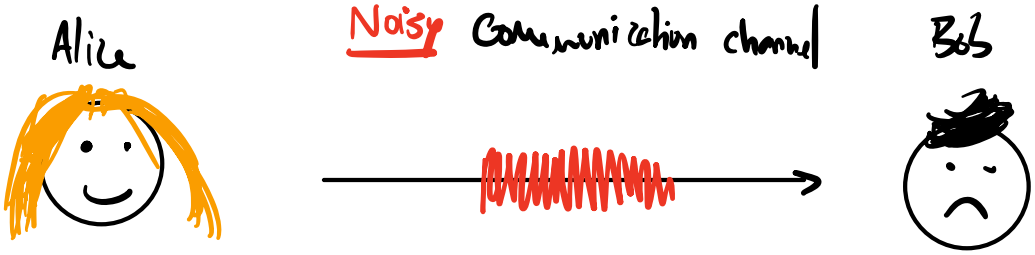
Introduction to Coding Theory

Recommended reading:

Essential Coding Theory (ECT)

Chapter 1

THE problem



"8pm dinner at Top Dog"

"7am dinner at Top Doo"

even one tiny error can kill a love story!

Noise is everywhere:

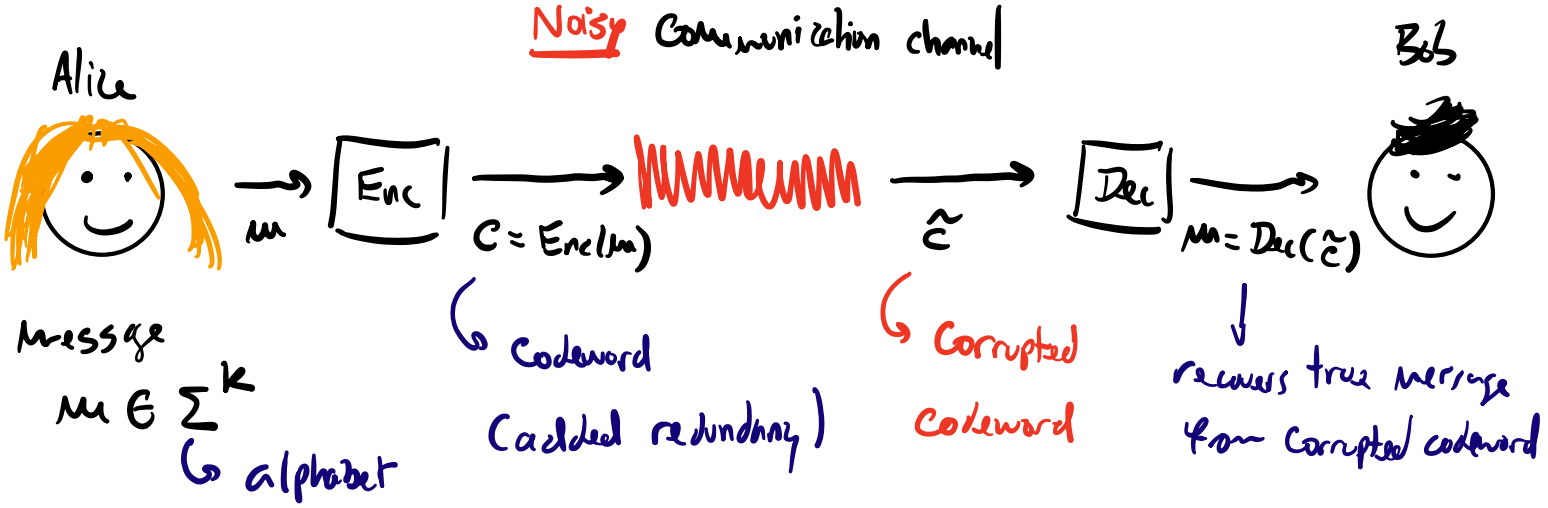
→ Communication (phone, streaming, deep space communication)

→ data storage (technically also a type of communication)

→ Computation

How can we help Alice and Bob communicate without errors?

ERROR-CORRECTING CODES



Codes are studied in coding theory and information theory (the distinction between these two subjects is not clear-cut)

Beautiful combination of algebra, combinatorics, geometry, probability + computer science + electrical engineering

Kickstarted by Claude Shannon (1948) and Richard Hamming (1950)

https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf
https://ieeexplore.ieee.org/document/6772729/

To say something meaningful we need to fix an error model.

t-error channel: Given an input x, the channel replaces up to t coordinates of x by other symbols
↳ the receiver does not know locations of errors!

We would like to design a code that allows us to recover the underlying message even when t errors occur.

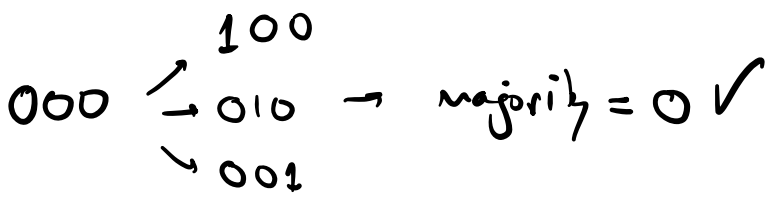
Let's look at a simple setting:

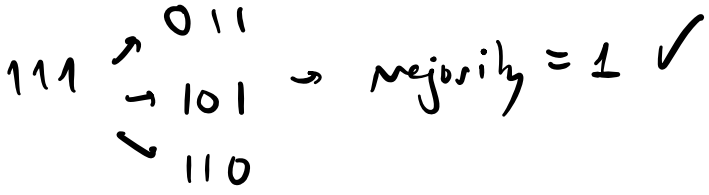
- $t = 1$
- binary alphabet $\Sigma = \{0,1\}$

Here's a naive way of encoding a message $x \in \{0,1\}^k$ so that we can correct 1 error: Repeat each bit of x 3 times!

$$x = 10010 \rightarrow C = Enc(x) = 111000000111000$$

How can we recover x from $\tilde{c}$ (obtained by applying 1 error to c)?
Take the majority bit in every consecutive block of 3 bits





This is a "repetition code". It works, but it's wasteful!
The codeword is 3 times as large as the message...
In practice, we want our codewords to be short, not much larger than the message (resources are limited!).

Can we do better? Before we think about this, let's define some basic concepts.

Def (Code): A code with blocklength n over an alphabet Σ is a subset $C \subseteq \Sigma^n$.

"Encoding" vs. "set" view of codes: We can also see a code $C \subseteq \Sigma^n$ of size M as an injective encoding function $Enc: \{1,\dots,M\} \to \Sigma^n$. We will often jump between these two (equivalent) perspectives on codes.

Above we mentioned wanting codewords to be "short". The "rate" of a code nicely captures this.

Def (Code rate): The rate of a code $C \subseteq \Sigma^n$ is

$$\frac{\log |C|}{\log |\Sigma|^n} = \frac{\log |C|}{n \log |\Sigma|}.$$

In other words, a code $C \subseteq \Sigma^n$ has rate R if $|C| = |\Sigma|^{Rn}$.

The rate of the binary repetition code above is $1/3$. You may think of this as saying that we need to send 3 bits over the noisy channel to communicate 1 bit of information.

Fundamental Question 1: Given a noisy channel and a block length n, what is the rate of the largest code that allows for reliable communication over this channel? We care mostly about the limit $n \to \infty$.

Note: It makes perfect sense to ask this question about noisy channels that behave probabilistically. For example, what about the channel that flips each input bit from 0 to 1 or 1 to 0 independently with probability p? This is called the binary symmetric channel (BSC). However, we must settle for vanishing error probability.

In this course we will mostly focus on "combinatorial" channels, where we wish to correct <u>any</u> possible error pattern. But we will work out the answer to this question for the BSC.

Let's get back to our t-error channel. Can we do better than the repetition code? Let's formalize things

Def (t-error-correcting code):

A code $C \subseteq \Sigma^n$ is t-error-correcting if for any distinct $c, c' \in C$ there is no $z \in \Sigma^n$ that can be obtained from both c and c' by applying up to t errors.

This may seem complicated, but there is a nice geometric interpretation of this stuff.

Def (Hamming distance): The Hamming distance between $x, y \in \Sigma^n$ is $d_H(x,y) = |\{i : x_i \neq y_i\}|$.

Exercise: Prove that the Hamming distance is a metric.

It turns out that the error-correction properties of a code $C \subseteq \Sigma^n$ are controlled by the Hamming distance between any two codewords.

Def (minimum Hamming distance): A code $C \subseteq \Sigma^n$ has minimum (Hamming) distance d if

$$\min_{\substack{c,c' \in C: \\ c \neq c'}} d_H(c,c') = d.$$

Thm: A code $C \subseteq \Sigma^n$ is t-error-correcting if and only if $t \leq \lfloor \frac{d-1}{2} \rfloor$, where d is the minimum distance of C.

↳ floor of $\frac{d-1}{2}$

Proof:

($\Leftarrow$) Suppose that there is $z \in \Sigma^n$ that can be obtained from both $c \in C$ and $c' \in C$ via $\leq t$ errors. Then $d_H(c,z) \leq t$ and $d_H(c',z) \leq t$. By the triangle inequality,

$$d_H(c,c') \leq d_H(c,z) + d_H(z,c') \leq t + t < d.$$

This contradicts the minimum distance of C.

($\Rightarrow$) Let $c, c' \in C$ be such that $d_H(c,c') = d$, and suppose that $t \geq \lfloor \frac{d-1}{2} \rfloor + 1 \geq d/2$.

Let $S = \{ i : c_i \neq c'_i \}$. Split S into two disjoint sets S_1, S_2 of size t and $d-t$, respectively. Note that $|S_1|, |S_2| \leq t$.

Consider z obtained by setting, for each $i \in \{1, \dots, n\}$,

$$z_i = \begin{cases} c'_i, & \text{if } i \in S_2. \\ c_i, & \text{if } i \notin S_2. \end{cases}$$

Then, z is obtained from c by applying $|S_2| \leq t$ errors at coordinates $i \in S_2$.

Likewise, z is obtained from c' by applying $|S_1| \leq t$ errors at coordinates $i \in S_1$.

This means that C is not t-error-correcting. ▨

It's a good exercise to show that the 3-repetition code has minimum distance $d = 3$, and so can correct 1 error, but can't correct 2 errors.

Towards finding better error-correcting codes, let's first focus on a simpler problem: error-detection. In this setting, the goal is only to detect that errors occurred (ie, the corrupted output is not a codeword), but we don't need to recover the original codeword.

Def (t-error-detecting code): A code $C \subseteq \Sigma^n$ is said to be t-error-detecting if for any codeword $c \in C$ and $z \in \Sigma^n$ obtained from c via at most t errors we have either $z = c$ or $z \notin C$.

As with error-correction, we can characterize error-detection in terms of the minimum distance.

Thm: A code $C \subseteq \Sigma^n$ is t-error-detecting if and only if $t \leq d-1$, with d the minimum distance of C.

Proof: Homework.

Can we find a better 1-error-detecting code than the 3-repetition code?

A first obvious improvement is to only repeat each symbol twice. But we can do better.

Let's focus on $\Sigma = \{0,1\}$. Consider the "parity code"

$$C_{par} = \{ c \in \{0,1\}^n : \sum_{i=1}^{n} c_i = 0 \bmod 2 \}$$

Note that $|C_{par}| = 2^{n-1}$. It's also not hard to compute the minimum distance of C_{par}.

Thm: C_{par} has minimum distance 2. Therefore, it detects 1 error.

The parity code is much better than the 2-repetition code. It has only 1 redundant bit, while the 2-repetition code has $n/2$ redundant bits. → redundancy $= n - \log|C|$

↳ in this course, log is base-2 logarithm.

↓ Equivalently

rate $= 1 - 1/n$

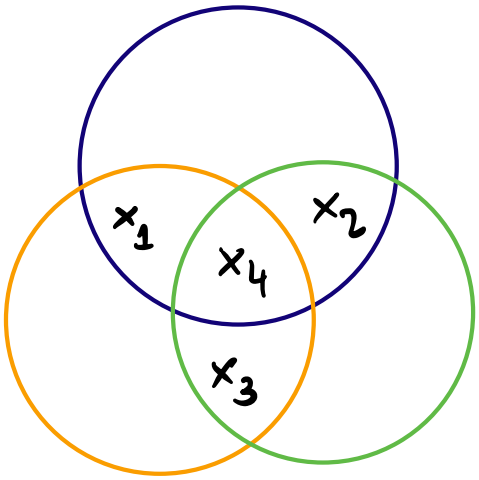
Now, let's go back to error-correction and try to find a large code correcting 1 error. Equivalently, this means finding a large code with minimum distance 3.

Adding a parity bit worked great for detecting 1 error, so maybe we can correct 1 error by adding some more (carefully chosen) parity bits.

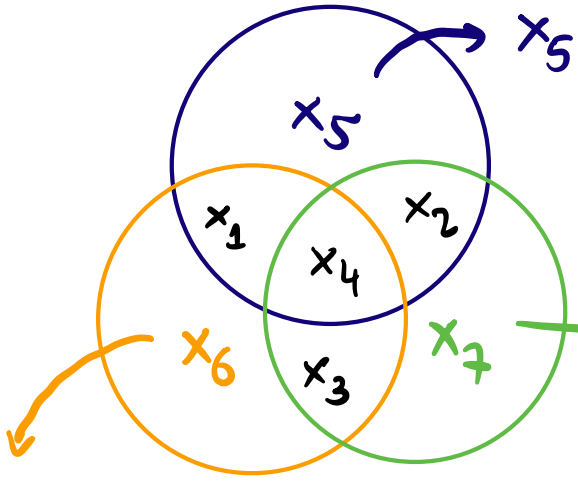
Starting off with concrete examples is always a good idea. Here's something that works if we are trying to encode messages of length 4:

Suppose that $(x_1, x_2, x_3, x_4) \in \{0,1\}^4$ is the message. We will encode this message as follows.

First, we draw some mysterious circles and place the message bits in the intersections.



For each circle we add the parity bit corresponding to the message bits contained in that circle.



$x_5 = x_1 + x_2 + x_4 \mod 2$

$x_7 = x_2 + x_3 + x_4 \mod 2$

$x_6 = x_1 + x_3 + x_4 \mod 2$

In other words, bits inside each circle sum to 0 mod 2

Then, the encoding of (x_1, x_2, x_3, x_4) is

$$(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \in \{0,1\}^7$$

This is called the Hamming code. It has block length $n=7$ and size $2^4 = 16$. Its rate is $4/7$, and it has $7-4=3$ bits of redundancy.

Thm: The Hamming code has minimum distance 3. Therefore, it corrects 1 error.

You can prove this theorem by case analysis, although that's not very satisfying. We will see a nicer, more systematic way of establishing this later.

So, we have a code correcting 1 error that is better than the 3-repetition code (which would require block length 12 to encode 4-bit messages). Can we do better than the Hamming code?

To study this question we introduce the following useful notion.

Def (Hamming ball): The radius-r Hamming ball centered at $y \in \Sigma^n$ is $B_r(y) = \{z \in \Sigma^n : d_H(y,z) \leq r\}$.

Note that a radius-r Hamming ball $B_r(y)$ has size

$$|B_r(y)| = \sum_{i=0}^{r} \binom{n}{i} (|\Sigma|-1)^i .$$

When $\Sigma = \{0,1\}$, we set $|B_r(y)| = |B_r(0^n)| = \sum_{i=0}^{r} \binom{n}{i}$.

Thm: Every 1-error-correcting code $C \subseteq \{0,1\}^n$ satisfies

$$|C| \leq \frac{2^n}{n+1}.$$

This is called the sphere-packing bound. The reason for this will become clear in the proof.

Proof: The statement follows from the insight that a 1-error-correcting code $C \subseteq \{0,1\}^n$ allows us to pack $|C|$ disjoint radius-1 Hamming balls inside $\{0,1\}^n$.

Indeed, note that if C is 1-error-correcting, then for any two distinct codewords $c, c' \in C$ we have

$$B_1(c) \cap B_1(c') = \emptyset .$$

Otherwise, if $z \in B_1(c) \cap B_1(c')$ then

$d_H(c,c') \leq d_H(c,z) + d_H(z,c') \leq 1+1 = 2$,

but we know that C has minimum distance ≥ 3.

Therefore, we have

$$\sum_{c \in C} |B_r(c)| \leq |\{0,1\}^n| = 2^n.$$

↳ $\sum_{c \in C} \left(\binom{n}{0} + \binom{n}{1} \right) = |C| \cdot (n+1)$

this implies that $C \leq \frac{2^n}{n+1}$. ∎

If we take $n = 7$, we see that any 1-error-correcting code $C \subseteq \{0,1\}^7$ has size $|C| \leq 2^7/8 = 16$.

⇒ The Hamming code is optimal! Cool!

But this discussion leaves a lot to be desired...

Our construction of the Hamming code was very ad hoc. Can we develop a more systematic way of designing and analyzing good codes?

→ How to generalize the Hamming code to $n > 7$?

→ Nicer ways of determining minimum distance?

→ How to correct more than 1 error with high rate?

→ How to "efficiently" encode messages and correct errors?

Towards a more systematic study of codes

The codes we have seen so far have a curious property. They are "linear". A code $C \subseteq \{0,1\}^n$ is <u>linear</u> if it is closed under addition mod 2. That is, for any $c, c' \in C$

$$c + c' = (c_1 + c'_1 \bmod 2, c_2 + c'_2 \bmod 2, \ldots, c_n + c'_n \bmod 2) \in C.$$

A subset of $\{0,1\}^n$ with this property is also called a "<u>linear subspace</u>" of $\{0,1\}^n$.

As an example, let's verify this for C_{par}. Let $c, c' \in C_{par}$ and define $c'' = c + c'$. To see that $c'' \in C_{par}$, it suffices to check that

$$c''_n = c_n + c'_n = \sum_{i=1}^{n-1} c_i + \sum_{i=1}^{n-1} c'_i = \sum_{i=1}^{n-1} (c_i + c'_i) = \sum_{i=1}^{n-1} c''_i.$$

Do you remember your linear algebra course? :)

Linear subspaces of $\mathbb{R}^n$ are spanned by a set of linearly independent vectors (a "basis"), and the dimension of the subspace is the size of the basis.

Likewise, linear subspaces of $\{0,1\}^n$ are also spanned by linearly independent vectors. We say vectors $v_1, \ldots, v_k \in \{0,1\}^n$ are linearly independent if

$$\sum_{i=1}^{k} \alpha_i v_i = 0 \iff \alpha_1 = \cdots = \alpha_k = 0,$$

where $\alpha_1, \ldots, \alpha_k \in \{0,1\}$ are scalars.

Let's set up some basic properties of binary linear codes.

Thm: Let $C \subseteq \{0,1\}^n$ be a linear code. Then:

1) $|C| = 2^k$ for some integer $k \geq 0$. (k is called the "dimension" of the code.)
2) C is the span of k linearly independent vectors $v_1, \ldots, v_k \in \{0,1\}^n$.

In other words, there exist linearly independent vectors $v_1, \dots, v_k \in \{0,1\}^n$ such that for

$$G = \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_k \\ | & | & & | \end{bmatrix}$$

we have $C = \{ Gx : x \in \{0,1\}^k \}$.

↳ operations done mod 2

x seen as column vector

Proof:

1) Suppose $2^k < |C| < 2^{k+1}$ for some integer $k \geq 0$.

We will show that C contains $k+1$ linearly indep. vectors $v_1, \dots, v_{k+1}$. The span

$$\left\{ \sum_{i=1}^{k+1} \alpha_i v_i : \alpha_1, \dots, \alpha_{k+1} \in \{0,1\} \right\}$$

has size 2^{k+1} and is contained in C, so $|C| \geq 2^{k+1}$, contradicting the assumption.

We construct the set S of linearly independent vectors as follows:

1st step: Choose any non-zero vector $v_1 \in C$ and set $S = \{v_1\}$

$t \le k+1$

t-th step: Suppose S already contains $t-1$ linearly independent vectors $v_1, \dots, v_{t-1}$. Their span has size 2^{t-1}. Since $2^{t-1} \le 2^k < |C|$, there is $v_t \in C$ not in the span. Add v_t to S.
It's a good exercise to convince yourself that S remains a collection of linearly independent vectors.

2) By (1) we know that $|C| = 2^k$ for some integer $k \ge 0$. The argument above also shows that C contains a set of k linearly independent vectors $v_1, \dots, v_k$. Since their span has size 2^k and is contained in C, we see that

$$C = \left\{ \sum_{i=1}^{k} \alpha_i v_i : \alpha_i \in \{0,1\} \right\} = \{ Gx : x \in \{0,1\}^k \}$$

with $G = \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_k \\ | & | & & | \end{bmatrix}$. ∎

The matrix G is usually called a "generator matrix" of C.

The 3-repetition code has generator matrix

$$\begin{bmatrix} 1 & 0 & & 0 \\ 1 & 0 & & \vdots \\ 1 & 0 & & \\ 0 & 1 & & \\ \vdots & 1 & & \\ \vdots & 1 & \ddots & 0 \\ \vdots & 0 & & 1 \\ 0 & 0 & & 1 \end{bmatrix}.$$

The parity code has generator matrix

$$\begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \\ 1 & 1 & \cdots & 1 \end{bmatrix}.$$

The Hamming code we saw has generator matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

The dual view of binary linear codes

Above, we characterized binary linear codes as the (equivalently, linear subspaces of $\{0,1\}^n$) column span of some (full-rank) $n \times k$ matrix G.

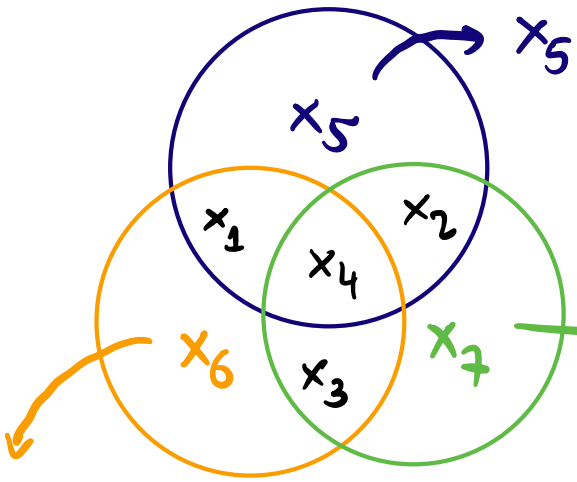
As is often done in linear algebra over $\mathbb{R}$, we may also write a linear subspace as a kernel of some $(n-k) \times n$ matrix H. That is, we can write $C = \ker(H) = \{ y \in \{0,1\}^n : Hy = 0 \bmod 2 \}$.

(0 ↦ the all-0s column vector)

We call H the parity-check matrix of C.

What's the parity-check matrix of the $n=7$ Hamming code?

Recall the circles:



$x_5 = x_1 + x_2 + x_4 \mod 2$

$x_7 = x_2 + x_3 + x_4 \mod 2$

$x_6 = x_1 + x_3 + x_4 \mod 2$

Note that the bits inside a circle must sum to $0 \mod 2$.

Therefore,

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Can you spot something curious?

Let's rearrange the columns of H (of course, this does not affect the properties of the code):

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

The columns of H correspond to all non-zero vectors in $\{0,1\}^3$!

So it's natural to try to generalize the $n=7$ Hamming code as follows:

For block length $n = 2^r - 1$ (r some integer ≥ 1), we take the Hamming code C with block length n to be the code whose parity-check matrix H has as columns all non-zero vectors in $\{0,1\}^r$.

Therefore, H is an $r \times (2^r - 1)$ matrix and C has dimension $2^r - 1 - r = n - \log(n+1)$.

↳ Equivalently, C only has $\log(n+1)$ redundant bits!

I claim that C has minimum distance 3, and so corrects 1 error. By the "sphere-packing bound" above, we know that C would be the largest 1-error-correcting code. But how can we determine the minimum distance?

Towards this, we prove some useful properties of linear codes. First, we need some definitions.

Def (Hamming weight): The Hamming weight of $x \in \{0,1\}^n$ is $w_H(x) = |\{i : x_i \neq 0\}| = d_H(x, 0^n)$.

Thm: Let C be a binary linear code. Then, the minimum distance of C equals its "minimum weight"

$$\min_{c \in C \setminus \{0\}} w_H(c).$$

Proof: Let d be the minimum distance of C.

Note that $w_H(c) = d_H(c, 0)$. Since $0 \in C$ because C is linear, clearly $d \leq w_H(c)$ for all $c \in C \setminus \{0\}$.

In the other direction, let $c, c' \in C$ be such that

$d_H(c,c') = d$. Then,

$$d_H(c,c') = d_H(c+c', 0) = w_H(c+c').$$

we have $c+c' \in C \setminus \{0\}$, so $\min_{c'' \in C \setminus \{0\}} w_H(c'') \leq w_H(c+c') = d$.

The parity-check view of linear codes turns out to be very useful for computing their minimum distance.

Thm: Let $C \subseteq \{0,1\}^n$ be a linear code with parity-check matrix H. Then, C has minimum distance d if and only if:

- Every subset of $d-1$ colums of H is linearly independent.
- There is a subset of d columns of H that is linearly dependent.

Proof: Homework!

Thm: The blocklength n Hamming code has minimum distance 3.

Proof. All columns of H are nonzero and distinct, and so all pairs of columns are linearly independent. On the other hand,

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}.$$

∎

Efficient encoding and error-correction

We have designed optimal 1-error-correcting codes. However, for these codes to be useful in practice they must come with efficient algorithms for encoding and error-correction.

Fundamental Problem 2: Device large codes with good error-correction properties and efficient encoding and error-correction algorithms

For now we won't be precise about what "efficient" means.

Linear codes are great from the perspective of efficient encoding since they have a compact description (just store the generator matrix or the parity-check matrix) and to encode a message x we just compute Gx.

Efficient error-correction is not as obvious (and not guaranteed in general → more on this later).

Let's show how to easily correct 1 error with a Hamming code C. The parity-check view will again be useful.

Suppose we receive $z \in \{0,1\}^n$ obtained by applying 1 error to some $c \in C$. Let H be the parity-check matrix of C. The syndrome of z is defined as Hz.

Note that:

- $Hz = 0 \iff z \in C$
- If $z = c + e$ for some $c \in C$, then

$$Hz = H(c+e) = Hc + He = He.$$

In our case, $z = c + e$ with e a vector of Hamming weight 1. So $Hz = He$, and, if the error occurred in the i-th position, then He is just the i-th column of H, which is the binary representation of i. So, to decode z we can just compute the syndrome Hz, convert it to an integer $i \in \{1, \dots, n\}$, and flip the ith coordinate of z.