ERROR-CORRECTING CODES

Want to transmit message $x$ over noisy channel
Encode message with redundancy so that it can be recovered even if transmission distorted

\[
\begin{align*}
x & \xrightarrow{\text{encode}} E(x) \xrightarrow{\text{channel}} y \xrightarrow{\text{decode}} D(y) \\
\end{align*}
\]

Want:
- $D(y) = x$ even if some noise was introduced
- Encoding and decoding efficient
- Encoding doesn't blow up message too much

Some commonly considered channels:
- Binary symmetric channel:
  Bits $\{0, 1\}$ get flipped with probability $p$
- Binary erasure channel:
  Bits get erased with probability $p$, in which case "?" is received
- Worst case:
  An evil adversary is allowed to flip a fraction $p$ of the bits in some malicious way

Coding theory initiated by two papers

Shannon 1948: "A mathematical theory of communication"
Hamming 1950: "Error-detecting and error-correcting codes"

Both papers in Bell System Technical Journal — Shannon and Hamming were colleagues at Bell Labs
Shannon analyzed exactly what can and cannot be done for binary symmetric channel, binary erasure channel, and many other channels.

Serious drawback: highly non-constructive results.

Hamming focused on constructive results
Explicit (and efficient) encoding and decoding functions + analysis of transmission guarantees

We will focus on worst-case guarantees (perhaps most well-studied channel, at least in TCS?)

**NOTATION AND TERMINOLOGY**

Codes are over ALPHABET \( \Sigma \), always \( g = |\Sigma| \)

**BINARY CODES** \( \Sigma = \{0,1\} \)

Other alphabet will also be of interest

**HAMILTON DISTANCE** \( \Delta(x,y) \quad x,y \in \Sigma^n \)

\[
\Delta(x,y) = | \{ i \in [n] \mid x_i \neq y_i \} |
\]

This is indeed a metric

\[
\Delta(x,y) = \Delta(y,x)
\]

\( \Delta(x,y) \geq 0 \) with equality if \( x = y \)

\[
\Delta(x,z) \leq \Delta(x,y) + \Delta(y,z)
\]

For now, identify code with the image of the encoding function.
Will get back to encoding/decoding later.
ERROR-CORRECTING CODE simply subset

\[ C \subseteq \Sigma^n \quad \text{for all codeword} \quad x \in C \]

Minimum distance of code \( \Delta(C) = \min_{x \neq y \in C} \{ \Delta(x, y) \} \)

Fundamental parameters for \( C \subseteq \Sigma^n \)

1) BLOCK LENGTH \( n \)

2) MESSAGE LENGTH \( k = \log_2 |C| \)
   (There are \( q^k \) possible messages that can be sent; can think of encoding function \( \mathbf{e} : \Sigma^k \rightarrow \Sigma^n \))

3) MINIMUM DISTANCE \( d = \Delta(C) \)
   (will affect possibility of correcting errors)

4) ALPHABET SIZE \( q = |\Sigma| \)

We say that \( C \) is an \((n, k, d)_q\) code

Broad goals for \((n, k, d)_q\) codes

a) Given \( k, d, q \), minimize block length \( n \)

b) Given \( n, d, q \), maximize message length \( k \)

c) Given \( n, k, q \), maximize distance \( d \)

d) Given \( n, k, d \), minimize alphabet size \( q \)

(a) - (c) obvious - small block length, large message size, and large distance always desirable

(d) not so clear. But empirically, the smaller the \( q \), the harder it is to get good parameters.

And in some sense, at least from a CS perspective, it seems \( q = 2 \) would be the really interesting case
In fact, often a good idea to
(i) build great code for large \( q \);
(ii) find clever way of reducing \( q \) without 
destroying other parameters

We will hopefully see some examples of this

Hamming considered 3 properties of a code \( C \)
1) Minimum distance
2) Error detection capacity
   \( C \) is \( e \)-error detecting if, under the promise that no more than \( e \) errors occur 
during transmission, it is always possible 
to detect whether errors have occurred or not. 
Requires minimum distance \( d \geq e + 1 \)
3) Error correction capacity
   \( C \) is \( t \)-error correcting if, under promise 
that no more than \( t \) errors occur, it is 
always possible to detect and correct 
errors (information-theoretically, but 
not necessarily efficiently) and \( t \) is maximal with this property
   \( t \)-error correcting code has min distance 
   \( 2t + 1 \) or \( 2t + 2 \)

We will focus on distance (and error-correcting 
capability)

Later discuss how to solve algorithmic side of 
error correction
Goals for lectures on error-correcting codes

- Cover some basic concepts, terminology, notation, facts
- Build some nice codes using vector spaces and polynomials over finite fields
- Prove some upper bounds on what kind of code constructions are possible
- Define/understand what "good codes" means (codes that get somewhat close to best possible)
- Give increasingly better constructions of codes (polynomials will play a key role)
- See that very good codes can be constructed that are also extremely efficient to encode and decode (expander graphs will play a key role)

Somewhere in the middle will have an interlude with a guest lecturer speaking about other topic (due to calendar constraints - such is life)

Let us start by warming up...
Some simple codes

\[ d = 1 \]
Trivial. Can use identity for encoding which yields \((n, n, 1)_{\mathbb{Z}}\) - code.

\[ d = 2 \]
Already interesting.
Append parity of all the message bits (if messages differ in just one coordinate, parities will also be different.
For general \( q \), identity \( \mathbb{Z}^l = \mathbb{Z}_q \) (additive group of integers mod \( q \)) and use sum of all message symbols as check symbol yields \((n, n-1, 2)_{\mathbb{Z}}\) - code.

\[ d = 3 \]
Non-trivial. Will get back to this.
Perhaps \((n, n-2, 3)_{\mathbb{Z}}\) - code possible?
Or more generally \((n, n+1-d, d)_{\mathbb{Z}}\) - code?

Nope. \( d = 3 \) counter-example, as proven by Hamming.

To discuss this, we need to develop some more terminology and concepts.
Associate $\Sigma$ with field $\mathbb{F}_q$ (so $q$ prime power). Think of codewords as living in vector space $\mathbb{F}_q^n$.

$L \subseteq \mathbb{F}_q^n$ is a \textbf{linear subspace} if
\[ \forall x, y \in L \quad \forall \lambda \in \mathbb{F}_q \]
\[ x + y \in L \]
\[ \lambda x \in L \]

$L$ is a \textbf{linear code} if $L \subseteq \mathbb{F}_q^n$ is a linear subspace of $\mathbb{F}_q^n$.

We denote a code $C$ is linear by using square brackets: $C$ is an $[n, k, d]_q$-code if $C$ is an $(n, k, d)$-code that is linear.

We can use linear algebra to succinctly represent linear codes.

\textbf{Generator Matrix}

Can specify linear subspace by basis linearly independent set of vectors $x_1, \ldots, x_k \in \Sigma^n$ such that
\[ C = \{ \sum_{i=1}^{k} \lambda_i \cdot x_i \mid \lambda_i \in \mathbb{F}_q \} \]

A \textbf{generator matrix} $G \in \mathbb{F}_q^{k \times n}$ for $C$ is a matrix with the rows of $G$ forming a basis of $C$.

\[ C = \{ x \cdot G \mid x \in \mathbb{F}_q^k \} \]
The null space of a linear subspace $L \subseteq F_q^n$ is $L^\perp = \{ y \in F_q^n \mid \langle x, y \rangle = 0 \ \forall x \in L \}$

**Fact 1:** The null space $L^\perp$ of a linear subspace $L \subseteq F_q^n$ is a linear subspace of $F_q^n$ of dimension $n - \dim(L)$

This might seem geometrically obvious. If $L \subseteq \mathbb{R}^3$ has dimension 2, then $L$ plane and $L^\perp$ orthogonal line

But $\langle \cdot, \cdot \rangle$ is not a true inner product.

**Example:** Suppose $q = 2$, $n = 2$

$$C = \{(0,0), (1,1)\}$$

Then $C^\perp = C$

$C^\perp$ is also a linear code; the dual code.

Let $H^\perp \in F_q^{(n-k) \times n}$ be a generator matrix for $C^\perp$. Let $H = (H^\perp)^T$.

Then $H \in F_q^{n \times (n-k)}$ is a PARITY CHECK MATRIX of $C$. Yes, succumbing to the notation in Madhu Sudan’s ECC lecture notes, we will now often write vectors as rows.
\text{FACT 2} \quad C = \{ y \in \mathbb{F}_q^n \mid y^T H = 0 \}^\perp

This is in some sense obvious, but here is an explanation why it is obvious.

Columns of $H$ are basis for $C^\perp$.

Hence, if $y^T H = 0$, then $y \in (C^\perp)^\perp$.

Clearly, all $x \in C$ satisfy this, so $C \subseteq (C^\perp)^\perp$.

But $\dim(C) = \dim((C^\perp)^\perp)$ by Fact 1, so $C = (C^\perp)^\perp$.

Generator matrix $\{ \}$
Parity check matrix $\{ \}$

Both $O(n^2 \log q)$-size representation of $C$ can be shown that representations are essentially equivalent computationally.

Can compute one from the other using linear algebra.

So why have both? Generator matrix would seem to be sufficient... useful since they!

But parity check matrices can show properties of a code.
Proposition 3

Error detection for linear codes can be done efficiently using the parity check matrix.

Proof

Suppose received word \( y \) is transmitted from a codeword in \( C \). Decode it and guarantee that fewer than \( d - 1 \) errors occurred in transmission.

Compute \( yH \).

- If \( yH = 0 \), then \( y \in C \) and no error occurred.
- If \( yH \neq 0 \), then \( y \) is not a codeword and an error must have occurred.

Problem

Given linear code \( C \), compute its minimum distance \( \Delta(C) \).

NP-hard problem...

But using parity check matrices can help us design codes with reasonable minimum distance.

The Hamming weight of \( x \in \mathbb{F}_2^n \) is given by:

\[ \text{wt}(x) = \left| \{ i \mid x_i \neq 0 \} \right| \]

i.e.

\[ \text{wt}(x) = \Delta(x, 0) \]
To find min distance of linear code, find non-zero codeword $x$ of smallest weight such that $x \cdot H = 0$. Then $wt(x) = \Delta(c)$.

**Proposition 7**

Let $C$ linear code with parity check matrix $H$. Then $\Delta(c)$ equals smallest $d$ such that there exist $d$ linearly dependent rows in $H$.

**Proof** Note that this is equivalent to the claim

$$\Delta(c) = \min_{x \neq 0} \{ wt(x) \mid x \cdot H = 0 \}$$

Let $h_i$ $i$th row of $H$

Let $x_{i_1}, \ldots, x_{i_d}$ non-zero coordinates of $x$

Then $\sum_{j=1}^{d} x_{i_j} \cdot h_{i_j} = 0$ and hence

$\{ h_{i_1}, \ldots, h_{i_d} \}$ is a linearly dependent set of rows. In the other direction, any collection of $d$ linearly dependent rows by definition yield $x$ of weight $d$ s.t. $x \cdot H = 0$.

So we need to show

$$\min \{ wt(x) \mid x \neq 0, x \cdot H = 0 \} = \Delta(c)$$

Clearly $\Delta(c) \leq wt(x)$ since $0$ is a codeword (by linearity) and

$$\Delta(x, 0) = wt(x)$$

Suppose $y \neq z \in C$ minimize $\Delta(y, z)$

Then $y-z \in C$ by linearity and

$$\Delta(c) = \Delta(y, z) = \Delta(y-z, 0) = wt(y-z)$$
Now we can construct codes with minimum distance $d = 3$

For simplicity, let's focus on $\mathbb{F}_2 = \{0, 1\}$

Want to construct matrix $H \in \mathbb{F}_2^{n \times (n-k)}$

such that if $\text{wt}(x) = 1$ or $2$ then $xH \neq 0$

Fix $c = n-k$ and try to maximize $c$

Let $h_i = i$th row of parity check matrix

If $\text{wt}(x) = 1$ then $x = e_i$ and $xH = h_i$

Hence if all rows in parity check matrix are non-zero then min dist of code $\geq 2$

If $\text{wt}(x) = 2$ then $x = e_i + e_j$ for some $i \neq j$.

and $xH = h_i + h_j$. We thus need $h_i + h_j$ for all $i \neq j$.

Conclusion: $H$ is parity check matrix

for code of distance $\geq 3$ if all rows distinct and non-zero

largest $n = \#\text{ distinct non-zero vectors in } \mathbb{F}_2^c$

there are $2^c - 1$ such vectors. This proves the following proposition

PROPOSITION 5: For every $c \in \mathbb{N}^+$, $c \geq 2$ there exists an $[2^c - 1, 2^c - c - 1, 3]_2$-code called the (binary) Hamming code

Can let $H$ be indices $1, 2, \ldots, 2^c - 2, 2^c - 1$

written in binary.
If \( k = 2^l - 1 \), then \( l = \Theta(\log k) \)

thus, Hamming codes are obtained by adding \( \Theta(\log k) \) parity check bits.

Min distance \( 3 \implies \) error correction capacity 1.

Given received word with one error how can we locate position of error?

1) brute force

Flip each bit in \( y \) until find \( y' = y + e \) with \( y'H = 0 \). Then error is in position \( i \).

\( O(n^3) \) algorithm

2) use parity check matrix in a smarter way.

One error in position \( i \) \( \implies \)

\( y = x + e \);

Since \( x \in C \) we have

\( y'H = (x + e_i)H = e_i; H = h_i \);

But \( h_i \) is index i written in binary

\( O(n^2) \) algorithm

Is it necessary to add logarithmically many bits to be able to (not only detect but) correct a single error?

Hamming proved that the answer is \textbf{yes}!
Let \( B_q(y, r) = \{ x \mid \Delta(x, y) = r \} \) denote the HAMMING ball of radius \( r \) centered at \( y \) in \( \mathbb{F}_q^n \).

Let \( \text{Vol}_q(r, n) \) denote the volume of any radius-\( r \) ball, i.e., in \( \mathbb{F}_q^n \), i.e.,

\[
\text{Vol}_q(r, n) = |B_q(y, r)| = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.
\]

For binary codes (i.e., \( \mathbb{F}_2^n \) for linear codes), we drop subscript.

**Theorem 6 (HAMMING Bound):**

If an \((n, k, d)_q\) code exists, then

\[ q^k \cdot \text{Vol}_q\left(\left\lfloor \frac{d-1}{2} \right\rfloor, n\right) \leq q^n \]

**Proof:** If \( x, y \) codewords of \((n, k, d)_q\) code and if we let \( r = \left\lfloor \frac{d-1}{2} \right\rfloor \), then \( B(x, r) \) and \( B(y, r) \) are disjoint.

Since \( \bigcup_{x \in C} B(x, r) \subseteq \mathbb{F}_q^n \) and sets on the are all disjoint we get

\[
\sum_{x \in C} B(x, r) = q^k \cdot \text{Vol}_q(r) \leq q^n
\]
For $d = 3$, $q = 2$ we have
\[ \text{Vol}(1,n) = 1 + n \]
and Thm 4 yields
\[ 2^k(n+1) = 2^n \]
For the Hamming code in Prop 5 we have
\[ n = 2^t - 1 \]
\[ k = 2^t - t - 1 \]
blocklength
message length
and
\[ 2^k(n+1) = 2^t - t - 1, \quad 2^t = 2^2 - 1 = 2^n \]
so Hamming codes meet Hamming bound exactly. (Can define $q$-ary Hamming codes and this is true for them as well.)
Thus, balls around codewords in Hamming code fill up ambient vector space $F_q^n$ perfectly.

DEF 7
An $(n,k,d)_q$-code is PERFECT if it meets the Hamming bound exactly, i.e., if
\[ q^k \text{Vol}_q\left(\frac{d-1}{2}, n\right) = q^n \]

Hamming codes are perfect.
Two specific codes by Golay [1949] are perfect.
No other perfect codes exist!
[van Lint 1971], [Tietavainen 1973]
So Hamming bound rules out $(n, n-d+1, d)_2$-code already for $d=3$. (Note this applies to binary codes $q=2$)

Can we get $(n, n-d+1, d)_q$-codes for $q$ large enough?

Why not be even more ambitious and ask for $k = n-d+1$?!

Because, as we will see next lecture, $k \leq n-d+1$ is an upper bound regardless of alphabet size.

Will also see next lecture that there are codes that meet this bound.

So-called REED SOLOMON CODES constructed by evaluating univariate polynomials over finite fields.