Distinguishers and predictors

• Distribution $D$ on $\{0,1\}^n$
  • $D \epsilon$-passes \textit{statistical tests} of size $s$ if for all circuits of size $s$:
    \[ |\Pr_{y \sim U_n}[C(y) = 1] - \Pr_{y \sim D}[C(y) = 1]| \leq \epsilon \]
  – circuit violating this is sometimes called an efficient "distinguisher"

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

Distinguishers and predictors

• D $\epsilon$-passes \textit{prediction tests} of size $s$ if for all circuits of size $s$:
  \[ \Pr_{y \sim D}[C(y) = 1] \leq \frac{1}{2} + \epsilon \]
  – circuit violating this is sometimes called an efficient "predictor"
• Yao showed essentially the same!
  – important result and proof ("hybrid argument")

Distinguishers and predictors

• Proof:
  – idea: proof by contradiction
  – given a size $s'$ distinguisher $C$:
    \[ |\Pr_{y \sim U_n}[C(y) = 1] - \Pr_{y \sim D}[C(y) = 1]| > \epsilon \]
  – produce size $s$ predictor $P$:
    \[ \Pr_{y \sim D}[P(y_1,2,\ldots,s) = y_i] > \frac{1}{2} + \epsilon/n \]
  – work with distributions that are "hybrids" of the uniform distribution $U_n$ and $D$
Distinguishers and predictors

• Hybrid distributions:

\[ D_0 = U_n : \text{...} \]

\[ D_{n-1} : \text{...} \]

\[ D_n : \text{...} \]

\[ D_k = D_i : \text{...} \]

April 29, 2021 CS151 Lecture 10

Distinguishers and predictors

– Define: \( p_i = \Pr_{y \leftarrow D_i}[C(y) = 1] \)

– Note: \( p_0 = \Pr_{y \leftarrow U_n}[C(y) = 1] \); \( p_n = \Pr_{y \leftarrow D_n}[C(y) = 1] \)

– by assumption: \( \epsilon < |p_n - p_0| \)

– triangle inequality: \( |p_n - p_0| \leq \sum_{1 \leq i \leq n} |p_i - p_{i-1}| \)

– there must be some \( i \) for which \( |p_i - p_{i-1}| > \epsilon/n \)

– WLOG assume \( p_i - p_{i-1} > \epsilon/n \)

• can invert output of \( C \) if necessary

April 29, 2021 CS151 Lecture 10

Distinguishers and predictors

– define distribution \( D'_i \) to be \( D_i \) with \( i \)-th bit flipped

\[ p'_i = \Pr_{y \leftarrow D'_i}[C(y) = 1] \]

\[ D_{i-1} : \text{...} \]

\[ D_i : \text{...} \]

\[ D'_i : \text{...} \]

– notice:

\( D_{i+1} = (D_i + D'_i)/2 \)

\( p_{i+1} = (p_i + p'_i)/2 \)

April 29, 2021 CS151 Lecture 10

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:

\[ u = y_1 y_2 \ldots y_i : \text{...} \]

\[ \text{circuit for } P : \text{...} \]

\( d' \)

\( w' \)

may need to add \( \neg \) gate

\( s' + O(n) = s \)

as promised

April 29, 2021 CS151 Lecture 10

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: \( d \in \{0, 1\} \)

– \( w = w_{i+1} w_{i+2} \ldots w_n \leftarrow U_n \)

– evaluate \( C(udw) \)

– if 1, output \( d \); if 0, output \( \neg d \)

Claim:

\( \Pr_{y \leftarrow D, d, w \leftarrow U_n}[P'(y_1 \ldots y_i) = y_i] > \frac{1}{2} + \epsilon/n. \)

April 29, 2021 CS151 Lecture 10

Distinguishers and predictors

• Proof of claim:

\[ \Pr_{y \leftarrow D, d, w \leftarrow U_n}[P'(y_1 \ldots y_i) = y_i] = \]

\[ = \Pr[y_i = d \mid C(u,d,w) = 1] \Pr[C(u,d,w) = 1] \]

\[ + \Pr[y_i = \neg d \mid C(u,d,w) = 0] \Pr[C(u,d,w) = 0] \]

\[ = \Pr[y_i = d \mid C(u,d,w) = 1][p_i] \]

\[ + \Pr[y_i = \neg d \mid C(u,d,w) = 0][1 - p_i] \]

April 29, 2021 CS151 Lecture 10
Distinguishers and predictors

\[ u = y_1 y_2 \ldots y_{t-1} \]

- Observe:
  \[
  \Pr[y_i = d \mid C(u, d, w) = 1] = \Pr[C(u, d, w) = 1] \mid y_i = d \Pr[y_i = d] / \Pr[C(u, d, w) = 1] = p/(2p_1)
  \]

\[
\Pr[y_i = -d \mid C(u, d, w) = 0] = \Pr[C(u, d, w) = 0] \mid y_i = -d \Pr[y_i = -d] / \Pr[C(u, d, w) = 0] = (1 - p_1) / (2(1 - p_1))
\]

- Note: stronger than we needed

\[ \Pr[\text{error} < 1/6; \text{seed length} t = m^5; \text{output length} m; \text{fooling size} s = m; \text{running time} m^c] \]

- Sufficient to have \( \varepsilon < 1/6; s = m \)

\[ \text{Theorem (BMY): for every } \delta > 0, \text{ there is a constant } c \text{ s.t. for all } d, e, G^\delta \text{ is a PRG with} \]

\[ \text{error } \varepsilon < 1/m^d \]

\[ \text{fooling size } s = m^d \]

\[ \text{running time } m^c \]

\[ \text{Success probability:} \]

\[ \Pr[y_i = d \mid C(u, d, w) = 1][p_1] + \Pr[y_i = -d \mid C(u, d, w) = 0][1 - p_1] \]

- We know:
  \[ - \Pr[y_i = d \mid C(u, d, w) = 1] = p/(2p_1) \]

\[ - \Pr[y_i = -d \mid C(u, d, w) = 0] = (1 - p_1)/2(1 - p_1) \]

\[ - p_1 = (p + p_s)^2 \]

\[ - p_1 > \varepsilon/n \]

\[ \text{Conclude:} \]

\[ \Pr[p^y \in \{0, 1\} \mid y] = \frac{1}{2} + p_1 - p_1 / 2 = \frac{1}{2} + p_1 - p_1 > \frac{1}{2} + \varepsilon/n. \]

\[ \text{Distinguishers and predictors} \]

\[ \text{The BMY Generator} \]

- Recall goal: for all \( 1 > \delta > 0 \), family of PRGs \( \{G_m\} \) with
  \[ \text{output length } m \]
  \[ \text{fooling size } s = m \]
  \[ \text{seed length } t = m^5 \]
  \[ \text{running time } m^c \]
  \[ \text{error } \varepsilon < 1/6 \]

- If one way permutations exist then WLOG there is OWP \( f = \{f_n\} \) with hard bit \( h = \{h_n\} \)

\[ \text{The BMY Generator} \]

- Generator \( G^\delta = \{G^\delta_m\} \):
  \[ - t = m^5 \]
  \[ - y_0 \in \{0, 1\} \]
  \[ - y_i = f_i(y_{i-1}) \]
  \[ - b_i = h_i(y) \]
  \[ - G^\delta(y_0) = b_{m-1}b_{m-2}b_{m-3} \ldots b_0 \]

- Prove:
  \[ \text{compputeable in time at most } m^{c+1} \]
  \[ \text{assume } G^\delta \text{ does not } (1/m^d) \text{-pass statistical test } C \]
  \[ (C_m) \text{ of size } m^c; \]
  \[ \Pr[y_i = y \mid C(y) = 1] - \Pr[z_i = d \mid C(z) = 1] > 1/m^d \]
The BMY Generator

Generator \( G^3 = \{G^3_m\} \):
\[ t = m^5; \quad y_0 \in \{0,1\}; \quad y_i = f_i(y_i) \quad b_i = h_i(y_i) \]
\[ -G^3_m(y_0) = b_{m_1}b_{m_2}b_{m_3} - b_0 \]

– transform this distinguisher into a predictor \( P \) of size \( m^5 + O(m) \):
\[ \Pr_y[P(b_{m_1}b_{m_2}b_{m_3}) = b_{m+1}] > \frac{1}{2} + 1/m^{d+1} \]

– What is \( b_{m+1} \)?
\[ b_{m+1} = h_{i+1}(y_{m+1}) = h_i(f_i^{-1}(y_{m+1})) \]

– We have described a family of polynomial-size circuits that computes \( h_i(f_i^{-1}(y)) \) from \( y \) with success greater than \( \frac{1}{2} + 1/\text{poly}(m) \)

– Contradiction.

Hardness vs. randomness

• We have shown:
  If one-way permutations exist then
  \( \text{BPP} \subseteq \bigcap_{k>0} \text{TIME}(2^{kn}) \subseteq \text{EXP} \)

• simulation is better than brute force, but just barely

• stronger assumptions on difficulty of inverting OWF lead to better simulations…
Hardness vs. randomness

- BMY: for every $\delta > 0$, $G^\delta$ is a PRG with
  - seed length $t = m^\delta$
  - output length $m$
  - error $\epsilon < 1/m^d$ (all $d$)
  - fooling size $s = m^e$ (all $e$)
  - running time $m^c$

- running time of simulation dominated by $2^t$

Hardness vs. randomness

- To get $\text{BPP} = \text{P}$, 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 $\text{BPP} = \text{P}$ with this type of PRG

Comparison

- BMY: $\forall \delta > 0$ PRG $G^\delta$
  - seed length $t = m^\delta$
  - output length $m$
  - error $\epsilon < 1/m^d$ (all $d$)
  - fooling size $s = m^e$ (all $e$)
  - running time $m^c$

- NW: PRG $G$
  - seed length $t = O(\log m)$
  - output length $m$
  - error $\epsilon < 1/m$
  - fooling size $s = m$

NW PRG

- NW: for fixed constant $\delta$, $G = \{G_n\}$ with
  - seed length $t = O(\log n)$
  - running time $n^c$
  - output length $m = n^\delta$
  - error $\epsilon < 1/m$
  - fooling size $s = m$

- Using this PRG we obtain $\text{BPP} = \text{P}$
  - to fool size $n^k$ use $G_{n^k}$
  - running time $O(n^k + n^{3k/5} 2^t) = \text{poly}(n)$

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 \( \mathbb{E} \)
- a “1-bit” generator family \( G = \{ G_n \} \):
  \[ G_n(y) = y \cdot f_{\log n}(y) \]
- Idea: if not a PRG then exists a predictor that computes \( f_{\log n} \) with better than \( \frac{1}{2} + \frac{1}{s(\log n)} \) agreement; contradiction.

Many bits

- Try outputting many evaluations of \( f \):
  \[ G(y) = f(b_1(y)) \cdot f(b_2(y)) \cdots f(b_m(y)) \]
- Seems that a predictor must evaluate \( f(b_i(y)) \) to predict \( i \)-th bit
- Does this work?

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 \( i \neq j \), \( |S_i \cap S_j| \leq a \)

**Lemma:** for every \( \varepsilon > 0 \) and \( m < n \) can in \( \text{poly}(n) \) time construct an
\((h = \log n, a = \varepsilon \log n)\) 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 \( \epsilon \log n/2 \)
  - probability overlap with \( S_i \) is \( > \epsilon \log n \) 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

**Theorem** (Nisan-Wigderson): \( G_n \) 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 \( \epsilon = 1/m \)

\[
G_n(y) = f_{\log n}(y_{|S_1}) \circ f_{\log n}(y_{|S_2}) \circ \cdots \circ f_{\log n}(y_{|S_m})
\]

\[010100101111101010111001010\]

\( f_{\log n} \): seed y
The NW generator

\[ G_n(y) = f_{\log n}(y_{|S_1}) \circ f_{\log n}(y_{|S_2}) \circ \cdots \circ f_{\log n}(y_{|S_m}) \]

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

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

\[ P \]

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 $\Sigma = \mathbb{F}_q$: field with $q$ elements
- message $m \in \Sigma^k$
- polynomial of degree at most $k-1$
  \[ p_m(x) = \sum_{i=0}^{k-1} m_i x^i \]
- codeword $C(m) = (p_m(x))_{x \in \mathbb{F}_q}$
- rate $= k/q$

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 $\mathbb{F}_q$
- polynomial $p(x) = p_m(x) - p_{m'}(x)$ has more than $k-1$ zeros
- but degree at most $k-1$
- contradiction.