Last time: – Unconditional lower bounds on SQ learning:
some $\mathcal{C}$'s are eff. PAC learnable, but
are not eff. SQ learnable.
↳ (any $\mathcal{C}, \mathcal{D}$ s.t. $\mathcal{C}$ has many functions that are all uncorrelated with each other under $\mathcal{D}$)
Parity functions: are not eff. SQ learnable

– Start unit on inherent unpredictability: concept classes $\mathcal{C}$ that don't have eff. learning alg's using any eff. evaluable hypothesis class (based on cryptography, i.e. average-case hardness assumptions).

Today: computational hardness of learning
$\mathcal{C}$ = all poly(n)-size Boolean circuits
can learn with poly(n) many examples.
based on existence of pseudorandom function families
- mapping the boundary of efficient learnability
- start hardness of learning based on public-key cryptography (trapdoor 1-way permutations)

Admin:
- Rocco's OH this week: only on Zoom
- stay tuned for Rocco's OH next week rescheduled
- exam next Thurs 12/04

Questions?

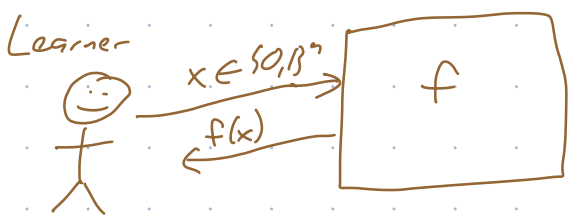
Pseudorand. + HoL

We'll argue: even an easy/powerful form of learning

can't learn $\mathcal{C}^* :=$ all poly(n)-size circuits over $\{0,1\}^n$, assuming
$\exists$ "pseudorandom functions"
"one-way functions".

- "unif. dist. learning": promised $\mathcal{D}$ is $\mathcal{U}$ = unif over $\{0,1\}^n$.

- Give learner "black-box" "membership query" "oracle" access to target fn $f: \{0,1\}^n \to \{0,1\}$.



learner chooses x's.

(Can choose unif. rand. $x \sim \mathcal{U}$).

Notation:
A = alg: "A^f" $\rightsquigarrow$ A given oracle access to f

PRFF (pseudorandom function family)
Pseudorandom fns
:
first, truly random functions.

"$f: \{0,1\}^n \to \{0,1\}$ is truly random":

f drawn unif. from $\mathcal{C}_{ALL}$ = all 2^{2^n} many fns $\{0,1\}^n \to \{0,1\}$.

(2^n rand. bits required.)

Let RAND = unif. dist. over $\mathcal{C}_{ALL}$.

"f is a pseudorandom fn" means:

(f ~ a family of 2^n fns (each is "simple" - computed by poly(n) size ckt), yet

f appears rand. to any poly(n) time observer.)

Def: Let $\mathcal{F}$ be a set of 2^n Bool fns $\{0,1\}^n \to \{0,1\}$

$\mathcal{F} = \{f_s : s \in \{0,1\}^n\}$.

(s = seed)

$\mathcal{F}$ is a PRFF if

1) (eff. computability) There's a $2n$ input ckt of poly(n) size s.t.

i.e.

each f_s has a poly size ckt.

2) (indistinguishability from truly random)

Let DIST be a poly(n) time alg. (proposed distinguisher) that has oracle access to a fn $\{0,1\}^n \to \{0,1\}$, + outputs "Random (R)" or "Pseudorandom (PR)". Then

$$\left| \Pr_{f \sim RAND}[DIST^{f} = \text{"PR"}] - \Pr_{s \sim \{0,1\}^n}[DIST^{f_s} = \text{"PR"}] \right| < \frac{1}{n^c}$$

(truly rand; pseudorand)

for every const. $c > 0$.

Fact: • if "one-way functions" exist, so do PRFF

• if factoring random* n-bit #s is hard on average,

then PRFF exist.

Thm: Suppose $\mathcal{F}$ is a PRFF.
Then there's no poly(n) time PAC learning alg. for $\mathcal{F}$, using any poly(n)-time evaluable $\mathcal{H}$, even if we only need to learn under $\mathcal{U}$ & we get oracle access to target fn.

Idea: if $\mathcal{F}$ were learnable, learning alg. would give a distinguisher: run it, get hyp h, check if h is acc.
fn pseudorand. : h will be acc :)
fn truly rand. : h won't be acc :(

Details: Let A be a hypothesized poly-time PAC learner for $\mathcal{F}$.
We'll get a distinguisher, violating PRFF property.

(either truly rand. or pseudo.)

Our dist. works as follows: given oracle access to f:

- run A^f on f using $\varepsilon = \delta = 0.01$. Get hyp h.
- draw unif. $z \sim \mathcal{U}$, eval. $f(z)$ & $h(z)$.
  Output "PR" if $f(z) = h(z)$, else "R".

Analysis:

1) Sps $f = f_s$ for some $s \overset{unif}{\sim} \{0,1\}^n$
   Then $\Pr_z[h(z) = f(z)] \geq 0.98$ ($\delta = \varepsilon = 0.01$, & I ran my learner for $\mathcal{F}$ on a fn from $\mathcal{F}$)

2) OTOH, suppose $f \sim RAND$.

Alg A saw f's output on $\text{poly}(n)$ of all 2^n inputs (b/c A was $\text{poly}(n)$-time);
if z is not one of those, $\Pr_f[f(z)=h(z)] = \frac{1}{2}$,

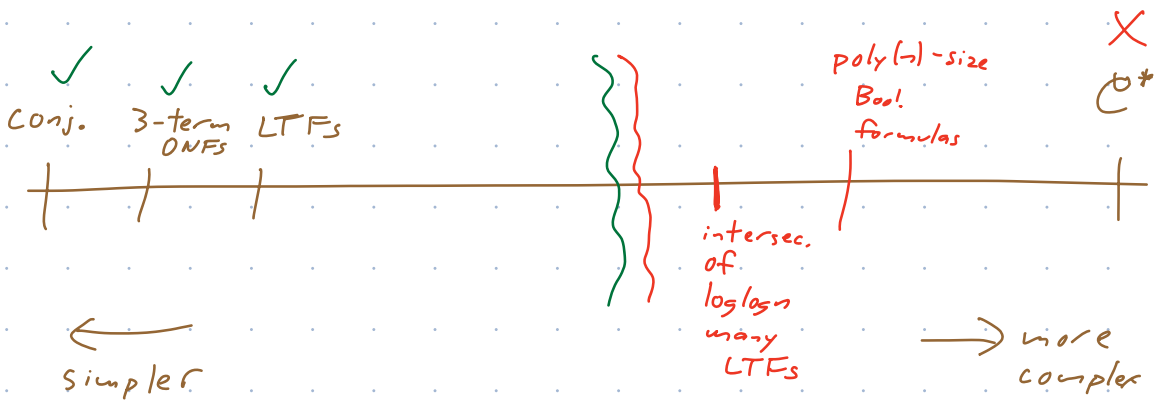
so overall $\Pr[f(z)=h(z)] \leq \frac{1}{2} + \frac{\text{poly}(n)}{2^n}$

So an eff learner A would violate that $\mathcal{F}$ is PRFF. ∎

Note: $|\mathcal{F}| = 2^n$, & each $f_s \in \mathcal{F}$ has a $\text{poly}(n)$-size ckt... $\text{poly}(n)$ examples are enough to PAC learn $\mathcal{F}$; but $\text{poly}(n)$ time is not enough.

So... the class C^* = all $\text{poly}(n)$ size ckts (comput.) is hard to learn

Q: how simple can fns be, & still be comput. hard to learn?



Next: a different reason why we get hardness of learning, not pseudorandomness...

Public Key Cryptography

Objects to discuss:

1) • permutations

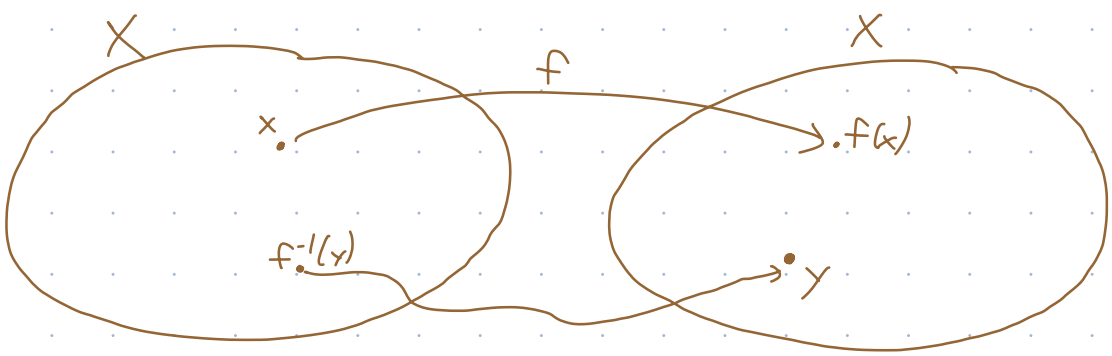
2) • one-way permutations

3) • trapdoor one-way permutations } $\Leftrightarrow$ public-key cryptosystems

Let $X = \{0,1\}^n$.

A permutation of X is a "1-1 & onto" function

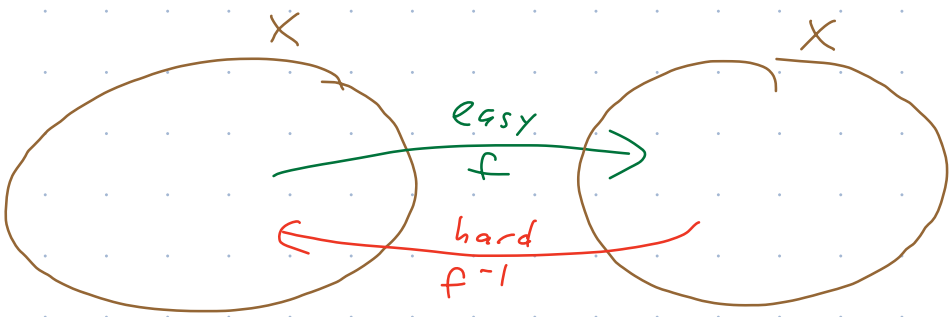
$f: X \to X$: for every $y \in X$, there is a unique $x \in X$ s.t. $f(x) = y$



Informal def. of one-way permutation:

a perm. $f: X \longrightarrow X$ s.t. ($\{0,1\}^n \longrightarrow \{0,1\}^n$)

- there's a poly(n) time alg. to compute f; but
- no poly(n) time alg. can successfully compute f^{-1} on even a $\frac{1}{\text{poly}(n)}$ frac. of pts in X.



Informal notion of "trapdoor one-way permutation": a one-way perm. s.t. if you have the secret "trapdoor" info (some poly(n) bit string), then it is easy to invert f.

Think of $N = p \cdot q$ p, q "secret" primes.

Everybody knows N; fn f is based on N;
↓ permut.

if know p, q it's easy to invert f (compute f^{-1}), but if you don't know p, q it's hard.

Next time: public-key crypto
∀
HoL from this.