Lecture I

Introduction to coding theory

Emmanuela Orsini

[1]Department of Computer Science
University of Bristol

CoCoNuT, 2015

Lecture outline

References

- van Lint, J.H., Introduction to Coding Theory, *Graduate Texts in Mathematics*, 86, (Third Edition) 1999, Springer-Verlag.
- Augot, D., Betti, E., Orsini, E., An Introduction to Linear and Cyclic Codes, in *Gröbner Bases, Coding, and Cryptography*, pp. 47-69 2009, Springer.
- MacWilliams, F.J. and Sloane, N.J.A., The Theory of Error Correcting Codes, *North-Holland Mathematical Library*, 1978.
- Huffman, W.C and Pless, V., Fundamentals of Error-Correcting Codes, ITPro collection, 2003, *Cambridge University Press*.
- Shannon, C. E., *A Mathematical Theory of Communication*, Bell System Technical Journal 27, 1948.
- Hamming, R. W., *Error detecting and Error Correcting Codes*, Bell System Technical Journal 29, 1950, pp. 147-160.
- MacKay, D., Information Theory, Inference, and Learning Algorithms, 2003, *Cambridge University Press*.
- Huffman, W. C. and Brualdi, Richard A., Handbook of Coding Theory, 1998, *Elsevier Science Inc.*, New York, NY, USA.
- Steven Roman, Introduction to coding and information theory, Springer, *Undergraduate texts in mathematics*, 1997

- Official page COMS20002:

  Communication, Complexity and Number Theory

- My personal homepage for news, slides and (maybe) exercises

  `http://www.cs.bris.ac.uk/home/cseao/`

Lecture outline

1. Introduction
   - Motivation
2. Error correcting codes
   - Error detection and correction
   - Hamming code
3. Block codes
   - First definitions
   - Hamming distance and minimum distance of a code
   - Maximum Likelihood Decoding (MLD)
   - Error correction and error detection capability
4. Modelling noisy channel: a simple example

What does she say?

“Wel*ome to t*is c*ass!” $\longrightarrow$

What does she say?

“Wel*ome to t*is c*ass!” $\longrightarrow$ “Welcome to this class!”

What does she say?

"Wel*ome to t*is c*ass!" $\longrightarrow$ "Welcome to this class!"

Why is this example working?

- English has in built redundancy, so that it can tolerate *errors*.

Motivation

More in general, consider the following applications of *data storage* or *transmission*:

- CDs and DVDs
- Satellite/Digital Television
- Deep space probes
- Internet communications
- Mobile phones
- Computer hard disks/memory/floppy etc

Motivation

More in general, consider the following applications of *data storage* or *transmission*:

- CDs and DVDs
- Satellite/Digital Television
- Deep space probes
- Internet communications
- Mobile phones
- Computer hard disks/memory/floppy etc

In all of these the data can become corrupted.

- It is prone to errors

However they still work

- How?

Motivation

Goal: reliable systems of communication

Motivation

Goal: reliable systems of communication

HOW?

Motivation

Goal: reliable systems of communication

HOW?

1. **Physical solution**: channels with no noise

Motivation

Goal: reliable systems of communication

HOW?

1. **Physical solution**: channels with no noise
2. **System solution**: accept channels as they are, but manipulate the data in order to obtain reliable systems

Coding theory - Applications

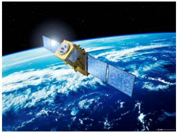
- Internet

Coding theory - Applications

- Internet
- Mobile phones

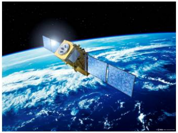
Coding theory - Applications

- Internet
- Mobile phones
- Satellite broadcast
    - TV



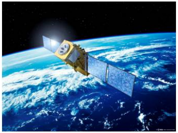
Coding theory - Applications

- Internet
- Mobile phones
- Satellite broadcast
  - TV
- Deep space telecommunications
  - Mars Rover



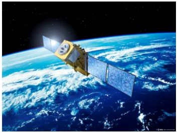
Coding theory - Applications

- Internet
- Mobile phones
- Satellite broadcast
  - TV
- Deep space telecommunications
  - Mars Rover
- Data storage



Coding theory - Applications

- Internet
- Mobile phones
- Satellite broadcast
  - TV
- Deep space telecommunications
  - Mars Rover
- Data storage

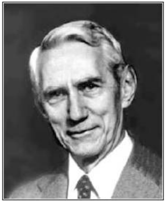


Codes are all around us!

Coding theory - The birth

"The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point"
(Claude Shannon, 1948)

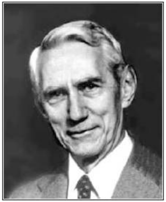
- In 1948, Claude E. Shannon wrote *"A Mathematical Theory of Communication"*, which marked the beginning of both Information and Coding Theory



Coding theory - The birth

"The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point"
(Claude Shannon, 1948)

- In 1948, Claude E. Shannon wrote *"A Mathematical Theory of Communication"*, which marked the beginning of both Information and Coding Theory



- In 1950, Richard W. Hamming wrote *"Error Detecting and Error Correcting Codes"*, which was the first paper explicitly introducing error-correcting codes

Coding and Information Theory

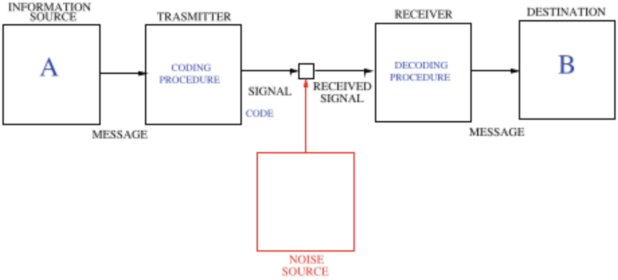
- **Coding Theory**: mainly concerned on explicit methods for efficient and reliable data transmission and storage

Coding and Information Theory

- **Coding Theory**: mainly concerned on explicit methods for efficient and reliable data transmission and storage
- **Information Theory**: it provides the performance limits on what can be done by suitable encoding of information
  - How should information be measured?
  - How much additional information is gained by some reduction in uncertainty?
  - What is the information content of a random variable?
  - How does the noise level in a communication channel limit its capacity to transmit information?
  - How does the bandwidth (in cycles/second) of a communication channel limit its capacity to transmit information?

Coding theory

The general idea is that of adding some kind of redundancy to the message that we want to send over a communication channel



Transmitted signal + noise = received signal

Lecture outline

1. Introduction
   - Motivation
2. Error correcting codes
   - Error detection and correction
   - Hamming code
3. Block codes
   - First definitions
   - Hamming distance and minimum distance of a code
   - Maximum Likelihood Decoding (MLD)
   - Error correction and error detection capability
4. Modelling noisy channel: a simple example

Digital Data

Digital data is sent as a series of ones and zeros.

- 11110101111101010100011010101011

Sometimes an error occurs:

- 1111011**1**1111101010100011010101011

We would like to be able to either detect or correct such errors.

Digital Data

Digital data is sent as a series of ones and zeros.

- 11110101111101010100011010101011

Sometimes an error occurs:

- 11110111111101010100011010101011

We would like to be able to either detect or correct such errors.

Detection

- Good if we can request a resend of the data

Digital Data

Digital data is sent as a series of ones and zeros.

- 1111010111110101010001101010101011

Sometimes an error occurs:

- 1111011111110101010001101010101011

We would like to be able to either detect or correct such errors.

Detection

- Good if we can request a resend of the data

Correction

- Needed if data cannot be resent (e.g. CD/DVD) or too costly to resend (e.g. deep space probe)

Simple Error Detection

Most data is first bundled up into a group of bits before sending

- e.g. 4, 8, 32 or 64 bits at a time

Simple Error Detection

Most data is first bundled up into a group of bits before sending

- e.g. 4, 8, 32 or 64 bits at a time

A simple detection trick is to add a parity bit

Suppose we wish to transmit 4 bits

- 0110

We add in an extra bit which signals whether the original data

has an even or odd number of ones

Simple Error Detection

Most data is first bundled up into a group of bits before sending

- e.g. 4, 8, 32 or 64 bits at a time

A simple detection trick is to add a parity bit

Suppose we wish to transmit 4 bits

- 0110

We add in an extra bit which signals whether the original data

has an even or odd number of ones

- The extra bit denotes the parity of the original bits

Simple Error Detection

Most data is first bundled up into a group of bits before sending

- e.g. 4, 8, 32 or 64 bits at a time

A simple detection trick is to add a parity bit

Suppose we wish to transmit 4 bits

- 0110

We add in an extra bit which signals whether the original data

has an even or odd number of ones

- The extra bit denotes the parity of the original bits

$$
\begin{aligned}
0110 &\longrightarrow 0110\mathbf{0} \\
1111 &\longrightarrow 1111\mathbf{0} \\
1000 &\longrightarrow 1000\mathbf{1} \\
1011 &\longrightarrow 1011\mathbf{1}
\end{aligned}
$$

Simple Error Detection

The previous example can be described mathematically as follows.

Simple Error Detection

The previous example can be described mathematically as follows.
We wish to send four message bits

$$m_1 m_2 m_3 m_4 \in \{0,1\}^4$$

Simple Error Detection

The previous example can be described mathematically as follows. We wish to send four message bits

$$m_1 m_2 m_3 m_4 \in \{0,1\}^4$$

To do this we add a fifth bit equal to

$$m_5 = m_1 \oplus m_2 \oplus m_3 \oplus m_4$$

where

$$x \oplus y = x + y \pmod 2$$

Simple Error Detection

The previous example can be described mathematically as follows.
We wish to send four message bits

$$m_1 m_2 m_3 m_4 \in \{0, 1\}^4$$

To do this we add a fifth bit equal to

$$m_5 = m_1 \oplus m_2 \oplus m_3 \oplus m_4$$

where

$$x \oplus y = x + y \pmod 2$$

The resulting five bits is called a codeword

$$C = \{c = m_1 m_2 m_3 m_4 m_5 \mid \sum_{i=1}^{5} m_i = 0\} \subseteq \{0, 1\}^5$$

Simple Error Detection

We can now detect whether a single error has occurred.

Suppose you receive the following data using the previous example:

- 10101
- 01110
- 11101
- 11111
- 00000
- 00001

Are there any errors ?

Simple Error Detection

We can now detect whether a single error has occurred.

Suppose you receive the following data using the previous example:

- 10101 Errors
- 01110
- 11101
- 11111
- 00000
- 00001

Simple Error Detection

We can now detect whether a single error has occurred.

Suppose you receive the following data using the previous example:

- 10101 Errors
- 01110 Errors
- 11101
- 11111
- 00000
- 00001

Simple Error Detection

We can now detect whether a single error has occurred.

Suppose you receive the following data using the previous example:

- 10101 Errors
- 01110 Errors
- 11101 No errors
- 11111
- 00000
- 00001

Simple Error Detection

We can now detect whether a single error has occurred.

Suppose you receive the following data using the previous example:

- 10101 Errors
- 01110 Errors
- 11101 No errors
- 11111 Errors
- 00000
- 00001

Simple Error Detection

We can now detect whether a single error has occurred.

Suppose you receive the following data using the previous example:

- 10101 Errors
- 01110 Errors
- 11101 No errors
- 11111 Errors
- 00000 No errors
- 00001

Simple Error Detection

We can now detect whether a single error has occurred.

Suppose you receive the following data using the previous example:

- 10101 Errors
- 01110 Errors
- 11101 No errors
- 11111 Errors
- 00000 No errors
- 00001 Errors

Simple Error Detection

We can now detect whether a single error has occurred.

Suppose you receive the following data using the previous example:

- 10101 Errors
- 01110 Errors
- 11101 No errors
- 11111 Errors
- 00000 No errors
- 00001 Errors

Trouble is we do not know where the errors occurred

Lecture outline

Detecting errors - Hamming code I

Again sticking to four bits of message

$$m_1 m_2 m_3 m_4$$

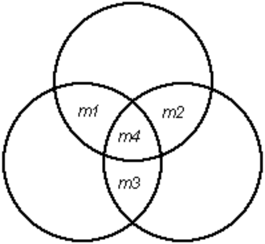
The idea is to use multiple parity-check bits.

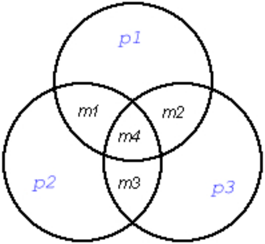
Detecting errors - Hamming code I

Again sticking to four bits of message

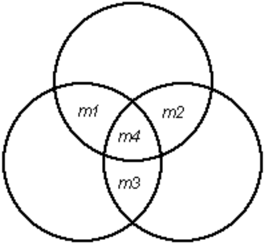
$$m_1 m_2 m_3 m_4$$

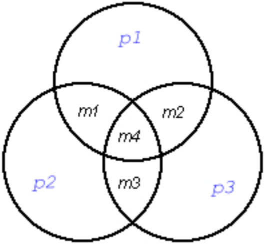
The idea is to use multiple parity-check bits.



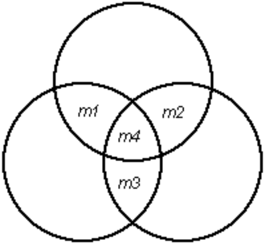


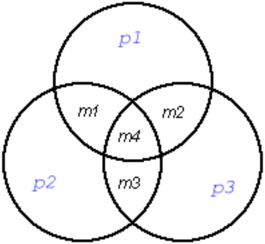
Detecting errors - Hamming code II



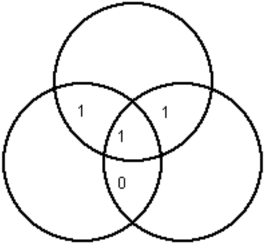


Detecting errors - Hamming code II

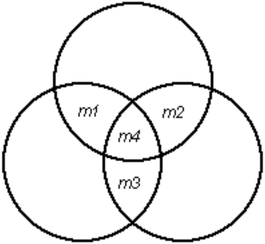


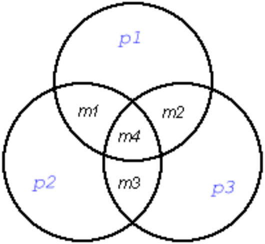


Suppose $\mathbf{m} = 1101$ is the message

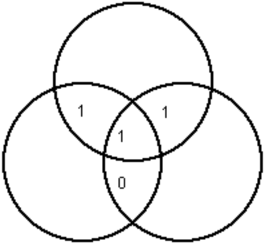


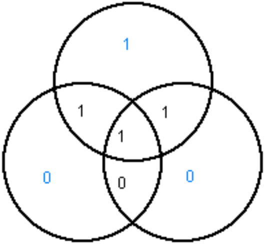
Detecting errors - Hamming code II



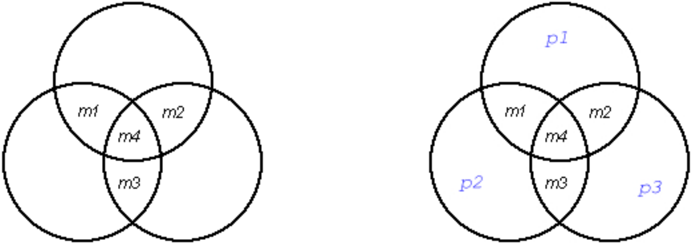


Suppose $\mathbf{m} = 1101$ is the message

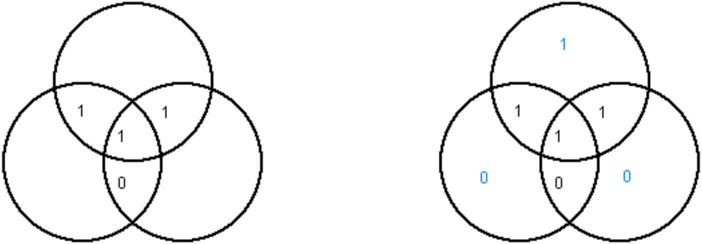




Detecting errors - Hamming code II



Suppose $\mathbf{m} = 1101$ is the message $\rightarrow 1101100$ is the codeword

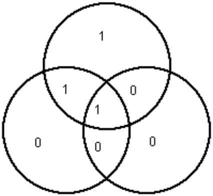


Detecting errors - Hamming code III

Suppose that after transmission one symbol is flipped and $\mathbf{r} = 1001100$ is received.

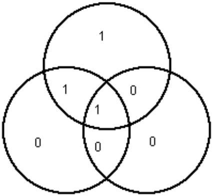
Detecting errors - Hamming code III

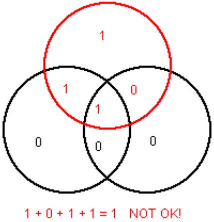
Suppose that after transmission one symbol is flipped and $\mathbf{r} = 1001100$ is received.



Detecting errors - Hamming code III

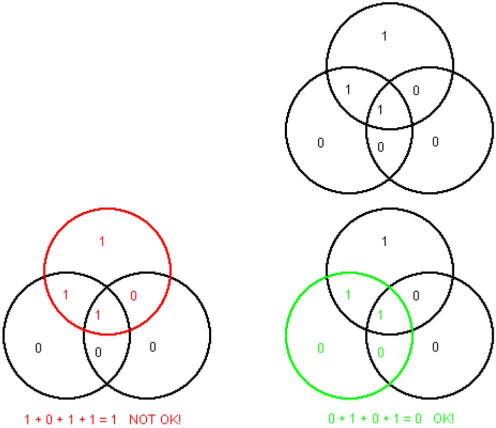
Suppose that after transmission one symbol is flipped and $\mathbf{r} = 1001100$ is received.





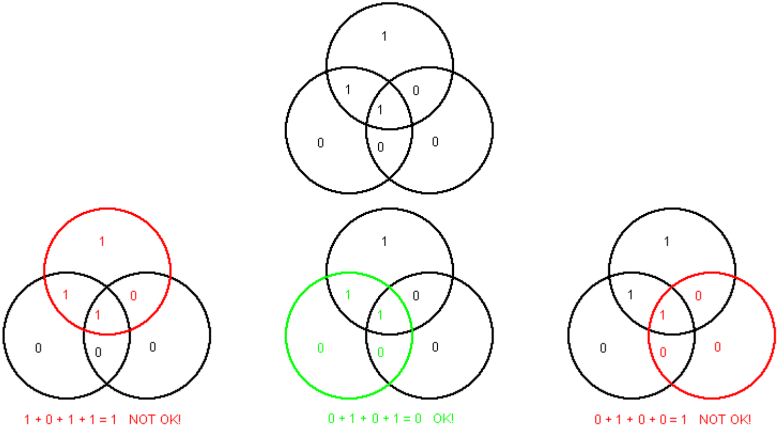
Detecting errors - Hamming code III

Suppose that after transmission one symbol is flipped and $\mathbf{r} = 1001100$ is received.



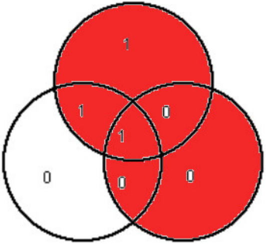
Detecting errors - Hamming code III

Suppose that after transmission one symbol is flipped and $\mathbf{r} = 1001100$ is received.

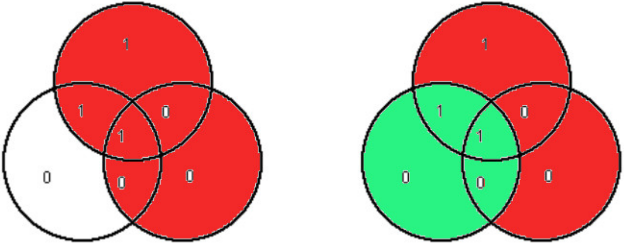


Correcting errors - Hamming code IV

$\mathbf{c} = 1101100$ and $\mathbf{r} = 1001100$

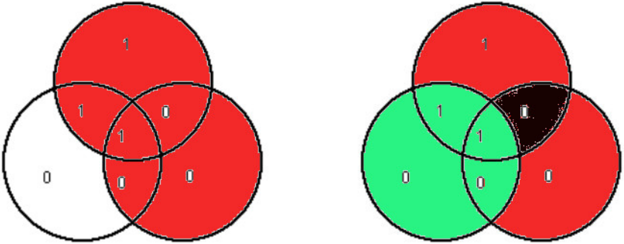


$\mathbf{c} = 1101100$ and $\mathbf{r} = 1001100$



Correcting errors - Hamming code IV

$\mathbf{c} = 1101100$ and $\mathbf{r} = 1001100$



$\rightarrow$ the error is at m_2

Hamming code - Basic idea

- Use multiple parity bits, each covering a subset of the message bits

$$
\begin{aligned}
m_1 + m_4 + m_2 + p_1 = 0 \quad &\longrightarrow \quad \{m_1, m_4, m_2, p_1\} \\
m_1 + m_3 + m_4 + p_2 = 0 \quad &\longrightarrow \quad \{m_1, m_3, m_4, p_2\} \\
m_2 + m_4 + m_3 + p_3 = 0 \quad &\longrightarrow \quad \{m_2, m_4, m_3, p_3\}
\end{aligned}
$$

Hamming code - Basic idea

- Use multiple parity bits, each covering a subset of the message bits

$$
\begin{aligned}
m_1 + m_4 + m_2 + p_1 = 0 \quad &\longrightarrow \quad \{m_1, m_4, m_2, p_1\} \\
m_1 + m_3 + m_4 + p_2 = 0 \quad &\longrightarrow \quad \{m_1, m_3, m_4, p_2\} \\
m_2 + m_4 + m_3 + p_3 = 0 \quad &\longrightarrow \quad \{m_2, m_4, m_3, p_3\}
\end{aligned}
$$

- the subsets overlap, i.e. each message-bit belongs to multiple subsets

Hamming code - Basic idea

- Use multiple parity bits, each covering a subset of the message bits

$$m_1 + m_4 + m_2 + p_1 = 0 \quad \longrightarrow \quad \{m_1, m_4, m_2, p_1\}$$
$$m_1 + m_3 + m_4 + p_2 = 0 \quad \longrightarrow \quad \{m_1, m_3, m_4, p_2\}$$
$$m_2 + m_4 + m_3 + p_3 = 0 \quad \longrightarrow \quad \{m_2, m_4, m_3, p_3\}$$

- the subsets overlap, i.e. each message-bit belongs to multiple subsets

Hamming code - Basic idea

- Use multiple parity bits, each covering a subset of the message bits

$$
\begin{aligned}
m_1 + m_4 + m_2 + p_1 = 0 \quad &\longrightarrow \quad \{m_1, m_4, m_2, p_1\} \\
m_1 + m_3 + m_4 + p_2 = 0 \quad &\longrightarrow \quad \{m_1, m_3, m_4, p_2\} \\
m_2 + m_4 + m_3 + p_3 = 0 \quad &\longrightarrow \quad \{m_2, m_4, m_3, p_3\}
\end{aligned}
$$

- the subsets overlap, i.e. each message-bit belongs to multiple subsets

Hamming code - Basic idea

- Use multiple parity bits, each covering a subset of the message bits

$$
\begin{aligned}
m_1 + m_4 + m_2 + p_1 = 0 &\quad \longrightarrow \quad \{m_1, m_4, m_2, p_1\} \\
m_1 + m_3 + m_4 + p_2 = 0 &\quad \longrightarrow \quad \{m_1, m_3, m_4, p_2\} \\
m_2 + m_4 + m_3 + p_3 = 0 &\quad \longrightarrow \quad \{m_2, m_4, m_3, p_3\}
\end{aligned}
$$

- the subsets overlap, i.e. each message-bit belongs to multiple subsets

Hamming code - Basic idea

- Use multiple parity bits, each covering a subset of the message bits

$$
\begin{aligned}
m_1 + m_4 + m_2 + p_1 = 0 &\quad\longrightarrow\quad \{m_1, m_4, m_2, p_1\}\\
m_1 + m_3 + m_4 + p_2 = 0 &\quad\longrightarrow\quad \{m_1, m_3, m_4, p_2\}\\
m_2 + m_4 + m_3 + p_3 = 0 &\quad\longrightarrow\quad \{m_2, m_4, m_3, p_3\}
\end{aligned}
$$

- the subsets overlap, i.e. each message-bit belongs to multiple subsets

Hamming code - Basic idea

- Use multiple parity bits, each covering a subset of the message bits

$$
\begin{aligned}
m_1 + m_4 + m_2 + p_1 = 0 \quad &\longrightarrow \quad \{m_1, m_4, m_2, p_1\} \\
m_1 + m_3 + m_4 + p_2 = 0 \quad &\longrightarrow \quad \{m_1, m_3, m_4, p_2\} \\
m_2 + m_4 + m_3 + p_3 = 0 \quad &\longrightarrow \quad \{m_2, m_4, m_3, p_3\}
\end{aligned}
$$

- the subsets overlap, i.e. each message-bit belongs to multiple subsets
- No two message bits belong to exactly the same subsets. In this way it is possible to correct one error and to detect two errors.

Hamming code - Basic idea

- Use multiple parity bits, each covering a subset of the message bits

$$m_1 + m_4 + m_2 + p_1 = 0 \quad \longrightarrow \quad \{m_1, m_4, m_2, p_1\} \; OK$$
$$m_1 + m_3 + m_4 + p_2 = 0 \quad \longrightarrow \quad \{m_1, m_3, m_4, p_2\}$$
$$m_2 + m_4 + m_3 + p_3 = 0 \quad \longrightarrow \quad \{m_2, m_4, m_3, p_3\} \; ERRORS$$

- the subsets overlap, i.e. each message-bit belongs to multiple subsets
- No two message bits belong to exactly the same subsets. In this way it is possible to correct one error and to detect two errors.

Example

$\mathbf{c} = 1101100$ and $\mathbf{r} = 1001000$, so two errors occurred, at m_2 and p_1. Then:

Hamming code - Basic idea

- Use multiple parity bits, each covering a subset of the message bits

$$
\begin{aligned}
m_1 + m_4 + m_2 + p_1 = 0 \quad &\longrightarrow \quad \{m_1, m_4, m_2, p_1\}\ \textit{OK} \\
m_1 + m_3 + m_4 + p_2 = 0 \quad &\longrightarrow \quad \{m_1, m_3, m_4, p_2\} \\
m_2 + m_4 + m_3 + p_3 = 0 \quad &\longrightarrow \quad \{m_2, m_4, m_3, p_3\}\ \textit{ERRORS}
\end{aligned}
$$

- the subsets overlap, i.e. each message-bit belongs to multiple subsets
- No two message bits belong to exactly the same subsets. In this way it is possible to correct one error and to detect two errors.

Example

$\mathbf{c} = 1101100$ and $\mathbf{r} = 1001000$, so two errors occurred, at m_2 and p_1. Then:

- The code detects that some errors occurred

Hamming code - Basic idea

- Use multiple parity bits, each covering a subset of the message bits

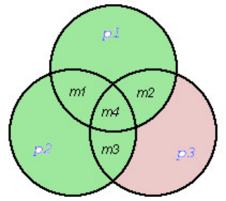
$$
\begin{aligned}
m_1 + m_4 + m_2 + p_1 = 0 \quad &\longrightarrow \quad \{m_1, m_4, m_2, p_1\}\ OK \\
m_1 + m_3 + m_4 + p_2 = 0 \quad &\longrightarrow \quad \{m_1, m_3, m_4, p_2\} \\
m_2 + m_4 + m_3 + p_3 = 0 \quad &\longrightarrow \quad \{m_2, m_4, m_3, p_3\}\ ERRORS
\end{aligned}
$$

- the subsets overlap, i.e. each message-bit belongs to multiple subsets
- No two message bits belong to exactly the same subsets. In this way it is possible to correct one error and to detect two errors.

Example

$\mathbf{c} = 1101100$ and $\mathbf{r} = 1001000$, so two errors occurred, at m_2 and p_1. Then:

- The code detects that some errors occurred
- The code concludes the error is at p_3, introducing an extra error



Hamming code

Return

- Enc : $\{0,1\}^4 \to \{0,1\}^7$ that maps the 2^4 strings of 4 bits $\mathbf{m}$ into a codeword $\mathbf{c}$

Hamming code

Return

- $\text{Enc} : \{0,1\}^4 \to \{0,1\}^7$ that maps the 2^4 strings of 4 bits $\mathbf{m}$ into a codeword $\mathbf{c}$
- We can write down all the codewords:

Information bits	Codeword	Information bits	Codeword
0000	0000000	1000	1000110
0001	0001111	1001	1001001
0100	0010101	1010	1010101
0011	0011100	1011	1011010
0010	0010011	1100	1100011
0101	0101010	1101	1101100
0110	0110110	1110	1110000
0111	0111001	1111	1111111

Hamming code

◂ Return

- $\mathrm{Enc}: \{0,1\}^4 \to \{0,1\}^7$ that maps the 2^4 strings of 4 bits $\mathbf{m}$ into a codeword $\mathbf{c}$
- We can write down all the codewords:

Information bits	Codeword	Information bits	Codeword
0000	0000000	1000	1000110
0001	0001111	1001	1001001
0100	0010101	1010	1010101
0011	0011100	1011	1011010
0010	0010011	1100	1100011
0101	0101010	1101	1101100
0110	0110110	1110	1110000
0111	0111001	1111	1111111

- C contains 16 codewords of length 7

Lecture outline

Block codes - Notation I

- Let $\mathcal{A}$ be an alphabet of cardinality q

Block codes - Notation I

- Let $\mathcal{A}$ be an alphabet of cardinality q
- We consider codes C over $\mathcal{A}$. If $q = 2$ the code is called binary

Block codes - Notation I

- Let $\mathcal{A}$ be an alphabet of cardinality q
- We consider codes C over $\mathcal{A}$. If $q = 2$ the code is called binary
- Let Enc be an injective map:

$$\mathrm{Enc} : \mathcal{A}^k \longrightarrow \mathcal{A}^n \qquad C \text{ is the image of Enc}$$

Block codes - Notation I

- Let $\mathcal{A}$ be an alphabet of cardinality q
- We consider codes C over $\mathcal{A}$. If $q = 2$ the code is called binary
- Let Enc be an injective map:

$$\text{Enc} : \mathcal{A}^k \longrightarrow \mathcal{A}^n \qquad C \text{ is the image of Enc}$$

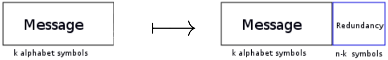
$\longmapsto$

the entire block is called codeword

Block codes - Notation I

- Let $\mathcal{A}$ be an alphabet of cardinality q
- We consider codes C over $\mathcal{A}$. If $q = 2$ the code is called binary
- Let Enc be an injective map:

$$\text{Enc} : \mathcal{A}^k \longrightarrow \mathcal{A}^n \qquad C \text{ is the image of Enc}$$



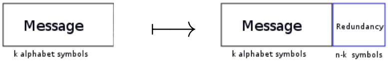
the entire block is called codeword

- n is the length of a codeword

Block codes - Notation I

- Let $\mathcal{A}$ be an alphabet of cardinality q
- We consider codes C over $\mathcal{A}$. If $q = 2$ the code is called binary
- Let Enc be an injective map:

$$\text{Enc} : \mathcal{A}^k \longrightarrow \mathcal{A}^n \qquad C \text{ is the image of Enc}$$



the entire block is called codeword

- n is the length of a codeword

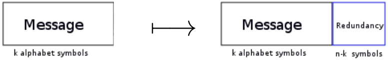
Definition

A Block code is a code with fixed length n, i.e. a non-empty subset of $\mathcal{A}^n$

Block codes - Notation I

- Let $\mathcal{A}$ be an alphabet of cardinality q
- We consider codes C over $\mathcal{A}$. If $q = 2$ the code is called binary
- Let Enc be an injective map:

$$\text{Enc} : \mathcal{A}^k \longrightarrow \mathcal{A}^n \qquad C \text{ is the image of Enc}$$



the entire block is called codeword

- n is the length of a codeword

Definition

A Block code is a code with fixed length n, i.e. a non-empty subset of $\mathcal{A}^n$

- If a block code $C \subseteq \mathcal{A}^n$ contains $M = q^k$ codewords, then M is the size of C

Block codes - Notation II

A block code of length n and size M is denoted by (n, M)-code

Block codes - Notation II

A block code of length n and size M is denoted by (n, M)-code

- $k = \log_q(M)$ message length

Block codes - Notation II

A block code of length n and size M is denoted by (n, M)-code

- $k = \log_q(M)$ message length
- $n - \log_q(M)$ redundancy

Block codes - Notation II

A block code of length n and size M is denoted by (n, M)-code

- $k = \log_q(M)$ message length
- $n - \log_q(M)$ redundancy
- $R = \frac{\log_q(M)}{n}$ information rate
  The rate of a code is the average amount of **real** information in each block of n symbols transmitted over the channel

Block codes - Notation II

A block code of length n and size M is denoted by (n, M)-code

- $k = \log_q(M)$ message length
- $n - \log_q(M)$ redundancy
- $R = \frac{\log_q(M)}{n}$ information rate
  The rate of a code is the average amount of **real** information in each block of n symbols transmitted over the channel

Hamming

Example

The Hamming code we have seen before is a binary $(7, 16)$ block code with information rate $4/7$.

Lecture outline

Hamming distance

Definition (Hamming distance)

Given two strings $\mathbf{x}$ and $\mathbf{y} \in \mathcal{A}^n$, the **Hamming distance** between $\mathbf{x}$ and $\mathbf{y}$ is

$$d(\mathbf{x}, \mathbf{y}) = |\{i | x_i \neq y_i\}|.$$

Hamming distance

Definition (Hamming distance)

Given two strings $\mathbf{x}$ and $\mathbf{y} \in \mathcal{A}^n$, the **Hamming distance** between $\mathbf{x}$ and $\mathbf{y}$ is

$$d(\mathbf{x}, \mathbf{y}) = |\{i | x_i \neq y_i\}|.$$

Example

$\mathbf{v}_1 = 01011$
$\mathbf{v}_2 = 11110$

$$d(\mathbf{v}_1, \mathbf{v}_2) = 3$$

Hamming distance

Definition (Hamming distance)

Given two strings $\mathbf{x}$ and $\mathbf{y} \in \mathcal{A}^n$, the **Hamming distance** between $\mathbf{x}$ and $\mathbf{y}$ is

$$d(\mathbf{x}, \mathbf{y}) = |\{i | x_i \neq y_i\}|.$$

Example

$\mathbf{v}_1 = 01011$
$\mathbf{v}_2 = 11110$

$d(\mathbf{v}_1, \mathbf{v}_2) = 3$

$\mathbf{w}_1 = 3211$
$\mathbf{w}_2 = 0213$

$d(\mathbf{w}_1, \mathbf{w}_2) = 2$

Hamming distance

Definition (Hamming distance)

Given two strings $\mathbf{x}$ and $\mathbf{y} \in \mathcal{A}^n$, the **Hamming distance** between $\mathbf{x}$ and $\mathbf{y}$ is

$$d(\mathbf{x}, \mathbf{y}) = |\{i | x_i \neq y_i\}|.$$

Example

$\mathbf{v}_1 = 01011$
$\mathbf{v}_2 = 11110$

$d(\mathbf{v}_1, \mathbf{v}_2) = 3$

$\mathbf{w}_1 = 3211$
$\mathbf{w}_2 = 0213$

$d(\mathbf{w}_1, \mathbf{w}_2) = 2$

Definition (Code distance)

The (Hamming) **minimum distance of a code** C is given by

$$d(C) = min\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\}.$$

Definition (Hamming weight)

The **Hamming weight** of a string $\mathbf{x}$, $\mathrm{wt}(\mathbf{x})$, is defined as the number of non-zero symbols in the string.

Lecture outline

Decoding problem

Why is the distance of a code important?

Decoding problem

Why is the distance of a code important?

Let C be an (n, M) code and suppose that a codeword $\mathbf{c}$ is sent over a noisy channel:

Decoding problem

Why is the distance of a code important?

Let C be an (n, M) code and suppose that a codeword $\mathbf{c}$ is sent over a noisy channel:

$\mathbf{c}$  $\mathbf{r}$

Decoding problem

Why is the distance of a code important?

Let C be an (n, M) code and suppose that a codeword $\mathbf{c}$ is sent over a noisy channel:

1. if $\mathbf{r} \in C$ then no correction is needed 

Decoding problem

Why is the distance of a code important?

Let C be an (n, M) code and suppose that a codeword $\mathbf{c}$ is sent over a noisy channel:

1. if $\mathbf{r} \in C$ then no correction is needed 
2. if $\mathbf{r} \notin C$, then some errors occurred 

Decoding problem - Why is $d(C)$ important?

If $\mathbf{r} \notin C$: the decoder has to find the codeword c that has been sent

Decoding problem - Why is $d(C)$ important?

If $\mathbf{r} \notin C$: the decoder has to find the codeword c that has been sent

A possible strategy is the Maximum Likelihood Decoding (MLD): find the most likely codeword transmitted, i.e. the codeword $\mathbf{c}$ which maximizes the probability that $\mathbf{r}$ is the received word given that $\mathbf{c}$ has been sent.

Decoding problem - Why is $d(C)$ important?

If $\mathbf{r} \notin C$: the decoder has to find the codeword c that has been sent

A possible strategy is the Maximum Likelihood Decoding (MLD): find the most likely codeword transmitted, i.e. the codeword $\mathbf{c}$ which maximizes the probability that $\mathbf{r}$ is the received word given that $\mathbf{c}$ has been sent.

We will see that for some types of channel MLD is equivalent to finding the coderword $\mathbf{c}$ closest to $\mathbf{r}$ in the Hamming distance (Nearest neighbour decoding):

$$\min_{\mathbf{c} \in C} d(\mathbf{r}, \mathbf{c})$$

Decoding problem - Why is $d(C)$ important?

If $\mathbf{r} \notin C$: the decoder has to find the codeword c that has been sent


A possible strategy is the Maximum Likelihood Decoding (MLD): find the most likely codeword transmitted, i.e. the codeword $\mathbf{c}$ which maximizes the probability that $\mathbf{r}$ is the received word given that $\mathbf{c}$ has been sent.

We will see that for some types of channel MLD is equivalent to finding the coderword $\mathbf{c}$ closest to $\mathbf{r}$ in the Hamming distance (Nearest neighbour decoding):

$$\min_{\mathbf{c} \in C} d(\mathbf{r}, \mathbf{c})$$

From now on we will assume a type of channel such that we can use the minimum distance decoding to perform MLD

Decoding problem - Why is $d(C)$ important?

Let $\mathbf{x} \in \mathcal{A}^n$ and $t \in \mathbb{N}$, define

$$\mathcal{B}_t(\mathbf{x}) = \{\mathbf{y} \in \mathcal{A}^n \mid d(\mathbf{x}, \mathbf{y}) \leq t\}$$

Decoding problem - Why is $d(C)$ important?

Let $\mathbf{x} \in \mathcal{A}^n$ and $t \in \mathbb{N}$, define

$$\mathcal{B}_t(\mathbf{x}) = \{\mathbf{y} \in \mathcal{A}^n \mid d(\mathbf{x}, \mathbf{y}) \leq t\}$$

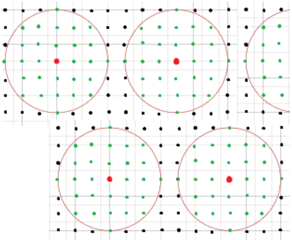
Image to cover the entire space $\mathcal{A}^n$ of balls of radius $\lfloor \frac{d-1}{2} \rfloor$ centered at distinct codewords:

Decoding problem - Why is $d(C)$ important?

Let $\mathbf{x} \in \mathcal{A}^n$ and $t \in \mathbb{N}$, define

$$\mathcal{B}_t(\mathbf{x}) = \{\mathbf{y} \in \mathcal{A}^n \mid d(\mathbf{x}, \mathbf{y}) \leq t\}$$

Image to cover the entire space $\mathcal{A}^n$ of balls of radius $\lfloor \frac{d-1}{2} \rfloor$ centered at distinct codewords:

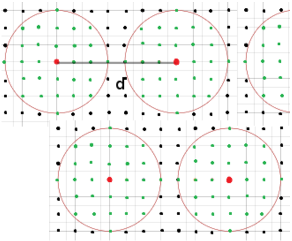


Decoding problem - Why is $d(C)$ important?

Let $\mathbf{x} \in \mathcal{A}^n$ and $t \in \mathbb{N}$, define

$$\mathcal{B}_t(\mathbf{x}) = \{\mathbf{y} \in \mathcal{A}^n \mid d(\mathbf{x}, \mathbf{y}) \leq t\}$$

Image to cover the entire space $\mathcal{A}^n$ of balls of radius $\lfloor \frac{d-1}{2} \rfloor$ centered at distinct codewords:



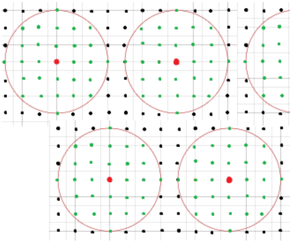
- because the distance of the code is d ball will be nonoverlapping

Decoding problem - Why is $d(C)$ important?

Let $\mathbf{x} \in \mathcal{A}^n$ and $t \in \mathbb{N}$, define

$$\mathcal{B}_t(\mathbf{x}) = \{\mathbf{y} \in \mathcal{A}^n \mid d(\mathbf{x}, \mathbf{y}) \leq t\}$$

Image to cover the entire space $\mathcal{A}^n$ of balls of radius $\lfloor \frac{d-1}{2} \rfloor$ centered at distinct codewords:



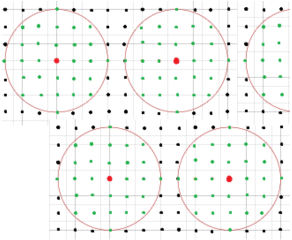
- because the distance of the code is d ball will be nonoverlapping
- if $\mathbf{r}$ = red word (codeword), then no errors

Decoding problem - Why is $d(C)$ important?

Let $\mathbf{x} \in \mathcal{A}^n$ and $t \in \mathbb{N}$, define

$$\mathcal{B}_t(\mathbf{x}) = \{\mathbf{y} \in \mathcal{A}^n \mid d(\mathbf{x}, \mathbf{y}) \leq t\}$$

Image to cover the entire space $\mathcal{A}^n$ of balls of radius $\lfloor \frac{d-1}{2} \rfloor$ centered at distinct codewords:



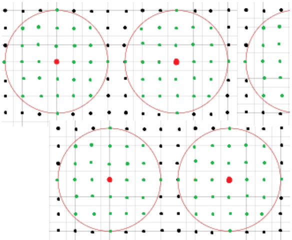
- because the distance of the code is d ball will be nonoverlapping
- if $\mathbf{r}$ = red word (codeword), then no errors
- if $\mathbf{r}$ = green word, we should correct it to the red coderword that is the center of the ball it lies in

Decoding problem - Why is $d(C)$ important?

Let $\mathbf{x} \in \mathcal{A}^n$ and $t \in \mathbb{N}$, define

$$\mathcal{B}_t(\mathbf{x}) = \{\mathbf{y} \in \mathcal{A}^n \mid d(\mathbf{x}, \mathbf{y}) \leq t\}$$

Image to cover the entire space $\mathcal{A}^n$ of balls of radius $\lfloor \frac{d-1}{2} \rfloor$ centered at distinct codewords:



- because the distance of the code is d ball will be nonoverlapping
- if $\mathbf{r}$ = red word (codeword), then no errors
- if $\mathbf{r}$ = green word, we should correct it to the red coderword that is the center of the ball it lies in
- if $\mathbf{r}$ = black word, then we are not able to correct because, if we increase the radii, balls would overlap

Lecture outline

More formally, we have the following definition:

- The error detection capability of a code C is the number e of errors that the code can detect. A *e-error detecting* code has minimum distance $d = e + 1$.

Error correction and error detection capability

More formally, we have the following definition:

- The error detection capability of a code C is the number e of errors that the code can detect. A *e-error detecting* code has minimum distance $d = e + 1$.

- The error correction capability of a code C is the number of errors that the code can correct. A *t-error correcting* code has minimum distance d such that $t = \lfloor \frac{d-1}{2} \rfloor$.

Erasures

In a similar way we can define the erasure correction capability of a code. An **erasure** occurs when a transmitted symbol is unreadable and at its place an extra symbol ϵ is introduced.

Erasures

In a similar way we can define the erasure correction capability of a code.
An **erasure** occurs when a transmitted symbol is unreadable and at its place an extra symbol ϵ is introduced.

Example

$\mathbf{c} = 1001100 \longrightarrow \mathbf{r} = 100*100$

Erasures

In a similar way we can define the erasure correction capability of a code.
An **erasure** occurs when a transmitted symbol is unreadable and at its place an extra symbol ϵ is introduced.

Example

$\mathbf{c} = 1001100 \longrightarrow \mathbf{r} = 100\epsilon100$

Erasures

In a similar way we can define the erasure correction capability of a code. An **erasure** occurs when a transmitted symbol is unreadable and at its place an extra symbol ϵ is introduced.

Example

$\mathbf{c} = 1001100 \longrightarrow \mathbf{r} = 100\epsilon100$

- A code can correct s erasures if $s < d$

Erasures

In a similar way we can define the erasure correction capability of a code. An **erasure** occurs when a transmitted symbol is unreadable and at its place an extra symbol ϵ is introduced.

Example

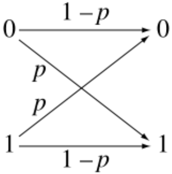
$\mathbf{c} = 1001100 \longrightarrow \mathbf{r} = 100\epsilon100$

- A code can correct s erasures if $s < d$
- The condition for simultaneous correction of t errors and s erasures is

$$d \geq 2t + s + 1.$$

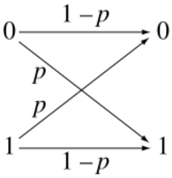
Binary Symmetric Channel (BSC)

$\mathcal{X} = \{0, 1\}$ input alphabet and $\mathcal{Y} = \{0, 1\}$ output alphabet.
A BSC is parametrized by the probability p, $0 \leq p < 1/2$, that an input bit is flipped. p depends on the noise level and is called *crossover probability*.



Binary Symmetric Channel (BSC)

$\mathcal{X} = \{0, 1\}$ input alphabet and $\mathcal{Y} = \{0, 1\}$ output alphabet.
A BSC is parametrized by the probability p, $0 \leq p < 1/2$, that an input bit is flipped. p depends on the noise level and is called *crossover probability*.



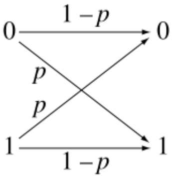
Example

Consider a binary code C of length 5.

- $\Pr(\mathbf{c}|\mathbf{c}) = ?$

Binary Symmetric Channel (BSC)

$\mathcal{X} = \{0, 1\}$ input alphabet and $\mathcal{Y} = \{0, 1\}$ output alphabet.
A BSC is parametrized by the probability p, $0 \leq p < 1/2$, that an input bit is flipped. p depends on the noise level and is called *crossover probability*.



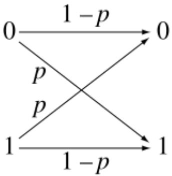
Example

Consider a binary code C of length 5.

- $\Pr(\mathbf{c}|\mathbf{c}) = (1-p)^5$

Binary Symmetric Channel (BSC)

$\mathcal{X} = \{0, 1\}$ input alphabet and $\mathcal{Y} = \{0, 1\}$ output alphabet.
A BSC is parametrized by the probability p, $0 \leq p < 1/2$, that an input bit is flipped. p depends on the noise level and is called *crossover probability*.



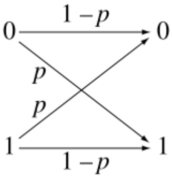
Example

Consider a binary code C of length 5.

- $\Pr(\mathbf{c}|\mathbf{c}) = (1-p)^5$
- If $\mathbf{c} = 10101$, $\Pr(01101|\mathbf{c}) = ?$

Binary Symmetric Channel (BSC)

$\mathcal{X} = \{0, 1\}$ input alphabet and $\mathcal{Y} = \{0, 1\}$ output alphabet.
A BSC is parametrized by the probability p, $0 \leq p < 1/2$, that an input bit is flipped. p depends on the noise level and is called *crossover probability*.



Example

Consider a binary code C of length 5.

- $\Pr(\mathbf{c}|\mathbf{c}) = (1 - p)^5$
- If $\mathbf{c} = 10101$, $\Pr(01101|\mathbf{c}) = p^2(1 - p)^3$

Binary Symmetric Channel

Suppose $\mathbf{c}$ is transmitted codeword and $\mathbf{r}$ is received word $\rightarrow \mathbf{c} = \mathbf{r} + \mathbf{e}$

Binary Symmetric Channel

Suppose $\mathbf{c}$ is transmitted codeword and $\mathbf{r}$ is received word $\rightarrow \mathbf{c} = \mathbf{r} + \mathbf{e}$
Given two codewords $\mathbf{c}_1, \mathbf{c}_2$, then

$$\begin{aligned}
\Pr(\mathbf{r}|\mathbf{c}_1) \leq \Pr(\mathbf{r}|\mathbf{c}_2) &\iff d(\mathbf{r}, \mathbf{c}_1) \geq d(\mathbf{r}, \mathbf{c}_2) \\
&\iff \mathrm{wt}(\mathbf{r} + \mathbf{c}_1) \geq \mathrm{wt}(\mathbf{r} + \mathbf{c}_2) \\
&\iff \mathrm{wt}(\mathbf{e}_1) \geq \mathrm{wt}(\mathbf{e}_2)
\end{aligned}$$

Binary Symmetric Channel

Suppose $\mathbf{c}$ is transmitted codeword and $\mathbf{r}$ is received word $\rightarrow \mathbf{c} = \mathbf{r} + \mathbf{e}$
Given two codewords $\mathbf{c}_1, \mathbf{c}_2$, then

$$\begin{aligned} \Pr(\mathbf{r}|\mathbf{c}_1) \leq \Pr(\mathbf{r}|\mathbf{c}_2) &\iff d(\mathbf{r}, \mathbf{c}_1) \geq d(\mathbf{r}, \mathbf{c}_2) \\ &\iff \mathrm{wt}(\mathbf{r} + \mathbf{c}_1) \geq \mathrm{wt}(\mathbf{r} + \mathbf{c}_2) \\ &\iff \mathrm{wt}(\mathbf{e}_1) \geq \mathrm{wt}(\mathbf{e}_2) \end{aligned}$$

The most likely codeword sent is the one corresponding to the error of smallest weight