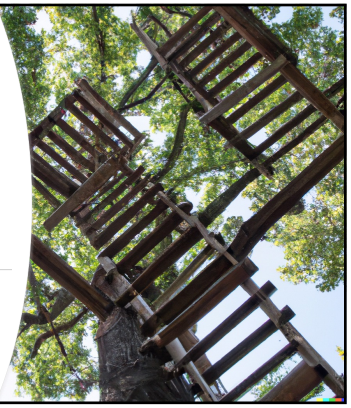
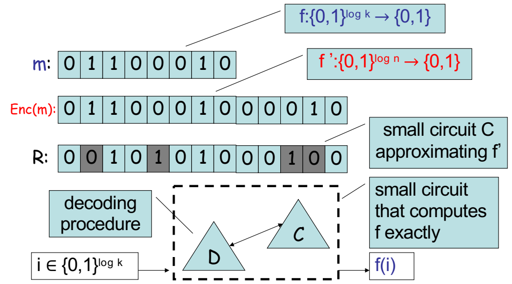
CS151 Complexity Theory



Lecture11
May 9, 2023

1

Codes and Hardness



2

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

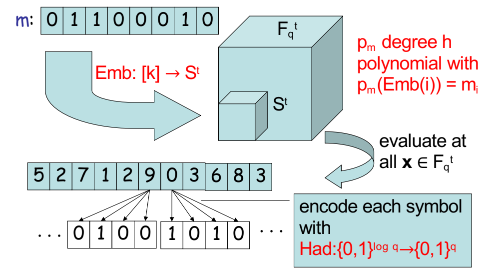
3

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 ½ by Schwartz-Zippel
  - final codeword: $(\text{Had}(p_m(\mathbf{x})))_{x \in F_q^t}$
    - evaluate p_m at all points, and encode each evaluation with the Hadamard code

4

Encoding



5

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 $\mathbf{x} \in F_q^t$ outputs "guess" for $p_m(\mathbf{x})$
    - C' computes {z : Had(z) has agreement $\frac{1}{2} + \delta/2$ with x-th block}, outputs random z in this set

6

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 poly(q)

7

Decoding

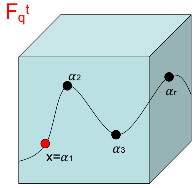
p_m:	5	2	7	1	2	9	0	3	6	8	3
R':	5	9	7	1	6	9	0	3	6	8	1

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

8

Restricting to a curve

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

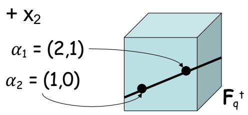


degree r·h·t *univariate* poly

9

Restricting to a curve

- Example:
  - $p_m(x_1, x_2) = x_1^2 x_2^2 + x_2$
  - $w_1 = 1, w_2 = 0$



  - $L_1(z) = 2z + 1(1-z) = z + 1$
  - $L_2(z) = 1z + 0(1-z) = z$
  - $p_m(L(z)) = (z+1)^2 z^2 + z = z^4 + 2z^3 + z^2 + z$

10

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 $w_1, w_2, \ldots, w_r; \alpha_2, \alpha_3, \ldots, \alpha_r$ to determine curve L
  - points on L are (r-1)-wise independent
  - random variable: $Agr = |\{z : C'(L(z)) = p_m(L(z))\}|$
  - $E[Agr] = \delta' q$ and $\Pr[Agr < (\delta' q)/2] < O(1/(\delta' q))^{(r-1)/2}$

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):
  - $agr = |\{z : C'(L(z)) = p_m(L(z))\}|$ is $\geq (\delta' q)/2$ with very high probability
  - compute using Reed-Solomon list-decoding:

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

12

Decoding

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

13

Decoding

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

$$\Pr[q(w_i) = p_m(L(w_i))] \leq (rht)/q$$
$$\Pr[\forall\, i,\ q(w_i) = p_m(L(w_i))] \leq [(rht)/q]^{r-1}$$

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

14

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

15

Decoding

- Putting it all together:
  - C approximating f' used to construct C'
    - C' makes q queries to C
    - C' runs in time 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 poly(q)
  - C'' has size poly(q, r, t, size of C)

16

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 accomodate 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: $r = \Theta(\log k)$, $q = \Theta(rht\delta^{-6})$

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
- $h = s$, $t = (\log k)/(\log s)$
- $r = \Theta(\log k)$, $q = \Theta(rht\delta^{-6})$

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

Claim: truth table of f' computable in time poly(k)

(so f' ∈ **E** if f ∈ **E**).
- poly(q^t) for R-M encoding
- poly(q)·q^t for Hadamard encoding
- $q \leq \text{poly}(s)$, so $q^t \leq \text{poly}(s)^t = \text{poly}(h)^t = \text{poly}(k)$

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 = Θ(log k), q = $\Theta(rht\delta^{-6})$

log k, δ^{-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 δ=1/s' with f'
- C'' has size poly(q, r, t, size of C) = s

19

Putting it all together

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

(proof on next slide)

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

Theorem (IW): **E** requires exponential size circuits ⇒ **BPP** = **P**.

20

Putting it all together

- Proof of Theorem 1:
  - let f = {f_n} be hard for size s(n) = $2^{\delta n}$ 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 **E** since f is.
    - if f ' is s'(n) = $2^{\delta' n}$-approximable, then f is computable exactly by size s(n) = $2^{\delta n}$ circuits.
  - contradiction.

21

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

22

Extractors

- "Hardware" side
  - what physical source?
  - ask the physicists…
- "Software" side
  - what is the minimum we need from the physical source?

23

Extractors

- imperfect sources:
  - "stuck bits":
  - "correlation":
  - "more insidious correlation": perfect squares

- there are specific ways to get independent unbiased random bits from specific imperfect physical sources

24

Extractors

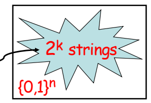
- want to assume we don't know details of physical source
- **general model** capturing all of these?
  - yes: "min-entropy"
- **universal procedure** for all imperfect sources?
  - yes: "extractors"

25

Min-entropy

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

string sampled uniformly from this set



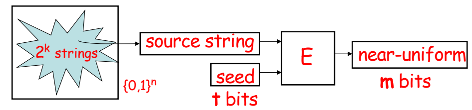
Definition: random variable X on $\{0,1\}^n$ has **min-entropy** $\min_x -\log(\Pr[X = x])$

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

26

Extractor

- Extractor: universal procedure for "purifying" imperfect source:



- E is efficiently computable
- truly random seed as "catalyst"

27

Extractor

"(k, ε)-extractor" ⇒ for all X with min-entropy k:

- output fools **all** circuits C:

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

- distributions $E(X, U_t)$, U_m "**ε**-close" (L_1 dist $\le 2\boldsymbol{\varepsilon}$)

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

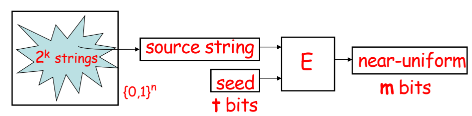
28

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

29

Extractors



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

30

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

31

Trevisan Extractor

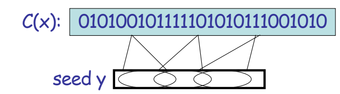
- Ingredients: (δ > 0, m are parameters)
  - error-correcting code
    $C:\{0,1\}^n \rightarrow \{0,1\}^{n'}$
    distance $(½ - ¼m^{-4})n'$ blocklength $n' = poly(n)$
  - (log n', a = δlog n/3) design:
    $S_1, S_2, \ldots, S_m \in \{1\ldots t = O(\log n')\}$

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

32

Trevisan Extractor

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



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 $\varepsilon \leq 1/m$

33

Trevisan Extractor

- Proof:
  - given $X \subseteq \{0,1\}^n$ of size 2^k
  - assume E fails to ε-pass statistical test C
    $|Pr_z[C(z) = 1] – Pr_{x\leftarrow X, y}[C(E(x, y)) = 1]| > \varepsilon$
  - **distinguisher** C ⇒ **predictor** P:
    $Pr_{x\leftarrow X, y}[P(E(x, y)_{1\ldots i-1})=E(x, y)_i] > ½ + \varepsilon/m$

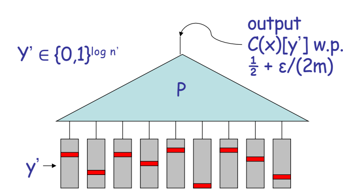
34

Trevisan Extractor

- Proof (continued):
  - for at least ε/2 of x ∈ X we have:
    $Pr_y[P(E(x, y)_{1\ldots i-1})=E(x, y)_i] > ½ + \varepsilon/(2m)$
  - fix bits α,β outside of S_i to preserve advantage
    $Pr_{y'}[P(E(x; \alpha y'\beta)_{1\ldots i-1})=C(x)[y']] > ½ + \varepsilon/(2m)$
  - as vary y', for j ≠ i, j-th bit of $E(x; \alpha y'\beta)$ varies over only 2^a values
  - (m-1) tables of 2^a values supply $E(x;\alpha y'\beta)_{1\ldots i-1}$

35

Trevisan Extractor



36

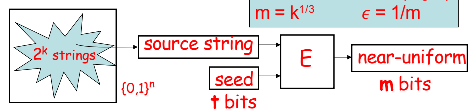
Trevisan Extractor

- Proof (continued):
  - (m-1) tables of size 2^a constitute a **description** of a string that has $½ + ε/(2m)$ agreement with C(x)
  - # of strings x with such a description?
    - $\exp((m-1)2^a) = \exp(n^{\delta 2/3}) = \exp(k^{2/3})$ strings
    - Johnson Bound: each string accounts for at most $O(m^4)$ x's
    - total #: $O(m^4)\exp(k^{2/3}) << 2^k(\varepsilon/2)$
    - contradiction

Extractors

- (k, ϵ)- extractor:

Trevisan:
$k = n^\delta$ $t = O(\log n)$
$m = k^{1/3}$ $\epsilon = 1/m$



  - E is efficiently computable
  - $\forall$ X with minentropy k, E fools **all** circuits C:

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

Strong error reduction

- $L \in$ **BPP** if there is a p.p.t. TM M:
  - $x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \ge 2/3$
  - $x \forall L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] \ge 2/3$
- Want:
  - $x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \ge 1 - 2^{-k}$
  - $x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] \ge 1 - 2^{-k}$
- We saw: repeat O(k) times
  - $n = O(k)\cdot|y|$ random bits; **2^{n-k} bad strings**

Want to spend $n = \text{poly}(|y|)$ random bits; achieve $<< 2^{n/3}$ bad strings

Strong error reduction

- Better:
  - E extractor for minentropy $k=|y|^3=n^\delta$, $\varepsilon < 1/6$
  - pick random $w \in \{0,1\}^n$, run $M(x, E(w, z))$ for all $z \in \{0,1\}^t$, take majority
  - call w "bad" if $\text{maj}_z M(x, E(w, z))$ incorrect
    $|\Pr_z[M(x,E(w,z))=b] - \Pr_y[M(x,y)=b]| \ge 1/6$
  - extractor property: at most 2^k bad w
  - n random bits; **2^{n^δ} bad strings**