CODING THEORY

Summer 2025

1 Introduction

Encoding information with redundancy with the purpose of establishing reliable communication in the presence of noise.

You can fill out omi$\underset{s}{\square}$sions and correct er$\underset{r}{X}$ors.

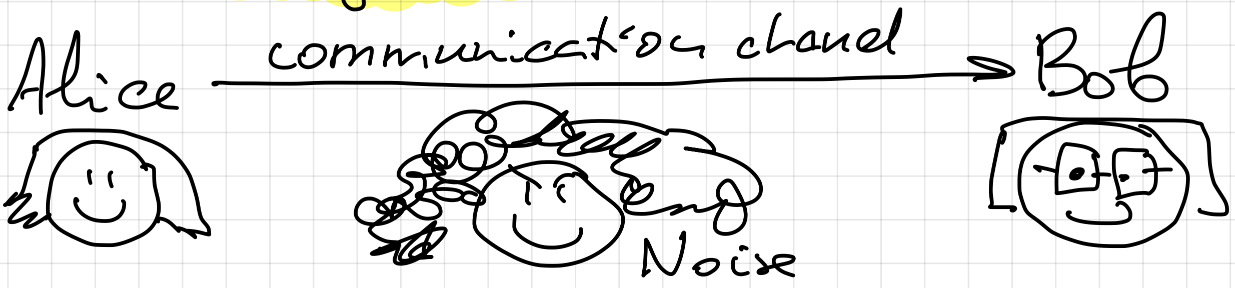
1.1 Applications in real life

Every where, where one needs to send/receive or store information in the presence of noise, one uses error correcting and error detecting codes.

Examples:

- mobile networks
- satelites
- QR codes
- Hard drives, USB sticks, chips

1.2 Model: Sending discrete data through a noisey channel.



Alice has a message m to be sent (some finite data). She encodes m as a codeword c. Because of the noise, c gets transmitted as $\tilde{c}$ to Bob. One wants to devise way of encoding with redundancy so that Bob has good chances to figure out what m was out of $\tilde{c}$ if the amount of noise is not too high.

The whole coding theory is about the trade off:

amount of redundancy one wants to introduce (to be minimized)

The number of errors one wants to detect/correct (to be maximized).

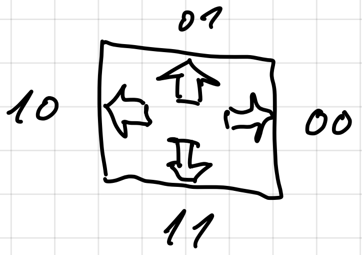
Three milestones we want to achieve:

- Bounds on codes (limitations)
- Shannon's theorem on the capacity of noisey channel.
- Understanding some codes that are used in practice.

1.3 Motivating examples

1.3.1 Encoding 2 to 3 bits for error detection

Set of messages $\{00, 01, 10, 11\}$



$$m_1 m_2 \in \{0,1\}^2 \xrightarrow{\text{encoding}} c_1 c_2 c_3 \in \{0,1\}^3$$

with

$$c_1 := m_1$$
$$c_2 := m_2$$

$$c_3 := m_1 \text{ XOR } m_2 ,$$

or equivalently

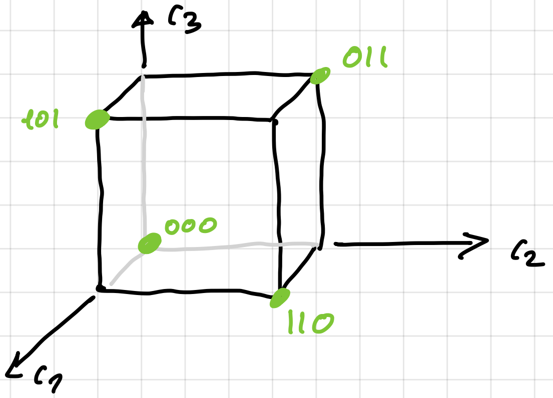
$$c_3 = \begin{cases} 1 & \text{if } m_1 \neq m_2 \\ 0 & \text{if } m_1 = m_2 \end{cases}$$

Description of the encoding via a table

message	code word
00	000
01	011
10	101
11	110

This code can detect one error, but it cannot correct it.

If the code word is 111 and there was one error, we know that the message was 10 or 01 or 11



No pair of the codewords is connected by an edge of the 0/1-cube in the picture.

This reflects the fact that one can detect one error.

Question for the lecture (about Linear Algebra, because we are going to need it).

$\boxed{LQ1}$

(a) Define a vector subspace U of $\mathbb{R}^n$ ($n \in \mathbb{N}$). What does it mean that U is a vector subspace of $\mathbb{R}^n$.

(b) Describe the line in $\mathbb{R}^3$ that contains $(0,0,0)$ and $(1,2,3)$ by a system of two linear equations. That is describe this line as a set of all (x,y,z) that satisfy certain two linear equations in x,y,z.

1.3.2 Encoding 2 bits via 6 for the correction one error.

$$x_1 x_2 \in \{0,1\}^2 \xmapsto{\text{encoding}} c_1 c_2 c_3 c_4 c_5 c_6 \in \{0,1\}^6$$

with
$$c_1 c_2 = x_1 x_2$$
$$c_3 c_4 = x_1 x_2$$
$$c_5 c_6 = x_1 x_2$$

The message is repeated 3 times.
That is the encoding is a map $f : \{0,1\}^2 \to \{0,1\}^6$ given by $f(x) := xxx$. You can also write with commas and brackets then

$f(x_1,x_2) = (x_1,x_2,x_1,x_2,x_1,x_2)$.

A way to spot and correct one error

$$x = x_1 x_2 \longmapsto f(x) = c = xxx \overset{\text{Noise}}{\underset{\text{1 error}}{\rightsquigarrow}} uvw$$

with $u = u_1 u_2 \in \{0,1\}^2$

$v = v_1 v_2 \in \{0,1\}^2$

$w = w_1 w_2 \in \{0,1\}^2$

If there has been exactly one error, then among u, v, w two of the strings are equal to x and one is not equal to x.

So, what we can do is the so-called majority voting (in a special case). That is:

The two-bit string among u, v, w that is present in two copies is the message x.

The decoder:

```
def g(u, v, w) :
    if u == v :
        return u
    if v == w:
        return v
    if u == w:
        return u
    raise Error("it seems to be too noisy today")
```

But we can do better...

1.3.3 Encoding 2 bits via 5 ~~for~~ the correction of one error.

Lets introduce a new encoding:

$$f : \{0,1\}^2 \longrightarrow \{0,1\}^5 \quad \text{with}$$

$$f(x_1, x_2) := (\underbrace{x_1, x_2}, \underbrace{x_1, x_2}, \underbrace{x_1 \text{ XOR } x_2})$$

two copies of the message

parity bit,
$= 1$ if $x_1 \neq x_2$
$= 0$ if $x_1 = x_2$.

Let's describe f via a table:

x_1 x_2	$f(x_1, x_2)$
0 0	00 00 0
0 1	01 01 1
1 0	10 10 1
1 1	11 11 0

code of size 4.

This encoding can correct one error:

The decoder:

```
def g(u_1, u_2, v_1, v_2, p):
    if u_1 u_2 == v_1 v_2:
        return u_1, u_2
    if u_1 xor u_2 == p:
        return u_1, u_2
    if v_1 xor v_2 == p:
        return v_1 v_2
```

This code can detect up to 2 errors.

Let's implement the error detector.

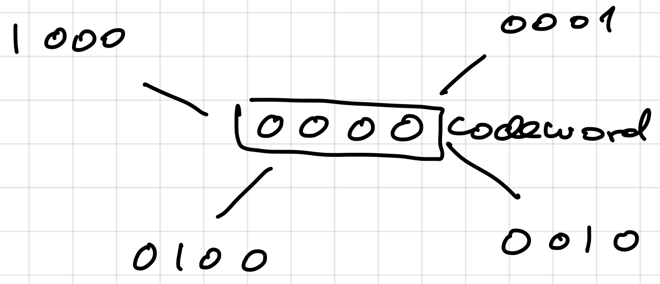
```
def error_occurred($u_1$, $u_2$, $v_1$, $v_2$, $p$):
    if $u_1 \neq v_1$:
        return "yes, error"
    if $u_2 \neq v_2$:
        return "yes, error"
    if $p \neq u_1 \text{ xor } u_2$:
        return "yes, error"
    return "No error,
            provided there
            have been no
            more than two
            errors"
```

$u_1 = x_1$
$u_2 = x_2$
$v_1 = x_1$
$v_2 = x_2$
$\Downarrow$ $p = x_1 \text{ xor } x_2$

$u_1 = v_1$
$u_2 = v_2$
$p = u_1 \text{ xor } u_2$

1.3.4 Encoding 2 bits via 4 with the correction one error: possible?

Assume, we could do it ....



1-error possibilities for a codeword.

If $c \in \{0,1\}^4$ is a codeword, then none of the 4 1-error deviations of c can be a codeword, if we want to be able to correct one error.

So, each codeword comes with 4 more "partners" making up a "neighborhood" of 5 words.

For each of the $2^2 = 4$ messages, we have one such neighborhood and those 4 neighborhoods with 5 nodes each don't share nodes. This makes up altogether $4 \cdot 5 = 20$ words. But the number of words of length 4 is $2^4 = 16$. $16 < 20$ and so we have a contradiction.

LQ2 Can one encode the three messages 00, 10, 01 via 4 bits with the possibility of the correction of one error.

In more technical terms: does there exist a binary code of size 3, length 4 and the minimum distance 3.