CS151
Complexity Theory
Lecture 9
April 27, 2015

NW PRG

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

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

One bit

• Suppose \( f = \{f_n\} \) is \( s(n) \)-unapproximable, for \( s(n) = 2^{\Omega(n)} \), and in \( \mathbf{E} \)
  - 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 \( 1/2 + 1/s(\log n) \) agreement; contradiction.

Many bits

• Try outputting many evaluations of \( f \):
  \[ G(y) = f(b_1(y)) \circ f(b_2(y)) \cdots \circ f(b_m(y)) \]

  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 $i \neq j$, $|S_i \cap S_j| \leq a$

**Lemma**: for every $\epsilon > 0$ and $m < n$ can in $\text{poly}(n)$ time construct an
  
  $(h = \log n$, $a = \epsilon \log n)$ design

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

The NW generator

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

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

  $f_{\log n} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$

  seed $y$

**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 $\epsilon = 1/m$
The NW generator

Proof:
- assume does not \( \varepsilon \)-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 \( s' = s + O(m) \):
  \[ Pr_y[P(G_n(y_1, \ldots, y_{i-1}) = G_n(y_i)] > \frac{1}{2} + \frac{\varepsilon}{m} \]

Proof (continued):
- \( G_n(y) \) is exactly \( f_{\log n}(\cdot) \)
- for \( j \neq i \), as vary \( y' \). \( G_n(\alpha y') \) varies over \( 2^a \) values!
- hardwire up to \((m-1)\) tables of \( 2^a \) values to provide \( G_n(\alpha y')_{1, \ldots, i} \)

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"
Error-correcting codes

- Error Correcting Code (ECC): \[ C: \Sigma^k \to \Sigma^n \]
- 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: \( d = \min \Delta(C(m), C(m')) \)

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 \((1/2 - \delta^2)n\) has at most \( O(1/\delta^2) \) codewords in any ball of radius \((1/2 - \delta)n\).

Example: Reed-Solomon

- alphabet \( \Sigma = F_q \): 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 \( C(m) = (p_m(x))_{x \in F_q} \)
- 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 \( F_q \)
  - polynomial \( p(x) = p_m(x) - p_{m'}(x) \) has more than \( k-1 \) zeros
  - but degree at most \( k-1 \)
  - contradiction.

Example: Reed-Muller

- Parameters: \( t \) (dimension), \( h \) (degree)
- alphabet \( \Sigma = F_q \): field with \( q \) elements
- message \( m \in \Sigma^k \)
- multivariate polynomial of total degree at most \( h \):
  \[ p_m(x) = \sum_{i=0}^{k-1} m_i M_i \]
  \( \{ M_i \} \) are all monomials of degree \( \leq h \)
Example: Reed-Muller

- $M_i$ is monomial of total degree $h$
  - e.g. $x_1^2x_2x_3^3$
  - need $\#$ monomials $(h+t\choose t) > k$
- codeword $C(m) = (p_m(x))_{x \in \mathbb{F}_q^h}$
- rate $= k/q$
- Claim: distance $d = (1 - h/q)q^t$
  - proof: Schwartz-Zippel: polynomial of degree $h$ can have at most $h/q$ fraction of zeros

Codes and hardness

- Reed-Solomon (RS) and Reed-Muller (RM) codes are efficiently encodable
- efficient unique decoding?
  - yes (classic result)
- efficient list-decoding?
  - yes (RS on problem set)

Codes and Hardness

- Use for worst-case to average case:
  - truth table of $f: \{0,1\}^{\log k} \rightarrow \{0,1\}$
    - (worst-case hard)
  - $m$: 01100010
  - truth table of $f': \{0,1\}^{\log n} \rightarrow \{0,1\}$
    - (average-case hard)
  - Enc($m$): 01100010000010

- if $n = \text{poly}(k)$ then $f \in \mathbb{E}$ implies $f' \in \mathbb{E}$
- Want to be able to prove:
  - if $f'$ is $s'$-approximable,
  - then $f$ is computable by a size $s = \text{poly}(s')$ circuit

Codes and Hardness

- Key: circuit $C$ that approximates $f'$ implicitly gives received word $R$
  - $R$: 0110010000100010
  - Enc($m$): 01100010000010
- Decoding procedure $D$ “computes” $f$ exactly
  - Requires special notion of efficient decoding
- small circuit $C$ approximating $f'$
- small circuit that computes $f$ exactly $f(i)$

April 27, 2015
Encoding

- use a (variant of) Reed-Muller code concatenated with the Hadamard code
  - \( q \) (field size), \( t \) (dimension), \( h \) (degree)
- encoding procedure:
  - message \( m \in \{0,1\}^k \)
  - subset \( S \subseteq F_q \) of size \( h \)
  - efficient 1-1 function \( \text{Emb}: [k] \to S' \)
  - find coeffs of degree \( h \) polynomial \( p_m : F_q^t \to F_q \)
    for which \( p_m(\text{Emb}(i)) = m \) for all \( i \)
    (linear algebra)

Decoding

- Decoding step 1 (continued):
  - produce circuit \( C' \) from \( C \)
    - for at least \( \delta/2 \) of blocks, agreement in block is at least \( \delta/2 \)
    - Johnson Bound: when this happens, list size is \( S = \Omega(1/\delta^2) \), so probability \( C' \) correct is \( 1/\delta^2 \)
  - altogether:
    - \( \Pr_x[C'(x) = p_m(x)] \geq \Omega(\delta^3) \)
    - \( C' \) makes \( q \) queries to \( C \)
    - \( C' \) runs in time \( \text{poly}(q) \)

Decoding

- Decoding step 2 (continued):
  - produce circuit \( C'' \) from \( C' \)
    - given \( x \in F_q^t \) outputs “guess” for \( p_m(x) \)
    - \( C'' \) computes \( \{ z : \text{Had}(z) \text{ has agreement } \delta + \delta/2 \text{ with } x \text{-th block} \} \), outputs random \( z \) in this set

Decoding

- small circuit \( C \) computing \( R \), agreement \( 1/2 + \delta \)
- Decoding step 1
  - produce circuit \( C' \) from \( C \)
    - given \( x \in F_q^t \) outputs “guess” for \( p_m(x) \)
    - \( C' \) computes \( \{ z : \text{Had}(z) \text{ has agreement } 1/2 + \delta/2 \text{ with } x \text{-th block} \} \), outputs random \( z \) in this set

Decoding

- small circuit \( C' \) computing \( R' \), agreement \( \delta^3 = \Omega(\delta^3) \)
- Decoding step 2
  - produce circuit \( C'' \) from \( C' \)
    - given \( x \in \text{emb}(1,2,...,k) \) outputs \( p_m(x) \)
    - idea: restrict \( p_m \) to a random curve; apply efficient R-S list-decoding; fix “good” random choices
Restricting to a curve

- points \( x = \alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_r \in \mathbb{F}_q \) specify a degree \( r \) curve \( L : \mathbb{F}_q \rightarrow \mathbb{F}_q \)
- \( w_1, w_2, \ldots, w_r \) are distinct elements of \( \mathbb{F}_q \)
- for each \( i \), \( L_i : \mathbb{F}_q \rightarrow \mathbb{F}_q \) is the degree \( r \) poly for which \( L_i(w_j) = (j)_i \) for all \( j \)
- Write \( p_m(L(z)) \) to mean \( p_m(L_1(z), L_2(z), \ldots, L_t(z)) \)
- \( p_m(L(w_1)) = p_m(x) \)

Example:
- \( p_m(x_1, x_2) = x_1^2 x_2^2 + x_2 \)
- \( w_1 = 1, w_2 = 0 \)
- \( \alpha_1 = (2, 1) \)
- \( \alpha_2 = (1, 0) \)
- \( L_1(z) = 2z + 1(1 - z) = z + 1 \)
- \( L_2(z) = 1z + 0(1 - z) = z \)
- \( p_m(L(z)) = (z+1)^2 z^2 + z = z^4 + 2z^3 + z^2 + z \)

Decoding

- small circuit \( C' \) computing \( R' \), agreement \( \delta' = \Omega(\delta^3) \)
- Decoding step 2 (continued):
  - pick random \( w_1, w_2, \ldots, w_r; \alpha_2, \alpha_3, \ldots, \alpha_r \) to determine curve \( L \)
  - points on \( L \) are \((r-1)\)-wise independent
  - random variable: \( \text{Agr} = |\{z : C'(L(z)) = p_m(L(z))\}| \)
  - \( E[\text{Agr}] = \delta' q \) and \( \text{Pr}[\text{Agr} < (\delta' q)/2] < O(1/(\delta' q)^{r-1/2}) \)

Decoding

- Decoding step 2 (continued):
  - assuming \((\delta' q)/2 > (2r \cdot t \cdot q)^{1/2} \)
  - Reed-Solomon list-decoding step:
    - running time = \( \text{poly}(q) \)
    - list size \( S \leq 4/\delta' \)
  - probability list fails to contain \( p_m(L(\cdot)) \) is \( O(1/(\delta' q))^{r-1/2} \)