

# MULTIVARIATE POLYNOMIALS AND INTRO TO CODING THEORY

- Recap:
- What are fields?
  - Fields of prime order
  - Univariate polynomials
  - Extension fields: fields of non-prime order
  - Application to communication complexity.

- Today:
- Multivariate polynomials
  - Schwartz-Zippel & application to polynomial identity testing
  - Error Correcting Codes
    - ↳ Basic definitions & notations
    - ↳ Linear codes
    - ↳ Properties of linear codes.

## References

- Essential Coding Theory - Guruswami, Rudra & Sudan
- Ryan's Lecture 12
- Generally, can find Venkat Guruswami's or Madhu Sudan's online lectures to learn coding theory.

# Multivariate Polynomials

## Polynomials vs Functions computed by them

- Every polynomial  $p(x_1, \dots, x_n) \in \mathbb{F}_q[x_1, \dots, x_n]$  computes a function  $f: \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ .
- Conversely, every function  $f: \mathbb{F}_q^n \rightarrow \mathbb{F}_q$  is computable by some polynomial  
(we saw for  $\mathbb{F}_2^n$ ,  $f$  computable by multilinear polynomials  $\rightarrow$  Fourier analysis can be generalized to  $\mathbb{F}_q^n$ )

$$|\{f: \mathbb{F}_q^n \rightarrow \mathbb{F}_q\}| = q^{q^n}$$

# polynomials in  $\mathbb{F}_q[x_1, \dots, x_n]$  : infinite

- $\Rightarrow \nexists$  some functions  $f$  computable by two distinct polynomials,  $p \neq q$  (what about  $p - q$ ?)  
This might confuse you: why?  
 $\nexists$  formally non-zero polynomials computing the everywhere-zero function!

Ex:  $x^2 - x$  in  $\mathbb{F}_2[x]$  is a non-zero polynomial computing the zero function.

$x^2 - x$  in  $\mathbb{F}_2[x]$   $\rightarrow$  why does it compute the zero function?  
(Exercise)

what this means?

$x^2$  &  $x$  compute the same value in  $\mathbb{F}_2$   
(i.e. functionally can replace  $x^2$  by  $x$ )

Corollary (Generalizing something we noted in Boolean world)  
 Given any polynomial  $p \in \mathbb{F}_q[x_1, \dots, x_n]$ , can reduce each  $x_i^m$  to  $x_i^{m \bmod (q-1)}$  changing function computed.  
 ↳ all individual degrees  $\leq q-1$  (for  $q > 2$ )

∴ By using this corollary, # reduced monomials is  $q^n$ .  
 (In boolean world ( $q=2$ ) then all individual degrees at most 1 & we have  $2^n$  monomials)

⇒ # reduced polynomials =  $q^{q^n}$

⇒ Every function  $f: \mathbb{F}_q^n \rightarrow \mathbb{F}_q$  has unique reduced polynomial representation.

## Schwartz-Zippel (L Ore, DeMillo, Lipton)

(about roots of multivariate polynomials)

Let  $p \in \mathbb{F}_q[x_1, \dots, x_n]$  be a non-zero polynomial over  $\mathbb{F}_q$  with total degree  $\leq d$ . Then

$$\Pr_{d_1, \dots, d_n \sim \mathbb{F}_q} [P(d_1, \dots, d_n) = 0] \leq \frac{d}{q}$$

Exercise

→ prove using induction on  $n$

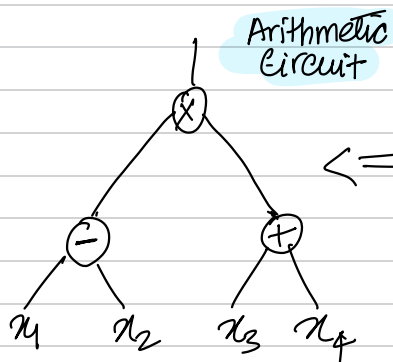
this means at a random point, the polynomial evaluates to 0 with very low probability if  $q \gg d$

A variation (specially useful for infinite fields):  
 For any  $S \subseteq \mathbb{F}$ ,  $\forall p \in \mathbb{F}[x_1, \dots, x_n]$  (non-zero & total deg  $d$ )

$$\Pr_{\alpha_1, \dots, \alpha_n \sim S} [P(\alpha_1, \dots, \alpha_n) = 0] \leq \frac{d}{|S|}$$

Special case

# Application of Schwartz-Zippel : Polynomial Identity Testing (PIT)



As a polynomial

$\Leftrightarrow$

$$p(x_1, x_2, x_3) = (x_1 - x_2) \times (x_3 + x_4) \\ = x_1 x_3 - x_2 x_3 + x_1 x_4 - x_2 x_4$$

(3-variable polynomial)

PIT : It checks if an arithmetic circuit/polynomial is identically the zero polynomial.  
(In general iff  $p \equiv q$  for polynomials  $p \neq q$ )

Circuit  $C$  with  $m$ -gates and  $n$  variables.

$\Rightarrow$

Polynomial  $p_C$  of degree  $\leq 2^m$   
 $\because$  Each gate has fan-in 2

$\therefore$  One can use Schwartz-Zippel to check if  $C \equiv 0$   
(the probability tells you, what size set  $S$  you should take of  $(S: \text{set of points from which you pick one at random \& check})$ )

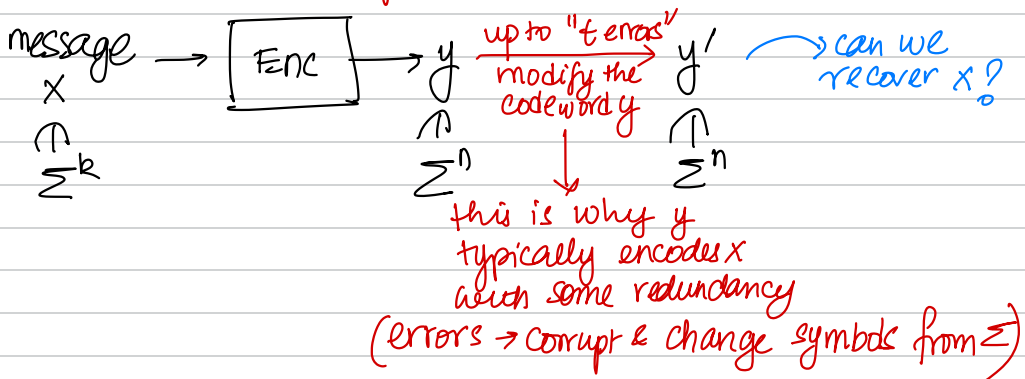
Need to be careful about what field you want to evaluate on, but I won't go into these details here, in the interest of time.  
If interested  $\rightarrow$  refer to **[Complexity Theory by Arora-Barak]**

## ERROR CORRECTING CODES

Definition: An error correcting code (ECC) is an injective map  $\text{Enc}: \Sigma^k \rightarrow \Sigma^n$ , where

- $\Sigma$  = alphabet,  $q := |\Sigma|$  (often  $q=2$ )
- $\Sigma^k$  = message space,  $k$ : msg length/dimension
- $n$  = (block) length
- $C := \text{range}(\text{Enc}) \subseteq \Sigma^n$  called the code
- $y \in C$  is called a codeword.
- $\boxed{\frac{k}{n} = \text{rate}}$  (For efficiency, it is clear that we want rate to be large.)

larger rate  $\Rightarrow$  you are adding lesser redundancy to the message  $\rightarrow$  desirable

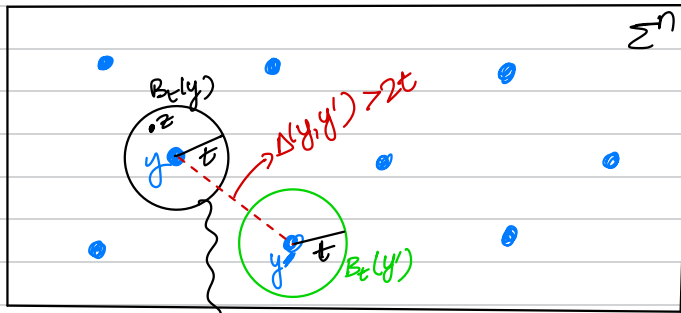


Remark. In EE, typically these errors are viewed as random errors (or from a distribution). In CS, the errors are "worst-case": the adversary can arbitrarily modify up to  $t$  symbols.

What is a good coding scheme & what  $t$  can you tolerate?  $\rightarrow$  need to define distance.

## Definition (Hamming Distance)

$$\Delta(y, z) = \#\{i : y_i \neq z_i\}$$



$$B_t(y) : \text{Hamming Ball of radius } t \text{ centered at } y = \{z \in \Sigma^n : \Delta(y, z) \leq t\}$$

$\Rightarrow$   $t$ -errors means the received  $z \in B_t(y)$

Q: When can you hope to recover the original  $y$ , given the string  $z \in B_t(y)$ ?

As long as  $z$  is not within distance  $t$  of a different codeword!  
i.e. as long as all these "hamming balls of radius  $t$  around each of the codewords are disjoint, a decoder can in principle recover a codeword from the received  $z$ .

$$\Rightarrow \forall y, y' \in C, \Delta(y, y') > 2t$$

This is formalized by minimum distance, which captures how many errors  $t$  can be tolerated, given a codeword, to be able to decode!

Definition (Minimum Distance):  $d = \min_{y \neq y' \in C} \{\Delta(y, y')\}$

The above discussion essentially shows:

Fact: "Unique decoding" possible iff  $t < \frac{d}{2}$

(given  $z$  with  $\Delta(y, z) \leq t$ , find  $y$ )

Discuss on board: examples  $\rightarrow$  parity code & repetition code.

Remark (Rate vs Min Distance)

While you want is a larger min distance, to be able to tolerate more errors but this conflicts with the rate (i.e., leads to smaller rate) more redundancy.

A common solution (often to show feasibility) is to pick things at random.

Q: What if we pick  $C \subseteq \mathbb{Z}^n$  uniformly at random?

$\rightarrow$  Gives great minimum distance & rate tradeoff.  
 $\rightarrow$  But, lacks efficient encoding/decoding.

Goal: Want to construct explicit codes with high rate & distance & fast encoding/decoding.

LINEAR ERROR CORRECTING CODES

(Important class of ECCs: almost all best ECCs are linear)

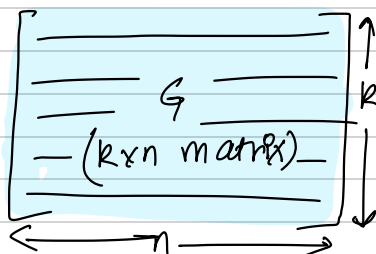
$\Sigma = \mathbb{F}_q$  (finite field) &

Enc:  $\mathbb{F}_q^k \rightarrow \mathbb{F}_q^n$  is a linear transformation.

i.e. encoding is just multiplication with a "generator matrix."

$\text{Enc}(x) := xG$ , where  $G$  is a full-rank matrix called generator matrix.

$[ \text{--- } x \text{ ---} ]_{1 \times k}$   
(row-vector)



(taking linear combination of rows of  $G$ )

∴  $C = \text{row-span}(G)$ :  $k$ -dimensional subspace of  $\mathbb{F}_2^n$   
 $\uparrow$   
 codeword

Remark: For linear codes, given  $G$ , encoding is just matrix-vector multiplication, which is efficient.

Q: What about decoding?

In general, finding closest  $y \in \text{Im}(G)$  to a given  $z$  is NP-hard! But for some nice linear codes, this problem is in P.

Notation:  $[n, k, d]_q$ : Linear code from  $\mathbb{F}_q^k$  to  $\mathbb{F}_q^n$  with min distance  $d$ .  
 square brackets to denote linear code.

Q: Given received word  $z$ , before correcting, how to check if it's in a linear code (were there no corruptions)? → Detecting if there is error or not.  
 This is characterized by vectors "orthogonal" to  $C$ .

Defn: For a  $k$ -dimensional subspace code  $C \subseteq \mathbb{F}_q^n$ ,  
 define  $C^\perp = \{w \in \mathbb{F}_q^n : w \cdot y = \sum_{i=1}^n w_i y_i = 0 \ \forall y \in C\}$   
 dot product or inner product

(Example: if you take  $V = \mathbb{R}^3$ , &  $S = \{e_3 = (0, 0, 1)\}$   
 then  $S^\perp$ :  $xy$ -plane)

Exercise: ① Show  $C^\perp$  is also a subspace of  $\mathbb{F}_q^n$  of dimension  $n-k$ .

② Show that  $(C^\perp)^\perp = C$

$C^\perp$  is an  $[n, n-k]_q$ -code called the "dual-code" with encoding:

$$\text{Enc}^\perp: \mathbb{F}_q^{n-k} \rightarrow \mathbb{F}_q^n$$

$$\text{Enc}^\perp(x) = xH \quad \begin{array}{l} (n-k) \times n \text{-matrix} \\ \hookrightarrow \text{called parity-check} \\ \text{matrix for } C \end{array}$$

Such that

$$\begin{array}{c} \updownarrow n-k \\ \boxed{\begin{array}{c} \text{---} \\ \text{---} H \text{---} \\ \text{---} \\ \text{---} \end{array}} \quad \begin{array}{c} \left[ \begin{array}{c} | \\ | \\ z \\ | \end{array} \right]_{n \times 1} = \left[ \begin{array}{c} 0 \end{array} \right]_{(n-k) \times 1} \Leftrightarrow z \in C \end{array}$$

$\leftarrow n \rightarrow$

$$C^\perp = \text{row-span}(H)$$

Why called parity-check?  
 $q=2$ , this is checking if for each codeword, #1's in it is even, i.e. parity is 0.

Now to check if a given string  $z$  has error or not,  
i.e. if  $z \in C$ , all we do now is check if  $H z = 0$ .

↪ (Error detection is easy in linear codes.) ↪

Linear Code Facts: (We will discuss this next class)

① Min-distance  $d(C)$  = minimum Hamming weight of a non-zero codeword. ↪ # non-zero entries.

Pf: Let  $y \neq y', y, y' \in C$   
then  $\Delta(y, y') = \text{HammingWeight}(y - y')$  (by linearity)  
hamming weight is distance from 0 notation for Hamming weight  
 $\therefore d(C) = \min_{y \neq y' \in C} \{\Delta(y, y')\} = \min_{y \neq 0 \in C} \{wt(y)\}$

②  $d(C) = \min \# \text{ cols of } H \text{ which are linearly dependent.}$

Pf:  $d(C) = \min \{wt(y) : y \in C, y \neq 0\}$

but  $y \in C \Leftrightarrow Hy = 0$

$\therefore d(C) = \min \{wt(y) : Hy = 0, y \neq 0\}$

So a min wt  $y$  means you find the sparsest linear combination of cols of  $H$  adding to 0.

this denotes linear combination of columns corresponding to non-zero entries in  $y$ .

Exercise: Can you come up with the parity check matrix corresponding to parity code & Repetition code discussed in class? (First show they are linear codes)

Next class

- Some examples of linear codes
  - ↳ Hamming code
  - ↳ Reed-Solomon codes.
- What are asymptotically "good" codes?

Was discussed on board ↓

Parity code

$$\Sigma = \{0, 1\}^4, \quad k=4, \quad n=5$$

$$\text{Enc}_{\text{PC}}(x_1, x_2, x_3, x_4) = (x_1, x_2, x_3, x_4, x_1 \oplus x_2 \oplus x_3 \oplus x_4)$$

$$\underline{\text{Rate}} = \frac{4}{5}$$

$$\underline{\text{Min Distance}} = 2$$

(cannot correct)  
(can detect 1 error)  
↓  
How?

Repetition code

$$\Sigma = \{0, 1\}^4, \quad k=4, \quad n=12$$

$$\text{Enc}_{\text{RC}}(x_1, x_2, x_3, x_4) = (x_1, x_1, x_1, x_2, x_2, x_2, x_3, x_3, x_3, x_4, x_4, x_4)$$

$$\underline{\text{Rate}} = \frac{4}{12} = \frac{1}{3}$$

$$\underline{\text{Min Distance}} = 3$$

(can correct 1 error)  
↓  
How many  
can you detect?

What would be decoding strategy?