# Computational Hardness of Learning

Anish Jayant

Fall 2024

The disciplines of cryptography and learning theory each characterize the space spanned by computational tractability and information-theoretic limits using largely distinct tools. Fundamental notions in public-key cryptography and pseudorandomness rely on simple mathematical constructs that are assumed to be difficult to invert with limited computational resources, despite being easily invertible mathematically. This manifests in the form of (length-preserving) *one-way functions*:

**Definition 0.1** (One-wayness)**.** A function $f : \{0,1\}^n \to \{0,1\}^n$ is one-way if it is computable in poly-time and, for any probabilistic polynomial time algorithm $\mathcal{A}$ and polynomial $p$,

$$\Pr[f(\mathcal{A}(f(x))) = f(x)] \leq \frac{1}{p(n)}.$$

where $x$ is drawn u.a.r from $\{0,1\}^n$.

This definition captures *average-case* hardness; instead of showing $\exists x \in \{0,1\}^n$ for which $\mathcal{A}$ inverts $f(x)$ poorly, like in standard NP reductions, it shows a result $\forall x$, without needing pathological construction. It follows that the existence of an $f$ satisfying Definition 0.1 is one of the strongest conjectures in computer science, directly implying $\mathsf{P} \neq \mathsf{NP}$ and even $\mathsf{RP} \neq \mathsf{NP}$. Following the construction in Kearns and Vazirani (1994), we show that the one-way assumption in conjunction with learning theory shows that many computational models (like the polynomial-size Boolean circuits, Theorem 3.2) are "inherently unpredictable." At a high level, this means that any algorithm $\mathcal{A}$ with limited computational resource cannot simulate even a polynomial-size Boolean circuit! This result can be further strengthened to *log-depth*, polynomial size circuits and even neural networks, though just slightly out of the scope of this report. We conclude by mentioning some other popular perspectives on the limitations of learning.

## 1 Learnability in Classification

Like statistics inverts data-generating processes by reasoning about distributions (e.g. parameter estimation), learning theory reverses decisions by thinking about explanatory *function classes*. The foundational work of Valiant (1984) motivates this perspective as understanding a set of behaviors: whether a gene contributes to a disease, an ad is clicked, or a stock rises or falls.

In (binary classification) learning terminology, a function class $\mathcal{F} \subset \{0,1\}^{\mathcal{X}}$ is called an *inductive hypothesis* and typically represents some prior information about the data-label relationship.

In *proper learning*, the setup we start with here, the learning procedure must output some $\hat{f} \in \mathcal{F}$ using access to $m$ samples $(x_i, y_i) \in \mathcal{X} \times \{0,1\}$ drawn i.i.d. from some distribution $\mathcal{D}$ such that

$$\Pr_{x\sim\mathcal{D}}[\hat{f}(x) \neq f^*(x)] - \inf_{f\in\mathcal{F}} \Pr_{x\sim\mathcal{D}}[f(x) \neq f^*(x)] \overset{m\to\infty}{\longrightarrow} 0, \tag{1}$$

where $f^*(x) = y$ for all $x \in \mathcal{X}$. In words, it must select a classifier $\hat{f}$ from $\mathcal{F}$ using only our limited view of $m$ such that it performs as well the best classifier in $\mathcal{F}$ asymptotically. We also invoke the *realizable* assumption, which ensures $f^* \in \mathcal{F}$, making the second term in Eq. (1) as 0.

Notice that the answer to this question is *distribution independent*, meaning that we need not assume the structure of $\mathcal{D}$. Though this seems too vague at first, we'll see that this notion is actually the most natural and fundamental. To start, we consider the simplest non-trivial learning problem:

**Example 1.1** (Learning Thresholds)**.** Consider the function class $\mathcal{F} = \{1(x \geq c) | c \in \mathbb{R}\}$ of left-thresholds. We receive a sample set $S = \{(x_i, y_i)\}_{i=1}^m$ drawn according to some distribution over $\mathcal{X} = \mathbb{R}$ and labeled such that $y_i = f^*(x_i) = 1(x_i \geq c^*)$, for some $c^* \in \mathbb{R}$.

The paradigm of *empirical risk minimization* observes that $f^*$ has no error on $S$, so we should narrow our search of $\hat{f}_S$ only to members of $\mathcal{F}$ that satisfy this as well. Clearly, it's possible (though intractable) to find such a hypothesis via a brute-force search, but we can be more tactful using structural properties of $\mathcal{F}$. Let $S_0$ be the set of $x$-values in $S$ that earned label 0, and $S_1$ be the remainder; for any $(x_0, x_1) \in S_0 \times S_1$ we must have $x_0 < x_1$. Thus, if we let $c = \min S_1$, then $\hat{f}(x) = 1(x \geq c)$ classifies $S$ perfectly.

It remains to analyze how $\hat{f}$ performs relative to the true distribution. Note that $c > c^*$ since $c \leq c^*$ implies some $x_1 \in S_1$ such that $x_1 \leq c^*$, a contradiction under realizability. Thus, if $\hat{f}$ incurs error larger than $\epsilon > 0$, it follows

$$\Pr_{x\sim\mathcal{D}}[\hat{f}(x) \neq f^*(x)] = \Pr_{x\sim\mathcal{D}}[x \in [c^*, c)] > \epsilon.$$

Since $c = \min S_1$ it follows that none of the $m$ samples in $S$ came from $[c^*, c)$, an event with probability at most $(1-\epsilon)^m \leq e^{-\epsilon m} \leq \delta$ using the independence of draws. Therefore, if $m \geq \log(1/\delta)/\epsilon$, it follows that $\Pr_{x\sim\mathcal{D}}(\hat{f}(x) \neq f^*(x)) \leq \epsilon$ with probability $1-\delta$. The language of this result indeed shows the class of thresholds is properly learnable:

**Definition 1.2** (PAC Learning)**.** A class $\mathcal{F}$ is *learnable* (under representation $\mathcal{H}$) if, for any $f^* \in \mathcal{F}$, there exists some function $m_{\mathcal{F}} : (0,1)^2 \to \mathbb{N}$ and learning algorithm given $m \geq m_{\mathcal{F}}(\epsilon, \delta)$ samples drawn i.i.d from $\mathcal{D}$ which returns a hypothesis $h \in \mathcal{H}$ such that $\Pr_{x\sim\mathcal{D}}[h(x) \neq f^*(x)] \leq \epsilon$ with probability at least $1-\delta$. When $\mathcal{H} = \mathcal{F}$, we say $\mathcal{F}$ is *properly* learnable.

Since its introduction, Definition 1.2 has been shown to be satisfied by a variety of natural hypothesis classes like halfspaces, 3-CNFs, and even neural networks, in addition to the class of thresholds (Example 1.1). In addition, it is satisfied by *all* finite classes (i.e., whenever $|\mathcal{F}| < \infty$). Thus, as most constructs in cryptography are grounded in finite groups, these encryption schemes are learnable with modest sample complexity.

However, our discussion so far has purely information-theoretic, failing to capture the *computational* complexity of learning. Though our idea for $\hat{f}_S$ in Example 1.1 can be implemented in $O(m_{\mathcal{F}}) = O(\log(1/\delta)/\epsilon)$, we heavily exploited the problem's geometric structure, which may

not be available in general. This motivates the following definition of *efficient learning*, which strengthens the prior notion with a limitation on runtime (and implicitly, sample complexity).

**Definition 1.3.** A class $\mathcal{F}$ is *efficiently learnable* (under representation $\mathcal{H}$) if all conditions in Definition 1.2 hold for a learning algorithm that runs in $O(p(\log(1/\delta), 1/\epsilon))$ for some polynomial $p$. Since reading a sample takes time, it follows immediately that the sample complexity need also be polynomial in $\log(1/\delta)$ and $1/\epsilon$.

Interestingly, efficient learnability also depends on how we frame the learning problem, a phenomenon called *representation-dependent hardness*. For example, though 3-CNFs and 3-DNFs represent the same set of Boolean functions, only 3-CNFs are efficiently learnable wheras learning 3-DNFs happens to be NP-complete, via 3-Coloring Kearns et al. (1987). Note that this is not a contradiction, as the conversion from a 3-DNF to 3-CNF may take exponential time. Thus, the impossibility of efficiently learning 3-DNFs can be sidestepped through improper learning, specifically by returning a 3-CNF formula instead. In the remainder of this report, we show that one-way functions imply an even stronger result, whose subtlety we can now appreciate: a class that's impossible to efficiently learn *regardless* of representation.

# 2 Discrete Cube Root Assumption

We begin by introducing a special case of RSA encryption, where the public key $(e, N)$ has fixed exponent $e = 3$ which doesn't divide $\phi(N) = (p-1)(q-1)$. Recall the RSA encryption function $f_N$ behaves as $x \mapsto x^3 \pmod N$ and is a permutation of $\mathbb{Z}_N^*$. To see this, we produce its inverse function $f_N^{-1} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$, which show a bijection. Recall that there exists $d$ that $3d \equiv 1 \pmod{\phi(N)}$ since $3 \nmid \phi(N)$. Thus,

$$f_N(x)^d \equiv (x^3)^d \equiv x^{3d} \equiv x^1 \pmod N$$

using Euler's $x^{\phi(N)} \equiv 1 \pmod N$. Therefore, the function $x \mapsto x^d \pmod N$ behaves as a *cube root* modulo-$N$. The security of RSA lies with finding $d$, which is suspected to be at least as hard as factoring (i.e., computing $\phi(N)$) in average-case.

**Definition 2.1** (Discrete cube root)**.** Let $N = pq$ be an $n$-digit number such that $3 \nmid \phi(N)$, and $p, q$ be primes of similar length. Let $f_N : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ behave as $x \mapsto x^3 \pmod N$ and the *discrete cube root* modulo-$N$ be $f_N^{-1}(y) = y^d$.

Finally, we are poised to present the main hardness assumption, in line with Definition 0.1, which posits $f_N(x)$ is hard to invert *on average*:

**Assumption 2.2** (DCRA)**.** Consider uniform distribution over pairs $(p, q)$ such that $pq$ is $n$-bits and $3 \nmid (p-1)(q-1)$. Let $N = pq$ be selected via this distribution and $x \in \mathbb{Z}_N^*$ be selected uniformly at random. Then, for all algorithms $A \in \mathsf{PPT}$ and all polynomials $p(\cdot)$,

$$\Pr[f_N(A(f_N(x))) = f_N(x)] \leq \frac{1}{p(n)}$$

where the randomness is over $N, x$, and $A$.

With Assumption 2.2 in hand, we hope that binarizing $f_N^{-1}$ will produce a similar result in the framework we've already established. But, how can we reduce permutations of $\mathbb{Z}_N^*$ to classification learning? For a fixed $N$, a simple trick is to collect the $i$-th binary digit, $f_{N,i}^{-1}(x)$. Indeed,

$$f_N^{-1} = f_{N,1}^{-1} \| f_{N,2}^{-1} \| \cdots \| f_{N,n}^{-1}$$

where $\|$ denotes concatenation. So, if we have all $f_{N,i}^{-1}$, we can easily recover $f_N^{-1}$ and vice versa. Additionally, this binarization naturally generates hypothesis class $\mathcal{C}_N = \{f_{N,i}^{-1} | i \in [n]\}$ where $\mathcal{C}_N \subset \{0,1\}^{\mathbb{Z}_N^*}$ is clearly finite in size. Putting this together for (finitely many) satisfactory $N$, we have $\mathcal{C} = \bigcup \mathcal{C}_N$ models all possible discrete cube root digit functions! Thus it follows from finite classes being learnable that $\mathcal{C}$ is indeed learnable.

**Theorem 2.3.** Under DCRA, $\mathcal{C}$ is not learnable efficiently using *any* hypothesis class $\mathcal{H}$.

*Proof.* Suppose $\mathcal{C}$ is learnable efficiently with $m$ samples. We show this implies $f_N$ may be inverted for any $N$ with good probability, which immediately contradicts the average-case notion in Assumption 2.2. We generate $x_i \in \mathbb{Z}_N^*$ uniformly at random and apply the encryption function $y_i = f_N(x_i)$, and rearrange them as $(y_i, x_i) = (y_i, f_N^{-1}(y_i))$. By binarizing as described above into we achieve dataset

$$\begin{bmatrix} (y_1, f_{N,1}^{-1}(y_1)) & (y_1, f_{N,2}^{-1}(y_1)) & \cdots & (y_1, f_{N,n}^{-1}(y_1)) \\ (y_2, f_{N,1}^{-1}(y_2)) & (y_2, f_{N,2}^{-1}(y_2)) & \cdots & (y_2, f_{N,n}^{-1}(y_2)) \\ \vdots & \vdots & \ddots & \vdots \\ (y_m, f_{N,1}^{-1}(y_m)) & (y_m, f_{N,2}^{-1}(y_m)) & \cdots & (y_m, f_{N,n}^{-1}(y_m)) \end{bmatrix}$$

where each sample occupies a row and each digit inverse a column. We can simultaneously (and independently!) run $n$ subroutines of the purported efficient digit learning algorithm with parameter $(\epsilon, \delta) = (1/n^2, 1/n^2)$ and retrieve outputs $(h_1, h_2, ..., h_n) \in \mathcal{H}^n$. If the learners are successful, we have that for all $i \in [n]$, $h_i$ differs from $f_{N,i}^{-1}$ on at most $1/n^2$ of $\mathbb{Z}_N^*$. Let this "good" event be $G$, where $\Pr[G] = (1-\delta)^n \geq 1 - \delta n = 1 - 1/n$. A union bound brings

$$\Pr[h_1 \| h_2 \| \cdots \| h_n \neq f_N^{-1} | G] \leq \sum_{i=1}^{n} \Pr[h_i \neq f_{N,i}^{-1} | G] \leq n \cdot 1/n^2 = 1/n,$$

thus,

$$\begin{aligned} \Pr[A(f_N(x)) = x] &= \Pr[A(f_N(x)) = x | G] \Pr[G] + \Pr[A(f_N(x)) = x | G^c] \Pr[G^c] \\ &\geq \left(1 - \frac{1}{n}\right)^2 + 0 = 1 - \frac{2}{n} + \frac{1}{n^2} \geq 1 - 2/n. \end{aligned}$$

Therefore, $A$ agrees with $f_N^{-1}$ on all but $2/n$ of $\mathbb{Z}_N^*$, contradicting DCRA. □

# 3 Boolean Circuits Can't Be Efficiently Learned

Despite the tedious and pedantic construction of $\mathcal{C}$ in the previous section, the result of Theorem 2.3 is quite fundamental! In fact, it implies any computational model that is capable of *iterative multiplication*, multiplying several $n$-digit numbers in some predetermined order, is impossible to learn efficiently. We study an instructive computational model which satisfies this property.

A *Boolean circuit* is a directed-acyclic graph which computes a function $f : \{0,1\}^n \to \{0,1\}$, where its size is measured by the number of vertices. Each vertex has indegree 0, 1, or 2. In general, nodes may have arbitrary outdegree, except for a single vertex of outdegree 0, which is the desginated *output vertex*. Nodes which have indegree 0 are labeled by an input variable, those with indegree 1 are labeled with unary operator $\neg$, and those with indegree 2 are labeled with either $\vee, \wedge$. In such a manner, one can compute the Boolean circuit's output on $(x_1, x_2, ..., x_n)$ by placing these values on vertices with 0 indegrees and resolving the value of vertices next in a topological sort. By examining our construction, one can see that Boolean formulae, which use only logical operators and "(" and ")" to denote order, are precisely Boolean circuits with the graph structure of a tree. Conversely, a Boolean circuit may be transferred to a Boolean formula, but may suffer a blow-up in terms of size. The special property of circuits, which we exploit to show the following lemma, is that they can repeat computation efficiently, like a "subroutine" in programming.

**Lemma 3.1.** The multiplication of two $O(n)$ digit numbers can be computed by a circuit whose size is polynomial in $n$.

Simply consider the (unoptimized) way of multiplying $x_1, x_2$ each of $n$ bits via the "grade-school method." By multiplying $x_1 \cdot x_{2^i} \cdot 2^{i-1}$ for $i \in [n]$ and then adding the results, we reduce multiplication into the addition of $n$-many numbers of size at most $2n$, each of which is clearly doable with a circuit of $O(n)$ vertices. Note that the each multiplication itself takes $O(n)$ as well. As we execute polynomially many circuit subroutines each requiring polynomial size, the overall computation is indeed computable by a circuit whose size is polynomial in $n$.

**Theorem 3.2.** Under DCRA, the class of Boolean circuits of polynomial size is not efficiently learnable, regardless of representation.

To show the result, we argue that $f_N^{-1}$, where $N$ is $n$ bits and defined as before, can be computed by a polynomial-size Boolean circuit. This immediately implies $\mathcal{C}_N$ can be computed by such a Boolean circuit and thus Theorem 3.2 follows from Theorem 2.3 by reduction. Recall that $f_N^{-1}$ is simply $y \mapsto y^d$ where $d$ can be determined from just $\phi(N)$ (using Extended GCD), so simply constructing a polynomial size circuit that computes $y \mapsto y^d$, where $d$ is "hard-coded," suffices.

Note that approaching this computation head-on is too much, as $d$ may be $O(2^n)$ requiring *exponentially* many iterated multiplications. However, we can speed this up by first computing a "basis" for $y^d$,

$$\mathcal{S} = \{1, y \pmod N, y^2 \pmod N, y^4 \pmod N, ..., y^{2^k} \pmod N\},$$

where $k = \lfloor \log d \rfloor$. Note that construction $\mathcal{S}$ uses only $O(\log d)$ multiplications and can be

used to compute $y^d$ using binary expansion $d = d_k d_{k-1} \cdots d_0$ as $y^d = \prod_{i=0}^{k} \mathcal{S}_i^{d_i}$, again using $O(\log d)$ multiplications. By noting $\log d \leq n$ and Lemma 3.1, the protocol outlined in this proof is computable with a circuit of polynomial size.

# 4 Parting Shots

In this report, we present an important result in computational learning theory informed by the discrete cube-root assumption (2.2) from cryptography. Along the way, we touch on proper and improper learning, highlighting the role of *representation* in learning-related hardness results. We show that for the fundamental computational model of *polynomial-size Boolean circuits* is impossible to efficiently learn regardless of representation – a result in *representation-independent* hardness. In fact, exposing the limitations of learning is an active and vibrant area in both theoretical computer science and learning theory. We conclude by pointing out major research directions related to these lower bounds:

1. (**Memory**): Note that our definition for "efficient learning" (1.3) only constrains time-complexity. Indeed, in casual conversation, efficiency is often synonymous with running-time. But, in many learning (and cryptography!) applications, *memory* can often be as, if not more, restrictive. Thinking about memory and samples together (i.e., showing that a memory-restricted algorithm must use at least some lower bound of samples) is an active area of learning theory, and has shown sharp characterizations of fundamental tasks, like learning parities with noise (LPN) and convex optimization Garg et al. (2021); Marsden et al. (2024).

2. (**Communication**): In the age of "big data," the samples used to train learning models often reside on different machines. The naive solution of sending all data to a central server and computing a model with full information is often unrealistic – *communication* becomes the limiting factor! In fact, the rich field of *communication complexity* quantifies how many bits of information must be transferred between machines to compute functions involving inputs from multiple machines, Rao and Yehudayoff (2020). The works of Braverman et al. (2016); Dagan and Shamir (2018) extend this idea to canonical learning problems, showing that limited communication, like memory, means that more samples are needed to learn. In fact, under some restrictions, communication protocols can be shown to reduce to memory-bounded streaming algorithms!

3. (**Information-theory**): Another technique particularly common in *unsupervised learning* is to use a reduction to a difficult statistical primitive. For example, using the *planted-clique* conjecture of Bollobas and Erdös (1976), one can show *statistical* impossibility of learning the existence of some combinatorial structure! In Berthet and Rigollet (2013), for example, the task of detecting a sparse principal component analysis (PCA) is shown to reduce to detecting a planted clique, and thus undetectable without enough "signal."

# References


Berthet, Q. and Rigollet, P. (2013). Computational lower bounds for sparse pca.

Bollobas, B. and Erdös, P. (1976). Cliques in random graphs. *Mathematical Proceedings of the Cambridge Philosophical Society*, 80(3):419–427.

Braverman, M., Garg, A., Ma, T., Nguyen, H. L., and Woodruff, D. P. (2016). Communication lower bounds for statistical estimation problems via a distributed data processing inequality.

Dagan, Y. and Shamir, O. (2018). Detecting correlations with little memory and communication.

Garg, S., Kothari, P. K., Liu, P., and Raz, R. (2021). Memory-sample lower bounds for learning parity with noise.

Kearns, M., Li, M., Pitt, L., and Valiant, L. (1987). On the learnability of boolean formulae. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 285–295. Copyright: Copyright 2020 Elsevier B.V., All rights reserved.

Kearns, M. and Vazirani, U. (1994). *An Introduction to Computational Learning Theory*. MIT Press.

Marsden, A., Sharan, V., Sidford, A., and Valiant, G. (2024). Efficient convex optimization requires superlinear memory.

Rao, A. and Yehudayoff, A. (2020). *Communication Complexity: and Applications*. Cambridge University Press.

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