1 Section Week 1 - Error Correcting Codes

Based roughly on Ryan O'Donnell's lecture for TCS Toolkit at CMU

An ECC is an injective map $E : \Sigma^k \to \Sigma^n$ where Σ is the alphabet, $q = |\Sigma|$, k is message length or sometimes dimension, n is block length, $C = img(E) \subset \Sigma^n$ is the code, any $y \in C$ is a codeword, $\frac{k}{n} \in (0, 1]$ is the rate. For all intents and purposes, we can think of the alphabet Σ as $\{0, 1\}$.

Noise model - individual characters can be corrupted to any other character in the alphabet, but we don't know which ones. In particular we're not thinking about insertions and deletions or probabilistic noise right now (maybe in a later 121.5 lecture).

Use diagram to explain intuition for $k < n$; what happens if $k = n$ and we receive a codeword and are told that there can be 1 error?

When we have arbitrary replacement of characters, we thus care about $\Delta(y, z)$ which is Hamming distance (number of different coordinates between vectors y and z). The Hamming weight of a vector y is the number of its nonzero entries, or $\Delta(y, 0)$, where 0 is the all zero vector.

Min distance d for a code C is defined as $\min_{y \neq y'} \Delta(y, y')$ for $y, y' \in C$.

Draw diagram for intuition of how decoding works - finding the nearest codeword. Each string has to be close to AT MOST ONE codeword (we call the code perfect if this is exactly one). Thus, for a model where up to t characters can be arbitrarily corrupted, we can decode iff $t \leq \lfloor \frac{d-1}{2} \rfloor$.

Packing intuition for the diagram and how it relates to the trade off between rate and decoding distance.

How do we achieve this? Ideas? How do you find nearest codeword (NP hard in general)? Random selection?

1.1 Linear ECCs

Use matrices to space codewords in a regular lattice.

- $\Sigma = \mathbb{F}_q$ for some prime q (Again, for the rest of this lecture we can think of $q = 2$ and $\mathbb{F}_2$ is equivalent to the bits $\{0, 1\}$)
- $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$ is a linear transformation $x \mapsto xG$ where here G is a $k \times n$ full-rank generator matrix and x is a row vector
- C is the k-dimensional subspace of $\mathbb{F}_q^n$ defined by G Aside - lattices and additive group
- Given G, encoding is efficient, but for decoding to be efficient (in P) we need to pick specific nice G for which it's easy to find $y \in C$ closest to received z

We write $[n, k, d]_q$ to mean a code $\mathbb{F}_q^k \to \mathbb{F}_q^n$ with min distance d.

To test if a word is a codeword we just need to check that it's in C, and thus orthogonal to all the vectors which are orthogonal to C. This motivates the definition of the dual $C^\perp$, or

$$C^\perp = \{w \in F_q^n | \, \langle w, y \rangle = 0, \, \forall y \in C\}$$

This dual $C^\perp$ is an $n - k$ dimensional subspace generated by some matrix H which is the parity check matrix for the code C generated by G. Prove that for any $y \in C$, $Hy = 0$.

$C^\perp$ is also called the $[n, n-k]_q$ dual code of C and naturally has the property $(C^\perp)^\perp = C$.

Theorem. *For a linear code C, the min distance $d(C)$ is the min hamming weight (number of nonzero entries) of a nonzero codeword.*

Proof. True by the fact that C is a subspace, and $\Delta(y, y')$ is equal to the hamming weight of $y - y'$. $y - y'$ is also a code word. □

Theorem. *It is also true that $d(C)$ is the minimum number of linearly dependent columns of H*

Proof. We know distance of code is min hamming weight of any $z \in C$, which is also the min hamming weight of any z s.t. $Hz = 0$. For nonzero z that satisfies this property, its nonzero entries correspond to some columns in H that, when taken as a linear combination by the nonzero entries of z, equals zero. Thus, those columns are linearly dependent, and the minimum hamming weight of any z corresponds to the smallest set of columns in H that are linearly dependent. □

1.2 Hamming Code

As an example, let's construct a code with low distance but very high rate. For simplicity, consider $q = 2$. To be able to decode a single error, we need $d = 3$.

Intuition: since our distance is small we want high rate $\frac{k}{n}$, so we want low $\frac{n-k}{n}$, so our parity check matrix should have a lot more columns than rows.

We are restricted in the number of columns by the fact that the min number of linearly dependent columns has to be at least 3, so we can't have two columns that are linearly dependent. So if the length of a column is some $r \triangleq n - k$, how do we get the max number of columns in $\mathbb{F}_2$?

Answer: include every distinct column except for the zero column, getting $n = 2^r - 1$, and we see no two columns are linearly dependent, but there exist groups of 3 columns that are linearly dependent as desired.

Thus, this parity check matrix H defines a $[2^r - 1, 2^r - 1 - r, 3]_2$ code. What is the rate in terms of n? It works out to be $1 - \frac{\log n}{n}$.

How do we decode $y + e_i$? When we multiply we just get $H(y + e_i) = Hy + He_i = He_i$ which is just the ith column of H, so we know e_i and can recover y.

Define perfect code - every codeword is distance t away from exactly (instead of at most) one codeword.

"Perfect codes which correct more than one error are very difficult to find." - Michel Goemans, MIT

"I believe that with a constant number of exceptions (perhaps the Golay codes and the ternary Golay codes)" (Madhu Sudan, Harvard) Hamming codes are the only perfect codes.

Golay code

1.3 Hadamard Code

The dual of the Hamming code is the Hadamard code with amazing distance and terrible rate. When H from Hamming is the generator, we basically for all x output $\langle x, a \rangle$ for every possible length r string a.

Theorem. *The Hadamard code is a* $[2^r, r, 2^{r-1}]_2$, *or* $[n, \log n, \frac{n}{2}]_2$

To look at min distance of the Hadamard code, let's rewrite in a different way: for $x \in \mathbb{F}_2^r$ let $L_x : \mathbb{F}_2^r \to \mathbb{F}_2^r$ be the linear polynomial that maps $a \mapsto x_1 a_1 + \ldots x_r a_r$. Here for fixed x, L_x is a polyomial on a, which will be a column of H. The Hadamard code encodes x as the truth table for L_x, since for each column a of H, $\langle x, a \rangle = L_x(a)$, so multiplying xH is equivalent to evaluating L_x on each of the columns of H.

Proof. Proof of this fact (in particular, the distance parameter)

Suppose we have different messages x, y and polynomials L_x, L_y. Since $x \neq y$, then there must exist some i s.t. $x_i - y_i = 1$. WLOG, let this $i = 1$.

Now consider the difference polynomial

$$L_{x-y} = (x_1 - y_1)a_1 + \cdots + (x_r - y_r)a_k$$

and split it into:

$$L_{x-y} = [(x_1 - y_1)a_1] + [(x_2 - y_2)a_2 \cdots + (x_r - y_r)a_r]$$

We argue that for exactly $\frac{n}{2}$ values of a, this difference polynomial evaluates to 1, implying that for exactly $\frac{n}{2}$ values of a, $\langle x, a\rangle \neq \langle y, a\rangle$ and thus the codewords for x and y differ on exactly $\frac{n}{2}$ coordinates.

In particular, $\#\{a|L_x(a) = L_y(a)\} = \frac{n}{2}$

Suppose we first determine each of $a_2, \ldots a_n$, and define some bit b to be

$$b \triangleq [(x_2 - y_2)a_2 \cdots + (x_r - y_r)a_r]$$

Now we have

$$L_{x-y} = (x_1 - y_1)a_1 + b$$

And since we assumed that $x_1 - y_1 = 1$,

$$L_{x-y} = a_1 + b$$

Now we get that if $a_1 = b$ then this polynomial evaluates to 0 and otherwise it evaluates to 1, so for every possible assignment of $a_2, \ldots a_n$, we get exactly one value of a where $\langle x, a\rangle = \langle y, a\rangle$ and exactly one value of a where $\langle x, a\rangle \neq \langle y, a\rangle$ and we are done.

In particular, for $a = ba_2 \ldots a_n$ we get equality, and for $a = (1 - b)a_2 \ldots a_n$ we get inequality.

□

The Hadamard code can be generalized to larger fields to get a $[q^r, r, (1 - \frac{1}{q})q^r]_q$. It is also optimal in the sense that for any code with $d = n(\frac{1}{2} + \epsilon)$, the number of code words $|C|$ is constant.