Cryptography and Learning

- Oracle access to an unknown f: query x, receive $f(x)$.
- Learning: predict $f(x)$ at fresh inputs from labeled examples.
- Cryptography: distinguish a keyed function F_s from a random function R.
- Information-theoretic learnability need not imply efficient learning.

Pseudorandom functions (PRFs)

A random function returns independent uniform values on new inputs, repeating the same value on repeated queries.

Experiment 0	$\approx$	Experiment 1
Choose $s \leftarrow \{0,1\}^k$. Reply to query x with $F_s(x)$.		Choose uniform random R. Reply to query x with $R(x)$.

Definition 1 (Pseudorandom function family)**.** *Let $\{F_s : \{0,1\}^n \to \{0,1\}^\ell\}_{s\in\{0,1\}^k}$ be a keyed family whose evaluation is polynomial-time computable. It is a (t,ε)-pseudorandom function family if $\forall$ oracle circuits A of size at most t,*

$$\left| \Pr_{s\leftarrow\{0,1\}^k}[A^{F_s}=1] - \Pr_{R}[A^{R}=1] \right| \le \varepsilon,$$

where $s \leftarrow \{0,1\}^k$ and R is a uniformly random function from $\{0,1\}^n$ to $\{0,1\}^\ell$.

- R: independent uniform answers on new inputs; consistent on repeats.
- Oracle queries may be adaptive; weak PRF: random $(x_i, F_s(x_i))$ pairs.
- Efficient = polynomial resources; secure = negligible distinguishing advantage.
- F, evaluation algorithm public; only sampled key s secret.

PAC learning

Definition 2 (PAC learning)**.** *An algorithm PAC learns a class C of functions if $\forall f \in C$, $\forall$ input distributions D, and $\forall \varepsilon, \delta > 0$, given independent pairs $(x, f(x))$ with $x \leftarrow D$, it outputs with probability at least $1-\delta$ a circuit h such that*

$$\Pr_{x\leftarrow D}[h(x) \neq f(x)] \le \varepsilon.$$

- Efficient learning: time $\mathrm{poly}(n, 1/\varepsilon, 1/\delta)$.
- Membership queries: may ask for $f(x)$ at chosen x.
- Proper: $h \in C$; improper: arbitrary efficiently evaluable h.
- PAC quantifies over every D; uniform D already suffices for hardness.

Example: linear functions

Let $\mathbb{F}$ be a finite field, $q = |\mathbb{F}|$. Unknown $\mathbf{s} \in \mathbb{F}^n$; independent uniform samples $\mathbf{a}_i \in \mathbb{F}^n$, labels $b_i = \mathbf{a}_i^\top \mathbf{s}$. Write $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)^\top$, $\mathbf{b} = (b_1, \ldots, b_m)^\top$; then $\mathbf{As} = \mathbf{b}$. If $\operatorname{rank}(\mathbf{A}) = n$, Gaussian elimination recovers $\mathbf{s}$. Over $\mathbb{R}$, noisy $\mathbf{b} = \mathbf{As} + \mathbf{e}$ admits least squares, e.g. $\widehat{\mathbf{s}} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$ when $\mathbf{A}$ has full column rank. Over $\mathbb{F}_2$, label flips give LPN (Lecture 2).

PRFs imply hardness of learning

Existence of PRFs implies P $\neq$ NP. Consider the NP language:

$$L_F = \Big\{((x_1, y_1), \ldots, (x_m, y_m)) : \ \exists s \in \{0,1\}^k \ \forall i, \ F_s(x_i) = y_i\Big\}.$$

- For distinct x_i, PRF outputs always satisfy this condition.
- Random outputs do so with probability at most $2^{k - m\ell}$.

- A PRF family has at most 2^k candidate functions h.
- With $O(k/\varepsilon)$ examples, $\Pr_{x \leftarrow D}[h(x) \neq f(x)] \leq \varepsilon$ holds with probability $\geq 2/3$.
- Information is not the obstacle. The challenge is efficient computation.

Theorem 1. *If $F = \{F_s\}$ is a PRF family and C is any class of functions containing F, then C is not efficiently PAC learnable.*

Proof. Enough to show this for the uniform distribution. Suppose an efficient learner L exists. Fix constants $\varepsilon, \delta > 0$ with $(1 - \varepsilon)(1 - \delta) > 1/2$.

Construct a distinguisher A with oracle O, either F_s or a random function R. Run L under the uniform input distribution: for each labeled example draw $x \leftarrow \{0,1\}^n$ and return $(x, O(x))$; forward membership queries to O. Once L outputs h, draw an independent $x^* \leftarrow \{0,1\}^n$ and accept iff $h(x^*) = O(x^*)$.

- If $O = F_s$, the learner sees its exact PAC experiment, so

$$\Pr[A^{F_s} = 1] \geq \Pr[L \text{ succeeds}] \Pr[h(x^*) = F_s(x^*) \mid L \text{ succeeds}] \geq (1 - \delta)(1 - \varepsilon).$$

- If $O = R$, L uses at most $m = \text{poly}(n)$ oracle inputs. "Fresh" x^*: previously unqueried.

$$\Pr[A^R = 1] \leq \Pr[x^* \text{ not fresh}] + \Pr[h(x^*) = R(x^*) \mid x^* \text{ fresh}] \leq \frac{m}{2^n} + 2^{-\ell}.$$

The distinguishing advantage is at least

$$(1 - \delta)(1 - \varepsilon) - 2^{-\ell} - \frac{m}{2^n} \geq (1 - \delta)(1 - \varepsilon) - \frac{1}{2} - o(1),$$

which is noticeable when $(1 - \delta)(1 - \varepsilon)$ is noticeably larger than $1/2$. □

- Reduction: an efficient learner yields an efficient PRF distinguisher.
- Exact simulation: reproduce the learner's view, not just its input format.
- Uniform inputs; examples and membership queries use the same oracle.
- Security model: computational power, oracle access, success criteria.
- Public: $\{F_s\}$, key distribution, evaluation; only sampled s is secret. Kerchkoffs principle.
- Improper learners ruled out; weak PRFs suffice without membership queries.

PRFs, PRGs, and OWFs

Let $\ell = \ell(k) > k$ and let $G : \{0,1\}^k \to \{0,1\}^\ell$ be polynomial-time computable.

Experiment 0	$\approx$	Experiment 1
Choose $s \leftarrow \{0,1\}^k$. Output $G(s)$.		Choose $r \leftarrow \{0,1\}^\ell$. Output r.

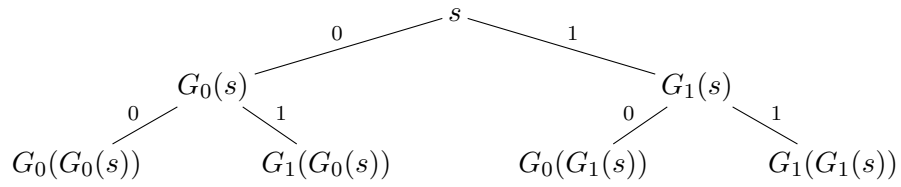
Definition 3 (Pseudorandom generator)**.** *G is a (t, ε)-PRG if every circuit A of size $\leq t$ distinguishes Experiments 0 and 1 with advantage at most ε.*

Existence of PRGs implies P $\neq$ NP. Consider the NP language:

$$L_G = \Big\{ y \in \{0,1\}^{\ell(k)} : \ \exists s \in \{0,1\}^k, \ G(s) = y \Big\}.$$

- PRF $\Rightarrow$ PRG: $G(s) = F_s(0) \| F_s(1) \| \cdots \| F_s(\ell)$ (choose enough distinct inputs for stretch).
- PRG $\Rightarrow$ PRF: Goldreich–Goldwasser–Micali construction.

Length-doubling $G(s) = G_0(s) \| G_1(s)$; both halves have k bits.



Follow the path $x_1, \ldots, x_n$ from the root:

$$F_s(x_1 \cdots x_n) = G_{x_n}\Big(G_{x_{n-1}}(\cdots G_{x_1}(s) \cdots)\Big) \in \{0,1\}^k.$$

- Size: $n \cdot \text{size}(G)$; depth: $n \cdot \text{depth}(G)$.
- Security: hybrid argument reduces a PRF distinguisher to breaking G.

One-way functions (OWFs)

An efficient f: given $y = f(x)$ for random x, finding any preimage is hard.

- PRG inversion yields a distinguisher: verify $G(s') = y$.
- OWF $\Rightarrow$ PRG (HILL); PRG $\Rightarrow$ PRF (GGM).

Theorem 2. *Pseudorandom function families exist if and only if one-way functions exist.*

- Computational asymmetry: one can generate $(x, f(x))$ but inversion is hard.
- A hard instance, potentially even for AGI, with solution x known by construction.

Circuit complexity of PRFs

- Size: number of gates; depth: sequential layers; basis: permitted gates.
- AC^0: polynomial size, constant depth, AND/OR/NOT.
- TC^0: polynomial size, constant depth, threshold gates.

A threshold gate takes bits $x_i \in \{0,1\}$ and integer weights $w_i \in \mathbb{Z}$, with integer threshold θ:

$$\mathbf{1}\left[\sum_i w_i x_i \geq \theta\right] = \begin{cases} 1, & \text{if } \sum_i w_i x_i \geq \theta, \\ 0, & \text{otherwise.} \end{cases}$$

- If C contains a PRF, no efficient learner succeeds on every $f \in C$.
- PRFs in simpler circuit classes yield hardness for simpler learning classes.

Theorem 3. *Under standard cryptographic hardness assumptions, there exist pseudorandom function families computable in* TC^0.

In contrast, AC^0 has low-degree Fourier structure that permits quasi-poly learning.

Theorem 4 (LMN, 1993)**.** *Any PRF family in* AC^0 *can be broken in quasi-poly time.*

Fourier analysis expresses a Boolean function as a weighted sum of parity tests on subsets of input bits. LMN shows that for AC^0 circuits, the terms involving many bits contribute very little. Estimating the remaining low-degree terms from random examples gives a quasipolynomial-time learner, and hence a distinguisher for any such PRF.

Neural networks as circuits

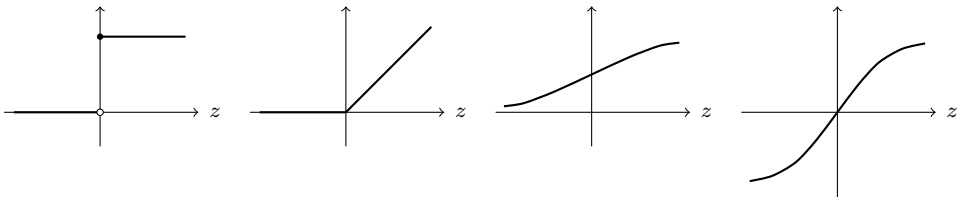
Neural networks compose layers of neurons with real-valued weights. A width-m layer on $\mathbf{x} \in \mathbb{R}^n$ is

$$\mathbf{x} \longmapsto \phi(\mathbf{A}\mathbf{x}), \qquad \mathbf{A} \in \mathbb{R}^{m \times n},$$

Here $\phi : \mathbb{R} \to \mathbb{R}$ acts coordinatewise (e.g., ReLU, sigmoid, step). Biases can be absorbed by appending a constant 1 to $\mathbf{x}$. Input, hidden, and output layers compose such maps.

- Size: number of neurons; depth: number of computational layers.
- The weights are fixed constants when evaluating a trained network.

Common activation functions ϕ:



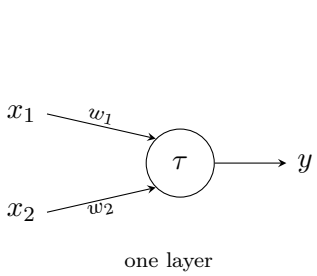
Step	ReLU	Sigmoid	Tanh
$\mathbf{1}[z \geq 0]$	$\max\{0, z\}$	$\frac{1}{1+e^{-z}}$	$\frac{e^{2z}-1}{e^{2z}+1}$

Representation power

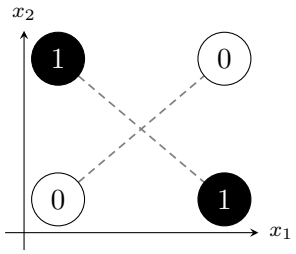
With Boolean inputs and step activation, a single output neuron computes

$$y = \mathbf{1}[w_1 x_1 + w_2 x_2 + b \geq 0].$$

Its positive inputs lie on one side of a line. In particular, it cannot compute XOR: no line with normal $(w_1, w_2) \in \mathbb{R}^2$ separates the 0-inputs from the 1-inputs.



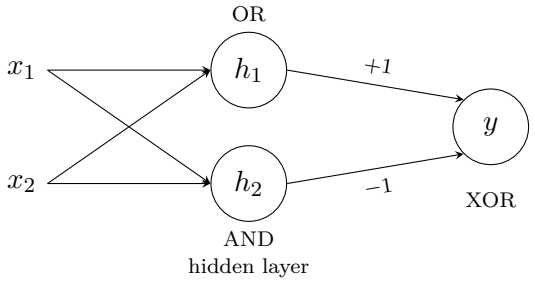
one layer



XOR labels

Two hidden step neurons compute OR and AND; the output subtracts the latter:

$$h_1 = \mathbf{1}[x_1 + x_2 \geq 1], \qquad h_2 = \mathbf{1}[x_1 + x_2 \geq 2], \qquad y = \mathbf{1}[h_1 - h_2 \geq 1] = x_1 \oplus x_2.$$



Thus adding a hidden layer increases the functions a network can represent.

Real weights and exact activations are mathematical idealizations. Implementations use B-bit weights and intermediate values, with specified rounding. For signed two's-complement $z = (z_{B-1}, \ldots, z_0)$, the most significant bit is the sign bit. Let

$$s = \mathrm{Step}(z) = 1 - z_{B-1}, \qquad \mathrm{ReLU}(z) = z\ \mathrm{Step}(z), \qquad y_i = z_i \wedge s.$$

Negate the sign bit z_{B-1} to obtain $s = \mathrm{Step}(z)$; then compute each output bit as $y_i = z_i \wedge s$. NOT and AND are threshold gates, giving a constant-depth ReLU circuit.

- Example (4 bits): $0011 \mapsto 0011$ $(3 \mapsto 3)$, $1101 \mapsto 0000$ $(-3 \mapsto 0)$.
- Sigmoid/tanh: approximate numerically at the chosen precision.
- Finite-precision forward passes give Boolean circuits, not necessarily TC^0.

Two ways to represent the same input-output computation:

1. Simulating neural networks by circuits: A forward pass with specified finite-precision arithmetic has a Boolean circuit implementation, not necessarily in TC^0.
2. Simulating threshold circuits by networks: Replace each threshold gate by a step neuron and each wire by a weighted connection. Hence every TC^0 circuit is representable by a constant-depth, polynomial-size step network.

Consequence for learning

1. **Representation.** Every TC^0 circuit is a constant-depth threshold network.
2. **Cryptography.** Under standard assumptions, PRFs exist in TC^0 (Theorem 3).
3. **Learning consequence.** TC^0 cannot be learned efficiently, even improperly.

Learning all such networks would break the PRFs they contain (Theorem 1).

Proposition 1. *Under standard cryptographic assumptions, constant-depth polynomial-size threshold networks are not efficiently PAC learnable.*

Modern networks: size versus sequential depth

- Qwen3-235B-A22B: 94 layers, 235 billion parameters.
- DINOv3 ViT-7B: 40 Transformer layers (blocks), about 6.7 billion parameters.
- T autoregressive tokens require T sequential forward passes.

T growing with input length need not lie in TC^0.

- Black-box distillation/extraction of a PRF teacher would contradict PRF security.
- White-box access to its parameters is outside this learning model.

Takeaways

- PRFs: low sample complexity but computational hardness of learning.
- Low-complexity PRFs: improper learning hardness for simple classes.