CS151
Complexity Theory

Lecture 11

May 4, 2021

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.

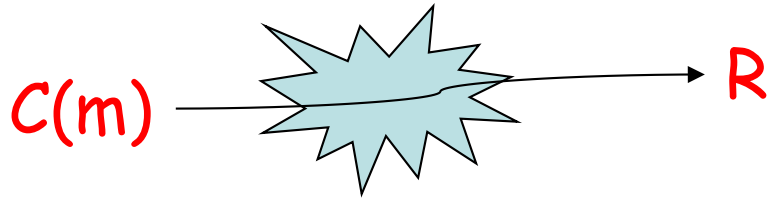
- Proof:
  - main tool: error correcting code

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

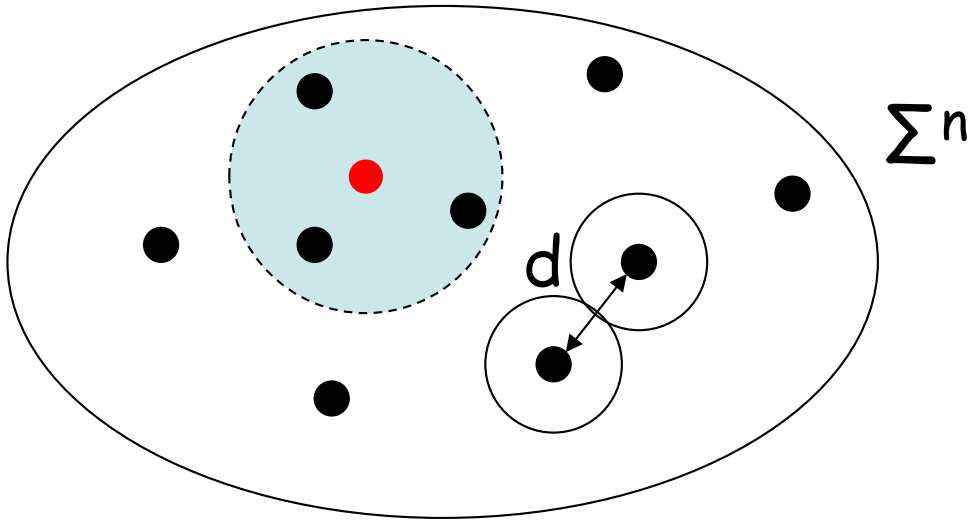


- if not too many errors, can decode: D(R) = m
- parameters of interest:
  - rate: k/n
  - distance:

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

Distance and error correction

- C is an ECC with distance d
- can **uniquely** decode from up to ⌊d/2⌋ 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.

Example: Reed-Muller

- Parameters: t (dimension), h (degree)
- alphabet $\Sigma = \mathbf{F}_q$: field with q elements
- message $m \in \Sigma^k$
- multivariate polynomial of total degree at most h:

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

$\{M_i\}$ are all monomials of degree $\leq h$

Example: Reed-Muller

- M_i is monomial of total degree h
  - e.g. $x_1^2 x_2 x_4^3$
  - need # monomials (h+t choose t) > k
- codeword $C(m) = (p_m(x))_{x \in (\mathbf{F}_q)^t}$
- rate = k/q^t
- 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: | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

truth table of $f':\{0,1\}^{\log n} \rightarrow \{0,1\}$

(average-case hard)

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

Codes and Hardness

- if n = poly(k) then

  f ∈ **E** implies f' ∈ **E**

- Want to be able to prove:

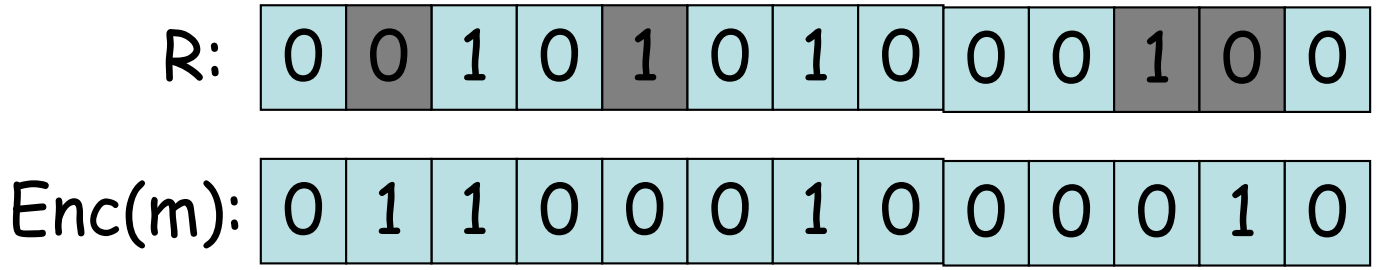
  if f' is s'-approximable,

  then f is computable by a

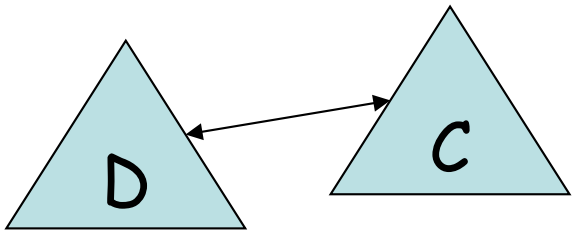
  size s = poly(s') circuit

Codes and Hardness

- Key: circuit C that approximates f' implicitly gives received word R

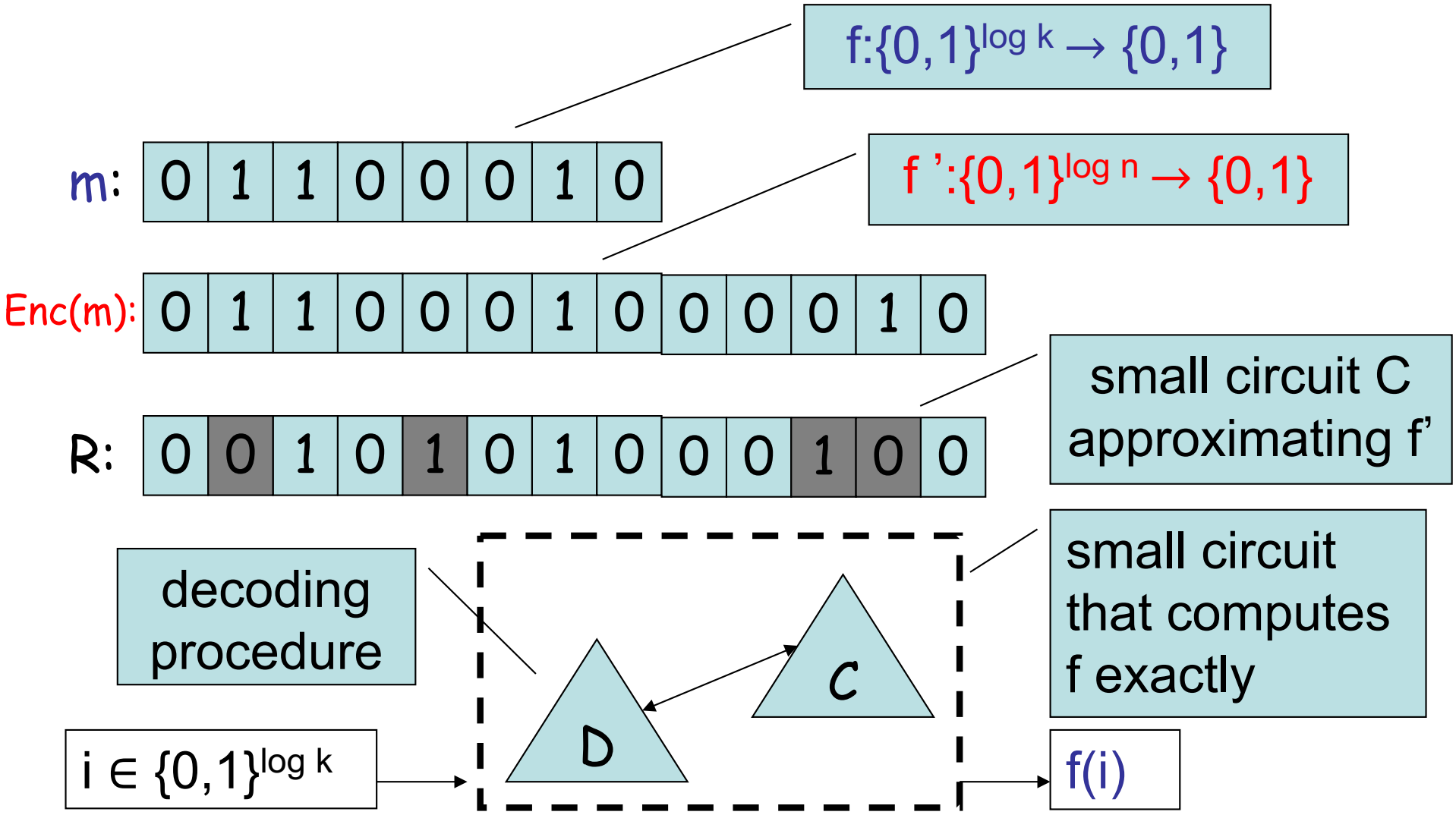


- Decoding procedure D "computes" f exactly



• Requires special notion of efficient decoding

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 $S \subseteq F_q$ of size h

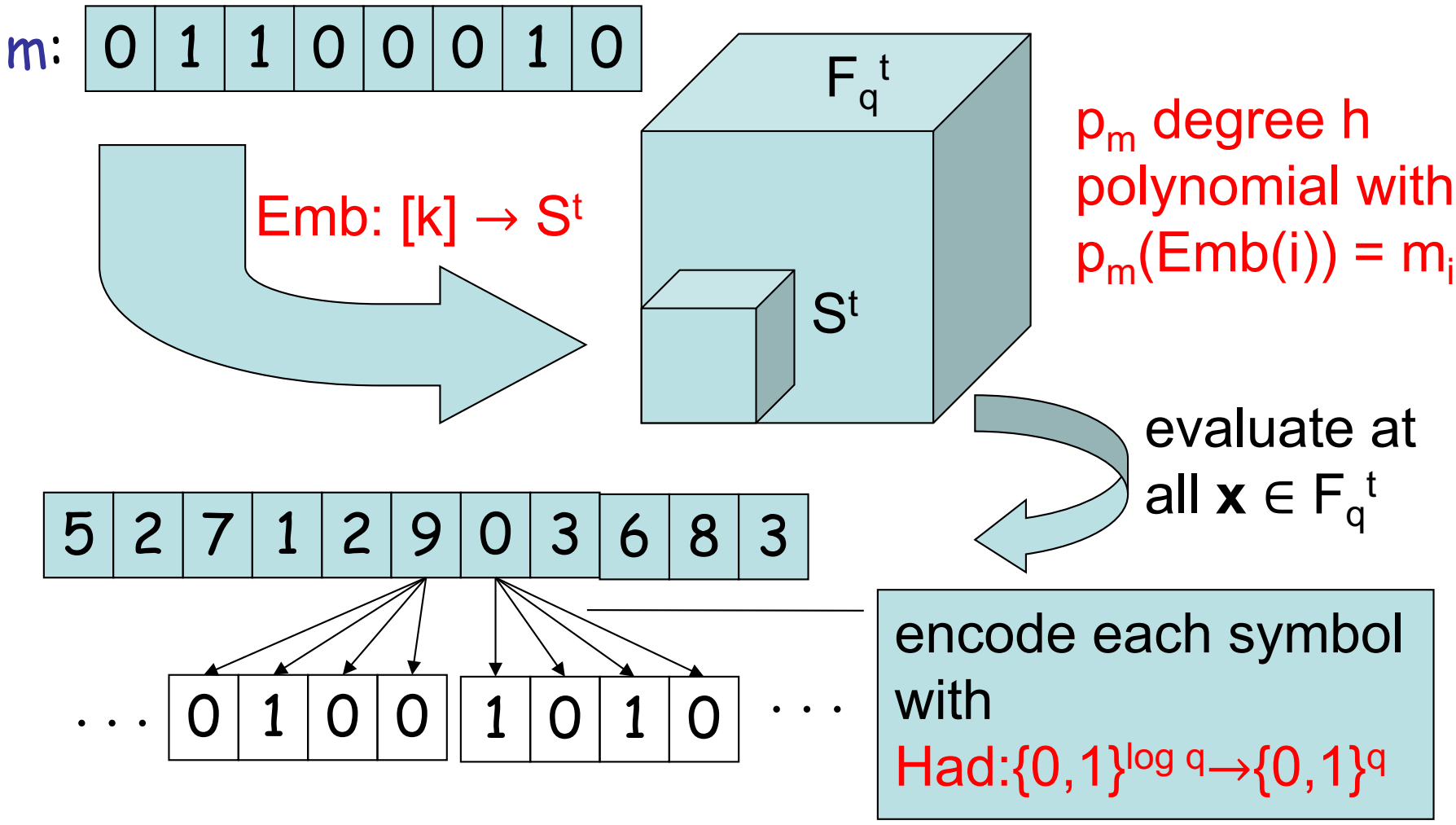
so, need $h^t \geq k$

  - 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)

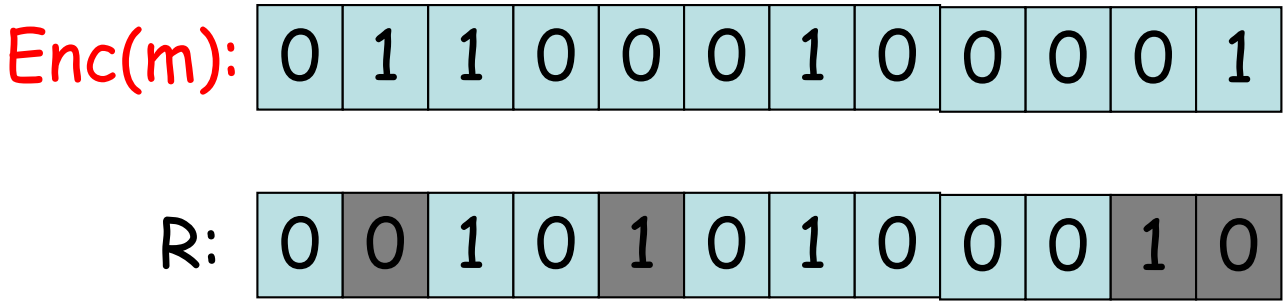
Encoding

- **encoding procedure** (continued):
  - Hadamard code $\mathrm{Had}:\{0,1\}^{\log q} \to \{0,1\}^{q}$
    - = Reed-Muller with field size 2, dim. log q, deg. 1
    - distance ½ by Schwartz-Zippel
  - final codeword: $(\mathrm{Had}(p_m(\mathbf{x})))_{x \in F_q^t}$
    - evaluate p_m at all points, and encode each evaluation with the Hadamard code

Encoding


m:
0 1 1 0 0 0 1 0
Emb: [k] → S^t
F_q^t
S^t
p_m degree h polynomial with p_m(Emb(i)) = m_i
evaluate at all x ∈ F_q^t
5 2 7 1 2 9 0 3 6 8 3
. . .
0 1 0 0 1 0 1 0
. . .
encode each symbol with
Had:{0,1}^log q→{0,1}^q

Decoding

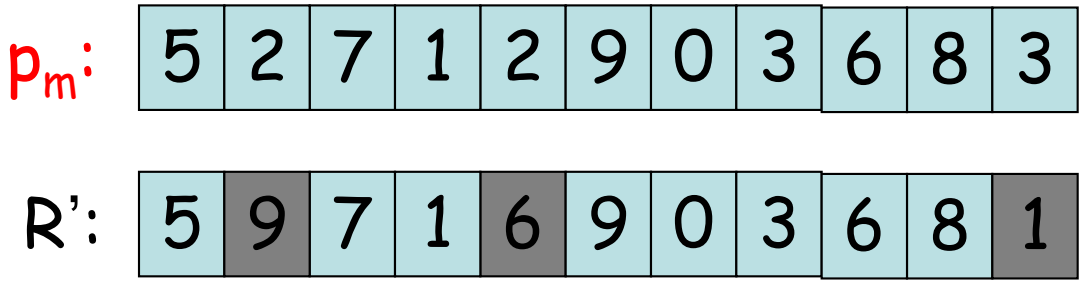


- small circuit C computing R, agreement $\frac{1}{2} + \delta$
- **Decoding step 1**
  - produce circuit C' from C
    - given $\mathbf{x} \in F_q^t$ outputs "guess" for $p_m(\mathbf{x})$
    - C' computes {z : Had(z) has agreement $\frac{1}{2} + \delta/2$ with x-th block}, outputs random z in this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

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)

Decoding



- small circuit C' computing R', agreement $\delta' = \Omega(\delta^3)$
- **Decoding step 2**
  - produce circuit C'' from C'
    - given **x** ∈ 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

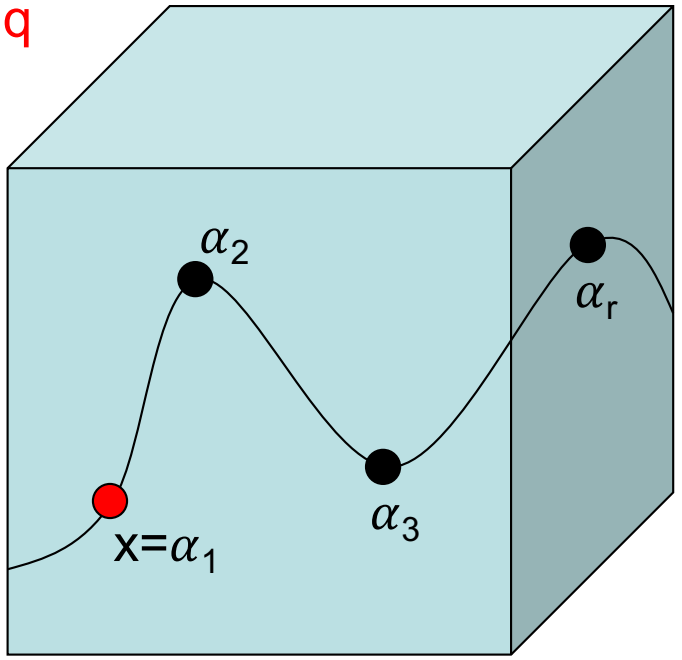
Restricting to a curve

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

- $w_1, w_2, \ldots, w_r$ 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

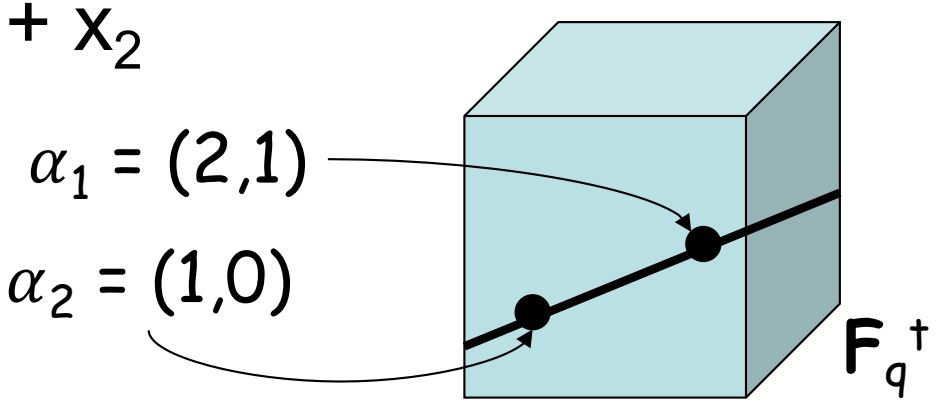
- Write $p_m(L(z))$ to mean $p_m(L_1(z), L_2(z), \ldots, L_t(z))$
- $p_m(L(w_1)) = p_m(x)$



degree r·h·t *univariate* poly

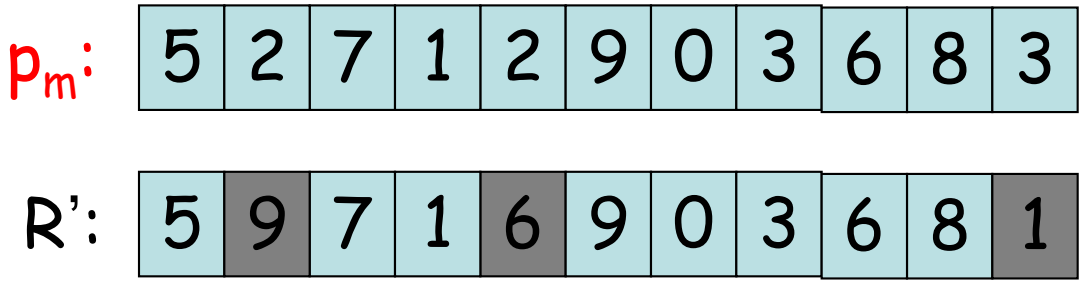
Restricting to a curve

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



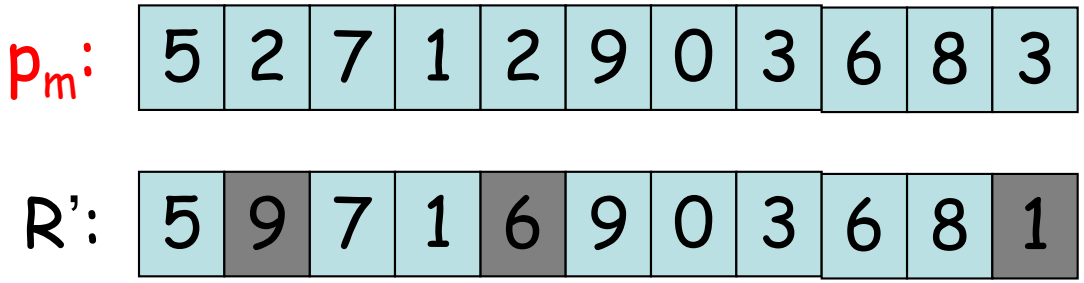
  - $L_1(z) = 2z + 1(1-z) = z + 1$
  - $L_2(z) = 1z + 0(1-z) = z$
  - $p_m(L(z)) = (z+1)^2 z^2 + z = z^4 + 2z^3 + z^2 + z$

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: $Agr = |\{z : C'(L(z)) = p_m(L(z))\}|$
  - $E[Agr] = \delta' q$ and $Pr[Agr < (\delta' q)/2] < O(1/(\delta' q))^{(r-1)/2}$

Decoding



- 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!

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 $O(1/(\delta q))^{(r-1)/2}$

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 $q \neq p_m(L(\cdot))$

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

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

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

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

      $$q(w_i) = p_m(L(w_i)) \; (= p_m(\alpha_i)) \text{ for } i = 2,3,\ldots,r$$

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

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)

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

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, δ^{-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
- $q \leq$ poly(s), so $q^t \leq$ poly(s)^t = poly(h)^t = poly(k)

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

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 $\Rightarrow$ **BPP** = **P**.

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.