

CS151 Complexity Theory

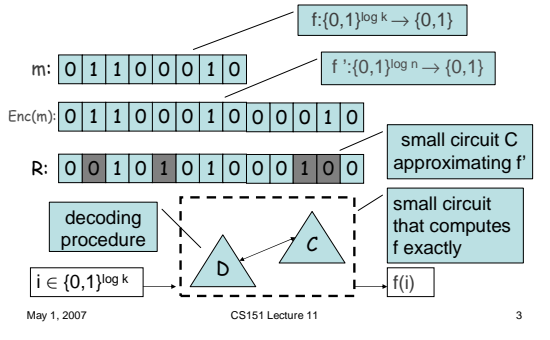
Lecture 11
May 1, 2007

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

Codes and Hardness



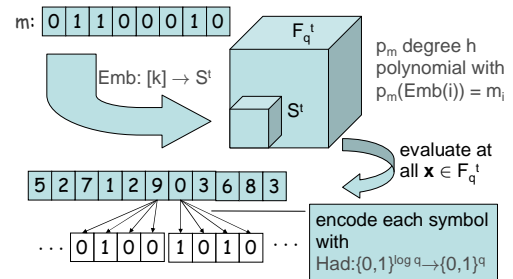
Encoding

- 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 $S \subseteq F_q^t$ of size h (so, need $h^t \geq k$)
 - efficient 1-1 function $Emb: [k] \rightarrow S^t$
 - find coeffs of degree h polynomial $p_m: F_q^t \rightarrow F_q$ for which $p_m(Emb(i)) = m_i$ for all i (linear algebra)

Encoding

- **encoding procedure (continued):**
 - Hadamard code $Had: \{0,1\}^{\log q} \rightarrow \{0,1\}^q$
 - = Reed-Muller with field size 2, dim. $\log q$, deg. 1
 - distance $\frac{1}{2}$ by Schwartz-Zippel
 - final codeword: $(Had(p_m(\mathbf{x})))_{\mathbf{x} \in F_q^t}$
 - evaluate p_m at all points, and encode each evaluation with the Hadamard code

Encoding



Decoding

Enc(m): 0 1 1 0 0 0 1 0 0 0 0 1

R: 0 0 1 0 1 0 1 0 0 0 1 0

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

May 1, 2007

CS151 Lecture 11

7

Decoding

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

May 1, 2007

CS151 Lecture 11

8

Decoding

p_m : 5 2 7 1 2 9 0 3 6 8 3

R: 5 9 7 1 6 9 0 3 6 8 1

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

May 1, 2007

CS151 Lecture 11

9

Restricting to a curve

– points $x = \alpha_1, \alpha_2, \alpha_3, \dots, \alpha_r \in F_q^t$ specify a degree r curve $L : F_q \rightarrow F_q^t$

- w_1, w_2, \dots, w_r are distinct elements of F_q

- for each i,

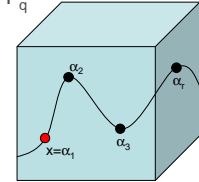
$$L_i : F_q \rightarrow F_q$$

is the degree r poly for which $L_i(w_j) = (\alpha_j)_i$ for all j

- Write $p_m(L(z))$ to mean

$$p_m(L_1(z), L_2(z), \dots, L_r(z))$$

$$p_m(L(w_1)) = p_m(x)$$



May 1, 2007

CS151 Lecture 11

10

Restricting to a curve

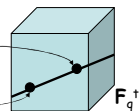
- Example:

$$- p_m(x_1, x_2) = x_1^2 x_2^2 + x_2$$

$$- w_1 = 1, w_2 = 0$$

$$\alpha_1 = (2, 1)$$

$$\alpha_2 = (1, 0)$$



$$- L_1(z) = 2z + 1(1-z) = z + 1$$

$$- L_2(z) = 1z + 0(1-z) = z$$

$$- p_m(L(z)) = (z+1)^2 z^2 + z = z^4 + 2z^3 + z^2 + z$$

May 1, 2007

CS151 Lecture 11

11

Decoding

p_m : 5 2 7 1 2 9 0 3 6 8 3

R: 5 9 7 1 6 9 0 3 6 8 1

- small circuit C' computing R', agreement $\delta' = \Omega(\delta^3)$
- **Decoding step 2 (continued):**
 - pick random $w_1, w_2, \dots, w_r; \alpha_2, \alpha_3, \dots, \alpha_r$ 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 $\Pr[\text{Agr} < (\delta'q)/2] < O(1/(\delta'q))^{(r-1)/2}$

May 1, 2007

CS151 Lecture 11

12

Decoding

p_m : 5 2 7 1 2 9 0 3 6 8 3

R : 5 9 7 1 6 9 0 3 6 8 1

- small circuit C' computing R' , agreement $\delta' = \Omega(\delta^3)$
- **Decoding step 2 (continued):**
 - $\text{agr} = |\{z : C'(L(z)) = p_m(L(z))\}| \geq (\delta'q)/2$ with very high probability
 - compute using Reed-Solomon list-decoding:

$$\{q(z) : \deg(q) \leq r-h-t, \Pr_{z \in L} [C'(L(z)) = q(z)] \geq (\delta'q)/2\}$$
 - if $\text{agr} \geq (\delta'q)/2$ then $p_m(L(\cdot))$ is in this set!

May 1, 2007

CS151 Lecture 11

13

Decoding

- **Decoding step 2 (continued):**
 - assuming $(\delta'q)/2 > (2r-h-t-q)^{1/2}$
 - 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 $O(1/(\delta q))^{(r-1)/2}$

May 1, 2007

CS151 Lecture 11

14

Decoding

- **Decoding step 2 (continued):**
 - Tricky:
 - functions in list are determined by the set $L(\cdot)$, independent of parameterization of the curve
 - Regard w_2, w_3, \dots, w_r as random points on curve L
 - for $q \neq p_m(L(\cdot))$

$$\Pr[q(w_i) = p_m(L(w_i))] \leq (rht)/q$$

$$\Pr[\forall i, q(w_i) = p_m(L(w_i))] \leq [(rht)/q]^{r-1}$$
 - $\Pr[\exists q \text{ in list s.t. } \forall i q(w_i) = p_m(L(w_i))] \leq (4/\delta')[(rht)/q]^{r-1}$

May 1, 2007

CS151 Lecture 11

15

Decoding

- **Decoding step 2 (continued):**
 - with probability $\geq 1 - O(1/(\delta q))^{(r-1)/2} - (4/\delta)[(rht)/q]^{r-1}$
 - list contains $q^* = p_m(L(\cdot))$
 - q^* is the *unique* q in the list for which

$$q(w_i) = p_m(L(w_i)) \quad (= p_m(\alpha_i)) \quad \text{for } i = 2, 3, \dots, r$$
 - circuit C'' :
 - hardwire $w_1, w_2, \dots, w_r; \alpha_2, \alpha_3, \dots, \alpha_r$ so that $\forall x \in \text{emb}(1, 2, \dots, k)$ both events occur
 - hardwire $p_m(\alpha_i)$ for $i = 2, \dots, r$
 - on input x , find q^* , output $q^*(w_1)$ ($= p_m(x)$)

May 1, 2007

CS151 Lecture 11

16

Decoding

- **Putting it all together:**
 - C approximating f' used to construct C'
 - C' makes q queries to C
 - C' runs in time $\text{poly}(q)$
 - C' used to construct C'' computing f exactly
 - C'' makes q queries to C'
 - C'' has $r-1$ elts of F_q^t and $2r-1$ elts of F_q hardwired
 - C'' runs in time $\text{poly}(q)$
 - C'' has size $\text{poly}(q, r, t, \text{size of } C)$

May 1, 2007

CS151 Lecture 11

17

Picking parameters

- k truth table size of f , hard for circuits of size s
- q field size, h R-M degree, t R-M dimension
- r degree of curve used in decoding
- $h^t \geq k$ (to accommodate message of length k)
- $\delta^6 q^2 > \Omega(rhtq)$ (for R-S list-decoding)
- $k[O(1/(\delta q))^{(r-1)/2} + (4/\delta)[(rht)/q]^{r-1}] < 1$
(so there is a “good” fixing of random bits)
- Pick: $h = s, t = (\log k)/(\log s)$
- Pick: $r = \Theta(\log k), q = \Theta(rht\delta^{-6})$

May 1, 2007

CS151 Lecture 11

18

Picking parameters

- k truth table size of f , hard for circuits of size s
- q field size, h R-M degree, t R-M dimension
- r degree of curve used in decoding
- $h = s, t = (\log k)/(\log s)$
- $r = \Theta(\log k), q = \Theta(rht\delta^{-6})$

$$\log k, \delta^{-1} < s$$

Claim: truth table of f' computable in time $\text{poly}(k)$
(so $f' \in \mathbf{E}$ if $f \in \mathbf{E}$).

- $\text{poly}(q)^t$ for R-M encoding
- $\text{poly}(q) \cdot q^t$ for Hadamard encoding
- $q \leq \text{poly}(s)$, so $q^t \leq \text{poly}(s)^t = \text{poly}(h)^t = \text{poly}(k)$

May 1, 2007

CS151 Lecture 11

19

Picking parameters

- k truth table size of f , hard for circuits of size s
- q field size, h R-M degree, t R-M dimension
- r degree of curve used in decoding
- $h = s, t = (\log k)/(\log s)$
- $r = \Theta(\log k), q = \Theta(rht\delta^{-6})$

$$\log k, \delta^{-1} < s$$

Claim: f' s' -approximable by C implies f
computable exactly in size s by C'' , for $s' = s^{\Omega(1)}$

- C has size s' and agreement $\delta = 1/s'$ with f'
- C'' has size $\text{poly}(q, r, t, \text{size of } C) = s$

May 1, 2007

CS151 Lecture 11

20

Putting it all together

Theorem 1 (IW, STV): If \mathbf{E} contains functions that require size $2^{\Omega(n)}$ circuits, then \mathbf{E} contains $2^{\Omega(n)}$ -unapproximable functions.
(proof on next slide)

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

Theorem (IW): \mathbf{E} requires exponential size circuits $\Rightarrow \mathbf{BPP} = \mathbf{P}$.

May 1, 2007

CS151 Lecture 11

21

Putting it all together

- Proof of Theorem 1:

- let $f = \{f_n\}$ be hard for size $s(n) = 2^{\Omega(n)}$ circuits
- define $f' = \{f'_n\}$ to be just-described encoding of (the truth tables of) $f = \{f_n\}$
- two claims we just showed:
 - f' is in \mathbf{E} since f is.
 - if f' is $s'(n) = 2^{\Omega(n)}$ -approximable, then f is computable exactly by size $s(n) = 2^{\Omega(n)}$ circuits.
- contradiction.

May 1, 2007

CS151 Lecture 11

22

Extractors

- PRGs: can remove randomness from algorithms
 - based on unproven assumption
 - polynomial slow-down
 - not applicable in other settings
- Question: can we use “real” randomness?
 - physical source
 - imperfect – biased, correlated

May 1, 2007

CS151 Lecture 11

23

Extractors



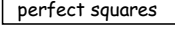
- “Hardware” side
 - what physical source?
 - ask the physicists...
- “Software” side
 - what is the minimum we need from the physical source?

May 1, 2007

CS151 Lecture 11

24

Extractors

- imperfect sources:
 - “stuck bits”: 
 - “correlation”: 
 - “more insidious correlation”: 
- there are specific ways to get independent unbiased random bits from specific imperfect physical sources

May 1, 2007

CS151 Lecture 11

25

Extractors

- want to assume we don't know details of physical source
- **general model** capturing all of these?
 - yes: “min-entropy”
- **universal procedure** for all imperfect sources?
 - yes: “extractors”

May 1, 2007

CS151 Lecture 11

26

Min-entropy

- General model of physical source w/ $k < n$ bits of hidden randomness

string sampled uniformly from this set



Definition: random variable X on $\{0,1\}^n$ has **min-entropy** $\min_x -\log(\Pr[X = x])$

- min-entropy k implies no string has weight more than 2^{-k}

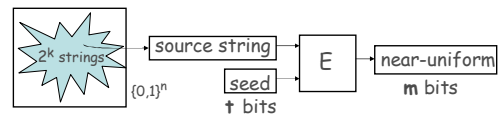
May 1, 2007

CS151 Lecture 11

27

Extractor

- Extractor: universal procedure for “purifying” imperfect source:



- E is efficiently computable
- truly random seed as “catalyst”

May 1, 2007

CS151 Lecture 11

28

Extractor

“(k, ϵ)-extractor” \Rightarrow for all X with min-entropy k :

- output fools **all** circuits C :

$$|\Pr_z[C(z) = 1] - \Pr_{y, x \leftarrow X}[C(E(x, y)) = 1]| \leq \epsilon$$

- distributions $E(X, U_t), U_m$ “ ϵ -close” (L_1 dist $\leq 2\epsilon$)

- Notice similarity to PRGs
 - output of PRG fools all **efficient** tests
 - output of extractor fools **all** tests

May 1, 2007

CS151 Lecture 11

29

Extractors

- Using extractors
 - use output in place of randomness in any application
 - alters probability of **any** outcome by at most ϵ
- Main motivating application:
 - use output in place of randomness in algorithm
 - how to get truly random seed?
 - enumerate all seeds, take majority

May 1, 2007

CS151 Lecture 11

30

Extractors

2^k strings
 $\{0,1\}^n$ source string
 seed t bits
 E
 near-uniform m bits

- Goals:

short seed	good: $O(\log n)$	best: $\log n + O(1)$
long output	$m = k^{\Omega(1)}$	$m = k + t - O(1)$
many k's	$k = n^{\Omega(1)}$	any $k = k(n)$

May 1, 2007 CS151 Lecture 11 31

Extractors

- random function for E achieves best !
 - but we need **explicit** constructions
 - many known; often complex + technical
 - optimal extractors still open
- Trevisan Extractor:
 - insight: use NW generator with source string in place of hard function
 - this works (!!)
 - proof slightly different than NW, easier

May 1, 2007 CS151 Lecture 11 32

Trevisan Extractor

- Ingredients: $(\delta > 0$ is a parameter)
 - error-correcting code
 - $C: \{0,1\}^n \rightarrow \{0,1\}^{n'}$
 - distance $(\frac{1}{2} - \frac{1}{4}m^{-4})n'$ blocklength $n' = \text{poly}(n)$
 - $(\log n', a = \delta \log n/3)$ design:
 - $S_1, S_2, \dots, S_m \subset \{1 \dots t = O(\log n')\}$

$$E(x, y) = C(x)[y_{|S_1}] \oplus C(x)[y_{|S_2}] \oplus \dots \oplus C(x)[y_{|S_m}]$$

May 1, 2007 CS151 Lecture 11 33

Trevisan Extractor

$$E(x, y) = C(x)[y_{|S_1}] \oplus C(x)[y_{|S_2}] \oplus \dots \oplus C(x)[y_{|S_m}]$$

$C(x)$: 010100101111101010111001010

seed y

Theorem (T): E is an extractor for min-entropy $k = n^\delta$, with

- output length $m = k^{1/3}$
- seed length $t = O(\log n)$
- error $\epsilon \leq 1/m$

May 1, 2007 CS151 Lecture 11 34

Trevisan Extractor

- Proof:
 - given $X \subseteq \{0,1\}^n$ of size 2^k
 - assume E fails to ϵ -pass statistical test C
 - $|\Pr_z[C(z) = 1] - \Pr_{x \in X, y}[C(E(x, y)) = 1]| > \epsilon$
 - distinguisher C \Rightarrow predictor P:
 - $\Pr_{x \in X, y}[P(E(x, y)_{1 \dots i-1}) = E(x, y)_i] > \frac{1}{2} + \epsilon/m$

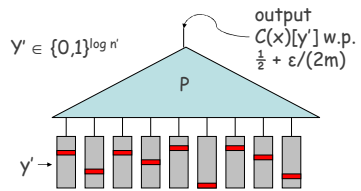
May 1, 2007 CS151 Lecture 11 35

Trevisan Extractor

- Proof (continued):
 - for at least $\epsilon/2$ of $x \in X$ we have:
 - $\Pr_y[P(E(x, y)_{1 \dots i-1}) = E(x, y)_i] > \frac{1}{2} + \epsilon/(2m)$
 - fix bits α, β outside of S_i to preserve advantage
 - $\Pr_y[P(E(x; \alpha y' \beta)_{1 \dots i-1}) = C(x)[y'_i]] > \frac{1}{2} + \epsilon/(2m)$
 - as vary y' , for $j \neq i$, j -th bit of $E(x; \alpha y' \beta)$ varies over only 2^a values
 - build up to $(m-1)$ tables of 2^a values to supply $E(x; \alpha y' \beta)_{1 \dots i-1}$

May 1, 2007 CS151 Lecture 11 36

Trevisan Extractor



May 1, 2007

CS151 Lecture 11

37

Trevisan Extractor

- Proof (continued):
 - $(m-1)$ tables of size 2^a constitute a **description** of a string that has $\frac{1}{2} + \epsilon/(2m)$ agreement with $C(x)$
 - # of strings x with such a description?
 - $\exp((m-1)2^a) = \exp(n^{\delta/3}) = \exp(k^{2/3})$ strings
 - Johnson Bound: each string accounts for at most $O(m^4)$ x 's
 - total #: $O(m^4)\exp(k^{2/3}) \ll 2^k(\epsilon/2)$
 - contradiction

May 1, 2007

CS151 Lecture 11

38