CS151
Complexity Theory

Lecture 10

April 29, 2021

Distinguishers and predictors

- Distribution D on $\{0,1\}^n$
- D **ε**-passes statistical tests of size s if for all circuits of size s:

$$|Pr_{y \leftarrow U_n}[C(y) = 1] - Pr_{y \leftarrow D}[C(y) = 1]| \leq \varepsilon$$

  – circuit violating this is sometimes called an efficient "distinguisher"

Distinguishers and predictors

- D **ε**-passes prediction tests of size s if for all circuits of size s:

$$Pr_{y \leftarrow D}[C(y_{1,2,\ldots,i-1}) = y_i] \leq \tfrac{1}{2} + \varepsilon$$

  – circuit violating this is sometimes called an efficient "predictor"
- predictor seems stronger
- Yao showed essentially the same!
  – important result and proof ("hybrid argument")

Distinguishers and predictors

Theorem (Yao): if a distribution D on $\{0,1\}^n$ (ε/n)-passes all prediction tests of size s, then it ε-passes all statistical tests of size s' = s – O(n).

Distinguishers and predictors

- Proof:
  – idea: proof by contradiction
  – given a size s' distinguisher C:

$$|Pr_{y \leftarrow U_n}[C(y) = 1] - Pr_{y \leftarrow D}[C(y) = 1]| > \varepsilon$$

  – produce size s predictor P:

$$Pr_{y \leftarrow D}[P(y_{1,2,\ldots,i-1}) = y_i] > \tfrac{1}{2} + \varepsilon/n$$

  – work with distributions that are "hybrids" of the uniform distribution U_n and D

Distinguishers and predictors

– given a size s' distinguisher C:

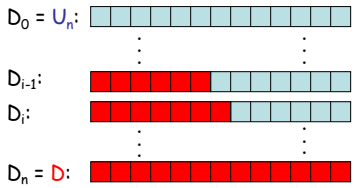
$$|Pr_{y \leftarrow U_n}[C(y) = 1] - Pr_{y \leftarrow D}[C(y) = 1]| > \varepsilon$$

– define n+1 hybrid distributions
– hybrid distribution D_i:
- sample $b = b_1 b_2 \ldots b_n$ from D
- sample $r = r_1 r_2 \ldots r_n$ from U_n
- output:

$$b_1 b_2 \ldots b_i \, r_{i+1} r_{i+2} \ldots r_n$$

Distinguishers and predictors

- Hybrid distributions:



Distinguishers and predictors

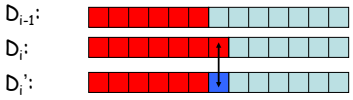
- Define: $p_i = Pr_{y \leftarrow D_i}[C(y) = 1]$
- Note: $p_0 = Pr_{y \leftarrow U_n}[C(y)=1]$; $p_n = Pr_{y \leftarrow D}[C(y)=1]$
- by assumption: $\varepsilon < |p_n - p_0|$
- triangle inequality: $|p_n - p_0| \le \Sigma_{1 \le i \le n} |p_i - p_{i-1}|$
- there must be some i for which

$$|p_i - p_{i-1}| > \varepsilon/n$$

- WLOG assume $p_i - p_{i-1} > \varepsilon/n$
  - can invert output of C if necessary

Distinguishers and predictors

- define distribution D_i' to be D_i with i-th bit flipped
- $p_i' = Pr_{y \leftarrow D_i'}[C(y) = 1]$



- notice:

$$D_{i-1} = (D_i + D_i')/2 \qquad p_{i-1} = (p_i + p_i')/2$$

Distinguishers and predictors

- randomized predictor P' for i^{th} bit:
  - input: $u = y_1 y_2 \ldots y_{i-1}$ (which comes from D)
  - flip a coin: $\mathbf{d} \in \{0,1\}$
  - $w = w_{i+1} w_{i+2} \ldots w_n \leftarrow U_{n-i}$
  - evaluate $C(u\mathbf{d}w)$
  - if 1, output d; if 0, output $\neg d$

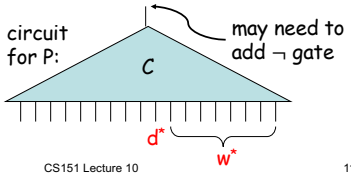
Claim:

$$Pr_{y \leftarrow D, d, w \leftarrow U_{n-i}}[P'(y_1 \ldots_{i-1}) = y_i] > \tfrac{1}{2} + \varepsilon/n.$$

Distinguishers and predictors

- P' is randomized procedure
- there must be some fixing of its random bits d, w that preserves the success prob.
- final predictor P has d^* and w^* hardwired:

Size is $s' + O(n) = s$ as promised



Distinguishers and predictors

- Proof of claim:

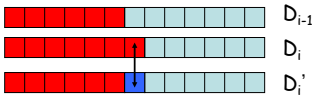
$u = y_1 y_2 \ldots y_{i-1}$

$Pr_{y \leftarrow D, d, w \leftarrow U_{n-i}}[P'(y_1 \ldots_{i-1}) = y_i] =$
$Pr[y_i = d \mid C(u,d,w) = 1] Pr[C(u,d,w) = 1]$
$+ Pr[y_i = \neg d \mid C(u,d,w) = 0] Pr[C(u,d,w) = 0]$

$= Pr[y_i = d \mid C(u,d,w) = 1](p_{i-1})$
$+ Pr[y_i = \neg d \mid C(u,d,w) = 0](1 - p_{i-1})$

Distinguishers and predictors

$u = y_1 y_2 \ldots y_{i-1}$



– Observe:

$\Pr[y_i = d \mid C(u,d,w) = 1]$
$= \Pr[C(u,d,w) = 1 \mid y_i = d]\Pr[y_i=d] / \Pr[C(u,d,w) = 1]$
$= p_i/(2p_{i-1})$

$\Pr[y_i = \neg d \mid C(u,d,w) = 0]$
$= \Pr[C(u,d,w)=0 \mid y_i= \neg d]\Pr[y_i=\neg d] / \Pr[C(u,d,w) = 0]$
$= (1 - p_i') / 2(1 - p_{i-1})$

Distinguishers and predictors

- Success probability:
  $\Pr[y_i=d|C(u,d,w)=1](p_{i-1}) + \Pr[y_i=\neg d|C(u,d,w)=0](1-p_{i-1})$
- We know:
  - $\Pr[y_i = d \mid C(u,d,w) = 1] = p_i/(2p_{i-1})$
  - $\Pr[y_i = \neg d \mid C(u,d,w) = 0] = (1 - p_i')/2(1 - p_{i-1})$
  - $p_{i-1} = (p_i + p_i')/2$
  - $p_i - p_{i-1} > \varepsilon/n$

  $p_i'/2 = p_{i-1} - p_i/2$
- Conclude:

$\Pr[P'(y_{1\ldots i-1}) = y_i] = ½ + (p_i - p_i')/2$
$= ½ + p_i/2 - (p_{i-1} - p_i/2) = ½ + p_i - p_{i-1} > ½ + \varepsilon/n.$

The BMY Generator

- Recall goal: for all $1 > \delta > 0$, family of PRGs $\{G_m\}$ with
  output length **m** fooling size **s** = m
  seed length **t =** m^δ running time m^c
  error **ε** < 1/6
- If one way permutations exist then WLOG there is OWP $f = \{f_n\}$ with hard bit $h = \{h_n\}$

The BMY Generator

- Generator $G^\delta = \{G^\delta_m\}$:
  - $t = m^\delta$
  - $y_0 \in \{0,1\}^t$
  - $y_i = f_t(y_{i-1})$
  - $b_i = h_t(y_i)$
  - $G^\delta(y_0) = b_{m-1}b_{m-2}b_{m-3}\ldots b_0$

The BMY Generator

Theorem (BMY): for every $\delta > 0$, there is a constant c s.t. for all d, e, G^δ is a PRG with

error **ε** < $1/m^d$

fooling size **s** = m^e

running time m^c

- Note: stronger than we needed
  - sufficient to have **ε** < 1/6; **s** = m

The BMY Generator

Generator $G^\delta = \{G^\delta_m\}$:
- $t = m^\delta$; $y_0 \in \{0,1\}^t$; $y_i = f_t(y_{i-1})$; $b_i = h_t(y_i)$
- $G^\delta_m(y_0) = b_{m-1}b_{m-2}b_{m-3}\ldots b_0$

- Proof:
  - computable in time at most
    $mt^c < m^{c+1}$
  - assume G^δ does not $(1/m^d)$-pass statistical test $C = \{C_m\}$ of size m^e:
    $|\Pr_{y \leftarrow U_m}[C(y) = 1] - \Pr_{z \leftarrow D}[C(z) = 1]| > 1/m^d$

The BMY Generator

Generator $G^\delta = \{G^\delta_m\}$:
- $-t = m^\delta$; $y_0 \in \{0,1\}^t$; $y_i = f_t(y_{i-1})$; $b_i = h_t(y_i)$
- $-G^\delta_m(y_0) = b_{m-1}b_{m-2}b_{m-3}\ldots b_0$

– transform this **distinguisher** into a **predictor** P of size $m^e + O(m)$:

$\Pr_y[P(b_{m-1}\ldots b_{m-i}) = b_{m-i-1}] > ½ + 1/m^{d+1}$

The BMY Generator

Generator $G^\delta = \{G^\delta_m\}$:
- $-t = m^\delta$; $y_0 \in \{0,1\}^t$; $y_i = f_t(y_{i-1})$; $b_i = h_t(y_i)$
- $-G^\delta_m(y_0) = b_{m-1}b_{m-2}b_{m-3}\ldots b_0$

– a procedure to compute $h_t(f_t^{-1}(y))$
- set $y_{m-i} = y$; $b_{m-i} = h_t(y_{m-i})$
- compute y_j, b_j for $j = m-i+1, m-i+2\ldots, m-1$ as above
- evaluate $P(b_{m-1}b_{m-2}\ldots b_{m-i})$
- f a permutation implies $b_{m-1}b_{m-2}\ldots b_{m-i}$ distributed as (prefix of) output of generator:

$\Pr_y[P(b_{m-1}b_{m-2}\ldots b_{m-i}) = b_{m-i-1}] > ½ + 1/m^{d+1}$

The BMY Generator

Generator $G^\delta = \{G^\delta_m\}$:
- $-t = m^\delta$; $y_0 \in \{0,1\}^t$; $y_i = f_t(y_{i-1})$; $b_i = h_t(y_i)$
- $-G^\delta_m(y_0) = b_{m-1}b_{m-2}b_{m-3}\ldots b_0$

$\Pr_y[P(b_{m-1}b_{m-2}\ldots b_{m-i}) = b_{m-i-1}] > ½ + 1/m^{d+1}$

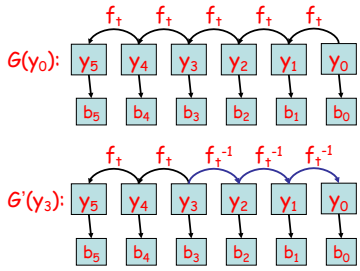
– What is b_{m-i-1}?

$b_{m-i-1} = h_t(y_{m-i-1}) = h_t(f_t^{-1}(y_{m-i})) = h_t(f_t^{-1}(y))$

– We have described a family of polynomial-size circuits that computes $h_t(f_t^{-1}(y))$ from y with success greater than ½ + 1/poly(m)

– Contradiction.

The BMY Generator



same distribution

Hardness vs. randomness

- We have shown:

  If one-way permutations exist then

  BPP $\subseteq \cap_{\delta>0}$ TIME(2^{n^δ}) $\subsetneq$ EXP

- simulation is better than brute force, but just barely
- stronger assumptions on difficulty of inverting OWF lead to better simulations…

Hardness vs. randomness

- Next, we will show:

  If **E** requires exponential size circuits then **BPP = P**

  by building a different generator from different assumptions.

E = $\cup_k$ DTIME(2^{kn})

Hardness vs. randomness

- BMY: for every $\delta > 0$, G^δ is a PRG with
  - seed length $\mathbf{t} = m^\delta$
  - output length $\mathbf{m}$
  - error $\boldsymbol{\varepsilon} < 1/m^d$ (all d)
  - fooling size $\mathbf{s} = m^e$ (all e)
  - running time m^c
- running time of simulation dominated by 2^t

Hardness vs. randomness

- To get BPP = P, would need $t = O(\log m)$
- BMY building block is one-way-permutation:

  $f:\{0,1\}^t \rightarrow \{0,1\}^t$
- required to fool circuits of size m^e (all e)
- with these settings a circuit has time to invert f by brute force!

can't get BPP = P with this type of PRG

Hardness vs. randomness

- BMY pseudo-random generator:
  - one generator fooling all poly-size bounds
  - one-way-permutation is hard function
  - implies hard function in **NP ∩ coNP**
- New idea (Nisan-Wigderson):
  - for each poly-size bound, one generator
  - hard function allowed to be in

$\mathbf{E} = \cup_k \mathbf{DTIME}(2^{kn})$

Comparison

	BMY: ∀ δ > 0 PRG G^δ	NW: PRG G
seed length	$\mathbf{t} = m^\delta$	$\mathbf{t} = O(\log m)$
running time	$t^c m$	m^c
output length	$\mathbf{m}$ <<	$\mathbf{m}$ >
error	$\boldsymbol{\varepsilon} < 1/m^d$ (all d)	$\boldsymbol{\varepsilon} < 1/m$
fooling size	$\mathbf{s} = m^e$ (all e)	$\mathbf{s} = m$

NW PRG

- NW: for fixed constant δ, $G = \{G_n\}$ with

seed length	$\mathbf{t} = O(\log n)$	$\mathbf{t} = O(\log m)$
running time	n^c	m^c
output length	$\mathbf{m} = n^\delta$	$\mathbf{m}$
error	$\boldsymbol{\varepsilon} < 1/m$	
fooling size	$\mathbf{s} = m$	

- Using this PRG we obtain **BPP = P**
  - to fool size n^k use $G_{n^{k/\delta}}$
  - running time $O(n^k + n^{ck/\delta})2^t = \text{poly}(n)$

NW PRG

- First attempt: build PRG assuming **E** contains **unapproximable** functions

Definition: The function family

$f = \{f_n\}$, $f_n:\{0,1\}^n \rightarrow \{0,1\}$

is s(n)-unapproximable if for every family of size s(n) circuits $\{C_n\}$:

$\Pr_x[C_n(x) = f_n(x)] \leq \frac{1}{2} + 1/s(n)$.

One bit

- Suppose f = $\{f_n\}$ is s(n)-unapproximable, for $s(n) = 2^{\Omega(n)}$, and in **E**
- a "1-bit" generator family G = $\{G_n\}$:

$$G_n(y) = y \circ f_{\log n}(y)$$

- Idea: if not a PRG then exists a predictor that computes $f_{\log n}$ with better than ½ + 1/s(log n) agreement; contradiction.

One bit

- Suppose f = $\{f_n\}$ is s(n)-unapproximable, for $s(n) = 2^{\delta n}$, and in **E**
- a "1-bit" generator family G = $\{G_n\}$:

$$G_n(y) = y \circ f_{\log n}(y)$$

  - seed length **t** = log n
  - output length **m** = log n + 1 (want n^δ)
  - fooling size **s** ≈ s(log n) = n^δ
  - running time n^c
  - error **ε** ≈ 1/s(log n) = $1/n^\delta$

Many bits

- Try outputting many evaluations of f:

$$G(y) = f(b_1(y)) \circ f(b_2(y)) \circ \ldots \circ f(b_m(y))$$

- Seems that a predictor must evaluate $f(b_i(y))$ to predict i-th bit
- Does this work?

Many bits

- Try outputting many evaluations of f:

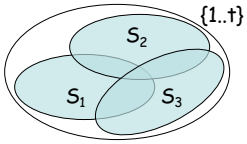
$$G(y) = f(b_1(y)) \circ f(b_2(y)) \circ \ldots \circ f(b_m(y))$$

- predictor might notice correlations without having to compute f
- but, more subtle argument works for a specific choice of $b_1 \ldots b_m$

Nearly-Disjoint Subsets

Definition: $S_1, S_2, \ldots, S_m \subseteq \{1 \ldots t\}$ is an (h, a) design if

- for all i, $|S_i| = h$
- for all $i \neq j$, $|S_i \cap S_j| \leq a$



Nearly-Disjoint Subsets

Lemma: for every ε > 0 and m < n can in poly(n) time construct an

(h = log n, a = ε log n) design

$S_1, S_2, \ldots, S_m \subseteq \{1 \ldots t\}$ with t = O(log n).

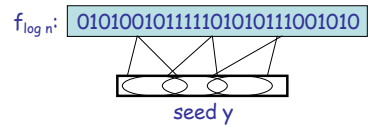
Nearly-Disjoint Subsets

- Proof sketch:
  - pick random (log n)-subset of {1…t}
  - set t = O(log n) so that expected overlap with a fixed S_i is $\varepsilon \log n/2$
  - probability overlap with S_i is > $\varepsilon \log n$ is at most 1/n
  - union bound: some subset has required small overlap with *all* S_i picked so far…
  - find it by exhaustive search; repeat n times.

The NW generator

- f ∈ **E** s(n)-unapproximable, for $s(n) = 2^{\delta n}$
- $S_1,\ldots,S_m \subseteq \{1\ldots t\}$ (log n, a = δlog n/3) design with t = O(log n)

$G_n(y)=f_{\log n}(y_{|S_1}) \circ f_{\log n}(y_{|S_2}) \circ \ldots \circ f_{\log n}(y_{|S_m})$



The NW generator

Theorem (Nisan-Wigderson): G={G_n} is a pseudo-random generator with:

- seed length **t** = O(log n)
- output length **m** = $n^{\delta/3}$
- running time n^c
- fooling size **s** = m
- error **ε** = 1/m

The NW generator

- Proof:
  - assume does not ε-pass statistical test C = {C_m} of size s:

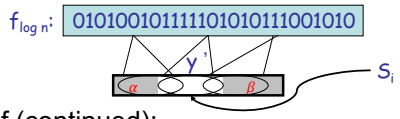
    $|Pr_x[C(x) = 1] - Pr_y[C(G_n(y)) = 1]| > \varepsilon$

  - can transform this **distinguisher** into a **predictor** P of size s' = s + O(m):

    $Pr_y[P(G_n(y)_{1\ldots i-1}) = G_n(y)_i] > ½ + \varepsilon/m$

The NW generator

$G_n(y)=f_{\log n}(y_{|S_1}) \circ f_{\log n}(y_{|S_2}) \circ \ldots \circ f_{\log n}(y_{|S_m})$



- Proof (continued):

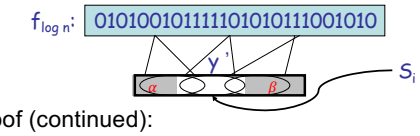
  $Pr_y[P(G_n(y)_{1\ldots i-1}) = G_n(y)_i] > ½ + \varepsilon/m$

  – fix bits outside of S_i to preserve advantage:

  $Pr_{y'}[P(G_n(\alpha y'\beta)_{1\ldots i-1}) = G_n(\alpha y'\beta)_i] > ½ + \varepsilon/m$

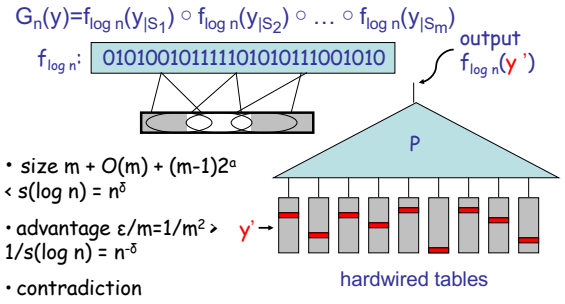
The NW generator

$G_n(y)=f_{\log n}(y_{|S_1}) \circ f_{\log n}(y_{|S_2}) \circ \ldots \circ f_{\log n}(y_{|S_m})$



- Proof (continued):
  - $G_n(\alpha y'\beta)_i$ is exactly $f_{\log n}(y')$
  - for j ≠ i, as vary y', $G_n(\alpha y'\beta)_j$ varies over 2^a values!
  - hard-wire up to (m-1) tables of 2^a values to provide $G_n(\alpha y'\beta)_{1\ldots i-1}$

The NW generator



- size $m + O(m) + (m-1)2^a < s(\log n) = n^\delta$
- advantage $\varepsilon/m = 1/m^2 > 1/s(\log n) = n^{-\delta}$
- contradiction

$y' \rightarrow$

Worst-case vs. Average-case

Theorem (NW): if **E** contains $2^{\Omega(n)}$-unapproximable functions then **BPP** = **P**.

- How reasonable is unapproximability assumption?
- Hope: obtain **BPP** = **P** from worst-case complexity assumption
  - try to fit into existing framework without new notion of "unapproximability"

Worst-case vs. Average-case

Theorem (Impagliazzo-Wigderson, Sudan-Trevisan-Vadhan)

If **E** contains functions that require size $2^{\Omega(n)}$ circuits, then **E** contains $2^{\Omega(n)}$–unapproximable functions.

- Proof:
  - main tool: error correcting code

Error-correcting codes

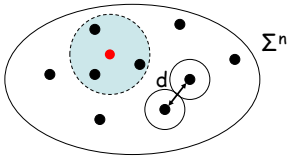
- Error Correcting Code (ECC):
  $C: \Sigma^k \rightarrow \Sigma^n$
- message $m \in \Sigma^k$
- received word R

  - C(m) with some positions corrupted
- if not too many errors, can decode: D(R) = m
- parameters of interest:
  - rate: k/n
  - distance:

$d = \min_{m \neq m'} \Delta(C(m), C(m'))$

Distance and error correction

- C is an ECC with distance d
- can **uniquely** decode from up to $\lfloor d/2 \rfloor$ errors



Distance and error correction

- can find **short list** of messages (one correct) after closer to d errors!

Theorem (Johnson): a binary code with distance $(\frac{1}{2} - \delta^2)n$ has at most $O(1/\delta^2)$ codewords in any ball of radius $(\frac{1}{2} - \delta)n$.

Example: Reed-Solomon

- alphabet $\Sigma = \mathbf{F}_q$: field with q elements
- message $m \in \Sigma^k$
- polynomial of degree at most k-1

$$p_m(x) = \Sigma_{i=0\ldots k-1}\, m_i x^i$$

- codeword $C(m) = (p_m(x))_{x \in \mathbf{F}_q}$
- rate = k/q

Example: Reed-Solomon

- Claim: distance d = q – k + 1
  - suppose $\Delta(C(m), C(m')) < q - k + 1$
  - then there exist polynomials $p_m(x)$ and $p_{m'}(x)$ that agree on more than k-1 points in $\mathbf{F}_q$
  - polnomial $p(x) = p_m(x) - p_{m'}(x)$ has more than k-1 zeros
  - but degree at most k-1…
  - contradiction.