{|z|} : \{0, 1\}^{n(|z|)} \rightarrow \{0, 1\}$ is a uniformly random function over the set of all functions with these parameters.

The next statement results from auxiliary-input analogues of standard transformations between one-way functions, distributional one-way functions, pseudorandom generators, and pseudorandom functions [IL89, HILL99, GGM86]. See, e.g., [ABX08] for a discussion.

Lemma 2.16. *The following statements are equivalent.*

- $\exists \text{AIOWF}$;

- $\exists \text{ distAIOFW}$;
- $\exists \text{ AIPRG}$;
- $\exists \text{ AIPRF}$.

For more information about one-way functions, pseudorandom generators, and pseudorandom functions, see [Gol01, KL07].

2.4.2 Kolmogorov Complexity

Kolmogorov complexity can be viewed as a universal measure of compression: it quantifies how concisely an individual object can be described by an effective procedure. Any particular compression scheme provides a way to encode certain objects succinctly, while a universal programming language can simulate every such scheme with only a fixed additional overhead. Consequently, Kolmogorov complexity captures, up to an additive constant, the best compression achievable by any computable method. Similarly, time-bounded Kolmogorov complexity captures the best compression achievable when decompression is required to run within a prescribed time bound.²

Formally, we fix a time-efficient universal machine U , and define time-bounded Kolmogorov complexity with respect to U (see [LV19] for the required background in Kolmogorov complexity). In the context of Kolmogorov complexity, whenever we say that a program M running on input y outputs x in t steps, formally, this means that $U(M, y, 1^t) = x$.

Definition 2.17 (K^t Complexity). For $x, y \in \{0, 1\}^*$ and $t \in \mathbb{N}$, the t -time-bounded Kolmogorov complexity of x conditioned on y is defined as

$$K^t(x \mid y) \triangleq \min_{M \in \{0, 1\}^*} \{|M| \mid U(M, y, 1^t) \text{ outputs } x\}.$$

We define $K^t(x)$ as $K^t(x \mid \epsilon)$.

The definition of K^t requires us to fix the time bound in advance. On the other hand, Levin's notion of time-bounded Kolmogorov complexity simultaneously considers description length and running time.

Definition 2.18 (Levin's Kt Complexity [Lev84]). For $x, y \in \{0, 1\}^*$, the *Levin complexity* of x conditioned on y is defined as

$$Kt(x \mid y) \triangleq \min_{t \in \mathbb{N}, M \in \{0, 1\}^*} \{|M| + \lceil \log t \rceil \mid U(M, y, 1^t) \text{ outputs } x\}.$$

We define $Kt(x)$ as $Kt(x \mid \epsilon)$.

When no time bound is considered, we simply write $K(x)$ to denote the (classical time-unbounded) Kolmogorov complexity of the string x .

We also consider *randomized* notions of time-bounded Kolmogorov complexity, in which the universal machine is given an additional random string as input. These complexity measures are particularly suitable in computational contexts where computations can be randomized, as is often the case with learning algorithms. Note that these measures include an additional parameter $\lambda \in [0, 1]$ to denote the probability with which the machine outputs the string in question. If λ is omitted, we take $\lambda = 2/3$.

²We refer to [Ble13] for an introduction to data compression.

Definition 2.19 (rK^t). For $x, y \in \{0, 1\}^*$, $\lambda \in [0, 1]$, $t \in \mathbb{N}$, the *randomized t -time-bounded Kolmogorov complexity* of x conditioned on y is defined as

$$\text{rK}_\lambda^t(x \mid y) \triangleq \min_{M \in \{0, 1\}^*} \left\{ |M| \mid \Pr_{r \sim \{0, 1\}^t} [U(M, y, r, 1^t) \text{ outputs } x] \geq \lambda \right\}.$$

We define $\text{rK}^t(x)$ as $\text{rK}^t(x \mid \epsilon)$.

Note that in rK^t , the string still has a deterministic, standard shorter description encoding a program; it is the program's execution that is randomized, so the natural decoding procedure uses randomness to recover the string. Randomization can provide significant savings in description length in some concrete settings [CLO⁺23]. Nevertheless, under derandomization assumptions from complexity theory, it follows that $\text{K}^{\text{poly}(t)}(x)$ and $\text{rK}^{\text{poly}(t)}(x)$ are related up to an additive logarithmic term for every large enough string x (see, e.g., [GKLO22, Appendix A.2]).

In recent years, another variant of time-bounded Kolmogorov complexity has been particularly useful due to its coding properties [LOZ22] and connections to pseudorandomness.

Definition 2.20 (pK^t [GKLO22]). For $x, y \in \{0, 1\}^*$, $\lambda \in [0, 1]$, $t \in \mathbb{N}$, the *probabilistic t -time-bounded Kolmogorov complexity* of x conditioned on y is defined as

$$\text{pK}_\lambda^t(x \mid y) \triangleq \min \left\{ k \mid \Pr_{r \sim \{0, 1\}^t} [\exists M \in \{0, 1\}^{\leq k} \text{ such that } U(M, y, r, 1^t) \text{ outputs } x] \geq \lambda \right\}.$$

We define $\text{pK}^t(x)$ as $\text{pK}^t(x \mid \epsilon)$.

Equivalently, $\text{pK}^t(x) \leq k$ if $\Pr_r[\text{K}^t(x \mid r) \leq k] \geq 2/3$. As a consequence, a string x has bounded pK^t complexity if for most choices of r it admits a small description given access to r . Note that here the description of x depends on r . This captures the setting where both encoder and decoder share access to a random string r . In contrast, in rK^t there is a fixed randomised machine that produces x with high probability over its internal randomness.³

We will also consider a randomized variant of Levin's Kt complexity that admits particularly useful coding properties (see [HLO26a] and references therein for additional context).

Definition 2.21 (rKt [Oli19]). For $x, y \in \{0, 1\}^*$, $\lambda \in [0, 1]$, the *randomized Levin complexity* of x conditioned on y is defined as

$$\text{rKt}_\lambda(x \mid y) \triangleq \min_{t \in \mathbb{N}, M \in \{0, 1\}^*} \left\{ |M| + \lceil \log t \rceil \mid \Pr_{r \sim \{0, 1\}^t} [U(M, y, r, 1^t) \text{ outputs } x] \geq \lambda \right\}.$$

We define $\text{rKt}(x)$ as $\text{rKt}(x \mid \epsilon)$.

For completeness, we mention that one can also consider a “probabilistic” variant of Kt called pKt [HLO24], which is similar to the extension of K^t to pK^t . While we will not employ pKt in the survey, we refer to [Pas26] for a recent application of a related notion in the context of machine learning.

A choice of universal machine U gives rise to a certain *universal distribution* that may be sampled by returning the output of U on a uniformly random input. In the case of time-unbounded Kolmogorov

³For the reader familiar with interactive proofs in the context of complexity theory, we mention that the distinction between rK^t and pK^t is analogous to the definitions of the complexity classes MA and AM , respectively.

complexity, for any string x , $K(x)$ is approximately equal to the logarithm of the inverse of the probability of x under this distribution.

Analogously, one may define a *time-bounded universal distribution* for a given time bound $t \in \mathbb{N}$. Namely, for a fixed universal machine U , [IL90] define $\mathcal{Q}^t$ as the distribution of the output of $U(r, 1^t)$ for a uniformly random string $r \sim \{0, 1\}^t$. For any auxiliary string $y \in \{0, 1\}^*$, let $\mathcal{Q}^{t,y}$ be the distribution with the same definition as $\mathcal{Q}^t$, except the universal machine U is additionally given access to y .

The following is the complexity measure corresponding to this family of distributions.

Definition 2.22 (q^t [HN23]). For $x, y \in \{0, 1\}^*$ and $t \in \mathbb{N}$, the q^t -complexity of x conditioned on y is defined as

$$q^t(x | y) \triangleq \log \left(\frac{1}{\Pr[\mathcal{Q}^{t,y} = x]} \right).$$

We define $q^t(x)$ as $q^t(x | \epsilon)$; that is, $\log(1 / \Pr[\mathcal{Q}^t = x])$.

We refer to the textbooks [LV19, SUV17] for a comprehensive treatment of Kolmogorov complexity. For a modern perspective on time-bounded Kolmogorov complexity, see [LO22].

2.4.3 Circuit Complexity

A *boolean circuit* is a finite directed acyclic graph with one or more source nodes, one sink node, and each internal node corresponding to some boolean function $g : \{0, 1\}^* \rightarrow \{0, 1\}$ from a fixed set G , called a *basis*. A standard choice of G is the set $\{\text{AND}, \text{OR}, \text{NOT}\}$. Each node of a boolean circuit is called a *gate*, with source nodes being called *input gates* and the sink node being called the *output gate*.⁴ A boolean circuit C with n input nodes induces a boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, computed by associating each bit of a given input string $x \in \{0, 1\}^n$ with the corresponding input gate of C , performing the computation specified by the structure of C and the functions of the internal gates, and taking $f(x) \in \{0, 1\}$ to be the value resulting at the output gate.

Since a single boolean circuit has a fixed number of input gates, it computes a function on inputs of only one length. To capture a boolean function $f : \{0, 1\}^* \rightarrow \{0, 1\}$ defined on all input lengths, we consider a *circuit family* $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$, where each C_n has n input gates and computes the restriction of f to $\{0, 1\}^n$. Two measures of the cost of a circuit are its *size*, i.e. the number of gates it contains, and its *depth*, i.e. the length of the longest path from an input gate to the output gate.

We use the term *fan-in* to refer to the number of inputs to a gate. For unrestricted boolean circuits, we adopt the convention that the fan-in of each gate is at most two. For bounded-depth circuit classes, we allow the fan-in to be unbounded unless stated otherwise. Note that NOT gates always have fan-in equal to 1.

From the perspective of compression, the truth table of a Boolean function f can be seen as a long string of length $N = 2^n$. A small circuit C that computes f can then be interpreted as a compressed representation with a local decoder: given an index $i \in [N]$, $C(i)$ outputs the i -th bit of f .

Next, we define several classes of boolean functions, which may all be specified according to a class of boolean circuit families.

Definition 2.23 (P/poly). P/poly is the class of boolean functions computable by a circuit family $\{C_n\}$ over the basis $\{\text{AND}, \text{OR}, \text{NOT}\}$, where for some constant $d \in \mathbb{N}$, each C_n has size at most n^d .

⁴One may also consider circuits with multiple output gates, but we are mainly interested here in circuits representing boolean functions, which have only one bit of output.

Definition 2.24 (AC^0). AC^0 is the class of boolean functions computable by a circuit family $\{C_n\}$ over the basis $\{\text{AND}, \text{OR}, \text{NOT}\}$, where for some constants $c, d \in \mathbb{N}$, each C_n has size at most n^c and depth at most d .

Definition 2.25 (TC^0). The class TC^0 has the same definition as AC^0 , except circuits are over the larger basis $\{\text{AND}, \text{OR}, \text{NOT}, \text{THR}\}$, where THR is a *threshold* gate (also known as a weighted majority), outputting 1 iff $\sum_i w_i \cdot z_i \geq t$, where $w_i, t \in \mathbb{R}$.⁵

TC^0 circuits, also known as threshold circuits, can be interpreted as a discrete analogue of neural networks. We recall that a current frontier in complexity theory is to establish super-polynomial circuit size lower bounds against depth-2 threshold circuits for an explicit function (see [KW16, CTW26] for partial progress and related discussion). We refer to Figure 2 for a diagram representation of this circuit class.

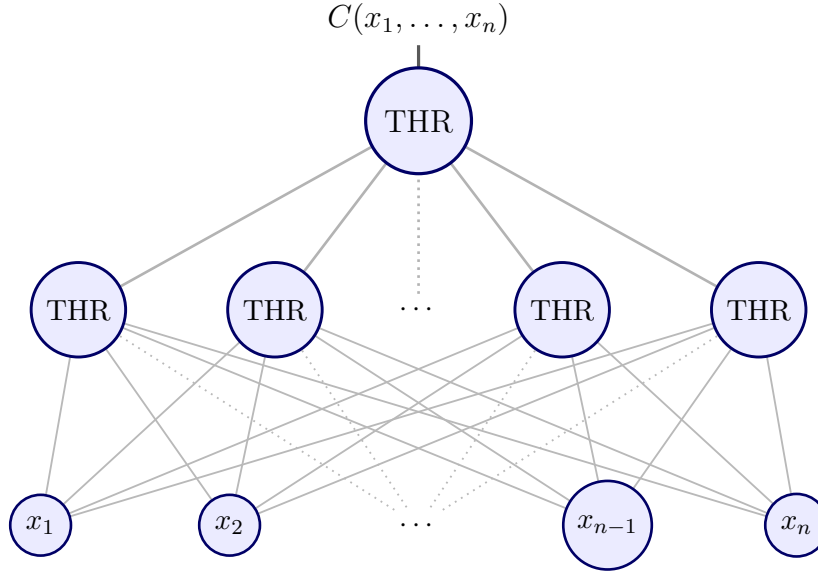


Figure 2: A depth-2 threshold circuit C . The top threshold gate receives the outputs taking values in $\{0, 1\}$ from the threshold gates in the bottom layer, each of which is connected to the input variables $x_1, \dots, x_n$.

Definition 2.26 (AC^0 with MOD_m gates). For a modulus $m \geq 2$, let MOD_m output 1 iff the number of its inputs equal to 1 is divisible by m . $AC^0[m]$ has the same definition as AC^0 , except circuits are over the basis $\{\text{AND}, \text{OR}, \text{NOT}, MOD_m\}$. The case of $m = 2$ gives the parity gate, sometimes written $AC^0[\oplus]$. ACC^0 is the class defined as $\bigcup_{m \geq 2} AC^0[m]$.

The following inclusion of circuit classes hold:

$$AC^0 \subseteq AC^0[\oplus] \subseteq ACC^0 \subseteq TC^0 \subseteq P/\text{poly}.$$

For standard background on Boolean circuits, nonuniform computation, and circuit classes, see [AB09, Juk12].

⁵Since each AND, OR, and NOT gate can be simulated by a THR gate with appropriate weights w_i and threshold t , we typically assume that TC^0 circuits contain THR gates only.

2.4.4 Meta-Complexity

Broadly speaking, problems in *meta-complexity* ask about the complexity of a given object. One may restrict these problems to different kinds of object (for example, the input may simply be a string, or it may be understood to represent a partly or fully specified truth-table), and one may ask about different measures of complexity (for example, one may ask about the circuit complexity or the Kolmogorov complexity of the input).

We begin with the Minimum Circuit Size Problem [KC00], a prototypical example of a problem in meta-complexity.

Definition 2.27 (MCSP). The *Minimum Circuit Size Problem*, denoted MCSP, is the problem defined as follows. The input consists of a truth-table T , represented as a string, along with a size threshold $s \in \mathbb{N}$. (T, s) is a yes-instance of MCSP if and only if there exists a boolean circuit of size at most s computing the function with truth-table T .

Definition 2.28 (Partial-MCSP). Partial-MCSP is the problem defined as follows. For some $n, m \in \mathbb{N}$, the input consists of a collection of input/output pairs $((x_i, b_i))_{i \in [m]}$, where $x_i \in \{0, 1\}^n$ and $b_i \in \{0, 1\}$ for each $i \in [m]$, and a threshold parameter $s \in \mathbb{N}$. $((x_i, b_i)_{i \in [m]}, s)$ is a yes-instance of Partial-MCSP iff there exists a boolean circuit C of size at most s such that $C(x_i) = b_i$ for all $i \in [m]$.

Next, we will consider computational problems related to the task of estimating the time-bounded Kolmogorov complexity of an input string (see [Sim23] for further context and resources).

Definition 2.29 (MINKT). Inputs to the problem MINKT consist of a string x along with a size threshold $s \in \mathbb{N}$ and time parameter $t \in \mathbb{N}$. $(x, 1^s, 1^t)$ is a yes-instance of MINKT if and only if $K^t(x) \leq s$.

Observe that MINKT is in NP. It is not in P under the existence of cryptographically secure one-way functions [KC00]. Interestingly, while believed to be intractable, MINKT is not known to be NP-complete (see [HLO26b] and references therein).

For the connections to learning and cryptography discussed in Section 4.3, the complexity of certain robust variants of MINKT is more relevant. We will informally refer to Gap-MINKT as the task of estimating the K^t complexity of an input string x of length n up to an additive term $O(\log n)$.⁶ We say that “Gap-MINKT $\in P$ ” if there is a polynomial-time algorithm that solves this problem on every input. We refer to [Hir21] for the relevant formal definition. We say that “Gap-MINKT $\in \text{HeurP}$ ” if there is a polynomial-time algorithm that is heuristically correct, i.e., it solves the problem on most inputs (but might output an incorrect estimate on a small fraction of inputs). We refer to [LP21] for the relevant formal definition.

It is known that the task of estimating the K^t complexity of an input string is essentially equivalent (under polynomial-time reductions) to the following problem [Hir21].

Definition 2.30 (The Gap K^t -vs- K Problem). We say that Gap K^t -vs- $K \in \text{pr-BPP}$ if there is a randomized polynomial-time algorithm A such that for every input $x \in \{0, 1\}^*$ of length n and every time bound $t \in \mathbb{N}$,

$$\Pr_A \left[K(x) \leq A(x, 1^t) \leq K^t(x) + O(\log n) \right] \geq 2/3.$$

Similarly, we consider the complexity of computing Levin’s K_t complexity. In the MKtP problem, we are given as input a tuple $(x, 1^s)$, and the task is to decide if $K_t(x) \leq s$. This problem is in the complexity class EXP, i.e., deterministic exponential time.

⁶A bit more formally, in this promise problem we are interested in distinguishing $K^t(x) \leq s$ from $K^{\text{poly}(t)}(x) \geq s + O(\log n)$.

While the statement $\text{MINKT} \notin \text{P}$ immediately implies that $\text{NP} \not\subseteq \text{P}$, a proof that $\text{MKtP} \notin \text{P}$ merely implies that $\text{EXP} \not\subseteq \text{P}$, which is already known by the time hierarchy theorem. Still, it remains a long-standing open problem to establish this lower bound for MKtP [ABK⁺06]. We will revisit this question in Section 7.3.

We refer to some general surveys of meta-complexity for broader context [All17, All21, Hir22a, MP25].

2.4.5 Natural Properties

In an influential work, [RR97] showed that many combinatorial circuit size lower bound arguments implicitly provide an efficient algorithm that distinguishes “structured” functions from “random” functions. As explained later in the survey, this provides a fruitful connection between complexity lower bounds and algorithmic upper bounds.

Let $\mathcal{C}$ be a Boolean circuit class, and for a size parameter $s: \mathbb{N} \rightarrow \mathbb{N}$, let $\mathcal{C}[s]$ denote the set of Boolean functions $f: \{0, 1\}^n \rightarrow \{0, 1\}$ computed by a $\mathcal{C}$ -circuit of size at most $s(n)$. We write $\mathcal{C}_n[s(n)]$ when we restrict the class to Boolean functions on n input bits.

A function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ can be represented by a string $\text{tt}(f) \in \{0, 1\}^N$ of length $N = 2^n$ that describes the truth table of f . Similarly, given a string $w \in \{0, 1\}^N$, we write $\text{fnc}^w: \{0, 1\}^n \rightarrow \{0, 1\}$ to denote the corresponding Boolean function.

In the discussion below, we will consider algorithms that receive as input the description of a Boolean function as an N -bit string. Roughly speaking, a natural property is an efficient algorithm that rejects every function in $\mathcal{C}[s]$ but accepts a large set of functions. In other words, it provides an efficient test to reject functions of low complexity while at the same time accepting a large collection of “random” functions.

Definition 2.31 (Natural Properties [RR97]). We say that there is a *prBPP-natural property* against a circuit class $\mathcal{C}[s]$ if there is a probabilistic polynomial-time algorithm A operating over input strings of length $N = 2^n$ for which the following conditions hold:

- *Constructive*: A runs in time polynomial in its input length N .
- *Largeness*: For every n and $N = 2^n$, there is a set $\mathcal{F}_n \subseteq \{0, 1\}^N$ with $\Pr_{w \sim \mathcal{U}_N}[w \in \mathcal{F}_n] \geq 1/2$ such that for every $w \in \mathcal{F}_n$, we have $\Pr_A[A(w) = 1] \geq 2/3$.
- *$\mathcal{C}[s]$ -usefulness*: For every n and $f: \{0, 1\}^n \rightarrow \{0, 1\}$ such that $f \in \mathcal{C}_n[s(n)]$, we have $\Pr_A[A(\text{tt}(f)) = 1] \leq 1/3$.

In the definition above, we use prBPP instead of BPP because the randomized procedure A might accept some strings with probability close to $1/2$. In the terminology of complexity theory, it might define a promise problem as opposed to a language. We refer to Figure 3 for a pictorial description of a natural property.

For broader background on circuit lower bounds and the natural proofs barrier, see [AB09, Juk12].

2.4.6 Pseudorandomness

A pseudorandom generator with *seed length* n and *output length* $m > n$ is a deterministic map

$$G: \{0, 1\}^n \rightarrow \{0, 1\}^m.$$

For a uniform seed $s \sim \mathcal{U}_n$, we write $G(\mathcal{U}_n)$ for the distribution of $G(s)$. The quantity $m - n$ is called the *stretch* of G . More generally, one considers families $\{G_n\}_{n \in \mathbb{N}}$ whose output length $m = m(n)$ grows with the seed length.

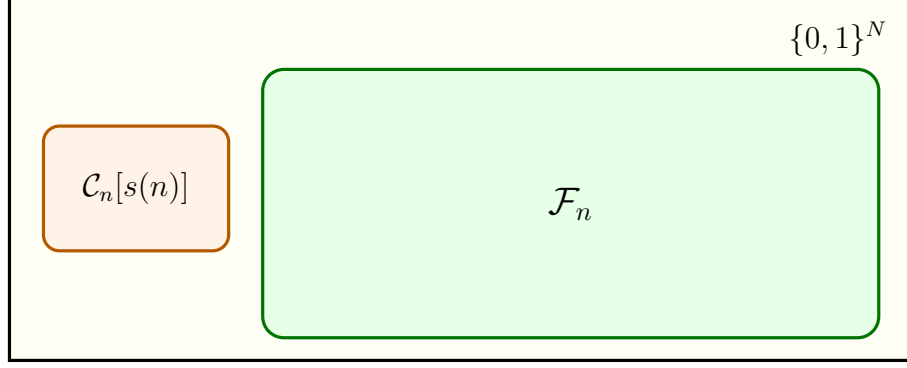


Figure 3: A diagram representation of a natural property (Definition 2.31). The outer rectangle is the ground set of all N -bit strings. The small region represents the “structured” strings corresponding to functions in the low complexity class $\mathcal{C}_n[s(n)]$. The large region represents the “random” strings in $\mathcal{F}_n \subseteq \{0, 1\}^N$. Its relative size illustrates the *large-ness* condition, while $\mathcal{F}_n \cap \mathcal{C}_n[s(n)] = \emptyset$ illustrates *usefulness* against $\mathcal{C}_n[s(n)]$. A natural property is *constructive*: it postulates the existence of an efficient algorithm that distinguishes these two sets, thereby separating the structured strings in $\mathcal{C}_n[s(n)]$ from the random strings in $\mathcal{F}_n$.

Definition 2.32 (Distinguishers and Pseudorandomness). A *distinguisher* is a Boolean function, or more generally a randomized procedure, $D: \{0, 1\}^m \rightarrow \{0, 1\}$. Its distinguishing advantage against G is

$$\text{Adv}_G(D) \triangleq \left| \Pr_{s \sim \mathcal{U}_n} [D(G(s)) = 1] - \Pr_{y \sim \mathcal{U}_m} [D(y) = 1] \right|,$$

where the probabilities also include the internal randomness of D , if any. We say that D ε -*distinguishes* the output of G from the uniform distribution if $\text{Adv}_G(D) > \varepsilon$. Given a class $\mathcal{C}$ of distinguishers, we say that G *fools* $\mathcal{C}$ *with error* ε if $\text{Adv}_G(D) \leq \varepsilon$ for every $D \in \mathcal{C}$. Equivalently, G is ε -*pseudorandom* against $\mathcal{C}$.

Pseudorandom generators commonly appear in two regimes. In the *cryptographic regime*, G is efficient and must fool distinguishers whose running time may be substantially larger than the time needed for one evaluation of G . The existence of cryptographic pseudorandom generators is equivalent, under polynomial-time reductions, to the existence of one-way functions and pseudorandom functions. These primitives are often used to establish conditional hardness results for learning.

In the *complexity-theoretic regime*, the generator may use more time than the distinguishers it fools. Here a generator is often accompanied by a *reconstruction procedure* that converts a distinguisher into useful computational structure. Such generators and reconstruction procedures will later be used to construct learning algorithms.

The survey will use several variants of these notions. We record the basic definitions here as a guide for the reader and refer to Vadhan’s monograph [Vad12] for a comprehensive treatment.

3 Classical Connections Between Learning and Compression

We start off with two classical connections between learning and compression. They allow us to link learnability, a statistical notion, with compressibility, a combinatorial one. In many contexts, this offers a particularly fruitful perspective for the design of learning algorithms.

3.1 Model Compression

We start with one of the most basic connections between learning and compression. In the (realizable) PAC learning setting, a labelled sample may contain many examples, but a learner can summarize the information they exhibit by a comparatively short hypothesis (“model”) that agrees with every observed label. We refer to this operation as *model compression*.⁷

Why should a compressed model generalize well, i.e., remain accurate on fresh examples that are not part of the training data? The answer is the principle commonly known as *Occam’s Razor*: among explanations that fit the observations, a simple explanation is less likely to fit merely by accident [BEHW87]. The informal justification is easy to understand: any fixed substantially incorrect hypothesis is unlikely to agree with many independent examples, and a short model description leaves few candidate hypotheses over which this unlikely event must be union bounded. Thus compression of the model controls generalization.

We now formalize this intuition. Let $\mathcal{C}$ be a concept class and $\mathcal{H}$ a hypothesis class, where $\mathcal{H}$ need not be equal to $\mathcal{C}$. For a distribution D_n over $\{0, 1\}^n$, define

$$\text{err}_{D_n}(h, c) \triangleq \Pr_{x \sim D_n} [h(x) \neq c(x)].$$

Definition 3.1 (Consistent Hypothesis Finder). A *consistent hypothesis finder* for $\mathcal{C}$ using $\mathcal{H}$ is a procedure A with the following property. For every $n, m \geq 1$, every $c \in \mathcal{C}_n$, and every labelled sample

$$S = \{(x_1, c(x_1)), \dots, (x_m, c(x_m))\},$$

$A(S)$ outputs a hypothesis $h \in \mathcal{H}_n$ such that $h(x_i) = c(x_i)$ for every $i \in [m]$.

Thus a consistent hypothesis finder solves a search problem with no explicit reference to a distribution or to the PAC parameters ε and δ . Note that the hypothesis class $\mathcal{H}$ records the set of possible models, and it may differ from the target class $\mathcal{C}$. The next result shows that consistency implies generalization whenever the range of possible output models is sufficiently small and the number of samples is large enough.

Theorem 3.2 (PAC Learning by Finding a Consistent Hypothesis [BEHW87]). *Let $\mathcal{C}$ be a concept class, $\mathcal{H}$ be a hypothesis class, and suppose that A is a consistent hypothesis finder for $\mathcal{C}$ using $\mathcal{H}$. Fix $n \geq 1$ and assume that $\mathcal{H}_n$ is finite. Let D_n be any distribution over $\{0, 1\}^n$, let $c \in \mathcal{C}_n$, and let $\varepsilon, \delta \in (0, 1)$. If A is run on m independent examples from $\text{EX}(c, D_n)$, where*

$$m \geq \frac{1}{\varepsilon} \left(\ln |\mathcal{H}_n| + \ln \frac{1}{\delta} \right),$$

then, with probability at least $1 - \delta$ over the sample, its output hypothesis h satisfies $\text{err}_{D_n}(h, c) \leq \varepsilon$.

⁷This differs from the *sample compression* schemes considered in the next subsection, where the learner retains a small part of the sample from which a hypothesis can later be reconstructed.

Proof. Let c be the unknown target concept, and let $\mathcal{B} \subseteq \mathcal{H}_n$ be the set of *bad* hypotheses h satisfying $\text{err}_{D_n}(h, c) > \varepsilon$. Fix $h \in \mathcal{B}$. A single example $(x, c(x))$ drawn from $\text{EX}(c, D_n)$ is consistent with h with probability $1 - \text{err}_{D_n}(h, c) < 1 - \varepsilon$. Since the m examples are independent, the probability that h is consistent with the entire sample is at most

$$(1 - \varepsilon)^m \leq e^{-\varepsilon m}.$$

If the consistent hypothesis finder outputs a bad hypothesis, then the sample must be consistent with at least one member of $\mathcal{B}$. Consequently, a union bound gives

$$\begin{aligned} \Pr[\text{err}_{D_n}(A(S), c) > \varepsilon] &\leq \Pr[\exists h \in \mathcal{B} \text{ consistent with } S] \\ &\leq \sum_{h \in \mathcal{B}} \Pr[h \text{ is consistent with } S] \\ &\leq |\mathcal{H}_n| e^{-\varepsilon m} \\ &\leq \delta, \end{aligned}$$

where the final inequality follows from the assumed lower bound on m . $\square$

The role of model compression can now be made explicit. Suppose every $h \in \mathcal{H}_n$ can be encoded using $\ell(n)$ bits. Then $\ln |\mathcal{H}_n| \leq \ell(n) \ln 2$ and it is therefore enough to take

$$m \geq \frac{1}{\varepsilon} \left(\ell(n) \ln 2 + \ln \frac{1}{\delta} \right).$$

Thus the number of examples required for generalization is controlled by the description length of the output model. The training sample is described by $m(n+1)$ bits, while the explanation extracted from it uses only $\ell(n)$ bits. If the consistent hypothesis finder runs in polynomial time, $\ell(n) = \text{poly}(n)$, and its output hypotheses are efficiently evaluable under the encoding, then this construction gives an efficient PAC learner.

To sum up, compressing the observations into a short model prevents overfitting because there are too few short models for a substantially incorrect one to agree with all the data, except with small probability.

A Classical Example: Learning Through Polynomial Threshold Representations. An important example of the preceding model compression perspective is the class of *linear threshold functions* (LTFs), also known as halfspaces or perceptrons. Such a function has the form $h(x) = \text{sign}(w_0 + \langle w, x \rangle)$. Given a labelled sample realizable by a linear threshold function, consistency can be expressed as a linear program. After encoding the labels as elements of $\{\pm 1\}$, one seeks w_0, w satisfying $y_i(w_0 + \langle w, x_i \rangle) \geq 1$ for every sample point (x_i, y_i) . Realizability guarantees feasibility after a suitable rescaling, and linear programming finds rational weights of polynomial description length in polynomial time. Thus linear programming provides an efficient consistent hypothesis finder. Moreover, there are only $2^{O(n^2)}$ distinct linear threshold functions on $\{0, 1\}^n$, so the preceding [Theorem 3.2](#) shows that the resulting hypothesis generalizes from polynomially many examples. This yields a distribution-independent PAC learner for linear threshold functions.

Feature expansion extends this observation to *polynomial threshold functions* (PTFs), the natural extension of LTFs to higher degree. Let $\Phi_d(x)$ be the vector whose coordinates are all multilinear monomials $\prod_{i \in T} x_i$ with $|T| \leq d$. Its dimension is $N = \sum_{j=0}^d \binom{n}{j} \leq (en/d)^d$, and every degree- d polynomial threshold function becomes a linear threshold function over the expanded representation $\Phi_d(x)$. Mapping

each example through Φ_d and applying the preceding linear programming algorithm therefore gives a PAC learner with running time $\text{poly}(n^d, 1/\varepsilon, \log(1/\delta))$. Consequently, proving that a concept class has polynomial threshold degree at most d immediately gives a learning algorithm with this running time. The learner need not know a representation of the target, since the linear program searches for any consistent polynomial threshold function.

This approach yields learning algorithms with provable guarantees for rich concept classes such as decision trees, DNF formulas, bounded-depth Boolean formulas, and more. We refer to the survey [HS07] for a comprehensive discussion of this technique and its applications.

The Occam principle and closely related ideas appear in the literature under several names. It admits several generalizations. For instance, infinite hypothesis classes can be treated using the theory of VC dimension (e.g., LTFs over $\mathbb{R}^n$), while exact consistency can be relaxed by allowing the hypothesis finder to output an h that agrees with most examples in the sample. For further background, we refer to standard textbooks such as [KV94a, SSBD14].

3.2 Sample Compression

The preceding subsection compressed a labelled sample into a short model. Sample compression takes a different approach: it retains only a few of the observed examples, together with a small amount of side information, and reconstructs a hypothesis from this summary. The reconstructed hypothesis need not itself have a short standalone description. What matters is that, relative to the original sample, it is determined by very few examples.

Fix an input length n . A *sample compression scheme* for a concept class $\mathcal{C}_n$ with parameters (k, ℓ) consists of a compression map Comp and a reconstruction map Rec with the following guarantee. Given any labelled sample sequence

$$S = ((x_1, c(x_1)), \dots, (x_m, c(x_m)))$$

consistent with some unknown $c \in \mathcal{C}_n$, the map Comp outputs a pair $(S|_I, \sigma)$, where $I \subseteq [m]$ has size at most k and $\sigma \in \{0, 1\}^\ell$. Here $S|_I$ denotes the subsequence of S indexed by I , listed in increasing order of indices. The reconstruction map uses only this pair to produce a hypothesis

$$h = \text{Rec}(S|_I, \sigma)$$

that is consistent with the entire original sample S . As usual, strings shorter than ℓ can be padded, and the reconstructed hypothesis may be improper. We call $k + \ell$ the *size* of the scheme.

A Classical Example: Rectangles in the Plane. Although we stated the definition over the Boolean cube, sample compression applies without change to arbitrary (possibly infinite) instance domains. Consider the class of axis-aligned rectangles in $\mathbb{R}^2$, where an example is labelled positive precisely when it belongs to an unknown target rectangle R^* . Given a realizable sample containing at least one positive example, Comp retains a leftmost, a rightmost, a lowest, and a highest positive example, breaking ties according to their order in the sample and retaining only distinct examples. The reconstruction map Rec outputs the smallest axis-aligned rectangle R_S containing these points. (If there are no positive examples, the compression is empty and Rec returns the empty hypothesis.) Every positive example lies in R_S by the choice of the four extreme points. On the other hand, $R_S \subseteq R^*$, since it is the bounding rectangle of points belonging to the axis-aligned rectangle R^* . Therefore every negative example lies outside R_S , and the reconstructed rectangle is consistent with the entire sample. This gives a sample compression scheme with parameters

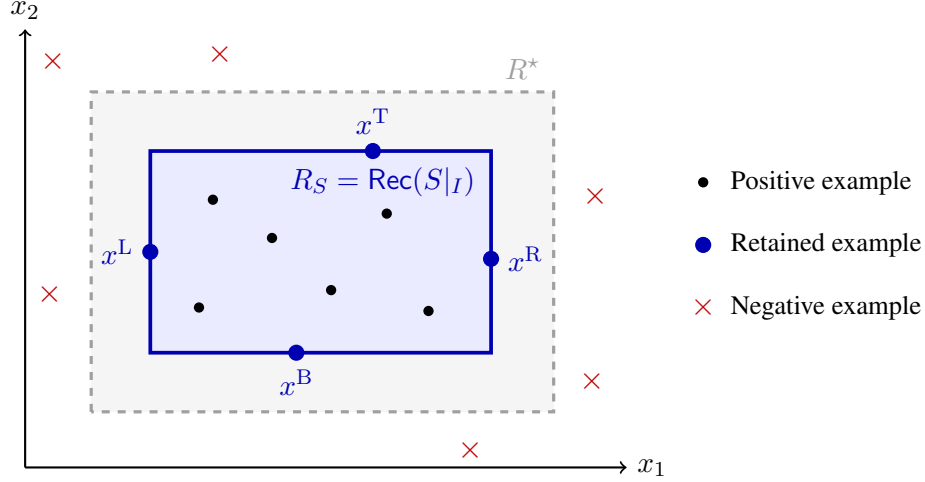


Figure 4: A sample compression scheme for axis-aligned rectangles in the plane. The four extreme positive examples determine the reconstructed rectangle R_S . Since $R_S \subseteq R^*$, where R^* denotes the unknown target rectangle, R_S contains every positive sample point and no negative sample point. All labels in the original sample can be reconstructed from the four extreme positive points.

($k = 4, \ell = 0$). We refer to Figure 4 for a pictorial description.

Next, we explain why any such compression scheme yields a PAC learner.

Theorem 3.3 (PAC Learning from Sample Compression [LW86]). *Suppose that $\mathcal{C}_n$ has a sample compression scheme with parameters (k, ℓ) . Let D_n be any distribution over $\{0, 1\}^n$, let $c \in \mathcal{C}_n$, and let $\varepsilon, \delta \in (0, 1)$. Draw m independent examples from $\text{EX}(c, D_n)$, apply the compression map, and output the reconstructed hypothesis h . Then*

$$\Pr[\text{err}_{D_n}(h, c) > \varepsilon] \leq 2^\ell \sum_{i=0}^k \binom{m}{i} e^{-\varepsilon(m-i)}.$$

In particular, it suffices to take

$$m = O\left(\frac{k \log(2/\varepsilon) + \ell + \log(1/\delta)}{\varepsilon}\right)$$

to obtain error at most ε with probability at least $1 - \delta$.

Proof Sketch. The subtlety compared with the model compression argument is that the reconstructed hypothesis depends on the retained examples, which are themselves random. We therefore cannot fix the hypothesis before drawing the sample and charge it for agreeing with all m examples.

Instead, fix a set of indices $I \subseteq [m]$ of size at most k and a message $\sigma \in \{0, 1\}^\ell$, and condition on the examples indexed by I . The reconstructed hypothesis $h_{I,\sigma} = \text{Rec}(S|_I, \sigma)$ is now fixed, while the remaining $m - |I|$ examples are still independent of it. If $\text{err}_{D_n}(h_{I,\sigma}, c) > \varepsilon$, the probability that it agrees with all these remaining examples is therefore at most

$$(1 - \varepsilon)^{m-|I|} \leq e^{-\varepsilon(m-|I|)}.$$

Although the compression map may choose (I, σ) adaptively after seeing the entire sample, if the scheme outputs a bad hypothesis then this event occurs for its selected pair. A union bound over the at most $2^\ell \binom{m}{i}$ choices with $|I| = i$, for every $0 \leq i \leq k$, therefore gives the claimed probability bound. The standard estimate $\sum_{i=0}^k \binom{m}{i} \leq (em/k)^k$, together with the stated choice of m , makes this bound at most δ . $\square$

Thus a bounded sample compression scheme yields a PAC learner. If compression, reconstruction, and evaluation of the reconstructed hypothesis are efficient, then the resulting learner is efficient as well. Without these requirements, the conclusion is information-theoretic only, providing an upper bound on the sample complexity of learning $\mathcal{C}_n$.

Remarkably, the converse also holds at the qualitative level in the information-theoretic setting. A (possibly infinite) binary concept class is PAC learnable exactly when it has finite VC dimension, and Moran and Yehudayoff [MY16] proved that every class of VC dimension d has a sample compression scheme of size $2^{O(d)}$. Consequently, for binary concept classes, the existence of a bounded sample compression scheme characterizes PAC learnability, even though the known quantitative relationship is far from tight.

Problem 3.4 (Sample Compression Conjecture). Does every binary concept class of VC dimension d admit a sample compression scheme of size $O(d)$?

Closing the existing exponential gap would show that the amount of information one needs to retain from a sample is governed, up to constant factors, by the same parameter that controls PAC sample complexity.

4 Complexity-Theoretic Foundations of Learning and Compression

4.1 Learning Paradises

We consider five possible learning scenarios, ordered from a strong unsupervised setting to a far weaker supervised setting in which only weak heuristic learnability is assumed. Still, even the weakest of these assumptions asks for learning power over *unrestricted* Boolean circuits, a capability well beyond current theoretical results and practical algorithms (see [Section 4.3](#) for more details). We therefore call each scenario a learning paradise.

- **P.I: Efficient Extrapolation.**

There are polynomials $p_1(\cdot), p_2(\cdot)$ and a randomized algorithm L running in time $p_1(n, m, s, t, 1/\varepsilon, 1/\delta)$ that solves the $(m, n, s, t, \varepsilon, \delta)$ -extrapolation task whenever $m \geq p_2(s, 1/\varepsilon, 1/\delta)$.

- **P.II: Efficient Distribution Learning.**

There is a polynomial $p(\cdot)$ and a randomized algorithm A running in time $p(n, s, t, 1/\varepsilon, 1/\delta)$ that ε -learns every distribution $D \in \mathcal{D}_n^{s,t}$ with probability at least $1 - \delta$.

- **P.III: Efficient Agnostic Learning (Under Feasible Example Distributions).** There is a polynomial $p(\cdot)$ and a randomized algorithm A running in time $p(n, 1/\varepsilon, \log(1/\delta), s, t, a)$ that agnostically learns $\text{SIZE}[s]$ over $\mathcal{D}^{a,t}$.

- **P.IV: Efficient PAC Learning (with MQs and Under Feasible Example Distributions).** There is a polynomial $p(\cdot)$ and a randomized algorithm A running in time $p(n, 1/\varepsilon, \log(1/\delta), s, t, a)$ that PAC learns $\text{SIZE}[s]$ over $\mathcal{D}^{a,t}$ with membership queries.

- **P.V: Efficient Heuristic Learning (without MQs).** For any polynomials $s(\cdot), t_1(\cdot), t_2(\cdot)$ and distribution families $P \in \text{SAMP}[t_1]$ and $D \in \text{SAMP}[t_2]$, there is a polynomial $p(\cdot)$ and a randomized algorithm A running in time $p(n, s, t_1, t_2)$ that weakly predicts $\text{SIZE}[s]$ on average over P and D .

There are many natural variants of assumption P.IV. For example, one can vary whether the learner is allowed to make membership queries, whether the examples come from the uniform distribution or from arbitrary distributions, and whether the learner is proper. To keep the presentation simple and to avoid the overly pessimistic distribution-free model, we adopt the membership query PAC model over feasible distributions as a representative supervised learning scenario.

Learning under feasible distributions is a natural restriction in complexity theoretic and cryptographic applications. There are two distributions to keep track of. In PAC Learning (P.IV), the feasible distribution is over input examples x , while the target concept f is an arbitrary function from the class. In Heuristic Learning (P.V), there is also a feasible distribution over the concepts themselves. Cryptographic reductions often have both kinds of efficient sampling: the target concept is generated by an efficient setup or key generation experiment, while the inputs are generated by an efficient challenge or encryption experiment. In particular, attacks on public-key encryption can often be phrased as learning a polynomial-size predicate, with the concept sampled by key generation and the examples sampled by encryption. We refer to [Section 8](#) for additional discussion.

4.2 Basic Relations

We now explain the basic ordering among the five learning paradises:

$$\text{P.I} \implies \text{P.II} \implies \text{P.III} \implies \text{P.IV} \implies \text{P.V.}$$

Throughout the proof sketches below, parameters are rescaled by fixed polynomials. We suppress these routine changes in notation.

Lemma 4.1 (P.I $\implies$ P.II). *If Efficient Extrapolation is possible, then Efficient Distribution Learning is possible.*

Proof Sketch. Fix a distribution $D \in \mathcal{D}_n^{s,t}$, and let M be a randomized machine of description length at most s and running time at most t that samples D . We reduce learning D from samples to extrapolating a sequence whose coordinates are independent samples from D .

Choose a sequence length m large enough for the extrapolation guarantee, with accuracy and confidence parameters η and γ to be fixed below. Let $M^{\otimes m}$ be the randomized machine that runs M independently m times and outputs the resulting sequence $(X_1, \dots, X_m)$. Its description length is $s' = s + O(\log m)$ and its running time is $t' = O(m(t+n))$. Since m only needs to dominate a polynomial in the relevant parameters, we may choose it polynomially large so that the assumption P.I applies to $(m, n, s', t', \eta, \gamma)$.

For this product source, the conditional distribution of the next block is always the original distribution:

$$(X_i \mid X_{<i} = x_{<i}) \equiv D$$

for every i and every prefix $x_{<i}$ in the support. Given sample access to D , the distribution learner draws samples $x_1, \dots, x_m$, chooses $I \sim [m]$, and forms the prefix $y = x_{<I}$. It outputs the sampler C_y that has y hardwired and, on fresh random coins, runs the extrapolation algorithm L on input y .

The distribution of (I, y) in this procedure is exactly the distribution of $(i, X_{<i})$ in the extrapolation experiment for $M^{\otimes m}$. Therefore, by assumption P.I,

$$\Pr_{I,y} [\Delta_{TV}(C_y, D) \leq \eta] \geq 1 - \gamma.$$

Taking $\eta = \varepsilon$ and $\gamma = \delta$ gives a distribution learner for D . The returned sampler is efficient because it only simulates the polynomial-time extrapolator with a polynomial-length prefix hardwired. $\square$

Lemma 4.2 (P.II $\implies$ P.III; [Xia10]). *If Efficient Distribution Learning is possible, then Efficient Agnostic Learning is possible.*

Proof Sketch. Fix an agnostic learning instance consisting of a joint example distribution $E_n \in \mathcal{D}_{n+1}^{a,t}$. Since E_n is feasibly samplable, it belongs to the distribution class handled by P.II, with polynomially related parameters.

Simulating the distribution learner on samples obtained from $\text{EX}(E_n)$ gives, with high probability, an explicit sampler $S = (S^{(0)}, S^{(1)})$ for a distribution $\hat{E}$ satisfying $\Delta_{TV}(\hat{E}, E_n) \leq \eta$. If we could sample from the conditional distribution of $S^{(1)}$ given $S^{(0)} = x$, then we would obtain labels that are close to the true labels according to E_n . The only obstacle is that S is an explicit (i.e., white-box) sampler for pairs, not an explicit conditional sampler.

Interestingly, distribution learning also supplies the ingredient needed to remove this obstacle. By Lemma 4.8, P.II rules out AIPRFs; by Lemma 2.16, this rules out AIOWFs; and by Lemma 4.11, the nonexistence of AIOWFs allows one to produce a randomised hypothesis h such that $(x, h(x))$, with x drawn from the marginal $S^{(0)}$, is close to $\hat{E}$.

By the triangle inequality, the joint distribution $(x, h(x))$ is close to E_n . Then, as in Lemma 4.12, by sampling many choices of internal randomness for h and taking the majority hypothesis, one obtains, with high probability, a deterministic hypothesis H such that

$$\Pr_{(x,b) \sim E_n} [H(x) = b] \geq \underset{n}{\text{opt}}(\mathcal{F}_{all}, E_n) - \varepsilon' \geq \underset{n}{\text{opt}}(\mathcal{C}, E_n) - \varepsilon',$$

where $\mathcal{F}_{all}$ denotes the set of all boolean functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$, $\mathcal{C} \subseteq \mathcal{F}_{all}$ is any desired concept class, and ε' depends on the closeness of E_n and $(x, h(x))$. Choosing η and all failure probabilities as small enough polynomial fractions of ε and δ yields the desired agnostic learning guarantee. More details about this approach can be found in [Section 4.5](#). □

Lemma 4.3 (P.III $\implies$ P.IV). *If Efficient Agnostic Learning is possible, then Efficient PAC Learning is possible.*

Proof. This implication is immediate from the definitions. More specifically, observe that PAC learning a target function $f \in \text{SIZE}[s]$ over a distribution $D_n \in \mathcal{D}_n^{a,t}$ is equivalent to agnostically learning $\text{SIZE}[s]$ over the distribution $E_n \in \mathcal{D}_{n+1}^{\text{poly}(a,s), \text{poly}(t,s)}$ that samples $x \sim D_n$ and outputs $(x, f(x))$. Since

$$\text{opt}_n(\text{SIZE}[s], E) = 1,$$

the definition of an agnostic learner A guarantees that, with probability at least $1 - \delta$, $A(1^n, 1^{\varepsilon^{-1}}, \delta^{-1})$ with oracle access to $\text{EX}(f, D_n) = \text{EX}(E_n)$ outputs h such that $\Pr_{x \sim D_n}[h(x) = f(x)] \geq 1 - \varepsilon$. This satisfies the requirement of a PAC learner. □

Lemma 4.4 (P.IV $\implies$ P.V). *If Efficient PAC Learning is possible, then Efficient Heuristic Learning is possible.*

Proof Sketch. Note that, under our definitions, the PAC learning algorithm (P.IV) is allowed to make membership queries, while the heuristic learner (P.V) is not. This means in particular that we cannot simply simulate the PAC learner when designing the heuristic learner from random examples.

The first step is to show that P.IV rules out auxiliary-input pseudorandom functions. Given a candidate AIPRF F_z , fix an auxiliary string z and a random key u , and view $g(x) = F_z(u, x)$ as a polynomial size circuit. The PAC learner can be run under the uniform distribution, with both its random examples and its membership queries answered by the oracle being tested. If the oracle is $F_z(u, \cdot)$, the learner outputs a hypothesis with high agreement with the oracle on a fresh uniform input. If the oracle is a truly random function, then after only polynomially many queries its value on a fresh input is independent of the learner's hypothesis except with negligible collision probability. Testing agreement on fresh inputs therefore distinguishes the AIPRF from a random function, contradicting security. (We refer the reader to [Lemma 4.6](#) for a more detailed argument.)

In turn, by [Lemma 2.16](#), the failure of AIPRFs gives the failure of AIOWFs, and in particular rules out infinitely-often one-way functions and pseudorandom generators. We can then apply the framework of Blum, Furst, Kearns, and Lipton [[BFKL93](#)], discussed in [Section 5.1](#). Their argument starts from the fact that every candidate generator can be distinguished at every large enough input length, applies a hybrid argument to obtain a next-bit predictor for the direct-product generator associated with a random target function, and interprets that predictor as a learner from random examples. We refer to [Section 5.1](#) for more details.

Consequently, for every feasible distribution P over polynomial-size target circuits and every feasible example distribution D , there is an efficient learner that predicts $f(y)$ with advantage $1/\text{poly}(n)$, where $f \sim P$, the learner receives examples from $\text{EX}(f, D)$, and $y \sim D$. This is exactly weak heuristic learning as in P.V. □

4.3 Impagliazzo’s Worlds and Meta-Complexity

In an influential paper, Impagliazzo [Imp95] introduced five potential computational worlds connected to the complexity of NP problems and the possibility of secure cryptography. The point was not only to ask whether $P = NP$, but to distinguish several possible realities according to whether hard problems remain hard on typical inputs and whether such hardness can support secure cryptography. The two relevant worlds in the context of our discussion are Heuristica and Pessiland.

In Heuristica, NP is hard in the worst case, but this hardness disappears on average: every NP problem has an efficient algorithm that is correct on almost all inputs. In the notation of the diagram in Figure 5, this corresponds to a world where $P \neq NP$ but $\text{DistNP} \subseteq \text{AvgP}$.

In Pessiland, hardness exists on average, but the hard problems are not structured enough to yield secure cryptography. The relevant feature of Pessiland is the failure of one-way functions, equivalently the existence of efficient inverters for every candidate one-way function.

We refer to [BT06] for a detailed discussion of Heuristica, Pessiland, and their connections to cryptography and average-case complexity theory.

The learning paradises from Section 4.1 sit between these two worlds, as summarized in Figure 5. The left side of the diagram starts from the average-case easiness of NP promised by Heuristica and passes to meta-complexity via “ $\text{Gap-MINKT} \in P$ ”, i.e., the worst-case easiness of estimating time-bounded Kolmogorov complexity.⁸ This leads to extrapolation and then, by the reductions in Section 4.1, to progressively weaker learning paradises. The right side records that the weakest of these paradises is already tied to the nonexistence of one-way functions and to average-case algorithms for estimating time-bounded Kolmogorov complexity. Thus the feasibility of different learning tasks is closely tied to meta-complexity, i.e., the complexity of estimating time-bounded Kolmogorov complexity.

Another consequence is that secure cryptography excludes all five learning paradises. Indeed, P.V rules out one-way functions by Lemma 4.6. Consequently, in a cryptographic world, unrestricted Boolean circuits cannot be provably learned efficiently even with the weak guarantees of P.V: any such theorem would also refute standard cryptographic assumptions.⁹

A precise characterization of the learning power associated with both Heuristica and Pessiland is possible, although in a different learning setting [HN26b]. It would be very interesting to obtain an equally precise understanding of how the learning paradises P.I through P.V relate to standard complexity and cryptographic statements.

While the preceding discussion has focused on the relationship between learning and compression, and on its connections to cryptography and average-case complexity, the broader role of compression problems in complexity theory remains far from fully understood. A central open question is whether efficient algorithms for estimating time-bounded Kolmogorov complexity would yield efficient algorithms for NP-complete problems. This is open even in an average-case sense: we do not know whether $\text{Gap-MINKT} \in P$ implies that NP is easy on average. Informally, such a result would say that the ability to perform compression, or to recognize strings admitting short efficient descriptions, is powerful enough to solve broad classes of computational search and optimization tasks. From the perspective of learning, this would elevate the role of compression beyond efficient prediction: it would show in a certain sense that compression also underlies

⁸Note that this is a nontrivial result: from the assumption that every problem in NP is easy on average, we derive that a specific NP problem from meta-complexity is easy in the worst case. This worst-case easiness seems particularly useful when applying the assumption to construct learning algorithms that must work on every input concept.

⁹This is not in apparent contradiction to the astonishing success of machine learning algorithms in practice. Recall that the learning guarantees here concern unrestricted Boolean circuits. Learning algorithms deployed in practice are not known to satisfy such guarantees, and they are not known to break standard cryptographic assumptions.

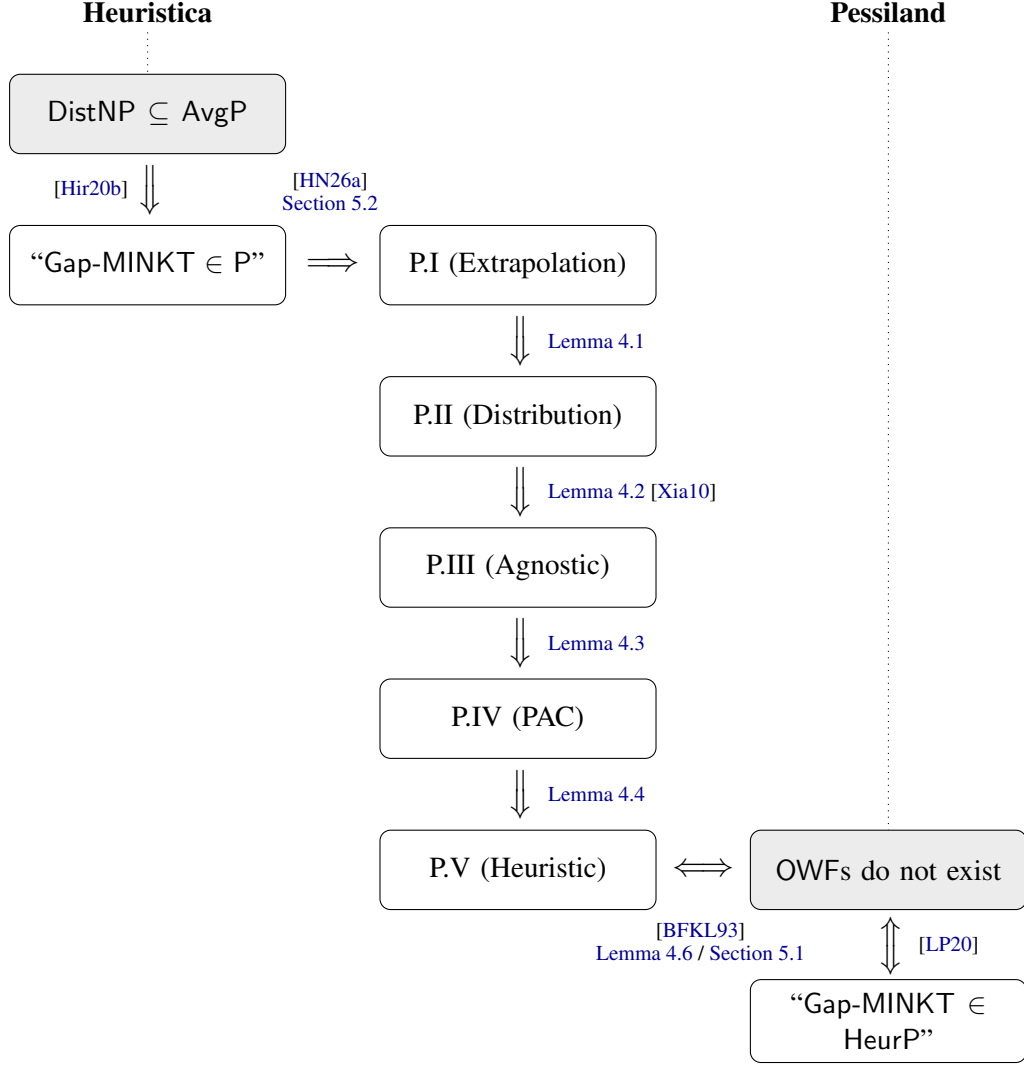


Figure 5: Learning paradises P.I – P.V (Section 4.1) between Impagliazzo’s worlds [Imp95] Heuristica and Pessiland, including their relationship to compression and meta-complexity via the computational task of estimating time-bounded Kolmogorov complexity.

efficient planning and decision-making. For recent papers on the computational complexity of compression, see [Hir22b, HLO26b] and references therein.

Problem 4.5 (Computational Complexity of Compression). What is the hardness of the task of computing/estimating the time-bounded Kolmogorov complexity of an input string? Formally, the following central problems from meta-complexity remain open:

- Does $\text{Gap-MINKT} \in \text{P}$ imply that $\text{DistNP} \subseteq \text{AvgP}$? In other words, does the easiness of estimating complexity imply that NP is easy on average?
- Does $\text{MINKT} \in \text{P}$ imply that $\text{NP} \subseteq \text{P}$? In other words, does the easiness of (exactly) computing complexity imply that NP is easy in the worst case?

4.4 Auxiliary Results

We collect here some lemmas needed in results mentioned in the preceding sections.

Lemma 4.6. *If Efficient Heuristic Learning is possible, then ioPRFs do not exist. It follows that (infinitely-often) one-way functions and pseudorandom generators do not exist.*

Proof. Let $f = \{f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}\}_{n \in \mathbb{N}}$ be an ioPRF, and let $n \in \mathbb{N}$ be a sufficiently large input length at which it is secure. Define a distribution P_n over target functions as follows: sample $u \sim \{0, 1\}^n$ uniformly at random and output a description of the circuit $f_u : \{0, 1\}^n \rightarrow \{0, 1\}$ defined by $f_u(x) = f(u, x)$ (with u hardcoded). Since u is sampled uniformly and f is polynomial-time computable, $P_n \in \text{PSAMP}$. Let $D_n = \mathcal{U}_n \in \text{PSAMP}$.

By assumption, Boolean circuits are heuristically learnable over PSAMP. Hence there exists an efficient machine A and a polynomial q' such that for all sufficiently large n ,

$$\Pr_{\substack{u \sim \mathcal{U}_n; \\ \text{EX}(f_u, \mathcal{U}_n); \\ x' \sim \mathcal{U}_n}} \left[A^{\text{EX}(f_u, \mathcal{U}_n)}(x') = f_u(x') \right] \geq \frac{1}{2} + \frac{1}{q'(n)}. \quad (2)$$

We now construct an efficient distinguisher D for f as follows.

Given oracle access to some function $\mathcal{O}$, which is either $f(u, \cdot)$ for a random $u \sim \{0, 1\}^n$ or a truly random function $R_n : \{0, 1\}^n \rightarrow \{0, 1\}$, D draws $x' \sim \mathcal{U}_n$, queries $\mathcal{O}$ to obtain b , and outputs 1 if and only if $A^{\text{EX}(\mathcal{O}, \mathcal{U}_n)}(x') = b$, where the oracle $\text{EX}(\mathcal{O}, \mathcal{U}_n)$ may be simulated by sampling $x \sim \mathcal{U}_n$ and querying $\mathcal{O}(x)$ to get the label.

When the oracle $\mathcal{O}$ is $f(u, \cdot)$ for $u \sim \{0, 1\}^n$, the learner A is run in a setting exactly matching the setting of Equation (2), so D outputs 1 with probability at least $(1/2) + (1/q'(n))$.

When the oracle $\mathcal{O}$ is R_n , $b = R_n(x')$ is uniformly random and independent of the output of A on any input not previously inspected by A . Since A inspects at most polynomially many inputs, D outputs 1 with probability exponentially close to $1/2$.

Therefore,

$$\left| \Pr_{u \sim \{0, 1\}^n} [D^{f(u, \cdot)}(1^n) = 1] - \Pr_{R_n} [D^{R_n(\cdot)}(1^n) = 1] \right| = \Omega\left(\frac{1}{q'(n)}\right),$$

contradicting the assumption that f is an ioPRF. Thus, ioPRFs do not exist. This implies that standard PRFs likewise do not exist. The non-existence of both infinitely-often and standard OWFs and PRGs follows by well-known reductions among these primitives [HILL99, GGM86]. $\square$

Using a very similar argument, we can rule out *auxiliary input* PRFs assuming easiness of PAC learning, even when membership queries are allowed.

Lemma 4.7. *If Efficient PAC Learning is possible, then AIPRFs do not exist.*

Proof Sketch. The proof is very similar to that of Lemma 4.6. Let $f_z : \{0, 1\}^{n(|z|)} \times \{0, 1\}^{n(|z|)} \rightarrow \{0, 1\}$ be a purported AIPRF. The key idea is that, for every choice of $z \in \{0, 1\}^*$ and $u \in \{0, 1\}^{n(|z|)}$, $f_z(u, -) \in \text{SIZE}[p]$ for some fixed polynomial p . We are guaranteed that the learner A will succeed (with high probability) on every such target function. Another difference is that A is allowed to make membership queries to the target function, which the distinguisher D may simulate by querying $\mathcal{O}$. $\square$

Finally, the same consequence holds assuming easiness of distributional learning. This lemma is a key ingredient in the proof that distribution learning is at least as strong as agnostic learning.

Lemma 4.8 ([KMR⁺94, Xia10]). *If Efficient Distribution Learning is possible, then AIPRFs do not exist.*

Proof Sketch. Let $f = \{f_z\}$ be a purported AIPRF, and let A be the distribution learner promised by P.II. Fix $z \in \{0, 1\}^*$, write $n = n(|z|)$, and for an oracle $\mathcal{O}: \{0, 1\}^n \rightarrow \{0, 1\}$ define the graph distribution

$$G_{\mathcal{O}} \equiv (X, \mathcal{O}(X)) \quad \text{where } X \sim \mathcal{U}_n.$$

If $\mathcal{O} = f_z(u, \cdot)$, then $G_{\mathcal{O}}$ is sampled by the efficient sampler that has (z, u) hardwired, chooses $X \sim \mathcal{U}_n$, and outputs $(X, f_z(u, X))$. Thus, for every z and u , P.II applies to $G_{f_z(u, \cdot)}$ with polynomially related size and time parameters.

We construct an oracle distinguisher D . Given z and oracle access to $\mathcal{O}$, it simulates sample access to $G_{\mathcal{O}}$ by drawing $X \sim \mathcal{U}_n$ and querying $\mathcal{O}(X)$. It runs A with a small constant accuracy parameter η and confidence parameter η , obtaining a sampler C over pairs $(x, b) \in \{0, 1\}^n \times \{0, 1\}$. It then validates C as follows. Letting Q be the set of inputs $x \in \{0, 1\}^n$ queried to $\mathcal{O}$ during the training phase, take one fresh sample $(x', b') \sim C$, and accept if and only if $x' \notin Q$ and $b' = \mathcal{O}(x')$.

On the pseudorandom side, A is being run on samples from a distribution in the class handled by P.II. Hence, with probability at least $1 - \eta$, the output sampler C satisfies $\Delta_{TV}(C(\mathcal{U}), G_{f_z(u, \cdot)}) \leq \eta$. Total variation closeness implies that C is supported almost entirely on the graph of $f_z(u, \cdot)$ (so $b' = \mathcal{O}(x')$ with high probability) and that its first marginal is close to $\mathcal{U}_n$ (so $x' \notin Q$ with high probability). Therefore the validation test accepts with probability $1 - O(\eta)$.

On the random side, let $\mathcal{O} = R$ be a uniformly random function. After conditioning on the polynomially many graph points revealed while simulating the samples for A , the values of R at fresh points are still independent fair bits. The “ $x' \notin Q$ ” test rules out the only simple way for the sampler C to pass the graph consistency test, namely putting noticeable mass on the already revealed inputs. Since C depends only on the coins of A and $R|_Q$, $R(x')$ is uniform and independent of C for $x' \notin Q$. Consequently, the output of any polynomial-time learner passes the “ $b' = \mathcal{O}(x')$ ” test with probability bounded away from the pseudorandom case, after fixing η sufficiently small.

Thus D distinguishes $f_z(u, \cdot)$ for $u \sim \mathcal{U}_n$ from a truly random function with inverse polynomial advantage for all sufficiently large relevant input lengths. This contradicts the auxiliary input pseudorandomness condition. Therefore AIPRFs do not exist. $\square$

4.5 White-Box Learning and One-Way Functions

In this section, we explore connections between *white-box learning*, in which a labelled example distribution is given explicitly as a sampling circuit, and the existence of (auxiliary-input) one-way functions. These connections are useful as tools to establish the ordering among the learning paradises of Section 4.1, particularly that P.II (Efficient Distribution Learning) implies P.III (Efficient Agnostic Learning), and may also be of interest in their own right.

4.5.1 White-Box Learning Frameworks

The model of white-box learning we consider is the same as Agnostic Learning (P.III), except that the learner is additionally provided an explicit description of a sampler for the example distribution in question. For this reason, we restrict ourselves here to distribution families $E = \{E_n\}_{n \in \mathbb{N}}$ samplable by polynomial-size circuits.

More specifically, consider a joint labelled example distribution $E_n \in \text{PSAMP/poly}$ supported over $\{0, 1\}^n \times \{0, 1\}$, where for $(x, b) \sim E_n$, $x \in \{0, 1\}^n$ is called the “example string” and $b \in \{0, 1\}$ is called the “label”. We let $E_n^{(0)}$ denote the marginal distribution over example strings and $E_n^{(1)}$ that over the labels.

As mentioned above, the learner is provided a sampling circuit S_n for E_n . Of course, this could potentially make the task of learning much easier. However, note that the learning task is still non-trivial: even though we can easily sample input/output pairs for the function in question, it may still be hard to compute the function on a desired input.

As in the setting of agnostic learning, we consider optimality with respect to a fixed concept class, $\mathcal{C}$. That is, define

$$\text{opt}_n(\mathcal{C}, E) \triangleq \max_{c \in \mathcal{C}_n} \left\{ \Pr_{(x,b) \sim E_n} [c(x) = b] \right\}.$$

Since a white-box learner has the sampler S_n available, it can simulate this circuit on random inputs to obtain labelled examples distributed according to E_n . For this reason, it does not require oracle access to $\text{EX}(E_n)$.

White-box learning is defined as follows.

Definition 4.9 (White-Box Learning [ABX08, Xia09]). Let $\mathcal{E} \subseteq \text{PSAMP/poly}$ be a class of joint distributions. An algorithm A *white-box learns* $\mathcal{C}$ over $\mathcal{E}$ if, for every $E = \{E_n\}_{n \in \mathbb{N}} \in \mathcal{E}$ and sequence $S = \{S_n\}_{n \in \mathbb{N}}$ with S_n being a sampling circuit for E_n , for all sufficiently large $n \in \mathbb{N}$, for all $\delta, \varepsilon \in (0, 1)$,

$$\Pr_A \left[A(1^n, 1^{\varepsilon^{-1}}, \delta^{-1}, S_n) \text{ outputs } h \text{ such that } \Pr_{(x,b) \sim E_n} [h(x) = b] \geq \text{opt}_n(\mathcal{C}, E) - \varepsilon \right] \geq 1 - \delta.$$

We will primarily be interested in White-Box Learning for the following choices of concept class $\mathcal{C}$: $\mathcal{C} = \text{SIZE}[s]$ for a fixed function $s : \mathbb{N} \rightarrow \mathbb{N}$, and $\mathcal{C} = \mathcal{F}_{\text{all}}$, where $(\mathcal{F}_{\text{all}})_n$ denotes the set of all boolean functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$. In particular, in the latter case, the learner aims to approximate the Bayes-optimal predictor for $\mathcal{E}$.

4.5.2 White-Box Learning and Breaking Auxiliary-Input One-Way Functions

In this section, we establish the following:

$$\neg \exists \text{AIOWF} \iff \text{White-Box Learning } \mathcal{F}_{\text{all}}.$$

The implication from $\neg \exists \text{AIOWF}$ to the easiness of White-Box Learning $\mathcal{F}_{\text{all}}$ was proved in [Xia09] and implicit in [ABX08].

Also note that, since White-Box Learning $\mathcal{F}_{\text{all}}$ implies White-Box Learning $\mathcal{C}$ for any desired concept class $\mathcal{C} \subseteq \mathcal{F}_{\text{all}}$, the above implies that

$$\neg \exists \text{AIOWF} \implies \text{White-Box Learning SIZE}[s]$$

for any $s : \mathbb{N} \rightarrow \mathbb{N}$. The latter may be used in a proof that P.II (Efficient Distribution Learning) $\implies$ P.III (Efficient Agnostic Learning), as in Lemma 4.2.

[Xia09] leaves as an open question whether White-Box Learning $\text{SIZE}[s]$, for a fixed polynomial s , implies that AIOWFs can be broken. Note that a positive answer would show the equivalence of White-Box Learning with respect to $\text{SIZE}[s]$ and $\mathcal{F}_{\text{all}}$ over distributions samplable in PSAMP/poly .

Problem 4.10 (White-Box Learning and AIOWF). Does White-Box Learning $\text{SIZE}[s]$ imply the non-existence of AIOWF? Alternatively, is there evidence to the contrary?

We begin with the following auxiliary lemma.

Lemma 4.11. *Assume $\neg\exists$ AIOWF. Let E be a family of labelled example distributions with sampling circuits S , as in [Definition 4.9](#). There exists a polynomial-time algorithm that, for all sufficiently large $n \in \mathbb{N}$, on input $(1^n, 1^{(\varepsilon')^{-1}}, S_n)$, outputs a randomized hypothesis h such that $\Delta((x, h(x)), E_n) \leq \varepsilon'(n)$, where $x \sim E_n^{(0)}$.*

Proof. Let $\{E_n = (E_n^{(0)}, E_n^{(1)})\}_{n \in \mathbb{N}}$, with each E_n supported over $\{0, 1\}^n \times \{0, 1\}^{m(n)}$, be a labelled example distribution samplable by poly-size circuits $\{S_n = (S_n^{(0)}, S_n^{(1)})\}_{n \in \mathbb{N}}$, each taking randomness of length $\rho(n) = \text{poly}(n)$ as input. Define a poly-time computable auxiliary input function f as follows.

$$f_{S_n^{(0)}}(r) = S_n^{(0)}(r).$$

That is, f has the sampler $S_n^{(0)}$ hard-coded as an auxiliary input. Its input is a choice of randomness r , and its output is the output of $S_n^{(0)}$ run on r .

Assuming $\neg\exists$ AIOWF, by [Lemma 2.16](#), $\neg\exists$ distAIOWF. For any desired inverse polynomial ε' , let I be a PPT ε' -distributional inverter for f . That is, for any sufficiently large $n \in \mathbb{N}$,

$$\Delta\left((f_{S_n^{(0)}}(r), I(S_n^{(0)}, f_{S_n^{(0)}}(r))), (f_{S_n^{(0)}}(r), r)\right) \leq \varepsilon'(n), \quad (3)$$

where $r \sim \mathcal{U}_{\rho(n)}$.

The desired algorithm will output the following randomized hypothesis h :

On input $x \in \{0, 1\}^n$, h invokes I on x with auxiliary input $S_n^{(0)}$ to produce a pre-image r ; then, h outputs $S_n^{(1)}(r)$.

Observe that for $r \sim \mathcal{U}_{\rho(n)}$ and $x = S_n^{(0)}(r)$ (that is, $x \sim E_n^{(0)}$),

$$\begin{aligned} \Delta((x, h(x)), E_n) &= \Delta((x, S_n^{(1)}(I(S_n^{(0)}, x))), (x, S_n^{(1)}(r))) && \text{(definition of } h) \\ &\leq \Delta((x, I(S_n^{(0)}, x)), (x, r)) \\ &= \Delta((f_{S_n^{(0)}}(r), I(S_n^{(0)}, f_{S_n^{(0)}}(r))), (f_{S_n^{(0)}}(r), r)) && \text{(definition of } f) \\ &\leq \varepsilon'(n). && \text{(Eq. (3))} \end{aligned}$$

This completes the proof. □

The next two lemmas establish an equivalence between $\neg\exists$ AIOWF and the easiness of White-Box Learning $\mathcal{F}_{all}$.

Lemma 4.12 ([\[ABX08, Xia09\]](#)). *If $\neg\exists$ AIOWF, then White-Box Learning $\mathcal{F}_{all}$ over PSAMP/poly is easy.*

Proof. The desired white-box learner, given a sampler S_n for E_n and parameters ε and δ , will construct h from S_n as in [Lemma 4.11](#) with parameter $\varepsilon' = \varepsilon^4$. It will then sample $m \triangleq 8\varepsilon^{-4} \cdot n \cdot \log(1/\delta)$ choices of randomness $r_1, \dots, r_m$ for h , and output the deterministic hypothesis H that, on input x , outputs $\text{maj}_{i \in [m]} \{h(x; r_i)\}$.

We now argue for the correctness of this white-box learner. Recall that h is such that $\Delta((x, h(x)), E_n) \leq \varepsilon'$, where $x \sim E_n^{(0)}$. Let $f_{opt} : \{0, 1\}^n \rightarrow \{0, 1\}$ be the function that maximises $\Pr_{(x,b) \sim E_n}[f_{opt}(x) = b]$, and, for $x \in \{0, 1\}^n$, let B_x denote the conditional distribution of $E_n^{(1)}$ given $E_n^{(0)} = x$. Define

$$\alpha(x) = \max_{b \in \{0,1\}} \{\Pr[B_x = b]\}.$$

Note that $f_{opt}(x) = \arg \max_{b \in \{0,1\}} \{\Pr[B_x = b]\}$, so $\Pr[f_{opt}(x) = B_x] = \alpha(x)$.

By Markov's inequality, the set $\mathcal{X}_{bad} = \{x : \Delta(h(x), B_x) > \varepsilon^2\}$ satisfies $\Pr_{x \sim E_n^{(0)}}[x \in \mathcal{X}_{bad}] \leq \varepsilon^2$. For all $x \notin \mathcal{X}_{bad}$, we have $|\Pr[h(x) = b] - \Pr[B_x = b]| \leq \varepsilon^2$ for every $b \in \{0, 1\}$.

For $x \notin \mathcal{X}_{bad}$, we consider two cases.

Case I: $\alpha(x) > 1/2 + 2\varepsilon^2$. Note that

$$\begin{aligned} \Pr_{r_i}[h(x; r_i) = f_{opt}(x)] &\geq \alpha(x) - \varepsilon^2 \\ &> \frac{1}{2} + \varepsilon^2. \end{aligned}$$

Then, by a Chernoff bound,

$$\Pr_{r_1, \dots, r_m}[H(x) = f_{opt}(x)] \geq 1 - \delta \cdot 2^{-n}.$$

◇

Case II: $\alpha(x) \leq 1/2 + 2\varepsilon^2$. Note that $\alpha(x) \geq 1/2$ also holds in this case, by definition of $\alpha(x)$. Consider any fixed H . If $H(x) = f_{opt}(x)$, then $\Pr[H(x) = B_x] \geq 1/2$. On the other hand, if $H(x) = 1 - f_{opt}(x)$, then $\Pr[H(x) = B_x] \geq 1 - \Pr[f_{opt}(x) = B_x] \geq 1/2 - 2\varepsilon^2$. It follows that

$$\Pr[f_{opt} = B_x] - \Pr[H(x) = B_x] \leq \frac{1}{2} + 2\varepsilon^2 - \left(\frac{1}{2} - 2\varepsilon^2\right) = 4\varepsilon^2.$$

◇

By a union bound, with probability $1 - \delta$ over the choice of $r_1, \dots, r_m$, all x satisfying Case I satisfy $H(x) = f_{opt}(x)$. Then, with probability $1 - \delta$, for all x satisfying either case,

$$\Pr[f_{opt} = B_x] - \Pr[H(x) = B_x] \leq 4\varepsilon^2.$$

Overall, with probability at least $1 - \delta$ over the randomness of the white-box learner, it outputs a hypothesis H such that

$$\Pr_{(x,b) \sim E_n}[H(x) = b] \geq \text{opt}_n(\mathcal{F}_{all}, E) - 4\varepsilon^2 - \varepsilon^2 \geq \text{opt}_n(\mathcal{F}_{all}, E) - \varepsilon,$$

assuming $\varepsilon < 1/5$ without loss of generality.

This completes the proof. □

Lemma 4.13. *If White-Box Learning $\mathcal{F}_{all}$ over PSAMP/poly is easy, then $\neg \exists AIOWF$.*

Proof. Toward a contradiction, suppose there exists AIOWF. By Lemma 2.16, there likewise exists an AIPRG, $G_z : \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$. By an exhaustive search running in time exponential in n , it is possible to distinguish outputs of G_z from truly random strings. Let D be this inefficient distinguisher. In particular, for any given z ,

$$\left| \Pr_{v \sim \mathcal{U}_n} [D(z, G_z(v)) = 1] - \Pr_{u \sim \mathcal{U}_{2n}} [D(z, u) = 1] \right| \geq 1 - o(1).$$

By Yao's hybrid argument (see Section 5.1.1), there exists an inefficient next-bit predictor P^D such that, for any z ,

$$\Pr_{P^D; i \sim [2n]; v \sim \mathcal{U}_n} [P^D(z, G_z(v)_{<i}) = G_z(v)_i] > \frac{1}{2} + \frac{1}{8n},$$

where y_j denotes the j^{th} bit of a string y , and $y_{<j}$ denotes the prefix of y up to and including position $j - 1$.

Now, consider the distribution E_z , which samples a random index $i \sim [2n]$ and a random seed $v \sim \mathcal{U}_n$, and then outputs

$$(z, i, G_z(v)_{<i}, 0^{2n-i+1}; G_z(v)_i).$$

It is easy to check that E_z is samplable by a polynomial-size circuit, which can be constructed in polynomial time given z . Moreover, by the existence of P^D , we have that

$$\text{opt}_N(\mathcal{F}_{\text{all}}, E_z) \geq \frac{1}{2} + \frac{1}{8n},$$

where $N = |z| + 2n + O(\log n)$ is the length of an example string from E_z . Going forward, we omit the trailing 0s used for padding.

We claim that a white-box learner for E_z may be used to break G_z . Let A' be a white-box learner with respect to $\mathcal{F}_{\text{all}}$. Consider the following adversary A :

On input (z, y) , construct a sampler S_z for E_z , and simulate $A'(S_z)$ with accuracy parameter $\varepsilon = n^{-2}$ and confidence $\delta = 2^{-n}$ to obtain a hypothesis h . Sample a random $j \sim [2n]$, and accept iff $h(z, j, y_{<j}) = y_j$.

First consider the case that $y \sim G_z(\mathcal{U}_n)$. In this case,

$$\begin{aligned} \Pr_{A, y} [A(z, y) = 1] &= \Pr_{A', y, j} [h(z, y_{<j}) = y_j] \\ &= \Pr_{A', (x, b) \sim E_z} [h(x) = b] \\ &\geq \frac{1}{2} + \frac{1}{8n} - \frac{1}{n^2} - \frac{1}{2^n} \\ &> \frac{1}{2} + \frac{1}{10n}. \end{aligned}$$

Next, in the case that $y \sim \mathcal{U}_{2n}$, y_j is a uniformly random bit independent of $(z, y_{<j})$, so

$$\Pr_{A, y} [A(z, y) = 1] = \frac{1}{2}.$$

We conclude that A breaks G_z , as desired. □

5 Learning via Meta-Complexity and Algorithmic Information Theory

5.1 Detecting Structure: Supervised Learning via Distinguishers

Valiant introduced his framework of *PAC learning* in 1984 [Val84]; in the same paper, he already recognized a tension between the possibility of efficient learning and the possibility of cryptography. He points out in particular that if PAC learning is easy, then it would be easy to break a pseudorandom function. Though the converse of this specific statement remains unknown, the connection also goes in the other direction, where algorithms for “detecting structure from randomness” may be used to carry out learning tasks.

In this section, we describe a general framework for establishing relationships in the latter direction. In this framework, algorithms for *meta-complexity* (namely, deciding whether a given object is of “high” or “low” complexity) are used to break cryptographic and complexity-theoretic pseudorandom generators, yielding learning algorithms by applying reconstruction properties of those generators. We consider two different styles of generator, which we refer to as the Direct Product Generator and the Nisan-Wigderson Generator, respectively.

The Direct Product Generator takes the form

$$G_{DP}^f(\bar{x}) = (x_1, \dots, x_m; b_1, \dots, b_m),$$

where $\bar{x} = (x_1, \dots, x_m) \in (\{0, 1\}^n)^m$ are input examples, and $(b_1, \dots, b_m) \in \{0, 1\}^m$ are some labels associated with a target function $f: \{0, 1\}^n \rightarrow \{0, 1\}$.

The Nisan-Wigderson Generator takes the form

$$G_{NW}^f(z) = (f(z|_{S_1}), \dots, f(z|_{S_m})),$$

where $z \in \{0, 1\}^{O(n^2)}$ is an input specifying some examples, $f: \{0, 1\}^n \rightarrow \{0, 1\}$ is a target function, and $S_1, \dots, S_m$ are *NW-designs*. An NW-design is a collection of m subsets (where m is the desired output length) chosen from a small ground set of size ℓ . They are constructed to have bounded intersections: for instance, any two distinct subsets S_i, S_j might overlap in at most $\log m$ elements. Roughly speaking, because the subsets intersect very little, the inputs fed into a “hard” function f producing the output bits remain mostly independent.

In the results we survey below, we will see various instantiations of these generators, which may be broken under various assumptions about the easiness of meta-complexity. We mention in advance that learners obtained by breaking the Direct Product Generator can work from *random examples*, whereas learners obtained by breaking the Nisan-Wigderson Generator require *membership queries*. In cases where the corresponding assumption from meta-complexity is known to hold, the approach has led to unconditional learning algorithms.

5.1.1 From Distinguishers to Predictors

The central technique driving all the results in this section is Yao’s hybrid argument, which converts a *distinguisher* into a *next-bit predictor* [Yao82]. First, we explain what it means for an algorithm to be a distinguisher. Consider an arbitrary joint distribution (X, Y) , where X is supported over $\{0, 1\}^m$ and Y over $\{0, 1\}^\ell$. Let A be a function (which we will typically identify with an algorithm) mapping $m + \ell$ input bits to one output bit. We consider the case that A *distinguishes* (X, Y) from $(X, \mathcal{U}_\ell)$ with advantage $\varepsilon > 0$, where $\mathcal{U}_\ell$ denotes the uniform distribution over $\{0, 1\}^\ell$. Namely,

$$\left| \Pr_A[A(X, Y) = 1] - \Pr_A[A(X, \mathcal{U}_\ell) = 1] \right| \geq \varepsilon.$$

Yao showed that, from any such A , one can construct another algorithm $P_A : \{0, 1\}^m \times \{0, 1\}^{<\ell} \rightarrow \{0, 1\}$ taking an output of X together with some prefix of an output of Y and producing the next unseen bit of Y with better-than-trivial success probability. In particular,

$$\Pr_{P_A, i \sim [\ell]} [P_A(X, Y_1, \dots, Y_{i-1}) = Y_i] \geq \frac{1}{2} + \frac{\varepsilon}{2\ell}.$$

We refer to such an algorithm P_A as a *next-bit predictor*.

To prove this statement, Yao employs a hybrid argument. Roughly, one defines “hybrid” distributions

$$H_j = (X, Y_1, \dots, Y_j, \mathcal{U}_{\ell-j})$$

for $j \in \{0, \dots, \ell\}$, so $H_0 = (X, \mathcal{U}_\ell)$ and $H_\ell = (X, Y)$. Observe that

$$\Pr[A(H_0) = 1] - \Pr[A(H_\ell) = 1] = \sum_{j \in \{0, \dots, \ell-1\}} \Pr[A(H_j) = 1] - \Pr[A(H_{j+1}) = 1].$$

Supposing that $\Pr[A(X, \mathcal{U}_\ell) = 1] - \Pr[A(X, Y) = 1] \geq \varepsilon$, so the above is at least ε , it holds that the difference $\Pr[A(H_j) = 1] - \Pr[A(H_{j+1}) = 1]$ is at least ε/ℓ in expectation over the choice of j .¹⁰ This expected advantage of A at the j^{th} bit of Y can then be leveraged to obtain a next-bit predictor at the corresponding position: when given a random bit $u \sim \mathcal{U}$ at position $j+1$, A is slightly less likely to output 1 when $u = Y_{j+1}$ than when $u \neq Y_{j+1}$. We refer the reader to [Vad12] for further detail.

5.1.2 Direct Product and Random Examples

From PRG Distinguishers to Learning Random Target Functions. An early instantiation of the framework described above can be seen in the work of Blum et al. [BFKL93], where the authors show that breaking a cryptographic pseudorandom generator yields a learner working in a certain average-case setting. In more detail, from a successful distinguisher against a candidate PRG construction, one obtains a learner succeeding with high probability over randomly sampled target functions (see Definition 2.10).

The starting point is the assumption that for every candidate PRG $G : \{0, 1\}^n \rightarrow \{0, 1\}^{\ell(n)}$ there exists an efficient algorithm $A : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}$ distinguishing outputs of G from uniformly random strings. That is, for some polynomial p , for all sufficiently large n ,

$$|\Pr[A(G(\mathcal{U}_n)) = 1] - \Pr[A(\mathcal{U}_{\ell(n)}) = 1]| \geq \frac{1}{p(n)}.$$

Let $P = \{P_n\}_{n \geq 1}$ be a collection of distributions over target functions. For the sake of this exposition, we will discuss the case of weak prediction over example strings drawn from the uniform distribution.¹¹ We define a candidate PRG G as follows, where $m = m(n)$ is a sufficiently large parameter.

On input $(r_P, x_1, \dots, x_m)$, with $r_P \in \{0, 1\}^{\text{poly}(n)}$ and each $x_i \in \{0, 1\}^n$, G samples $\sigma_f \sim P_n$ using randomness r_P , where σ_f is the representation of a function f . For each $i \in [m]$, it then computes $b_i \triangleq f(x_i)$. G outputs $(x_1, \dots, x_m, b_1, \dots, b_m)$.

¹⁰The case that $\Pr[A(X, Y) = 1] - \Pr[A(X, \mathcal{U}_\ell) = 1] \geq \varepsilon$ can be handled similarly.

¹¹[BFKL93] use a different argument, inverting an appropriate one-way function, to yield strong prediction over examples drawn from other distributions in PSAMP. See Theorem 11 in [BFKL93].

The goal is then to transform a distinguisher A for G into a learner for P_n over $\mathcal{U}_n$ in the sense of Definition 2.10. By definitions of G and A , we have

$$\left| \Pr_{\substack{\sigma_f \sim P_n; \\ x_1, \dots, x_m \sim \mathcal{U}_n}} [A(x_1, \dots, x_m, f(x_1), \dots, f(x_m)) = 1] - \Pr[A(\mathcal{U}_{m \cdot n + m}) = 1] \right| \geq \frac{1}{p(n)}.$$

By Yao's hybrid argument, for some PPT machine A' built from A ,

$$\Pr_{\substack{\sigma_f \sim P_n; A'; i \sim [m]; \\ x_1, \dots, x_m \sim \mathcal{U}_n}} [A'(x_1, \dots, x_m, f(x_1), \dots, f(x_{i-1})) = f(x_i)] \geq \frac{1}{2} + \frac{1}{2m \cdot p(n)}.$$

We may then define a learner A'' as follows.

On input $((x_1, f(x_1)), \dots, (x_m, f(x_m)); x')$, sample $i \sim [m]$, and output $A'(x_1, \dots, x_{i-1}, x', x_{i+1}, \dots, x_m, f(x_1), \dots, f(x_{i-1}))$.

By the definitions of A' and A'' , we have that

$$\Pr_{\substack{\sigma_f \sim P_n; \\ (x_i, f(x_i))_{i \in [m(n)]} \sim \text{EX}(f, \mathcal{U}_n)^{m(n)}; \\ x' \sim \mathcal{U}_n}} [A''((x_1, f(x_1)), \dots, (x_{m(n)}, f(x_{m(n)})), x') = f(x')] \geq \frac{1}{2} + \frac{1}{2m \cdot p(n)}.$$

Thus, A'' satisfies the requirement of weak prediction.

From RRHS Refutation to PAC Learning over Arbitrary Distributions. Since the insight of [BFKL93], several papers continued to employ distinguishers to obtain learners. In an elegant example of such work, Vadhan [Vad17] formalises a distinguishing task to capture *worst-case* PAC learning over *arbitrary* example distributions. Specifically, for a concept class P and its dual class $P^* = Q$ (where for an evaluation function $\text{Eval} : \{0, 1\}^s \times \{0, 1\}^t \rightarrow \{0, 1\}$, $p_y(x) = q_x(y) = \text{Eval}(y, x)$), Vadhan defines the *Random Right-Hand Side (RRHS) Refutation* problem for Q , where the task is to distinguish

- $(x_1, b_1), \dots, (x_m, b_m)$, for independently random bits $b_i \sim \mathcal{U}$, from
- $(x_1, b_1), \dots, (x_m, b_m)$, where $b_i = q_{x_i}(y) = p_y(x_i)$ for a fixed concept $f = p_y \in P$.

Observe that this is precisely the task of distinguishing the output of the Direct Product Generator

$$G_{DP}^f(x_1, \dots, x_m) = (x_1, \dots, x_m, f(x_1), \dots, f(x_m))$$

from

$$(x_1, \dots, x_m, \mathcal{U}_m),$$

for arbitrary strings $x_1, \dots, x_m$.

We are interested here in the reduction from PAC learning to RRHS Refutation. Let A be an RRHS Refuter for some concept class $\mathcal{C}$: that is, for any $f \in \mathcal{C}$ and any choice of $x_1, \dots, x_m$, A distinguishes the two distributions above. As in the work of [BFKL93], Yao's hybrid argument yields the existence of a machine A' such that

$$\Pr_{r \sim \mathcal{U}_{\text{poly}(n)}; i \sim [m]} [A'(x_1, \dots, x_m, f(x_1), \dots, f(x_{i-1}); r) = f(x_i)] \geq \frac{1}{2} + \frac{1}{\text{poly}(n)}.$$

Letting D_n be an arbitrary example distribution, an averaging argument implies that for some polynomial p , with probability at least $1/p(n)$ over the choice of r, i , and $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_m \sim D_n^{m-1}$,

$$\Pr_{x_i \sim D_n} [A'(x_1, \dots, x_m, f(x_1), \dots, f(x_{i-1}); r) = f(x_i)] \geq \frac{1}{2} + \frac{1}{p(n)}.$$

Now, let B be the algorithm that takes m examples $(x_1, f(x_1)), \dots, (x_m, f(x_m))$ from its oracle $\text{EX}(f, D_n)$, chooses a random $i \sim [m]$ and randomness r for A' , and then outputs the hypothesis

$$h(x) \triangleq A'(x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_m, f(x_1), \dots, f(x_{i-1}); r).$$

By the above,

$$\Pr_{B; \text{EX}(f, D_n)} \left[B^{\text{EX}(f, D_n)} \text{ outputs } h \text{ such that } \Pr_{x \sim D_n} [h(x) = f(x)] \geq \frac{1}{2} + \frac{1}{p(n)} \right] \geq \frac{1}{p(n)}.$$

Note that the confidence parameter $\delta = 1 - p(n)^{-1}$ can be boosted by parallel repetition, using empirical estimation to determine the best hypothesis produced.

In this way, the easiness of RRHS Refutation yields a weak PAC learner. A classic result of Schapire shows that such a learner can be *boosted* to obtain a (strong) PAC learner [Sch90].

Correlative RRHS Refutation and Agnostic Learning. In work independent of [Vad17], Kothari and Livni [KL18] give a related refutation problem, denoted *correlative RRHS Refutation*, to characterise the harder task of *agnostic learning*. Recall that in the setting of agnostic learning, examples from the oracle EX are not guaranteed to correspond exactly to any fixed target function from the relevant concept class; the learner is only asked to produce a hypothesis that is “almost” as accurate as the most accurate function from the concept class [KSS94, Hau92].

One difference compared to the RRHS Refutation task of [Vad17] is that correlative RRHS Refutation is defined in a *distribution-specific* setting. Then, the task of correlative RRHS Refutation over some particular example distribution characterises that of agnostic learning over the *same* distribution, thanks to the existence of distribution-specific boosting algorithms for agnostic learners [KK09, Fel10]. This is unlike the setting of [Vad17], which requires an RRHS refuter working on *every* distribution to obtain a learner working on every distribution.

As in the works discussed above, a weak learner in [KL18] arises from a hybrid argument. More specifically, suppose we have a *correlative RRHS refuter* that is capable of detecting samples with correlation δ with circuits in $\text{SIZE}[s]$. Namely, let $\mathcal{E}$ be a joint example distribution satisfying

$$\max_{c \in \text{SIZE}[s(n)]} \left\{ \Pr_{(x,b) \sim \mathcal{E}_n} [c(x) = b] \right\} \geq \frac{1}{2} + \delta,$$

and suppose we have an algorithm A that $\Omega(1)$ -distinguishes

- $(x_1, b_1), \dots, (x_m, b_m)$, for independently random $(x_i, -) \sim \mathcal{E}_n$ and $b_i \sim \mathcal{U}$, from
- $(x_1, b_1), \dots, (x_m, b_m)$, where $(x_i, b_i) \sim \mathcal{E}_n$ independently.

Then, by a hybrid argument, given m random examples from $\mathcal{E}_n$, A yields a construction of a hypothesis h such that

$$\Pr_{(x,b) \sim \mathcal{E}_n} [h(x) = b] \geq \frac{1}{2} + \frac{1}{O(m)}.$$

This is sufficient to apply known boosting algorithms [KK09, Fel10] to obtain a (strong) agnostic learner over $\mathcal{D}$.

From Average Case Easiness of NP to Agnostic Learning. Hirahara and Nanashima [HN21] build on [KL18] to prove that efficient agnostic learning of P/poly over distributions in PSAMP/poly is possible under the assumption that $\text{DistNP} \subseteq \text{AvgP}$. The paper also builds on an earlier line of work due to Hirahara, which proved that several promise problems in meta-complexity exhibit *worst-case to average-case reductions* (e.g. [Hir18, Hir20a]). In particular, if an appropriate ‘gap’ version of the time-bounded Kolmogorov complexity problem, denoted Gap-MINKT, can be solved in deterministic poly-time on average over the uniform distribution, then it is likewise easy in the worst case.

The key contribution of [HN21] consists in reducing correlative RRHS Refutation to Gap-MINKT. The intuition is that samples drawn according to the ‘random’ case are of high complexity with high probability, whereas the K^t complexity of samples from the ‘correlative’ case can be bounded. For simplicity, consider the setting of realizable PAC learning, so the examples in the latter case are actually consistent with some function in $\text{SIZE}[s]$ for a polynomial s .

In the ‘random’ case, by Symmetry of Information for Kolmogorov complexity and a counting argument, one can show that for any time-bound t , with high probability over $x_i \sim \mathcal{D}_n$ and $b_i \sim \mathcal{U}$,

$$K^t(x_1, \dots, x_m, b_1, \dots, b_m) \gtrsim K(x_1, \dots, x_m) + m.$$

On the other hand, if the labels b_i are equal to $f(x_i)$ for some function $f \in \text{SIZE}[s]$, for a sufficiently large polynomial t' , we have

$$\begin{aligned} K^{\text{poly}(t')}(x_1, \dots, x_m, b_1, \dots, b_m) &\leq K^{t'}(x_1, \dots, x_m) + O(s(n)) + O(\log n) \\ &\ll K^{t'}(x_1, \dots, x_m) + m, \end{aligned}$$

provided that m is sufficiently larger than s . This follows by encoding a description of an s -size circuit for f , and computing each b_i by running this circuit on input x_i . The idea is then to run an algorithm for Gap-MINKT on the given samples to distinguish “high” complexity from “low” complexity.

At a technical level, several difficulties arise from the differing (finite and infinite) time-bounds involved, and from the fact that one cannot exactly compute K^t complexity for a desired $t \in \mathbb{N}$ under the assumption $\text{DistNP} \subseteq \text{AvgP}$. In fact, the definition of Gap-MINKT is such that one only gets a poly-time algorithm A such that $A(1^t, x)$ outputs $k \in \mathbb{N}$ such that

$$K^{p(t)}(x) - \log p(t) \leq k \leq K^t(x)$$

for some polynomial p . Roughly speaking, the authors overcome this by proving that, for a sufficiently large time-bound t (depending on the running time to sample from a given distribution $\mathcal{D}_n \in \text{PSAMP}/a(n)$),

$$K^t(\bar{x}) \approx K(\bar{x}) \pm a(n)$$

with high probability over strings $\bar{x} = (x_1, \dots, x_m)$ sampled from $(\mathcal{D}_n)^m$. Thus, the differences between complexity measures with different time-bounds can be made immaterial by taking the number of samples m sufficiently larger than $a(n)$.

The approximate equality above follows from a *coding theorem* for time-bounded Kolmogorov complexity, which holds under the assumption $\text{DistNP} \subseteq \text{AvgP}$. In particular, for any distribution $\mathcal{D}_n \in \text{PSAMP}/a(n)$ and any string x in the support of $\mathcal{D}_n$, for sufficiently large $t \in \mathbb{N}$ (depending on the exact complexity of sampling $\mathcal{D}$),

$$K^t(x) \leq \log(1/\mathcal{D}_n(x)) + a(n) + O(\log n).$$

Since $\mathcal{D}_n \in \text{PSAMP}/a(n)$ implies $(\mathcal{D}_n)^m \in \text{PSAMP}/a(n)$ when $m = \text{poly}(n)$, one obtains a similar statement for $\bar{x} \in \text{Support}((\mathcal{D}_n)^m)$. Given this statement, it holds that

$$\begin{aligned} \mathbb{E}_{\bar{x} \sim \mathcal{D}_n^m} [K^t(\bar{x}) - K(\bar{x})] &\leq \mathbb{E}_{\bar{x} \sim \mathcal{D}_n^m} [K^t(\bar{x})] - H(\mathcal{D}_n^m) + O(\log n) \\ &\leq \mathbb{E}_{\bar{x} \sim \mathcal{D}_n^m} [\log(1/\mathcal{D}_n^m(\bar{x}))] + a(n) + O(\log n) - H(\mathcal{D}_n^m) \\ &\leq a(n) + O(\log n), \end{aligned}$$

yielding the desired closeness between time-bounded and -unbounded Kolmogorov complexity in expectation over $\mathcal{D}_n^m$.

A subsequent work of Goldberg, Kabanets, Lu, and Oliveira [GKLO22] extends the result from [HN21], obtaining the same conclusion of efficient agnostic learning from the weaker assumption that $\text{DistNP} \subseteq \text{AvgBPP}$: that is, average-case easiness of NP for *randomised* algorithms. In this setting, one does not have derandomisation (which is available to [HN21] via [BFP05]), and thus does not have a worst-case to average-case reduction for Gap-MINKT. Instead, the authors obtain worst-case easiness of a notion of time-bounded *probabilistic* Kolmogorov complexity, pK^t , which generalizes K^t and rK^t . Given this, the rest of the proof proceeds similarly to [HN21].

Agnostic Learning from Average Case Kolmogorov Complexity. Goldberg and Kabanets [GK23] further weaken the prerequisite assumptions for agnostic learning, exhibiting a direct connection from the average-case easiness of time-bounded (deterministic) Kolmogorov complexity to the existence of agnostic learners. The authors consider average-case easiness of K^t complexity as well as another complexity measure denoted KT [ABK⁺06], showing that agnostic learners follow from either assumption. KT complexity is known to be polynomially related to boolean circuit size, bringing the assumption close to a ‘natural property’ in the sense of Razborov and Rudich [RR97], and thus drawing a connection to a line of work obtaining learners from natural properties, discussed further in Section 5.1.3 below. For both K^t and KT, [GK23] assume average-case easiness for randomised algorithms over the uniform distribution.

Compared to the settings of [HN21] and [GKLO22], a few additional challenges arise. One difficulty has to do with the task of showing, as in [HN21], that $K^t(\bar{x}) \approx K(\bar{x})$ with high probability over strings $\bar{x} \sim (\mathcal{D}_n)^m$. As mentioned above, in the prior works, this was proved via coding theorems for time-bounded Kolmogorov complexity. In the case of [HN21], a coding theorem holds for K^t under the stronger assumption of average-case easiness of NP. In the case of [GKLO22], due to [LO22], a coding theorem holds unconditionally for pK^t . In the setting of [GK23], though a coding theorem is not available, a similar statement can be shown to hold with high probability in the presence of an additional random string r . Namely, for sufficiently large t ,

$$\Pr_{r \sim \mathcal{U}_t} \left[K^{\text{poly}(t)}(x, r) \leq \log(1/\mathcal{D}_n(x)) + a(n) + |r| + O(\log n) \right] \geq 1 - o(1).$$

By incorporating this additional randomness r into the algorithm for correlative RRHS refutation, one may obtain an agnostic learner as desired.

Consideration of KT rather than K^t leads to additional challenges, primarily stemming from the fact that KT complexity does not admit a worst-case to average-case reduction in the regime of a small approximation gap. On the other hand, the assumed average-case easiness of KT is still sufficient to learn *over the uniform distribution*. Hence, one method to obtain a learner is to reduce the task of learning over arbitrary distributions in PSAMP/poly to the task of learning over the uniform distribution. This can be

accomplished by using an average-case algorithm for KT as an inverter for a distributional one-way function; more details about this general approach can be found in the discussion of [BCK⁺22] below. However, in the case of non-uniformly sampled target distributions, the learner resulting from this argument needs to know the distribution explicitly.¹² [GK23] show that this caveat can be removed by assuming worst-case easiness of computing KT complexity, employing a more complicated amortization argument to bound the KT complexity of many concatenated samples from $\mathcal{D}_n$ in lieu of a coding theorem (see also [AGvM⁺18]). It is left open whether the caveat can be removed assuming only average-case easiness of KT.

5.1.3 Nisan-Wigderson and Membership Queries

All the results we have covered so far have employed distinguishers for variants of the Direct Product Generator, G_{DP}^f , to construct efficient learners. We now shift attention to learners arising from distinguishers for the Nisan-Wigderson Generator, G_{NW}^f .

The NW PRG is very well known for its applications in hardness/randomness tradeoffs [NW94, BFNW93, IW97]. In that context, one instantiates the PRG with a function f known to be hard. The NW PRG then takes the form

$$G_{NW}^f(z) = (f(z|_{S_1}), f(z|_{S_2}), \dots, f(z|_{S_m})),$$

where $f: \{0,1\}^n \rightarrow \{0,1\}$, $|z| = d = O(n^2)$, $m < 2^n$, and $S_1, \dots, S_m \subseteq [d]$ are “NW designs”, with $|S_i| = n$ and $|S_i \cap S_j| \leq \log m$ for all $i \neq j \in [m]$. A key idea is that the generator admits a *reconstruction procedure*, R . Namely, if D is an efficient algorithm $(1/\text{poly}(m))$ -distinguishing G_f from uniform, then $R^f(D)$ outputs a $\text{poly}(m)$ -size circuit approximating f with advantage $1/\text{poly}(m)$ with high probability. For the purposes of the hardness/randomness paradigm, if we assume in advance that $\text{poly}(m)$ -size circuits cannot $(1/2 + \text{poly}^{-1}(m))$ -approximate f , then we reach a contradiction, and we conclude that the NW PRG is indeed pseudorandom.

Learning from Natural Properties via NW Reconstruction. Carmosino et al. [CIKK16] provide the first and most influential example of this approach, showing that a distinguisher for G_{NW}^f implies the easiness of PAC learning with membership queries. As in the previous section, for the purposes of learning, [CIKK16] apply this reconstruction framework in reverse. The starting point is an assumption of the existence of a *natural property*. Razborov and Rudich introduced the notion of a natural property to pinpoint limitations of existing techniques for proving circuit lower bounds [RR97]. In particular, known proofs of circuit lower bounds were based on natural properties for those circuit classes (including lower bounds for AC^0 [FSS84] and $\text{AC}^0[p]$ for prime p [Raz87, Smo87]). On the other hand, there is no (say, P/poly-constructive) natural property for general boolean circuits unless cryptographic PRGs do not exist. Very roughly, a natural property for a given circuit class C is equivalent to average-case easiness of the Minimum Circuit Size Problem for C : the existence of a poly-time algorithm accepting all truth-tables of low C -complexity and rejecting a large enough fraction of truth-tables with high C -complexity (see Definition 2.31 for the formal details). The assumption of [CIKK16] is in this sense an assumption of the easiness of meta-complexity. The assumed natural property algorithm becomes a distinguisher for G_{NW}^f , and one applies a hybrid argument to obtain a next-bit predictor as specified by the reconstruction procedure.

There are several differences here compared to the setting of G_{DP} , which arise from how the generator makes use of its input. In the case of G_{DP} , several “input” strings $x_1, \dots, x_m$ are given explicitly and independently, and each x_i leads to a distinct label b_i depending on f . In the case of G_{NW} , the input strings

¹²Unlike the setting of white-box learning discussed in Section 4.5, here, the learner only needs the code of a sampler for the marginal distribution over example strings without labels.

to f are given implicitly by a single seed z ; “extracting” the strings x_i from z requires computing some restrictions based on the designs S_i . Due to this more “economical” use of the input, G_{NW} can be made to yield a much longer output from a short seed. For this reason, a natural property suffices to distinguish outputs of G_{NW} from uniformly random strings. On the other hand, in the case of G_{DP} , inputs of length $n \cdot m$ lead to outputs of length only $n \cdot m + m$, and it is not clear whether outputs of G_{DP} have circuit complexity less than this.

A drawback of G_{NW} compared to G_{DP} is that the reconstruction procedure requires oracle access to the function f . This is due to the fact that a given seed $z \sim \mathcal{U}_d$ gives rise to *dependent* input strings $x_1, \dots, x_m = z|_{S_1}, \dots, z|_{S_m}$. Due to the overlap between sets in the NW design, if one wants to know the value of f on a particular choice of x , this imposes restrictions on the values of the other strings x_j appearing in the hybrid. The NW reconstruction procedure handles this with a pre-processing stage, making oracle queries to f to build a table containing $f(x_j)$ for all possible completions of x_j depending on x . Ultimately, this results in a learner requiring membership queries.

Regardless of the limitations of learners resulting from natural properties, the connection to the literature on circuit lower bounds makes their construction worthwhile: by the known natural property for $AC^0[p]$, Carmosino et al. unconditionally obtain a quasi-polynomial time learning algorithm for this circuit class (see Figure 6).

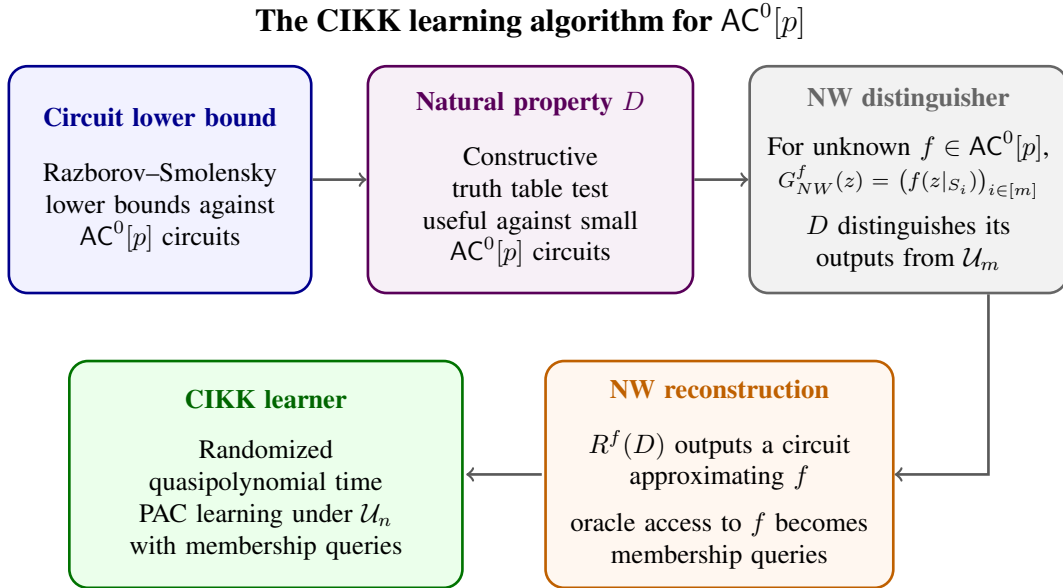


Figure 6: The diagram presents the key steps of the $AC^0[p]$ learning algorithm of Carmosino, Impagliazzo, Kabanets, and Kolokolova [CIKK16], a breakthrough result in computational learning theory. The Razborov and Smolensky circuit lower bounds [Raz87, Smo87] yield a natural property [RR97] against $AC^0[p]$. For an unknown target $f \in AC^0[p]$, this property distinguishes the output of the Nisan-Wigderson generator G_{NW}^f [NW94] from a random truth table. The NW reconstruction procedure converts the distinguisher into a hypothesis using membership query access to f . To date, this complexity-theoretic approach remains the only known learning technique for this circuit class.

In a 2017 follow-up, the authors further show that, from so-called *tolerant* natural properties, one obtains *agnostic* learners [CIKK17]. A tolerant natural property is essentially equivalent to easiness of the Minimum Approximate Circuit Size Problem, MACSP, which asks, for a given truth-table T and parameters s and δ , whether there exists a circuit of size s computing T on at least a $1 - \delta$ fraction of inputs. Under

this framework, the authors generalize the learning algorithm for $AC^0[p]$ to target functions with distance $1/\text{poly}(n)$ from $AC^0[p]$.

Problem 5.1 (Learning $AC^0[p]$ from random examples). Is there a subexponential time algorithm that learns $AC^0[p]$ circuits under the uniform distribution using random examples instead of membership queries?

For recent work related to [Problem 5.1](#), we refer to [\[GJK⁺24, CJ26\]](#).

From Uniform Learning to Samplable Distributions via Distributional Inversion. The approach of [\[CIKK16\]](#) as described above only yields a PAC learner working on the uniform distribution. A subsequent work due to Binnendyk et al. [\[BCK⁺22\]](#), following the *indirect learning* approach of [\[FJS91\]](#), shows that the result of [\[CIKK16\]](#) can be extended to obtain learners working over any efficiently samplable distribution, provided (1) that the code of the sampler is known to the learner, and (2) that the target function *composed* with this sampler belongs to the learner’s concept class. Below, we restrict our attention to the case that the concept class as well as the sampler complexity of the target distribution are unrestricted polynomial-size circuits.

It is well-known that a natural property would preclude the existence of a PRG with polynomial stretch: outputs of the generator have bounded circuit complexity, while uniformly random strings almost always have maximum circuit complexity [\[RR97\]](#). Then, by a sequence of results from cryptography [\[GGM86, HILL99, Yao82, IL89\]](#), a natural property implies the existence of efficient distributional inverters for every poly-time function. Roughly, when given the output y of some function $g : \{0, 1\}^n \rightarrow \{0, 1\}^{m(n)}$, such an inverter outputs a *nearly uniformly distributed* member of $\{x \mid g(x) = y\}$.

Let A be the PAC learner of [\[CIKK16\]](#) working over the uniform distribution with membership queries. Suppose we would like to learn some target function f over a distribution $D \in \text{PSAMP}$. The definition of PSAMP means that D can be sampled by a polynomial-time machine; i.e. there is a poly-time algorithm S such that $S(r)$ is distributed according to D when the string r is uniformly distributed. Let I be a distributional inverter for S .

The key idea is to simulate A on the target function $f(S(-))$. Clearly, knowing S and given membership-query access to f , we can simulate membership queries to $f(S(-))$. Then by [\[CIKK16\]](#), with high probability, A outputs a hypothesis h such that

$$\Pr_{r \sim \mathcal{U}_{\text{poly}(n)}} [h(r) = f(S(r))] \geq 1 - 1/\text{poly}(n).$$

We output the hypothesis $h'(-) \triangleq h(I(-))$.

Intuitively, on input $x \sim D$, we have $h'(x) = h(I(S(r))) \approx h(r)$, where $r \sim \mathcal{U}_{\text{poly}(n)}$. Since $h(r) = f(S(r)) = f(x)$ with high probability, one can show that the hypothesis h' successfully approximates f over D , as desired.

We refer to [\[Pic25\]](#) for additional settings where one can derive learning consequences from complexity lower bound assumptions.

5.2 Solomonoff Induction: Unsupervised Learning via Kolmogorov Complexity

In this section, we explain how to efficiently achieve sequence prediction under a meta-complexity assumption about the feasibility of estimating time-bounded Kolmogorov complexity. Formally, we have the following result.

Theorem 5.2 ([HN26a, Theorem 2.1]). *If $\text{GapK}^t\text{-vs-K} \in \text{pr-BPP}$, then $(m, n, s, t, \varepsilon, \delta)$ -extrapolation can be solved in randomized time $\text{poly}(n, m, s, t, 1/\varepsilon, 1/\delta)$ for $m \geq \tilde{O}(n^2 \cdot s \cdot \varepsilon^{-6} \cdot \delta^{-5})$.*

First, we review Solomonoff induction, which considers this extrapolation task in the time-unbounded setting.

5.2.1 The Classical Theory

In Solomonoff induction [Sol64a, Sol64b] (see [HQC24] for a modern introduction), given a prefix $x_{<i} = (x_1, \dots, x_{i-1})$, the extrapolation algorithm predicts the next block $x_i \in \{0, 1\}^n$ by *conditioning* a certain “universal” distribution on the event that the generated sequence begins with $x_{<i}$. Equivalently, one gives higher probability weight to continuations that make the full string $(x_{<i}, x_i)$ have small Kolmogorov complexity.¹³ Roughly, we predict the next block x_i given the prefix $x_{<i}$ according to the Solomonoff (semi)measure using

$$S(x_i \mid x_{<i}) \propto \sum_y 2^{-K(x_{<i}x_iy)},$$

where the symbol $\propto$ means “proportional to”. This is a form of *Occam’s Razor*, i.e., short explanations are preferred. It is then possible to show that, for sequences generated by a randomized program of description length s , the resulting predictor is accurate on most $i \in [m]$ once the number of observations $m \gg s/\varepsilon^2$.

To show that this predictor works as claimed above, we combine the following ingredients:

1. *Universality of the Solomonoff measure.* If the full sequence $x = (x_1, \dots, x_m)$ is generated by a randomized program M of description length s , then the measure S dominates the true source μ_M up to a multiplicative factor $2^{-O(s)}$, i.e.,

$$S(x_1, \dots, x_m) \geq 2^{-O(s)} \cdot \mu_M(x_1, \dots, x_m)$$

for every sequence $(x_1, \dots, x_m)$.

2. *Log-loss bound and average KL-divergence.* Taking logarithms and averaging over $x = (x_1, \dots, x_m) \sim \mu_M$, one obtains the log-loss bound

$$\mathbb{E}_{x \sim \mu_M} \left[\log \frac{\mu_M(x_1, \dots, x_m)}{S(x_1, \dots, x_m)} \right] \leq O(s).$$

The key point is that this global bound can be decomposed into one-step prediction errors. In more detail, for every $x = (x_1, \dots, x_m)$ with $\mu_M(x) > 0$, we can factor both distributions sequentially:

$$\mu_M(x) = \prod_{i=1}^m \mu_M(x_i \mid x_{<i}), \quad S(x) = \prod_{i=1}^m S(x_i \mid x_{<i}).$$

Consequently,

$$\log \frac{\mu_M(x)}{S(x)} = \sum_{i=1}^m \log \frac{\mu_M(x_i \mid x_{<i})}{S(x_i \mid x_{<i})}.$$

¹³In this section, we take K to denote *prefix complexity*, which is the same as our stated definition of Kolmogorov complexity, except it requires that the set of inputs on which the chosen universal machine halts is *prefix-free*. By Kraft’s inequality, $\sum_y 2^{-K(wy)} \leq 1$ in the case of prefix complexity, whereas the same sum over all finite strings y diverges in the case of plain complexity. Though the ideas in this section can be reformulated to accommodate plain Kolmogorov complexity, we adopt this convention to match the classical literature.

Now taking expectation over $x \sim \mu_M$ and applying linearity of expectation,

$$\begin{aligned} \mathbb{E}_{x \sim \mu_M} \left[\log \frac{\mu_M(x)}{S(x)} \right] &= \sum_{i=1}^m \mathbb{E}_{x_{\leq i} \sim \mu_M} \left[\log \frac{\mu_M(x_i | x_{<i})}{S(x_i | x_{<i})} \right] \\ &= \sum_{i=1}^m \mathbb{E}_{x_{<i} \sim \mu_M} \left[\sum_{y \in \{0,1\}^n} \mu_M(y | x_{<i}) \log \frac{\mu_M(y | x_{<i})}{S(y | x_{<i})} \right]. \end{aligned}$$

But the inner sum is exactly the KL-divergence from the true conditional distribution of the next block to the Solomonoff posterior. Therefore, combining the equations above,

$$\sum_{i=1}^m \mathbb{E}_{x_{<i} \sim \mu_M} \left[\text{KL}(\mu_M(\cdot | x_{<i}) \| S(\cdot | x_{<i})) \right] \leq O(s).$$

3. *Passing to statistical distance.* Finally, for any large enough $m = O(s \cdot 1/\varepsilon^2 \cdot 1/\delta)$, using a standard Markov argument and applying Pinsker's inequality, one obtains that the predicted next-block distribution is ε -close in total variation distance to the true conditional distribution produced by M except for at most a δ -fraction of $i \in [m]$.

Thus, in the time-unbounded setting, Solomonoff induction solves the extrapolation problem once the sequence is sufficiently longer than the description of the underlying generator program.

5.2.2 The Time-Bounded Setting

We consider a simpler setting where $\delta > 0$ is a small constant, and describe a randomized algorithm that runs in time $\text{poly}(n, m, t, 1/\varepsilon)$ and succeeds as long as $m = m(n, s, \varepsilon)$ is sufficiently large. The proof of [Theorem 5.2](#), which provides a polynomial-time procedure when $\delta = o(1)$, is significantly more involved. We refer to [\[HN26a\]](#) for the details.

To move to the time-bounded world, we replace the universal distribution by the *time-bounded* universal distribution $Q_{t,z}$, where z is an auxiliary string.

Definition 5.3 (Time-Bounded Universal Distribution with Advice). Fix $t \in \mathbb{N}$ and an advice string $z \in \{0, 1\}^*$. The t -time-bounded universal distribution $Q_{t,z}$ is the distribution induced by the random variable $U^r(w, z, 1^t)$, where $w \sim \mathcal{U}_t$ is chosen uniformly from $\{0, 1\}^t$. Equivalently, for every string $y \in \{0, 1\}^*$,

$$Q_{t,z}(y) \triangleq \Pr_{w \sim \mathcal{U}_t, U^r} [U^r(w, z, 1^t) = y].$$

When $z = \epsilon$, we omit the advice string and write Q_t instead of $Q_{t,z}$. For convenience, we also define

$$q^t(y | z) \triangleq -\log Q_{t,z}(y).$$

We also need the following definition.

Definition 5.4 (Universal Extrapolation Distribution). The *universal extrapolation distribution*

$$\text{Next}_n(y; Q_{t,z})$$

is the distribution of the next n bits of a $Q_{t,z}$ -sample, conditioned on extending the prefix y .

It is important to distinguish the two roles here: z is an advice string in the Kolmogorov complexity sense, while y is the observed prefix in the probabilistic conditioning sense. This distinction is a crucial idea from [HN26a] to implement Solomonoff’s theory in the time-bounded setting.¹⁴ Jumping ahead, what is special in the time-bounded setting is that one splits the given prefix into advice and context strings for efficiency reasons and then proves that this restricted-context extrapolation is still correct.

In more detail, prior work already gave a *universal sampling procedure* for approximately sampling from $\text{Next}_n(y; Q_{t,z})$, but its running time is exponential in the *computational depth* of y given z [IL90, HN23].¹⁵ In other words, consider the computational depth quantity

$$\text{cd}^t(y \mid z) \triangleq q^t(y \mid z) - K(y \mid z),$$

where we set $\text{cd}_t(y) \triangleq \text{cd}_t(y \mid \epsilon)$. If one tries to extrapolate directly from the full prefix by setting $y = x_{<i}$ and $z = \epsilon$, the runtime can be exponential in $\text{cd}_t(x_{<i})$. For some sequences generated by an s -bit randomized program, this depth can be as large as $\Omega(s)$.

As alluded to above, a key idea from [HN26a] is to reorganize the observed prefix into an *advice part* and a *context part*. For appropriate parameters $w, b \in \mathbb{N}$ with $m = b \cdot w$, the sequence $(x_1, \dots, x_m)$ is partitioned into b larger blocks, each consisting of w of the original n -bit symbols. When predicting position $j \in [w]$ inside block $i \in [b]$, the learner uses all previous blocks as advice, but keeps only the prefix of the current block up to position $j - 1$ as context. If we write $\bar{x}_i$ for the i -th large block, $\bar{x}_{<i}$ for the concatenation of the previous blocks, and $\bar{x}_i^{<j}$ for the prefix of the current block before position j , then the predictor outputs according to

$$\text{Next}_n(\bar{x}_i^{<j}; Q_{t, \bar{x}_{<i}}).$$

In other words, the long-range history is moved into *advice*, while a short local prefix is kept as *context*.

Next, we explain in more detail how this approach reduces the running time while maintaining correctness of the corresponding predictor. For a formal presentation, see [HN26a].

I. Bounding the running time. For the runtime analysis, we need two ingredients: the *slow growth law* and the *chain rule for computational depth*.

The slow growth law states that if $x \sim \mu_M = U^r(M, \epsilon, 1^t)$, where M is a t -time randomized machine of description length s , then with high probability

$$\text{cd}^{\text{poly}(t)}(x) \leq O(s).$$

The chain rule states that, for a string $x = (x_1, \dots, x_{m'})$ with each $|x_i| \leq n'$, and any large enough t' , we have

$$\sum_{i=1}^{m'} \text{cd}^{\text{poly}(t')}(x_i \mid x_{<i}) \leq \text{cd}^{t'}(x) + m' \cdot O(\log t').$$

These two results are known to hold under the assumption of [Theorem 5.2](#), as established by [HN26a].

Putting the results above together, we get that the expected value of $\text{cd}^{\text{poly}(t)}(x_i \mid x_{<i})$ over a random position $i \sim [m]$ is $O(s/m + \log t) = O(\log t)$. By Markov’s inequality,

$$\Pr_{i \sim [m]} \left[\text{cd}^{\text{poly}(t)}(x_i \mid x_{<i}) \geq \delta^{-1} \cdot O(\log t) \right] \leq \delta.$$

¹⁴In the classical theory, since computational efficiency is not relevant, there is no need to consider an advice string, and the entire observed prefix is used for conditioning. We stress that in both settings – time bounded and time unbounded – moving the entire prefix to the advice portion does not guarantee extrapolation.

¹⁵This algorithm assumes that auxiliary-input one-way functions do not exist, which follows from the hypothesis of [Theorem 5.2](#).

In other words, for most choices of i and x , the time-bounded universal sampling procedure (approximately) samples from $\text{Next}_n(x_i; Q_{t, \bar{x}_{<i}})$ in time $2^{O(\delta^{-1} \cdot \log t)} = \text{poly}(t)$, for $\delta = \Omega(1)$.¹⁶

One can perform a similar runtime analysis once we break the sequence $x = (x_1, \dots, x_m)$ into b larger blocks, each consisting of w of the original n -bit symbols x_i , and use the universal extrapolation procedure to sample instead from $\text{Next}_n(\bar{x}_i^{<j}; Q_{t, \bar{x}_{<i}})$. (In order to guarantee correctness, the proof employs $w = O(\varepsilon^{-2} \cdot \delta^{-1} \cdot \log t)$, as explained below.)

II. Correctness of the time-bounded predictor. Recall that our predictor partitions $x = (x_1, \dots, x_m)$ into b length- w blocks. Given an index $j \in [w]$ inside block $i \in [b]$, it uses the universal sampling procedure in order to predict according to $\text{Next}_n(\bar{x}_i^{<j}; Q_{t, \bar{x}_{<i}})$. By the analysis explained above, for most blocks the sampler is correct and we (approximately) sample from $\text{Next}_n(\bar{x}_i^{<j}; Q_{t, \bar{x}_{<i}})$. It remains to show that this is close to the true distribution, i.e.,

$$\Pr_{i, j, (\bar{x}_{<i}, \bar{x}_i^{<j}) \sim \mu_M} \left[\Delta_{\text{TV}}(\bar{\mathcal{X}}_i^j \mid \bar{x}_{<i} \bar{x}_i^{<j}, \text{Next}_n(\bar{x}_i^{<j}; Q_{t, \bar{x}_{<i}})) \leq \varepsilon \right] \geq 1 - \delta.$$

To prove this, let $\bar{\mathcal{X}}_i$ denote the random i -th block under μ_M , and let $\bar{\mathcal{X}}_{<i}$ denote the concatenation of the previous blocks. Up to replacing t by $\text{poly}(t)$ (which we suppress in the notation), similarly to the chain rule for computational depth described above, it is possible to show that for every $(\bar{x}_1, \dots, \bar{x}_b)$,

$$\sum_{i=1}^b q^t(\bar{x}_i \mid \bar{x}_{<i}) \leq q^t(\bar{x}_1, \dots, \bar{x}_b) + b \cdot O(\log t).$$

On the other hand, the time-bounded universal distribution Q_t dominates every s -bit source μ_M up to a factor $2^{-O(s)}$. Equivalently,

$$q^t(\bar{x}_1, \dots, \bar{x}_b) \leq O(s) - \log \mu_M(\bar{x}_1, \dots, \bar{x}_b)$$

for every sequence in the support of μ_M . Combining the two inequalities and expanding the source probability block-by-block,

$$\sum_{i=1}^b \left(q^t(\bar{x}_i \mid \bar{x}_{<i}) + \log \mu_M(\bar{x}_i \mid \bar{x}_{<i}) \right) \leq O(s) + b \cdot O(\log t).$$

Next, we average over $(\bar{x}_1, \dots, \bar{x}_b) \sim \mu_M$. For each $i \in [b]$, we claim that the expectation of the i -th summand is exactly

$$\mathbb{E}_{\bar{x}_{<i} \sim \mu_M} \left[\text{KL}((\bar{\mathcal{X}}_i \mid \bar{x}_{<i}) \parallel Q_{t, \bar{x}_{<i}}) \right].$$

Indeed, by definition, $q^t(y \mid \bar{x}_{<i}) = -\log Q_{t, \bar{x}_{<i}}(y)$, and so

$$q^t(y \mid \bar{x}_{<i}) + \log \mu_M(y \mid \bar{x}_{<i}) = \log \frac{\mu_M(y \mid \bar{x}_{<i})}{Q_{t, \bar{x}_{<i}}(y)}.$$

Taking expectation over $y \sim (\bar{\mathcal{X}}_i \mid \bar{x}_{<i})$ yields the desired statement. As a consequence,

$$\mathbb{E}_{i \sim [b]} \mathbb{E}_{\bar{x}_{<i} \sim \mu_M} \left[\text{KL}((\bar{\mathcal{X}}_i \mid \bar{x}_{<i}) \parallel Q_{t, \bar{x}_{<i}}) \right] \leq \frac{O(s)}{b} + O(\log t).$$

¹⁶With low probability over i and x , we can abort the algorithm and output “fail” if its running time reaches a fixed threshold.

In other words, the amortized KL-divergence over the b blocks is $O(\log t)$ for $b = \Omega(s/\log t)$.

The remaining steps follow from standard arguments. First, we apply the chain rule for KL-divergence to distribute the bound of $O(\log t)$ across the w positions in a block, yielding an average KL-divergence of $O(\log t)/w$ over positions $j \in [w]$. For $w = \Omega(\varepsilon^{-2} \cdot \delta^{-1} \cdot \log t)$, we obtain a bound of $O(\log t)/w = \varepsilon^2 \cdot \delta$ per symbol. Then, from Markov's inequality, the per-symbol KL-divergence is at most ε^2 for a $(1 - \delta)$ -fraction of positions inside a block. Finally, the desired bound on statistical distance then follows from Pinsker's inequality.

To sum up, the predictor satisfies the desired guarantees provided that $m = b \cdot w = \Omega((s/\log t) \cdot (\varepsilon^{-2} \delta^{-1} \log t)) = \Omega(s \cdot \varepsilon^{-2} \cdot \delta^{-1})$.

As mentioned above, in order to obtain an efficient predictor when $\delta = o(1)$, a more involved argument seems required, and the known lower bound on m as a function of ε and δ is sub-optimal.

Problem 5.5 (Optimal Parameters in Extrapolation from Meta-Complexity). Prove that if $\text{GapK}^t\text{-vs-K} \in \text{pr-BPP}$, then $(m, n, s, t, \varepsilon, \delta)$ -extrapolation can be solved in randomized time $\text{poly}(n, m, s, t, 1/\varepsilon, 1/\delta)$ for $m \geq \tilde{O}(s \cdot \varepsilon^{-2} \cdot \delta^{-1})$.

5.3 Equivalences Between Learning and Meta-Complexity

In [Section 5.1](#) we described several settings where one can get efficient learning from suitable distinguishers. In turn, the necessary distinguishers follow from the existence of efficient algorithms for meta-computational problems. The latter allow one to distinguish randomness from structure, or equivalently, input strings that can be compressed from (a large fraction of) incompressible ones.

We now discuss two settings where appropriate relaxations of the meta-computational problems are actually *computationally equivalent* to the learning task. Let $\mathcal{C}$ be a class of functions (e.g., polynomial-size Boolean circuits), and let Γ be a learning model (e.g., PAC learning under the uniform distribution from random examples). We present equivalences of the following form:

$\mathcal{C}$ admits efficient learning algorithms in model $\Gamma \iff$ The meta-computational problem $L_{\mathcal{C}, \Gamma}$ is easy

In the equivalence, $L_{\mathcal{C}, \Gamma}$ is a suitable computational problem that we associate to the learning model and concept class.

The idea that the design of learning algorithms is equivalent to the search for efficient procedures that detect patterns in data offers a fundamentally new perspective: while *learning is inherently a sublinear task*, where an algorithm must inspect a small portion of a large object (the unknown function) and produce an approximate representation of it (the output hypothesis), a typical problem from meta-complexity, such as MCSP and its variants, involves deciding if an input string is “random” or “structured”, a standard algorithmic problem where *the algorithm has access to the entire input*.

Concretely, as explained below, [\[ILO20\]](#) shows that a circuit class $\mathcal{C}$ can be PAC learned under the uniform distribution from random examples in polynomial time if and only if a variant of MCSP for partial functions and with respect to $\mathcal{C}$ -circuit complexity is easy on average. Similarly, [\[CIKK16\]](#) can be phrased (see [\[OS17\]](#)) as an equivalence between the efficient learnability of $\mathcal{C}$ with membership queries under the uniform distribution and the existence of a promise-BPP natural property against $\mathcal{C}$ (see [Section 2.4.5](#)).

Next, we explain in more detail the aforementioned equivalences. For simplicity, we mostly focus on the case where $\mathcal{C}$ corresponds to general Boolean circuits, though the equivalences hold more generally under mild assumptions on $\mathcal{C}$ related to their closure properties and basic expressive power.

Learning from Random Examples. Ilango, Loff, and Oliveira [ILO20] rephrase the ideas developed in [Vad17] (see Section 5.1.2), making more explicit a connection between the easiness of meta-complexity and the possibility of learning. More specifically, the authors show that if a certain problem from meta-complexity, denoted Partial-MCSP, is easy on average, then PAC learning general Boolean circuits on the uniform distribution from random examples is easy. Conversely, a learning algorithm that uses random examples implies the easiness on average of Partial-MCSP.

In [ILO20], Partial-MCSP refers to the following formulation of a circuit minimisation problem on partially defined functions.

An instance of the language Partial-MCSP consists of an integer $n \in \mathbb{N}$ represented in unary, a collection of input/output pairs $((x_i, b_i))_{i \in [m]}$ for some $m \in \mathbb{N}$, where $x_i \in \{0, 1\}^n$ and $b_i \in \{0, 1\}$ for each $i \in [m]$, and a threshold parameter $s \in \mathbb{N}$.

$(1^n, ((x_i, b_i))_{i \in [m]}, s)$ is a yes-instance of Partial-MCSP iff there exists a Boolean circuit C of size at most s such that $C(x_i) = b_i$ for all $i \in [m]$.

The authors further consider an *average-case* notion of Partial-MCSP, in which an algorithm A is said to solve Partial-MCSP on average over the uniform distribution with m samples, advantage γ , and parameter s if for every boolean function $f \in \text{SIZE}[s]$,

$$\left| \Pr_{\substack{A, \\ x_1, \dots, x_m \sim \mathcal{U}_n^m}} [A(x_1, \dots, x_m; f(x_1), \dots, f(x_m)) = 1] - \Pr_{\substack{A, \\ x_1, \dots, x_m \sim \mathcal{U}_n^m \\ b_1, \dots, b_m \sim \mathcal{U}_1^m}} [A(x_1, \dots, x_m; b_1, \dots, b_m) = 1] \right| \geq \gamma(n).$$

For large enough $m(n) = \text{poly}(s(n))$ and small enough $\gamma(n) = \Omega(1/\text{poly}(n))$, we remark that this is a rather lenient notion of average-case easiness over the uniform distribution: it is enough to distinguish labels consistent with a function from the class from random labels.

Now observe that the task above is precisely the problem of breaking the Direct Product Generator G_{DP}^f with the input $\bar{x}$ sampled uniformly at random and the target function f belonging to $\text{SIZE}[s]$. Applying the techniques discussed in the context of [Vad17], such an algorithm may be transformed into a weak learner for $\text{SIZE}[s]$ working on examples drawn from the uniform distribution. To obtain a strong PAC learner for $\text{SIZE}[s]$ over the uniform distribution, the authors apply a result of [BL93] to boost accuracy.

Conversely, it is not difficult to show that an algorithm that learns $\text{SIZE}[s]$ under the uniform distribution from random examples can be used to solve the average-case version of Partial-MCSP introduced above. Given a sequence $(x_1, \dots, x_m; \ell_1, \dots, \ell_m)$, where we assume $m(n)$ is large enough, our task is to decide whether the labels $\ell_1, \dots, \ell_m$ are *consistent* with a fixed but unknown $f \in \text{SIZE}[s]$ or *random*. We simulate the learner over the sample set $\{(x_1, \ell_1), \dots, (x_{m-1}, \ell_{m-1})\}$. Let h be its output hypothesis. If $h(x_m) = \ell_m$, we output 1 (meaning “consistent”); otherwise we output 0 (meaning “random”). On the one hand, if the labels are consistent with some function $f \in \text{SIZE}[s]$, h is likely to correctly label x_m by ℓ_m , in which case we output 1. On the other hand, in the random case, no matter the output hypothesis h , since the label ℓ_m is an independent random bit we get that $h(x_m) = \ell_m$ with probability exactly $1/2$. Therefore, we distinguish the two cases with noticeable probability, i.e., we solve Partial-MCSP on average.

Learning from Membership Queries. The preceding discussion used the term natural property informally. The precise randomized formulation relevant here is the notion of a pr-BPP-natural property introduced in Section 2.4.5. Such a property is decided by a randomized algorithm running in time polynomial

in the truth-table length $N = 2^n$. It accepts with probability at least $2/3$ every truth table in some dense set, while accepting the truth table of each function in $\mathcal{C}[s]$ with probability at most $1/3$. No guarantee is imposed on the remaining truth tables: on these inputs the acceptance probability may lie between $1/3$ and $2/3$, which explains the promise in pr-BPP. Informally, the algorithm efficiently recognizes many unstructured functions while rejecting every function of low $\mathcal{C}$ -complexity. This can be seen as a weak average-case procedure for the MCSP problem for $\mathcal{C}$ -circuits.

As discussed above, Carmosino, Impagliazzo, Kabanets, and Kolokolova [CIKK16] turn such a property into a learner for $\mathcal{C}$ under the uniform distribution: the property distinguishes truth tables produced by the Nisan–Wigderson generator from random truth tables, and the NW reconstruction procedure converts this distinguisher into a hypothesis for the target. The reconstruction uses oracle access to the target, which accounts for the membership queries.

Oliveira and Santhanam [OS17] noted that the reverse implication from learning to natural properties also holds, even if one starts with a barely non-trivial learning algorithm. Concretely, a non-trivial learner is a randomized oracle algorithm A^f that, for some constant $c \geq 1$ and every $f \in \mathcal{C}[s]$, outputs with probability at least $1 - 1/n$ a hypothesis h satisfying

$$\Pr_{x \sim \mathcal{U}_n} [h(x) = f(x)] \geq \frac{1}{2} + \gamma(n), \quad \gamma(n) = \frac{1}{n^c},$$

and runs in time $T(n) = 2^n / n^{\omega(1)}$. The super-polynomial saving in the denominator is a convenient way to make the definition independent of c ; a fixed sufficiently large polynomial saving also works.

The idea works as follows. Given $w \in \{0, 1\}^N$, simulate A^f for $f = \text{fnc}^w$, answering each membership query directly from w . Estimate the agreement of the resulting h with f to additive error $\gamma/4$ using polynomially many fresh uniform inputs. Reject if the estimate exceeds $1/2 + \gamma/2$, and accept otherwise. If $f \in \mathcal{C}[s]$, the learner succeeds and the test rejects with probability at least $2/3$.

To see why the test accepts most functions, fix a size bound $S \leq T(n)$. By a simple counting argument, there are at most $2^{O(S \log S)}$ Boolean circuits of size S . For any fixed circuit h and a uniformly random function f , the values $1[h(x) = f(x)]$ are independent unbiased bits. A Chernoff bound therefore gives

$$\Pr_f \left[\Pr_x [h(x) = f(x)] \geq \frac{1}{2} + \frac{\gamma}{4} \right] \leq \exp(-\Omega(\gamma^2 2^n)).$$

Thus the fraction of functions that can be approximated by a fixed hypothesis h is at most $\exp(-\Omega(\gamma^2 2^n))$. A union bound over all possible h (circuits of size at most S) shows that the fraction of functions approximated by any such circuit is at most $\exp(O(S \log S) - \Omega(\gamma^2 2^n)) = o(1)$ under the non-trivial time bound. Since every hypothesis output by A has size at most $T(n)$, the test accepts almost every truth table with high probability. It runs in time $T(n) \cdot \text{poly}(n) = \text{poly}(N)$, so it satisfies constructivity, density, and $\mathcal{C}[s]$ -hardness from Section 2.4.5. Truth tables outside the two guaranteed cases may have intermediate acceptance probability, which is precisely why the promise class pr-BPP is appropriate. Together with the learning-from-natural-properties direction, this gives the desired equivalence, up to standard parameter translations.

It would be interesting to further develop the dictionary between meta-complexity and learning.

Problem 5.6 (Equivalences Between Learning and Meta-Complexity). Can we characterize efficient learnability in other learning frameworks via meta-complexity?

Next, we discuss a surprising application of the equivalence between learning and meta-complexity.

5.4 Learning Speedups: Reducing the Running Time via Meta-Complexity

A PAC learning algorithm for a class $\mathcal{C}$ of functions comes with three parameters of interest: the *confidence* that it succeeds, the *error* of its output hypothesis, and its *running time*. While it is not difficult to amplify the confidence of a learning algorithm [KV94a, Chapter 4], reducing its error is much more subtle. Originally developed to address a question in learning theory, *boosting* (see [SF14]) is a key technique in machine learning that allows one to reduce the error of the output hypothesis of a PAC learner. Thus both the confidence and error parameters can be boosted. Could it be the case that the *running time* of a learning algorithm can also be improved in a generic fashion?

Surprisingly, [OS17] relies on complexity-theoretic techniques [CIKK16] to show that, in some cases, one can make learning algorithms run much faster! They termed this phenomenon *learning speedups*: a generic approach to improve the *running time* of a learning algorithm. How is this possible? Key to the learning speedup compiler is to explore a fourth dimension of learning algorithms: they tend to solve more difficult tasks than the ones we typically apply them to.

To achieve a learning speedup, we explore the *complexity* of the unknown function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ in $\mathcal{C}$ that we are challenged to learn, by *reducing this task* to the learnability of a *more complex* function $g: \{0, 1\}^m \rightarrow \{0, 1\}$ over a *smaller* number m of input bits. The rationale is simple: on an instance of the learning problem of length n , if we run the original (slow) learning algorithm on a problem of much smaller dimension $m \ll n$, we obtain a faster learning algorithm as a function of n . For this to succeed, we need to guarantee that g , while more complex than f , is still in $\mathcal{C}$ (since the original learner is only promised to learn functions in $\mathcal{C}$), and we must discover a way to implement the reduction between learning problems. To achieve the latter, we *implicitly* simulate the original learning algorithm on g (while only having access to f), and show that a good output hypothesis for g can be transformed into a good hypothesis for f .¹⁷

Here we explain the proof of the following speedup result.

Theorem 5.7 (From Nontrivial Learning to Subexponential Time Learning [OS17]). *Let $\mathcal{C}$ be a circuit class at least as powerful as bounded-depth circuits with parity gates. Suppose that, for every constant $k \geq 1$, $\mathcal{C}$ -circuits of size n^k can be PAC learned under the uniform distribution with membership queries in time $\leq 2^n/n^{\omega(1)}$. Then, for every $k \geq 1$ and $\varepsilon > 0$, $\mathcal{C}$ -circuits of size n^k can be PAC learned under the uniform distribution with membership queries in time $O(2^{n^\varepsilon})$.*

Proof Sketch. Fix $k \geq 1$ and $\varepsilon > 0$. The argument is the composition of the two directions described in Section 5.3. First, the assumed non-trivial learner yields a pr-BPP-natural property (Definition 2.31). Indeed, on the truth table of an m -variate function g , we simulate the learner using the truth table to answer its membership queries, and then test whether its output hypothesis has noticeable agreement with g . The learner succeeds when g is computed by a small $\mathcal{C}$ -circuit, whereas the counting argument from the preceding section shows that almost every function cannot be noticeably approximated by a circuit of size $2^m/m^{\omega(1)}$. Reversing the output of this test therefore gives, for every constant d , a pr-BPP-natural property against $\mathcal{C}[m^d]$. Its running time is polynomial in the truth-table length 2^m , and hence is $2^{O(m)}$.

We now apply the learning-from-natural-properties construction discussed in Section 5.1. Let $f \in \mathcal{C}[n^k]$ be the target function, and choose a parameter $m = n^\gamma$, where $\gamma > 0$ will be fixed below. The black-box Nisan–Wigderson generator associates with f a family of auxiliary functions $g_z: \{0, 1\}^m \rightarrow \{0, 1\}$. Since $\mathcal{C}$ contains bounded-depth circuits with parity gates and has the standard closure properties, each g_z is

¹⁷At the technical level, this idea is implemented in a more sophisticated fashion, by reducing the learnability of f to the task of learning a collection $\{g_i\}_{i \in \mathcal{S}}$ of functions, in a way that the smaller functions g_i collectively encode enough information that allows us to approximate f using approximators to randomly sampled functions g_i that are implicitly constructed from f .

computed by a $\mathcal{C}$ -circuit of size m^d , for some constant $d = d(k, \gamma)$. This is the reason the theorem assumes non-trivial learners for $\mathcal{C}[n^d]$ for every constant d , rather than only for one fixed size bound.

The natural property against $\mathcal{C}[m^d]$ rejects every truth table $\text{tt}(g_z)$ while accepting a dense set of random truth tables. It therefore distinguishes the NW outputs associated with f from a uniformly random function on m bits. The NW reconstruction procedure converts this distinguisher into a hypothesis that approximates f under the uniform distribution. As explained earlier, the reconstruction can form the required truth-table entries of the functions g_z using oracle access to f ; these become the membership queries of the resulting learner.

The natural property and the reconstruction run in time polynomial in 2^m , so the total running time is $2^{O(m)} \cdot \text{poly}(n) = 2^{O(n^\gamma)}$. Choose $\gamma < \varepsilon$ small enough to absorb the constant hidden in the exponent. Standard accuracy and confidence amplification then gives a PAC learner running in time $O(2^{n^\varepsilon})$, as desired. $\square$

As explained above, the key conceptual insight behind the learning speedup compiler is that to learn a function $f \in \mathcal{C}[n^k]$ in sub-exponential time, we invoke the nontrivial learner on concepts of size m^ℓ , where ℓ is larger than k but the input length m is smaller than n . Crucially, we save time by using that the original learner is able to succeed on relatively harder learning tasks of smaller instance length. The tradeoff is considerably more general. In particular, a nontrivial learner that learns exponential size concepts can be transformed into a polynomial-time learner for polynomial-size concepts [OS17].

Despite its obvious relevance and potential for applications, this intriguing phenomenon is currently restricted to PAC learning algorithms that work under the uniform distribution and can make membership queries. It also requires the class $\mathcal{C}$ to be reasonably expressive, since this is needed to implement the techniques from [CIKK16].

Problem 5.8 (Learning Speedups in Different Learning Frameworks). Systematically investigate learning speedups in different learning models. In particular, is there a generic speedup for learning under the uniform distribution from random examples?

6 Learning vs Compression via Shannon’s Information Theory

We have seen several connections between learning and compression through the lens of meta-complexity and algorithmic information theory. In this section, we explore connections from the perspective of Shannon’s information theory, taking the theory of source coding as our point of departure. As we will see, the relationship between computational learning and meta-complexity can be understood as a complexity-theoretic counterpart of the classical relationship between learning and universal coding. Our goal is to review several tools for establishing such connections and to explain how notions from meta-complexity fit into this framework. We also briefly discuss how this perspective relates to language modeling, one of the central paradigms of modern machine learning.

6.1 Starting from Shannon’s Information Theory

Shannon’s information theory [Sha48] provided a precise mathematical framework for quantifying “information” and characterized what can and cannot be achieved in data compression and communication. Originally developed to address problems in communication engineering, the theory laid the foundations for the mathematical study of information and has profoundly influenced theoretical computer science.

One of the central results of Shannon’s theory, and the one most relevant to our discussion, is the noiseless source coding theorem. In the setting of this theorem, a message x is drawn from a fixed distribution $\mathcal{D}$ over a finite message space $\mathcal{X}$, and $\mathcal{D}$ is known to both the encoder and the decoder. The goal is to encode x as a short binary string from which x can be uniquely recovered. By fixing a canonical binary representation, we may assume without loss of generality that $\mathcal{X} = \{0, 1\}^n$ for some n .

Shannon’s construction begins by ordering the messages in $\text{Support}(\mathcal{D})$ as $x_1, \dots, x_m$ so that $p_1 \geq p_2 \geq \dots \geq p_m$, where $p_i \triangleq \mathcal{D}(x_i)$. For each i , let

$$\ell_i \triangleq \left\lceil \log \frac{1}{p_i} \right\rceil \quad \text{and} \quad P_{< i} \triangleq \sum_{j < i} p_j.$$

The encoder Enc maps x_i to the first ℓ_i bits after the binary point in the binary expansion of $P_{< i}$. One can verify that the original message is uniquely determined by its encoding; we omit the argument.¹⁸

By construction, the expected codeword length satisfies

$$\begin{aligned} \mathbf{E}_{x \sim \mathcal{D}} [|\text{Enc}(x)|] &= \sum_{x \in \text{Support}(\mathcal{D})} \mathcal{D}(x) \left\lceil \log \frac{1}{\mathcal{D}(x)} \right\rceil \\ &< \sum_{x \in \text{Support}(\mathcal{D})} \mathcal{D}(x) \left(\log \frac{1}{\mathcal{D}(x)} + 1 \right) \\ &= H(\mathcal{D}) + 1. \end{aligned}$$

Notably, Shannon also established the converse, showing that entropy gives a lower bound on the expected codeword length of any reasonable coding. In particular, every uniquely decodable¹⁹ binary encoding $C: \text{Support}(\mathcal{D}) \rightarrow \{0, 1\}^*$ satisfies

$$\mathbf{E}_{x \sim \mathcal{D}} [|C(x)|] \geq H(\mathcal{D}).$$

¹⁸For an introduction to data compression, we refer to [Ble13].

¹⁹An encoding is uniquely decodable if every finite concatenation of codewords admits a unique decomposition into codewords. Shannon did not state his result using this notion; the corresponding formulation for uniquely decodable codes was later established by McMillan [McM56].

In this sense, Shannon’s information theory *solves* the compression problem, provided that the source distribution is known to both the encoder and the decoder.

For data arising in the real world, however, the source distribution is rarely given explicitly. For instance, we encounter many different kinds of data in everyday life, including text, music, and images. To apply Shannon’s construction to all such data using a single encoder, we would first have to specify a distribution over the possible inputs. But can we identify a natural distribution that captures the diverse forms of data generated by human activity? Even in a much narrower setting, such as compressing an email written by a professor, the resulting code may differ substantially depending on whether we model the message as drawn from a distribution over all human-written text or from one restricted to academic writing. How, then, should we specify the underlying source distribution? Should a description of the distribution itself be included as part of the encoding?

6.1.1 Universal Coding and Kolmogorov Complexity as Its Ultimate Goal

The problem of compressing strings without knowing the underlying source distribution is known as *universal coding* and dates back to the early works [Kol65, Fit66, Dav73]. For instance, Lempel–Ziv algorithms are prominent examples [ZL77, ZL78]. For every stationary ergodic source (roughly, a source whose statistical behaviour is stable over time and can be inferred from a single sufficiently long sample sequence), their amortized code length converges to the entropy rate without requiring an explicit description of the source. Such guarantees are source-universal but do not capture every simple individual sequence. For example, consider the infinite sequence obtained by concatenating the binary representations

$$0, 1, 10, 11, 100, \dots$$

of the nonnegative integers. The sequence has a short generating program, but LZ78 does not asymptotically compress it.

Kolmogorov complexity and its time-bounded variants, which are central notions in meta-complexity, can be seen as the ultimate goal of universal coding. Recall that the Kolmogorov complexity $K(x)$ of a string x is the length of the shortest program that outputs x and thus measures the length of its shortest algorithmic description without presupposing any source distribution. In this sense, $K(x)$ represents the ideal compression length of x in the absence of computational constraints. Its time-bounded variant $K^t(x)$ additionally requires the program to output x within t steps and thus provides the corresponding ideal when the decoding time is also constrained.

6.1.2 Universal Coding and Learning

We now relate universal coding to distribution learning. Both problems concern an unknown source distribution: universal coding seeks to compress its outputs without knowing the distribution in advance, whereas distribution learning uses samples to approximate it.

Learning $\Rightarrow$ Universal Coding. In one direction, suppose that the unknown source distribution is $\mathcal{D}$ over $\{0, 1\}^n$, and that we have a distribution learner for a class containing $\mathcal{D}$.

Let us first consider the case in which the learning guarantee is measured in KL divergence. Given samples from $\mathcal{D}$, suppose that the learner outputs a distribution $\mathcal{E}$ satisfying $\text{KL}(\mathcal{D} \parallel \mathcal{E}) \leq \epsilon$. We can then use $\mathcal{E}$ as the coding distribution in place of $\mathcal{D}$. A Shannon code based on $\mathcal{E}$ assigns each string x a codeword

of length at most $\log \frac{1}{\mathcal{E}(x)} + 1$. Ignoring this standard one-bit rounding overhead, the excess expected code length over the entropy of $\mathcal{D}$ is exactly the KL divergence:

$$\mathbf{E}_{x \sim \mathcal{D}} \left[\log \frac{1}{\mathcal{E}(x)} \right] - H(\mathcal{D}) = \text{KL}(\mathcal{D} \parallel \mathcal{E}) \leq \epsilon.$$

Thus, distribution learning in KL divergence immediately yields a code whose expected length is close to the optimal expected length for $\mathcal{D}$.

Now consider the case in which the learning guarantee is measured in total variation distance, as in [Definition 2.7](#). Suppose that, given samples from $\mathcal{D}$, the learner outputs a distribution $\mathcal{E}$ satisfying $\Delta_{\text{TV}}(\mathcal{D}, \mathcal{E}) \leq \epsilon$. Again, we would like to use $\mathcal{E}$ in place of $\mathcal{D}$ when constructing a Shannon code. However, closeness in total variation distance alone does not imply any bound on $\text{KL}(\mathcal{D} \parallel \mathcal{E})$, which is precisely the expected excess ideal code length incurred by using $\mathcal{E}$ instead of $\mathcal{D}$. Indeed, the KL divergence can be infinite if $\mathcal{E}(x) = 0$ for some x satisfying $\mathcal{D}(x) > 0$.

The following proposition provides a substitute for such a KL-divergence bound. Intuitively, it shows that, with high probability over $x \sim \mathcal{D}$, the ideal code length assigned by $\mathcal{E}$ is not much larger than that under the true distribution $\mathcal{D}$. We will use it to derive two consequences: a bound on the expected length of a Shannon code and a comparison with Kolmogorov complexity.

Proposition 6.1. *Let $\mathcal{D}$ and $\mathcal{E}$ be distributions over a finite set satisfying $\Delta_{\text{TV}}(\mathcal{D}, \mathcal{E}) \leq \epsilon$. Then, for every $\delta \in (0, 1]$,*

$$\Pr_{x \sim \mathcal{D}} \left[\log \frac{1}{\mathcal{E}(x)} \leq \log \frac{1}{\mathcal{D}(x)} + \log \delta^{-1} \right] \geq 1 - \delta - \epsilon.$$

Proof. Let

$$B \triangleq \{x : \mathcal{E}(x) < \delta \mathcal{D}(x)\}.$$

By the definition of B , we have $\mathcal{E}(B) < \delta \mathcal{D}(B) \leq \delta$. Since total variation distance bounds the difference in probability assigned to any event,

$$\mathcal{D}(B) \leq \mathcal{E}(B) + \Delta_{\text{TV}}(\mathcal{D}, \mathcal{E}) \leq \delta + \epsilon.$$

For every $x \in \bar{B} \cap \text{Support}(\mathcal{D})$, we have $\mathcal{E}(x) \geq \delta \mathcal{D}(x) (> 0)$, which is equivalent to

$$\log \frac{1}{\mathcal{E}(x)} \leq \log \frac{1}{\mathcal{D}(x)} + \log \delta^{-1}.$$

This completes the proof of the proposition. $\square$

As a first consequence, we show that the expected length of a code based on $\mathcal{E}$ is close to the entropy of $\mathcal{D}$. In constructing the code, we mix $\mathcal{E}$ with the uniform distribution to avoid zero probabilities, which could otherwise make the KL divergence infinite.

Corollary 6.2. *Let $\mathcal{D}$ and $\mathcal{E}$ be distributions over $\{0, 1\}^n$ satisfying $\Delta_{\text{TV}}(\mathcal{D}, \mathcal{E}) \leq \epsilon$. Define*

$$\bar{\mathcal{E}} \triangleq \frac{1}{2} \mathcal{E} + \frac{1}{2} \mathcal{U}_n.$$

Then,

$$\mathbf{E}_{x \sim \mathcal{D}} \left[\log \frac{1}{\bar{\mathcal{E}}(x)} \right] \leq H(\mathcal{D}) + \epsilon n + O(1),$$

where $O(1)$ is a universal constant.

Proof. For each $x \in \text{Support}(\mathcal{D})$, let $Z(x) \triangleq \log(1/\bar{\mathcal{E}}(x)) - \log(1/\mathcal{D}(x))$ and $Z_+(x) \triangleq \max\{Z(x), 0\}$. Since $\bar{\mathcal{E}}(x) \geq 2^{-n-1}$, we have $Z(x) \leq n + 1$. Moreover, since $\bar{\mathcal{E}}(x) \geq \mathcal{E}(x)/2$, for every integer $k \geq 0$,

$$\Pr_{x \sim \mathcal{D}}[Z(x) > k + 1] \leq \Pr_{x \sim \mathcal{D}} \left[\log \frac{1}{\mathcal{E}(x)} > \log \frac{1}{\mathcal{D}(x)} + k \right] \leq 2^{-k} + \epsilon,$$

where the last inequality follows from [Proposition 6.1](#) with $\delta = 2^{-k}$. Summing these tail bounds gives

$$\mathbf{E}_{x \sim \mathcal{D}}[Z_+(x)] \leq 1 + \sum_{k=0}^n (2^{-k} + \epsilon) \leq \epsilon n + O(1).$$

Since $\mathbf{E}_{x \sim \mathcal{D}}[\log(1/\bar{\mathcal{E}}(x))] = H(\mathcal{D}) + \mathbf{E}_{x \sim \mathcal{D}}[Z(x)]$, the result follows. $\square$

As a second consequence, we show that, with high probability, the ideal code length assigned by $\mathcal{E}$ is close to the Kolmogorov complexity, i.e., its idealized description length.

Corollary 6.3. *Let $\mathcal{D}$ and $\mathcal{E}$ be distributions over $\{0, 1\}^n$ satisfying $\Delta_{\text{TV}}(\mathcal{D}, \mathcal{E}) \leq \epsilon$. Then, for every $\delta \in (0, 1]$,*

$$\Pr_{x \sim \mathcal{D}} \left[\log \frac{1}{\mathcal{E}(x)} \leq K(x) + O(\log n + \log \delta^{-1}) \right] \geq 1 - \delta - \epsilon.$$

Proof. For every $s > 0$,

$$\Pr_{x \sim \mathcal{D}} \left[\log \frac{1}{\mathcal{D}(x)} > K(x) + s \right] = \sum_{\substack{x \in \{0, 1\}^n: \\ \mathcal{D}(x) < 2^{-K(x)-s}}} \mathcal{D}(x) \leq 2^{-s} \sum_{x \in \{0, 1\}^n} 2^{-K(x)}.$$

For each $k \geq 0$, by counting the number of programs of length k , there are at most 2^k strings of Kolmogorov complexity k . Moreover, for any string x of length n , $K(x) \leq n + O(\log n)$. Therefore, the expression above is upper-bounded by

$$2^{-s} \cdot \sum_{k=0}^{n+O(\log n)} \sum_{\substack{x \in \{0, 1\}^n: \\ K(x)=k}} 2^{-k} = 2^{-s} \cdot \sum_{k=0}^{n+O(\log n)} O(1) = 2^{-s} \cdot O(n).$$

Thus, by choosing $s = O(\log n + \log \delta^{-1})$, we obtain $\log(1/\mathcal{D}(x)) \leq K(x) + O(\log n + \log \delta^{-1})$ except with probability at most $\delta/2$.

By [Proposition 6.1](#) with parameter $\delta/2$, we also have

$$\log \frac{1}{\mathcal{E}(x)} \leq \log \frac{1}{\mathcal{D}(x)} + \log \frac{2}{\delta}$$

except with probability at most $\delta/2 + \epsilon$. Combining the two inequalities and taking a union bound proves the claim. $\square$

In particular, if $\mathcal{E}$ admits a short description, then incorporating this description into the code yields, for most $x \sim \mathcal{D}$, an encoding of x whose length is close to optimal.

Universal Coding $\Rightarrow$ Learning. Conversely, any code $C: \mathcal{X} \rightarrow \{0, 1\}^*$ for an unknown source distribution $\mathcal{D}$ over $\mathcal{X}$, with code-length function $c(x) \triangleq |C(x)|$, naturally induces a distribution $\mathcal{D}_c$ by normalizing $2^{-c(x)}$:

$$\mathcal{D}_c(x) \triangleq \frac{2^{-c(x)}}{Z_c}, \quad Z_c \triangleq \sum_{y \in \mathcal{X}} 2^{-c(y)}. \quad (4)$$

This construction is natural since the ideal code length $c(x) = -\log \mathcal{D}(x)$, with the convention $-\log 0 = \infty$, yields $\mathcal{D}_c = \mathcal{D}$. It is therefore natural to ask how close $\mathcal{D}_c$ is to the true distribution $\mathcal{D}$ when $c(x)$ only approximates the ideal code length.

The following proposition answers this question. It shows that the KL divergence from $\mathcal{D}$ to $\mathcal{D}_c$ is determined by the excess expected code length over the entropy of $\mathcal{D}$, together with the normalization term $\log Z_c$.

Proposition 6.4. *Let $\mathcal{D}$ be a distribution over $\mathcal{X}$, and let $C: \mathcal{X} \rightarrow \{0, 1\}^*$ be a code with code-length function $c(x) \triangleq |C(x)|$. Then*

$$\text{KL}(\mathcal{D} \parallel \mathcal{D}_c) = \left(\mathbf{E}_{x \sim \mathcal{D}}[c(x)] - H(\mathcal{D}) \right) + \log Z_c.$$

Proof. By the definition of $\mathcal{D}_c$, for every $x \in \text{Support}(\mathcal{D})$,

$$\log \frac{1}{\mathcal{D}_c(x)} = \log \frac{Z_c}{2^{-c(x)}} = c(x) + \log Z_c.$$

Therefore,

$$\begin{aligned} \text{KL}(\mathcal{D} \parallel \mathcal{D}_c) &= \mathbf{E}_{x \sim \mathcal{D}} \left[\log \frac{\mathcal{D}(x)}{\mathcal{D}_c(x)} \right] \\ &= \mathbf{E}_{x \sim \mathcal{D}} \left[\log \frac{1}{\mathcal{D}_c(x)} \right] - H(\mathcal{D}) \\ &= \mathbf{E}_{x \sim \mathcal{D}}[c(x)] - H(\mathcal{D}) + \log Z_c. \end{aligned}$$

□

Consequently, if $\mathbf{E}_{x \sim \mathcal{D}}[|C(x)|] \leq H(\mathcal{D}) + \epsilon$, then

$$\text{KL}(\mathcal{D} \parallel \mathcal{D}_c) \leq \epsilon + \log Z_c.$$

Moreover, if C is a prefix-free code on $\mathcal{X}$, meaning that no codeword is a prefix of another, then Kraft's inequality gives

$$Z_c = \sum_{y \in \mathcal{X}} 2^{-|C(y)|} \leq 1.$$

Thus, $\log Z_c \leq 0$, and hence

$$\text{KL}(\mathcal{D} \parallel \mathcal{D}_c) \leq \epsilon.$$

Applying Pinsker's inequality, we further obtain the total variation bound

$$\Delta_{\text{TV}}(\mathcal{D}, \mathcal{D}_c) \leq O(\sqrt{\epsilon}).$$

Although the preceding proposition gives a direct transformation from universal coding to distribution learning, its hypothesis may appear somewhat stringent. The entropy $H(\mathcal{D})$ is the minimum expected code

length attainable when the source distribution $\mathcal{D}$ is known, whereas a universal code must be constructed without knowledge of $\mathcal{D}$. It is therefore natural to ask what can be concluded when $c(\cdot)$ instead approximates the Kolmogorov complexity $K(\cdot)$, viewed as an idealized universal code length.

The following proposition answers this question. It gives a similar bound at the cost of an additive term $|\mathcal{D}|$, where $|\mathcal{D}|$ denotes the length of the shortest description of a probabilistic program that samples exactly from $\mathcal{D}$ (with no restriction on its running time).

Proposition 6.5. *Let $\mathcal{D}$ be a distribution over a finite set $\mathcal{X}$, and let $|\mathcal{D}|$ denote the length of the shortest description of a probabilistic program that samples exactly from $\mathcal{D}$. Let $C: \mathcal{X} \rightarrow \{0, 1\}^*$ be a code, and let $c(x) \triangleq |C(x)|$. Then*

$$\text{KL}(\mathcal{D} \parallel \mathcal{D}_c) \leq \mathbf{E}_{x \sim \mathcal{D}} [c(x) - K(x)] + |\mathcal{D}| + \log Z_c + O(1).$$

Proof. By the coding theorem,

$$K(x) \leq \left\lceil \log \frac{1}{\mathcal{D}(x)} \right\rceil + |\mathcal{D}| + O(1)$$

for every $x \in \text{Support}(\mathcal{D})$. Therefore,

$$\mathbf{E}_{x \sim \mathcal{D}} [K(x)] \leq H(\mathcal{D}) + |\mathcal{D}| + O(1).$$

From [Proposition 6.4](#), we obtain

$$\begin{aligned} \text{KL}(\mathcal{D} \parallel \mathcal{D}_c) &= \mathbf{E}_{x \sim \mathcal{D}} [c(x)] - H(\mathcal{D}) + \log Z_c \\ &= \mathbf{E}_{x \sim \mathcal{D}} [c(x) - K(x)] + \mathbf{E}_{x \sim \mathcal{D}} [K(x)] - H(\mathcal{D}) + \log Z_c \\ &\leq \mathbf{E}_{x \sim \mathcal{D}} [c(x) - K(x)] + |\mathcal{D}| + \log Z_c + O(1). \end{aligned}$$

□

Again, if C is prefix-free and $\mathbf{E}_{x \sim \mathcal{D}} [c(x) - K(x)] \leq \epsilon$, then

$$\text{KL}(\mathcal{D} \parallel \mathcal{D}_c) \leq |\mathcal{D}| + \epsilon + O(1).$$

However, since $|\mathcal{D}| \geq 1$ this bound does not yield a small total variation distance via Pinsker's inequality.

One might hope to make the contribution of $|\mathcal{D}|$ negligible so that Pinsker's inequality yields a small total variation distance. This can be achieved by compressing a batch of independent samples and thereby amortizing $|\mathcal{D}|$ over the batch. Suppose that a prefix-free code $C_m: \mathcal{X}^m \rightarrow \{0, 1\}^*$ satisfies

$$\mathbf{E}_{\bar{x} \sim \mathcal{D}^m} [|C_m(\bar{x})| - K(\bar{x})] \leq \epsilon_m,$$

and let $\mathcal{D}_{c_m}$ be the distribution induced by the length function $c_m(\bar{x}) \triangleq |C_m(\bar{x})|$.

An exact sampler for $\mathcal{D}^m$ simply runs an exact sampler for $\mathcal{D}$ independently m times. It therefore has description length

$$|\mathcal{D}^m| \leq |\mathcal{D}| + O(\log m).$$

Consequently, [Proposition 6.5](#) gives

$$\text{KL}(\mathcal{D}^m \parallel \mathcal{D}_{c_m}) \leq |\mathcal{D}| + \epsilon_m + O(\log m).$$

To turn this approximation of $\mathcal{D}^m$ into an approximation of $\mathcal{D}$, choose I uniformly at random from $[m]$, draw $X_{<I} \sim \mathcal{D}^{I-1}$, and let $\hat{\mathcal{D}}$ be the conditional distribution of the I -th coordinate under $\mathcal{D}_{C_m}$, conditioned on $I - 1$ independent samples from $\mathcal{D}$.

By the chain rule for KL divergence (Lemma 2.4),

$$\mathbf{E}_{I, X_{<I}} [\text{KL}(\mathcal{D} \parallel \hat{\mathcal{D}})] \leq \frac{|\mathcal{D}| + \epsilon_m + O(\log m)}{m}.$$

Here, the expectation is taken over both the uniformly random index I and the prefix $X_{<I} \sim \mathcal{D}^{I-1}$. Thus, provided that $\epsilon_m = o(m)$, the expected KL divergence tends to zero as $m \rightarrow \infty$. Pinsker's inequality and Jensen's inequality then imply

$$\begin{aligned} \mathbf{E}_{I, X_{<I}} [\Delta_{\text{TV}}(\mathcal{D}, \hat{\mathcal{D}})] &\leq \sqrt{\frac{1}{2} \mathbf{E}_{I, X_{<I}} [\text{KL}(\mathcal{D} \parallel \hat{\mathcal{D}})]} \\ &\leq \sqrt{\frac{1}{2} \cdot \frac{|\mathcal{D}| + \epsilon_m + O(\log m)}{m}} \xrightarrow{m \rightarrow \infty} 0. \end{aligned}$$

Conclusion. Taken together, these elementary arguments give information-theoretic transformations in both directions. A distribution learner yields a code whose expected length is close to the source entropy, while a code approximating Kolmogorov complexity can be converted into a learner for distributions having short sampler descriptions. These transformations do not, however, address computational efficiency. Kolmogorov complexity is uncomputable, and evaluating or sampling from the conditional distribution $\hat{\mathcal{D}}$ need not be efficient. This leads to the corresponding complexity-theoretic question: for efficiently samplable distributions, how is efficient learning related to efficient compression near time-bounded Kolmogorov complexity? Efficient learning belongs to computational learning theory, whereas efficiently approximating time-bounded Kolmogorov complexity is a central problem in meta-complexity.

6.2 Computational Learning and Meta-Complexity

As we saw in Section 6.1, distribution learning is closely related to compression, particularly universal coding, at the information-theoretic level. The relationship between efficient distribution learning and meta-complexity can be viewed as a complexity-theoretic counterpart of this connection. The natural question is how much of this connection survives when the transformations between learning and compression must be efficient and code lengths are measured against *polynomial-time-bounded* Kolmogorov complexity.

More specifically, we consider both directions of this question. Can an efficient learner be converted into an efficient encoder that produces short time-bounded descriptions of samples from the target distribution? Conversely, can the ability to find short time-bounded descriptions be converted into an efficient learner? The first direction requires turning an efficiently samplable distribution into a code, whereas the reverse direction requires turning description lengths into a probability distribution that can be efficiently sampled from and, in some settings, conditioned. The purpose of this section is to survey how these two transformations are connected to meta-complexity.

For simplicity, we informally use K^{poly} to denote K^t for an unspecified polynomial time bound t , and use $\text{r}K^{\text{poly}}$ and q^{poly} analogously. We leave the particular polynomials implicit and do not track polynomial overheads in the time bounds.

6.2.1 Learning to Compression via Arithmetic Coding

In the learning-to-compression transformation from [Section 6.1.2](#), the learner produces a hypothesis distribution $\mathcal{E}$. Applying Shannon coding to the smoothed distribution $\tilde{\mathcal{E}}$ from [Corollary 6.2](#) yields an expected description length close to the entropy of the source. This argument, however, is information-theoretic. The Shannon code described in [Section 6.1](#) first orders all messages and then encodes each message using the cumulative probabilities of the preceding messages. For the message space $\{0, 1\}^n$, carrying out this construction directly may require examining exponentially many strings. Moreover, a distribution learner may output only a sampler for $\mathcal{E}$, rather than an algorithm that evaluates $\mathcal{E}(x)$. Given a sampling circuit for $\mathcal{E}$, computing $\mathcal{E}(x)$ amounts to counting the random seeds on which the circuit outputs x , and a brute-force computation may take exponential time. Thus, the information-theoretic argument alone does not yield a polynomial-time encoder and decoder. Indeed, this computational gap is inherent for general efficiently samplable sources: under the plausible assumption that $\text{SZK} \neq \text{BPP}$, some such sources cannot be compressed close to their entropy by any polynomial-time encoder [[TVZ05](#)].

One way to avoid this global enumeration is to require the learner to output a *next-bit predictor*, rather than merely a sampler, and then apply *arithmetic coding*. For a distribution $\mathcal{D}$ over $\{0, 1\}^n$, write

$$\mathcal{D}(b \mid x_{<i}) \triangleq \Pr_{X \sim \mathcal{D}}[X_i = b \mid X_{<i} = x_{<i}].$$

A next-bit predictor approximates these conditional probabilities to inverse-polynomial additive accuracy. This requirement is a quantitative strengthening of Yao’s next-bit prediction criterion, which asks only for a non-negligible advantage over random guessing in predicting the next bit at a uniformly random position from the preceding prefix (see [Section 5.1.1](#)). Given a next-bit predictor satisfying this stronger quantitative guarantee, arithmetic coding accesses the learned distribution one bit at a time and constructs the desired encoder and decoder without enumerating its support.

We briefly review how arithmetic coding works. At a high level, it reconstructs the cumulative-probability interval associated with a message using only next-bit conditional probabilities. The procedure can be viewed as repeatedly zooming in on the probability interval $[0, 1)$. Initially, the encoder sets $s = 0$ and $t = 1$, so that the current interval is $[s, t) = [0, 1)$. After reading a prefix $x_{<i}$, it computes the splitting point

$$u \triangleq s + (t - s)\mathcal{D}(0 \mid x_{<i}).$$

The encoder then reads the next bit x_i . If $x_i = 0$, it updates $t \leftarrow u$ and retains the interval $[s, u)$. If $x_i = 1$, it updates $s \leftarrow u$ and retains the interval $[u, t)$. In either case, the length of the current interval is multiplied by $\mathcal{D}(x_i \mid x_{<i})$. Consequently, after reading the last bit, the retained interval has length

$$\prod_{i=1}^n \mathcal{D}(x_i \mid x_{<i}) = \mathcal{D}(x).$$

The encoder then chooses a binary prefix whose corresponding dyadic interval is contained in the final interval. Such a prefix can be chosen to have length

$$\log \frac{1}{\mathcal{D}(x)} + O(1).$$

The decoder reverses this procedure. Given the codeword and access to the same next-bit predictor, it interprets the codeword as specifying a dyadic interval. At each step, it partitions the current interval according

to the predicted conditional probability and determines which of the two resulting subintervals contains the dyadic interval, thereby recovering the next bit.

Thus, encoding and decoding require only the n conditional probabilities along the path determined by x , rather than an enumeration of the entire support. They can therefore be performed in polynomial time whenever the next-bit predictor is efficient. Moreover, decoding remains correct as long as the encoder and decoder obtain the same conditional probabilities on every query; the accuracy of these probabilities affects only the resulting code length.

If the next-bit predictor is randomized, its random coins must be fixed in advance and shared by the encoder and decoder so that they obtain identical answers to the same queries. Thus, if the learner outputs the hypothesis distribution $\mathcal{E}$ in the form of a randomized next-bit predictor, then, given shared access to this predictor, the construction yields a pK^{poly} -type compressor (namely, the resulting program may depend on the shared random string) whose description length is close to $\log(1/\mathcal{E}(x))$, up to the loss incurred by approximating the conditional probabilities.

One may ask how much stronger it is to require a next-bit predictor rather than merely a sampler. In general, there is a gap if AIOWFs exist: if G_z is an AIPRG constructed from an AIOWF and $\mathcal{U}$ is uniform, then $(G_z(\mathcal{U}), \mathcal{U})$ is efficiently samplable but admits no efficient next-bit predictor. Conversely, if AIOWFs do not exist (as is implied by efficient distribution learning for P/poly-samplable distributions; see [Lemma 4.8](#)), the distributional-inversion argument of Impagliazzo and Luby [IL89] can be used to construct an efficient next-bit predictor from an explicit description of any efficient sampler. Essentially the same argument appears in our discussion of white-box learning in [Section 4.5.2](#).

Application: A Coding Theorem for rK^{poly} . The arithmetic-coding construction above has an application in meta-complexity: a coding theorem for randomized time-bounded Kolmogorov complexity. Recall that $\text{rK}^{\text{poly}}(x)$ is the minimum length of a randomized program that outputs x in polynomial time with probability at least $2/3$. Although this is a natural way to incorporate randomness into time-bounded Kolmogorov complexity, no optimal coding theorem is known for rK^{poly} over all efficiently samplable distributions, in contrast to pK^{poly} . Hirahara, Lu, and Nanashima [HLN24] showed that every next-bits-predictable distribution (i.e., every distribution with a randomized next-bit predictor) admits an efficient coding theorem for rK^{poly} , with arithmetic coding as a central ingredient.

The main obstacle to obtaining rK^{poly} -type compression is ensuring that the encoder and decoder reconstruct exactly the same arithmetic-coding intervals. In the rK setting, however, they cannot share random coins, because the encoder must fix the description before the decoder's random coins are drawn. Independent executions of the randomized next-bit predictor may produce slightly different estimates of the same conditional probability, and even a small discrepancy can lead the encoder and decoder to use different interval boundaries, causing decoding to fail. To prevent this, the construction uses the “add noise and round” technique of Goldberg and Sipser [GS91] and Trevisan, Vadhan, and Zuckerman [TVZ05], which adds a small random shift to each estimate and rounds the result to a sufficiently coarse grid. With high probability over the shift, independent accurate executions of the predictor produce the same rounded estimates. Thus, only a logarithmic-length description of the shift, rather than the entire randomness of the predictor, needs to be included in the code. Consequently, if $\mathcal{D} = \{\mathcal{D}_n\}$ is next-bits predictable, then, for every constant $\varepsilon > 0$, there is a polynomial time bound for which

$$\text{rK}^{\text{poly}}(x) \leq (1 + \varepsilon) \log \frac{1}{\mathcal{D}_n(x)} + O(\log n)$$

for every $x \in \text{Support}(\mathcal{D}_n)$.

Here, the factor $1 + \varepsilon$ arises from next bits having very small conditional probabilities. For such a bit, the rounding error may be too large relative to its true probability, so the construction records the bit and its position explicitly rather than encoding it through the estimated probability. The cost of these explicit records accounts for the multiplicative loss in the worst-case bound. For a sample $x \sim \mathcal{D}_n$, however, such low-probability next bits occur only with correspondingly small probability. Their contribution can therefore be absorbed into the additive overhead on average, removing the multiplicative $1 + \varepsilon$ loss. Consequently, for every polynomial q , there exists a polynomial p such that

$$\Pr_{x \sim \mathcal{D}_n} \left[rK^{p(n)}(x) \leq \log \frac{1}{\mathcal{D}_n(x)} + \log p(n) \right] \geq 1 - \frac{1}{q(n)}.$$

The construction also gives efficient encoding and decoding algorithms whose expected encoding length is at most $H(\mathcal{D}_n) + O(\log n)$.

Combining this average-case theorem with the distributional-inversion argument mentioned above yields an average-case coding theorem for rK^{poly} over arbitrary efficiently samplable distributions under the nonexistence of one-way functions. Such coding theorems relate $rK^{\text{poly}}(x)$ to both the statistical information content $\log(1/\mathcal{D}_n(x))$ and the unbounded Kolmogorov complexity $K(x)$. This connection allows characterizations of one-way functions previously formulated using pK^{poly} to be extended to rK^{poly} .

6.2.2 Compression to Learning over Polynomially Many Messages

We now turn to the reverse direction: transforming an *efficient* compressor, ideally one whose code lengths approximate time-bounded Kolmogorov complexity, into an *efficient* learner. Recall that in [Section 6.1.2](#), any code-length function c was converted into a distribution $\mathcal{D}_c$ whose probability mass at each string x is proportional to $2^{-c(x)}$.

Observe that the KL-divergence bound in [Proposition 6.5](#) uses no particular property of $K(\cdot)$ beyond its coding theorem: for a source distribution $\mathcal{D}$,

$$K(x) \lesssim \log \frac{1}{\mathcal{D}(x)}.$$

Consequently, the same argument applies when $c(\cdot)$ approximates a complexity measure other than time-unbounded Kolmogorov complexity, provided that the measure satisfies an analogous coding theorem for the source distributions under consideration. In particular, when $\mathcal{D}$ is efficiently samplable and $c(x) \approx pK^{\text{poly}}(x)$, the coding theorem for pK^{poly} yields the same KL-divergence bound for the induced distribution $\mathcal{D}_c$. Thus, if we disregard the efficiency of constructing and sampling from $\mathcal{D}_c$, a compressor approximating pK^{poly} already gives an information-theoretic transformation into a distribution learner for efficiently samplable distributions. The remaining question is whether this transformation itself can be implemented efficiently.

From a complexity-theoretic perspective, the immediate obstacle is efficient sampling. Even if each value $c(u)$ can be evaluated efficiently, computing the normalization constant and sampling from $\mathcal{D}_c$ may require enumerating exponentially many possible values of u . Indeed, overcoming this sampling problem is one of the central obstacles to deriving efficient learners from meta-complexity assumptions.

One simple way to set aside this obstacle is to consider the case in which each outcome has logarithmic length, so that the sample space has polynomial size. In this case, both the normalization constant and a sampler for $\mathcal{D}_c$ can be computed by direct enumeration. We therefore begin with this setting. Taken by itself, however, this setting is not particularly interesting from the perspective of distribution learning, since

an arbitrary distribution over a polynomial-size domain can already be learned efficiently by estimating the probability of every possible outcome from samples.

Nevertheless, generating an output from a polynomial-size alphabet becomes a nontrivial and well-motivated learning problem in the conditional setting. Suppose that the learner receives i.i.d. samples from a joint distribution $\mathcal{D}$ over pairs (X, Y) , where X is an observed input or context and Y is the output to be predicted. We allow X to have length n , whereas Y has length $O(\log n)$, so its range $\mathcal{Y}$ has polynomial size. This setting is well motivated. Rather than generating an entire pair consisting of an image and its label, we may wish to estimate the probability of each possible label for a given image. Similarly, we may wish to generate an answer conditioned on a given question, rather than generating a pair consisting of a question and an answer. In addition, an output of arbitrary length can be divided into logarithmic-length blocks and generated sequentially. We return to this idea in [Section 6.2.3](#).

Accordingly, in the remainder of this subsection, we focus on the following conditional learning task. Given i.i.d. samples from $\mathcal{D}$ and a new input x , the learner aims to construct a hypothesis distribution approximating $\mathcal{D}(\cdot | x)$, where

$$\mathcal{D}(y | x) \triangleq \Pr_{(X,Y) \sim \mathcal{D}}[Y = y | X = x].$$

Since the range of X may be exponentially large, this task does not reduce to estimating the probability of every pair (x, y) . By contrast, since $\mathcal{Y}$ has polynomial size, for any fixed x , once a weight can be evaluated efficiently for every $y \in \mathcal{Y}$, the learner can normalize these weights and sample from them by direct enumeration. The remaining question is which description length should be used to define these conditional weights. Below, we discuss two natural candidates.

The first is the conditional description length $c(y | x)$, where x is treated as auxiliary input available to both the encoder and the decoder. This gives the hypothesis distribution

$$\hat{\mathcal{D}}_c^{\text{cond}}(y | x) \triangleq \frac{2^{-c(y|x)}}{\sum_{z \in \mathcal{Y}} 2^{-c(z|x)}}.$$

This construction requires access to a good conditional compressor, which can be substantially more demanding than unconditional compression. Even in the idealized setting, it is currently unclear whether efficient computation of $K^{\text{poly}}(\cdot)$ implies efficient computation of $K^{\text{poly}}(\cdot | \cdot)$, whereas the converse is immediate by taking the auxiliary input to be the empty string. Thus, even given an efficient unconditional compressor that closely approximates $K^{\text{poly}}(\cdot)$, this construction itself does not yield an efficiently computable conditional hypothesis distribution.

The second candidate requires only an unconditional description-length measure. It uses the increase in description length caused by appending y to x :

$$\hat{\mathcal{D}}_c^{\text{inc}}(y | x) \triangleq \frac{2^{-(c(xy) - c(x))}}{\sum_{z \in \mathcal{Y}} 2^{-(c(xz) - c(x))}} = \frac{2^{-c(xy)}}{\sum_{z \in \mathcal{Y}} 2^{-c(xz)}}.$$

This construction has a direct interpretation in terms of the distribution $\mathcal{D}_c$ defined in [Equation \(4\)](#):

$$\hat{\mathcal{D}}_c^{\text{inc}}(y | x) = \mathcal{D}_c(y | x).$$

Thus, whenever $\mathcal{D}_c$ is close to $\mathcal{D}$ in KL divergence, the chain rule for KL divergence ([Lemma 2.4](#)) implies

that $\widehat{\mathcal{D}}_c^{\text{inc}}(\cdot | x)$ is close to $\mathcal{D}(\cdot | x)$ in KL divergence on average over $x \sim \mathcal{D}_X$:

$$\begin{aligned} \mathbf{E}_{x \sim \mathcal{D}_X} \left[\text{KL} \left(\mathcal{D}(\cdot | x) \parallel \widehat{\mathcal{D}}_c^{\text{inc}}(\cdot | x) \right) \right] &= \mathbf{E}_{x \sim \mathcal{D}_X} [\text{KL}(\mathcal{D}(\cdot | x) \parallel \mathcal{D}_c(\cdot | x))] \\ &= \text{KL}(\mathcal{D} \parallel \mathcal{D}_c) - \text{KL}(\mathcal{D}_X \parallel (\mathcal{D}_c)_X) \\ &\leq \text{KL}(\mathcal{D} \parallel \mathcal{D}_c), \end{aligned}$$

where $\mathcal{D}_X$ and $(\mathcal{D}_c)_X$ denote the respective marginals of $\mathcal{D}$ and $\mathcal{D}_c$ on $\mathcal{X}$. Since $\mathcal{Y}$ has polynomial size, if $c(\cdot)$ is efficiently computable and polynomially bounded, then $\widehat{\mathcal{D}}_c^{\text{inc}}(\cdot | x)$ can be efficiently computed and sampled from by enumerating all $y \in \mathcal{Y}$.

The preceding argument is sufficient when the induced joint distribution $\mathcal{D}_c$ is close to $\mathcal{D}$. However, requiring a good approximation to the entire joint distribution can be stronger than necessary for conditional learning, which aims only to approximate $\mathcal{D}(\cdot | x)$ and need not model the marginal distribution of X . It is therefore natural to ask whether the desired conditional guarantee can be obtained directly, without first approximating the entire joint distribution. Deriving such a conditional learner from meta-complexity assumptions without implicitly learning the marginal distribution of the conditioning variable remains a challenging problem. In meta-complexity, this question is closely connected to two notions, *conditional coding* and *symmetry of information*, which we discuss at a conceptual level below.

Conditional coding is the conditional counterpart of the coding theorem discussed above. Informally, it asks whether a sample y drawn from $\mathcal{D}(\cdot | x)$ admits a K^{poly} -description of length close to the conditional information content $\log(1/\mathcal{D}(y | x))$, with x available as an auxiliary input. More generally, the same question can be formulated for any conditional description-length measure. For the learning application, an average-case guarantee is sufficient, since the learner is evaluated on a fresh pair $(x, y) \sim \mathcal{D}$. If such a guarantee is available for an efficiently computable conditional description-length measure, the same argument as [Proposition 6.5](#) can be applied conditionally to analyze $\widehat{\mathcal{D}}_c^{\text{cond}}$. For time-unbounded Kolmogorov complexity K , conditional coding follows immediately from the ordinary coding theorem, since the conditional distribution of a samplable joint distribution is itself samplable when no running-time bound is imposed. By contrast, the validity of its time-bounded analogues is closely connected to complexity-theoretic and cryptographic assumptions. Consequently, conditional coding and the complexity of time-bounded conditional descriptions have received considerable attention in meta-complexity [[HIL⁺23](#), [LS24](#), [HLN24](#), [HLO26b](#), [HMO26](#)].

Symmetry of information connects the second construction, which uses only unconditional description lengths, to the first, which describes y with x available as auxiliary input. Roughly speaking, it asks whether $K^{\text{poly}}(xy)$ can be decomposed as $K^{\text{poly}}(x) + K^{\text{poly}}(y | x)$ up to a small additive loss. The analogous question can also be posed for other description-length measures. If such a decomposition holds, the incremental description length $K^{\text{poly}}(xy) - K^{\text{poly}}(x)$ behaves like the conditional description length $K^{\text{poly}}(y | x)$. For time-unbounded Kolmogorov complexity K , symmetry of information holds unconditionally, up to the logarithmic additive loss [[ZL70](#)]. By contrast, whether its time-bounded analogues hold is closely connected to central questions in complexity theory and cryptography, including the P versus NP problem and the existence of one-way functions [e.g., [Lev03](#), [LM93](#), [LW95](#)]. Accordingly, time-bounded forms of symmetry of information, their connections to one-way functions and average-case complexity, and unconditional failures in certain formulations have been studied extensively in the meta-complexity literature [[LR05](#), [Hir21](#), [LP22](#), [GKLO22](#), [GK22](#), [Hir22c](#), [HIL⁺23](#), [HLO24](#), [HLN24](#), [HLO26b](#), [GHL⁺25](#), [HMO26](#)].

We have seen why conditional coding and symmetry of information are natural tools for analyzing the two constructions above. To the best of our knowledge, however, existing results do not directly convert an unconditional compression guarantee over the target distribution into a conditional KL guarantee for the distribution obtained by normalizing the incremental weights $2^{-(c(xy)-c(x))}$, except in settings where the

assumptions under consideration already imply that the context x is also compressed nearly optimally on average under the corresponding marginal distribution. Namely, it remains open whether conditional coding and symmetry of information can instead yield direct conditional-learning guarantees beyond those obtained by approximating the entire joint distribution. In particular, it would be of great interest to establish such a transformation for a restricted class of distributions without relying on implausible assumptions that rule out standard cryptography.

Another closely related result is the KL-sampling characterization of conditional pseudoentropy due to Vadhan and Zheng [VZ12]. For polynomial-size output alphabets, their result characterizes conditional pseudo-Shannon entropy in terms of the minimum average conditional KL divergence achievable by an efficient sampler. This provides another complexity-theoretic link between conditional information and efficient conditional sampling, although it is formulated in terms of pseudoentropy rather than description-length scores. Recent work has revisited and generalized this characterization using multicalibration and weight-restricted calibration [CDV24, HV25].

6.2.3 Compression to Learning over General Messages

For polynomial-length messages, the output space may be exponentially large, making it infeasible to evaluate the score of every possible y and normalize the resulting weights as in Section 6.2.2. We discuss two approaches to this setting. It is worth keeping in mind, however, that the connection between distribution learning and meta-complexity in this time-bounded regime remains largely open.

Universal Extrapolation. The first approach is to use universal extrapolation. Recall that $q^t(x) \triangleq -\log Q^t(x)$, where Q^t is the time-bounded universal distribution. Thus, for the code-length function $c(x) = q^t(x)$, the distribution $\mathcal{D}_c$ defined in Equation (4) coincides exactly with Q^t . The universal distribution satisfies the following domination property. For every distribution $\mathcal{D}$ samplable in time t by a sampler of description length $|\mathcal{D}|$, and every $x \in \text{Support}(\mathcal{D})$,

$$Q^{\text{poly}(t)}(x) \geq 2^{-O(|\mathcal{D}|)} \cdot \mathcal{D}(x)$$

Indeed, the universal sampler can use an initial portion of its random input to specify the sampler for $\mathcal{D}$ and use the remaining bits as its random seed. The probability of selecting this sampler is at least $2^{-O(|\mathcal{D}|)}$, and, conditioned on doing so, the output distribution is exactly $\mathcal{D}$. This bounds the KL divergence from an unknown samplable distribution $\mathcal{D}$ to $Q^t = 2^{-q^t}$ in terms of the description length of a sampler for $\mathcal{D}$, since the domination property immediately implies the following coding-theorem type bound:

$$q^t(x) = -\log Q^t(x) \leq \log \frac{1}{\mathcal{D}(x)} + O(|\mathcal{D}|).$$

The claimed KL-divergence bound then follows by the same argument as in Proposition 6.5. To obtain the total variation bound, recall the batch-amortization argument following Proposition 6.5, in which the description-length term is amortized by conditioning a uniformly random coordinate on the preceding independent samples. Implementing this argument here requires sampling from the time-bounded universal distribution conditioned on an observed prefix of the batch. This conditional-sampling task is exactly what is called universal extrapolation in [HN23], which is equivalent to the task described by Definition 5.4 together with the procedure that follows it, specialized here to the case where no additional advice is given, as required for distribution learning.

Under the nonexistence of infinitely-often one-way functions, universal extrapolation (with no advice) can be performed in time polynomial in the number of bits to be extrapolated, the time bound, the inverse accuracy, and $2^{q^t(x) - K(x)}$ [IL90, HN23]. Its running time is therefore exponential in the computational depth $q^t(x) - K(x)$ of the conditioning prefix x . The slow-growth law for computational depth (see Section 5.2.2) states that efficient randomized computation increases computational depth by at most a logarithmic amount with high probability. Consequently, if the description of a sampler has low computational depth, then the conditioning prefixes generated by that sampler also have low computational depth with high probability. One concrete example is a sampler that is secretly selected by a uniform efficient randomized algorithm. Although the selected sampler may itself be a nonuniform object of polynomial length, its computational depth is logarithmic with high probability. For samplers of logarithmic computational depth, universal extrapolation therefore runs in polynomial time and, combined with the batch-amortization argument following Proposition 6.5, yields the following result without any restriction on the size of the message space.

Theorem 6.6 ([HN23]). *If infinitely-often one-way functions do not exist, then every efficiently samplable distribution represented by a sampler of logarithmic computational depth is distribution learnable in polynomial time (with respect to total variation distance, as defined in Definition 2.7).*

A limitation of this approach is that it requires ruling out one-way functions. Known constructions of universal extrapolation rely on consequences of the nonexistence of one-way functions, such as distributional inversion, and this dependence is unavoidable for general distribution classes because efficient universal extrapolation itself implies the nonexistence of one-way functions. Consequently, even for a restricted class of distributions that might be efficiently learnable unconditionally, this approach still invokes the seemingly implausible assumption that one-way functions do not exist. Thus, universal extrapolation reveals strong connections among learning, cryptography, and meta-complexity, but it remains unclear whether it can yield unconditional learning results for restricted distribution classes that exclude cryptographically hard sources.

Another major open question concerns the complexity of universal extrapolation itself. Existing techniques based on distributional inversion require time exponential in the computational depth of the conditioning prefix. They therefore do not give efficient conditional generation when the prefix has large computational depth.

Problem 6.7. What is the weakest assumption under which universal extrapolation can be performed efficiently for every conditioning prefix? In particular, is there a meta-complexity assumption that characterizes this worst-case form of universal extrapolation?

Sequential Generation. The second approach is to divide a polynomial-length message into polynomially many short blocks and generate them autoregressively. Write a message as $x = x_1 \cdots x_m$, where each block x_i has length $O(\log n)$. After generating the preceding blocks $x_{<i}$, the learner constructs a conditional distribution over the possible values of x_i and samples the next block from this distribution. Since each block has only polynomially many possible values, the required normalization can be performed by direct enumeration, as in Section 6.2.2.

Which compression guarantee is sufficient to construct such a sequential generator? To formulate this question, first observe the easy converse direction: the same conditional distributions can be used to compress the message block by block. Once $x_{<i}$ has been decoded, the conditional distribution associated with this prefix determines a code for x_i through the arithmetic-coding construction reviewed in Section 6.2.1. This form of sequential blockwise coding is known as *prequential compression*. A prequential compressor processes $x_1, \dots, x_m$ online, encoding each block x_i with the preceding blocks $x_{<i}$ available as auxiliary

input. Ideally, its total conditional description length should be comparable to a one-shot time-bounded description of the entire message x , such as $K^{\text{poly}}(x)$ or one of its variants that admits optimal coding, up to an additive $m \cdot O(\log n)$ overhead for encoding the blocks separately.

This objective is formalized by an *algorithmic chain rule*, as studied in [KK25, HN26a]. For q^{poly} , suppressing polynomial changes in the time bounds, such a chain rule takes the form

$$\sum_{i=1}^m q^{\text{poly}}(x_i \mid x_{<i}) \leq q^{\text{poly}}(x_1 \cdots x_m) + m \cdot O(\log n).$$

For a source distribution $\mathcal{D}$, we define the corresponding average-case notion by requiring the expected sum of conditional description lengths to be close to the source entropy, up to the same accumulated block-wise overhead²⁰. Namely, average-case prequential q^{poly} -compression requires the following bound

$$\mathbf{E}_{X \sim \mathcal{D}} \left[\sum_{i=1}^m q^{\text{poly}}(X_i \mid X_{<i}) \right] \leq H(\mathcal{D}) + |\mathcal{D}| + m \cdot O(\log n),$$

where the constant in the “ $O(\log n)$ ” term is fixed independently of the block length. Here, $|\mathcal{D}|$ denotes the description length of a sampler for $\mathcal{D}$ and is absorbed into the final term whenever $m \geq |\mathcal{D}|$. We emphasize that this is purely a statement about the source distribution $\mathcal{D}$; we do not assume the existence of an explicit efficient encoder that realizes the conditional description lengths $q^{\text{poly}}(\cdot \mid \cdot)$. Replacing q^{poly} with rk^{poly} gives the corresponding average-case notion of prequential rk^{poly} -compression.

Hirahara and Nanashima [HN26a] use this viewpoint to give an unconditional, distribution-by-distribution characterization. Here, it is important to distinguish the elementary symbols to be predicted from the longer blocks used for compression. Let $Z = (Z_1, \dots, Z_N) \sim \mathcal{D}$, where each Z_i is a bit or, more generally, a symbol of constant length. We group consecutive symbols into blocks $X_1, \dots, X_m$, each consisting of $b = O(\log n)$ symbols. This grouping is used for prequential compression, while the sequential generator still predicts one elementary symbol Z_i at a time.

Let $\mathcal{A}(Z_{<i})$ denote the output distribution of a sequential generator after observing $Z_{<i}$. For accuracy $\epsilon > 0$ and confidence error $\delta > 0$, the desired guarantee is

$$\Pr_{\substack{Z \sim \mathcal{D} \\ I \sim [N]}} [\Delta_{\text{TV}}(\mathcal{A}(Z_{<I}), (Z_I \mid Z_{<I})) \leq \epsilon] \geq 1 - \delta,$$

where I is uniform over $[N]$. The authors show that, in the constant-accuracy and constant-confidence regime, the existence of an efficient sequential generator satisfying this guarantee is equivalent to average-case prequential q^{poly} -compression of the block sequence $X_1, \dots, X_m$ and, equivalently, to average-case prequential rk^{poly} -compression.

Although this does not yield the total variation guarantee required in Definition 2.7, it does yield the following Yao-style next-bit learning guarantee, with no restriction on the support size of the target distribution. This follows from an averaging argument over prefixes and a standard application of Markov’s inequality.

²⁰This is equivalent, in the constant-parameter regime, to the formulation in [HN26a], which allows an additive ηN term, where N denotes the total message length. Indeed, if each block has length b , then $m = \Theta(N/b)$ and hence $m \cdot O(\log n) = O(N \log n/b)$. Choosing $b = \Theta(\eta^{-1} \log n)$ makes this overhead $O(\eta N)$. Conversely, when $b = \Theta(\log n)$ and η is constant, an additive ηN term is of the form $m \cdot O(\log n)$. Thus, the two formulations are interchangeable by adjusting the constant in the block length.

Theorem 6.8. *For every $\epsilon, \delta \in (0, 1]$, there exist a constant C and a polynomial-time learner L such that the following holds. Let $\mathcal{D}$ be any P/poly-samplable distribution over $\{0, 1\}^n$. Suppose that, for every batch size m , the sequence obtained by concatenating m independent samples from $\mathcal{D}$ and partitioning the resulting string into consecutive blocks of length $C \log n$ admits average-case prequential q^{poly} -compression or average-case prequential rK^{poly} -compression.*

Then, given sample access to $\mathcal{D}$, the learner L outputs a polynomial-time randomized next-bit predictor h satisfying

$$\Pr_{h \leftarrow L^{\mathcal{D}}} \left[\mathbf{E}_{\substack{x \sim \mathcal{D} \\ i \sim [n]}} [\Delta_{\text{TV}}(h(x_{<i}), (\mathcal{D}_i \mid \mathcal{D}_{<i} = x_{<i}))] \leq \epsilon \right] \geq 1 - \delta.$$

The transformation from prequential q^{poly} -compression to sequential generation applies the advised universal extrapolation procedure discussed in [Section 5.2.2](#). As reviewed there, the completed blocks $X_{<j}$ are treated as advice. The algorithmic chain rule for prequential compression incurs an additional $O(\log n)$ KL loss per block. To obtain the total variation bound stated above, this loss must once again be amortized through conditional sampling. Rejection sampling provides the required conditioning. When generating Z_i in the current block X_j , rejection sampling is used only to condition on the previously generated symbols in X_j , since the completed blocks $X_{<j}$ are already available as advice. Since the current block contains only $b = O(\log n)$ symbols, every such conditioning prefix has length $O(\log n)$. The corresponding conditioning event therefore has inverse-polynomial probability, and rejection sampling runs in polynomial time. Unlike the universal-extrapolation approach discussed above, this transformation does not require distributional inversion or any assumption ruling out one-way functions.

Accordingly, the result does not claim that every efficiently samplable distribution can be learned, which would be incompatible with cryptographic hardness. Instead, it gives a distribution-by-distribution characterization: a distribution family admits efficient sequential generation precisely when it satisfies the corresponding prequential-compression condition.

The short-block construction balances coding overhead against the running time of rejection sampling. To reduce an additive $O(\log n)$ overhead per block to an ϵ fraction of the block length, the block length must be $\Omega(\epsilon^{-1} \log n)$. Rejection sampling on such a block takes time $2^{O(\epsilon^{-1} \log n)} = n^{O(1/\epsilon)}$.²¹ This is polynomial in n for every constant ϵ , but it is not polynomial jointly in n and ϵ^{-1} . The current characterization is therefore limited to constant accuracy and confidence, and extending it to general parameters remains open.

Problem 6.9. Does the equivalence between efficient sequential generation, prequential q^{poly} -compression, and prequential rK^{poly} -compression extend to arbitrary polynomial-length blocks and arbitrary accuracy and confidence parameters, with running time polynomial in the message length, the block length, ϵ^{-1} , and δ^{-1} ?

In particular, it is important that ϵ can be made inverse-polynomially small in n , as this allows the next-bit prediction guarantee in [Theorem 6.8](#) to be converted into the standard total variation guarantee via the triangle inequality.

6.2.4 Language Modeling

The framework developed above appears directly in modern language modeling. A *token* is a symbol from a finite vocabulary, such as a byte or a fragment of a word. An *autoregressive language model* $\mathcal{E}$ assigns, to every prefix $x_{<i}$, a distribution $\mathcal{E}(\cdot \mid x_{<i})$ over the next token. It generates a message by

²¹More precisely, here we use the elementary fact that the universal distribution assigns probability at least $2^{-O(b)}$ to any b -bit prefix, since an $O(b)$ -bit program can simply print an extension of that prefix, ignoring the advice.

repeatedly sampling the next token from this distribution. Thus, with each token viewed as a short block, language modeling is precisely the sequential conditional-generation setting considered in [Section 6.2.3](#).

The standard objective for training an autoregressive language model is the next-token log loss [e.g., [RWC⁺19](#), [BMR⁺20](#)]:

$$L_{\mathcal{E}}(x) \triangleq \sum_{i=1}^N \log \frac{1}{\mathcal{E}(x_i \mid x_{<i})} = \log \frac{1}{\mathcal{E}(x)}.$$

As reviewed in [Section 6.2.1](#), arithmetic coding uses the same conditional probabilities to encode x in $L_{\mathcal{E}}(x) + O(1)$ bits. Hence, minimizing log loss is equivalent to minimizing the length of the corresponding lossless encoding. In this sense, a language model is also a compressor, provided that the model is available to both the encoder and the decoder.

The reverse connection uses the normalized incremental weights discussed in [Section 6.2.2](#), assigning each possible continuation y the weight $2^{-(c(xy)-c(x))}$. For each prefix x , normalizing these weights defines a distribution over the next token, and iterating this construction turns a compressor into an autoregressive generator. A good overall compression rate, however, does not in general guarantee that the resulting conditional distributions are accurate.

An early neural implementation of this connection predates modern language models. Schmidhuber and Heil [[SH96](#)] trained a feedforward network to estimate conditional next character probabilities and coupled its outputs with Huffman or arithmetic coding for lossless text compression; in their offline scheme, the trained predictor was shared by the encoder and decoder.

Delétang et al. [[DRD⁺24](#)] investigate both directions of this correspondence experimentally. They combine the next-token probabilities produced by large language models with arithmetic coding to obtain lossless compressors for text, images, and audio. Conversely, they normalize incremental code lengths produced by compressors such as gzip and use the resulting conditional distributions for autoregressive generation. Related experiments use prequential code length to evaluate neural predictors trained sequentially on previously revealed examples [e.g., [BO18](#), [BLH23](#)]. Together, these recent developments place the relationship between sequential prediction and compression directly within modern machine learning.

The complexity-theoretic results discussed above provide an idealized time-bounded formulation of the same prediction–compression principle, with K^{poly} and related measures such as rK^{poly} and q^{poly} serving as ideal universal benchmarks for the shortest descriptions that can be decoded in polynomial time. The existing correspondences thus describe the relationship between prediction and compression at this idealized time-bounded level, but do not yet yield guarantees for concrete large language models. This raises a natural question: can meta-complexity shed light on concrete language models, for instance, by providing new methods that transfer compression guarantees to sequential-generation guarantees without quantitative loss while preserving computational efficiency? Developing a connection between the idealized time-bounded theory and modern language models remains an important direction for future research.

7 Meta-Complexity and Computational Hardness from Learning

Ideas from complexity have long informed the design of new learning algorithms (see, e.g., [LMN93]); conversely, ideas from learning have yielded insights in complexity (see, e.g., [IW01]). More recently, there have been indications of deep and far-reaching connections between the two fields that go well beyond these results. They are part of an emerging theory that shows useful links between *proving complexity lower bounds* and *designing new algorithms*. The learning algorithm of [CIKK16] and the complexity lower bound of [Wil14] are examples of breakthroughs achieved in this way.

Learning algorithms in different models are known for several important classes of Boolean functions, including DNFs [Jac97], constant-depth circuits with AND, OR, and NOT gates [LMN93], and constant-depth circuits extended with MOD_p gates, where p is prime [CIKK16]. However, despite major efforts, a learning algorithm for the class TC_2^0 , i.e., two layers of polynomially many threshold gates, is unknown and corresponds to a central problem in computational learning theory. Note that every reasonable concept class can be learned in time $2^n \cdot \text{poly}(n)$, simply by querying the unknown function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on all input strings. In contrast, even if we allow the learning algorithm to run in *quantum time* slightly below 2^n and make *quantum membership queries*, there is currently no known algorithm for learning TC_2^0 .

In a seminal work, [FK06] demonstrated that the existence of polynomial-time learning algorithms implies complexity lower bounds. Subsequent results strengthened this connection by showing that even learning in time slightly below 2^n suffices to derive lower bounds. Put differently, any “non-trivial” learning algorithm can be transformed into a source of hard functions. Hardness can be extracted from non-trivial deterministic learning algorithms [KKO13], randomised PAC learning algorithms [OS17], and quantum PAC learning algorithms [AGG⁺21]. In general, the existence of any non-trivial learning algorithm for a class $\mathcal{C}$ of functions (such as TC_2^0) implies that there are explicit problems that cannot be computed in $\mathcal{C}$.²² These results shed some light on the difficulty of designing provably correct learning algorithms for classes such as TC_2^0 , for which super-polynomial size complexity lower bounds remain unknown.

In the next sections, we survey results of this kind and present a concrete example illustrating how learning algorithms can be used to prove lower bounds in meta-complexity.

7.1 Deterministic Learners

Below we consider *typical* circuit classes $\mathcal{C}$ that are closed under restrictions and complementation,²³ and contain the identically constant functions.²⁴

As discussed above, any class $\mathcal{C}$ can be learned in time $2^n \cdot \text{poly}(n)$. For this reason, we say that a learning algorithm is *nontrivial* if its running time is of the form $2^n / n^{\omega(1)}$, i.e., if it admits an arbitrarily large polynomial speed-up over the trivial time bound. Our next result shows that the existence of nontrivial deterministic learning algorithms has significant complexity-theoretic consequences.

Recall that in MKtP (Minimum Kt Problem), we are given a string $x \in \{0, 1\}^n$ and a parameter $s \in \mathbb{N}$, and the goal is to decide if $\text{Kt}(x) \leq s$. Here $\text{Kt}(x)$ denotes the Levin time-bounded Kolmogorov complexity

²²Interestingly, previous work [ST17, Kar24] has shown that non-trivial learning algorithms of the required form can be designed for a large number of circuit classes for which lower bounds are already known.

²³Recall that a class is closed under restrictions if any function g obtained by setting a subset of input bits of $f \in \mathcal{C}$ to 0 and 1 is also in $\mathcal{C}$. Closure under complementation means that if a Boolean function $f \in \mathcal{C}$ then the function $g(x) = 1 - f(x) \in \mathcal{C}$.

²⁴If a circuit class $\mathcal{C}$ does not satisfy one of these properties, it is not difficult to use the $\mathcal{C}$ -circuit evaluation problem to define an explicit language in the complexity class P that is not in $\mathcal{C}$.

of x [Lev84], which is defined as

$$\text{Kt}(x) \triangleq \min_{\substack{\Pi \in \{0,1\}^* \\ t \in \mathbb{N}}} \{|\Pi| + \lceil \log t \rceil \mid \text{program } \Pi \text{ outputs } x \text{ within } t \text{ steps}\}.$$

We can also think of MKtP as a language contained in $\{0,1\}^*$ by fixing the threshold parameter s to a large bound such as $s(n) = n - \log n$. Showing that $\text{MKtP} \notin \text{P}$ is a longstanding open problem in meta-complexity [ABK⁺06].

Theorem 7.1 (MKtP Lower Bounds from Deterministic Learning). *The following results hold:*

- (i) **Non-Uniform Hardness for MKtP from Subexponential Time Deterministic Learning:**
Suppose that a typical circuit class $\mathcal{C} = \{\mathcal{C}_n\}_{n \geq 1}$ can be PAC-learned to error $\varepsilon \leq 0.499$ under the uniform distribution using membership queries in deterministic time $2^{n/2}/n^{\omega(1)}$. Then $\text{MKtP} \notin \mathcal{C}$.
- (ii) **Uniform Hardness for MKtP from Nontrivial Deterministic Learning:**
Suppose that, for every $k \geq 1$, Boolean circuits of size n^k can be PAC-learned to error $\varepsilon \leq 0.499$ under the uniform distribution using membership queries in deterministic time $2^n/n^{\omega(1)}$. Then $\text{MKtP} \notin \text{P}$.

Proof Sketch. To prove the first item, let $A^{\text{MQ}(\cdot)}(1^n)$ be a deterministic algorithm running in time $T(n)$ that learns every $f \in \mathcal{C}_n$ to error ε . Let $\bar{0}: \{0,1\}^n \rightarrow \{0,1\}$ be the constant zero function. Suppose towards a contradiction that $\text{MKtP} \in \mathcal{C}$. Consider the Boolean function $f_n: \{0,1\}^n \rightarrow \{0,1\}$ such that $f_n(x) = 1$ if and only if $\text{Kt}(x) \geq n - \log n$. Then $\{f_n\}_{n \geq 1} \in \mathcal{C}$ under the assumption that $\text{MKtP} \in \mathcal{C}$, since f_n is the complement of a restriction of the language MKtP.

Next, consider the execution of $A^{\text{MQ}(\cdot)}(1^n)$ on $\bar{0}$, and let $h_{\bar{0}}$ be its output hypothesis. Similarly, let h_{f_n} be the output hypothesis produced on f_n . Then

$$\text{dist}(h_{\bar{0}}, \bar{0}) \leq \varepsilon \quad \text{and} \quad \text{dist}(h_{f_n}, f_n) \leq \varepsilon.$$

We claim that if $T(n) = 2^{n/2}/n^{\omega(1)}$ and n is large enough, then $h_{\bar{0}} \equiv h_{f_n}$, i.e., they compute the same function. It is enough to argue that $A^{\text{MQ}(\bar{0})}(1^n)$ and $A^{\text{MQ}(f_n)}(1^n)$ have the same computation transcript, and therefore produce the same hypotheses. To see this, we argue that every membership query to $\text{MQ}(f_n)$ produces 0 (as in $\text{MQ}(\bar{0})$). Indeed, it is not hard to prove by induction on i that if $q_i \in \{0,1\}^n$ is the i -th query, where $i \leq T(n)$, then $\text{Kt}(q_i) < n - \log n$. This is because we can describe q_i by knowing the code of A , n , and i , and by simulating A on input 1^n for at most $T(n)$ steps. Consequently,

$$\text{Kt}(q_i) \leq (|A| + \log n + \log i) + \log(T(n) \cdot \text{poly}(n)) \leq O(\log n) + 2 \log T(n) \leq n - \omega(\log n) < n - \log n,$$

where we used that $T(n) = 2^{n/2}/n^{\omega(1)}$. Consequently,

$$\text{dist}(h_{\bar{0}}, h_{f_n}) = 0.$$

Applying the triangle inequality, we obtain that

$$\text{dist}(\bar{0}, f_n) \leq 2\varepsilon.$$

In contrast, since a random n -bit string has Kt complexity $\geq n - \log n$ with probability $1 - o(1)$, we have

$$\text{dist}(\bar{0}, f_n) \geq 1 - o(1).$$

This contradicts our choice of $\varepsilon \leq 0.499$.

For the proof of the second item, an additional idea is used. Assume now that $T(n) = 2^n/n^{\omega(1)}$ and $\text{MKtP} \in \text{P}$. For each k , let $\mathcal{C}^k$ be the class of circuits of size n^k . Adapting the proof of the first item, we argue below that $\text{MKtP} \notin \mathcal{C}^k$. Since this holds for all k , it implies in particular that $\text{MKtP} \notin \text{P}$, which is a contradiction.

Fix $k \in \mathbb{N}$. Note that, in the argument above, we had to assume a lower running time for the learning algorithm because of the additional $\log T(n)$ contribution coming from the encoding of $i \in [T(n)]$, the index of the query. We avoid the extra term coming from the encoding of the index. We will use that $\text{MKtP} \in \text{P}$ to show that $\text{Kt}(q_i) < n - \log n$ for every query $q_i \in \{0, 1\}^n$ made by the learner. In more detail, towards a contradiction, suppose that for some i we have $\text{Kt}(q_i) \geq n - \log n$, and let i be the first such index. Then, we can describe q_i as follows: q_i is the first query made by $A(1^n)$ during the simulation such that $\text{Kt}(q_i) \geq n - \log n$. Since the latter condition can be checked in polynomial time under the assumption that $\text{MKtP} \in \text{P}$, it is not hard to see that

$$\text{Kt}(q_i) \leq O(1) + \log n + \log(T(n) \cdot \text{poly}(n)) < n - \log n,$$

for large enough n . This contradiction completes the proof. $\square$

Complementing this result, we mention that separations against $\mathcal{C}$ (i.e., non-uniform lower bounds) for some problem in $\text{E} = \text{DTIME}[2^{O(n)}]$ (instead of $\text{MKtP} \in \text{E}$) also follow from deterministic learners running in time $T(n) = 2^n/n^{\omega(1)}$. For more details, see the results and techniques in [KKO13].

7.2 Randomized and Quantum Learners

As we consider more powerful learning algorithms, the task of constructing a computationally hard function from them becomes more elusive. For instance, in the proof of [Theorem 7.1](#), we used in a crucial way that the set of queries produced by the learner is uniquely determined once we fix the function $f \in \mathcal{C}_n$ for the learning task. If the learning algorithm is *randomized*, this property no longer holds, and the strategy described above simply does not work.

Nevertheless, using more sophisticated arguments that rely on the theory of computational pseudorandomness, it is possible to extract complexity lower bounds from any non-trivial learning algorithm. Let $\text{BPE} \triangleq \text{BPTIME}[2^{O(n)}]$ denote randomized exponential time, i.e., the randomized analogue of E .

Theorem 7.2 (Lower Bounds from Nontrivial Randomized Learning [OS17]). *Let $\mathcal{C}$ be a typical circuit class. If polynomial-size $\mathcal{C}$ -circuits can be learned using membership queries with error $\varepsilon < 0.499$ under the uniform distribution in randomized time $2^n/n^{\omega(1)}$, then $\text{BPE} \not\subseteq \mathcal{C}(\text{poly})$.*

It is also possible to consider a more expressive class of *quantum* learning algorithms (see [AdW17] for more on quantum learning theory). Let $\text{BQE} \triangleq \text{BQTIME}[2^{O(n)}]$ denote quantum exponential time.

Theorem 7.3 (Lower Bounds from Nontrivial Quantum Learning [AGG⁺21]). *Let $\mathcal{C}$ be a typical circuit class. If polynomial-size $\mathcal{C}$ -circuits can be learned using quantum membership queries with error $\varepsilon < 0.499$ under the uniform distribution in quantum time $2^n/n^{\omega(1)}$, then $\text{BQE} \not\subseteq \mathcal{C}(\text{poly})$.*

In particular, it follows that if there is a randomized (or even quantum) learning algorithm for the class of depth-2 threshold circuits (TC_2^0) that runs in time $2^n/n^{\Omega(1)}$, then a new complexity lower bound would follow (see [Figure 7](#) for a pictorial representation and a related discussion). We refer to [APS26] and references therein for progress on the learnability of subclasses of depth-2 threshold circuits.

A typical circuit class $\mathcal{C}$

Example: $\mathcal{C} = \text{TC}_2^0$

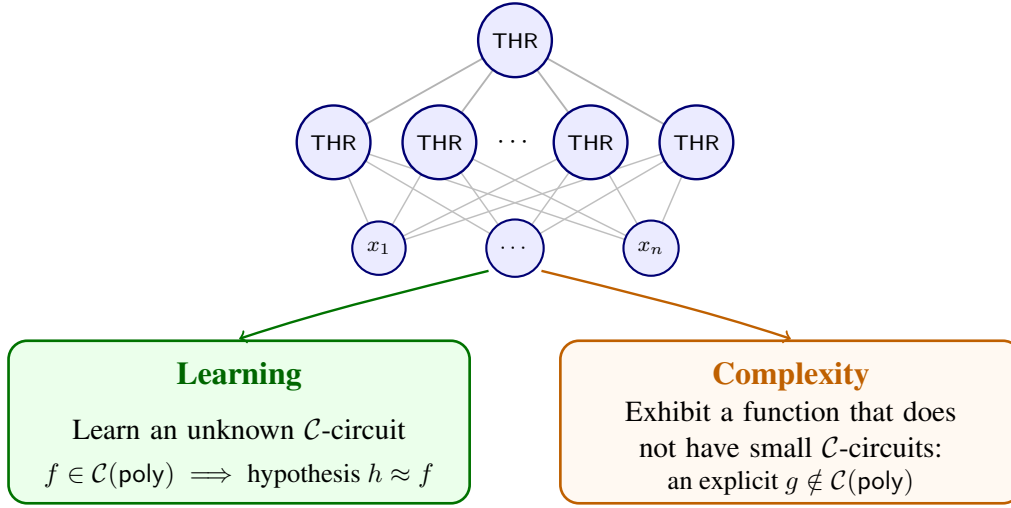


Figure 7: Two closely related questions about a typical circuit class $\mathcal{C}$, illustrated by the class TC_2^0 of depth-2 threshold circuits (see Section 2.4.3). By Theorem 7.2 and Theorem 7.3, a nontrivial classical or quantum learner for $\mathcal{C}$ yields a new circuit size lower bound against $\mathcal{C}$ for an explicit function. Conversely, a combinatorial lower bound that naturalises yields a constructive property that can be converted into a learning algorithm for $\mathcal{C}$, as discussed in Section 5.1. A recurring pattern across circuit classes at the frontier is that the two questions have the same status: either learners and lower bounds are known, or both remain open. In particular, for TC_2^0 , neither a nontrivial learner nor a superpolynomial circuit size lower bound for an explicit function is known.

The original proof of Theorem 7.2 [OS17] makes use of the learning speedups discussed in Section 5.4, which reduces the result to a proof that *subexponential* time learning in time $2^{n^{o(1)}}$ implies complexity lower bounds. On the other hand, [AGG⁺21] explores a more direct perspective that can be used to derive both Theorem 7.2 and Theorem 7.3. We refer to [OS17, AGG⁺21] for more information about these theorems and their proofs.

While the results discussed in this section apply to a significantly larger class of learning algorithms, they come with a technical caveat: the hard function extracted from the learning assumption is hard only on infinitely many input lengths n . In contrast, it can be checked that the lower bound arguments in Theorem 7.1 establish hardness for every sufficiently large input length n . This caveat is a consequence of the more indirect nature of their proofs. It would be interesting to introduce techniques to close this gap.

Problem 7.4 (Almost-Everywhere Hardness from Learning). Show that if a typical circuit class $\mathcal{C}$ admits nontrivial randomized learning algorithms, then there is a language in BPE that is not in $\mathcal{C}(\text{poly})$ on every large enough input length.

We refer to [Vol16] for additional settings where one can derive complexity-theoretic consequences from learning algorithms.

7.3 Unconditional Meta-Complexity Lower Bounds via Learning

Nontrivial learning algorithms have been designed for a large number of circuit classes [ST17, Kar24]. However, in all existing results, lower bounds are already known against the corresponding circuit class. A concrete class for which both lower bounds and learning remain elusive is the class of Boolean functions that can be approximated by low-degree Torus polynomials (see [BHLR19]).²⁵

Problem 7.5 (Nontrivial Learnability of Torus Polynomials). Is there a nontrivial randomized or quantum learning algorithm for the class of Boolean functions that can be $1/n$ -approximated by $\text{poly}(\log n)$ -degree Torus polynomials?

Since any function in the circuit class ACC^0 admits such approximating polynomials [BHLR19], a positive answer to Problem 7.5 leads to a new lower bound against ACC^0 via Theorem 7.3.²⁶ The latter would be a breakthrough result in complexity theory. We refer to [Wil14] for more information on ACC^0 .

A closely related problem is that of learning the class of Boolean circuits $f: \{0, 1\}^n \rightarrow \{0, 1\}$ that can be written as a disjoint union (DOR) of quasi-polynomially many MAJ-AND_t circuits, where the bottom AND gate has fan-in $t = \text{poly}(\log n)$.²⁷

Problem 7.6 (Nontrivial Learnability of DOR-MAJ-AND Circuits). Is there a nontrivial randomized or quantum learning algorithm for the class of quasi-polynomial size DOR-MAJ-AND circuits of polylogarithmic bottom fan-in?

It is not hard to prove that a positive solution to this problem also implies lower bounds against ACC^0 , by combining Theorem 7.3 with the simulation $\text{ACC}^0 \subseteq \text{SYM}^+ \subseteq \text{DOR}_k\text{-MAJ-AND}_t$ for $k = \exp(\text{poly}(\log n))$ and $t = \text{poly}(\log n)$ [BT94].

Complexity Lower Bounds from Learning. While Theorem 7.2 and Theorem 7.3 have not yet led to new complexity lower bounds in a *direct* way, it is possible to use these results in an *indirect* way to establish *unconditional* complexity lower bounds. We describe next a general template:

$$\text{Easiness Assumption} \xrightarrow{(i)} \text{Learning Algorithm} \xrightarrow{(ii)} \text{Computational Hardness}$$

In more detail, we proceed as follows:

- (i) First, from a suitable Easiness Assumption, we show that nontrivial learning *is* possible.
- (ii) Next, we invoke a result such as Theorem 7.2 to derive some form of Computational Hardness.

To obtain an unconditional lower bound, we carefully balance the initial Easiness Assumption with the resulting Computational Hardness consequence, with the goal of achieving a contradiction between the two.

²⁵Degree- d Torus polynomials and degree- d polynomial threshold functions (PTFs) can both be viewed as weighted majorities over the same feature representation. Let $\Phi_d(x)$ denote the vector of all multilinear monomials in x of degree at most d . In both cases, one first computes a weighted sum $p(x) = \langle w, \Phi_d(x) \rangle$. A PTF applies a *sign* activation $\mathbf{1}_{[p(x) \geq 0]}$, whereas a Torus polynomial uses a *periodic* activation: it reduces $p(x)$ modulo 1 and interprets values close to 0 and $1/2$ as the Boolean values 0 and 1, respectively. Thus, the two models use the same degree-bounded feature expansion but different activation functions. As discussed in Section 3, degree- d PTFs can be learned in time $\text{poly}(n^d, 1/\varepsilon, \log(1/\delta))$.

²⁶Recall that an ACC^0 circuit is a constant-depth polynomial-size Boolean circuit consisting of unbounded fan-in AND, OR, NOT, and MOD_m gates, where m is a fixed integer and MOD_m computes the sum of the input bits modulo m .

²⁷Since MAJ-AND_t circuits of bottom fan-in $t = \text{poly}(\log n)$ can be viewed as degree- t PTFs, MAJ-AND_t circuits (without a top DOR gate) can be learned in quasi-polynomial time.

If we are able to implement this plan, it follows that the original Easiness Assumption is *false*, i.e., a corresponding complexity lower bound follows.

Hardness of Estimating Complexity. This approach has borne fruit in the form of an unconditional complexity lower bound in Meta-Complexity. In [Oli19], we considered the Minimum rKt Problem (MrKtP): given a string $w \in \{0, 1\}^n$, estimate its rKt complexity, a measure of the compressibility, or regularity, of w .²⁸ This is a randomised analogue of a fundamental measure introduced by L. Levin in 1984 [Lev84] called Kt complexity. While the complexity of estimating the Kt complexity of an input string (MKtP) remains open, we were able to show that MrKtP $\notin$ BPP, i.e., the rKt complexity of a string cannot be computed in probabilistic polynomial time.

This can be shown by analysing the following Easiness Assumption: MrKtP $\in$ BPP. From this and the technique of [CIKK16] described in Section 5.1, it follows that polynomial-size Boolean circuits can be learned in randomised polynomial time. This is Item (i) in the template presented above.

On the other hand, the existence of such a learning algorithm implies by Theorem 7.2 that some problem in BPE (randomised exponential time) cannot be computed by polynomial size Boolean circuits. This corresponds to Item (ii) in the aforementioned template.

Finally, since every problem in BPE can be reduced using polynomial-size Boolean circuits to MrKtP, it must be the case that MrKtP $\notin$ BPP. This contradicts the original Easiness Assumption.

In other words, it follows unconditionally that MrKtP $\notin$ BPP. Note that our original problem, MrKtP, makes no reference to learning algorithms!

Recall that it is a longstanding open problem to show the analogous result for MKtP, i.e., to prove that MKtP $\notin$ P. In light of the above discussion and Theorem 7.1, it is not hard to see that to obtain this result it would be sufficient to show that if MKtP $\in$ P then nontrivial *deterministic* learning of polynomial-size Boolean circuits is possible. In turn, this would be feasible if we had a *deterministic* analogue of the techniques from Section 5.1.

Problem 7.7 (Deterministic Learning from Meta-Complexity). Is there a way to obtain deterministic learning algorithms assuming the easiness of meta-computational problems such as MKtP?

This is closely related to the problem of derandomizing learning algorithms. For more about this line of work, we refer the reader to [OS18].

To conclude this section, we note that we have only discussed lower bounds that can be extracted from PAC learning algorithms. In principle, it should be possible to obtain stronger complexity-theoretic consequences from the existence of algorithms for more expressive learning tasks (see Section 4). We leave this as an interesting direction for further research.

Problem 7.8 (Complexity-Theoretic Consequences of Learning). Derive stronger complexity-theoretic consequences from the existence of algorithms for learning tasks that are more expressive than PAC learning under the uniform distribution, such as distribution learning and sequence extrapolation.

²⁸Random strings have rKt complexity close to n , while structured strings can be succinctly described and have much smaller rKt complexity.

8 Related Work

Here we highlight some related lines of work that are not covered in the preceding sections, following roughly the order of the survey. We also provide pointers to some references that cover more practical aspects of learning and its connections to compression.

Search vs Decision in Meta-Complexity. The mismatch between recognising the existence of a short description and producing one is highlighted in [Section 1](#). The learning constructions surveyed in [Section 5](#) often circumvent this mismatch by converting a decision procedure into a predictor through a reconstruction argument. Search-to-decision reductions for time-bounded Kolmogorov complexity address the issue directly, asking when a decision procedure suffices to recover a short program. We refer to [\[MP24, HKLO24, HLN24\]](#) for recent work on this direction and related open problems.

Compression in Supervised Learning. The term *model compression* ([Section 3](#)) has a different meaning in modern machine learning, where one starts with a trained model and seeks a smaller approximation. Representative approaches include transferring the predictive behaviour of a large model or ensemble to a smaller network through knowledge distillation [\[HVD15\]](#), compressing an explicitly stored network through pruning, quantization, and entropy coding [\[HMD16\]](#), and training stochastic network parameters subject to an explicit information-theoretic coding budget [\[HPH19\]](#). Dataset distillation moves the compression to the data side, replacing a large training set by a small collection of synthetic examples from which a prescribed training procedure recovers comparable performance [\[WZTE18\]](#).

[Section 3](#) also develops the basic connection between sample compression and generalisation, mentions the qualitative equivalence with finite VC dimension, and states the sample compression conjecture. The classical binary sample compression framework was introduced by Littlestone and Warmuth [\[LW86\]](#) and developed further by Floyd and Warmuth [\[FW95\]](#). Subsequent work extends or refines this perspective in several directions, including multiclass and general loss learning [\[DMY16\]](#), stable compression schemes [\[HK21\]](#), list compression and its limitations [\[HMW24\]](#), and reductions from compression problems in other learning models to classical binary compression [\[AHR25\]](#).

Distributional Restrictions and the Complexity of Learning. Departing from the pessimistic distribution-free definition of PAC learning, the learning paradises of [Section 4.1](#) deliberately restrict the example distributions to feasible ones. This choice can change the computational complexity of the learning task substantially. Blanc et al. [\[BKL⁺26\]](#) show that requiring success only on samplable distributions can enlarge the power of efficient learning, even at the level of sample complexity. Blanc, Lange, Strassle, and Tan [\[BLST25\]](#) instead ask when a learner for a restricted family of example distributions can be lifted to arbitrary example distributions. The latter result is governed by the complexity of expressing the new distribution as a mixture of distributions from the original family. This is distinct from the distributional inversion argument in [Section 5](#), which moves between example distributions by inverting their samplers.

Practical Analogues of the Learning Paradises. The assumptions P.I through P.V from [Section 4.1](#) are much stronger than anything currently achieved in practice, but the tasks they formalize have recognizable counterparts in modern machine learning. These analogies concern the form of each learning task, not the strength of its guarantee. Practical methods exploit regularities in natural data and address restricted families of instances, whereas the paradises require uniform guarantees over broad families of sources, example distributions, or circuits.

P.I is closest to predicting the continuation of a sequence from an observed context, as in models for text, video, or dynamical systems. For instance, Delétang et al. [DRD⁺24] study the practical connection between prediction and compression in this setting, while Grau-Moya et al. [GMGH⁺24] investigate neural predictors trained to approximate universal prediction.

P.II resembles generative modeling, where independent examples are used to learn a procedure for producing new samples. The denoising diffusion model of Ho, Jain, and Abbeel [HJA20] is a prominent example. Generative models can also serve as compressors. Yang, Mandt, and Theis [YMT23] survey this area, while Kingma et al. [KSPH21] turn a variational diffusion model into a lossless compressor whose rate is close to its variational coding bound. From the perspective of this survey, these systems also expose resources not captured by code length, notably encoding and decoding time and the size of the shared learned model. We note that practical systems are generally evaluated by prediction loss, sample quality, or application performance, rather than by guarantees that hold uniformly over all efficiently samplable distributions, as P.I and P.II require.

P.III is closest to learning when labels may be noisy and the target need not belong to the chosen hypothesis class. Haussler [Hau92] extends the PAC model to general loss minimization, while Kearns, Schapire, and Sellie [KSS94] formulate the agnostic model, in which performance is compared with that of the best predictor in a class. Tractable results for noisy labels generally assume additional structure. For example, Natarajan et al. [NDRT13] study loss correction when the probability of a label error depends only on the true label. This is one example from a vast theoretical and empirical literature, which we do not attempt to survey here.

P.IV considers learning through chosen queries. Angluin [Ang88] provides a foundational framework for query learning. Practical counterparts arise when a learner can choose which inputs are labelled, as in active learning [CGJ96], or can experiment with an implemented system, as in black box checking [PYY99] and model extraction [TZJ⁺16]. Membership queries provide stronger access than the common pool-based model of active learning, since they allow the learner to query any input rather than only examples from a fixed unlabelled sample.

P.V has a broader practical analogue. A heuristic learning algorithm may succeed on concepts that arise naturally in an application even if it fails on adversarial or otherwise atypical concepts from the same representation class. This is precisely the relaxation captured by P.V.

The empirical success of modern machine learning remains compatible with the hardness results discussed above. As explained before, the learning paradises concern uniform guarantees over broad families of efficient sources, distributions, or circuits, which can be far more general than success on the structured instances encountered in practice. These stronger theoretical guarantees enable the connections to cryptography and average-case complexity, which in turn yield hardness of learning results.

Cryptographic Hardness of Learning. The cryptographic literature approaches learning from the opposite direction to much of this survey, using cryptographic assumptions to prove that particular concept classes are hard to learn. Classical examples include the reductions of Pitt and Valiant [PV88] and of Kearns and Valiant [KV94b], followed by other representative examples such as Klivans and Sherstov [KS09] and Daniely et al. [DLSS14] (see also [AOW15] and subsequent work [Dan16, DV21]). There is an extensive literature on the hardness of learning tasks under a variety of cryptographic conjectures, and reviewing these results is beyond the scope of this section.

A related line of work investigates whether weaker complexity-theoretic assumptions are sufficient to rule out efficient learning. Applebaum, Barak, and Xiao identify barriers to basing such results on standard worst-case assumptions [ABX08]. Goldberg and Kabanets [GK25] show that certain formulations of NP-

hardness of learning characterize witness encryption, while Blanc et al. [BKST25] derive computational and statistical tradeoffs from NP hardness.

The discussion of white-box learning from Section 4.5 concentrates on one-way functions. Liu, Mazon, and Pass [LMP25] obtain a corresponding characterization of public-key encryption through a richer white-box learning problem. The existence of secure key-agreement protocols has also been characterized via notions from Kolmogorov complexity and meta-complexity [BLMP23].

Description Length in Statistical Inference and Prediction. The treatment in Section 6 uses coding as a bridge between probabilistic prediction and compression, with particular attention to computational efficiency. Minimum Message Length (MML), originating in the work of Wallace and Boulton [WB68], and Minimum Description Length (MDL), introduced by Rissanen [Ris78], instead use description length as a criterion for statistical inference and model selection. Barron, Rissanen, and Yu [BRY98] give a standard account of the connections among MDL, universal coding, and statistical inference, while Vitányi and Li [VL00] relate MDL and MML to Bayesian inference and Kolmogorov complexity.

Dawid [Daw84] formulated the prequential (“predictive and sequential”) approach, which evaluates a statistical method through the predictions it issues sequentially; under log loss, these predictions themselves define a code. Rissanen [Ris84] connects universal coding with prediction and estimation, while Feder, Merhav, and Gutman [FMG92] study universal prediction of individual sequences. Cesa-Bianchi and Lugosi [CL06] give a standard treatment of sequential prediction and regret. These traditions are close to the coding interpretation developed above, but they generally concern statistical performance rather than the efficient computational transformations that are central here.

Building on Kolmogorov’s earlier ideas about separating structure from randomness, Gács, Tromp, and Vitányi [GTV01] develop algorithmic statistics, asking how much of an individual string’s shortest description should be regarded as an explanatory model and how much as unexplained randomness. This is related to the use of algorithmic information theory for extrapolation in Section 5.2, but its primary object is the structure of a given string rather than prediction from a sequence.

Similarity via Compression. A different use of compression in machine learning, not covered in the survey, treats the lengths produced by a compressor as a measure of similarity between individual objects. Li et al. [LCL⁺04] introduce normalized information distance and demonstrate practical approximations using standard compressors, while Cilibrasi and Vitányi develop normalized compression distance into a general clustering method [CV05]. More recently, Jiang et al. [JYT⁺23] illustrate the same idea in text classification by combining gzip with a nearest neighbor rule. This perspective is complementary to the connections between learning and compression discussed in the survey: compression is used directly as a data analysis primitive, without learning an explicit model.

References

- [AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 2009.
- [ABK⁺06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.
- [ABX08] Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Symposium on Foundations of Computer Science (FOCS)*, pages 211–220, 2008.
- [AdW17] Srinivasan Arunachalam and Ronald de Wolf. A survey of quantum learning theory. *ACM SIGACT News*, 48(2):41–67, 2017.
- [AGG⁺21] Srinivasan Arunachalam, Alex B. Grilo, Tom Gur, Igor C. Oliveira, and Aarthi Sundaram. Quantum learning algorithms imply circuit lower bounds. In *Symposium on Foundations of Computer Science (FOCS)*, pages 562–573, 2021.
- [AGvM⁺18] Eric Allender, Joshua A. Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum circuit size, graph isomorphism, and related problems. *SIAM J. Comput.*, 47(4):1339–1372, 2018.
- [AHR25] Idan Attias, Steve Hanneke, and Arvind Ramaswami. Sample compression scheme reductions. In *International Conference on Algorithmic Learning Theory (ALT)*, pages 134–162, 2025.
- [All17] Eric Allender. The complexity of complexity. In *Computability and Complexity*, pages 79–94. Springer, 2017.
- [All21] Eric Allender. Vaughan Jones, Kolmogorov Complexity, and the new complexity landscape around circuit minimization. *New Zealand journal of mathematics*, 52:585–604, 2021.
- [Ang88] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.
- [AOW15] Sarah R. Allen, Ryan O’Donnell, and David Witmer. How to refute a random CSP. In *Symposium on Foundations of Computer Science (FOCS)*, pages 689–708, 2015.
- [APS26] Josh Alman, Shyamal Patel, and Rocco A. Servedio. Learning functions of halfspaces. In *Symposium on Theory of Computing (STOC)*, pages 1581–1591, 2026.
- [BCK⁺22] Eric Binnendyk, Marco Carmosino, Antonina Kolokolova, R. Ramyaa, and Manuel Sabin. Learning with distributional inverters. In *International Conference on Algorithmic Learning Theory (ALT)*, pages 90–106, 2022.
- [BEHW87] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam’s razor. *Inf. Process. Lett.*, 24(6):377–380, 1987.
- [BFKL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology (CRYPTO)*, volume 773, pages 278–291, 1993.
- [BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.
- [BFP05] Harry Buhrman, Lance Fortnow, and Aduri Pavan. Some results on derandomization. *Theory Comput. Syst.*, 38(2):211–227, 2005.

- [BHLR19] Abhishek Bhrushundi, Kaave Hosseini, Shachar Lovett, and Sankeerth Rao. Torus polynomials: An algebraic approach to ACC^0 lower bounds. In *Innovations in Theoretical Computer Science Conference (ITCS)*, 2019.
- [BKL⁺26] Guy Blanc, Caleb Koch, Jane Lange, Carmen Strassle, and Li-Yang Tan. Samplability makes learning easier. In *Innovations in Theoretical Computer Science Conference (ITCS)*, volume 362, pages 20:1–20:12, 2026.
- [BKST25] Guy Blanc, Caleb Koch, Carmen Strassle, and Li-Yang Tan. Computational-statistical tradeoffs from NP-hardness. In *Symposium on Foundations of Computer Science (FOCS)*, pages 512–530, 2025.
- [BL93] Dan Boneh and Richard J. Lipton. Amplification of weak learning under the uniform distribution. In *Conference on Computational Learning Theory (COLT)*, pages 347–351. ACM, 1993.
- [Ble13] Guy E. Blelloch. Introduction to data compression. *Lecture Notes – Computer Science Department, Carnegie Mellon University*, 2013.
- [BLH23] Jörg Bornschein, Yazhe Li, and Marcus Hutter. Sequential learning of neural networks for prequential MDL. In *International Conference on Learning Representations (ICLR)*, 2023.
- [BLMP23] Marshall Ball, Yanyi Liu, Noam Mazon, and Rafael Pass. Kolmogorov comes to Cryptomania: On interactive Kolmogorov complexity and key-agreement. In *Symposium on Foundations of Computer Science (FOCS)*, pages 458–483, 2023.
- [BLST25] Guy Blanc, Jane Lange, Carmen Strassle, and Li-Yang Tan. A distributional-lifting theorem for PAC learning. In *Conference on Learning Theory (COLT)*, pages 375–379, 2025.
- [BMR⁺20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
- [BO18] Léonard Blier and Yann Ollivier. The description length of deep learning models. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 2220–2230, 2018.
- [BRY98] Andrew R. Barron, Jorma Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, October 1998.
- [BT94] Richard Beigel and Jun Tarui. On ACC. *Computational Complexity*, 4(4):350–366, 1994.
- [BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1), 2006.
- [CDV24] Sílvia Casacuberta, Cynthia Dwork, and Salil P. Vadhan. Complexity-theoretic implications of multi-calibration. In *Symposium on Theory of Computing (STOC)*, pages 1071–1082, 2024.
- [CGJ96] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.
- [CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Conference on Computational Complexity (CCC)*, pages 10:1–10:24, 2016.

- [CIKK17] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Agnostic learning from tolerant natural proofs. In *International Workshop on Randomization and Approximation Techniques (RANDOM)*, volume 81, pages 35:1–35:19, 2017.
- [CJ26] Marco Carmosino and Mandar Juvekar. Parallel Kolmogorov complexity with connections to learning theory and cryptography. Unpublished preprint, September 2026.
- [CL06] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
- [CLO⁺23] Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. Polynomial-time pseudodeterministic construction of primes. In *Symposium on Foundations of Computer Science (FOCS)*, pages 1261–1270, 2023.
- [CT91] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, 1991.
- [CTW26] Lijie Chen, Avishay Tal, and Yichuan Wang. Superquadratic lower bounds for depth-2 linear threshold circuits. In *Symposium on Theory of Computing (STOC)*, pages 874–885, 2026.
- [CV05] Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *IEEE Trans. Inf. Theory*, 51(4):1523–1545, 2005.
- [Dan16] Amit Daniely. Complexity theoretic limitations on learning halfspaces. In *Symposium on Theory of Computing (STOC)*, pages 105–117, 2016.
- [Dav73] L. Davisson. Universal noiseless coding. *IEEE Transactions on Information Theory*, 19(6):783–795, 1973.
- [Daw84] A. P. Dawid. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)*, 147(2):278–290, 1984.
- [DLSS14] Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Symposium on Theory of Computing (STOC)*, pages 441–448, 2014.
- [DMY16] Ofir David, Shay Moran, and Amir Yehudayoff. Supervised learning through the lens of compression. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 2784–2792, 2016.
- [DRD⁺24] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. Language modeling is compression. In *International Conference on Learning Representations (ICLR)*, 2024.
- [DV21] Amit Daniely and Gal Vardi. From local pseudorandom generators to hardness of learning. In *Conference on Learning Theory (COLT)*, pages 1358–1394, 2021.
- [Fel10] Vitaly Feldman. Distribution-specific agnostic boosting. In *Innovations in Theoretical Computer Science (ITCS)*, pages 241–250, 2010.
- [Fit66] B. M. Fitingof. Optimal coding in the case of unknown and changing message statistics. *Problems of Information Transmission*, 2(2):1–7, 1966.
- [FJS91] Merrick L. Furst, Jeffrey C. Jackson, and Sean W. Smith. Improved learning of AC^0 functions. In *Conference on Learning Theory (COLT)*, pages 317–325, 1991.
- [FK06] Lance Fortnow and Adam R. Klivans. Efficient learning algorithms yield circuit lower bounds. In *Annual Conference on Learning Theory (COLT)*, pages 350–363, 2006.

- [FMG92] Meir Feder, Neri Merhav, and Michael Gutman. Universal prediction of individual sequences. *IEEE Transactions on Information Theory*, 38(4):1258–1270, July 1992.
- [FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Syst. Theory*, 17(1):13–27, 1984.
- [FW95] Sally Floyd and Manfred K. Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304, December 1995.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
- [GHL⁺25] Halley Goldberg, Jinqiao Hu, Zhenjian Lu, Jingyi Lyu, and Igor C. Oliveira. Synergies between complexity theory and nondeterministic Kolmogorov complexity. *Electron. Colloquium Comput. Complex.*, TR25, 2025.
- [GJK⁺24] Tom Gur, Mohammad Mahdi Jahanara, Mohammad Mahdi Khodabandeh, Ninad Rajgopal, Bahar Salamatian, and Igor Shinkar. On the power of interactive proofs for learning. In *Symposium on Theory of Computing (STOC)*, pages 1063–1070, 2024.
- [GK22] Halley Goldberg and Valentine Kabanets. A simpler proof of the worst-case to average-case reduction for polynomial hierarchy via symmetry of information. *Electron. Colloquium Comput. Complex.*, 7:1–14, 2022.
- [GK23] Halley Goldberg and Valentine Kabanets. Improved learning from Kolmogorov complexity. In *Computational Complexity Conference (CCC)*, pages 12:1–12:29, 2023.
- [GK25] Halley Goldberg and Valentine Kabanets. Witness encryption and NP-hardness of learning. In *Computational Complexity Conference (CCC)*, pages 34:1–34:43, 2025.
- [GKLO22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Probabilistic Kolmogorov complexity with applications to average-case complexity. In *Computational Complexity Conference (CCC)*, pages 16:1–16:60, 2022.
- [GMGH⁺24] Jordi Grau-Moya, Tim Genewein, Marcus Hutter, Laurent Orseau, Gregoire Deletang, Elliot Catt, Anian Ruoss, Li Kevin Wenliang, Christopher Mattern, Matthew Aitchison, and Joel Veness. Learning universal predictors. In *International Conference on Machine Learning (ICML)*, pages 16178–16205, 2024.
- [Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Tools*. Cambridge University Press, 2001.
- [GS91] Andrew V. Goldberg and Michael Sipser. Compression and ranking. *SIAM J. Comput.*, 20(3):524–536, 1991.
- [GTV01] Péter Gács, John T. Tromp, and Paul M. B. Vitányi. Algorithmic statistics. *IEEE Transactions on Information Theory*, 47(6):2443–2463, September 2001.
- [Hau92] David Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inf. Comput.*, 100(1):78–150, 1992.
- [HIL⁺23] Shuichi Hirahara, Rahul Ilango, Zhenjian Lu, Mikito Nanashima, and Igor C. Oliveira. A duality between one-way functions and average-case symmetry of information. In *Symposium on Theory of Computing (STOC)*, pages 1039–1050, 2023.

- [HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.
- [Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018.
- [Hir20a] Shuichi Hirahara. Characterizing average-case complexity of PH by worst-case meta-complexity. In *Symposium on Foundations of Computer Science (FOCS)*, pages 50–60, 2020.
- [Hir20b] Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In *Computational Complexity Conference CCC*, pages 20:1–20:47, 2020.
- [Hir21] Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *Symposium on Theory of Computing (STOC)*, pages 292–302, 2021.
- [Hir22a] Shuichi Hirahara. Meta-computational average-case complexity: A new paradigm toward excluding Heuristica. *Bull. EATCS*, 136, 2022.
- [Hir22b] Shuichi Hirahara. NP-hardness of learning programs and partial MCSP. In *Symposium on Foundations of Computer Science (FOCS)*, pages 968–979, 2022.
- [Hir22c] Shuichi Hirahara. Symmetry of information from meta-complexity. In *Computational Complexity Conference (CCC)*, pages 26:1–26:41, 2022.
- [HJA20] Jonathan Ho, Ajay N. Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 6840–6851, 2020.
- [HK21] Steve Hanneke and Aryeh Kontorovich. Stable sample compression schemes: New applications and an optimal SVM margin bound. In *International Conference on Algorithmic Learning Theory (ALT)*, pages 697–721, 2021.
- [HKLO24] Shuichi Hirahara, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Exact search-to-decision reductions for time-bounded Kolmogorov complexity. In *Computational Complexity Conference (CCC)*, pages 29:1–29:56, 2024.
- [HLN24] Shuichi Hirahara, Zhenjian Lu, and Mikito Nanashima. Optimal coding for randomized Kolmogorov complexity and its applications. In *Symposium on Foundations of Computer Science (FOCS)*, pages 369–378, 2024.
- [HLO24] Shuichi Hirahara, Zhenjian Lu, and Igor C. Oliveira. One-way functions and pKt complexity. In *Theory of Cryptography (TCC)*, pages 253–286, 2024.
- [HLO26a] Jinqiao Hu, Zhenjian Lu, and Igor C. Oliveira. Equivalence between coding and complexity lower bounds. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 110:1–110:23, 2026.
- [HLO26b] Jinqiao Hu, Zhenjian Lu, and Igor C. Oliveira. Hardness of computing nondeterministic Kolmogorov complexity. In *Computational Complexity Conference (CCC)*, 2026.
- [HMD16] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.
- [HMO26] Jinqiao Hu, Yahel Manor, and Igor C. Oliveira. Failure of symmetry of information for randomized computations. In *Symposium on Theory of Computing (STOC)*, pages 1592–1603, 2026.

- [HMW24] Steve Hanneke, Shay Moran, and Tom Wajnane. List sample compression and uniform convergence. In *Conference on Learning Theory (COLT)*, pages 2360–2388, 2024.
- [HN21] Shuichi Hirahara and Mikito Nanashima. On worst-case learning in relativized Heuristica. In *Symposium on Foundations of Computer Science (FOCS)*, pages 751–758, 2021.
- [HN23] Shuichi Hirahara and Mikito Nanashima. Learning in Pessiland via inductive inference. In *Symposium on Foundations of Computer Science (FOCS)*, pages 447–457, 2023.
- [HN26a] Shuichi Hirahara and Mikito Nanashima. Complexity-theoretic universal inductive inference. In *Symposium on Theory of Computing (STOC)*, 2026.
- [HN26b] Shuichi Hirahara and Mikito Nanashima. A sharp characterization of Pessiland. In *Symposium on Theory of Computing (STOC)*, 2026.
- [HPH19] Marton Havasi, Robert Peharz, and José Miguel Hernández-Lobato. Minimal random code learning: Getting bits back from compressed model parameters. In *International Conference on Learning Representations (ICLR)*, 2019.
- [HQC24] Marcus Hutter, David Quarel, and Elliot Catt. *An introduction to universal artificial intelligence*. Chapman and Hall/CRC, 2024.
- [HS07] Lisa Hellerstein and Rocco A Servedio. On PAC learning algorithms for rich boolean function classes. *Theoretical Computer Science*, 384(1):66–76, 2007.
- [HV25] Lunjia Hu and Salil P. Vadhan. Generalized and unified equivalences between hardness and pseudoentropy. In *Theory of Cryptography (TCC)*, pages 258–288, 2025.
- [HVD15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.
- [IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *Symposium on Foundations of Computer Science (FOCS)*, pages 230–235, 1989.
- [IL90] Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Symposium on Foundations of Computer Science (FOCS)*, pages 812–821, 1990.
- [ILO20] Rahul Ilango, Bruno Loff, and Igor C. Oliveira. NP-hardness of circuit minimization for multi-output functions. In *Computational Complexity Conference (CCC)*, 2020.
- [Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference*, pages 134–147, 1995.
- [IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on Theory of Computing (STOC)*, pages 220–229. ACM, 1997.
- [IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001.
- [Jac97] Jeffrey C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *J. Comput. Syst. Sci.*, 55(3):414–440, 1997.
- [Juk12] Stasys Jukna. Boolean function complexity: advances and frontiers. *Algorithms and combinatorics*, 27, 2012.

- [JYT⁺23] Zhiying Jiang, Matthew Y. R. Yang, Mikhail Tsirlin, Raphael Tang, Yiqin Dai, and Jimmy Lin. "low-resource" text classification: A parameter-free classification method with compressors. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.
- [Kar24] Ari Karchmer. Agnostic membership query learning with nontrivial savings: New results and techniques. In *International Conference on Algorithmic Learning Theory (ALT)*, pages 654–682, 2024.
- [KC00] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.
- [KK09] Adam Kalai and Varun Kanade. Potential-based agnostic boosting. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 880–888, 2009.
- [KK25] Valentine Kabanets and Antonina Kolokolova. Chain rules for time-bounded Kolmogorov complexity. *Electron. Colloquium Comput. Complex.*, TR25-89, 2025.
- [KKO13] Adam R. Klivans, Pravesh Kothari, and Igor C. Oliveira. Constructing hard functions using learning algorithms. In *Conference on Computational Complexity (CCC)*, pages 86–97, 2013.
- [KL07] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman and Hall/CRC, 2007.
- [KL18] Pravesh K. Kothari and Roi Livni. Improper learning by refuting. In *Innovations in Theoretical Computer Science Conference (ITCS)*, pages 55:1–55:10, 2018.
- [KMR⁺94] Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. On the learnability of discrete distributions. In *Symposium on Theory of Computing (STOC)*, pages 273–282, 1994.
- [Kol65] Andrey N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of information transmission*, 1(1):1–7, 1965.
- [KS09] Adam R. Klivans and Alexander A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. *J. Comput. Syst. Sci.*, 75(1):2–12, 2009.
- [KSPH21] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, volume 34, pages 21696–21707, 2021.
- [KSS94] Michael J. Kearns, Robert E. Schapire, and Linda Sellie. Toward efficient agnostic learning. *Mach. Learn.*, 17(2-3):115–141, 1994.
- [KV94a] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.
- [KV94b] Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, January 1994.
- [KW16] Daniel M. Kane and Ryan Williams. Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits. In *Symposium on Theory of Computing (STOC)*, pages 633–643, 2016.
- [LCL⁺04] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul M. B. Vitányi. The similarity metric. *IEEE Trans. Inf. Theory*, 50(12):3250–3264, 2004.
- [Lev84] Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.
- [Lev03] Leonid A. Levin. The tale of one-way functions. *Probl. Inf. Transm.*, 39(1):92–103, 2003.

- [LM93] Luc Longpré and Sarah Mocas. Symmetry of information and one-way functions. *Inf. Process. Lett.*, 46(2):95–100, 1993.
- [LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.
- [LMP25] Yanyi Liu, Noam Mazon, and Rafael Pass. On white-box learning and public-key encryption. In *Innovations in Theoretical Computer Science Conference (ITCS)*, 2025.
- [LO22] Zhenjian Lu and Igor C. Oliveira. Theory and applications of probabilistic Kolmogorov complexity. *Bull. EATCS*, 137, 2022.
- [LOZ22] Zhenjian Lu, Igor C. Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 92:1–92:14, 2022.
- [LP20] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *Symposium on Foundations of Computer Science (FOCS)*, pages 1243–1254, 2020.
- [LP21] Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on $\text{EXP} \neq \text{BPP}$. In *Annual Cryptology Conference (CRYPTO)*, pages 11–40, 2021.
- [LP22] Yanyi Liu and Rafael Pass. On one-way functions from NP-complete problems. In *Conference on Computational Complexity (CCC)*, pages 36:1–36:24, 2022.
- [LR05] Troy Lee and Andrei E. Romashchenko. Resource bounded symmetry of information revisited. *Theor. Comput. Sci.*, 345(2-3):386–405, 2005.
- [LS24] Zhenjian Lu and Rahul Santhanam. Impagliazzo’s worlds through the lens of conditional Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 110:1–110:17, 2024.
- [LV19] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition*. Texts in Computer Science. Springer, 2019.
- [LW86] Nick Littlestone and Manfred Warmuth. Relating data compression and learnability. Technical report, University of California Santa Cruz, 1986.
- [LW95] Luc Longpré and Osamu Watanabe. On symmetry of information and polynomial time invertibility. *Inf. Comput.*, 121(1):14–22, 1995.
- [McM56] B. McMillan. Two inequalities implied by unique decipherability. *IRE Transactions on Information Theory*, 2(4):115–116, 1956.
- [MP24] Noam Mazon and Rafael Pass. Search-to-decision reductions for Kolmogorov complexity. *Electron. Colloquium Comput. Complex.*, 3:1–22, 2024.
- [MP25] Noam Mazon and Rafael Pass. Guest column: On cryptography and meta-complexity. *SIGACT News*, 56(2):63–101, June 2025.
- [MY16] Shay Moran and Amir Yehudayoff. Sample compression schemes for VC classes. *Journal of the ACM*, 63(3):1–10, 2016.
- [Nan21] Mikito Nanashima. A theory of heuristic learnability. In *Annual Conference on Learning Theory (COLT)*, 2021.

- [NDRT13] Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep K. Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in Neural Information Processing Systems*, volume 26, pages 1196–1204, 2013.
- [NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.
- [Oli19] Igor C. Oliveira. Randomness and intractability in Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 32:1–32:14, 2019.
- [OS17] Igor C. Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Computational Complexity Conference (CCC)*, pages 18:1–18:49, 2017.
- [OS18] Igor C. Oliveira and Rahul Santhanam. Pseudo-derandomizing learning and approximation. In *International Workshop on Randomization and Approximation Techniques (RANDOM)*, 2018.
- [OW93] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Israel Symposium on Theory of Computing Systems (ISTCS)*, pages 3–17, 1993.
- [Pas26] Rafael Pass. The value of a prompt: An LLM-relative Kolmogorov-complexity approach, 2026.
- [Pic25] Ján Pich. Learning algorithms from circuit lower bounds. *Comput. Complex.*, 34(1):6, 2025.
- [PV88] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, October 1988.
- [PVY99] Doron Peled, Moshe Y. Vardi, and Mihalis Yannakakis. Black box checking. In *Formal Methods for Protocol Engineering and Distributed Systems (FORTE/PSTV)*, pages 225–240, 1999.
- [PW25] Yury Polyanskiy and Yihong Wu. *Information theory: From coding to learning*. Cambridge University Press, 2025.
- [Raz87] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41:333–338, 1987.
- [Ris78] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, September 1978.
- [Ris84] Jorma Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, 30(4):629–636, July 1984.
- [RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.
- [RWC⁺19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019. Accessed: 2024-11-15.
- [Sch90] Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5:197–227, 1990.
- [Ser25] Rocco A. Servedio. The probably approximately correct learning model in computational learning theory. *CoRR*, abs/2511.08791, 2025.
- [SF14] Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. MIT Press, 2014.
- [SH96] Jürgen Schmidhuber and Stefan Heil. Sequential neural text compression. *IEEE Transactions on Neural Networks*, 7(1):142–146, 1996.
- [Sha48] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

- [Sim23] Meta-Complexity. Research Program at the Simons Institute for the Theory of Computing, UC Berkeley, Spring 2023. <https://simons.berkeley.edu/programs/Meta-Complexity2023>.
- [Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Symposium on Theory of Computing (STOC)*, pages 77–82, 1987.
- [Sol64a] Ray J. Solomonoff. A formal theory of inductive inference. Part I. *Information and Control*, 7(1):1–22, March 1964.
- [Sol64b] Ray J. Solomonoff. A formal theory of inductive inference. Part II. *Information and Control*, 7(2):224–254, June 1964.
- [SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.
- [ST17] Rocco A. Servedio and Li-Yang Tan. What circuit classes can be learned with non-trivial savings? In *Innovations in Theoretical Computer Science Conference (ITCS)*, pages 30:1–30:21, 2017.
- [SUV17] Alexander Shen, Vladimir A. Uspensky, and Nikolay Vereshchagin. *Kolmogorov complexity and algorithmic randomness*. American Mathematical Society, 2017.
- [TVZ05] Luca Trevisan, Salil P. Vadhan, and David Zuckerman. Compression of samplable sources. *Comput. Complex.*, 14(3):186–227, 2005.
- [TZJ⁺16] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *USENIX Security Symposium (USENIX Security)*, pages 601–618, 2016.
- [Vad12] Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012.
- [Vad17] Salil P. Vadhan. On learning vs. refutation. In *Conference on Learning Theory (COLT)*, pages 1835–1848, 2017.
- [Val84] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.
- [VL00] Paul M. B. Vitányi and Ming Li. Minimum description length induction, Bayesianism, and Kolmogorov complexity. *IEEE Transactions on Information Theory*, 46(2):446–464, March 2000.
- [Vol16] Ilya Volkovich. A guide to learning arithmetic circuits. In *Conference on Learning Theory (COLT)*, pages 1540–1561, 2016.
- [VZ12] Salil P. Vadhan and Colin Jia Zheng. Characterizing pseudoentropy and simplifying pseudorandom generator constructions. In *Symposium on Theory of Computing (STOC)*, pages 817–836, 2012.
- [WB68] C. S. Wallace and D. M. Boulton. An information measure for classification. *The Computer Journal*, 11(2):185–194, August 1968.
- [Wil14] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014.
- [WZTE18] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation. *CoRR*, abs/1811.10959, 2018.
- [Xia09] David Xiao. *New perspectives on the complexity of computational learning, and other problems in theoretical computer science*. Phd thesis, Princeton University, 2009.
- [Xia10] David Xiao. Learning to create is as hard as learning to appreciate. In *Conference on Learning Theory (COLT)*, pages 516–528, 2010.

- [Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.
- [YMT23] Yibo Yang, Stephan Mandt, and Lucas Theis. An introduction to neural data compression. *Foundations and Trends in Computer Graphics and Vision*, 15(2):113–200, 2023.
- [ZL70] Alexander K. Zvonkin and Leonid A. Levin. The complexity of finite objects and the algorithmic concepts of randomness and information. *UMN (Russian Math. Surveys)*, 25(6):83–124, 1970.
- [ZL77] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory*, 23(3):337–343, 1977.
- [ZL78] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inf. Theory*, 24(5):530–536, 1978.