AICC II

Arthur Herbette
Prof. Michael Gastpar

May 28, 2025

Contents

List of lectures

Note

This PDF has been done is LaTeXusing Vim. All 90% of the style file has been done by Joachim Favre. He also helped me getting my vim setup ready, so a Big thank to him. For those interested in how to setup vim there is Gille Castel who introduced here `https://castel.dev/post/lecture-notes-1/` how he does it.

Chapter 1

Introduction

1.1 About this course

In this course, there will be three main topics that will be studied:

- Communication
- Information and Data science
- Cryptography, Secrecy, Privacy

1.2 Cours Grading

- 90% Final exam during exam period
- 10 % Quizzes (online on Moodle)
  - There will be 6 quizzes. BO5
  - On the quizzes, you can update your answer as many times as you want before the deadline
- Quizzes are highly coorelated with homework.

1.2.1 How to be efficient and do well in this course

Before class:

- Browse through the slides to know what to expect
- review the background material as needed

After class:

- read the notes: they are the reference
- do the review questions

Before the exercice session

- are you up to date with the theory?
- Solve what you can ahead of time and finish during the exercice session
- write down **your** solution

February 18, 2025 — **Lecture 1 : Discrete Probability**

Chapter 2

Entropy

2.1 Initial case: Finite Ω: set of all possiblie outcomes

Definition 1 ***Sample space*** *Ω is the set of all possible outcomes*

Definition 2 ***Event*** *E: a subset of Ω. Since the outcomes are equally likely:*

$$p(E) = \frac{|E|}{|\Omega|}$$

2.2 Conditional Probability

Conditional probability

Definition 3 *The* ***conditional probability*** *$p(E|F)$ is the probability that E occurs, given that F has occured (hence assuming that $|F| \neq 0$) :*

$$p(E|F) = \frac{|E \cap F|}{|F|}$$

Independent Events

Event E and F are called **independent** if $p(E|F) = p(E)$

Personal remark

this means that even if we know that F has occured the probability of E is still the same.

General Case: Finite $\boldsymbol{\Omega}$, arbitary $\boldsymbol{p(\omega)}$

Having equally likely outcomes is pretty rare in real life, juste take two dices and do the sum of the result and you will se that all the possible outcome doesn't have the same probability. In order to express those types of distribution we use the probability mass function:

Definition 4 ***Sample space*** *Ω: set of all possiblie outcomes*
Probability distribution (probability mass function) *p :*
A function $p : \Omega \to 1$ such that:

$$\sum_{\omega \in \Omega} p(\omega) = 1$$

If we sum up all the probablity it gives us 1.

muss function to a subset

Given $E \subset \Omega$ we can define the domain of the probability mass function p is extended to the power set of Ω :

$$p(E) = \sum_{\omega \in E} p(\omega)$$

2.3 Conditional probability and Independent Events

General form

The general form for the conditional probability is:

$$p(E|F) = \frac{p(E \cap F)}{p(F)}$$

for F such that $p(F) \neq 0$

Independet events

As before E and F are called independent if $p(E|F) = p(E)$, Equivalently, E and F are independent iff $p(E \cap F) = p(E)p(F)$.

Disjoin event

if E_1 and E_2 are disjoint event then:

$$p(E_1 \cup E_2) = p(E_1) + p(E_2)$$

Law of total probability

For any $F \subseteq \Omega$ and its complement F^c,

$$p(E) = p(E|F)p(F) + p(E|F^c)p(F^c)$$

which sounds very intuitive because by definition F and F^c are disjoint.

Generally

Theorem 1 *If Ω is the union of disjoint event $F_1, F_2, \dots, F_n$ then:*

$$p(E) = p(E|F_1)p(F_2) + p(E|F_2)p(F_2) + \cdots + p(E|F_n)p(F_n)$$

Proof

We prove the law of total probability for $\Omega = F \cup F^c$ (the general case follows straighforwardly)

$$\begin{aligned} p(E) &= p(\underbrace{(E \cap F) \cup (E \cap F^c)}_{\text{union of disjoint sets}}) \\ &= p(E \cap F) + p(E \cap F^c) \\ &= \frac{p(E \cap F)}{p(F)} p(F) + \frac{p(E \cap F^c)}{p(F^c)} p(F^c) \\ &= p(E|F)p(F) + p(E|F^c)p(F^c) \end{aligned}$$

Bays' Rule

Theorem 2

$$p(F|E) = \frac{p(E|F)p(F)}{p(E)}$$

Proof We use the definition of conditional probability to write $p(E \cap F)$ two ways and solve for $p(F|E)$:

$$p(F|E)p(E) = p(E \cap F) = p(E|F)p(F)$$

2.4 Random variable

Random variable

Definition 5 *A Random variable is a function X such as $X : \Omega \to \mathbb{R}$*

Probability distribution

p_x, $p_x(X = x)$ or $p_x(x)$ is the probability that $X = x$, i.e, the probability of the event

$$E = \{\omega \in \Omega : X(\omega) = x\}$$

Hence,

$$p_x(x) = \sum_{w \in E} p(\omega)$$

Example You rolle a dice.
if the outcome is 6, you receive 10CHF. Otherwise, you pay 1 CHF.

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$

$$\text{For each } \omega, p(\omega) = \frac{1}{6}$$

Then define:

$$X(\omega) = \begin{cases} 10, & \omega = 6 \\ -1, & \omega \in \{1, 2, 3, 4, 5\} \end{cases}$$

Hence, we have

$$p_x(X) = \begin{cases} \frac{1}{6}, & x = 10 \\ \frac{5}{6}, & x = -1 \end{cases}$$

2.4.1 Two random variables

Two random variables

Definition 6 *Let $X : \Omega \to \mathbb{R}$ and $Y : \Omega \to \mathbb{R}$ be two random variables. The probability of the event $E_{x,y} = \{w \in \Omega : X(\omega) = x$ and $Y(\omega) = y\}$ is:*

$$p_{x,y}(x, y) = \sum_{w \in E_{x,y}} p(\omega)$$

- p_x is called **marginal distribution** (of $p_{x,y}(x, y)$ with respect to x)
- p_y can be computed similarly

2.5 Expected Value

Expected value

Definition 7 *The expected value $\mathbb{E}[X]$ of a random variable $X : \Omega \to \mathbb{R}$ is :*

$$\mathbb{E}[X] = \sum_{\omega} X(\omega)p(\omega)$$

$$= \sum_{x} x p_x(x)$$

linearity

Expectation is a linear operation in the folowwing sence:
Let $X_1, X_2, \ldots, X_n$ be random variables and $\alpha_1, \alpha_2, \ldots, \alpha_n$ be scalars. Then:

$$\mathbb{E}\left[\sum_{i=1}^{n} X_i \alpha_i\right] = \sum_{i=1}^{n} \alpha \mathbb{E}[X_i]$$

Random variable and independecy

Two random variable X and Y are independent if and only if, for all realizations x and y:

$$p(\{X = x\} \cap \{Y = y\}) = p(\{X = x\})p(\{Y = y\})$$

Or, more concisely, iff

$$p_{x,y}(x, y) = p_x(x)p_y(y)$$

Generalization

Theorem 3 *Given n random variables, $X_1, \ldots, X_n$ are independent if and only if:*

$$p_{x_1,\ldots,x_n}(x_1, \ldots, x_n) = \prod_{i=1}^{n} p_{x_i}(x_i)$$

Condition probability

The conditional distrivution of Y given X is the function:

$$p_{x,y}(x|y) = \frac{p_{x,y}(x, y)}{p_x(x)}$$

Independent random variables

The folowig statement are equivalent to the statemant that X and Y are two indepedent random variables:

- $p_{x,y} = p_x p_y$
- $p_{y|x}(y|x) = p_y(y)$
- $p_{y|x}(y|x) = p_y(y)$ is not a function of x
- $p_{x|y}(x|y) = p_x(x)$

- $p_{x|y}(x|y)$ is not a function of y

Summary 1 • *Random Variable*

- *Probability distribution*
  - *Joint distribution of multiple variables*
  - *Marginal distribution*
  - *Conditional distribution*
- *Independence*

February 19, 2025 — **Lecture 2 : Source and entropy**

Expected value and operation The addition works well with Expectation such that

$$\mathbb{E}[X+Y] = \mathbb{E}[x] + \mathbb{E}[Y]$$

However, the product doesn't work well,

$$\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$$

if and only if X and Y are independent random variables.

2.6 Entropy

Introduction We communicate be revealing the value of sequence of variables that we call (**Symbols**), **Information**
In modern language, Hartley was saying that the value of a symbole provides information if and only if the symbol is a **random variable**.
How much information is carried by a symbol such as S?

- Suppose that $S \in \mathcal{A}$ is a symbol that can take $|\mathcal{A}|$ possible values
- The amount of information conveyed by n such symbol should be n times the informations conveyed
- there are $|\mathcal{A}|^n$ possible values for n symbols
- This suggests that $\log|\mathcal{A}|^n = n\log|\mathcal{A}|$ is the appropriate mesure for information

However, this approach doesn't works:

> *Example* Imagine having a town where there are 360 days and 5 rainy days, this leads to have only to possibilities, $|\mathcal{A}| = 2$ which make the quantity of information $\log_2 2 = 1$ bits. Which intiutively sounds kind of false, the forecast doesn't give us that information knowing that it is sunny $\frac{360}{365}$ % of the times, it is kind of excpected.

An article in 1948 from Shannon fixes the problem by defining **Entropy**

Definition the **uncertainty** or **entropy** $H(S)$ associated to a discrete random variable S:

Definition 8

$$H_b(S) = -\sum_{S \in supp(p_s)} p_s(s) \log_b p_s(s)$$

Where $supp(s) = \{s : p_s(s) > 0\}$.

Few comments

$$H_b(S) = -\sum_{S \in supp(p_s)} p_s(s) \log_b p_s(s)$$

- The condition $S \in supp(p_s)$ is needed because $\log_b p_s(s)$ is not define when $p_s(s) = 0$ this convention allows us to use the notation :

$$H_b(S) = -\sum_{s \in \mathcal{A}} p_s(\log_b p_s(s))$$

- The choice of b determines the unit, $b = 2$ is the **bit**

We also can see this as an "*average*" of $-\log_b p_s(S)$ which is:

$$H(S) = \mathbb{E}[-\log_b p_s(S)]$$

Example

A sequence of 4 decimal digits, s_1, s_2, s_3, s_4 representing the number to open Anne's lock can be senn as the output of a source S_1, S_2, S_3, S_4 with $S_i = \{0, \dots, 9\}$.
If Anne picks all digits at randm and indepedently, the all outcomes are equally likely:

$$p_{S_1,S_2,S_3,S_4}(S_1, S_2, S_3, S_4) = \frac{1}{10^4}$$

If we search the entropy of this we get:

$$H_2(S) = \log_2 |\mathcal{A}| = \log_2 10^4 \approx 13.3 \; bits$$

2.6.1 Information-Theory Inequality

Lemma (IT-Inequality)

lemme 1 *For a positive real number* r*,*

$$\log_b r \leq (r-1)\log_b(e)$$

with equality if and only if $r = 1$

This proof juste using the deriative

Entropy Bounds

Theorem 4 *The entropy of a discrete random variable $S \in \mathcal{A}$ satisfies:*

$$0 \leq H_b(S) \leq \log_b |\mathcal{A}|$$

With equality on the left if and only if $p_s(S) = 1$ and on the right if and only if $p_s(S) = \frac{1}{|\mathcal{A}|}$ for all s.

2.6.2 Random variables and Entropy

***n* random variable**

the formula for entropy can be expanded to any number of random variables. If X and Y are two discrete random variables, with (joint) probability distribution $p_{x,y}$ then:

$$H(X, Y) = -\sum_{(x,y)\in X\times Y} p_{x,y}(x, y) \log p_{x,y}(x, y)$$

1.4 of textbooks

Theorem 5 *Let $S_1, \ldots, S_n$ be discrete random variables. Then*

$$H(S_1, S_2, \ldots, S_n) \leq H(S_1) + H(S_2) + \cdots + H(S_n)$$

With equality if and only if $S_1, \ldots, S_n$ are indepedent.

February 25, 2025 — **Lecture 2 : suite**

Ex hat party 1950

- n men, all have the same hat
- they throw hats in a corner
- leaving, they randomly take a hat

Solution

$$\text{Let } R_i = \begin{cases} 1, & \text{if person} i \text{leaves with their own hat} \\ 0, & \text{otherwise} \end{cases}$$

Entropy

$$H_2(S) = \sum_i p(s) \log \frac{1}{2p(s)} \tag{2.1}$$

$$= \frac{1}{8} \log_2 \frac{8}{2} + \frac{1}{8} \log_2 8 \tag{2.2}$$

$$\approx \frac{1}{8} + \frac{1}{8} \cdot 3 \tag{2.3}$$

personal remark

We can see it as an average of ”surprise”.
Where the average is the randomness. (≈ 0.55)

2.6.3 Entropy bounds

Bound

$$0 \leq H_b(S) \leq \log_b \mathcal{A}$$

2.7 Source Coding Purpose

Source coding is often seen as a way to compress the source.
More generally, the foal of source coding is to efficiently describe how much information there is to a *file*

2.7.1 Setup

Setup The **encoder** is specified by: :

- the input alphabet$\mathcal{A}$ (the same as the source alphabet)
- the output alphabet $\mathcal{D}$(typically $\mathcal{D} = \{0, 1\}$);
- the codebook $\mathcal{C}$ Which consists of finite sequences over $\mathcal{D}$;
- By the one to one encoding map $\Gamma : \mathcal{A}^k \to \mathcal{C}$ where k is a positive integer.

For now, $k = 1$.

Example For each code, the encoding map Γ is specified in the following table: A mettre une image.

> *Example* Code C or B are uniquely decodable : (A mettre une image 106)

Prefix Free codes

Definition 10 *If no codeword is a prefix of another codeword, the code is said to be prefix free.*

> *Example* The codeword **01** is a prefix of **01**1.

- A prefix free code is always uniquely decodable
- A uniquely decodable code is **not necessarily** prefix free

> *A prefix code* A prefix free code is also called instantaneous code :
>
> - Think of phone numbers
> - Think about streaming: instantaneous codes minimize the decoding delay (for given codeword length)

Code for one random variable We start by considering codes that encode **one single random variable** $S \in \mathcal{A}$.
To encode a sequence $S_1, S_2, \ldots$ of random variables, we encode one random variable at a time.

Complete tree of a code Slide 113 screen.

Binary tree

- There is a root (the beginning)
- A vertex (another node)

- A **leaf** is the last vertex
- Which is like a (arbre généalogique)

Ternary Tree The same as a binary tree but with three children.

With/Without prefix slide 115.

Decoding tree

- Obtained from the complete tree by keeping only branches that form a codeword
- Useful to visualize the decoding process

Slide 116

2.7.2 Codeword length

- The codeword length is defined the obvious way:
- Example: *ct*

$\mathcal{A}$ Γ_B codeword lengths
a 0 1
b 10 2
c 110 3
d 1110 4 height

- We would like the average codeword length to be as small as possible.

2.7.3 Kraft McMillan

Part 1. Necessary condition for the code to be uniquely decodable

Theorem 6 *If a D-ary code is uniquely decodable then its codeword length $i_1, \dots, i_M$ satisfy*

$$D^{-l_1} + \cdots + D^{-l_M} \leq i$$

Kraft's inequality

Example For code O we have :

$$2^{-2} + 2^{-2} + 2^{-2} + 2^{-2} = 1$$

Recall Kraft McMillan

Theorem 7

Example A For code A we have $2^{-1} + 2^{-2} + 2^{-2} + 2^{-2} = 1.25 > 1$.
KRaft-McMillan's inequality is not fulfilled.
There exists no uniquely decodable code with those codeword lengths.

Proof of K-MM Part I We prove a slightly weaker result, namely that the codeword lengths of prefix free codes satisfy K-MM inequality.
Let $L = \max_i l_i$ be the complete tree's depth.

- There are D^L terminal leaves
- There are D^{L-l_i}
- No two codewords share a terminal leaf (The code is prefix free)
- Hence $D^{L-l_i} + D^{L-l_2} + \dots + D^{L-l_m} \leq D^L$

After dividing both sides by D^L we obtain Kraft's inequality:

$$D^{-l_1} + D^{-l_2} + \dots + D^{-l_M} \leq 1$$

Exercice What is the **converse** of Kraft McMillan part 1?
The **Converse** of Kraft McMillan part 1 is not true (Consider e.g. two codewords: 01 and 0101)
However, the following statement is almost as good :

> **Theorem 8** *If the positive integer $I_1, \dots, I_M$ satisfy Kraft's inequality for some positive integer D,then there exists a D-ary **prefix free code** (hence uniquely decodable) that has codewords*

This says that if the inequality is true, then we **can** find D such that there exists a binary prefix which makes it decodable **and** prefix free!

2.7.4 Important Consequence of Kraft McMillan

Part I

> **Theorem 9** *If a **D-ary code is uniquely decodable**, then its codeword length $I_1, \dots I_M$ satisfy Kraft's inequality :*
>
> $$D^{-l_1} + \dots + D^{-l_M} \leq 1$$

Part II

> **Theorem 10** *If the positive integer $l_1, \dots, l_M$ satisfy Kraft's inequality for some positive integer D, then there exists a D-ary **prefix free code** that has those codeword lengths.*

The Kraft McMillan theorem implies that any uniquely decodable code can be substituted by a prefix free code of the same codeword lengths.

Prefix free codes Our focus will be on prefix free codes. Reasons:

- No loss of optimality: codewords can be as short as for any uniquely decodable code;
- a prefix free codeword is recognized as soon as its last digit is seen:

- important, e.g. a phone number;
- advantageous to limit the decoding delay in, say streaming

Average Codeword length

- The typical use of a code is to encode a sequence of random variables
-

Example

$$\mathcal{A} = \{a, b, c, d\} \quad D = 2$$

Blackboard with table *cct* s ∈A
$\Gamma(s)$
$l(s)$
$p(s)$

a
0
1
0.05
b
10
2
0.05
c
110
3
0.1
d
1111
4
0.8

$$\mathcal{E}[\text{length}] = 0.05 + 1 + 0.05 \cdot 2$$

Definition 11 *Let $l(\Gamma(s))$ be the length of the codeword assiociated to $s \in \mathcal{A}$ The average codeword length is:*

$$L(S, R) = \sum_i p_s(s) i(\Gamma(s))$$

Units The unit of $L(S, \Gamma)$ are **code symbols**
When $D = 2$, the unit of $L(S, \Gamma)$ are bits.

Average codeword length: Lower Bound

Theorem 11 *Let $\Gamma : \mathcal{A} \to \mathcal{C}$ be the encoding map of a D-ary*

Proof We want to prove that:

$$
\begin{aligned}
H(s) - \sum_s p(s) l(s) \\
&= -\sum_s p(s) \log p(s) - \sum_s p(s) l(s) \\
&= -\sum_s p(s) \log p(s) - \sum_s p(s) \log 2^{l(s)} \\
&= -\sum_s p(s) \log (p(s) \cdot 2^{l(s)}) \leq \dots
\end{aligned}
$$

Therefore:

$$
\begin{aligned}
&= \sum_s p(s) \log(\frac{1}{p(s)} 2^{-l(s)}) \\
&\leq \sum_s p(s) \left(\frac{1}{p(s)} 2^{-l(s)} - 1 \right) \cdot C \\
&= (\sum_s 2^{-l(s)} - \sum_s p(s)) \cdot C \\
&\leq 0
\end{aligned}
$$

We know that the left side is less or equal to 1 because of the Kraft Inequality, therefore it is bounded.

February 26, 2025 — **Lecture 4 : Continue**

Key observation The right hand side of:

$$L(S, \Gamma) = \sum_{s \in \mathcal{A}} p(s) l(\Gamma(s))$$

$$H_D(S) = \sum_{s \in \mathcal{A}} p(s) \log_D \frac{1}{p_S(s)}$$

are identical if $l(\Gamma(s))$

- Unfortunately $l(\Gamma(s)) = \log_D \frac{1}{p_S(s)}$ is often not possible (not an integer)
- How about choosing

Theorem 12 • *For every random variable $S \in \mathcal{A}$*

Theorem

Theorem 13 *The average codeword length of a D-ary Shannon-Fano code for the random variable S fulfils:*

$$H_D(S) \leq L(S, \Gamma_{SF}) < H_D(S) + 1$$

Proof it suffices to prove the upper bound (we have already proved the lower bound)
First suppose that we could use $l_i = -\log p_i$. The average

length would be:

$$L(S,\Gamma) = \sum_i p_i l_i = \sum_i p_i(-\log_D p_i) = H_D(S)$$

Instead we use $l_i = \lceil -\log p_i \rceil < -\log p_1 + 1$

March 4, 2025 — Lecture 5 : Conditional Entropy

Key Idea Pack multiple symbols into " *supersymbols*"

- $(S_1, S_2, S_3, \dots, S_n)$
- Now, apply our Main result to such supersymbols

Theorem 14 *The average codeword-length of a uniquely decodable code Γ for S must satisfy:*

$$H_D(S_1, S_2, \dots, S_n) \leq L((S_1, S_2, \dots, S_n), \Gamma)$$

And there exists a uniquely decodable code Γ_{SF} satisfying:

$$L((S_1, S_2, \dots, S_n), \Gamma_{SF}) < H_D(S_1, S_2, \dots, S_n) + 1$$

Our Next Nugget Understand

Example Audio recording:

- We can easily anticipate the next image in a video, there

KEy(simple) Independent

Definition 12 *The source models a seuquence $S_1, S_2, \dots, S_n$ of n coin flips*
So $S_i \in \mathcal{A} = \{H, T\}$ where H stands for heat, T for tails.
$p_{S_i}(H) = p_{S_i}(T) = \frac{1}{2}$ for all $(s_1, S_2, \dots, S_n) \in \mathcal{A}^n$

Not independent

Definition 13 *The source models a sequence $S_1, S_2, \dots, S_n$ of weather conditions.*
So $S_i \in \mathcal{A} = \{S, R\}$, where S stands for sunny and R for rainy
The weather on the first day is uniformly distributed in $\mathcal{A}$.
For all other days, with probability $q = \frac{6}{7}$ the weather is as for the day before

Conditional Probability Recall how to determine the conditional probability:

$$p_{X|Y}(x \mid y) = \frac{p_{X,Y}(x,y)}{p_Y(y)}$$

It gives the probability of the event $X = x$, given that the event $Y = y$ has occured.
it is defined for all y for which $p_Y(y) > 0$

Remark There is good slide with good schema in slide 176-179

Conditional Expectation of X given $Y = y$

$$p_{X|Y}(\cdot \mid y)$$

is the probability distribution of the alphabet of X, juste like $p_x(\cdot)$

Definition 14 *The conditional expectation of X given $Y = y$ is defined as:*

$$\mathcal{E}[X \mid Y = y] = \sum_{x\in\mathcal{X}}$$

Conditional Entropy of X given $Y = y$

$p_{X|Y}(\cdot \mid y)$ is a probability distribution on the alphabet of X, juste like $p_X(\cdot)$ Every probability distribution has an entropy associated to it:

- $p_x(\cdot) \to H(X)$
- $p_{X|Y}(\cdot \mid y) \to H(X \mid Y = y)$

Definition 15 *The conditional entropy of X given $Y = y$ is defined as:*

$$H_D(X \mid Y = y) = -\sum_{x\in\mathcal{X}} p_{X|Y}($$

Example A faire

Entropy Bounds

Theorem 15 *The conditional entropy of a discrete random variable $X \in \mathcal{X}$ conditioned on $Y = y$ satisfies:*

$$0 \leq H_D(X \mid Y = y) \leq \log_D \mid \mathcal{X} \mid$$

With equality on the left iff $p_{X|Y}(x, y) = 1$ for some x, and with equality on the right iff $p_{X|Y}(x \mid y) = \frac{1}{|\mathcal{X}|}$

The proff is identical to our proof of the basic entropy bounds

Example

Question?
Do we also have the following entropy bound:

$$H_D(X \mid >= y) \overbrace{\leq}^{???} H_D(X)?$$

Answer: no.

Example (Or " *counterexample*" if better), Juste for ease of calculation, let us set $\delta = 0$ (but this is not necessary for the example to work). Then, we have:

$$H_D(X \mid Y = 0) h_D(\varepsilon) \text{ and } H_D(X \mid Y = 1) = 0$$

where $h_d(\cdot)$ is the binary entropy function (with $\log_D(\cdot)$). But we have:

$$H_D(X) = h_D(\frac{1-\varepsilon}{2})$$

Conditional entropy can either go up or down (if we give the answer the entropy is 0)

Conditional Entropy of X given Y

The most useful and impactful definition is the *average* conditional entropy of X given $Y = y$, averaged over all values of y under the marginal distribution $p_Y(y)$. Formally, we thus define:

Definition 16 *The conditional entropy X given Y is defined as:*

$$H_D(X \mid Y) = \sum_{y \in \mathcal{Y}} p_Y(y) \left(- \sum_{x \in \mathcal{X}} p_{X|Y}(x \mid y) \log_D p_{X|Y}(x \mid y) \right)$$

Example For the Bit flipper channel, we have;

$$H_D(X \mid Y) = p(Y = 0) H_D(X \mid Y = 0) + p(Y = 1) H_D(X \mid Y = 1)$$

We search now:

$$H(X \mid Y) = p(Y \text{ is Head}) H(XY \text{ is head}) + p(Y \text{ is Tail}) H(X \mid Y \text{ is tail}) = \frac{1}{2} \cdot$$

Conditional Entropy of X given Y

Theorem 16 *The conditional entropy of discrete random variable $X \in \mathcal{X}$ conditioned on Y satisfies:*

$$o \leq H_D(X \mid Y) \leq \log_D \mid \mathcal{X} \mid$$

With equality on the left iff for every y there exists and y such that $p_{X|Y}(x \mid y) = 1$ and with equality on the right iff $p_{X|Y}(x \mid y) = \frac{1}{|\mathcal{X}|}$ for all x and all y.

This follows directly from our bounds on $H_D(X \mid Y = y)$

Having $p_{X|Y}$

We know that $p(X \mid Y) = \frac{1}{|\mathcal{X}|}$ for all y.

$$\begin{aligned} p(x) &= \sum_{y \in \mathcal{Y}} p(y) p(x \mid y) \\ &= \sum_{y} p(y) \frac{1}{\mid \mathcal{X} \mid} \\ &= \frac{1}{\mid \mathcal{X} \mid} \cdot \sum_{y} \overbrace{p(y)}^{=1} \end{aligned}$$

Conditioning Reduces Entropy

The following bound is important and impactful (and also intuitively pleasing!)

Theorem 17 *For any two discrete random variables X and Y,*

$$H_D(X \mid Y) \leq H_D(X)$$

with equality iff X and Y are independent random variables

In words, **On average**, the uncertainty about X can only become smaller if we know Y.

As we have seen, this is not true point-wise: We may have $H_D(X \mid Y = y) > H_D(X)$ for some values of y.
It works only on average.

Proof

$$
\begin{aligned}
H(X \mid Y) - H(X) = & \\
&= \sum_y p(y)\left(-\sum_x p(x \mid y)\log p(x \mid y)\right) + \sum_x p(x)\log p(x) \\
&= \sum_{x,y} p(y)p(x \mid y)\log\frac{1}{p(x \mid y)} + \sum_{x,y} p(y \mid x)p(x)\log p(x) \\
&= \sum_{x,y} p(x,y)\log\frac{p(x)}{p(x \mid y)} \\
&\leq \sum_{x,y} p(x,y)\left(\frac{p(x)}{p(x \mid y} - 1\right)\cdot\log e \qquad = \\
&= \sum_{x,y}(p(x)p(y) - p(x,y))\log(e) \\
&= \left(\left(\sum_y p(x)p(y)\right) - \left(\sum_x p(x)p(y)\right)\right)
\end{aligned}
$$

Conditional Entropy of $f(x)$

Let X be an arbitrary random variable. Let $f(x)$ be a (deterministic) function of x.

$$H(f(x) \mid X) = 0$$

Proof

To find this conditional entropy:
Let $Y = f(x)$

$$p(y \mid y) = \begin{cases} 1, & y = f(x) \\ 0, & y \neq f(x) \end{cases}$$

the probability that y is $f(x)$ is only true if $f(x) = y$.
This implies that the entropy is equal to 0:

$$H(y \mid x) = 0$$

Conditioning reduced Entropy

A generalization of the previous bound is also interest to us:

Theorem 18 *For any three discrete random variables X, Y and Z,*

$$H_D(X \mid Y, Z) \leq H_D(X \mid Z)$$

With equality iff X and Y are conditionally independent random variables given Z (that is, if and only if $p(x, y \mid z) = p(x \mid z)p(y \mid z)$ for all x, y, z,

You can see it as make the Z fall which makes it $p(x, y) = p(x)p(y)$

Proof It is only mathematics:

$$\begin{aligned} H_D(X \mid Y, Z) - H_D(X \mid Z) &= \mathbb{E}\left[\log_D \frac{1}{p_{X|Y,Z}(X \mid Y, Z)}\right] + \mathbb{E}[\log_D p_{X|Z}(X \mid Z \\ &= \mathbb{E}\left[\log_D \frac{p_{X|Z}(X \mid Z)}{p_{X|Y}(X \mid Y, Z)}\right] \\ &= \mathbb{E}\left[\log_D \frac{p_{X|Z}(X \mid Z)p_{Y|Z}(Y \mid Z)p_Z(Z)}{niquesamere}\right. \end{aligned}$$

March 4, 2025 — **Lecture 6 : Conditional Entropy review**

Main definitions

We have here two mains definitions:
The entropy for for a "*case*" of a random variable:

$$H(X \mid Y = y) = -\sum_x p(X \mid y) \log p(X \mid y)$$

And, the conditional entropy on a random variable:

$$\begin{aligned} H(X \mid Y) &= \sum_y p(y) H(X \mid Y = y) \\ &= -\sum_y \sum_x p(x, y) \log p(x \mid y) \end{aligned}$$

The main thing to understand here is that $H(X \mid Y)$ is the *Generalization* of the first definition. It is all the possible values of Y together. This is why we sum up all possible value of y. The second way to write $H(X \mid Y)$ is like taking all the possible pairs together and calculating the entropy of each pairs.
Main Result: The main result behind this is:

$$\begin{aligned} 0 \leq H(X \mid Y = y) \leq \log \mid \mathcal{X} \mid \\ 0 \leq H(X \mid Y) \leq \log \mid \mathcal{X} \mid \end{aligned}$$

And the inequality:

$$\boxed{H(X \mid Y) \leq H(X)}$$

Conditional entropy of $f(x)$

Let X be an arbitrary random variable.
Let $f(x)$ be a (deterministic) function of x:

$$H(f(x) \mid x) = 0$$

For example:

$$X \in \{0,1,2,3\}$$
$$f(x) = X \mod 2$$

Which is:

$$f(x) = \begin{cases} 0 \text{ if x is even} \\ 1 \text{ if x is odd} \end{cases}$$

Then,

$$P(f(x) \mid X) = \begin{cases} 0, & \text{if } x = 0,2 \\ 1, & \text{if } x = 1,3 \end{cases}$$

If we now compute the entropy for $X = 0$ and $X = 1$ etc..., we get:

$$H(f(x) \mid X = 0) = 0$$
$$H(f(x) \mid X = 1) = 0$$
$$\vdots$$

Lisa rolls two dice

Lisa rolls two dice and annoucnes the sum L written as a two digit number. The alphabet of $L = L_1 L_2 = \{02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12\}$ Where the alphabet of $L_1 = \{0,1\}$ and the alphabet of $L_2 = \{0,1,2,\ldots,9\}$.
We are looking for the probability that $L_2 = 2$ knowing that $L_1 = 1$:

$$p_{L_2|L_2}(2 \mid 1)$$

What we are doing here is the joint distribution:

$$p_{L_2|L_1}(2 \mid 1) = \frac{P_{L_1,L_2}(1,2)}{P_{L_1}(1)} = \frac{\frac{1}{36}}{\frac{1}{6}} = \frac{1}{6}$$

After running over all possible values for (i,j), we obtain:

$L_2 = j$ $L_1 = i$	0	1	$p_{L_2}(j)$
0	0	$\frac{3}{36}$	$\frac{3}{36}$
1	0	$\frac{2}{36}$	$\frac{2}{36}$
2	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{2}{36}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$p_{L_i}(i)$	$\frac{5}{6}$	$\frac{1}{6}$	height

now you can do the same with conditional probability and then after computing all those value:

$$H(L_2 \mid L_1) = \frac{5}{6} \cdot 2.857 + \frac{1}{6} \cdot 1.459 = 2.624 \text{ bits}$$

Now we can observe that:

$$2.624 = H(L_2 \mid L_1) \leq H(L_2) = 3.22$$

Which says that on average, knowing something takes out some randomness

The chain rule for entropy

Recall that the joint entropy of two random variables X, Y is completely naturally defined as:

$$H_D(X,Y) = -\sum_x \sum_y p_{X,Y}(x,y) \log_D p_{X,Y}(x,y)$$

Or as seen earlier:

$$H_D(X,Y) = H_D(X) + H_D(Y)$$

Using the fact the $p_{X,Y}(x,y) = p_X(x) p_{Y|X}(y \mid x)$, we can write this as:

$$\begin{aligned}
H_D(X,Y) &= -\sum_x p_X(x) \left(\sum_Y p_{Y|X}(y \mid x) \log_D (p_X(x) p_{Y|X}(y \mid x)) \right) \\
&= -\sum_x p_X(x) \left(\sum_y p_{Y|X}(y \mid x) (\log_D p_X(x) + \log_D p_{Y|X}(y \mid x)) \right) \\
&= -\sum_x p_X(x) \left[\left(\sum_y p_{Y|X}(y \mid x) \log_D p_X(x) \right) + \left(\sum_y p_{Y|X}(y \mid x) \log_D p_{Y|X}(y \mid x) \right) \right] \\
&= H(X) + H(Y \mid X)
\end{aligned}$$

Theorem 19

$$H(X,Y) = H(X) + H(Y \mid X)$$

Professor remark

Firstly:

$$H(Y,X) = H(X,Y)$$

Which is either proved by:

$$\begin{aligned}
H(Y,X) &= H(Y) + H(X \mid Y) \\
&= H(X,Y)
\end{aligned}$$

The relation proved before in words it:
To find the joint entropy of two random variables, we can first calculate the entropy of one of the two, and then add to it the conditional entropy of the second, given the first.

The chain rule entropy

Theorem 20 *Let $S_1, \dots, S_n$ be discrete random variables. Then:*

$$H_D(S_1, S_2, \dots, S_n) = H_D(S_1) + H_D(S_2 \mid S_1) + \dots + H_D(S_n \mid S_1, \dots, S_{n-1})$$

The above result says that the uncertainty of a collection of random variables (in any order) is the uncertainty of the first, plus the uncertainty of the second when the first is known, plus the uncertainty of the third when the first two are know, etc...
Let us see how:

$$\begin{aligned}
&H(\underbrace{S_1, S_2, \dots, S_{n-1}}_{=Z}, S_n)\\
&= H(Z) + H(S_n \mid Z)\\
&= H(\underbrace{S_1, S_2, \dots, S_{n-2}}_{=Z'}, S_{n-1}) + H(S_n \mid S_1 \dots, S_{n-1})\\
&= H(Z') + H(S_{n-1} \mid Z') + H(S_n \mid S_{1,n-1}
\end{aligned}$$

Until we get $Z^{'\dots'} = S_1$.

Example Let X, Y, Z be discrete random variables. We have:

$$\begin{aligned}
H(X, Y, Z) &= H(X) + H(Y \mid X) + H(Z \mid X, Y)\\
&= H(X) + H(Z \mid X) + H(Y \mid X, Z)\\
&= H(Y) + H(X \mid Y) + H(Z \mid X, Y)\\
&= H(Y) + H(Z \mid Y) + H(X \mid Y, Z)\\
&= H(Z) + H(X \mid Z) + H(Y \mid X, Z)\\
&= H(Z) + H(Y \mid Z) + H(X \mid Y, Z)
\end{aligned}$$

Theorem 21 *Let $S_1, \dots, S_n$ be discrete random variables. Then:*

$$H(S_1, S_2, \dots, S_n) \leq H(S_1) + H(S_2) + \dots + H(S_n)$$

With equality iff, $S_1, \dots, S_n$ are independent

Proof

$$\begin{aligned}
H(S_1, S_2, S_3) &= H(S_1) + H(S_2 \mid S_1) + H(S_3 \mid S_1, S_2)\\
&\leq H(S_1) + H(S_2) + H(S_3)
\end{aligned}$$

Another way around

Sometimes it is convenient to compute the conditional entropy using the chain rule for entropies. For instance:

$$\boxed{H(X \mid Y) = H(X, Y) - H(Y)}$$

It can be useful to make it easier to compute $H(X \mid Y)$ because on the right side, it is only marginal entropies with $p \log p$ which are *"easy to compute"*

corollaire 1

$$H(X,Y) \geq H(X)$$
$$H(X,Y) \geq H(Y)$$

The above inequalities follow from the chain rule for entropies and the fact that entropy (condition or not) is nonnegative.

Example

From lisa rolls two dice:

$$\begin{aligned} H(L_1, L_2) &= 3.2744 \\ H(L_1) &= 0.6500 \\ H(L_2) &= 3.2188 \end{aligned}$$

We compute:

$$\begin{aligned} H(L_2 \mid L_1) &= H(L_1, L_2) - H(L_1) = 3.2744 - 0.6500 = 2.6254 \\ H(L_1 \mid L_2) &= H(L_1, L_2)_H(L_2) = 3.2744 - 3.2188 = 0.056 \end{aligned}$$

And verify that indeed:

$$\begin{aligned} H(L_1 \mid L_2) \leq H(L-1) \leq H(L_1, L_2) \\ H(L_2 \mid L_1) \leq H(L_2) \leq H(L_1, L_2) \end{aligned}$$

2.7.5 Random Processes

A.K.A Source models

Definition 17 *The source models a sequence $S_1, S_2, \ldots, S_n$ of n coin flips*
So $S_i \in \mathcal{A} = \{H, T\}$, where H stands for heads, T for tails, $i = 1, 2, \ldots, n$
$p_{S_i}(H) = p_{S_i}(T) = \frac{1}{2}$ for all i, and coin flips are independent.
Hence,

$$p_{S_1, S_2, \ldots, S_n}(S_1, S_2, \ldots, S_n) = \frac{1}{2^n}, \quad \forall (S_1, S_2, \ldots, S_n) \in \mathcal{A}^n$$

Definition 18 *The source models a sequence $S_1, S_2, \ldots, S_n$ of weather conditions.*
So $S_i \in \mathcal{A} = \{S, R\}$, where S stands for sunny and R for rainy, $i = 1, 2, \ldots, n$.
The weather on the first day is uniformly distributed in $\mathcal{A}$.
For all other days, with probability $q = \frac{6}{7}$ the weather is as for the day before

What we can see here that is the conditional probability, for example:

$$\begin{aligned} p(S_2 = \text{ sun} \mid S_1 = \text{ sun}) &= q \\ p(S_2 = \text{ rain} \mid S_1 = \text{ sun}) &= 1 - q \end{aligned}$$

However:

$$\begin{aligned} &p(S_3 = \text{sun} \mid S_1 = \text{sun}, S_2 = \text{sun}) \\ &= p(S_3 = \text{ sun} \mid S_2 = \text{ sun}) = q \end{aligned}$$

More generally:

$$P(S_n \mid S_1, S_2, \dots, S_{n-1}) = p(S_n \mid S_{n-1})$$

March 11, 2025 — **Lecture 7 : Entropy and algorithm**

Experience little play

- Think of something
- Ask yes or no question
- Find the answer

the game was called twenty questions in old U.S tv. We want to use entropy to understand this game.

Last Week

$$H_D(X) = H_D(P) - -\sum_x p(x) \log_D p(x)$$

We also saw those two bounds:

$$0 \leq H_D(X) \leq \log_D \mid \mathcal{A} \mid$$

Information is always about option, more options you have, more information (the first way to introduce *"entropy"*)
We also saw:

$$\begin{aligned} H(X \mid Y = y) = -\sum_x p(x \mid y) \log_D p(x \mid y) \\ H(X \mid Y = y) = -\sum_y \dots \end{aligned}$$

And we also saw that on average:

$$\begin{aligned} H(X \mid Y) \leq H(X) \\ H(X \mid Y, Z) \leq H(X \mid Y) \leq H(X) \end{aligned}$$

We also saw the chain rule:

$$\begin{aligned} &H(S_1, S_2, S_3, S_4) \\ &= H(S_2, S_4, S_1, S_3) \end{aligned}$$

The order in entropy doesn't matter,

$$= H(S_1) + H(S_2 \mid S_1) + H(S_3 \mid S_1, S_2) + H(S_4 \mid S_1, S_2, S_3)$$

An intresting way to use this, is if we combine the inequalities and the chain rule. The equality on the right sight is true if and only if X and Y are independent. there fore:

$$H(S_1, S_2, S_3, S_4) = H(S_1) + H(S_2) + H(S_3) + H(S_4)$$

this equality is true if and only if S_1, S_2, S_3, S_4 are independent.

The 20 question problem

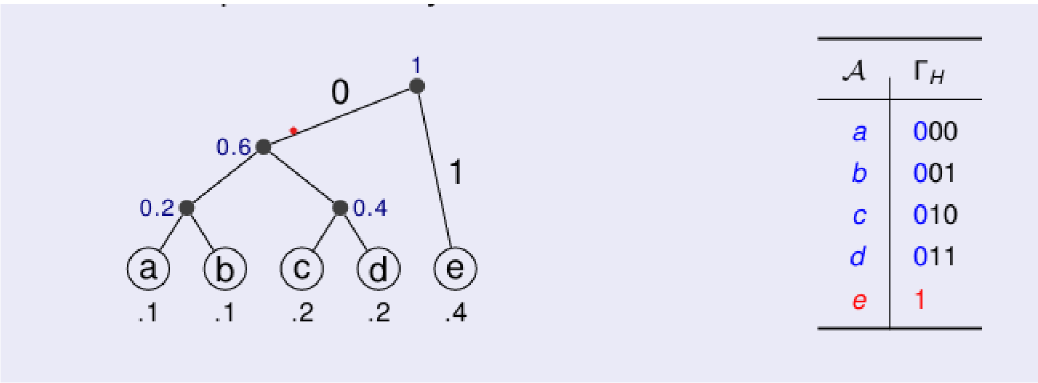
Let X be a random variable. What is the minimum number of "yes/no" question needed to identify X?, which question should be asked.

Solution

Let us consider a binary code Γ for $X \in \mathcal{X}$
Once Γ is fixed, we know $x \in \mathcal{X}$ if and only if we know the codeword $\Gamma(x)$.
The strategy consists in asking the ith question so as to obtein the ith bit of the codword $\Gamma(x)$.
The expected number of question $L(X, \Gamma)$, which is minimized if Γ is the encoding map of Huffman code

> *Example* Suppose that we know that $\mathcal{X} = \{$ cat, dog, pony$\}$, with:
>
> $$p(cat) = \frac{1}{2}$$
> $$p(dog) = \frac{1}{4}$$
> $$p(pony) = \frac{1}{4}$$
>
> We want to make it the best way, the question we should ask is:
>
> - is the animal a cat?
>
> X a b c d e height

We then do a Huffman tree:



$\mathcal{A}$	Γ_H
a	000
b	001
c	010
d	011
e	1

We know here that this, will be optimal

Optimality

We have seen that a prefix free code for $X \in \mathcal{X}$ leads to a querying strategy to find the realization of X.
Similarly, a deterministic querying strategy leads to a binary prefix-free code for X. Here is why:

- Before the first question we know that $x \in \mathcal{X}$
- Without loss of generality, the first question can be formulated in terms of "is $x \in \mathcal{A}$"? for some $\mathcal{A} \subset \mathcal{X}$, (The choice of $\mathcal{A}$ is determined from the strategy, that we fix once and for all)

- Is the answer is YES, the we know that $x \in \mathcal{A} \subset \mathcal{X}$. Otherwise $x \in A^c \subset \mathcal{X}$. Either way we have reduced the size of the set that contains x.
- We continue asking similar questions until the value of x is fully determined, the we stop.

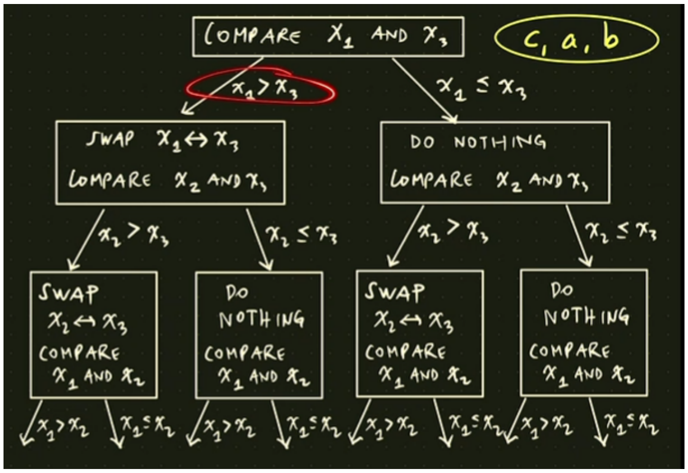
Here, the sequence of Yes or no answers is a binary codeword associated to x. The code obtained when we consider all possible values of x is a binary prefix-free code. Since the tree is prefix free, its averag codeword-length cannot be smaller than that of a Huffman code.

Sorting via pairwise comparisons

Given an **unsorted** List with n elements.
For example $l = [c, a, b]$ with $n = 3$
Repeat:

1. Select two position $1 \leq i \leq j \leq n$
2. Compare and swap:
   - If $x_i > x_j$
   - Then swap elements $x_i \iff x_j$
   - Else do nothing

One way to understands how it works:



The first observation:
The sequence of pairwise comparisons must identify the exact order of the unsorted list.
The second observation:
The sequence of pairwise comparisons in a uniquely decodable (actually, prefix-free) binary code for x.
There fore, we must have:

$$\mathbb{E}[\text{number of comparisons}] \geq H_2(X)$$

However what is the X? We see it as a random variable because we don't really know what the unsorted list is.
For example $n = 3$ we have $\mathcal{X} = \{abc, acb, bac, bca, cab, cba\}$ where $\mathcal{X}$ is the set of all permutations.
However what is $p(x)$? Here, we want to talk about our algorithm working

for all $p(x)$.

$$\begin{aligned} E &\geq max_{p(x)} H_2(X) = \log_2 \mid \mathcal{X} \mid \\ &= \log_2 n! \end{aligned}$$

We already know a bounds on factorial:

$$\frac{n^n}{e^{n-1}} \leq n! \leq \frac{n^{n+1}}{e^{n-1}}$$

Therefore:

$$\begin{aligned} H_2(x) &\approx \log_2 \frac{n^n}{e^{n-1}} \\ &= n \log_2 n - (n-1) \log_2 e \end{aligned}$$

Which is *"dominated"* by $n \log_2 n$

Billard Balls

There are 14 billards balls numbered as shown:

Among balls $1 - 13$, at most one **could** be heavier/lighter than the others. What is the minimum number of weightings to simultaneously determine:

- If one ball is different
- if there is such a ball which one,
- And whether the different ball is heavier/lighter

Here we want to use entropy to solve this problem. The goal here is to associated the number of weightings to code. The goal is to see it as a tree.

The steps of picking two sets is *"mandatory"* we have to pick two sets in order to compare something, and in order to compare something, you have to compare something...

From this comparisons, there will be three possibilities. with three possibilites, We are specifying a Ternary code. The issue here is that we are losing information, yes we only get a binary tree however we wouldn't be able to have the same amount of information as with a ternary tree.

What we are saying here is, with any strategy to solve this problem **can** be written in this way. Hence we can read this tree as a ternary code.

*But a code for **What**?*

What are we finding with this code?

A code for X:

- $X = 0$: all balls are equals
- $X = +1$: ball 1 is heavier
- $\vdots$

- $X = +13$ ball 13 is heavier
- $X = -1$ balle 1 is lighter
- $\vdots$
- $X = -13$ ball 13 is lighter

Then we know that $\mid \mathcal{X} \mid = 27$. This is one way to answers those question.

1. If $X = 0$ or not (then there is or not a different ball)
2. Then $\mid X \mid$ gives us the information
3. the sign of X if the ball is heavier or lighter

Observation The number of weighings is equal to the length of the ternary codeword

Then:

Theorem 22

$$\mathbb{E}[number\ of\ weighings] \geq H_3(X)$$

It has to be three by the way the problem is stated. The code is ternary **Therefore** the base for the entropy is 3.
Moreover, our strategy must work **irrespective** of the probability distribution of X.
We can also see:

Theorem 23

$$\mathbb{E}[number\ of\ weighings] \geq max_{p(x)}(H_3(X))$$

Where in our example gives us:

$$\log_3 27 = 3$$

It doesn't need the be an integer it is only the professor that choose on purpose to make it clean

But does there indeed exists such a code

FACT:

Entropy does **not** guarantee the existence of such a strategy

Entropy serves as a lower bound and **not** the best way to do it.

But can what if?
Let us suppose it exists! Then entropy tells us a few basic facts.

Fact 1 **if** 3 weighings S_1, S_2, S_3 uniquely specify X, Then we **must have**:

$$H_3(X) = H_3(S_1, S_2, S_3)$$

Proof

$$\begin{aligned} H(X, S_1, S_2, S_3) &= H(X) + \overbrace{H(S_1, S_2, S_3 \mid X)}^{=0} \\ &= H(S_1, S_2, S_3) + \overbrace{H(X \mid S_1, S_2, S_3)}^{=0} \end{aligned}$$

It is true because if we know S_1, S_2, S_3 then we know all X then the entropy of 0.
For $H(X \mid S_1, S_2, S_3)$, because S_1, S_2, S_3 uniquely specify X then knowing them implies that this entropy is o.

Fact 2 **If** 3 weighings S_1, S_2, S_3 uniquely specify X, then we must have:

- S_1, S_2, S_3 uniformly distributed
- S_1, S_2, S_3 independent

Proof

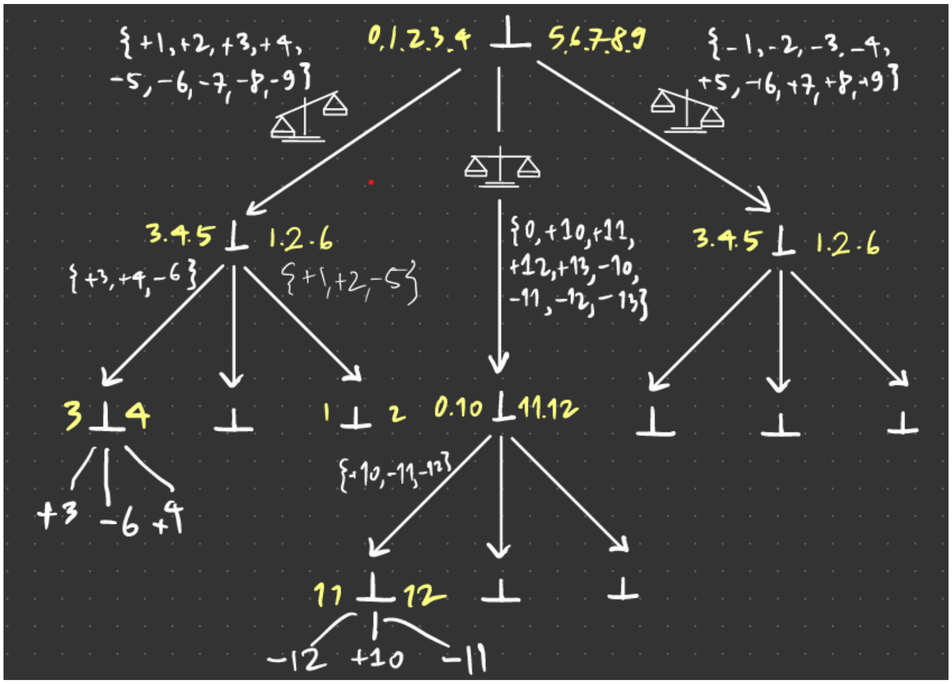
$$H_3(S_1, S_2, S_3) = 3$$

This is a *must.*
But also:

$$\begin{aligned} H_3(S_1) + H(S_2 \mid S_1) + H(S_3 \mid S_1, S_2) &\leq H_3(S_1) + H(S_2) + H(S_3) \\ &\leq \log_3 3 + \log_3 3 + \log_3 3 \end{aligned}$$

Where it is an equality if and only if the distribution is uniform and independent.

Example Let's see how to actually find a way to ask those question:



March 12, 2025 — **Lecture 8 : Prediction, learning, and Cross-Entropy-Loss**

Billard Balls Can we use the 20 questions approach to solve the 14 bullars riddle?

> *Answer* No, because the kind of questions that we can ”ask”, when wa are weighing, is quite limited.
> For instance, the first question cannot be ”is 1 or 2 heavy?”.

Strategies But is there a strategy that requires only 3 weighings?
From source compression, we can establish the following facts?

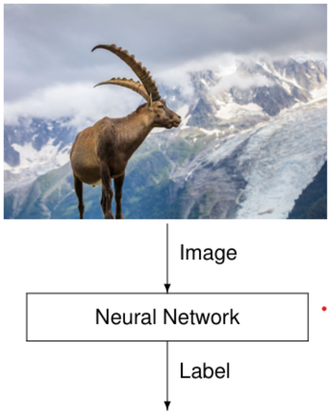
- For each weighings, the three outcomes must be equally likely
- The weighings must be independent of each other

It is because we carefully selected the numbers (alphabet size of 27; each weighing has 3 possible outcomes) that there is a strategy that exactly matches the entropy lower bound 3 weighings. If you change the numbers, it will not generally be true that there is a strategy that *exactly* matches the lower bound.

2.7.6 Prediction, Learning and cross-Entropy Loss

The goal here is to change the way to use entropy, entropy has always be seen as something that *means* something, a lower bound, a quantity of information. Here we will use it to do calculation juste like a *tool*.

Example



Label	*Probability*
Ibex	0.98
Kangaroo	0.005
Lynx	0.002
Wombat	0.002
Dog	0.001
Cat	0.001
Turtle	0.001
Dolphin	0.001
Elephant	0.001
Kookaburra	0.001
Other	0.005

There weren’t probability at this time in the slide so imagine without it

The question we want to ask is, *”Is our neural network performing well”*

- Given an image $\mathcal{X}$
- Our machine (Neural network)
- Outputs $Q(x)$
- The label: Label(x)

Zero-one loss

$$\begin{aligned} &L\{Q(x) \neq \text{Label}(x)\} \\ &= \begin{cases} 1 \text{ if } Q(x) \neq \text{Label}(x) \\ 0 \text{ if } Q(x) = \text{Label}(x) \end{cases} \end{aligned}$$

Given a lot of image, we want to have a **Classification error**:

$$\frac{\sum_{\mathcal{X}} L\{Q(x) \neq \text{Label}(x)\}}{\text{number of images}}$$

Is the function of mis-labeled images.

Pros and Cons

Pros

- Very intuitive
- Interpretable

Cons

- Not differentiable

With probability

Our neural network produces:

$$Q(\text{label} \mid \text{image})$$

The true label distribution is:

$$P_{true}(\text{label} \mid \text{image}) = \begin{cases} 1, & \text{correct label} \\ 0, & \text{wrong label} \end{cases}$$

(We are assuming for simplicity that for each image, there is a single correct label).

- Ideally, we would like:

$$Q(\text{label} \mid \text{image}) = P_{true}(\text{label} \mid \text{image}) \ \ \forall \text{pairs}$$

However this is only a dream

- Instead, people like to consider **cross entropy loss**
- that is, we wish ou $Q(\text{label}|\text{image})$ to **minimize**

$$\begin{aligned} &L(P_{true}(\text{label} \mid \text{image}), Q(\text{label} \mid \text{image}) \\ &= -\sum_{\text{label}} P_{true}(\text{label} \mid \text{image}) \log_D Q(\text{label} \mid \text{image}) \end{aligned}$$

- Given training data (image, label), for $i = 1, 2, \ldots, n$ we select $Q($ label $\mid$ image$)$ to minimize the cross entropy loss.

Cross entropy loss

$$L(P, Q) = -\sum_{y} P(y) \log_D Q(y)$$

Where

- P is the true distribution
- Q is our approximation (via neural network)

Why is it popular?

- Good properties for training with ”gradient descent” in certain standard architectures.
- Theoretical properties.

A (very) simple neural network

Takes a screen of the blackboard

- it transform the image into a vector
- Then takes is through the weighs w_i all the way to d
- the we take it through the soft max which is two functions:

$$Q(o \mid x) = \frac{e^{z_0}}{e^{z_0} + e^{z_1}}$$
$$Q(1 \mid x) = \frac{e^{z_1}}{e^{z_0} + e^{z_1}}$$

The goal is given a lot of training data, we want to select the w_0, b_0, w_1, b_1 such at to minimize the total cross entropy loss.

For a single image $\boldsymbol{\mathcal{X}}$

because why is juste binary we use:

Total Loss

$$L_{total}(w_o, b_o, w_1, b_1) = -\sum_{i=1}^{k} \log \frac{e^{x_i w_0 + b_0}}{e^{x_i w_0 + b_0} + e^{x_i \cdot w_1 + b_1}} - \sum_{i=k+1}^{n} \log \frac{e^{w_1 k_i + b_1}}{e^{w_0 x_i + b_0} + e^{w_1 x_i + b_1}}$$

Cross entropy loss

Cross entropy loss:

$$L(P, Q) = -\sum_{y} P(y) \log_D Q(y)$$

Theorem 24 *For a fixed probability distribution P, the minimum:*

$$min_Q L(P, Q)$$

Is attained if and only if we selected $Q^* = P$ in this case,

$$L(P, Q^*) = L(P, P) = H(P)$$

Where $H(P)$ is the entropy of the probability distribution P

Proof The proof, which will be done in class, uses once again the ”IT inequality”.

The theorem is saying this:

$$H(P) \leq L(P,Q)$$

With equality in one case which is $P = Q$.

$$\begin{aligned} H(P) - L(P,Q) &\leq 0 \\ -\sum_y P(y) \log P(y) + \sum_y P(y) \log Q(y) &\leq 0 \\ = sum_y P(y) \log \frac{Q(y)}{P(y)} \leq \sum_y P(y) \left[\frac{Q(y)}{P(y)} - 1 \right] \log(e) & \\ = \sum_y (Q(y) - P(y)) \log(e) & \\ = 0 & \end{aligned}$$

Note

We don't see it in AICC II but let's introduce the notion: **KL-Divergence** (aka KL distance):

$$D_{kl}(p \mid\mid k) = \sum_y p(y) \log \frac{P(y)}{Q(y)}$$

- Fact 1:
  $D_{kl}(P \mid\mid Q) \geq 0$ with equality iff $P = Q$ (this is just the proof seen earlier

2.8 Summary of chapter 1

Entropy

$$H_D(X) = -\sum_x p(x) \log_D p(x)$$

For $D = 2$, we simply write $H(X)$ and we all the units bits.
Entropy has many useful properties, including:

- $0 \leq H_D(X) \leq \log_D \mid \mathcal{X} \mid$
- $H_D(X \mid Y) \leq H_D(X)$ with equality if and only if X and Y are independent
- $H_D(X,Y) = H_D(X) + H_D(Y \mid X)$

Data Compression

- Every uniquely decodable binary code must use at least $H(X)$ bits per symbol on average
- There exists a binary code that uses between $H(X)$ and $H(X) + 1$ bits per symbol on average
- Hence, for a source string of length n:
  - Every uniquely decodable binary code must use at least $H(S_1, S_2,$

Models

Coin Flip The coin flip is not convertible, With a file of result, there is no way to compress the file

Sunny Rainy Here, the entropy, is not 1 then we are able to compress the file here.
This is the first view of mark of model.
Given $S_1, S_2, S_3, \ldots$, Are S_1, S_3 independent?

$$\begin{aligned} p(S_1, S_3) &= \sum_{S_2} p(S_1, S_2, S_3) \\ &= \sum_{S_2} p(S_1)p(S_2 \mid S_1)p(S_3 \mid S_2) \end{aligned}$$

Entropy and algorithm

We explored examples where entropy can give a lower bound on algorithmic performance.

- Example: in search-type problems, give a lower bound on the minimum number of necessary queries.

Cross-Entropy Loss

- Machine (e.g., Neural Network) outputs a distribution $Q(y)$ over all possible labels
- Cross entropy loss: Select $Q(y)$ to minimize:

$$L(P, Q) = -\sum_{y} P(y) \log_D Q(y)$$

March 18, 2025 — Lecture 9 : Introduction to cryptography

Chapter 3

Cryptography

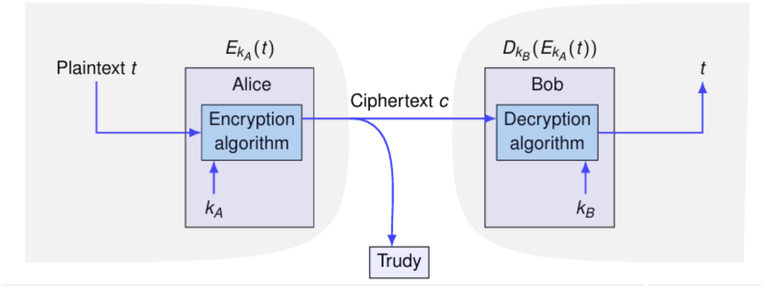
3.1 One-Time pad, Perfect Secrecy, Public-Key

Why cryptography

cryptography gives us the tools to:

- authenticate the sender and the receiver
- verify the integrity of the message
- keep the message confidential

Basic setup for condidentiality



Here Alice want to sent the plaintext t to Bob:

- She encrypts t using her key k_A. Theresult is the ciphertext $c = E_{k_A}(t)$
- She sends c to Bob over a public channel
- Bob decrypts c using his key k_B. The result is $D_{k_B}(E_{k_A}(t))t$
- For Trudy, it is nearly impossible to recover t from c without knowing k_B

Basic Terminology

- pleintext, ciphertext (also called cryptogram), key, encrypter, decrypter
- cryptography: the art of composing cryptograms
- cryptanalysis the art of breaking cryptograms
- a cryptanalyst has broken the system when he cann quickly determine the plaintext from the cryptogram, no matter what key is used
- attacker: same as cryptanalyst

Ancient cryptography

Caesar's cipher

Suppose that we are using the English alphabet augmented by a few special characters, "space", "comma", and "period". An alphabet of 29 characters, represented by the integers $0, 1, \ldots, 28$

- The key k is an integer between 0 and 28, known to Alice and Bob and to nobody else.
- The encryption algorithm substitues the i-th letter of the alphabet with the $(i + k)th$ letter ($\mod 29$)
- The decryption algorithm substitues the j-th letter with the $(j - k)$-th $\mod 29$

Therefore, here the secrecy of the message rely only on the secrecy of the algorithm.

Various attacks possible

We distinguish between the following attacks:

- **ciphertext-only**: one or more cryptograms available to the cryptanalyst know to have been encrypted with the same key
- **known plaintext**: the cryptanalyst has one or more plaintext and the resulting cryptograms, know to have been encrypted with the same key
- **chosen plaintext** for any plaintext that he requires, the cryptanalyst can obtain the cryptogram under the same key

Ideally, a cryptographic system should be secure against a chosen plaintext attack, At the very least, it should be secure against a cipthertext-only attack.

However with a computer, the key can easily be found using the letter-frequency attack. The question now is how to make the letter frequency attack unfruitful?

Example (Vigenère's cipher with an n-length key

- Chosen plaintext attack: encode the same letter until you have the $n-$length key
- **known plaintext attack** compare input/output until you have the n-length key
- **ciphertext-only attack**
  - brute force approach: try all 29^n keys is you know n
    - for $n = 21$ the number of key is 5.13^{30}
    - for $n = 100$, the number of keys is 1.73^{146}
  - If you know n, you can partition input/output into n parts, each of which is a Caesar cipher with its own key
  - Letter frequency approach: effective if the plaintext-length to key-length ration is sufficiently large

The one-time pad

Preliminary assumptions

- The plaintext t, the key k and the cryptogram c are n-length binary sequences over the alphabet $\mathcal{A} = \{0, 1\}$
- The key k, is produced by selecting each bit independently and with uniform distribution
- Alice and Bob use a private chanel to exchange the key ahead of time

Encryption

$$c = t \oplus k$$

Decryption

$$c \oplus k = (t \oplus k) \oplus k = t \oplus (k \oplus k) = t$$

Perfect secrecy

Definition 19 *A cryptosystem has* ***perfect secrecy*** *if the plaintext T and the cryptogram C are statistically independent*

Perfect secrecy is the ultimate kind of security against a ciphertext-only attack: The attacker cannot do better than guessing the plaintext T

Perfect secrecy of the one time pad

- The n-length key k is selected at random (uniform distribution ober $\{0, 1\}^n$)
- The key k and the message t are selected independently
- The ciphertext is $c = t \oplus k$

$$p_{C|T}(c \mid t) = p_{K|T}(c \ominus t \mid t) = p_K(c \ominus t) = \frac{1}{2^n}$$

Hence C and T are independent: knowledge of C is useless in guessing T

Weakness of the one time pad

Example

A cryptanalyst that has the plaintext t and the corresponding cryptogram c immediately gets the key:

$$k = c \ominus t$$

Hence the pad (the key) should be used only once

Pros and Cons of one time pad

Pros

- Very simple algorithm
- as secure as it gets against a ciphertext-only attack and key used one
- of instructional value to prove that the perfect secrecy is possible

Cons

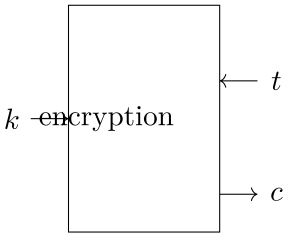
- The key is as long as the plaintext (this is fundamental, see later)

- The key needs to be exchanged ahead of time over a private chanel
- a ciphertext-only attack can break the system if the key is used twice (see homework)
- a known plaintext attack reveals the key

The ”one-time pad” has been used extensively in diplomatic and espionage circles

Perfect secrecy requires high entropy keys

The following theorem makes no assumption on the encryption algorithm:



Theorem 25 *Perfect secrecy implies:*

$$H(T) \leq H(K)$$

Proof Perfect secrecy $H(T) = H(T \mid C)$ and decodability ($H(T \mid K, C) = 0$) imply:

$$\begin{aligned} H(T) &= H(T \mid C) \\ &\leq H(T, K \mid C) \\ &= H(K \mid C) + H(T \mid K, C) \\ &= H(K \mid C) \\ &\leq H(K) \end{aligned}$$

Given T, K, C we search for the entropy:

$$\begin{aligned} H(t,k,c) &= H(c) + \overbrace{H(t \mid c)}^{=H(t)} + H(k \mid t, c) \\ &= H(c) + H(k \mid c) + \underbrace{H(t \mid k, c)}_{=0} \\ &\implies H(t) + H(k, \mid t, c) \leq h(k \mid c) \end{aligned}$$

This implies:

$$H(t) + \underbrace{H(k \mid t, c)}_{\geq 0} \leq H(k)$$

Because if we take out the *”Knowing c”* we cannot have some bigger. Therefore:

$$H(t) \leq H(k)$$

Entropy plays a key role also in cryptography

March 19, 2025 — **Lecture 10 : Encryption?**

Exercice Determine the minimum average length of the binary key for a cryptosystem that has the following characteristics

- the message is an uncompressible binary string of length n
- the system achieves perfect secrecy

Solution

- $H(T)$ must be essentially n bits (otherwise further compression is possible
- Perfect secrecy requires $H(T) \leq H(K)$
- hence $H(K)$ is at least n
- The average blocklength of the binary key is at least n bits

Symettric-Key crypto Systems: key-distribution problem A symmetric-key cryptosystem is one for which both ends use the same key ($k_A = k_B = k$). All example considered so fare rely on a symmetric key There exists fast (ans secure) symmetric key cryptosystems, but:

- Anybody that has the key can encrypt and/or decrypt
- The key cannot be sent over an insecure channel
- In an n user network, each user needs $n-1$ keys to communicate privately with every other user. Key distribution is a problem as it hat to be done over a secure channel. And keys have to be changed frequently
- We have a real problem (the first 6min. and 20 secs of:

`https://www.youtube.com/watch?v=YEBfamv-_do`

However, is there a way to distribute keys over a pubic channel?

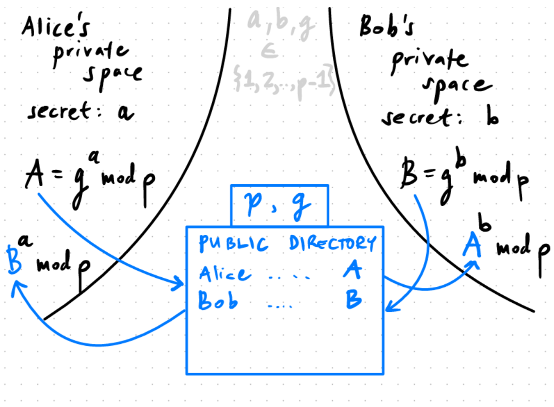
Solution In 1976, Diffie and Hellman came up with a solution.

Example You pick a number $p = 7$ which is a prime $\mathcal{A} = \{0, 1, 2, 3, 4, 5, 6\}$. We then take $g = 3$

i	g^i mod 7
0	
1	3
2	2
3	6
4	4
5	5
6	1

The gives us a permutation. We have to be careful with choosing the g because for example if we take $g = 2$, and take g^1 and g^4 the result is 2 which leads that we don't have a permutations

How does it is seen:



We have a public directory like a phone book where everyone has access. Alice pick a random a. Then she takes a public $A = g^a \mod p$ which we will be written in the public directory. Then Bob do the same thing.
When Alice and Bob want to have an interaction. Alice will look for Bob in the phonebook, take the B in her private space, take the result of $B^a \mod p$ (she is the only one to know a). Bob do the same thing: $A^b \mod p$
At this point

- Alice has $B^a \mod p$

$$B^a = (g^b \mod p)^a \mod p$$

- Bob has $A^b \mod p$

$$A^b = (g^a \mod p)^b \mod p$$

Fact

Theorem 26 *For all $x, y, m \in \mathbb{Z}$:*

$$[(x \mod m) \cdot (y \mod m)] \mod m = xy \mod m$$

We then get $B^a = g^{ab} \mod p$ and $A^b = g^{ab} \mod p$ We juste need to find the inverse of $g^i \mod p$ Which is a discrete logarithm problem. It is very hard to find the discrete logarithm. The gives the secrecy of this encryption. The secrecy is only here because it is hard to compute this inverse

Secrecy

But can't anybody generate this key?
We all know g and we know that $A = g^a \mod p$ we juste has to find

Eve wants to listen

Assuming that the cryptosystem used by Bob and Alise is secure, the best option for Eve is to find the key k.
She know p, g, A and B.
In generale, there seems to be no better way than finding the number a for which $g^a = A$, and the comput $k = B^a$
This is a problem. Let us check out some number. Suppose

p is a 2048 bit number. (it must be prime, but let us neglect this and assume $p = 2^{2048}$ How long does it take to compute:

$$2 \log_2 p = 4096$$

Multiplications to performs $a \to g^a$ (called discrete exponentiation). With a computer that performs 10^{10} multiplications per seconds, the exponentiation is done seamlessly.
It takes roughly:

$$\exp\left(\left(\frac{64}{9}\right)^{\frac{1}{3}} (\ln p)^{\frac{1}{3}} \ln \ln p)^{\frac{2}{3}}\right) \approx 10^{35}$$

Mutliplication to perform $g^a \to a$ (called discrete logarithm to the base g. With the same computer, it takes about 10^{25} seconds, which is about $7 \cdot 10^7$ times the ages of the earth.

Conclusion Diffie and Hellman's public key distribution scheme is clever, efficient, and it seems to be secure.

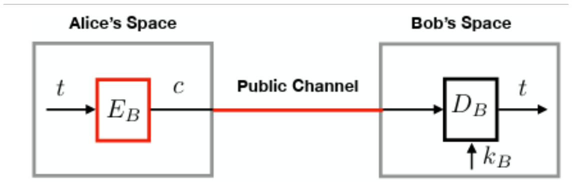
Paradigm shift The issue with the security is the computation. For example in maybe a couple of years the quantum computer will be able to compute this discrete algorithm very fast. The DG system would instanlty become insecure.
To the constrast, perfect secrecy offers provable security even when the enemy has infinite time and computing power. Most cryptographic systems rely on computational security

One way function Discrete exponentiation is an example of a one way function: a function for which a fast algorithm exists one way but not in the other way. The case for the discrete logarithm is that there is a way back but it is very slow. However, the best case would be that there is not way back.

Example IF a computer were to save user's names and passwords, a sstem manager would have access to both.
This is not the case if the operating system stores, along the name a one-way function f of our password.(The password itself is never stored)

Trapdoor One-way function A tropdoor one way function is a one-way function with an extra feature called the trapdoor information: with this information, the hard-to-carry out inverse computation becomes easy.
Diffie and Hellmann realized that with such a tool the key distribution problem would diseappear

Asymetric cryptography Suppose that Allice wants to send private information to Bob
Bob has a trapdoor one-way function, implemented by an algorithm E_B that he publsishes in a open directory
He is the only one who has the trapdoor information k_B. Hence he has the algorithm D_B that implements the inverse function.
Alice and Bob no longer need a shared key:

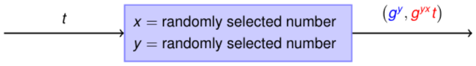


ElGammal's trapdoor one way function

Setup

- Fix a large prime number p. Hereafter all the numbers are in $\{0, 1, \ldots, p1\}$ and arithmetic is modulo p (mor on it later).
- Pick a generator g
- Pick randomly selected numbers x and y. Unlike p and g,x, and y are kept secret.

- Here is a trapdoor one-way function, with **trapdoor information** x.


t
x = randomly selected number
y = randomly selected number
$(g^{y}, g^{yx}t)$

Given the trapdoor information x, we can invert the function as follows: Compute the inverse of $g(^{y})^{x} = g^{xy}$, multiply the result with $g^{xy}t$. The result is t.

More concretly

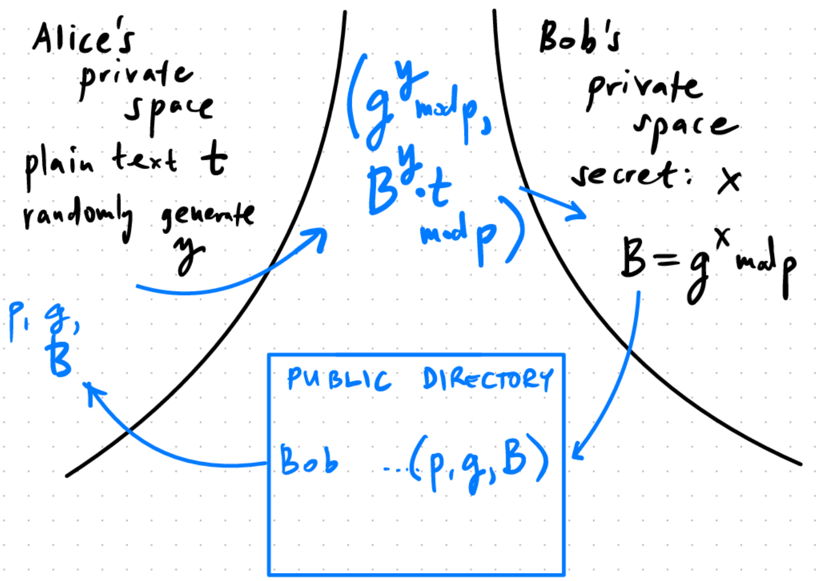
Alice has:

- t = plaintex, t
- y = random number, y

Bob:

- x = random number x.

The scheme look like this:



Bob sends g^x to Alice, then Alice sends the cryptogram $(g^y, g^{xy}t)$ to Bob

Note: x and y are transaction specific

March 25, 2025 — **Lecture 11 : Number Theory**

3.2 Number theory

Introduction In the digital world, the information is represented by the elements of a finite set, and we should be able to do math with them. Which means that the finite set should be a finite field. Our bigger goal of the next few lectures is to develop the tools to understand when and how we can turn a finite set into a finite field.

Operation with integers Within $\mathbb{Z}$ (the set of integers) we can

- add, subtract, multiply
- but not divide $\frac{7}{2}$ is not an integer
- What comes closest to the (regular) division is the euclidean division

Euclidiean division

Example For example if we take 7 divided by 2 this is equal to 3.5. (because $2 \cdot 3.5 = 7$)
But if we take 25 divided by $4 = 6.25$ but,

$$25 \bmod 4 = 1$$

Which leads to $25 = 4 \cdot 6 + 1$.

the goal here is to find that future “inverse” in the integer world.

The division algorithm

Given integers a (the dividend) and m the divisor:

$$a = mq + r, \quad 0 \leq r <\mid m \mid$$

The computation of q and r as above is called euclidean division

euclidean division in mainstream programming languages

In C/C++/Java/ Python we use the operator % to compute r as follows:
if a and m are both positive, then $r = \%m$
If one or the other or both are negative, different languages behave differently, but the general rule is:

- if $a\%m$ is nonnegative, then $r = a\%m$
- if $a\%m$ is **negative**, then $r = a\%m + m$

Congruence

Sometimes we are interested in knowing if two numbers have the same remainder when divided by m.

Definition 20 *Two integers a and b are said to be* ***congruent modulo m*** *, denoted:*

$$a \equiv b (\bmod m)$$

if $m \mid a - b$. (or m divided $a - b$)

Note

Do not confuse the **relation** $a \equiv b \bmod m$ and the **function** $a \to a \bmod m$.

Some laws that can be useful

- $a \equiv b \bmod m$
- $(a - b) \bmod m = 0$
- $a \bmod m = b \bmod m$

Congruence is an equivalence relation

A binary relation $\sim$ on a set is an **equivalence relation** if and only if the following three axioms are satisfied:

- $a \sim a$ (reflexivity)
- if $a \sim b$ then $b \sim a$ (symmetry)
- if $a \sim b$ and $b \sim c$ then $a \sim c$ (transitivity)

Substriture $a \sim a$ with $a \equiv a \mod m$ etc..., to see that congruene is an equivalence relation
One of the consequences is that we can form equivalence classes and we can work with one representative of each class (this will become useful later.)

Equivalence classes

An equivalence relation $\sim$ breaks A into disjoint sets, called **equivalence classes**.

It is like every element that "have the same relation" as a:

$$[a] = \{x \in A \mid x \sim a\}$$

This will be very useful to works with $\mathbb{Z}/m\mathbb{Z}$.

Modulo

Theorem 27 *If:*

$$\begin{aligned} a &\equiv a' \mod m \\ b &\equiv b' \mod m \end{aligned}$$

then:

$$\begin{aligned} a + b &\equiv a' + b' \mod m \\ ab &\equiv a'b' \mod m \\ a^n &\equiv (a')^n \mod m \end{aligned}$$

In particular, if $a' = (a \mod m)$ and $b' = b \mod m)$, then we obtain the following facts (useful is mon calculation):

- $(a + b) \equiv ((a \mod m) + (b \mod m)) \mod m$
- Hence:

$$a + b \mod = (a \mod m) + (b \mod m)) \mod m$$

- $ab \equiv ((a \mod m)(b \mod m) \mod m$
-

$$ab \mod m = ((a \mod m)(b \mod m) \mod m$$

- $a^n \equiv (a \mod m)^n \mod m$

$$a^n \mod m = (a \mod m)^n \mod m$$

Example is $9^{1000} + 9^{10^6}$ divisible by 5?
We compute first:

$$9 \equiv -1 \mod 5$$

Which gives us:

$$9^{1000} + 9^{10^6} \equiv (-1)^{1000} + (-1)^{10^6} \equiv 1 + 1 \equiv 2 \mod 5$$

Hence, $9^{1000} + 9^{10^6}$ is not divisible by 5

March 26, 2025 — **Lecture 12 : Modulo**

Check digits mod 97

Write down an integer in decimal notation *e.g*:

$$021\ 235\ 1234$$

Compute its remainder after division by 97:

$$021\ 235\ 1234 \quad \text{mod } 97 = 95$$

Appends the remainder to the number, as check digit:

$$021\ 235\ 1234\mathbf{95}$$

A common mistake consists in transposing two digits:

$$021\ 253\ 1234\ 95$$

The check digits are no longer consistent:

$$021\ 253\ 1234 \quad \text{mod } 97 = 63$$

Procedure mod 97 − 10

It's a variant of the previous one:

1. Append 00 (i.e., multiply the number by 100)
2. Let r be the remainder of the division by 97
3. The check digits are $c = 98 - r$ (wrriten as a 2 digit number, e.g. 03)
4. Replace 00 with c
5. Check: The resulting number $\bmod 97$ equals 1.

Which look like this if you wanted to do a function:

$$n \to 100n + \underbrace{98 - (100n \bmod 97)}_{\text{check digit}}$$

To check if the encoding is correct, we juste have to see if the result mode 97 is equals to 1, for instance:

$$\begin{aligned} &100n + 98 - (100n \bmod 97) \bmod 97 \\ &\qquad = 100n + 98 - 100n \bmod 97 \\ &\qquad = 98 \bmod 97 \\ &\qquad = 1 \end{aligned}$$

IBAN (Internation Bank account number)

Main difference to MOD 97 - 10: The check digits are in position 3 and 4:

Example

1. Account number: $\overbrace{00243}^{\text{bank, 5 digits}}\ \ \overbrace{0001\,243\,6789}^{\text{account, 12 digits}}$
2. Append CH (for swiss bank account): $00243\,0001\,2345\,6789$ **CH**
3. We then convert the letter into number ($A \to 10, Z \to 35$, we then finally use the MOD 97 - 10 procedure

April 1, 2025 — Lecture 13 : Multiplicative Inververse

Why modular arithmetic Modular arithmetic is the foundation of number theory, therefore we need number theory for cryptography and for channel coding.

Introduction to $\mathbb{Z}/m\mathbb{Z}$ Instead of considering integers and congruences $(\mod m)$ and write equation like:

$$a + b \equiv c(\mod m)$$

We would like to make our life easier like this:

$$a + b = c$$

This can be done, if we give a new meaning to a, b and c. namely we make them the congruence classe $[a]_m, [b]_m[c]_m$ and $[c]_m$.

Definition 21 *Let $m > 1$ be an integer, called the modulus.*
The set of all integers congruent to $a(\mod m)$ is called the congruence class of a modulo m.
it is denoted by $[a]_m$.

It is the same foundation as the one of finite field (corps fini) in linear algebra. (With prof. Sherer).

Other definition **Definition 22** *The set of all congruence classes modulo m is denoted by $\mathbb{Z}/m\mathbb{Z}$ (which is read $\mathbb{Z} \mod m$.*

Note Some authors use the notation $\mathbb{Z}_m$

Example Some more example to see how this works:
if we have the class $[a]_m$ and:

$$a = mq + r, \text{ with } 0 \leq r \leq m - 1$$

Then we have that:

$$[a]_m = [r]_m$$

If we take for example $[-13]_9$ and $[5]_9$ we can see here that there are equal such that

$$[-13]_9 = [5]_9$$

Sum In $\mathbb{Z}/m\mathbb{Z}$ we define the sum and the product as follows:

- $[a]_m + [b]_m = [a + b]_m$
- $[a]_m[b]_m = [ab]_m$

The result is the same regardless the choice of representative. In fact:

- if we choose $[a + km]_m$ instead of $[a]_m$

- and $[b+lm]_m$ instead of $[b]_m$
- Then we obtain $[a+km]_m + [b+lm]_m = [a+km+b+lm]_m$ which is equal to $[a+b]_m$

Properties of + in $\mathbb{Z}/m\mathbb{Z}$

The sum has the following properties:

- $[a]_m + ([b]_m + [c]_m) = ([a]_m + [b]_m) + [c]_m$
- There exists an additive identity, namely $[0]_m$:
$$[a]_m + [0]_m = [0]_m + [a]_m = [a]_m$$
- There exists an inverse with respect to addition: every $[a]_m$ has an inverse, denoted $[-a]_m$ such that:
$$[a]_m + [-a]_m = [-a]_m + [a]_m = [0]_m$$
- Commutativity
$$[a]_m + [b]_m = [b]_m + [a]_m$$

Properties of × in $\mathbb{Z}/m\mathbb{Z}$

The multiplication has the following properties:

- associativity
$$[a]_m([b]_m[c]_m) = ([a]_m[b]_m)[c]_m$$
- multiplicative identity, namely $[1]_m$:
$$[a]_m[1]_m = [1]_m[a]_m = [a]_m$$
- commutativity
$$[a]_m[b]_m = [b]_m[a]_m$$

Mixed properties

- Distributivity:
$$[a]_m([b]_m + [c]_m) = [a]_m[b]_m + [a]_m[c]_m$$

The notation $k[a]_m$ in $\mathbb{Z}/m\mathbb{Z}$

For an arbitrary positive integer k, $k[a]_m$ is a short hand for
$$\underbrace{[a]_m + [a]_m + \cdots + [a]_m}_{\text{k times}}$$
We can easily verify that:
$$k[a]_m = [ka]_m = [k]_m[a]_m$$

Multiplicative Inverse

Some elements of $\mathbb{Z}/m\mathbb{Z}$ have the **multiplicative inverse**.
The multiplicative inverse of $[a]_m$, if it exists is an element $[b]_m$ such that:
$$[a]_m[b]_m = [1]_m$$

Theorem 28 *The multiplicative inverse if it exits it is unique, and it is denoted by* $([a]_m^{-1})$.

Furthermore $(([a]_m)^{-1})^{-1} = [a]_m$

Proof Suppose first that $ab = 1$ and $ac = 1$ (a has two inverse). The we now that $ab = ac$. If we multiply both side by b:

$$bab = bac$$

However we know that $ab = 1 = ba$:

$$\begin{aligned} b \cdot 1 &= c \cdot 1 \\ b &= c \end{aligned}$$

April 2, 2025 — **Lecture 14 : Read Slide**

Powers in $\mathbb{Z}/m\mathbb{Z}$

For an positive integer k,

- $([a]_m)^k$ is a short hand for $\underbrace{[a]m[a]m\dots[a]_m}_{\text{k times}}$
- $([a]_m)^0 = [1]_m$

Note that we do not consider negative power because it is problematic in general except -1 which is juste the multiplicative inverse

Exercise

Suppose $[a]_m \in \mathbb{Z}/m\mathbb{Z}$ has a multiplicative inverse.
Does there exist k such that:

$$([a]_m)^k = [0]_m$$

We denote first $[b]_m = ([a]_m)^{-1}$. The first statement implies that:

$$([b]_m)^k \cdot ([a]_m)^k = [0]_m$$

Which is:

$$\begin{aligned} [0]_m &= ([b]_m)^{k-1} \overbrace{[b]_m[a]_m}^{=[1]_m}([a]_m)^{k-1} \\ &= ([b]_m)^{k-1}([a]_m)^{k-1} \\ &= \dots = [1]_m \end{aligned}$$

Which is false. Therefore, The answers is no.

Function with multiplicative inverse

Theorem 29 *In $\mathbb{Z}/m\mathbb{Z}$ the following statement are equivalent:*

- *$[a]_m$ has an inverse*
- *For all $[b]_m$, $[a]_m x = [b]_m$ has a unique solution*
- *There exists $a[b]_m$, such that $[a]_m x = [b]_m$ has a unique solution*

Proof 1 → 2 We multiply both side of $[a]_m x = [b]_m$ by $[a]_m^{-1}$ then we have $x = [a]_m^{-1}[b]_m$ showing that there is a solution and the solution is unique.

Proof $2 \to 1$ For $[b]_m = [1]_m$ we obtain $[a]_m x = [1]_m$ which is a solution by assumption. The solution is the inverse of a.

Proof $2 \to 1$ True since (3) is a weaker statement than (2). (one says for all and the other says there exists)

Proof 3 → 2 Here the claim is: $\exists b$ it has a unique solution $\implies \forall b$ there is a unique solution. The negation of this claim is:
$\exists b$ either there is no solution or multiple solution $\implies \nexists b$ there is a unique solution.
It says that if there is a b with **multiple** solution implies that there is no b with a unique solution.

Proof Let $[a]_m[x_1]_m = [b^*]_m$ and $[a]_m[x_2]_m = [b^*]_m$. Now we define:

$$[x_3]_m = [x_1]_m - [x_2]_m[a]_m[x_3]_m = [0]_m \tag{3.1}$$

Now we select **any** $\bar{b}_m$, we then suppose that $[a]_m[x_4]_m = \left[\bar{b}\right]_m$, Because of the fact that we have the solution x_3 before, we can juste add them up:

$$[a]_m([x_3]_m + [x_3]_m) = \left[\bar{b}\right]_m$$

Proof 3 $\implies$ 2 We prove this using the contrapositive i.e, we assume that there is a $[b]_m$ such that $[a]_m x = [b]_m$ has either no solution or multiple solutions, and we prove that for no $[b]_m$, $[a]_m x = [b]_m$ has a unique solution.

- So suppose that $[\mathrm{a}]_m \mathrm{x} = [b]_m$ has no solution or multiple solutions
- By the pigeonhole principle, the map $x \to ax$ is neither **injective** nor **surjective**.
- We can find $a[b^*]_m$ $[a]_m x = [b^*]_m$ has multiple solutions say, x_1, x_2. We define then $x_3 = x_1 - x_2 \neq [0]_m$
- Hence, $[a]_m x_3 [a]_m x_1 - [a]_m x_2 = [b^*]_m - [b^*]_m = [0]_m$
- So the solution $[a]_m x = [0]_m$ has at least two solution x_3 and $[0]_m$
- If $[a]_m x = [b]_m$ has a solution, say x_4, then $x_3 + x_4$ is also a solution.
- We conclude that for no $[b]_m$, $[a]_m x = [b]_m$ has a unique solution.

Example If it exists, find the solution of $[2]_7 x + [3]_7 = [1]_7$

$$\begin{aligned}
[2]_7 x &= [1]_7 + (-[3]_7) \\
&= [2]_7 x = [-2]_7 \\
&= x \qquad\qquad\qquad\qquad = ([-2]_7)^{-1}[5]_7 \\
&= [4]_7[5]_7 \\
&= [20]_7 \\
&= [6]_7
\end{aligned}$$

$\mathbb{Z}/p\mathbb{Z}$

$\mathbb{Z}/p\mathbb{Z}$ with p prime if p is prime, all elements of $\mathbb{Z}$ $p\mathbb{Z}$ except $[0]_p$ have a multiplicative inverse The proof is only taking the fact that the gcd between every number in the classe and p is 1, therefore has an inverse. (except 0 which have p).

Euclidiean algorithm For all this part I think the best is the slide in the pdf Slides 2025 week7 between 100 and 135. or a youtube video.

April 8, 2025 — **Lecture 15 : Commutative Groups**

What's next After $\mathbb{Z}$ $m\mathbb{Z}$ we could proceed in two directions:

Finite groups Focus on finite groups, which are finite sets with one operation, like $(\mathbb{Z}/m\mathbb{Z}, +)$ we do so now because we need them for cryptography.

Finite field Focus on finite field, which are finite sets with two operations, like $(\mathbb{Z}/m\mathbb{Z}, +, \cdot)$ With the extra property that every non-zero element gas a multiplicative inverse. We do so later as we need finites fields for channel coding

3.2.1 Commutative Group

Definition 23 *A **commutative group** (also called Abelian group) is a set G endowed with a binary operation * that combines any two elements a and b to form another element denoted a * b. The groupe operation * must satisfy the following five axioms:*

- *(Closure) For all $a, b \in G$, a * (b * c) = (a * b) * c*
- *(Associativity) For all $a, b \in G$ a *(b * c) = (a * b) * c*
- *(Identity element): There exists an element $e \in G$ such that for all $a \in G$,* ***a * e = e * a = e***
- *(Inverse element) For all $a \in G$ there exists a $b \in G$ such that a * b = b * a = e*
- *(Commutativity) For all $a, b \in G$ a * b = b * a*

$\mathbb{Z}/m\mathbb{Z}*$ To obtain a commutative group with modulo multiplication, we take only the elements of $\mathbb{Z}/m\mathbb{Z}$ that have multiplicative inverse. The resulting set is denoted $\mathbb{Z}/m\mathbb{Z}*$ For every integer $m > 1$ ($\mathbb{Z}/m\mathbb{Z}^* m\cdot$) is a commutative group.

Euler function

Definition 24 *Euler's function $\varphi(n)$ (also called Euler's totient function) is the number of positive integers in $\{1, \ldots, n\}$ that are relatively prime to n.*

Observation Here we can see two main thing :

- $\varphi(m)$ is the cardinality of $\mathbb{Z}/m\mathbb{Z}^*$
- if p is prime, $\varphi(p) = p - 1$

The cartesian product of a commutative groupe is a commutative group Recall of the axioms of a commutative group:

- (Closure) For all $a, b \in G$, a * (b * c) = (a * b) * c
- (Associativity) For all $a, b \in G$ a *(b * c) = (a * b) * c
- (Identity element): There exists an element $e \in G$ such that for all $a \in G$, **a * e = e * a = e**
- (Inverse element) For all $a \in G$ there exists a $b \in G$ such that a * b = b * a = e
- (Commutativity) For all $a, b \in G$ a * b = b * a

We can see the $(a_1, a_2) \in (G_1, op_1) \times (G_2, op_2)$ we see that this **is** a commutative group.

Isomorphism Some sets endowed with an operation might look different, buit they are actually the same once their elements are re-labeled.

Definition 25 *Let (G, *) and (H, $\oplus$) be sets, each endowed with an arbitrary binary operation.*
an ***isomorphism*** *from (G, *) to (H, $\oplus$) is a bijction $\psi : G \to H$ such that*

$$\psi(a*) = \psi(a) \oplus \psi(b)$$

holds for all $a, b \in G$
*We say that (G, *) and (H $\oplus$) are* ***isomorphic*** *if there exists an isomorphism between them.*

Je me suis arreeter vers les slide 50

April 15, 2025 — **Lecture 17 : public key cryptography**

Proof that $m1$ and m_2 coprime $\implies \varphi$ is bijective First we prove that ψ is one-to-one:

- $\iff$ m_1 and m_2 divide $(k - k')$
- because m_1 and m_2 have no common facots, $m_1 m_2$ divides $(k - k')$
- Hence $[k]_{m_1 m_2}$

April 16, 2025 — **Lecture 18 : Last lecture on crypto**

with m a prime First we select m a prime, we then take $m = p$.
Then, we select e such that gcd $(e, p - 1) = 1$
We select d such that $[ed]_{p-1} = 1$
Works but is not secret.

Select $m = pq$ select e such that $\gcd(e, p-1) = 1$
select d such that $[ed]_{p-1} = 1$

Difference Here that thing that is hard for people who want to crack the code, is to find p and q from m.

Chinese remainder If we take as before $\mathbb{Z}\ m_1, m_2, \mathbb{Z}$, with m_1, m_2 coprime.

$$[k]_{m_1,m_2} \to \left([k]_{m_1}, [k]_{m_2}\right)$$

$$[0]_{m_1,m_2} \to \left([0]_{m_1}, [0]_{m_2}\right)$$

We see here that **this** doesn't have a multiplicative inverse, the reason is that from the Chinese remainder, we will get 0:

$$[m_1]_{m_1,m_2} \to \left([0]_{m_1}, [m_1]_{m_2}\right)$$

Now We do the Chinese remainder theorem with p and q (two distinct prime): $\mathbb{Z}\ pq\mathbb{Z}$: the question is what are **all** the elements that do not have a multiplicative inverse?. The answer is all the multiple of q or p, which you can do a mapping with the Chinese remainder:

$$[k]_{pq} \to \left([0]_p, [k]_q\right)$$

Or:

$$\to \left([k]_p, [0]_q\right)$$

The issue is from does who have a 0 is one of their "multiplicative".

> *Remark* We only use the Chinese remainder **ONLY** for proving, we won't use it to compute in RSA. (if Alice know p and q she can then find every cipher text).

Fermat + Chinese Remainders

- Let p q be distinct primes
- Let k be a multiple of both $(p-1)$ and $(q-1)$
- for all non-negative l

$$\left([a]_p\right)^{lk+1} = [a]_p$$

$$\left([a]_q\right)^{lk+1} = [a]_q$$

- Using the Chinese remainders theorem, we combine into:

$$\left([a]_{pq}\right)^{lk+1} = [a]_{pq}$$

We have proved the following result (TextBook Thm 10.3):

> **Theorem 30** *Let p and q be disctinct prime number and let k be a multiple*

> *Proof, idea of the proof* Because $(\mathbb{Z}/pq\mathbb{Z}, \cdot)$ is isomorhpic to (by the mapping of the Chinese remainder) $(\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}, \cdot)$ The relation is equivalent

to:

$$\left([n]_p\right)^{1+m} = [n]_o$$

And:

$$\left([n]_q\right)^{1+m} = [n]_q$$

If $[n]_p = 0$ then the equation is true, else, $[n]_p$ is inversible because p is prime, therefore, by Euleur theorem $\left([n]_p\right)^{p-1} = [1]_p$. Furthermore, m is a multiple of $p-1$, this means, there exists an integer $l \geq 0$ such that $m = l(p-1)$:

$$\Big([n]_p m = \left([n]_p\right)^{(p-1)l} = \left([1]_p\right)^e ll = [1]_p$$

3.2.2 RSA, Rivest

High level Suppose that we can find:

- Integer m (modulus)
- Integer e (encoding exponent)
- integer d (decoding exponent)

Such that, for all integers $t \in \mathbb{Z}/m\mathbb{Z}$ (plaintext)

$$\left[(t^e)^d\right]_m = [t]_m$$

Then:

- The receiver generates m, e, d (we sill se later on how)
- (m, e) is the public encoding key –Announced in a phone like public directory
- (m, d) is the priate decoding key – d never leaves the receiver
- To send the plaintext $t \in \mathbb{Z}/m\mathbb{Z}$
- The encoder forms the cryptogram $c = t^e \bmod m$ Exponentiation is easy
- The intended decoder performs $c^d \bmod m$ and obtains the plaintext t.

Example suppose we have $m = 33, e = 7, d = 3$
suppose that the plaintext is $t = 2$
The encryption is $c = t^e \bmod m = 2^7 \bmod 33 = 128 \bmod 33 = 29$
The decryption is:

$$c^d \bmod m = 29^3 \bmod 33 = \ldots = 2$$

s expected.

RSA keys generation

- generate large prime p and q at random (which is approximately $\frac{\log n}{n}$), we want to use a prime that has never been use before, imagine using a m that as already be used, we can just check if another key match our and if it is, it will be cracked.
- $m = pq$ is the modulus used for encoding and decoding
- let k be a multiple of $(p-1)$ and $(q-1)$ to be kept secret

- For instance $k = \varphi(pq)$ or $k = lcm(p-1, q-1)$
- Produce the public (encoding) exponent e such that gcd(e, k) = 1
- (a common choice is $e = 65537 = 2^{16} + 1$ which is a prime number. No need for e to be distinct for each recipient)
- the public key is (m, e)

The choice for $e = 2^{16} + 1$ is here because it is very easy to compute big power like this: $\left(\left(\left(t^2\right)^2\right)^2 \dots\right)^2 \cdot t = t^{2^{16}+1}$ which makes it easy to comput the cipher text to the power of e.

How decoding workds

$[t]_m \in \mathbb{Z}/m\mathbb{Z}$ with $m = pq$ Hence:

$$\begin{aligned}\left([t]_m^2\right)^d &= [t]_m^{ed} \\ &= [t]_{pq}^{1-kl} \\ &= [t]_{pq} \text{ Fermat + CRs} \\ &= [t]_m\end{aligned}$$

Example (toy-key generation)

taking $p = 3$, $q = 11$, $m = 33$, $k = \varphi(pq) = (2-1)(11-1) = 10$.

- $e = 7$ which is relatively prime with k
- $d = 3$ (check that $ed \bmod k = 1$)
- the public key is $(m, e) = (33, 7)$
- The private key is $(m, d) = (33, 3)$

Now let us encrypt. Each letter of the alphabet is converted into a number in $\{1, 2, \dots, m-1 = 32\}$ (we avoid 1 to avoir $c = 0 = t$)

- We use the natural order, $a \to 1, b \to 2$ etc...

Attack

How to decrypt not knowing d? here the possibilites (that we know of):

- factor m to find p and q. Very hard to do if m is large (say $\approx 2^{500}$)
- in $\mathbb{Z}\ m\mathbb{Z}$ solve $c = x^e$ for x Which is very hard to do if m is large.
- guess k
- guess t
- guess d

The trapDoor one-way function Behind RSA

The trapdoor one-way function is:

$$t \to x = t^e \bmod m$$

Where e is called the encoding exponent.
Instead of publishing the function, it suffices to publish (m, e). This is called the public key.
Someone that knows (m, d) can perform

$$c \to t = c^d \bmod m$$

where d is called the decoding exponent
Hence the trapdoor informations is (m, d). It is called the private key.

3.3 Digital signature

We have used trapdoor one-way functions for privacy. In conjunction with hash function, they are equally suited for authenticity.
The goal here to “prove” that this is actually us that written the message.

Issue

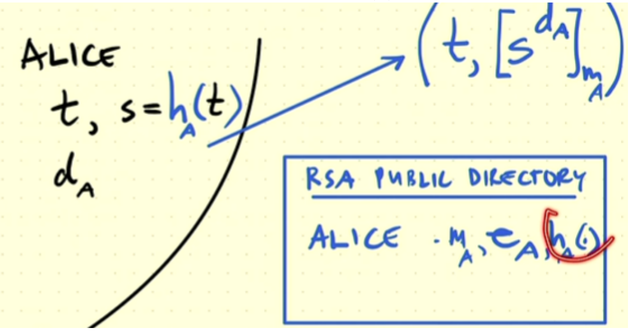
Public receives a message from Alice?
But is the message really from Alice?

RSA public directory

With RSA, Alice sends the message t but she also sends $\left[t^{da}\right]_{m_A}$. The question is how do we know with this additional information that this is Alice that sent this message.

Following Questsion

Alice has a message **and** a hashing function $s = h_A(t)$ which sends then in the public: $\left(t, \left[s^{d_A}\right]_m\right)$


ALICE
$t, s = h_A(t)$
d_A
$\left(t, \left[s^{d_A}\right]_{m_A}\right)$
RSA PUBLIC DIRECTORY
ALICE $\cdot m_A, e_A, h_A(\cdot)$

How does the public check?

Step 1

$$\left[\left(\left[s^{d_A}\right]_m\right)^{e_a}\right]_m = s^a$$

Step 2

$$t \to h_A(t) = s$$

Step 3 if $s =$

Hash function

A hash function is a many-to one function, used to map a sequence of arbitrary length to a fixed length bit sequence of, say, 200 bits
what we expect from a hash function, is that even the smallest change in the

input produces a different output
Ideally it should be so that one has to try about 2^{200} alternative inputs to hope to find a sequence that produces a given output.

Digital Signature to sign a document, we appends to the document a hash function of the docmunent in such a way that only the signee could have done it

Trusted Agency How do we know that the directory storing all the public keys has not been tampered with?

> *Example* Alies queries the public directory for bob's directory

> *oui* The directory information signed by a trusted agency, Say Symantex. Here is how:
>
> Symantec's public key is distributed one and for all via a channel that cannot be tapered with (e.g., hard coded into the crypto hardware)
>
> - Each directory entry is digitally signed by symantec. We call the result a certificate
> - Anybody that has Symantec's public key can verify that the information areceived from the directory is authentic

3.3.1 Summary of chapter 2

Perfect secrecy is possible but requires long keys:

One-time pad In onte time pad, the cryptogram is compute like this:

$$\text{Cryptogram} = \text{Plaintext} \oplus \text{SharedKey}$$

- If the sharedKey is perfectly (uniformly) random and shared between encrypter and decrypter ahead of time
- and the sharedKey is kept secret from anyone else
- Then the One-time pad offers perfect secrecy
- Hence: it is expensive to implement. Only worth it for spies and such.

Practical cryptography is based on algorithmic/computational complexity
pPublic key cryptography. Most public key cryptographic algorithm fall into one of the following two categories:

- those that are based on the belied that diescrete exponentiation (in mulitplicative cyclic group) is a one way function (e.g. Diffie-Hellamn and ElGamal)
- Those that are based on the difficulty of factoring (e.g. RSA)

To understand RSA and Diffie-Hellman, we need Number Theory and algebra.

3.3.2 Number theory and algebra

First with modulo Operation and the euclid Algorithm (we can find the multiplicative inverse gcd etc....

Group

- $\mathbb{Z}/m\mathbb{Z}$ with addition is always a group
- $\mathbb{Z}/m\mathbb{Z}$ with mulitplication: need to retain only those elements that have a multiplicative inverse: $\mathbb{Z}/m\mathbb{Z}^*$
- Finding multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$: Bézout identity; Extended Euclid algorithm
- How many element in $\mathbb{Z}/m\mathbb{Z}$ have a multiplicative inverse? the answer is Euler's function (torient function) $\varphi(n)$
- Group isomorphism
- Order of group elements Lagrance's theorem: Order of any group element must divide the cardinality of the group

Product group

Theorem 31 *Main theorem: Cartesian product of groups is again a group*

The special Isomorphism The special isomorphism between $\mathbb{Z}/m_1m_2\mathbb{Z}$ and $\mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}$ when m_1 and m_2 are coprime:

- Holds for both addition and multiplication, including for elements that do not have multiplicative inverse,
- Hence, this is more than juste a group ismorphism

Computationally hard problem

Discrete logarithm leads to Diffie Hellman(and, by slight extension, El Gamal)

- Encryption $A = g^a, B = g^b$
- Leads to a shered key: $C = A^b = B^a$
- To understand that it workds, we need cyclic groups.

Factorization of large integers leads to Cocks and RSA.

- Encryption: $t^e \bmod m$ where t is the plaintext and $m = pq$, where p and q are primes
- Decryption $(t^e)^d \bmod m$
- To understant that it works (meaning that $(t^e)^d \bmod m = t$ for all plaintexts t), we need to understand $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$

Authencity: Digital Signatures In practice, so called symetric key cryptoSystems are important. The common secret key is typically only a few hundred bits, distributed e.g., via Diffie-Hellman. Encryption/decryption can be implemented more efficiently (faster algorithm, smaller hardware). Think: one-time pad, but with an imperfect key. There is no proof that the resulting algorithm is secure.

Chapter 4

Error Channel

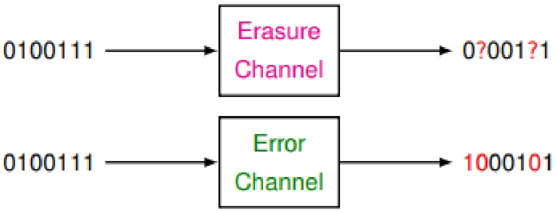
April 29, 2025 — **Lecture 19 : Error Correction**

Point to point communication system Now we have compressed the source coding, we encrypted it. What we want now is to actually transfer the message to the receiver side. However we have to protect ourself from the mean world. For example with the communication over the internet it is our application which does the compression and encryption.

Motivation Channel Model The question is how the bad thing happens:

- The internet often drops packets due to congestion
- Not all the bits on a storage device can be retrieved
- **Wireless signals are very noisy**

We consider two types of channel models:



If we model the world has being a **Erasure** channel mode, then the following happens:
So your model replace some of your bit with question mark. (those bit are no longer readable). So for example you store your holiday picture into your hdd but some water fell on it, then there will be some error and those error is the question mark.
The model can **only** replace the bit by a question mark.
The question is "how we decide which bit has to be replace with a question mark?" For example like this:
So for the first 0 let us say we throw a dice, if the outcome is 6, we will replace the outcome by a question mark or we won't. Therefore, here we can say that the erosion probability is $\frac{1}{6}$.

However maybe the eraser is not random, maybe someone who know what we're gonna do (the algorithm to find the erasure) will find the one that will be a question mark.
Here the channel input alphabet is not necessarily binary.
We can take for the alphabet $\mathcal{A} = \{0, 1, 2, 3, 4\}$ and works with this.
For the second model (Error Channel), The channel can completely change entire symbol. The output is in the same alphabet as the input (which make the thing "easier"). What we can do is the substitute (0 to 1, 1 to 0 and then 0 to 0).
This makes the channel way harder to know which of these bits have been flipped.

The channel input alphabet doesn't have to be binary

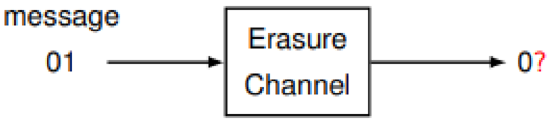
Erasure Weight

Definition 26 *We define the* ***erasure weigh*** *p (resp.* ***error weight p****) as the total number of erasures (resp. errors).*

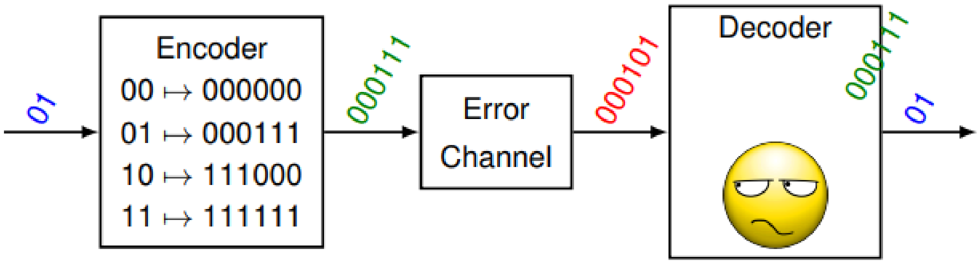
For example if we take the erasure channel from before, erasure weight: $p = 2$. (error weight: $p = 3$ for the error channel).

Channel coding to deal with erasures

Here this is the beginning of every algorithm for channel coding which goes: Suppose that the source outputs 2 bits, and we store them as is (no channel coding):



f any bit is erased, there is no way to determine the original message (all hypothesis are equally possible).
So the goal here is the put redundancy, the goal is the create a redundancy that match exactly the message. The redundancy serves to deal with the back eyes (i don't rembeber).
So first we got rid of redundancy, we encrypt, and the finaly we add back in redundancy (which is not the same as the one we got rid off).
So first we got rid of redundancy, we encrypt, and the finaly we add back in redundancy (which is not the same as the one we got rid off).
Now suppose that we do channel coding like this:



The channel output is not a valid codeword. The decoders recognizes it, and assumes that the transmitted codeword is the one that agrees in most position with the observed channel output.
We add redundancy like some backups bits to make it stronger.

We study only block codes

The above is an (n, k) **block code** with $n = 6$ and $k = 2$: each k source symbols are substituted by n channel symbols over the same alphabet. Since the alphabet is $\{0, 1\}$, we call it a **binary** (n, k) block code.

We consider only block codes

Definition 27 *A* ***block code*** *of length n, defined on a alphabet $\mathcal{A}$,* ***is*** *a subset of $\mathcal{C}$ of $\mathcal{A}^n$, which means it is a sequence of n elements of $\mathcal{A}$. Element of the block code are called* ***codeword****.*

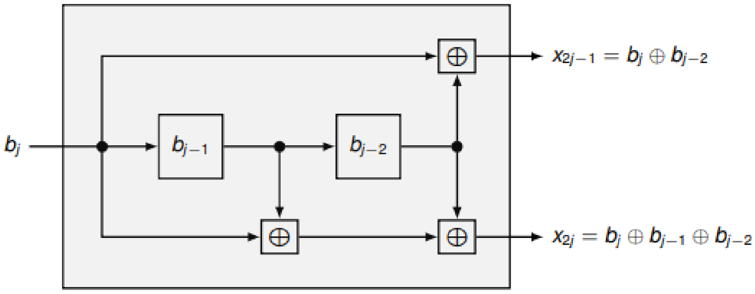
Rate

Definition 28 *the* ***rate*** *of the code is defined by:*

$$r = \frac{1}{n} \log_{|\mathcal{A}|} |\mathcal{C}|$$

Example

The following is a convolutional encoder: every encoder input symbol produces two encoder output symbols.
The output pair produced at any given time is a linear function of the corresponding encoder input and encoder state (the previous two inputs).


b_j
b_{j-1}
b_{j-2}
$x_{2j-1} = b_j \oplus b_{j-2}$
$x_{2j} = b_j \oplus b_{j-1} \oplus b_{j-2}$

So here we put the bit (from the encryption or whatever) we stores the previous bit. We then compute the two output.
Here this is not a really good code (too slow for our smartphone but it is okay for satellite communications).

Terminology

- The code $\mathcal{C}$ is the set of codewords
- A codeword c is an element of $\mathcal{A}^n$. (the alphabet $\mathcal{A}$ is $\{0, 1\}$ in our example)
- The block length is n (which is the length after the →)
- Each codeword carries $k = \log_2 |\mathcal{C}|$ bits of information (however we don't have to use $\log_2$ we can also use $\log_{|\mathcal{A}|}$ which won't have bit as a unit).
- The rate is $\frac{k}{n}$ bits per symbol

Here $|\mathcal{C}|$ doesn't have to be equal to $\mathcal{A}^n$.

4.1 Hamming Distance

Hamming distance

Definition 29 *The **Hamming distance $d(x, y)$** between two n-tuples x and y is the number of positions in which they differ.*

Example

- $x = (101110)$, $y = (100110)$, $d(x, y) = 1$
- $x = (0427222)$, $y = (1227986)$, $d(x, y) = 5$

The Hamming distance is indeed a distance

In math, a function of two variables is a distance if it satisfies the following axioms:

Definition 30 *Distance axioms:*

- ***non-negativity**: $d(x, y) \geq 0$ with equality if and only if $x = y$*
- ***Symmetry**: $d(x, y) = d(y, x)$*
- ***triangle inequality**: $d(x, z) \leq d(x, y) + d(y, z)$*

Proof We can first rewrite the distance like this:

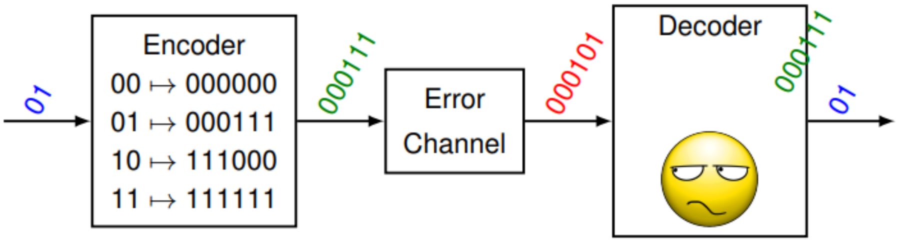
$$d(x, y) = \sum_{i=1}^{n} d(x_i, y_i)$$

Where the x_i and y_i have a distance of 1.
The claim is: $d(x_i, z_i) \leq d(x_i, y_i) + d(y_i, z_i)$.
To prove this we do a proof by cases:

- **Case 1**: $x_i = z_i \iff d(x_i, z_i) = 0$ there fore the inequality hold ($0 \leq d(x_i, y_i) + d(y_i + z_i)$
- **Case 2**: $x_i \neq z_i \iff d(x_i, z_i) = 1$
  Therefore in this case, there is at least one of the $d(x_i, y_i) = 1$ or $d(y_i, z_i)1$ (or both) which is greater or equal to 1. (here we use one because we said before that we were using "vecteurs unitaires".

Minimum Distance decoder

- The decoder guesses the encoder input based on the channel output
- Here we consider only **minimum distance decoders**.



How it works is pretty simple. It take the output codeword and compute the hamming distance with every element of the alphabet $\mathcal{C}$. And chose the one with the smallest hamming distance.
For our example let us compute the hamming distance:

$$d_H(000101, 00000) = 2$$
$$d_H(000101, 000111) = 1$$
$$d_H(000101, 111000) = 5$$

We called here 000111 the winner! (we don't know for sure if it is **but** it is a good answer, a good starting point).

Minimum distance decoder

Definition 31 *Let y be the channel output observed by the decoder. A minimum-distance decoder decides that the channel input is (one of) the $\hat{c} \in \mathcal{C}$ for which $d(y, \hat{c})$ is minimized.*

$$\hat{c} \in arg\ min_{x \in \mathcal{C}} d(y, x)$$

We pick one in the minimum if there are more than one with small Hamming distance.
The justification is that a small error weight is more likely than a large one.

Minimum distance

Definition 32 *The minimum distance of a code $\mathcal{C}$ is*

$$d_{min}(\mathcal{C}) = min_{x,y \in \mathcal{C}: x \neq y} d(x, y)$$

Example Given the set $\mathcal{C} = \{000000, 100110, 011001, 111111\}$, we get that

$$d_{min}(\mathcal{C}) = 3$$

What to expect from a decoder

For an error channel:

- Channel error correction: The best is if the decoder recognize and corrects the channel errors. In this case, the encoder input is recovered error-free.

- Channel-error detection: in some circumstances, the encoder is able to detect the presence channel errors but it is unable to correct them. The receiver may or may not ask for retransmission
- Decoding error: the worst is if the decoder tries to do as in (1) and makes the wrong decision.

For an erasure channel:

- erasure correction: the best is if the decoder is capable of filling in the erased positions. In this case, the encoder input is recovered error.free
- (erasure detection: unlike errors, erasure are always detected)
- decoding error: the worse is if the decoder fills in one or more erased position with incorrect symbols.

Error detection The question now is how does error detection is related to $d_{min}(\mathcal{C})$

Theorem 32 *Error detection:*

1. *Channel errors of weight $p < d_{min}(\mathcal{C})$ do not lead to a codeword. Hence they are detected.*
2. *Some channel errors of weight $p \geq d_{min}(\mathcal{C})$ do lead to another codeword. Hence they cannot be detected by a minimum-distance decoder.*

I juste want to put the theorem found in the book given in moodle because I fond it clearer:

Theorem 33 *Détection d'erreur (theorem 11.2):*

1. *Un code $\mathcal{C}$ est capable de détecter toutes les erreurs de poids $p < d_{min}(\mathcal{C})$*
2. *Inversement, si un code $\mathcal{C}$ peut détecter toutes les erreurs de poids $\leq p$, alors $p < d_{min}(\mathcal{C})$*

Note Erasures are always detected (by definition)

Proof Firstly, let $c \in \mathcal{C}$ be transmitted and y be received. We know that if $p = d(c, y) < d_{min}(\mathcal{C})$, y cannot be a codeword, therefore the error is detected.
We construct an example in which a channel error of weight $p = d_{min}(\mathcal{C})$ cannot be detected.
Let c and c' be a codeword at distance $d_{min}(\mathcal{C})$.
Suppose that c is the channel input and the channel output is $y = c'$.
y is a codeword. A minimum distance decoder will decide that no channel error has occured.

Example (Error detection) Let the encoding map be the MOD 97-10 procedure:

$$u \to v = (100 \cdot u) + (98 - [100 \cdot u]_{97})$$

Recall that v is considered as valid if $[v]_{97} = 1$
For example $u = 0216936631 \to v = 021693663165$ Suppose v is transmitted and v' is received, $d(v, v') = 1$.
We can always write $v' = v + a10^k$ with $a \in \{-9, \ldots, -1, 1, \ldots, 9\}$.
The only way for v' to be a valid codeword is if $\left[a10^k\right]_{97} = 0$.
Since $[10]_{97}$ is invertible, so is $\left[10^k\right]_{97}$, hence $a = 0$.
Therefore all weight 1 errors are detected, implying that the minimum distance is at least 2.

April 30, 2025 — **Lecture 20 : The rate**

Let use have the code $\mathcal{C}$:

$$\begin{gathered}0000\\0011\\1100\\1111\end{gathered}$$

We want to find n, k, rate and d_{min}.

- $n = 4$
- $k = \log_2|\mathcal{C}| = \log_2 4 = 2$
- rate: $\frac{2}{4} = \frac{1}{2}$
- $d_{min} = 2$

If we take the code:

$$\begin{gathered}0000\\1111\end{gathered}$$

- $n = 4$
- $k = \log_2|\mathcal{C}| = \log_2 2 = 1$
- rate: $\frac{1}{4} = \frac{1}{4}$
- $d_{min} = 4$

For example without binary code:

$$\begin{gathered}000\\001\\002\\\vdots\\222\end{gathered}$$

- $n = 3$
- $k = \log_2|\mathcal{C}| = \log_2 27$
- rate: $\frac{\log_2 27}{3} = \frac{\text{bits}}{\text{channel use}}$ which is the thing you get when you ask swisscom how much 4g you have.
  If we take for example a better universe:
  $$k = \log_3|\mathcal{C}| = \log_3 27 = 3$$

$$\text{rate} = \frac{k}{n} = \frac{3}{3} = 1$$

- $d_{min} = 1$

But we can take the alphabet in a smart way:

000
011
022
110
220
121
102
201
212

- $n = 3$
- $k = \log_3 |\mathcal{C}| = \log_3 9 = 2$
- rate: $\frac{k}{n} = \frac{2}{3}$
- $d_{min} = 2$

Erasure correction: how it relates to $d_{min}(\mathcal{C})$

Theorem 34 *A minimum-distance decoder for a code $\mathcal{C}$* ***corrects*** *(fill in)* ***all the erasures*** *of weight p if and only if $p < d_{min}(\mathcal{C})$*

Let us do it for the previous code (the one right before).
If we take for example 220 → ERASURE → 2?2 the claim now is the we actually can recover from this. Let us check the alphabet which one has a 2 one the first side and a 0 on the other side. We check and we see that there is only one in the alphabet that check all the ticks: 220. We can also say we chose the one that has the smallest hamming distance with the erased codeword.

Proof $\Longleftarrow$

Suppose that $p < d_{min}(\mathcal{C})$.
Let c and y be the input and the output of an erasure channel, respectively, with $d(c, y) = p$.
We show that there is only one way to fille in the erased positions.
let $c \in \mathcal{C}$ and $\overline{c} \in \mathcal{C}$ be two codewords that agree with y in the non-erased positions.
Clearly, $d(c, \overline{c}) \leq p < d_{min}(\mathcal{C})$ This is possible only if $c = \overline{c}$.

Proof $\Longrightarrow$

We use contraposition.
Suppose that $p = d_{min}(\mathcal{C})$
We construct an example where the decoder will not always decode correctly.
Let c and c' be codewords at distance $d_{min}(\mathcal{C})$.
Let c be the channel input, and suppose

Illustrative example

given the alphabet

$$\begin{array}{c} 0000 \\ 0011 \\ 1100 \\ 1111 \end{array}$$

You can see that if you take the erased 00?? you cannot know which one is it is. we took here the $p = d_{min}(\mathcal{C})$ and as you can see we cannot guess.

Error correction

Theorem 35 *A minimum-distance decoder for a code $\mathcal{C}$* ***corrects all channel errors*** *of weight p if and only if $p < \frac{d_{min}(\mathcal{C})}{2}$.*

Proof

Let have the true codeword C, noisy channel output y. Given the assumption, $d(c, y) = p$. Let $\bar{c}$ be the output of the minimum distance decoder. Therefore if we want to comput the hamming distance between $\bar{c}$ and y we get:

$$d(\bar{c}, y) \leq p$$

Then by the triangle inequality:

$$\begin{aligned} d(c, \bar{c}) &\leq d(c, y) + d(y, \bar{c}) \\ &\leq 2p < d_{min}(\mathcal{C}) \end{aligned}$$

Hence, $c = \bar{c}$
Hence, the minimum distance decoder output the correct answer.

A finir la preuve

Summary

	detection guaranteed if	correction guaranteed if
erasure channel	(not applicable)	$p < d_{min}$
error channel	$p < d_{min}$	p < $\frac{d_{min}}{2}$

Upper bound to $d_{min}(\mathcal{C})$

Recall the important parameters of a block code $\mathcal{C}$ over a D-ary alphabet:

- n, the block length
- $k = \log_D |\mathcal{C}|$ the number of information symbols carried by a codeword. (Equivalently, $\mathcal{C} = D^k$)

Singleton's bound

Theorem 36 *Refardless of the alphabet size, the minimum distance of a block code satisfies:*

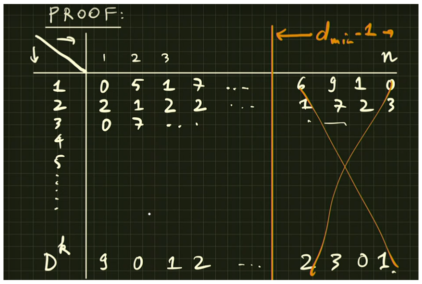
$$d_{min} - 1 \leq n - k$$

Block codes that satisfy the singleton bound with equality are called **Maximum distance separable** codes. (**MDS** codes).

Proof

	1	2	3	...	n
1	0	5	1	...	0
2	2	1	2	...	3
$\vdots$					
D^k	9	0	2	...	2

You can then get rid of the $d_{min} - 1$ from the right like this:



PROOF:

	1	2	3						n
1	0	5	1	7	...	6	9	1	0
2	2	1	2	2	...	1	7	2	3
3	0	7	...						
4									
5									
$\vdots$									
D^k	9	0	1	2	...	2	3	0	1

The next question after this is: how many different strings of length $n - (d_{min} - 1)$ can we have?
We can you combination with the alphabet cardinilality being D we get:

$$nbr_{comb} = D^{n-(d_{min}-1)}$$

Hence we must have:

$$D^k \leq D^{n-(d_{min}-1)}$$

> *Remark* here k doesn't have to be a integer, only D^k **has to** be an integer (by assumption)

Pigeonhole principle a lire mais pas vu en cours donc voila ("it is a bit overkill")

Exercise

4.2 Finite Fields and Vector spaces

May 7, 2025 — **Lecture 22 : PPO midterm hurts**

Yesterday

> *Example* Let $V = \mathbb{F}_7^3$ and define $S = \{(x_1, x_2, x_3) : x_i \in \mathbb{F}_7 \text{ and } x_1 + 2x_2 + 3x_3 = 0\}$
> S is a subspace of V. (Be sure that you see why)

Let us verify that $\vec{x}, \vec{y} \in S$:

1. is $a\vec{x} \in S$?

2. Is $\vec{x} + \vec{y} \in S$?

1. $ax_1 + 2ax_2 + 3ax_3 = a(x_1 + 2x_2 + 2x_3) = 0$
2. $(x_1 + y_1) + 2(x_2 + y_2) + 3(x_3 + y_3)$ We want to check if this is still in the subspace:

$$x_1 + 2x_2 + 3x_3 + y_1 + 2y_2 + 3y_3 = 0$$

Here the number $1, 2, 3$ are values in $\mathbb{F}_7$ (all the scalar are also in the field)

Which leads that S is a subspace.

Some definitions

Definition 33 *A linéaire combinations of a* ***list*** $(\vec{v}_1, \dots, \vec{v}_n)$ *of vectors in* V *is a vector of the form* $\sum_{i=0}^{n} \lambda_i \vec{v}_i$ *where* $\lambda_1, \dots, \lambda_n \in \mathbb{F}$.
The set of all linear combinations of $(\vec{v}_1, \dots, \vec{v}_n)$ *is called the* ***span*** *of* $(\vec{v}_1, \dots, \vec{v}_n)$ *denoted span*$(\vec{v}_1, \dots, \vec{v}_n)$.
if span $(\vec{v}_1, \dots, \vec{v}_n) = V$, *we say that* $(\vec{v}_1, \dots, \vec{v}_n)$ ***spans*** V.
A vector space is called ***finite-dimensional*** *if some liste of vectors in it spans the whole space. (A list has finite length by definition).*

vector

Theorem 37 *A list* $(\vec{v}_1, \dots, \vec{v}_n)$ *of vector in* V *is a basis of* V *iff every* $\vec{v} \in V$ *can be written* ***uniquely*** *in the form*

$$\vec{v} = \sum_{i=1}^{n} \lambda_i \vec{v}_i$$

Proof $\implies$ The think we say here is, that if there is a basis **then** there is a unique representation. The proof is by contradiction:
Suppose that there is two distinct representation:

$$\begin{aligned}(\vec{v}) &= \sum_{i=1}^{n} \lambda_i \vec{v}_i - \sum_{i=1}^{n} \beta_i \vec{v}_i = \vec{0} \\ &= \sum_{i=1}^{n} (\lambda_i - \beta_i) \vec{v}_i = \vec{0}\end{aligned}$$

But all the $\vec{v}_i$ are "indepedant"therefore all the $\lambda_i - \beta_i$ has to be equal to 0 therefore, there are equals.

proof $\Longleftarrow$ hm

Span

Theorem 38 *Every spanning list in a vector space can be reduced to a basis of the vector space*

Proof Remove all the zero-element of the list
Of the new list, remove the second element if it is in the linear span of the first. Repeat the same until we have a list in which the second element is not in the linear span of the first.
Of the new list, remove the third element if it is in the linear span of the first two,

We just continue
At the end, the result is a list of vector that span the vector space and are lineartly independent (or else one vector can be written as a combination of other vector)
Here we could have use Gram-Schmidt

Theorem 39 *Any two bases of a finite dimensional vector space have the same length.*

The **dimension** of a finite dimensional vector space, V denoted by $\dim(v)$, is defined to be the length of any basis of V.

Few properties of the dimension of a vector space

let V be a vector space and suppose that $\dim(V) = n$:

- if $(\vec{v}_1, \dots, \vec{v}_n)$ is a list of linearly independent vectors in V then, it is a basis of V.

Example

We works in $\mathbb{F}_5^3$ and let $S = \{\vec{v} : \vec{v} = \alpha(1,2,3)\}$
So we can write the vector as:

$$\begin{pmatrix} \alpha \\ 2\alpha \\ 3\alpha \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Which implies that:

$$\begin{cases} 2v_1 = v_2 \\ 3v_1 = v_3 \end{cases} \iff \begin{cases} 2v_1 + 4v_2 = 0 \\ 3v_1 + 4v_3 = 0 \end{cases}$$

Therefore:

$$S' = \{\vec{v} : 2v_1 + 4v_2 = 0 \text{ and } 3v_1 + 4v_3 = 0\}$$

Second key example

$v = \mathbb{F}_5^3$ where $(v_1, v_2, v_3)(2\ 3\ 1)^T = 0$ which the subspace:

$$S = \{\vec{v} : 2v_1 + 3v_2 + v_3 = 0\}$$

The goal is the describe S by a **basis**.
Let $v_1 = \alpha$ and $v_2 = \beta$. where $\alpha, \beta \in \mathbb{F}_5$. Then we know that:

$$v_3 = -2\alpha - 3\beta = 3\alpha + 2\beta$$

We have there two free variables which leads to:

$$\begin{aligned} \vec{v} &= (\alpha, \beta, 3\alpha + 2\beta) \\ &= \alpha g(1,0,3) + \beta(0,1,2) \end{aligned}$$

Exercise

Let S be the subspace of $\mathbb{F}_7^3$ spanned by $\vec{v} = (4,3,1)$. Define S by means of equations.

Solution

We first see the it is a one dimensional subspace of $\mathbb{F}_7^3$, Therefore, we only need 2 equations to describe it.

Theorem

Theorem 40 *The set of solutions in $V = \mathbb{F}^n$ of m linear homogeneous equations in n variables is a subspaces S of V.*
Let r be the dimensionality of the vector space spanned by the coefficient vectors. Then dim $(S) = n - r$.
In particular, if the m vectors of coefficients are linearly independent, then, dim $(S)ahn - m$.
Conversely, if S is a subspace of $V = \mathbb{F}^n$ with dim $(S) = k$, there exists a set of $n - k$ linear equations with coefficients that form linearly independent vectors in V, the solution of which are the vectors in S

4.2.1 Rank of matrix

Definition

Definition 34 *For any matrix with entries in a field $\mathbb{F}$:*

- *The dimension of the vector space spanned by its rows*
- *Equals the dimension of the vector spanned by its columns.*

It is called the ***rank of the matrix***

theorem 12.2 textbook

Theorem 41 *an n-dimensional vector space V over a finite field $\mathbb{F}$:*

- *is finite,*
- *has cardinality*

$$card(V) = [card(\mathbb{F})]^n$$

ca me fait chier si ca marche pas

May 13, 2025 — **Lecture 23 : linear codes**

Last Week

Last week, we did linear algebra but in modulo. We use finite field with linear algebra like this:

$$\vec{x} \in \mathbb{F}^n$$

We also work with subspace:

$$S \subseteq \mathbb{F}^n$$

4.3 Vector Spaces and Linear Codes

What and why Linear codes?

- Linear code have more structure

We use that structure to simplify our tasks, notably:

- To determine the code's performance (d_{min} in particular)
- To simplify the encoding
- To simplify the decoding

Linear Code

Definition 35 *A block code is a* ***linear code*** *if the codewords form a subspace of $\mathbb{F}^n$ for some finite field $\mathbb{F}$.*

Example Let $\mathcal{C} \subset \mathbb{F}_2^7$ be the block code that consists of the listed codewords. Is it linear?

Code $\mathcal{C}$
$\vec{v}_1 = 0000000$
$\vec{v}_2 = 0011100$
$\vec{v}_3 = 0111011$
$\vec{v}_4 = 1110100$
$\vec{v}_5 = 1101000$
$\vec{v}_6 = 1001111$
$\vec{v}_7 = 1010011$

We have that:

$$\begin{aligned}
\vec{c}_4 &= \vec{c}_1 + \vec{c}_2 \\
\vec{c}_5 &= \vec{c}_1 + \vec{c}_3 \\
\vec{c}_6 &= \vec{c}_2 + \vec{c}_3 \\
\vec{c}_7 &= \vec{c}_1 + \vec{c}_2 + \vec{c}_3
\end{aligned}$$

Therefore $\mathcal{C} = \text{span}(\vec{c}_1, \vec{c}_2, \vec{c}_3) \subset \mathbb{F}_2^7$ is a linear code (over the finite field $\mathbb{F}_2$).

How to check To check if a code is a linear code, we juste have to check if it is a subspace, is there the zero vector is in.
The number of codeword must be 2^k for some k. (you cannot be a subspace if you are not the cardinality to the power k).
After having check those two, we don't know if it is a subspace, those are only a necessity.
To check we need to find a basis.

Size vs Dimension

We have seen that a k-dimensional subspace of $\mathbb{F}^n$ has cardinality $[\text{card}(\mathbb{F})]^k$. (Count the number of linear combinations you can form with the vectors that form the basis, with coefficients in $\mathbb{F}$)

If the size of a binary block code is not of the form 2^k, then the code is not linear.

Hamming weight (not distance)

Definition 36 *Let $\vec{x} = (x_1, \ldots, x_n)$ be an n-tuple with components in a finite field.*
The ***Hamming weight*** *of $\vec{x}$, denoted $w(\vec{x})$, is the number of its non-zero components in $(x_1, \ldots, x_n)$, i.e.*

$$w(\vec{x}) = d((0, \ldots, 0), (x_1, \ldots, x_n))$$

Exercise (Academic question) In the definition of Hamming weight, we are requiring that the components of $\vec{x}$ take value in a (finite) field $\mathbb{F}$. Why?

Solution If there are not, there is no guarantee that the alphabet contains the 0 element.
Recall that in a finite field $\mathbb{F}$, no matter how we label its elements, one is the 0 element (the identity element with respect to addition). Hence the hamming weight is well defined.

Example

- The weight of $(1, 0, 1, 1, 0)$ is 3
- The weight of $(3, 0, 4, 1, 1, 2)$ is 5

Theorem

Theorem 42 *The minimum distance of a linear code $\mathcal{C}$ is the smallest weight of a codeword in $\mathcal{C}$, zero-vector excluded, i.e.:*

$$d_{min}(\mathcal{C}) = min_{\vec{c}\in\mathcal{C}:\vec{c}\neq\vec{0}} w(\vec{c})$$

note before the proof In the proof that follows, we use the following facts:

- For all $\vec{u}\vec{v} \in \mathbb{F}$

  $$d(\vec{u}, \vec{v}) = w(\vec{u} - \vec{w})$$

  Let us proof this:

  $$d(\vec{u}, \vec{v}) = \sum_{i=1}^{n} d(u_i, v_i)$$
  $$w(\vec{u} - \vec{v}) = \sum_{i=1}^{n} w(u_i - v_i)$$

  Therefore:

  $$d(u_i, v_i) = w(u_i - v_i)$$

  ($\vec{u}$ and $\vec{v}$ are different at position i if and only if $\vec{u} - \vec{v}$ is non-zero at position i)
- Let $f : \mathcal{B} \to \mathbb{R}$ be an arbitrary function and $A \subseteq B$ be finite sets. Then

  $$\min_{x\in\mathcal{A}} f(x) \geq \min_{x\in\mathcal{B}} f(x)$$

  (We might find a smaller minimum if we enlarge the set)

Proof

$$\begin{aligned} d_{min}(\mathcal{C}) &= \min_{\vec{u},\vec{v}\in\mathcal{C}:\vec{u}\neq\vec{v}} d(\vec{u}, \vec{v}) \\ &= \min_{\vec{u},\vec{v}\in\mathcal{C}:\vec{u}\neq\vec{v}} w(\vec{u} - \vec{v}) \\ &\geq \min_{\vec{c}\in\mathcal{C}:\vec{c}\neq\vec{0}} w(\vec{c}) \end{aligned}$$

Which leads to:

$$\begin{aligned} w\min_{\vec{c}\in\mathcal{C}:\vec{c}\neq\vec{0}}(\vec{c}) &= \min_{\vec{c}\in\mathcal{C}:\vec{c}\neq\vec{0}} d(\vec{c},0) \\ &\geq \min_{\vec{c}\in\mathcal{C}:\vec{c}\neq\vec{0}} d(\vec{c},\vec{v}) \\ &= d_{min}(\mathcal{C}) \end{aligned}$$

Example Let us take the code $\mathcal{C} \subseteq \mathbb{F}_2^n$:

$$\mathcal{C} = \{\text{all sequences with an even number of 1's}\}$$

In the yellow universe we have:

$$\{\vec{s} : s_1 + s_2 + \dots s_n = 0\}$$

Then the question is $\dim(\mathcal{C}) = n - 1$, why? We get rid of the yellow universe which is like the “noyau” in French.
Now we search for $d_{min}(\mathcal{C}) = 2$ which is the smallest number (except 0) with have a even number of 1 which is 2.

Here we see that this meets singleton bound, which means that this is a very good code.
In this example, linearity allows us to determine d_{min} via deductive reasoning rather than by inspection.

Exercise if is the subset of $\mathbb{F}_2^n$ that consists of two codewords, namely $(0, \dots, 0)$ and $(1, \dots, 1)$.
Determine k and d_{min}.

Solution The code is linear: it is the subspace of $\mathbb{F}_2^n$ spanned by $(1, \dots, 1)$ (because the $\vec{0}$ is just $0 \cdot (1, \dots, 1)$.
Therefore we can find that $k = 1$. (one vector)
$d_{min} = n$ (the weight of the only non zero vector) (which is always not equal to 1)

The above two codes are not terribly useful, but they have an interesting property shared by the next code which is even less useful:

Exercise again $\mathbb{F}_2^n$ satisfies the definition of a linear code
Determine k and d_{min}.

Solution $k = n$ we can juste take the “vecteur unitaire” and the each one of those vector cannot be written using other vector.
$d_{min} = 1$ which is vectors with only one 1.

The above three code families fulfill the singleton bound with equality. Hence, they are MDS codes, Moreover, they are also linear codes.
No other family of binary linear codes is MDS
No other family of binary linear codes is MDS.
But there are non binary codes that are MDR, e.g. The family of Reed solomon codes (studied next week)

Generator Matrix

Definition

Definition 37 *Let $(\vec{c}_1, \ldots, \vec{c}_k)$ be a basis of a linear code $\mathcal{C} \subset \mathbb{F}^n$ over some finite field $\mathbb{F}$. The $k \times m$ matrix that has $\vec{c}_i$ as its i^{th} row is called* ***generator matrix*** *of $\mathcal{C}$.*

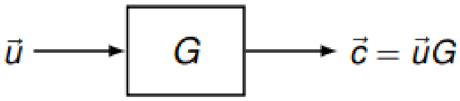
Example $(\vec{c}_1, \vec{c}_2, \vec{c}_3)$ form a basis for $\mathcal{C}$. Therefore:

$$G = \begin{pmatrix} \vec{c}_1 \\ \vec{c}_2 \\ \vec{c}_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Is a generator matrix of $\mathcal{C}$.

Encoding

A $k \times n$ generator matrix specifies an **encoding map** that sends an **information vector** $\vec{u} \in \mathbb{F}^k$ to the corresponding $\vec{c} = \vec{u}G$.



So the way to encode is really to multiply the $\vec{u}$ by the matrix G and **this is** the codeword

Example

$$\vec{u} = (1,0,1) \to \vec{c} = (1,0,1) \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$
$$= (1,1,0,1,0,0,0)$$

A generator matrix is **not unique** you can take for a matrix:

$$G_1 = \begin{pmatrix} \vec{c}_1 \\ \vec{c}_2 \\ \vec{c}_3 \end{pmatrix}$$

Or:

$$\begin{pmatrix} \vec{c}_1 \\ \vec{c}_1 + \vec{c}_2 \\ \vec{c}_1 + \vec{c}_3 \end{pmatrix}$$

Exercise

How many generator matrices for a binary linear code of block length $n = 7$ and dimension $k = 3$.

Solution It is the number of lists that form a basis. A q-ary linear code of dimension k has q^k codewords and the number of bases is:

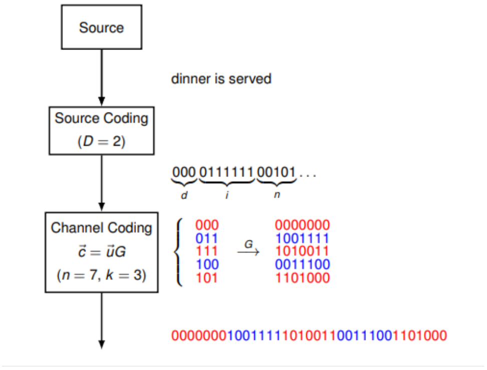
$$\left(q^k - 1\right)\left(q^k - q\right)\ldots\left(q^k - q^{k-1}\right)$$

For a binary code ($q = 2$) we have:

$$\left(2^3 - 1\right)\left(2^3 - 2\right)\left(2^3 - 2\right) = 7 \cdot 64 = 168$$

Transmitter (big picture)

Here we got rid of the crypto part and we assumed we already have nice bits to transfer.



The crypto module would by between the source coding and the channel coding

Exercice

given the code $\mathcal{C}$ (the same as before), is $(\vec{c}_2 + \vec{c}_3, \vec{c}_1 + \vec{c}_2, \vec{c}_1)$ a basis of $\mathcal{C}$? if yes,

- Specify the generator matrix
- Explicitly specify the map $u_1u_2u_3 \rightarrow c_1c_2c_3c_4c_5c_6c_7$

Systematic form

the above matrix G' is in systematic form

Definition 38 *A generator matrix G_s is in* ***systematic form*** *if:*

$$G_s = \left(I_k, P_{k\times(n-k)}\right)$$

Notice that a systematic generator matrix is a matrix in reduced echelon form.
When the generator matrix is in systematic form, each codeword is written as

$$\vec{c} = \vec{u}G_s = (u_1, \ldots, u_k, c_{k+1}, \ldots, c_n)$$

How to find the systematic form

1. Find a basis $\{\vec{c}_1, \ldots, \vec{c}_k\}$ of $\mathcal{C}$
2. Form the generator matrix: $G = \begin{pmatrix} \vec{c}_1 \\ \vdots \\ \vec{c}_k \end{pmatrix}$
3. Row-reduce G (Gaussian elimination on rows) to obtain a matrix in reduced echelon form.

May 14, 2025 — **Lecture 24 : Decoding**

How to decode Decoding is about deciding the information word from the channel output. If the channel output $\vec{y}$ is a codeword, then we assume that it equals the channel input.

In this case decoding is about inverting the encoding map. This is trivial if the generator matrix is in systematic form. (We read out the first k symbols of $\vec{y}$).

But how to know if the channel output is a codeword?

We use the fact that a linear block code, like every subspace of a vector space, can be defined by a system of homogeneous linear equations.

The channel output is a codeword if and only if it satisfies those equations.

First question After having received the word $\vec{y} \in \mathbb{F}^n$ the question is:

$$\text{is } \vec{y} \in \mathbb{F}^n \text{ a codeword?}$$

> *Recall* We recall that
>
> $$S = \left\{\vec{y} : \vec{y}H^T = \vec{0}\right\}$$
>
> Is $\vec{y}$ in the nullspace of H^T? If yes, then it is a codeword.

Example This example is from last lecture.

Given the relation:

$$\begin{cases} c_4 = 3c_1 + 3c_2 \\ c_5 = 2c_1 + 4c_2 \end{cases} \implies \begin{cases} -3c_1 - 3c_2 + c_4 = 0 \\ -2c_1 - 4c_2 + c_5 = 0 \end{cases} \implies \begin{cases} 2c_1 + 2c_2 + c_4 = 0 \\ 3c_1 + c_2 + c_5 = 0 \end{cases}$$

Therefore $\vec{y} \in \mathcal{C}$ if and only if:

$$\begin{cases} 2y_1 + 2y_2 + y_4 = 0 \\ 3y_1 + y_2 + y_5 = 0 \end{cases}$$

Which gives us:

$$\begin{pmatrix} y_1 & y_2 & y_3 & y_4 & y_5 \end{pmatrix} \underbrace{\begin{pmatrix} 2 & 3 \\ 2 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}}_{H^T} = \vec{0}$$

Parity Check Matrix

Parity Check

Definition 39 ***A parity-check Matrix*** *H of a linear (n, k) code is an $(n-k) \times n$ matrix that contains the coefficients of a systems of homogeneous linear equations that defines the code.*

Example

$$\begin{pmatrix} y_1 & y_2 & y_3 & y_4 & y_5 \end{pmatrix} \begin{pmatrix} 2 & 3 \\ 2 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \vec{0} \iff \begin{cases} 2y_1 + 2y_2 + y_4 = 0 \\ 3y_1 + y_2 + y_5 = 0 \end{cases}$$

Which is the same thing just the other way around (There is no fancy things here)

Theorem

Theorem 43 *If $G = (I_k, P)$ where P is a $k \times (n-k)$ matrix, is a generator matrix (in systematic form) of a linear (n, k) block code, then:*

$$H = \left(-P^T, I_{n-k}\right)$$

is a parity-check matrix of the same code.

Proof $H = \left(-P^T, I_{n-k}\right)$ has rank of $n-k$, hence it defines a system of equations, the solution of which is a subspace of $\mathbb{F}^n$ of dimension k.
We want to show that $\vec{u}GH^t = \vec{0}$ for all information vector $\vec{u}$. This is true if and only if GH^T is the zero matrix (of size $k \times (n-k)$).

$$GH^T = \underbrace{(I_k, P)}_{G} \overbrace{\begin{pmatrix} -P \\ I_{n-k} \end{pmatrix}}^{H^T} = -P + P = 0$$

Example

$G = \begin{pmatrix} 1 & 0 & 0 & 3 & 2 \\ 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ is the generator matrix of a $(5, 3)$ code over $\mathbb{F}_5$.
So to find the parity check matrix We can just do:

$$H = \begin{pmatrix} -3 & -3 & 0 & 1 & 0 \\ -2 & -4 & 0 & & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Is a corresponding parity check matrix.

Sub-example Code of all sequences with even number of ones. The H matrix was juste:

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \end{pmatrix}$$

So for k we have it to be equal to $n-1$.
Therefore for the generator matrix if we a k identity matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

La matrice est pas juste

Syndrome

Definition 40 *Let H be the $(n-k)\times n$ parity-check matrix of a linear block code $\mathcal{C} \subset \mathbb{F}^n$ and let $\vec{y} \in \mathbb{F}^n$.*
The ***syndrome*** *of $\vec{y}$ is the vector:*

$$\vec{s} = \vec{y}H^T$$

By definition,

$$\vec{y} \in \mathcal{C} \iff \vec{s} = \vec{0}$$

The syndrome helps us to find the codeword, if when noise comme in the information. If the vector is not all zeros, we know that there has noise in the information, it will reveal to us the error sequence.

4.3.1 Summary

Linear Code Linear code have a specific structure that allows us to:

- Easily understand code performance
- Simplify encoding
- Simplify decoding

Where the codewords forms a subspace of $\mathbb{F}^n$.

1. First the important thing is $d_{min}(\mathcal{C}) = \min_{\vec{c}\neq\vec{0}} w(\vec{c})$
2. The goal is to replace the **whole** encoding map to only the generator matrix (which is smaller to an encoding map).
   For a code $\mathcal{C} \subset \mathbb{F}^n$ with basis $(\vec{c}_1, \dots, \vec{c}_n)$:

$$G = \begin{pmatrix} \vec{c}_1 \\ \vdots \\ \vec{c}_n \end{pmatrix} \in \mathbb{F}^{k\times n}$$

$$\text{where } G = \begin{pmatrix} 1 & 0 & 0 & \mid & p_1 \\ 0 & \ddots & 0 & \mid & p_2 \\ 0 & 0 & 1 & \mid & p_3 \end{pmatrix}, (I_k \mid P) \text{ Which gives us:}$$

$$\vec{u}G = \left(\underbrace{u_1, \ldots, u_k}_{\vec{u}I}, \underbrace{c_{k+1}, \ldots, c_n}_{\vec{u}P} \right)$$

3. To check if the received codeowrd $\vec{y}$ is a codeword we do $\vec{y}H^T = \vec{0}$

Example (Hamming codes) For every integer $m \geq 2$ there exists a binary Hamming code of parameters:

$$n = 2^m - 1$$
$$k = n - m$$

The parity-check matrix is the $m \times n$ matrix whose columns consist of all non-zero vectors of length m. (which means all elements of $\mathbb{F}^m$)
Hamming codes are easy to encode and to decode.

Example (cont.) For instance, let $m = 3$.
A valid parity check matrix is:

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

Where, for convenience, the i^{th} column is the binary representation of i.
The block length is $n = 2^m - 1 = 7$.
The rank of H is m, hence the code dimension is $k = n - m = 4$.
$\vec{c} = (1, 1, 1, 0, \ldots, 0)$ is a codeword because $\vec{c}H^T = 0$.
Hence $d_{min} \leq 3$.
We show that $d_{min} = 3$ by showing how to correct all error patterns of weight 1.

The reason why $\vec{c}$ is a codeword is:
First, why does the codeword is in the nullspace of H^T. We have said it when we introduced the parity check matrix, this matrix **Is** the matrix that contains the system of **homogeneous linear equations** which means something like his: $\vec{y}H^T = \vec{0}$ (you can check the example above). In fact the code is literally defined like this:

$$\mathcal{C} = \left\{\vec{c} \in \mathbb{F}_2^n : \ H\vec{c}^T = \vec{0}\right\}$$

Which is the kernel of the subspace.
For the vector of our example we can just compute all the matrix, but you don't have to because: you can get rid of the 4 last rows (columns in H) because our vector $\vec{c}$ is always null after the third column. Therefore for the first column you get $1 + 1 = 0$ same thing for the second and $0 + 0 + 0 = 0$ for the third. Which gives only zeros.
Because the weight of the vector $\vec{c} = 3$ we can easily say that $d_{min} \leq 3$ because it is upper bound by $\vec{c}$.
As said in the course, to correct a 1 bit error, you must have $d_{min} \geq 3$. and because we had shown that $d_{min} \leq 3$ the only way is $d_{min} = 3$.

And now you can see how it works because independently of the last 4 rows, which gives us that the information is "still" here after the encoding.

Example

Let $\vec{y} = \vec{c} + \vec{e}$ be the channel output, where $\vec{c} \in \mathcal{C}, \vec{e} \in \mathbb{F}_2^n$ and $w(\vec{e}) = 1$.

$$\vec{s} = \vec{y}H^T = \vec{c}H^T + \vec{e}H^T = \vec{e}H^T$$

$\vec{s} = \vec{e}H^T$ is the binary representation of the position of the error.
Hence we can correct the error.
For instance let's take a canonical vector $\vec{e}_i = \left(0, 0, \dots, \overbrace{1}^{i} \dots, 0, 0\right) H^T$, this will give us the i^{th} row of $H^T c$ or the i^{th} column of H.

May 20, 2025 — **Lecture 25 : Still linear codes**

Minimum distance over the partiy check matrix

Theorem 44 *Let H be any parity check matrix of a linear code. The minimum distance of the code is the smallest positive integer d such that there are d coluns of H that are linearly dependent.*

In other words This means that, if you take d_{min}, then there exists some column that are linearly dependent. (I think this was on the exercise set of last week)

Proof For a linear code, the minim distance is the smallest weight of a non zero vector codeword. Let $\vec{c} \neq \vec{0}$ be a codeword with the smallest weight d_{min}. The fact that $\vec{c}H^T = \vec{0}$ proves that H has d_{min} linearly dependent columns.
We need to argue that fewer than d columns of H are not linearly dependent. Suppose that H has $t < d$ linearly dependent columns. THen we could find a non zero codeword $\vec{c}$ of weight smaller than d such that $\vec{c}H^T = 0$ which is a contradiction.

for the first fact, $\vec{c}H^T = 0$ proves that H has d linearly dependent columns. You can see it first like this. If you take a vector with a weight of d then you know that if you take the multiplication like this: $\vec{c}H^T$ you multiply every columns of H by an element of $\vec{c}$ (like each column of H^T is multiplied by a scalar (not a litteraly scalar in $\mathbb{R}$)) therefore you have like imagine you vector lie this:

$$\vec{c} = \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix}$$

Then if you take the multiplication said before you get something like this:

$$c_1 \cdot \text{col}_1 + \overbrace{c_2}^{=0} \cdot \text{col}_2 + c_3 \cdot \text{col}_3 = 0$$

But because we know the weight is d (here 2) then you know that there is d column such that a non-trivial equation gives 0, therfore, there are linearly dependent.

Example The following is a parity check matrix for a Hamming code of parameters $n = 7$, $k = 4$.

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

Clearly no two columns are linearly dependent. Column, 1, 2 and 3 are linearly dependent.
Hence $d_{min} = 3$.

Standard Array: Background Material

Equivalence Relation (review)

- $\mathcal{G}$ a set
- $\sim$ an equivalence relation on $\mathcal{G}$
- $[a]$ the equivalence classe of $a \in \mathcal{G}$

The key property that we will use is: an equivalence relation on a set partitions (sèpare) the set into disjoint equivalence classes.

Example For instance:

- $\mathcal{G}$ is the set of all students in Switzerland
- $a \sim b$ if a and b attend the same university
- $[a]$ is the subset of all student that attended the same univeristy as a.

> As in the above example, equivalence class doesn't have to be the same cardinality

Special case: Group-Theoretic construction

When $\mathcal{G}$ forms a commutative group $(\mathcal{G}, \cdot)$and $(\mathcal{H}, \cdot)$ is a subgroup, there is a natural choice for $\sim$ defined as follows:

- $a \sim b$ if there exists an $h \in \mathcal{G}$ such that $b = a \cdot h$

Equivalently:

$$a \sim b \text{ if } a^{-1} \cdot b \in \mathcal{H}$$

Claim: the above $\sim$ is indeed an equivalence relation.

Proof

- (reflexive) $a \sim a$
  True because $a^{-1} \cdot a \in \mathcal{H}$
- (symmetric) if $a \sim b$ then $b \sim a$:
  this is true because if you take $a^{-1} \cdot b = h \in \mathcal{H}$ then $b^{-1} \cdot a = h^{-1} \in \mathcal{H}$ (this just use the definition saif before.)
- (transitive) if $a \sim b$ and $b \sim c$ then $a \sim c$:
  True because if $a^{-1} \cdot b = h \in \mathcal{H}$ and $b^{-1} \cdot cahh_2 \in \mathcal{H}$, then $\mathcal{H}$ contains also $h_1 \cdot h_2$ which is $a^1 \cdot \overbrace{b \cdot b^{-1}}^{=1} \cdot c = a^{-1} \cdot c$

Since in this case an equivalence class has the form:

$$[a] = \{a \cdot h : h \in \mathcal{H}\}$$

It also make sens to write:

$$[a] = a \cdot \mathcal{H}$$

Example Let $(\mathcal{G}, +) = (\mathbb{Z}/10\mathbb{Z}, +)$ and let $\mathcal{H} = \{0, 5\}$.
Then $(\mathcal{H}, +)$ is a subgroup of $(\mathcal{G}, +)$ and the equivalence classes

are;

$$\begin{aligned}[0] &= \mathcal{H} = \{0,5\}\\ [1] &= 1+\mathcal{H} = \{1,6\}\\ [2] &= 2+\mathcal{H} = \{2,7\}\\ [3] &= 3+\mathcal{H} = \{3,8\}\\ [4] &= 4+\mathcal{H} = \{4,9\}\end{aligned}$$

In the group theoretic language, $[a]$ is called the coset of $\mathcal{H}$ with respect to a.

Claim

Theorem 45 *All closets of $\mathcal{H}$ have the same cardinality card($\mathcal{H}$).*

Proof

let $h_1, h_2 \in \mathcal{H}$ such that $h_1 \neq h_2$ then we know that $a \cdot h_1 \neq a \cdot h_2$. this shows that this is like an injective function, which leads that $a \cdot \mathcal{H}$ has the same cardinality as $\mathcal{H}$[a].

[a] I don't really understand this, so how I did it is by using the definition (we assume that $f(h_1) = f(h_2)$ which implies that $ah_1 = ah_2$, and then $a^{-1}(ah_1) = a^{-1}(ah_2) \implies h_1h_2$). The surjective part is that every element of $a\mathcal{H}$ is of form $a \cdot h$, so every thing in $a\mathcal{H}$ is hit by our mapping, therefore is has the same cardinality (bijection)

Here is the group and subgroup of interest to us:

- The group $(\mathcal{G}, \cdot)$ is $(\mathbb{F}^n, +)$ for some finite field $\mathbb{F}$ and a positive integer n
- The subset $\mathcal{H}$ is a linear code $\mathcal{C} \subset \mathbb{F}^n$
- Then if $x, y \in \mathbb{F}^n, x \sim y$ if and only if $-x + y \in \mathcal{C}$
- (equivalently, $x \sim y$ if and only if $y = x + c$ for some $c \in \mathcal{C}$)

Standard Array

Definition 41 *The* ***standard array*** *is an array that has the elements of $\mathcal{C}$ in the top row, starting with the 0 codeword, and each row forms a coset of $\mathcal{C}$, Each element of $\mathbb{F}^n$ shows up exactly once in the standard array.*

$$\left|\begin{array}{ccccc} c_0 = 0 & c_1 & c_2 & \dots & c_{M-1} \\ t_1 & t_1 + c_1 & t_1 + c_2 & \dots & t_1 + c_{M-1} \\ t_2 & t_2 + c_1 & t_2 + c_2 & \dots & t_2 + c_{M-1} \\ \vdots & \vdots & \vdots & & \vdots \\ t_{L-1} & t_{L-1} + c_1 & t_{L-1} + c_2 & \dots & t_{L-1} + c_{M-1} \end{array}\right| \begin{array}{c} \leftarrow [\mathcal{C}] \\ \leftarrow [t_1] \\ \leftarrow [t_2] \\ \vdots \\ \leftarrow [t_{L-1}] \end{array}$$

Where for each $j = 1, \dots, L-1$ t_j is such that:

$$t_j \notin \left(\mathcal{C} \bigcup_{k=1}^{j-1} [t_k]\right)$$

Later we will choose the coset leader more carefully

Decoding regions

Suppose that card$(\mathcal{C}) = M$
Think of the decoder as being specified by M decoding regions $\mathcal{D}_0, \dots, \mathcal{D}_{M-1}$ that partion $\mathbb{F}^n$:

$$\mathcal{D}_1 \bigcap D_J = \emptyset \text{ if } i \neq j$$

$$\bigcup_{i=0}^{M-1} \mathcal{D}_i = \mathbb{F}^n$$

Upon observing $y \in \mathbb{F}^n$, the decoder finds the i such that $y \in \mathcal{D}_i$ and declares;

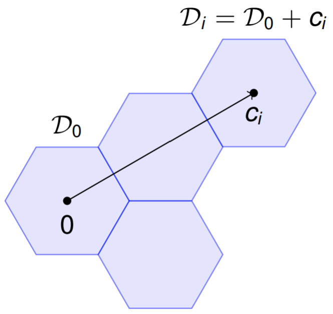
$$\hat{\boldsymbol{c}} = c_i$$

The coset decoder: how to decode with the standard Array

$$\begin{array}{|ccccc}
c_0 = 0 & c_1 & c_2 & \dots & c_{M-1} \\
t_1 & t_1 + c_1 & t_1 + c_2 & \dots & t_1 + c_{M-1} \\
t_2 & t_2 + c_1 & t_2 + c_2 & \dots & t_2 + c_{M-1} \\
\vdots & \vdots & \vdots & & \vdots \\
t_{L-1} & t_{L-1} + c_1 & t_{L-1} + c_2 & \dots & t_{L-1} + c_{M-1} \\
\hline
\uparrow & \uparrow & \uparrow & \dots & \uparrow \\
\mathcal{D}_0 & \mathcal{D}_1 & \mathcal{D}_2 & & \mathcal{D}_{M-1}
\end{array}$$

Note that $\mathcal{D}_i = c_i + \mathcal{D}_0$

Geometrical Interpretation



The union of all $\mathcal{D}_i$ is $\mathbb{F}^n$. Hence every $y \in \mathbb{F}^n$ is in exactly one decoding region.

How to implement

To find the codeword associated to a channel output y, we could find y in the standard array and read out the entry on top of the same column.
Storing the whole standard array is impractical (often impossible for large codes).
The first column describes the geometry of all decoding regions. We should be able to leverage on that.

Claim

Theorem 46 *In the standard array, each element of a row has the same syndrome as the coser leader.*

proof

- The elements of $[t_i]$ have the form $t_1 + c$ for some $c \in \mathcal{C}$
- The syndrome of such element is:

$$(t_1 + c)H^T = t_1 H^T + cH^T = t_i H^T$$

Which is the syndrome of the coset leader t_i.

Uniquely identifier

Theorem 47 *The syndrome uniquely identiies the coset leader*

Proof Let t_i and t_j be coset leaders
Suppose that $t_i H^T = t_j H^T$, then we know that $(t_i - t_j)H^T = 0$, then by definition we know that $t_i - t_j = c_k \in \mathcal{C}$
And because $t_i - t_j \in \mathcal{C}$ this implies directly that $t_i \equiv t_j \bmod \mathcal{C} \implies t_i + \mathcal{C} = t_j + \mathcal{C}$ which means they are both in the same coset. Since both are in the same coset and both are coset leaders, $t_i = t_j$.

Theorem

Theorem 48 *$\vec{y}_1$ and $\vec{y}_2$ have the* ***same syndrome*** *if and only if they are in the same coset*

The is the theorem 46th with the theorem 47th.

Proof Because this theorem is the theorem 46 $\iff$ theorem 47 kind of the proof is juste the two proofs together.
In the second condition we say that having the same coset implies the same syndrome.

May 21, 2025 — **Lecture 26 : almost the end**

How to decode

Hence the coseet decoder can be implemented as follows:

1. We precompute and store the coset leaders and the corresponding syndrome
2. To decode y, we compute its syndrom $s = yH^T$
3. s encodes the row of y
4. We use the lookup table to determine the corresponding coset leader, say, t_i
5. t_i and y uniquely determine the column of y namely $y = t_i + c_j$
6. hence $c_j = y - t_i$
7. the decoder declares

Choosing coset leaders

Rule "MD" In every coset, the leader is the minimum weight sequence.

Theorem 49 *The coset decoder is a minimum distance decoder*

Proof Suppose $\vec{y}$ is in row i , column j of the standard array. We know then $\vec{y} = \vec{c}_j + \vec{t}_i$.
Then we know that the coset decoder will:

- decode: $\vec{y} \to \vec{c}_j$
- Hence $d(\vec{y}, \vec{c}_j) = w\left(\vec{t}_i\right)$

Now consider any $\vec{c}_k$:

$$\begin{aligned} d(\vec{y}, \vec{c}_k) &= w(\vec{y} - \vec{c}_k) \\ &= w\left(\vec{c}_j + \vec{t}_i - \vec{c}_k\right) \\ &= w\left((\vec{c}_k - \vec{c}_k) + \vec{t}_i\right) \end{aligned}$$

Now the question is: who is the leader of this?
We know because of $\vec{t}_i$ we know that we are in the row i.
Then we want to show that $d(\vec{y}, \vec{c}_k) \geq w\left(\vec{t}_i\right)$.
By construction, we choose the coset leader by the one who is the lowest weight. Then as we said before we know that $d(\vec{y}, \vec{c}_k) = w\left(\vec{c}_j - \vec{c}_k + \vec{t}_i\right)$ which is also in row i. And because $\vec{t}_i$ is the lowest weight then it has to be lower than $d(\vec{y}, \vec{c}_k)$.

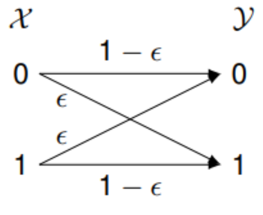
Example ((6, 3) binary linear block code) The code is defined by the following parity check matrix:

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

We can see here that two columns are linearly dependent. The first three columns are linearly dependent. Hence $d_{min} = 3$.
H hat the form (P, I), The generator matrix G has the form $\left(I, -P^T\right)$

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Error probability There are many ways to choose $\mathcal{D}_0$. In fact every element of every row of the standard array can be chosen as the coset leader. Next, we learn how to choose the coser leaders so as to minimize the decoding error-probability.
Suppose that the channel is the following binary symmetric channel with input alphabet $\mathcal{X} = \{0, 1\}$ and output alphabet $\mathcal{Y} = \{0, 1\}$. (Binary symmetric channel)



We tale $\vec{e}$ a string of **independent** binaries with $P(e_i = 1) = \varepsilon$, hence what is:

$$\begin{aligned} P(\vec{e}) &= (0,1,0,0,1,0,1) = p(e_i = 0)p(e_2 = 1)\dots \\ &= (1-\varepsilon)\varepsilon(1-\varepsilon)\dots \\ &= (1-\varepsilon)^{\text{nbr of 0}} \cdot \varepsilon^{\text{nbr of 1}} \\ &= (1-\varepsilon)^4\varepsilon^3 \end{aligned}$$

We know that in a sequence $\vec{e}$ the number of one:

$$nbr_1 = w(\vec{e})$$

And for the number of zeroes:

$$nbr_0 = n - w(\vec{e})$$

Then we can see that:

$$\boxed{\begin{aligned} P(\vec{e}) &= (1-\varepsilon)^{n-w(\vec{e})}\varepsilon^{w(\vec{e})} \\ &= (1-\varepsilon)^n\left(\frac{\varepsilon}{1-\varepsilon}\right)^{w(\vec{e})} \end{aligned}}$$

In a lot of exercise and exam question, it is easier to use the probability the we decode **correctly** the given code.
We start by taking $\vec{c}_0 = \vec{0}$, we then look for $P_c(0) = P(\text{error pattern was “good”})$ Then:

$$\begin{aligned} &= p(\text{ error pattern is one from the column below}\vec{c}_0) \\ &= p(\vec{e} \in \mathcal{D}_0) \\ &= \sum_{e \in \mathcal{D}_0} p(\vec{e}) \\ &= \sum_{\vec{e} \in \mathcal{D}_0} (1-\varepsilon)^{n-w(e)}\varepsilon^{w(e)} \end{aligned}$$

In other words, when $c_0 \in \mathcal{C}$ is transmitted, the decoder makes the correct decision whenever $y \in \mathcal{D}_0$. But when c_0 is transmitted, the event $y \in \mathcal{D}_0$ is the same as the event $e \in \mathcal{D}_0$.

What we do to comput the probability we do the sum over all elements of the column with the exact weight... This can be said like this:

$$P_C(0) = \sum_{j=0}^{L-1} \varepsilon^{w(t_j)}(1-\varepsilon)^{n-w(t_j)}$$

Next

Now, what if $\vec{c}_i$ is the transmitted codeword?

$$\begin{aligned} P_c(i) &= P(\vec{y} \in \mathcal{D}_i) \\ &= P(\vec{c}_i + \vec{e} \in \mathcal{D}_i) \qquad\qquad = P(\vec{c}_i + \vec{e} \in \vec{c}_i + \mathcal{D}_0) \\ &= P(\vec{e} \in \mathcal{D}_0) \end{aligned}$$

What is the probability to make an error if it is 111000
We have assumed for simplicity a binary symmetric channel, but the same idea applies to nonbinary channel (with input and output alphabet $\mathbb{F}$) for which the probability of an error pattern $e \in \mathbb{F}^n$ is decreasing with $w(e)$
$P_C(c_i)$ is the probability that $y \in \mathcal{D}_i$ when c_i is transmitted
This is the probability that $\underbrace{c_i + e}_{y} \in \underbrace{c_i + \mathcal{D}_0}_{\mathcal{D}_i}$.
Hence, for all $i, P_C(c_i) = P_C(0)$
Hence, the unconditional probability of correct decoding is $P_C = P_C(0)$

4.4 Reed Solomon Codes

Polynomials over finite fields

Not surprisingly, the notion of polynomial extends to finite fields.

- let $\vec{u} = (u_1, \ldots, u_k)$ for some finite field F
- We associate to $\vec{u}$ the polynomial:

$$P_{\vec{u}}(x) = u_1 + u_2 x + \cdots + u_k x^{k-1}$$

- $P_{\vec{u}}(x)$ can be evaluated at any $x \in \mathbb{F}$
- The degree of a polynomial is the highest exponent i for which x^i has a non-zero coefficient.
- By convention, the zero polynomial has degree $-\infty$.

Example

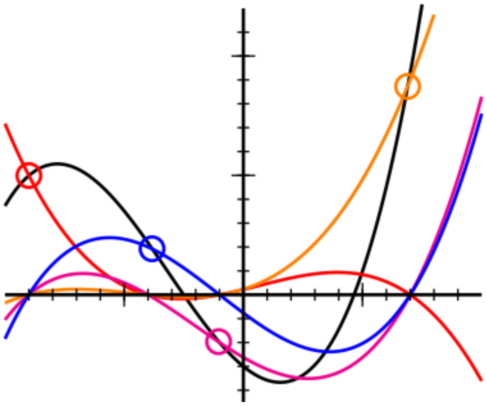
- $\mathbb{F} = \mathbb{F}_5$
- $P(x) = 2 + 4x + 3x^2$ is a polynomial of degree 2 over $\mathbb{F}$
- $P(x) = P_{\vec{u}}(x)$ for $\vec{u} = (2, 4, 3) \in \mathbb{F}_5^3$
- A polynomial $P(x)$ over a field $\mathbb{F}$ can be evaluated at any $x \in \mathbb{F}$:

$$P_{\vec{u}}(2) = 2 + 4 \cdot 2 + 3 \cdot 2^2 = 2 + 3 + 2 = 2$$

Interpolation via polynomials

Problem: given a field $\mathbb{F}$ and k pairs $(a_i, y_i) \in \mathbb{F}^2$, where the a_i are all distinct is there a polynomial $P(X)$ over $\mathbb{F}$ of degree at most $k-1$ (hence described by at most k coefficients) such that:

$$P(a_i) = y_i, \;\; i = 1, \ldots, k$$



Lagrange's interpolation polynomials

To simplify notation, we demonstrate how it works by means of examples:

> *Example*
>
> - Fix a field $\mathbb{F}$ and distinct field elements a_1, a_2, a_3 as well as y_1, y_2, y_3 (not necessarily distinct)
> - We seek polynomial $P(x)$ of degree at most 2 and coefficient in $\mathbb{F}$ such that $P(a_i) = y_i$
> - Suppose we can find a polynomial $Q_1(x)$ of degree at most 2 such that:

4.4.1 Reed Solomon

May 27, 2025 — **Lecture 27 : Reed-Solomon**

Polynomial

Definition 42 *Let $\mathcal{K}$ be a finite field. A polynomial P with coefficient in $\mathcal{K}$ is a mapping $\mathcal{K} \to \mathcal{K}$ of the form:*

$$X \to P(X) = a_1 + a_2 X + \ldots + a_{m+1} X^m$$

Where $a_1, \ldots, a_{m+1}$ is a sequence of elements in $\mathcal{K}$.
The power of the polynom is the greatest power affected by a non-zero elements.

For all sequence $\vec{u} = (u_1, \ldots, u_k) \in \mathcal{K}^k$ of k elements of $\mathcal{K}$, we called $P_{\vec{u}}$ the polynom where the coefficient are $u_1, \ldots, u_k$ by increasing power. In other terms:

Definition 43

$$P_{\vec{u}}(X) \underbrace{=}_{def} u_1 + u_2 X + u_3 X^2 + \cdots + u_k X^{k-1}$$

Therefore, $P_{\vec{u}}$ is of degree $k - 1$.

We can now define the Reed-Solomon Code:

Reed-Solomon

Definition 44 *Let n and k be integers with $1 \leq k \leq n$. A Reed-Solomon code with the parameters (n, k) is defined as:*

- *The alphabet is a finite field $\mathcal{K}$ of cardinality $\geq n$*
- *Choosen n disctinct elements of $\mathcal{K}$, $a_1, a_2, \ldots, a_n$. A sequence of symbols $\vec{u} = (u_1, \ldots, u_k) \in \|^k$ is encoded in the n sequence symbols $\vec{x} = (x_1, \ldots, x_n) \in \mathcal{K}^n$ defined by:*

$$x_i = P_{\vec{u}}(a_i) \quad \forall i, 1 \ldots n$$

The Reed Solomon $\mathcal{C}$ is the subset of all the encoding $\vec{x}$ possible, for all $\vec{u} \in \mathcal{K}^k$. This is therefore, a codeword of length n.

Example

Let us build a Reed Solomon code on $\mathbb{F}_5$. Let us choose the maximum possible $n = 5$. (all the elements of $\mathbb{F}_5$).
We also have to choose 5 elements of $\mathbb{F}_5$, which is all the elements of $\mathbb{F}_5$ $(a_1 = 0, a_2 = 1, \ldots, a_5 = 4)$
Let us compute the encoding. For instance if the message is $\vec{u} = (0, 0)$ the polynomial to evaluate is $P_{\vec{0}} = 0$, therefore $P_{\vec{0}}(a_1) = \cdots = P_{\vec{0}}(a_5) = 0$, and the codeword corresponding to it is $\vec{x} = (0, 0, 0, 0, 0)$.
If $\vec{u} = (3, 2)$, then $P_{\vec{u}}(X) = P_{32}(X) = 3 + 2X$ then:

$$\begin{aligned} P_{\vec{u}}(a_1) &= P_{\vec{u}}(0) = 3 \\ P_{\vec{u}}(a_2) &= P_{\vec{u}}(1) = 0 \\ P_{\vec{u}}(a_3) &= P_{\vec{u}}(2) = 2 \\ P_{\vec{u}}(a_4) &= P_{\vec{u}}(3) = 4 \\ P_{\vec{u}}(a_5) &= P_{\vec{u}}(4) = 1 \end{aligned}$$

Therefore having $\vec{u} = (3, 2)$, we get the encoding:

$$\vec{x} = (3, 0, 2, 4, 1)$$

By doing this for the 25 sequence possible of $k = 2$ elements of $\mathbb{F}_5$ we obtain the table of encoding:

$\vec{u}$	$P_{\vec{u}}(X)$	$\vec{x}$
00	0	00000
01	X	01234
02	$2X$	02413
03	$3X$	03142
04	$4\ X$	04321
10	1	11111
11	1 + X	12340
12	1 + 2X	12024
13	1 + 3X	14203
14	1 + 4X	10432
20	2	22222
21	2 + X	23401
22	2 + 2X	24130
23	2 + 3X	20314
24	2 + 4X	21043
30	3	33333
31	3 + X	34012
23	3 + 2X	30241
34	3 + 4X	32104
40	3	44444
41	4 + X	40123
42	4 + 2X	41302
43	4+3X	42031
44	4 + 4X	43210

Property

We are going to see now that Reed-Solomon Code have **great** property. To be able to remark this, we need a fondamental result of polynomial, which say that a polynomial has a number of root smaller or equal at his degree.

Theorem 50 *Let $\vec{u} \in \mathcal{K}^k$ where $\mathcal{K}$ is an abelian group. The polynomial $P_{\vec{u}}$ is of degree $k-1$. If there exists k elements all distinct $a_1, \dots, a_k$ of $\mathcal{K}$ such that $P_{\vec{u}}(a_i) = 0$, then $\vec{u} = \vec{0}$*

This is saying that you cannot write your polynomial with more this $(x-a)$ than k times.

Proof

Here we are doing the proof on finite field which is enough for us (but the real theorem is true even on abelian group $\mathcal{K}$ which is infinite).

Let φ the mapping $\mathcal{K}^k \to \mathcal{K}^k$ which has $\vec{v} \in \mathcal{K}^k$ abritrary being associated with $(P_{\vec{v}}(a_1), \dots, P_{\vec{v}}(a_k))$. The hypothesis is that if $\varphi(\vec{u}) = \vec{0}$ we want to show that $\vec{u} = \vec{0}$.

To do so, we will show that φ is injective. By the pigeon hole principle, because the set of départ and the set d'arrivée, are finite and of same cardinality, we juste have to prove that it is surjective, which is what we will do.

Let $(x_1, \dots, x_k) \in \mathcal{K}^k$ be an arbitrary vector. Let us look for a polynomial P of degree $\leq k-1$ such that

$(P(a_1) = x_1, \dots, P(a_k) = x_k)$. Such a problem is called an interpolation: the value of x_i and the point of evaluation a_i are known and we need to find the polynomial d'interpolation of **Lagrange**. Let $Q_i, i = 1, \dots$ the polynomial defined by:

$$Q_1(X) = \frac{(X - a_1)\cdots(X - a_{i-1})(X - a_{i+1})\cdots(X - a_k)}{(a_i - a_1)\cdots(a_i - a_{i-1})(a_i - a_{i+1})\cdots(a_i - a_k)}$$

Here what we are doing is skipping the a_i coefficient and putting all the zero possible for all other a_n. This is the reason why for every element except a_i $Q_i(X) = 0$.
The reason why it is 1 when $X = a_i$ is:
If you take the polynomial at a_i, you get:

$$Q_1(X) = \frac{(a_i - a_1)\cdots(a_i - a_{i-1})(a_i - a_{i+1})\cdots(a_i - a_k)}{(a_i - a_1)\cdots(a_i - a_{i-1})(a_i - a_{i+1})\cdots(a_i - a_k)} = 1$$

This is a polynomial of degree $\leq k-1$ (the product of $k-1$ termes of degree 1) and it has the property that $Q_i(a_i) = 1$ and $Q_i(a_j) = 0$ for $i \neq j$.
Soit maintenant P be the polynomial defined by:

$$P(X) = x_1 Q_1(x) + \cdots + x_k Q_k(X)$$

Such that P is a polynomial of degree $\leq k-1$ (because we are doing the sum of all the polynomial of degree $k-1$).
Let v_1 = the coefficient of degree 0 of P, etc … v_k =the coefficient of de degree $k-1$ of P, such that $P = P_{\vec{v}}$ and then $\psi(\vec{v}) = \vec{x}$.
We have then shown that ψ is surjective.
Therefore, φ is injective and $\varphi\left(\vec{0}\right) = \vec{0}$. Finally, $\vec{u}$ and $\vec{0}$ has the same image by φ which leads that $\vec{u} = \vec{0}$.

Linearity of Reed-Solomon codes:

In fact, the reed Solomon code are linear codes:

Theorem 51 *A Reed-Solomon codes of parameters (n, k) is a linear blockcode of length n and dimension k.*

In order to prove this, we have to show that the subspace $\mathcal{C}$ is a vector space of $\mathcal{C}^n$

Proof To prove this we will do as we did in linear algebra:
Let $\vec{u}, \vec{v} \in \mathbb{F}^k$ and $\alpha \in \mathbb{F}$.Notice that:

$$P_{\alpha\vec{u}}(x) = \alpha u_1 + \alpha u_2 x + \cdots + \alpha u_k x^{k-1} = \alpha P_{\vec{u}}(x)$$

For the second clause:

$$P_{\vec{u}+\vec{v}}(x) = (u_1 + v_1) + (u_2 + v_2)x + \cdots + (u_k + v_k)x^{k-1} = P_{\vec{u}}(x) + P_{\vec{v}}(x)$$

Hence:

$$P_{\alpha\vec{u}+\vec{v}}(x) = \alpha P_{\vec{u}}(x) + P_{\vec{v}}(x)$$

Then, is we have the following mapping

$$\vec{u} \to \vec{x} \in \mathcal{C}$$
$$\vec{v} \to \vec{y} \in \mathcal{C}$$

Then we can say that

$$\alpha\vec{u} + \vec{v} \to \alpha\vec{x} + \vec{y} \in \mathcal{C}$$

Design-parameters k

Theorem 52 *The design parameter k is indeed the dimension of the RS-code.*

Proof It suffices to prove that the map is injective (one to one) implying that there are $[\text{card}(\mathbb{F})]^k$ disctinct codewords, hence that k is the dimension of the code.
When we proved the fundamental theorem of algebra, we showed that the map:

$$\psi : \mathbb{F}^k \to \mathbb{F}^k$$
$$\vec{u} \to (P_{\vec{u}}(a_1), \dots, P_{\vec{u}}(a_k))$$

is one to one.
This garantese the encoding map:

$$\mathbb{F}^k \to \mathcal{C}$$
$$\vec{u} \to (P_{\vec{u}}(a_i), \dots, P_{\vec{u}}(a_k), P_{\vec{u}}(a_{k+1}), \dots, P_{\vec{u}}(a_n))$$

is one to one

Reed solomon Code are MDS

Theorem 53 *Reed Solomon code are MDS*

proof We want to prove that $d_{min} = n - k + 1$:

- Let $\vec{u} \in \mathbb{F}^k$ be a non zero information vector. $P_{\vec{u}}(x)$ is a non zero polynomial of degree at most $k-1$. Hence it has at most distinct $k-1$ roots.
- The corresponding codeword $\vec{c}$ (obtained by evaluating $P_{\vec{u}}(x)$ at n distince values) has at most $k-1$ zeros
- Hence $w(\vec{c}) \geq n(k-1) = n - k + 1 \quad \forall\vec{c}$
- Since $\vec{c}$ is an arbitrary non zero codeword and RS code are linear code

$$d_{min} \leq n - k + 1$$

- By the Singleton's bound, $d_{min} \leq n - k + 1$

Hence $d_{min} = n - k + 1$

Reed Solomon is goated

Theorem Textbook 14.3

Theorem 54 *A Reed Solomon code with design parameters k and n is linear (n, k) code of minimum distance $d_{min} = n - k + 1$, i.e. it attains singleton's bound with equality*

Note that the condition:

$$|\mathbb{F}| \geq n$$

Is necessary or else we can't find n discting field elements $a_1, \ldots, a_n$.

Exercise

Find a generator matrix G for this code over $\mathbb{F}_3$.
Here because we are in F_3 we know that $n = 3$ to be able to have a_1, a_2, a_3 distinct elements ($|\mathbb{F}_3| = 3$), we then take $k = 2$. We have here 9 distincts vectors and each has a mapping like this:

$\vec{u}$	$P_{\vec{u}}(x)$	$\vec{c}$
00	0	000
01	x	012
02	2x	021
1height0	1	111
11	1 + x	120
12	1 + 2x	102
20	2	222
21	2 + x	210
22	2 + 2x	210

To make the matrix G we need to find two codewords that are linearly independent
Here are a few choices:

$$G = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad G = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 0 & 1 \end{pmatrix} \quad G = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & 2 \end{pmatrix}$$

We just recall that each generator matrix defined an input output map.
The map that we originally used to define RS coes:

$$\vec{u} \to P_{\vec{u}}(x) \to \vec{c} = P_{\vec{u}}(a_1), \ldots, P_{\vec{u}}(a_n)$$

is just one of many possible maps. (each G gives a different mapping (like $\vec{u}G_1 \neq \vec{u}G_2$ but still in the same $\mathcal{C}$).
In fact there are $\left(q^k - 1\right)\left(q^k - q\right) \cdots \left(q^k - q^{k-1}\right)$ maps for a linear code of dimension k over $\mathbb{F}_q$.

May 28, 2025 — **Lecture 28 : Last lecture**

I juste wanted to make it juste clearer what is lagrange's interpolation polynomials (because in the book it isn't really mentionned):

Lagrange's interpolation polynomials

Interpolation via polynomials

Problem Given a field $\mathbb{F}$ and k pairs $(a_i, y_i) \in \mathbb{F}^2$, where the a_i are all distinct, is there a polynomial $P(X)$ over $\mathbb{F}$ of degree at most $k-1$ (hence described by at most k coefficients) such that:

$$P(a_i) = y_i, \quad i = 1, \ldots k$$

The answer is yes, and we obtain those via Lagrange's interpolation polynomials.
This is the same process that we explained in the proof of yeasterday (in the remark). For instance let us take some examples:

Example We first fix a field $\mathbb{F}$ and disctinct field elements a_1, a_2, a_3 as well as y_1, y_2, y_3 (not necessarily distinct).
We are looking for a polynomial $P(X)$ of degree at most 2 $(k-1)$ and coefficients in $\mathbb{F}$ such that $P(a_i) = y_i$.
Suppose we can find a polynomial $Q_1(x)$ of degree at most 2, such that:

$$Q_1(x) = \begin{cases} 1, & x = a_1 \\ 0, & x = a_i \neq a_1 \end{cases}$$

(which is the same thing for our $Q_i(X)$ before)
Now we suppose that $Q_2(x)$ behaves the same as for a_2 and the same for $Q_3(x)$ and a_3.
Then the desired polynomial is

$$P(X) = y_1 Q_1(x) + y_2 Q_2(x) + y_3 Q_3(x)$$

Now if we want to construct the first $Q(X)$ this is then:

$$Q_1(X) = \frac{(x-a_2)(x-a_3)}{(a_1-a_2)(a_1-a_3)}$$

We do the same for the two other Q_i.

Concrete example Over $\mathbb{F}_5$ find the polynomial $P(X)$ of degree not xceeding 2 for which $P(a_i) = y_i$ and this for:

i	(a_i, y_i)
1	$(2, 3)$
2	$(1, 0)$
3	$(0, 4)$

To do so, we then will use the same process as before:

$$P(x) = y_1 Q_1(x) + y_2 Q_2(x) + y_3 Q_3(x)$$

We then construct the Q_i:

$$\begin{aligned} P(x) &= 3\left(\frac{(x-1)x}{(2-1)2}\right) + 0 \cdots + 4\frac{(x-2)(x-1)}{(4-2)(4-1)} \\ &= 3(x-1)x + 3(x-2)(x_1) \\ &= 4x^2 - 4x + 2x^2 - 6x + 4 \\ &= x^2 + 4 \end{aligned}$$

Conclusion It should be obvious from the above example that we can proceed similarly for any field $\mathbb{F}$ for any positive integer k, and for any given set of k points with components in $\mathbb{F}$.

Fundamental theorem of Algebra I already wrote this before but too much is better than not enough:

Theorem 55 *Let $P(x)$ be a polynomial of degree at most $k-1$ over a field. if $P(x) \neq 0$ then the number of its distinct roots is at most $k-1$.*

Example You can see this it for polynomial of degree 1: $ax + b$ this is already a "root" so the number is one.
For a polynomial of degree 2 we can go up to two roots, three for cubic polynomial...

I already done the proof above in lecture 26.

4.5 Quiz 5

I found this quiz kind of hard so I wanted to do a recap to have it again if I needed it.

Question 1

Question Consider a communication system consisting of a binary block code with uniformly at random chosen, an error channel, and a minimum-distance decoder. Check the correct statement about the minimum distance decoder.

1. None of the others can be stated with certainty due to missing information
2. It minimizes the error probability if the channel is a binary symmetric channel with crossover (flip) probability smaller than $\frac{1}{2}$
3. It always minimizes the error probability
4. It minimizes the error probability if the channel is a binary symmetric channel.

Here we have two things that we need to know:
The minimum distance decoder is **maximum distance decoder** when the channel is memoryless and symmetric which means that it minimizes the probability of the decoding error. This means that the third option doesn't work. We have two options left (three). For the second option we have to go back to the exercices set of the week 10 exercice 3. We are looking for the

probability:
For choosing a k bits out of n, there are $\binom{n}{k}$ ways to do so, and each of them have a probability of $p^k(1-p)^{n-k}$ which leads to: $\binom{n}{k}p^k(1-p)^{n-k}$.
However if we know c we can get rid of the combinatorial computation and have:

$$P(yc) = p^k(1-p)^{n-k}$$

So now we are looking for the minimum when $p \leq \frac{1}{2}$, because of this we can see that the number of the right is only bigger than the number on the left:

$$\frac{p}{1-p} < 1$$

And Now if we want to make this the biggest the goal is making d the smallest and $n-d$ the biggest, A.K.A., d the smallest.
So, When $d \leq \frac{1}{2}$ all we said works find, the minimum distance decoding works and is optimal.
When $p = \frac{1}{2}$ all codewords are equally likely $\implies$ any decoding strategy is equally good, or bad.
And when $p > \frac{1}{2}$ we would be better using the "inverse" of the minimum distance decoder.
Therefore, the answer is the second.

Question 2

> *Question* How many $x \in \mathbb{Z}/23\mathbb{Z}$ satisfy the equation $0 = 1 - x + x^2 - x^3 + \cdots - x^{21} + x^{22} - x^{23}$, when all operations are with respect to the field $(\mathbb{Z}/23\mathbb{Z}, +, \cdot)$?

So here the question is, how may solution to the equation:

$$0 = \sum_{n=0}^{23}(-1)^n x^n$$

Which is a geometric series. (I didn't have that first but) you can replace the geometric series by its formula:

$$\sum_{n=0}^{23}(-1)^n x^n = \frac{(-x)^{24}-1}{x+1} = \frac{x^{24}-1}{-x-1}$$

So from here it becomes easier, we want to get rid of this 24th power which is painful. To be able to do so we know something, which is the little theorem of Fermat:

$$x^{p-1} = 1 \bmod p$$

therefore we have that $x^22 = 1 \implies x^{24} = 1 \cdot x^2$. Which give:

$$0 = \frac{x^2-1}{-x-1}$$

Where $-x-1 = x+1$:

$$\begin{aligned} 0 &= \frac{x^2-1}{x+1} \\ &= \frac{(x+1)(x-1)}{x+1} \\ &= x-1 \\ x &= 1 \end{aligned}$$

Which give us only one solution.

Question 3 For the last question,

> *Question* Let E be a subspace of $\mathbb{F}_7^4$ which consists of elements $\vec{x} = (x_1, x_2, x_3, x_4)$ satisfying:
>
> $$\begin{aligned} x_1 + 6x_2 + 3x_3 + 4x_4 &= 0 \\ 3x_1 + 6x_2 + x_3 + 3x_4 &= 0 \\ 5x_1 + 2x_2 + x_3 + 3x_4 &= 0 \end{aligned}$$
>
> What is the dimension of E?

Here how I see it is like asking the dimension of the kernel (noyau). We have a matrix and we ask what the dimension of the solution of the equation $A \cdot \vec{x} = \vec{0}$. In fact the subspace if the kernel of the application which has the matrix:

$$A = \begin{pmatrix} 1 & 6 & 3 & 4 \\ 3 & 6 & 1 & 3 \\ 5 & 2 & 1 & 3 \end{pmatrix}$$

There fore you can use Gauss's method and be careful because we are in $\mathbb{F}_7$. and the find the kernel like always. Here the first mistake is to reduce the system and then to give all the linearly independent vector as the dimensions. But this is the opposite, this would gives us the rank. which as you can see is not the subspace E.

$$E = \left\{ \vec{x} \in \mathbb{F}_7^4 : A \cdot \vec{x} = \vec{0} \right\}$$

Where:

$$A = \begin{pmatrix} 1 & 6 & 3 & 4 \\ 3 & 6 & 1 & 3 \\ 5 & 2 & 1 & 3 \end{pmatrix}$$

If you did it correctly, the answer is 1.