PAC Learning: Complexity Treatment

Shafi Goldwasser

PAC- Terminology

- Alg is given sample S = {(x,y)} drawn from distribution P over instance space X, labeled by some Boolean *target* function *f* in *concept* class C, i.e y=f(x)
- [say x is positive example or x in f: f(x)=1]

- Goal: Algo produces hypothesis *h* in class H which is ε-close to *f* over distribution P.

Concepts vs. Representation

- C specifies both a concept and representation with an associated size
- Ex: $C=\{\langle x_1 \ldots x_n , f(x_1 \ldots x_n)=xor(a_1x_1 \ldots a_nx_n)\rangle\}$
  - with Representation: Boolean circuit (size n), or DNF (size is exponential in n)
- Ex: convex polytope
  - Representation: list of vertices or linear equations to define the faces of the polytope.

PAC learning; focus on complexity today

Let $X = \{X_n\} = \{0,1\}^n$ (or n-dim Euclidean space)

Let $C=\{C_n\}$ where C_n is a representation class over X_n

Algo A PAC-learns concept class C by hypothesis class H if for <u>any</u> unknown target f in C <u>any</u> distribution P over X, <u>any</u> $\varepsilon, \delta > 0$,

- A uses at most poly(n,$1/\varepsilon$,$1/\delta$, size(f)) examples and running time to produce h.
- with probability $1-\delta$

  h in H & h agrees with f with error at most ε.

  (allows failure w.prob δ: S may be non-representative)

Algorithmic/model issues

PAC model talks of learning C by H.

- In practice, most natural to fix H, allow C to be arbitrary.
  - H is under our control, target function isn't
  - Try to find reasonable h in H if one exists.

- Today will see negative results on what can cannot be PAC-learned when
  - C=H (proper),
  - $C \neq H$ (improper) and

Representation independence = H is Efficient to evaluate

- Efficient (Non-uniform) Algorithm:
  CKT={CKT_n} poly-size circuits over n inputs

- H should be efficient to evaluate : ∃an efficient algorithm that on $x \in X_n$ and $h \in H_n$ computes h(x) in time poly(n, |h|). Why? Otherwise makes little sense.

- Furthermore, consider only poly-time target function
  Schapire: any representation class C which is not polynomial time evaluable can not be learned

[PittValiant]: NP $\neq RP$ =>
Proper Learning Impossibility

- Claim: K-term-DNF not PAC-learnable by k-term DNP if NP $\neq$RP
- K-term DNP: $T_1 v T_2 v \ldots T_k$ where each T_i is conjunction of subsets of literals of $x_1 \ldots x_n$
- Size = 2kn

- Show Reduction from NP-complete problem of k coloring a graph to k-term- DNF. Here: k=3

Assumption: if P=NP, trivial to learn (take random examples

[PittValiant]: NP $\neq RP$ => Proper Learning Impossibility

- 3-term DNP: All $T_1 v T_2 v T_3$ where each T_i is conjunction of subsets of literals of $x_1 \ldots x_n$
- 3-coloring graph
  - Input: $G=(V,E)$
  - Output: 1 iff there exists col: V ->{0,1,2} s.t. if $(u,v) \in E$ then $col(v) \neq col(u)$

Reduction: We reduce graphs G to an instance of learning 3-term DNF f. Namely, a polynomial set of positive and negative examples of a formula and

[PittValiant]: NP ≠ RP => Proper Learning Impossibility

- To: learn 3-term DNP
- From deciding: 3-coloring graph

Reduction: We reduce graphs G to set of examples S to emulate an oracle over uniform distribution in S

s.t. G is 3-colorable if S is *consistent* with some k-term

Set $\varepsilon = 1/2|S|$. If there exists 3-term DNF consistent with S, learning algo will find h which is consistent with S (otherwise errs with 2ε)., If there is no 3-term DNF consistent with a 3-term DNF, algorithm will not find it..

[PittValiant]: NP ≠ *RP* => 3-term-DNF hard to properly learn

Given graph G=(V,E): construct set S of examples over n vars

- Positive: For every vertex i in V={1…n}, add example

(v(i),1) where v(i)=(1..101..1 ,with 0 only in the ith position. Namely, x_j=1 except for j=i where xi=0 makes the DNF true.

- Negative: For ever edge (i,j) , add example

(e(i,j),0) where e(i,j)=(11101110111) with 0 in i and j- positions.

Namely, when x_k=1 except that x_i=xj=1 makes the DNF false.

Claim: G is 3-colorable implies S is consistent with 3-term DNF

Pf: Fix legal coloring, let R={ red vertices i}, B={blue vertices}, **B**={black vertices}. Fix T_R ($T_{B,}$ T_B)=conjunction of variables whose are not colored R (B and C analogously). Then for each vertex i, colored R, v(i) satisfies T_R since it only gave 0 to variable x_i which does not appear in $T_{R.}$ Similarly for B and **B**

Furthermore, edge e(i,j) will not satisfy T_R (or $T_{R,}$ T_B) since both I and J cannot be colored red , one of them must appear in T_R and since its 0 it will make T_R false. T

[PittValiant]: NP $\neq$ *RP* => 3-term-DNF hard to properly learn

Claim 2: Suppose formula $T_R \vee T_B \vee T_B$ is consistent with S. Then G is 3-colorable

Pf: set the coloring as follows: set the color of vertex i is R if v(i) satisfies T_R (analogously T_B & T_B)

-Since the formula is consistent with S, every v(i) satisfies some T_c, and each vertex has a color.

-Furthermore, coloring is legal since if i and j are assigned same color then both v(i) and v(j) satisfy same T_c but i-th bit of v(i) is 0 and i-th bit of v(j) is 1 so they cant both appear in same T_c.

Also, v(i) and e(i,j) only differ in the j-th bit, so if v(i) satisfies T_c so does e(i,j) and (i,j) is not an edge.

[PittValiant]: Its all about representation

Claim 2: k-term DNF is learnable by k-CNF

Claim [Valiant]:k-CNF is properly learnable by k-CNF.

Corollary: k-term DNF can be learnable by a k-CNF.

[PittValiant]: More impossibilities

Claim 2: K-Boolean-threshold not PAC-learnable

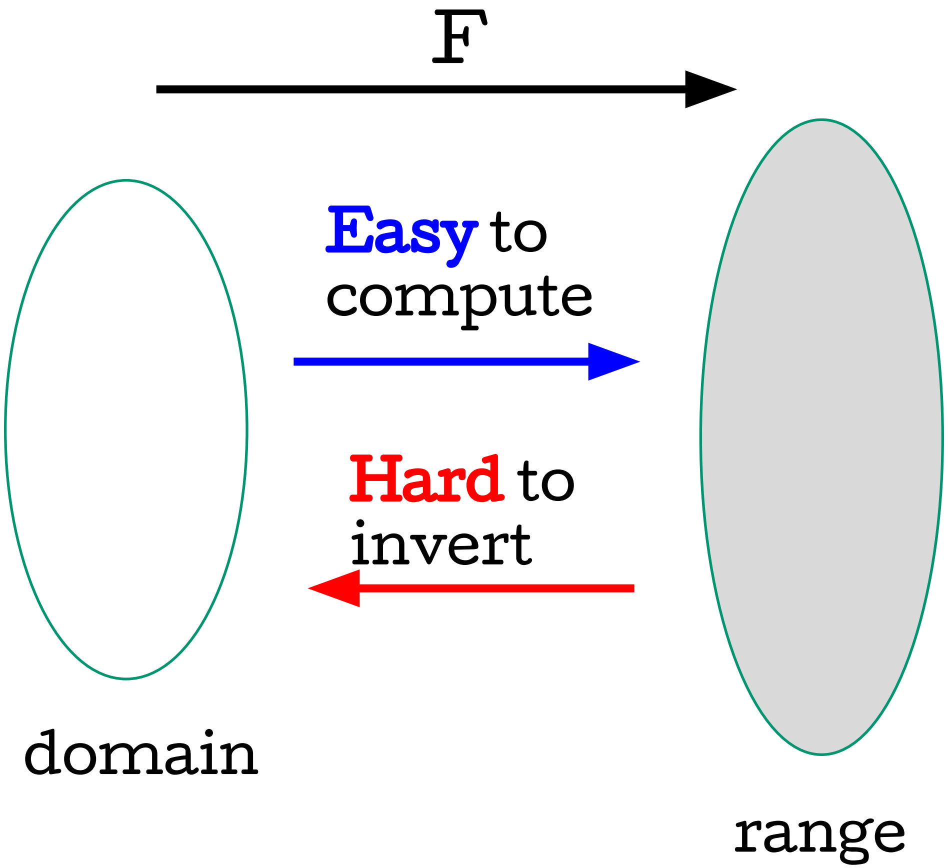
- K-Boolean threshold:
  - Fix y in $\{0,1\}^n$.
  - Then all x s.t. inner product xy>k mod 2
- Reduction from 0/1 integer programming which is NP-complete

[GoGoMi] How about Hardness which is Representation Free?

- Construct a concept class C
- Show that its hard to learn independently of H, i.e for any H which is polynomial time circuit family

- Assumption: One-Way Functions exist

One-way Functions (Informally)



Def: One-way Functions

A function (family) $\{F_n\}_{n\in\mathbb{N}}$ where $F_n: \{0,1\}^n \rightarrow \{0,1\}^{m(n)}$ is one-way if for every p.p.t. adversary A, there is a negligible function μ s.t.

$$\Pr[A(1^n, y) = \boldsymbol{x' \ s.t. \ y = F_n(x')}] \leq \mu(n)$$

probability taken over $x \leftarrow \{0,1\}^n; y = F_n(x)$;

- Can always find *an* inverse with unbounded time

but should be hard with probabilistic polynomial time

Special cases: **One-way Permutations**:

One-way Functions: Candidates

$G(a_1, \ldots, a_n, x_1, \ldots, x_n) = (a_1, \ldots, a_n, \sum_{i=1}^{n} x_i a_i \bmod 2^{n+1})$

G(p,q)=pq. Factoring problem

One-way functions candidates are abundant in nature.
Can construct a universal one-way function .

G^{-1}: intuitively hard to “learn”
To put in our framework, One Way Boolean Functions?

Hardcore Bits

If F is a one-way function, we know it's hard to compute a pre-image of $F(x)$ for a randomly chosen x.

But… you may be able to compute some bit of x

Exercise: There are one-way functions for which it is easy to compute the first half of the bits of the inverse.

Nevertheless, there has to be a hardcore set of hard to invert inputs. Thus: Does there necessarily exist some bit of x that is hard to compute?

Hardcore Bits

If F is a one-way function, we know it's hard to compute a pre-image of $F(x)$ for a randomly chosen x.

But… you may be able to compute some bit of x

Exercise: There are one-way functions for which it is easy to compute the first half of the bits of the inverse.

Does there exist <u>some bit</u> of x <u>that is hard to guess with probability non-negligibly better than 1/2</u>?

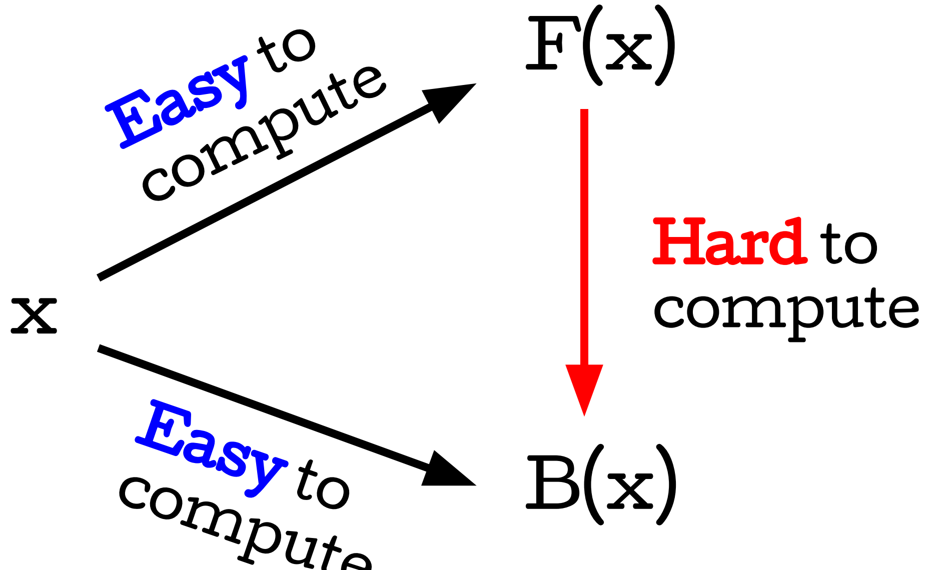
- Any bit can be guessed correctly w.p. 1/2

From Hardcore Bits to Hardcore Predicates

For any function (family) $F: \{0,1\}^n \rightarrow \{0,1\}^m$, a function $B: \{0,1\}^n \rightarrow \{0,1\}$ is a hardcore predicate if for every p.p.t. adversary A, there is a negligible function μ s.t.

$$\Pr[A(y) = B(x)] \leq \frac{1}{2} + \mu(n)$$

Prob taken over $x \leftarrow \{0,1\}^n; y = F(x)$.



Every OWF Has an Associated Hard Core Predicate [GL]

Let F be a one-way function.
Let $\{B_r : \{0,1\}^n \to \{0,1\}\}$ where

$$B_r(x) = \langle r, x \rangle = \sum_{i=1}^{n} r_i x_i \bmod 2$$

be a collection of predicates (one for each r).
Then, a *random* B_r is hardcore predicate for F.
For every PPT A, there is a negligible function μ s.t.

$$\Pr[A(F(x), r) = B_r(x)] \le \frac{1}{2} + \mu(n)$$

Prob taken over $x \leftarrow \{0,1\}^n; r \leftarrow \{0,1\}^n$

<u>Interpretation</u>: For every one-way function F, there is a related one-way function $F'(x,r) = (F(x), r)$ which has a *deterministic* hardcore predicate B(x,r)= $\sum_{i=1}^{n} r_i x_i \bmod 2$.

A concept which is Representation-Free Hard to Learn

- Let F be a family of one-way functions
- Let B be the associated hard-core Boolean predicate for F

- Concept C_n={(f(x), B(x): f in F_n}
  - Easy to compute if you know x
  - Hard to learn if you don't know x
  - Reduction: if can (weakly) learn B can invert f, contradiction!
  - Note: can even generate samples. Take z in domain of f, then {f(z), B(z)} is in C.

What about membership queries?

- Q: Are there concepts *C* which are hard to learn even if you can ask for (x,g(x)) for x of your choice?
- A: yes

- Theorem [GoGoMi]

  One-Way Functions $\Rightarrow$

  Pseudo Random Functions $\Rightarrow$

Boolean Pseudo Random Functions(PSRF) [GGM86]

•

F_n = Collection of indexed functions

f_s:{0,1}^{n} $\Longrightarrow$0,1} is pseudo-random if

– [Poly Time Evaluation given s] Given s, can compute f_s (x) is efficiently computable

– [Impossible to guess without s]

No adversary can distinguish between

(x, f_s (x)) for x of its choice, and

(x, U) (truly random function values).

Boolean Pseudo Random Functions(PSRF)

-

F_n = collections of indexed Boolean functions $f_s:\{0,1\}^n \Rightarrow \{0,1\}$ is pseudo-random if

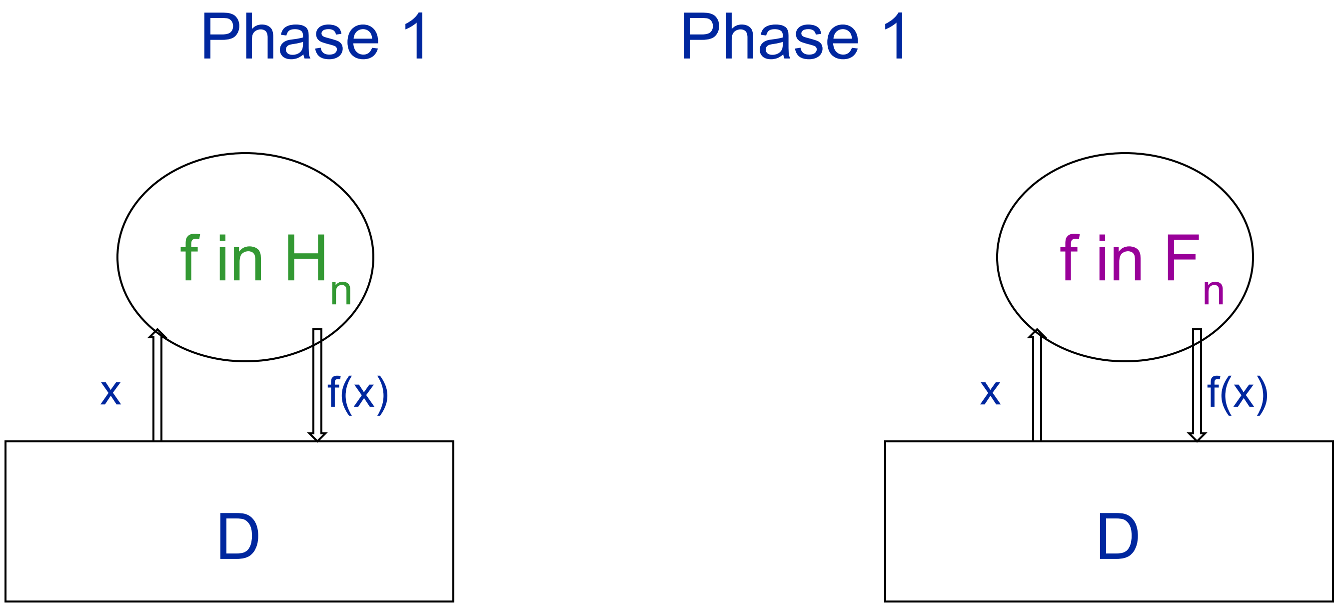
- [[Poly Time Evaluation given s] Given s, $f_s(x)$ is efficiently computable
- [Impossible to guess without s]

  For all PPT query-algorithms D^f, for all sufficiently large n

  $|prob(D^f(1^n) = 1:\ f$ is a random Boolean function on $\{0,1\}^n) - prob(D^f(1^n) = 1): f \in F_n) = negl(n)$

NOTE: D^f makes polynomial queries to f

Pseudo-Random F is indistinguishable from Random



Prob (D^f says 1 in Phase 2) ≈ Prob (D says 1 in phase 2)

Existence of PSRF's

Theorem: If one-way functions exist, then collections of pseudo random functions exist

Proof: Start with length doubling strong pseudo random generator (PRG) $G:\{0,1\}^n \to \{0,1\}^{2n}$

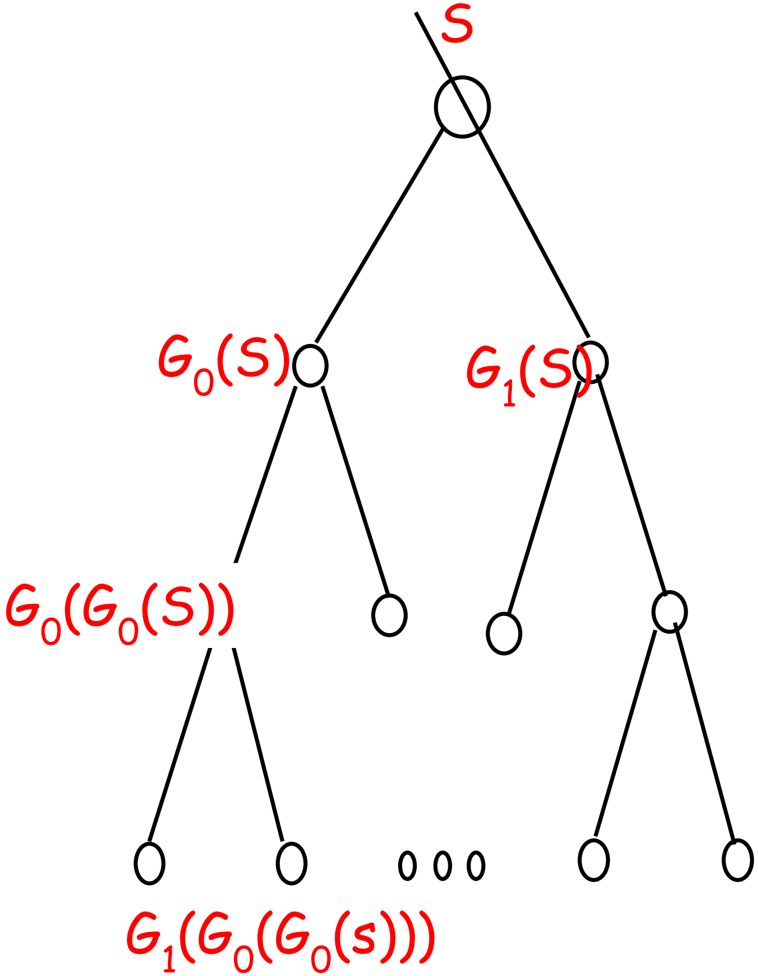
A function $G: \{0,1\}^n \rightarrow \{0,1\}^{t(n)}$ is a strong pseudorandom generator if no p.p.t. can distinguish between $G(U_n)$ and $U_{t(n)}$.

U_n= uniform distribution on n bits.

$U_{t(n)}$= uniform distribution on t(n) bits.

EX: G(x) = f(x)|B(x) for hard core predicate B, and t(n)=n+1

Next. PSRF tree Like Construction



$G_0(s)$ = Run PRG $G:\{0,1\}^n \rightarrow \{0,1\}^{2n}$ on seed s and output the <u>first</u> n output bits

$G_1(s)$ = Run a PSRG $G:\{0,1\}^n \rightarrow \{0,1\}^{2n}$ o
seed s and output the <u>2nd</u> n output bits

$G_{00}(s) = G_0(G_0(s))$

$G_{01}(s) = G_1(G_0(s))$ …

$G_x(s) = G_{xnxn-1\ x1}(s) = G_{xn}(G_{xn-1}\ (\ldots G_{x1}(s)).$

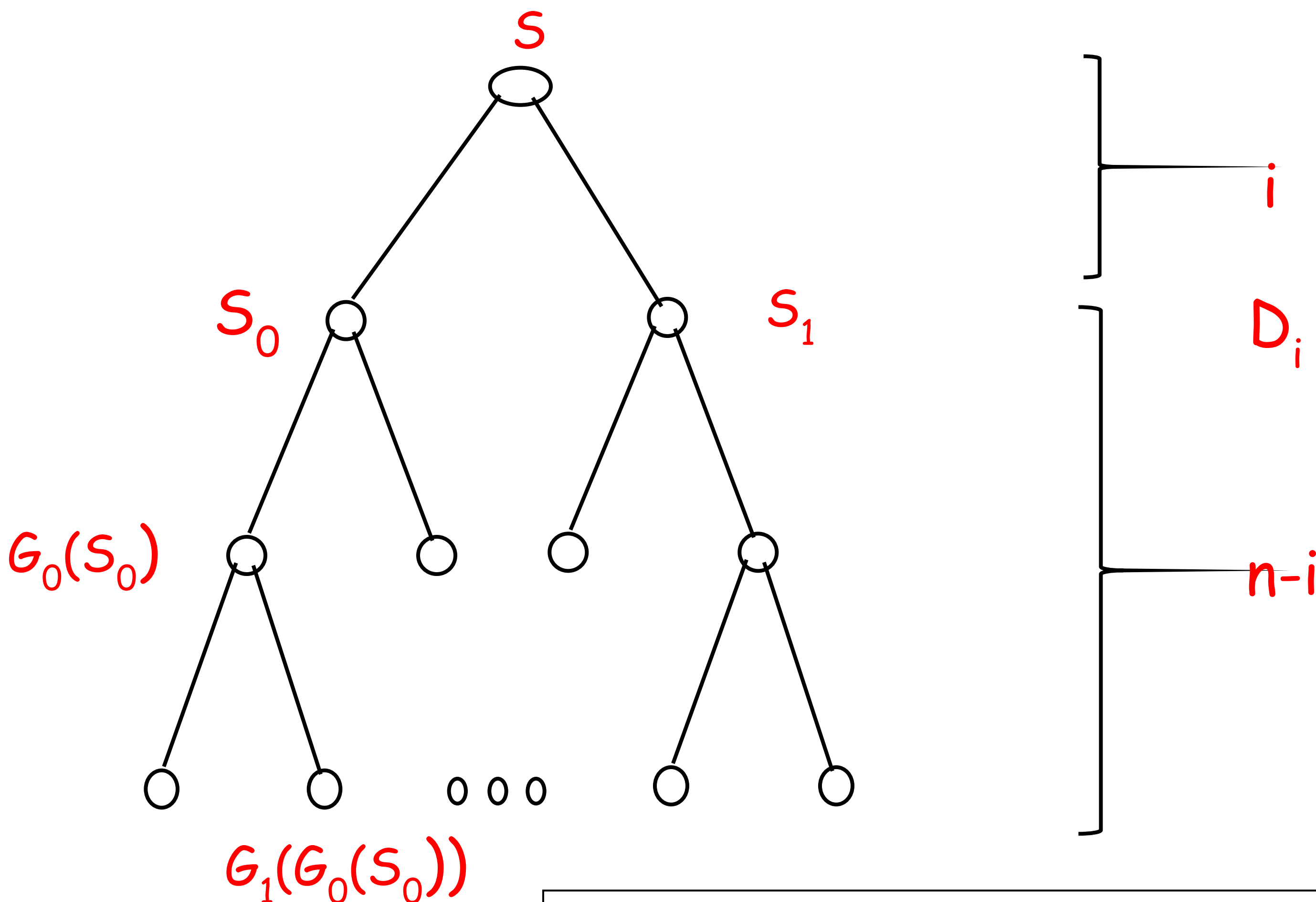
Each leaf corresponds to path $x \in \{0,1\}^n$.

$f_s(x) = G_x(s)$ =LSB(label of leaf x) e.g. $f_i(0000)$= LSB($G_0(G_0$ $(G_0(G_0((s)))))$

Set PSRF family $F = \{F_n\}$ and $F_n = \{f_s\}_{|s|=n}$

Theorem: If G is strong PRG, then F is psrf

Hybrid Argument



p_i = prob ($g \in D_i$: $D^g(1^n) = 1$).

Theorem: If G is cs-prg, then F is psrf

Proof outline: By contradiction. Assume, algorithm D^f exists which "distinguishes" F_n from H_n with probability ε after poly many queries to f (f is either from F_n or all from H_n), then can construct algorithm A to "distinguish" outputs of $G(U_n)$ from U_{2n} with probability $\varepsilon' = \varepsilon/n$

Hybrid argument by levels of the tree

D_i : functions defined by filling *truly* random labels in nodes at level i and then filling lower levels with Pseudo-random values from i+1 down to n

Let $p_i = \text{prob}(f \in D_i : D^f(1^n) = 1)$.

Then $p_1 = \text{prob}(f \in F_n : D^f(1^n) = 1)$ and

$p_n = \text{prob}(f \in H_n : D^f(1^n) = 1)$

and $|p_n - p_1| > \varepsilon \Rightarrow \exists\ 1 < i < n$ s.t. $|p_i - p_{i-1}| \geq \varepsilon/n = \varepsilon'$

Evaluating PSRF

- Given s, n sequential invocations of G
- Polynomial Time but a high polynomial, O(n) evaluations of G (depth)*

  O(n) evaluations of f (per node)

But does the job, its polynomial time!

Let F_n be a collection of PSRF,

Unlearnable concept class by any polynomial time algorithm=$\{c_s\}$where

$c_s=\{(x,f_s(x))\}$ even if learning algorithm can query for f(x) of x of its choice

From Learning to Cryptography: Interesting Consequences

- PRFs cannot be implemented by linear threshold functions as can be learned
- PRF cannot be implemented by polynomial size formula in DNF form, as can be learned for uniform distribution
- etc

Kearns Valiant 87

- Very nice GGM, so there exists poly-time C which cannot be PAC learned independent of representation
- But maybe all C which can be evaluated by simple computational models (within P), can be PAC-learned
- KV87: if Factoring is hard (or RSA is hard to invert or discrete log is hard), then the following cannot be PAC learned
  - the class C_n of polynomial size, log (n) depth, fanin-2 Boolean circuits
  - Finite automata, Constant depth threshold

Klivans Shertov

- if approximating unique Shotest Vector in Lattice (uSVP) is hard then intersection of half spaces is not PAC-learnable independent of representation
- $\Sigma a_i x_i > t$
- Previously: only proper learning impossibility

Review: Number Theory

Let's review some number theory .

Let $N = pq$ be a product of two large primes.

<u>Fact</u>: $Z_N^* = \{a \in Z_N : \gcd(a, N) = 1\}$ is a group.

- group operation is multiplication mod N.
- the order of the group is $\phi(N) = (p-1)(q-1)$

The RSA Trapdoor Permutation

Let e be an integer with $\gcd(e, \phi(N)) = 1$.
RSA assumption: assume that the map $RSA_{N,e}(x) = x^e \bmod N$ is a one-way permutation (i.e hard to compute x from $x^e \bmod N$)

Key Fact: given d such that $ed = 1 \bmod \phi(N)$, it is easy to compute x given x^e mod N
Proof: $(x^e)^d = x^{k\phi(N)+1} = (x^{\phi(N)})^k \cdot x = x \bmod N$
Crypto-speak: $d = e^{-1} \bmod \phi(N)$ is a trapdoor for e

Key Theorem[ACGS86]: LSB(x) is a hardcore predicate for RSA(x).
Can use an oracle to LSB(X) which guesses non-negligebly better than random to invert RSA(x)

Kearns Valiant Concept Class

- Let members concept C_n be defined by RSA triples {p,q, e} s.t. $|p|=|q|=|e|=k$, $n= 10k^2$

Define bin-powers (z,N) =
<z mod N, z^2 mod N,…, $z^{2ceiling(logN)}$ mod N>

Labeled Examples for {p,q,e} in C_n:
{bin-powers($RSA_{N,e}(x)$), $N, e, LSB(x)$)}

Kearns Valiant Concept Class is Hard to Learn

Claim: If C can be (even weakly) PAC-learned, then can invert RSA.

Proof: Each time learner requests an example, choose x s.t. LSB(x)=0/1 (pos/neg) and output labeled (<bin-powers($\mathrm{RSA}_{\mathrm{N,e}}(\mathrm{x}), \mathrm{N}, \mathrm{e} >$, $LSB(x)$)

If Learner putputs h which learns LSB(x) non-neg better then guessing at random on unlabeled <bin-powers($\mathrm{RSA}_{\mathrm{N,e}}(\mathrm{x})), \mathrm{N}, \mathrm{e} >$, then RSA is easy to break as follows;

Kearns Valiant: *C* is in NC1 circuit

Q: Why the business with bin-powers (z,N) =

<z mod N, z^2 mod N,…, $z^{2(\log N)}$ mod N> ?

A: to enable labeling of examples by a low depth NC1 circuit.

Claim: ∃ an NC1 circuit to output the labels of examples for concepts (p,q,e) in class C_n

Proof: Recall d s.t. $(x^e)^d = x \bmod N$, *The NC1 circuit has wired in d =d_0…d_n and on input*

<bin-powers(RSA (x)), N, e>

Augmented PRFs [2003 - present]

Type	Applications	Assumptions
Key homomorphic	Updatable encryption, Distributed PRFs	LWE
Related-key secure	Tweakable block cipher, Simpler CBC-MAC	DDH + CRH
Constrained	Applications of IO	OWF Multilinear maps
Algebraic	Oblivious PRF evaluation, Verifiable computation	DDH

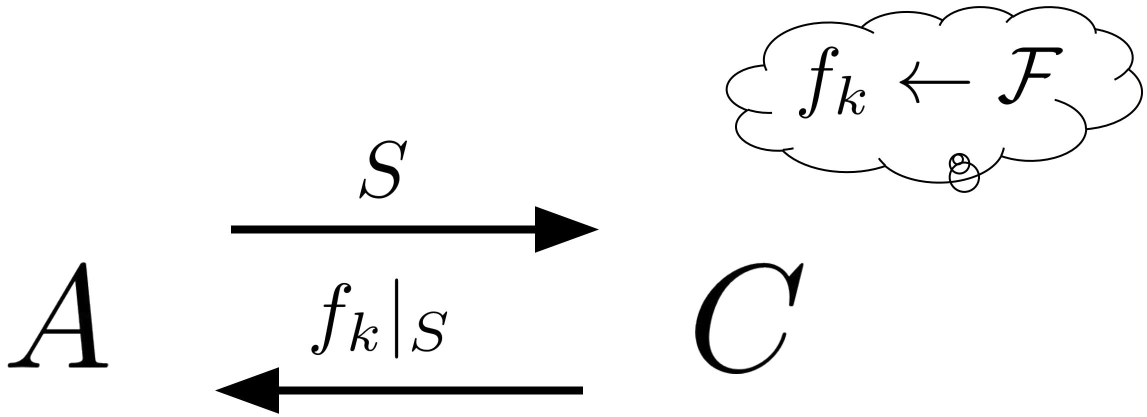
PRFs and Learning: Still hard to Learn even when

PRFs
Constrained
Aggregate
Key homomorphic
Related-key secure

⟺

Learning is HARD
Can label some of the examples of the concept yourself
Can get sums of labels in an interval you specify
Labels satisfy arithmetic
Can receive answers to Queries on related concepts

Constrained PRFs [2013]



$$f_k|_S = Constrain(f_k, S)$$

f_k retains pseudorandomness on $x \notin S$