Learning with distributional inverters

author names withheld

Editor: Under Review for ALT 2022

Abstract

We generalize the "indirect learning" technique of Furst et al. (1991) to reduce from learning a concept class over a samplable distribution μ to learning the same concept class over the uniform distribution. The reduction succeeds when the sampler for μ is both contained in the target concept class and efficiently invertible in the sense of Impagliazzo and Luby (1989). We give two applications.

- We show that $\mathsf{AC}^0[q]$ is learnable over any succinctly-described product distribution. $\mathsf{AC}^0[q]$ is the class of constant-depth Boolean circuits of polynomial size with AND, OR, NOT, and counting modulo q gates of unbounded fanins. Our algorithm runs in randomized quasi-polynomial time and uses membership queries.
- If there is a strongly useful natural property in the sense of Razborov and Rudich (1997) — an efficient algorithm that can distinguish between random strings and strings of non-trivial circuit complexity — then general polynomial-sized Boolean circuits are learnable over any efficiently samplable distribution in randomized polynomial time, given membership queries to the target funuction.

Keywords: List of keywords

1. Introduction

Simple concepts should be efficiently learnable. Exploring this intuition via formal measures of complexity — such as VC dimension, Littlestone dimension, and sample compression cost — drives progress in learning theory (Blumer et al., 1989; Littlestone, 1987; Moran and Yehudayoff, 2016). Recent work has taken an *algorithmic* perspective: some learning problems can be reduced to estimating the complexity of a given concept (Carmosino et al., 2016; Oliveira and Santhanam, 2017). This is different from characterizing the sample complexity of learning via a combinatorial dimension, because the relationship is inherently algorithmic.

However, some learning algorithms are distribution-specific: they only produce accurate hypotheses for a target function f under the *uniform distribution* over the domain. This is quite limiting, as most distributions of interest in nature are *not* uniform. For example, the accuracy of image classification is certainly not assessed under the uniform distribution on pixels. Formalizing this observation, the Probably Approximately Correct (PAC) setting of Valiant (1984) is distribution-free: it requires that a learning algorithm produce an accurate hypothesis over *any* given distribution.

In this paper, we consider the intermediate setting of learning over efficiently samplable distributions. We reduce the task of learning a concept class Λ over a samplable distribution μ to learning Λ over the uniform distribution and inverting a sampler for μ. As a concrete application of this reduction, we give the first membership-query learning algorithm over product distributions for $\mathsf{AC}^0[p]$ — constant-depth circuits with AND, OR, NOT, and counting modulo p gates for any fixed prime number p. This generalizes previous work that could only learn AC^0 circuits — which lack counting modulo p gates — over product distributions (Furst et al., 1991). Though learning AC^0 can be done with random examples, it is a strictly less expressive concept class than $\mathsf{AC}^0[p]$ (Razborov, 1987; Smolensky, 1987).

Theorem 1 $\mathsf{AC}^0[p]$ *is membership-query learnable over product distributions in randomized quasipolynomial time.*

In the general case, we strengthen connections between learning and complexity lower bounds. Algorithms that distinguish between random and circuit-compressible strings are called *natural properties*, because they are implicit in "natural" proofs of complexity lower bounds for Boolean circuits (Razborov and Rudich, 1997). If there is a natural property against general Boolean circuits, we can use it twice: first to learn circuits over the uniform distribution, and second to invert any efficiently samplable distribution over examples. This construction expands applicability of the "learning from natural properties" technique from only the uniform distribution to every polynomial-time samplable distribution (PSAMP) — arguably the most natural class of distributions over which to measure the accuracy of any hypothesis.

Theorem 2 *If there is a strongly useful* P*-natural property, then general polynomial-size circuits can be learned over any distribution in* PSAMP *using membership queries in randomized polynomial time.*

1.1. Our Approach

Our main technical lemma reduces learning over a distribution μ to learning over the uniform distribution and inverting a sampler for μ. To sketch the construction, let f be a target function, S_μ be the efficient machine that transforms uniformly random bits into samples from μ, and let I_μ be an inverter for S_μ. Consider the composed function $g(r) = f(S_\mu(r))$ — given coins for the sampler as input, g evaluates f over the resulting sample from μ. By assumption, we can learn g over the uniform distribution, getting a preliminary hypothesis circuit h. Our final hypothesis is $h'(x) = h(I_\mu(x))$. Intuitively, since $I_\mu(x)$ maps x back to sampler coins r such that $S_\mu(r) = x$ and h is a "good" hypothesis for g, the composed function h' will generally agree with f on inputs sampled from μ. This is a natural generalization of the "indirect approach" used by Furst et al. (1991) to learn AC^0 over product distributions; see Section 1.2 for a detailed comparison.

We generalize and extend the "indirect approach" by selecting an appropriate definition of "successful inversion" such that the simple reduction above is correct. We use *distributional inversion*, which requires an inverter for the function g to produce, on input y, *uniformly random* pre-images of y from $g^{-1}(y)$. This "nice" guarantee on the distribution over pre-images allows us to reduce to learning g over the uniform distribution.

Distributional inversion was introduced to show that one-way functions — easy to compute but hard to invert — are essential for cryptography (Impagliazzo and Luby, 1989). Instead, we use the definition and associated cryptographic techniques to give (conditional and unconditional) *upper bounds* in learning theory. Then, since from Carmosino et al. (2016) we already know (1) natural properties imply membership-query learning over the uniform distributon for general circuits and (2) membership-query learning of $\mathsf{AC}^0[p]$ over the uniform distribution, we work to produce expressive distributional inverters for natural classes of distributions each setting.

(1) Learning over PSAMP From Natural Properties. We compose existing results showing that natural properties can break any standard one-way function (Impagliazzo et al., 2018) with a uniform cryptographic construction of distributional one-way functions from standard one-way functions (Impagliazzo and Luby, 1989). Because the proofs of these results are implicitly constructive

— they give algorithms that reduce the task of finding uniformly distributed pre-images to finding *any* pre-image — we can use them to reduce distributional inversion for any samplable distribution to natural properties.

(2) Learning $\mathsf{AC}^0[p]$ Over Product Distributions. The above construction requires natural properties able to distinguish random strings from strings compressible by linear-depth circuits. But the natural properties that we actually know just distinguish $\mathsf{AC}^0[p]$-compressible from random strings — and $\mathsf{AC}^0[p]$ only contains *constant-depth* circuits! Even against TC^0 circuits, let alone log-depth, natural properties are unknown and indeed unlikely to exist (Naor and Reingold, 2004). So, instead, we identify a well-studied class of distributions — product distributions — for which distributional inverters and samplers exist unconditionally. Crucially, the samplers are computable in AC^0 — this ensures that the natural properties for $\mathsf{AC}^0[p]$ suffice to learn over the uniform distribution.

1.2. Related Work

Utility & Hardness of Compressability Testing To formalize "compressability testing," we refer to the Minimum Circuit Size Problem (MCSP): given the truth table of a Boolean function f and a number s, does f have a circuit of size at most s? It is a major open question to understand the hardness of MCSP — the problem is clearly in NP, but not known to be in P, and determining if MCSP is NP-hard is a major open question. The "simple" gadget reductions that suffice for textbook NP-hardness results cannot be used to show NP-hardness of MCSP (Murray and Williams, 2017). However, there are efficient randomized Turing reductions from factoring, graph isomorphism, and any problem with a Statistical Zero-Knowledge proof system to MCSP (Allender et al., 2006; Allender and Das, 2017). All these reductions exploit MCSP to invert efficiently-computable functions that encode information about the problem instance. We use this ability (as Lemma 10) in our reduction from inverting samplable distributions to natural properties.

Learning from MCSP Different variants of MCSP are useful for different learning tasks. With membership queries over the uniform distribution, there is in fact an *equivalence* between efficient compressability testing and efficient learning (Oliveira and Santhanam, 2017). If we have a *tolerant* natural property — which can detect strings that are just *close* to circuit-compressible strings — then we can learn over the uniform distribution even when some fraction of membership queries are adversarially corrupted (Carmosino et al., 2017). MCSP for *partial* truth tables — which allow some "don't care" entries — can even be used to learn from random examples (Ilango et al., 2020). Similar to the works above, these learning results all use (variants of) MCSP to break a pseudorandom object that encodes information about the target concept: they are distinguisher to predictor transformations. We combine both types of reductions to expand the class of learning problems that reduce to standard natural properties.

Learning vs. One-Way Functions The existence of one-way functions is related to (in)feasibility of learning in a variety of settings. Kearns et al. (1994) introduced the setting of learning a *distribution* — building an approximate sampler from random examples, instead of an approximate function. If one-way functions exist, then there are distributions that are easy to sample from but hard to learn in this setting. *Inductive inference* is the problem of observing an efficiently generated sequence and then predicting the next element (Solomonoff, 1964). Inductive inference (in an average-case setting) is possible if and only if one-way functions do not exist (Impagliazzo and Levin, 1990). Later work even showed that inductive inference for adaptively changing sequences

is possible if and only if one-way functions do not exist (Naor and Rothblum, 2006). Though we use some of the same relationships between cryptographic primitives, our results are for a different, PAC-type setting.

Learning Constant-Depth Circuits The techniques used in proving complexity lower bounds can often be re-purposed to build learning algorithms. Linial et al. (1989) used the lower bounds against AC^0 of Furst et al. (1984) to learn AC^0 over the uniform distribution. Then, Furst et al. (1991) generalized this approach to learn AC^0 over product distributions. Carmosino et al. (2016) used the lower bounds against $\mathsf{AC}^0[p]$ of Razborov (1987); Smolensky (1987) to learn $\mathsf{AC}^0[p]$ over the uniform distribution (with membership queries). It was therefore natural to ask if we can obtain learning for $\mathsf{AC}^0[p]$ over product distributions via similar ideas.

Furst et al. (1984) give two different techniques for learning AC^0 over product distributions. The first generalizes the underlying Fourier techniques of Linial et al. (1989) to account for evaluation on product distributions. This technique is not applicable to $\mathsf{AC}^0[p]$, which lacks the required Fourier concentration properties. We generalize *the analysis of* their second approach, which composes an AC^0-sampler for product distributions with the target function, and an inverter with the hypothesis. Essentially, we use a stronger definition of "successful inversion." They require only that the inverter produce a distribution that is close — as a vector of reals — to the uniform distribution. This suffices to extend Fourier-based learning to product distributions, but could fail for algorithms that use membership queries. Furst et al. (1984) noted that "it is not at all clear" how their second technique could be used on an arbitrary distribution. Our analysis using distributional inversion (Lemma 9) gives straightforward conditions under which the technique works for *any samplable* distribution.

2. Preliminaries

We denote by $\circ$ concatenation of strings, and by $\mathcal{U}_m$ the uniform distribution over m bits. Let $\mathcal{D}_p$ be a distribution over $\{0,1\}$ with p the probability of a 1. A product distribution $\mathcal{D} = \mathcal{D}_{p_1} \times \mathcal{D}_{p_2} \times \cdots \times \mathcal{D}_{p_n}$ is a distribution over $\{0,1\}^n$ where each coordinate x_i is independently 1 with probability p_i. We say that $\mathcal{D}_p$ is *concise* (has a concise description) if there exists a constant k such that $p = s/2^k$, where s is a k-bit binary string encoding an integer; we refer to k as the *precision* for p.

We will distinguish between uniform and non-uniform complexity classes. A complexity class Λ is generally defined by restricting Boolean devices by size, depth, or admissible gates. We furthermore say that a function $f\colon \{0,1\}^\star \to \{0,1\}^\star$ is in P-uniform Λ if there is a polynomial-time algorithm that, on input 1^n, prints a Λ-device that computes $f_n\colon \{0,1\}^n \to \{0,1\}^\star$, the "slice" of f on n-bit inputs.

2.1. Sampling & Inversion

Definition 3 ($t(n)$-Samplable Ensemble) *An ensemble $\mu = \{\mu_n\}_{n\in\mathbb{N}}$ is a sequence of probability distributions over binary strings of length n. The ensemble is $t(n)$-samplable if there is a machine S_μ running in $t(n)$ time such that, for every n and for every $x \in \{0,1\}^n$:*

$$\Pr_{r\sim\mathcal{U}}[S_\mu(1^n, r) = x] = \mu_n(x)$$

The class of "efficiently samplable" distributions is PSAMP, every poly-samplable ensemble. We will invert samplers for μ to learn over μ. That is, we will learn over distributions with samplers

that are *not* one-way functions (OWFs) — easy to compute but hard to invert. OWFs are fundamental objects required for cryptography (Yao, 1982; Goldreich, 2001). Theorems about OWFs are often proved via efficient reductions between inversion problems; these reductions are useful subroutines in our learning setting. Below we define two fundamental inversion problems by negating standard definitions of OWFs.

Definition 4 (Invertible Functions) *Let $f\colon \{0,1\}^\star \to \{0,1\}^\star$ be a function on bitstrings. We say that f is (η, A)-invertible if there exists a probabilistic polynomial-time oracle Turing Machine I running in time $t(n)$ such that:*

$$\Pr_{x\sim\mathcal{U}_n}\left[f(I^A(f(x))) = f(x)\right] \geq \eta$$

Whenever a probability is taken over a probabilistic algorithm, the coins used by the algorithm are implicitly part of the sample space. When η is $1/\mathsf{poly}(n)$, we call the inversion "weak" because it is not particularly likely, but still noticeable by an efficient machine. When η is $(1 - 1/\mathsf{poly}(n))$ we call the inversion "strong."

The above definition ignores the distribution on $f(x)$ pre-images produced by the inverter. It only prescribes the time complexity of inversion and the probability that *some* pre-image will be found. We will require that the inverter produce "well-distributed" (i.e., uniformly random) pre-images for any given input. To define this, we first give a strong information-theoretic notion of distributional closeness.

Definition 5 (Statistical Indistinguishability) *Probability distributions μ_n^0 and μ_n^1 on $\{0,1\}^n$ are **statistically indistinguishable within** δ, denoted $\mu_n^0 \equiv_\delta \mu_n^1$, if*

$$\forall T \subseteq \{0,1\}^n \left| \Pr_{x\sim\mu_n^0}[x \in T] - \Pr_{x\sim\mu_n^1}[x \in T] \right| \leq \delta$$

Then, we negate the definition of a distributionally one-way function (Impagliazzo and Luby, 1989; Impagliazzo, 1989) to define well-distributed inversion. Though these definitions are stated for general functions, our applications will always invert the sampler for a probability ensemble.

Definition 6 (Distributionally Invertible Function) *We say that a function $f\colon \{0,1\}^\star \to \{0,1\}^\star$ is (δ, A)-distributionally invertible if there is an efficient probabilistic oracle Turing Machine I such that the distributions $x \circ f(x)$ and $I^A(f(x)) \circ f(x)$ where $x \sim \mathcal{U}_n$ are statistically indistinguishable within δ for all but finitely many lengths n. That is,*

$$x \circ f(x) \equiv_\delta I^A(f(x)) \circ f(x) \;\; \textit{where } x \sim \mathcal{U}_n.$$

In particular, we'll need distributional inversion for all polynomially small statistical distances. So we say that a function f is $(1/\mathsf{poly}, A)$-distributionally invertible if, for every constant $c > 0$, we have that f is $(1/n^c, A)$-distributionally invertible.

2.2. Learning Model

Our learning model is a relaxation of Valiant's Probably Approximately Correct (PAC) setting Valiant (1984). We allow queries, target a specific distribution, and only require inverse-polynomial error and failure bounds. Formally, let f be a Boolean function. Our learners are allowed *membership queries* to f, meaning that it may query an input $x \in \{0,1\}^n$ and get back the value $f(x)$ from an oracle in unit time.

Definition 7 (Query learning over the ensemble μ) *Let Λ be any class of Boolean functions, and let $\mu = \{\mu_n\}$ be a probability ensemble. We say that Λ* is query-learnable over *μ if, for any error bound $\varepsilon \in \mathsf{poly}^{-1}(n)$ and failure bound $\delta \in \mathsf{poly}^{-1}(n)$, there exists an algorithm A that, given membership queries to any n-bit $f \in \Lambda$, outputs a hypothesis h such that $\Pr_{x\sim\mu_n}[h(x) \neq f(x)] \leq \varepsilon(n)$, with probability at least $1-\delta(n)$ over its internal randomness. The runtime of A is measured as a function $T = T(n,\ 1/\delta,\ 1/\varepsilon)$.*

Naturally, we say that Λ is learnable over a *class* of ensembles $\mathcal{D}$ — such as PSAMP — if Λ is learnable over every $\mu \in \mathcal{D}$. Note that a different algorithm may be used to learn over each distribution in the class, and indeed to guarantee each inverse-polynomial error and failure rate. However, our algorithms are uniform with respect to these parameters. Given the code of a sampler for μ and the functions ε and δ, we can efficiently print the code of A for learning over μ to error ε with failure probability δ.

2.3. Natural Properties

Let $\mathcal{F}_n$ be the collection of all Boolean functions on n input variables. A *combinatorial property* of Boolean functions is a sequence of subsets of $\mathcal{F}_n$, one for each n. We denote a circuit class by Λ, and a machine class by Γ.

Definition 8 (Natural Property) *A combinatorial property $\mathcal{R} = \{\mathcal{R}_n\}_{n\in\mathbb{N}}$ is Γ-Natural against Λ with density $\delta_n : \mathbb{N} \to [0,1]$ if it has the following properties:*

- ***Constructivity:*** *When $f \in \mathcal{F}_n$ is given as a truth table — a 2^n-bit string — the predicate $f_n \overset{?}{\in} \mathcal{R}_n$ is computable in Γ.*
- ***Largeness:*** *$|\mathcal{R}_n| \geq \delta_n \cdot |F_n|$*
- ***Usefulness:*** *For any sequence of functions f_n, if $f_n \in \Lambda$ then $f_n \notin \mathcal{R}_n$ for all but finitely many n.*

Denote by $\mathsf{SIZE}[s(n)]$ the set of functions computed by unrestricted Boolean circuits with at most $s(n)$ gates on n input bits. We then call a natural property $\mathcal{R}$ *strongly useful* if there is some a, $0 < a < 1$, such that $\mathcal{R}$ is useful against $\mathsf{SIZE}[2^{an}]$ and has density $\delta_n \geq 1/2$.

3. Learning Over μ Reduces to Learning Over $\mathcal{U}$ and Inverting μ

For efficiently samplable distributions μ, learning a concept class Λ over μ reduces to learning Λ over the uniform distribution and producing a distributional inverter for μ. To sketch the argument, let Λ be any class of functions that is closed under composition and contains the sampler for μ.

Then, the μ-sampler composed with the target concept f will also be a function $f_\mu \in \Lambda$ — and thus learnable over the uniform distribution. If we compose the output hypothesis C that results from learning f_μ with a distributional inverter I_μ for μ, inputs distributed according to μ will be indistinguishable from the coins used to sample from μ.

Then, we show that $C(I_\mu(\cdot))$ approximates f over the distribution μ, because C approximates f_μ over the uniform distribution and the inverter is "good." Intuitively, if the event "$C(I_\mu(\cdot))$ is wrong about f" occurs "too much" compared to the event "C is wrong about f_μ", we can use this gap as a statistical distinguisher for the ability of I_μ to invert — a contradiction.

Lemma 9 (Inversion to Shift Benchmark Distribution) *Let μ be any ensemble with a $(1/\mathsf{poly}, A)$-distributionally invertible sampler $S_\mu \in \mathsf{P}$-uniform Λ. If Λ is closed under composition and query-learnable over the uniform distribution, then Λ is query-learnable over μ.*

Proof Fix arbitrary $\alpha, \beta \in \mathsf{poly}^{-1}(n)$ to given error and failure bounds, as in Definition 7. Let f be any n-bit function in Λ, and $r_S(n)$ be the number of random bits required by sampler S_μ to simulate μ_n. Define the composed function $f_\mu \colon \{0,1\}^{r_S(n)} \to \{0,1\}$ as $f_\mu(w) = f(S_\mu(1^n, w))$. Because Λ is closed under composition, $f_\mu \in \Lambda$.

Furthermore, given membership query access to f and using the uniformity of S_μ, we can simulate membership queries to f_μ. Therefore, since Λ is query learnable over $\mathcal{U}$, we can efficiently approximate f_μ over the uniform distribution to within any inverse-polynomial error bound and failure rate. In particular, since $\alpha, \beta \in \mathsf{poly}^{-1}(n)$, there is a learner L that prints (with probability at least $1 - \beta$) a circuit C such that $\Pr_{w \sim \mathcal{U}}[C(w) \neq f_\mu(w)] \leq \alpha/2$. Our algorithm first runs L with simulated membership queries to produce such a C. The second part of our hypothesis is an appropriate inverter. As guaranteed by the $(1/\mathsf{poly}, A)$-invertability assumption and $\alpha \in 1/\mathsf{poly}(n)$, let I_μ be an $(\alpha/2, A)$-distributional inverter for S_μ. Denote by $r_I(n)$ the number of random bits required by I_μ to invert S_μ. Output $C'(x) = C(I_\mu^A(x))$ as our hypothesis for f over μ_n. Notice that the output hypothesis will require an oracle for A and random bits to run the inverter.

We bound the error of C' over μ by combining the error of C over $\mathcal{U}$ and the statistical indistinghisiability of I_μ. Define a statistical test $\mathrm{err}(w, x)$ which is equal to 1 if $C(w) \neq f(x)$ and 0 if $C(w) = f(x)$. Consider the following two distributions:

$$\mathcal{S} = w \circ S_\mu(1^n, w) \text{ where } w \sim \mathcal{U}_{r_S(n)}$$
$$\mathcal{I} = I_\mu(S_\mu(1^n, w), z) \circ S_\mu(1^n, w) \text{ where } w \sim \mathcal{U}_{r_S(n)} \text{ and } z \sim \mathcal{U}_{r_I(n)}$$

Because I_μ distributionally inverts S_μ, we know $\mathcal{S} \equiv_{\alpha/2} \mathcal{I}$. That is, suppressing the parameter 1^n for readability, we have the following bound for *any* statistical test, and so for err in particular:

$$\left| \Pr_{w,z \sim \mathcal{U}}[\mathrm{err}(I_\mu(S_\mu(w), z),\ S_\mu(w))] - \Pr_{w \sim \mathcal{U}}[\mathrm{err}(w, S_\mu(w))] \right| \leq \alpha/2.$$

Unrolling the definition of err, this becomes:

$$\left| \Pr_{w,z \sim \mathcal{U}}[C(I_\mu(S_\mu(w), z)) \neq f(S_\mu(w))] - \Pr_{w \sim \mathcal{U}}[C(w) \neq f(S_\mu(w))] \right| \leq \alpha/2.$$

By definition of f_μ, and because S_μ samples from μ, we obtain:

$$\left| \Pr_{z \sim \mathcal{U}, x \sim \mu_n}[C(I_\mu(x, z)) \neq f(x)] - \Pr_{w \sim \mathcal{U}}[C(w) \neq f_\mu(w)] \right| \leq \alpha/2.$$

Finally, we are left with a bound on the difference in error rates between C' and C over their respective input distributions.

$$\left| \Pr_{z\sim\mathcal{U},x\sim\mu_n}[C'(x,z) \neq f(x)] - \Pr_{w\sim\mathcal{U}}[C(w) \neq f_\mu(w)] \right| \leq \alpha/2$$

Re-arranging and using the error bound for L completes the proof:

$$\Pr_{z\sim\mathcal{U},x\sim\mu_n}[C'(x,z) \neq f(x)] \leq \alpha/2 + \alpha/2 \leq \alpha$$

Thus, f is learnable over μ with the error α and failure probability β.

■

4. Natural Properties $\Longrightarrow$ Query-Learning over PSAMP

If there is a strongly useful P-Natural property against $\mathsf{P/poly}$, then $\mathsf{P/poly}$ is query-learnable over PSAMP. To show this using the reduction above (Lemma 9), we just need to construct distributional inverters for all of PSAMP from natural properties. This will follow by efficient and uniform composition of hardness results about Natural Properties with classical relationships between various kinds of one-way functions. Below, we restate these results as reductions from the inversion problems defined in Section 2.1.

Lemma 10 (Weak Inversion Reduces to Natural Properties) *Let $\mathcal{R}$ be any strongly useful natural property. There exists $k \in \mathbb{N}$ such that any poynomial-time computable function on bitstrings is $(1/n^k, \mathcal{R})$-invertible.*

Lemma 10 first appeared as part of hardness results for testing resource-bounded Kolmogorov complexity (Allender et al., 2006). Later, it was used to show that if there is indistinguishability obfuscation against $\mathsf{BPP}^{\mathcal{R}}$, then $\mathsf{NP} \subseteq \mathsf{ZPP}^{\mathcal{R}}$ (Lemma 45 of Impagliazzo et al. (2018)). Note that the (weak) inversion success probability is a fixed inverse polynomial n^{-k}.

Lemma 11 (Strong Inversion Reduces to Weak Inversion) *Let f be an n-input Boolean function, and let q and p be any polynomials. Define the $t(n)$-**direct product** of f as $f'(x_1, \ldots, x_{t(n)}) = f(x_1) \circ \cdots \circ f(x_{t(n)})$. Suppose f' is $(1/q(m), A)$-invertible for some oracle A. Then, f is $(1 - 1/p(n), A)$-invertible.*

Lemma 11 first appeared implicitly in Yao (1982) — see Theorem 2.3.2 of Goldreich (2001) for a proof. Here, it is important for us that even a fixed inverse-polynomial weak inversion probability amplified to strong inversion probability $1 - p(n)$ for *any* inverse-poynomial $p(n)$ — because of Lemma 10, which only gives fixed inverse-polynomial probaility of inversion. The cost for this flexibility is just a larger polynomial runtime of the resulting inverter, proportional to the increased success probability.

Below, let $H_{n,m}$ be a family of hereditarily universal hash functions from n-bit strings to m-bit strings. We write $z\downarrow_i$ to mean "the first i bits of binary string z."

Lemma 12 (Distributional Inversion Reduces to Strong Inversion) *Let f be an n-input Boolean function. Define the c-**truncating hash** of f as $f'(h,i,x) = h \circ i \circ f(x) \circ h(x) \restriction_i$, where $i \in [m]$, $h \in H_{n,m}$, and $m = n + (6c+6)\log n$. Suppose f' is $(1 - 1/n^{6c}, A)$-invertible for some oracle A. Then, f is $(2/n^c, A)$-distributionally invertible.*

Lemma 12 was used to show that many cryptographic tasks are equivalent to the standard definition of a one-way function Impagliazzo and Luby (1989) — see Theorem 4.2.2 of Impagliazzo (1989) for a proof. We are now ready to obtain distributional inverters from natural proofs by composition of the above.

Lemma 13 (Natural Proofs Imply Distributional Inverters) *If there is a strongly useful P-natural property $\mathcal{R}$, then every μ in PSAMP is $(\mathsf{poly}, \mathcal{R})$-distributionally invertible.*

Proof Fix an arbitrary ensemble $\mu \in \mathrm{PSAMP}$, and arbitrary polynomial $p(n)$ as the target distribution and parameter of statistical distance for distributional inversion. Let f be the polynomial-time sampler associated with μ. Choose c such that $1/p(n) < 2/n^c$. Then, let h be the c-truncating hash of f and g be the n^{6c}-direct product of h. Because f is efficiently computable, so are h and g — we only took a polynomial-sized direct product of h and there are efficient universal hash families (Carter and Wegman, 1979).

We now unwind the above transformation of f to obtain a distributional inverter. First, Lemma 10 applies and g is $(1/n^k, \mathcal{R})$-invertible for some fixed k. Then, Lemma 11 applies to g and h, and so h is $((1-1/n^{6c}), \mathcal{R})$-invertible. Finally, Lemma 12 means that f is $(1/p(n), \mathcal{R})$-distributionally invertible. As $p(n)$ was an arbitrary polynomial and $\mu \in \mathrm{PSAMP}$ was arbitrary, the lemma follows. ■

Having obtained the required inverters from natural properties, we formally recall that sufficiently useful natural properties imply efficient learning of P/poly over $\mathcal{U}$.

Theorem 14 (Natural Proofs Imply Learning over $\mathcal{U}$ (Carmosino et al., 2016)) *If there is a strongly useful P-Natural property, then P/poly is query-learnable over $\mathcal{U}$ in randomized polynomial time.*

Theorem 2 now follows by combining the learning algorithm of Theorem 14 with the distributional inverters of Theorem 13 via Lemma 9 for each $\mu \in \mathrm{PSAMP}$.

5. $\mathsf{AC}^0[p]$ is Query-Learnable Over Bounded Product Distributions

We already know that $\mathsf{AC}^0[p]$ is query-learnable over $\mathcal{U}$ in randomized quasipolynomial time. So, to learn over product distributions via Lemma 9, we just need to construct samplers for product distributions that are both expressible in $\mathsf{AC}^0[p]$ and $(\mathsf{poly}, \emptyset)$-distributionally invertible. As we need the explicit description of the distribution (that is, the list of biases p_i), we require all our distributions to be concise.

We begin by explicitly sampling and inverting a single bit with bias $p \in (0,1)$, where p is given with precision k in binary. More precisely, we interpret a k-bit binary string encoding p as the numerator of a fraction with denominator 2^k, or, equivalently, as a dyadic k-bit number with exact k-bit expansion $bin(p) = b_1 \ldots b_k$, so that $p = \sum_{i=1}^{k} b_i \times 2^{-i}$

With the dyadic representation, to sample a single bit with probability p it is enough to sample a k-bit string r uniformly at random. This r is also interpreted as a dyadic number, and the sampler

returns 1 iff $r < p$ and 0 otherwise. Let $\texttt{Samp}(p)$ be the procedure that performs this operation; if we need to specify randomness r explicitly, we will use the notation $\texttt{Samp(p;r)}$. To sample from a product distribution, sample each bit with its respective bias, using fresh randomness for each sample; let $\texttt{ProdSamp}(p_1, \ldots, p_n)$ be the respective sampler. Note that both the Samp and ProdSamp are computable by AC^0 circuits with auxiliary inputs for random bits (sampling of individual bits by ProdSamp can be done in parallel, since they are sampled independently).

Now, we construct a distributional inverter for $\texttt{Samp}(p)$. Let $\mathcal{U}_k$ be the uniform distribution over k-bit binary strings used as randomness in $\texttt{Samp}(p)$. If we look at $bin(p)$ as a binary number, strings in $[0, bin(p) - 1]$ are preimages of 1, and strings in $[bin(p), 2^k - 1]$ are preimages of 0, so the inverter just needs to return a uniformly random k-bit binary number from the corresponding interval. As the construction uses rejection sampling, we also include a failure probability γ, which can be amplified to be arbitrarily small (such as $\gamma = 1/2^{2n}$). This inverter $\texttt{BitInv}(p, b, \gamma)$ will return a uniformly random preimage of a bit b sampled from D_p by $\texttt{Samp}(p)$, or fail (return $\perp$) with probability at most γ.

The bit inverter works as follows. To sample a preimage of 1, choose $C \leq k$ such that $2^{C-1} \leq p < 2^C$ and sample a uniformly random C-bit string y, then expand this string to k bits by adding $k - C$ leading 0s, if necessary; let r be the resulting k-bit string. Now r can be interpreted as a k-bit dyadic number in the range $[0 \ldots \frac{2^C-1}{2^k}]$. As $p \geq 2^{C-1}$, with probability $\geq 1/2$ it will be $r < p$. In that case, return r. Otherwise, sample another C-bit string y'. Repeat this process until either $r < p$ is obtained, or the number of rounds (attempts to sample) exceeds $\lceil \log 1/\gamma \rceil$. Since in each round r is sampled independently, and each round succeeds and returns a string with probability $\geq 1/2$, after $\lceil \log 1/\gamma \rceil$ rounds the failure probability will be $1/2^{\lceil \log 1/\gamma \rceil} \leq \gamma$. Finally, to sample a preimage of 0, sample $r < 1 - p$ using the same procedure as sampling $r < p$ above, then return $p + r$. As the failure probability in each round is always $\leq 1/2$ irrespective of b, p, k, γ, the same $\lceil \log 1/\gamma \rceil$ number of rounds suffices to make the probability of failure $\leq \gamma$ for sampling the preimage of either bit.

Note that this procedure does not have to be sequential: for a fixed γ and k, the bit inverter can be implemented as an AC^0 circuit with inputs b, p, together with $k \cdot \lceil \log 1/\gamma \rceil$ bits of randomness as auxiliary inputs, since essentially all it does is comparing k-bit numbers. The circuit will have $k + 1$ outputs, where one bit denotes success/failure and the rest are the bits of the first $r_i < p$.

Lemma 15 *$\texttt{BitInv}$ samples uniformly from the preimage of $\texttt{Samp}$. That is, for any p,*

$$\forall b \in \{0,1\} \forall r, r' \in \mathit{Samp}^{-1}(b) \;\; \Pr[\mathit{BitInv}(b, p, \gamma) = r] = \Pr[\mathit{BitInv}(b, p, \gamma) = r']$$

Proof Without loss of generality, suppose that $b = 1$. Then $r, r' < p$, and thus since $bin(p) < 2^C$ they can be written as $r = 0^{k-C}y$ and $r' = 0^{k-C}y'$. Moreover, r and r' are uniquely determined by y and y'. By construction, the probability that BitInv chooses a specific C-bit binary string is $1/2^C$. Thus, $\Pr[\texttt{BitInv}(b, p, \gamma) = r] = \Pr[\texttt{BitInv}(b, p, \gamma) = r'] = 1/2^C$ (note that $\Pr[\texttt{BitInv}(b, p, \gamma) = \perp] = \Sigma_{z \in \{0^{k-C} z_1 \ldots z_C | z \geq p\}} 1/2^C = (2^C - bin(p))/2^C$ ■

Finally, let $\mathcal{D} = \mathcal{D}_{p_1} \times \mathcal{D}_{p_2} \times \cdots \times \mathcal{D}_{p_n}$, where $\mathcal{D}_p$ is the distribution sampled by $\texttt{Samp}(p)$. For each p_i, let k_i be its precision (that is, number of bits in its dyadic representation). A preimage of $x = x_1 \ldots x_n$ sampled from $\mathcal{D}$ is a list of n strings $r_1, \ldots, r_n$, where each r_i is uniformly distributed among preimages of x_i (that is, k_i-bit strings that when used as randomness in $\texttt{Samp}(p_i)$ result in output x_i). The distributional inverter for $\mathcal{D}$, $\texttt{ProdInv}(x, p_1 \ldots p_n, \gamma)$, obtains each r_i by calling $\texttt{BitInv}(x_i, p_i, \gamma)$; if at least one call to BitInv fails, then ProdInv fails as well.

Lemma 16 *$\texttt{ProdInv}$ samples uniformly from preimage of $\texttt{ProdSamp}$. That is, for $x \in \{0,1\}^n$:*

$$\Pr[\texttt{ProdInv}(x) = (r_1, \ldots, r_n)] = \Pr[\texttt{ProdInv}(x) = (r'_1, \ldots, r'_n)]$$
$$\forall (r_1, \ldots, r_n), (r'_1, \ldots, r'_n) \in \texttt{ProdSamp}^{-1}(x)$$

Proof Let $(r_1, \ldots, r_n), (r'_1, \ldots, r'_n) \in \texttt{ProdSamp}^{-1}(x)$. As a preimage of each x_i is sampled using independent fresh random bits, for every $i \neq j$, $z_i \in \texttt{Samp}^{-1}(x_i)$, $z_j \in \texttt{Samp}^{-1}(x_j)$ $\Pr[\texttt{BitInv}(x_i) = z_i \wedge \texttt{BitInv}(x_j) = z_j] = \Pr[\texttt{BitInv}(x_i) = z_i] \cdot \Pr[\texttt{BitInv}(x_j) = z_j]$.

For each i, the distribution over preimages r_i of x_i sampled by $\texttt{BitInv}$ is uniform by lemma 15. Now, a direct product of uniform distributions is equivalent to a uniform distribution over respective direct products, completing the proof. ■

Lemma 17 *The failure probability of $\texttt{ProdInv}(x, p_1 \ldots p_n, \gamma)$ is at most γn, for $\gamma < 1/n$.*

Proof As all calls to $\texttt{BitInv}$ have to be successful for $\texttt{ProdInv}$ not to fail, the probability of success of $\texttt{ProdInv}$ is $(1-\gamma)^n$. Using Taylor series approximation and the $\gamma < 1/n$ assumption, $(1-\gamma)^n \geq 1 - \gamma n$. Thus, the probability of failure of $\texttt{ProdInv}$ is at most $1 - (1-\gamma)^n \leq 1 - (1 - \gamma n) = \gamma n$. ■

With this, we obtain a distributional sampler and inverter for product distributions computable by a family of AC^0 (provided all biases p are representable using constant precision). Setting $\gamma = 2^{-n}$, $\texttt{ProdInv}$ succeeds with probability exponentially close to 1, and uses $n^2 \cdot \max_i k_i = O(n^2)$ random bits to sample each $x \in \mathcal{D}$. Thus, we can combine it with the Carmosino et al. (2017) learning algorithm for $\mathsf{AC}^0[p]$ over the uniform distribution to learn $\mathsf{AC}^0[p]$ over product distributions.

Lemma 18 (Distributional Inverters for concise product distributions) *Concise product distributions have distributional inverters. Moreover, these inverters are computable in AC^0 with auxiliary random bits.*

5.1. Learning $\mathsf{AC}^0[p]$ with distributional inverters

For any prime p, the class $\mathsf{AC}^0[p]$ is closed under composition with AC^0 functions. Since the paired inverter is distributional, this satisfies the requirements for applying our reduction from samplable learning to uniform learning. We now formally recall that $\mathsf{AC}^0[p]$ is indeed learnable over the uniform distribution.

Theorem 19 ((Carmosino et al., 2016)) *For every prime p, the class $\mathsf{AC}^0[p]$ is query-learnable over $\mathcal{U}$ in randomized quasi-polynomial time.*

Theorem 1 now follows by combining the learning algorithm of Theorem 19 with the distributional inverters of Lemma 18 via Lemma 9 for each μ a concise product distribution.

6. Conclusions & Future Directions

A key difference between the learning algorithms we obtain from natural properties and the general PAC setting is the quantifier ordering. Our learning algorithms depend on the distribution; PAC learning requires that a single algorithm work for all possible distributions. If a *universal* distributional inverter can be constructed from a natural property, then this would further close the gap between the PAC setting and the learning problems that reduce to natural properties. This is a natural direction for future investigations, especially given recent progress in constructing one-way functions from hardness assumptions about compression problems related to the Minimum Circuit Size Problem (Liu and Pass, 2021).

We have expanded the scope of the "indirect" method for reducing from learnability over a samplable distribution to over uniform. Product distributions are just the most natural and well-studied example of a target distribution for which the technique works unconditionally. What other interesting ensembles of probability distributions have efficient samplers and distributional inverters?

References

Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. *Inf. Comput.*, 256:2–8, 2017. doi: 10.1016/j.ic.2017.04.004. URL https://doi.org/10.1016/j.ic.2017.04.004.

Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. doi: 10.1137/050628994. URL https://doi.org/10.1137/050628994.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989. doi: 10.1145/76359.76371. URL https://doi.org/10.1145/76359.76371.

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

Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Agnostic Learning from Tolerant Natural Proofs. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*, volume 81 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:19, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-044-6. doi: 10.4230/LIPIcs.APPROX-RANDOM.2017.35. URL http://drops.dagstuhl.de/opus/volltexte/2017/7584.

Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979. doi: 10.1016/0022-0000(79)90044-8. URL https://doi.org/10.1016/0022-0000(79)90044-8.

Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Syst. Theory*, 17(1):13–27, 1984. doi: 10.1007/BF01744431. URL https://doi.org/10.1007/BF01744431.

Merrick L. Furst, Jeffrey C. Jackson, and Sean W. Smith. Improved learning of ac^0 functions. In Manfred K. Warmuth and Leslie G. Valiant, editors, *Proceedings of the Fourth Annual Workshop on Computational Learning Theory, COLT 1991, Santa Cruz, California, USA, August 5-7, 1991*, pages 317–325. Morgan Kaufmann, 1991. URL http://dl.acm.org/citation.cfm?id=114866.

Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, Cambridge, 2001.

Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. Np-hardness of circuit minimization for multi-output functions. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 22:1–22:36. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi: 10.4230/LIPIcs.CCC.2020.22. URL https://doi.org/10.4230/LIPIcs.CCC.2020.22.

Russell Impagliazzo. *Pseudo-random generators for cryptography and for randomized algorithms*. PhD thesis, University of California, Berkeley, 1989.

Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 812–821. IEEE Computer Society, 1990. doi: 10.1109/FSCS.1990.89604. URL https://doi.org/10.1109/FSCS.1990.89604.

Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235. IEEE Computer Society, 1989. doi: 10.1109/SFCS.1989.63483. URL https://doi.org/10.1109/SFCS.1989.63483.

Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The power of natural properties as oracles. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi: 10.4230/LIPIcs.CCC.2018.7. URL https://doi.org/10.4230/LIPIcs.CCC.2018.7.

Michael J. Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. On the learnability of discrete distributions. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 273–282. ACM, 1994. doi: 10.1145/195058.195155. URL https://doi.org/10.1145/195058.195155.

Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle*

Park, North Carolina, USA, 30 October - 1 November 1989, pages 574–579. IEEE Computer Society, 1989. doi: 10.1109/SFCS.1989.63537. URL https://doi.org/10.1109/SFCS.1989.63537.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.*, 2(4):285–318, 1987. doi: 10.1007/BF00116827. URL https://doi.org/10.1007/BF00116827.

Yanyi Liu and Rafael Pass. Cryptography from sublinear-time average-case hardness of time-bounded kolmogorov complexity. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 722–735. ACM, 2021. doi: 10.1145/3406325.3451121. URL https://doi.org/10.1145/3406325.3451121.

Shay Moran and Amir Yehudayoff. Sample compression schemes for VC classes. *J. ACM*, 63(3): 21:1–21:10, 2016. doi: 10.1145/2890490. URL https://doi.org/10.1145/2890490.

Cody D. Murray and R. Ryan Williams. On the (non) np-hardness of computing circuit complexity. *Theory Comput.*, 13(1):1–22, 2017. doi: 10.4086/toc.2017.v013a004. URL https://doi.org/10.4086/toc.2017.v013a004.

Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004. doi: 10.1145/972639.972643. URL https://doi.org/10.1145/972639.972643.

Moni Naor and Guy N. Rothblum. Learning to impersonate. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 649–656, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143926. URL https://doi.org/10.1145/1143844.1143926.

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

A. 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(4):333–338, 1987. ISSN 1573-8876. doi: 10.1007/BF01137685. URL http://dx.doi.org/10.1007/BF01137685.

Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. doi: 10.1006/jcss.1997.1494. URL https://doi.org/10.1006/jcss.1997.1494.

Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82. ACM, 1987. doi: 10.1145/28395.28404. URL https://doi.org/10.1145/28395.28404.

Ray J. Solomonoff. A formal theory of inductive inference. part I. *Inf. Control.*, 7(1):1–22, 1964. doi: 10.1016/S0019-9958(64)90223-2. URL https://doi.org/10.1016/S0019-9958(64)90223-2.

Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. doi: 10.1145/1968.1972. URL https://doi.org/10.1145/1968.1972.

Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982. doi: 10.1109/SFCS.1982.45. URL https://doi.org/10.1109/SFCS.1982.45.