Lecture 10 - Error Correcting Codes

Goal of ECCs: add small, clever amount of "redundancy" to data so that when stored/transmitted, if some of the bits/bytes corrupted, can still recover the original msg. Info thy/ECE people focus on random errors, TCS people (we) focus on worst-case errors.

def: An ECC is an injective map \( \text{Enc} : \Sigma^k \rightarrow \Sigma^n \).
- \( \Sigma \) = alphabet, \( k = |\Sigma|^k \), often 2
- \( \Sigma^k \) = message space
- \( k = \) message dimension/ msg (en)
- \( \Sigma^n \) = (block) length
- \( n \) = (block) length
- \( C = \text{range(Enc)} \subseteq \Sigma^n \) sometimes called the code
- \( \frac{k}{n} = \text{rate} \). Want it large [for efficiencies sake].

Fine, so why not just make \( n = k \)? Want to handle errors.

\[ \Sigma^k \xrightarrow{\text{Enc}} \Sigma^n \]

\[ \text{up to 6 "errors" (changed coords)} \]

\[ \Sigma^n \xrightarrow{\text{recover } x} \Sigma^n \]

def: \( \Delta(y, z) \) = Hamming distance = \# \( i : y_i \neq z_i \).

\[ \Sigma^n \]

\[ \xi \in C \]

Higher rate = more packed in \( C \).
\[ \min_{y, y' \in C} \{ \Delta(y, y') \} \]

**def:** min distance \(d = \min_{x \neq x'} \{ \Delta(\text{Enc}(x), \text{Enc}(x')) \} \)

**fact:** "Unique decoding" possible iff \( t \leq \frac{d}{2} \) \[\text{want large:} \]
\[\text{extension with rate...} \]

**Q:** Why not choose \(C = \mathbb{F}^n\) of size \(q^n\) randomly?

**A:** It will give a great dist/rate tradeoff?

Lacks efficient poly time \(\text{encoding (and decoding)}\)

We want "explicit" codes with good tradeoff.

"All the best codes we know are "linear codes".

**def:** Linear code: \(\text{Enc}: \mathbb{F}_q^k \rightarrow \mathbb{F}_q^n\)

\[x \mapsto Gx\] full rank

\(G = \text{generator mtx}\)

\(C = \text{span Im}(G)\) is \(k\)-dim subspace of \(\mathbb{F}_q^n\)

\(\text{rem: Given } G, \text{ encoding is efficient.} \quad \text{[why, just matrix multiply]}\]

\(\text{Notation: a } [n, k(d)]_q\text{ code.} \quad \text{[if not linear, } - ] \text{ becomes } (\text{.})\]

**Decoding:** sometimes \(\text{NP-hard, but sometimes in } P \text{ for nice codes} \]

\(\text{Checking for corruption is easy...}\]

**Another characterization of a subspace is all the vectors you get to \(f\)

\(C^\perp = \{ w \in \mathbb{F}_q^n : \forall y \in C, y^T w = 0 \} \quad \text{Ex: subspace of dim } n-k\)

Given \(C\) gives rise to \([n, n-k]_q\text{ code}, \text{ Enc } : \mathbb{F}_q^{n-k} \rightarrow \mathbb{F}_q^n\) \(\text{dual code}\)

\(\text{w } \rightarrow \text{ HW}\)
where $H$ is $(n-k) \times n$, the parity check matrix of Enc/C.

$$H = \begin{bmatrix}
H
\end{bmatrix}$$

$C^e = \text{rowspan}(H)$ \quad $\{ \text{all vecs $\perp$ to everything in } C \}$

$z \in C \iff H z = [0]^k$ \quad $\text{"parity checks if } g = 2 \text{ efficient check for } e \in C^e \text{"}$

\text{lin code}

facts: \quad d(C) = \min, \text{ Hamming wt of non-zero codeword.}

\begin{align*}
pf: \quad & d(y, y') = \text{Ham\#wt}(y - y') \\
& \text{a codeword when } y, y' \text{ are !} \\
& 0 \notin y - y'.
\end{align*}

= \min \quad \# \text{ cols of } H \text{ which are linearly dependent.}

pf: \quad \min \quad \# \text{ wt}(z) : z \in C \neq 0

= \min \quad \# \text{ wt}(z) : H z = 0, z \neq 0

(\text{in comb. of } H \text{\'s cols})

As long as $H$ doesn't have all-0 cols, has all 1 cols (indep.

"two ident. cols," $\rightarrow$ 2

And for good rate we want $H$ ... flat as poss.

So why not just take all distinct cols? $

\text{def: Hamming Code: defd by } H = \begin{bmatrix}
0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix} \quad \{ \text{all nonzero strings in } [x_1 ... x_n] \}$

So it's a $[2^k - 1, 2^k - k - 1, 3]_2$ code, 2^k-1

$\text{full rank $\implies$ error correcting codes.}$

"Perfect code" $\text{the red-1 } H \text{ cols around each codeword protect...}
Hadamard code: \( x \mapsto (x^T a) \in \mathbb{F}_2^n \)

**Hadamard code:** For \( x \in \mathbb{F}_2^n \), define \( L_x : \mathbb{F}_2 \to \mathbb{F}_2 \)
\[ a \mapsto x^T a \]

A linear polynomial, \( \sum_i x_i a_i \) (deg 1)

**Fact:** \( x \mapsto \) "truth table of \( L_x \)"

**Proof:** Need all \( \mathbb{F}_2^n \) codewords have \( x \neq 0 \) \( \Rightarrow \) \( x \neq 0 \)
\[ \mathbb{P}_{x \in \mathbb{F}_2^n} \left[ L_x(x) \neq 0 \right] \geq \frac{1}{2} \]

schwarz-zippel, or simply \( \sum x_i a_i + \cdots + x_i a_i \cdots + x_i a_i \) (syz suggests)

**Example:** Generalize to \( q^r, r, (1-\frac{1}{q}) q^r \)

Great dist! in fact, any binary code with dist \( \geq (\frac{1}{2} + \frac{1}{2}) n \)

Terrible rate!!
For $1 \leq k < n$, $q^n \geq n$, $S \subseteq \mathbb{F}_q$ with $|S| = n$...

\[ \text{Enc} : \mathbb{F}_q^k \rightarrow \mathbb{F}_q^n \]
\[ m \rightarrow (P_m(x))_{x \in S} \quad \text{where} \quad P_m(x) \in \mathbb{F}_q[x] \]
\[ \text{Deg} \leq k-1 \]

Rem.: usually $q = n$, $S = \mathbb{F}_q$, $m \rightarrow \text{"truth table" of } P_m$.
- indeed linear: \[ \text{sum of two codewords is codeword?} \]
- gen. mtx. is "Vandermonde" \[ \text{rows are } \{x^0, x^1, \ldots, x^{k-1}\} \quad \text{as } x \text{ varies in } S \]

\[ \text{distance is at least } n - (k-1) \]

Pf: min. Ham. wt. of nonzero codeword
\[ n = \text{max } \# \text{ Os of nonzero codeword} \rightarrow \text{Deg. Min.} \]

\[ [n, k, n-k+1]_q \quad \text{(assuming } q \geq n) \]

\[ \text{optimal} \quad \text{[prove: Min dist. no bigger than } n-(k-1)] \]

Singleton Bound: For $(n, k, d)$ code \[ \text{not nec. linear} \quad k \leq n - d + 1 \]

Pf: $|\text{FCC} : C| > q^{n-d+1}$. \[ \text{FCC} = \text{Families of C with some first } n-d+1 \text{ codes.} \]
\[ \text{Then } \Delta(y, y') \leq d-1 < d, \quad \exists \]

eq: $k = \frac{n}{2}$, rate $\frac{1}{2}$, (rel.) min dist. $= \frac{1}{2}$ ☺ But $q \geq n$ ☹
"Asymptotically good codes." Fix $q$, let $C = (C_n)$ be a seq. of $[n, k(n), d(n)]_q$ codes.

e.g. Hamming: $[n, n - \log(n), 3]_2$, $n = 2^n - 1$

Over a $q$-ary field $\mathbb{F}_q$: $[n, k, d]_q$, $n = 2^n - 1$

Asymptotic rate, $R(C) = \lim_{n \to \infty} \frac{k(n)}{n}$

Relative distance, $S(C) = \lim_{n \to \infty} \frac{d(n)}{n}$

\begin{align*}
R (\text{Ham}) &= 1, \ S (\text{Ham}) = 0 \\
R (\text{Hd}) &= \frac{1}{2}, \ S (\text{Hd}) = \frac{1}{2} \\
R (R.S.) &= \frac{1}{4}, \ S (R.S.) = 1 - \frac{1}{4} \\
\end{align*}

def. "Asympt. Good Code fam." $R(C) = R_0 > 0, \ S(C) > S_0 > 0$

Fact: They exist.

Gilbert-Varshamov Bd: $A_q^n, \ \forall S \subset \{0, 1 \cdot \frac{1}{2} \}, \ |C| \geq \binom{V_{R_0}}{S}$

with $S(C) = S, \ R(C) \geq 1 - R_0(S) > 0$

Any ensemble of $2^{nS_0}$ $[n, k, d]_q$ codes gives a code for

\begin{align*}
\log_q \left( \frac{1}{2} \left( 1 - \log_q \left( \frac{1}{2} \right) \right) \right) \geq nS_0 \\
\end{align*}

Rem: for $q = 4$, $2^n \approx 49$ explicit codes BEATING GV based on alg. geom, !!!!!

Hint [Justesen '78]: $3$-poly$(n)$-time constructible asympt. good codes

\begin{align*}
&\forall C, R < 1 \\
&\text{Also decodable from } \left\lfloor \frac{S_0}{2} \right\rfloor \text{ errors } (S_0 \leq \frac{1}{2} (1 - R_0)) \\
&\text{in } \text{poly}(n) \text{ time with small binary code} \\
&\text{Rem: [Forney '66] "Concatenate" Reed-Solomon with small binary code} \\
&\text{Try brute force/ensemble}
\end{align*}
Concat: \([\text{outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer, outer,outer, outer,