# AICC II

Arthur Herbette
Prof. Michael Gastpar

March 20, 2025

# Contents

# List of lectures

# 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 $\Omega$: set of all possiblie outcomes

**Definition 1** *Sample space* $\Omega$ *is the set of all possible outcomes*

**Definition 2** *Event* $E$: *a subset of* $\Omega$. *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 $\Omega$, arbitary $p(\omega)$**

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* $\Omega$: *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 $\Omega$ :

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

## 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*

> **Theoreme 1** *If* $\Omega$ *is the union of disjoint event* $F_1, F_2, \ldots, F_n$ *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)

$$p(E) = p((\underbrace{(E \cap F) \cup (E \cap F^c)}_{\text{union of disjoint sets}}))$$

$$= p(E \cap F) + p(E \cap F^c)$$

$$= \frac{p(E \cap F)}{p(F)}p(F) + \frac{p(E \cap F^c)}{p(F^c)}p(F^c)$$

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

**Bays' Rule**

**Theoreme 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 indeppendecy**

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**

> **Theoreme 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 $S_i = \{0, \ldots, 9\}$.
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**

> **Theoreme 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**

> **Theoreme 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. ($\approx 0.55$)

### 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 $\Gamma$ 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 **011**.

- 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: $ct$      $\mathcal{A}$
  $\Gamma_B$

| 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**

> **Theoreme 6** *If a D-ary code is uniquely decodable then its codeword length $i_1, \ldots, i_M$ 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**

> **Theoreme 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 $D^{L-l_i} + D^{L-l_2} + \cdots + D^{L-l_m} \leq D^L$

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

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

*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 :

> **Theoreme 8** *If the positive integer $I_1, \ldots, I_M$ 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**

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

**Part II**

> **Theoreme 10** *If the positive integer $l_1, \ldots, l_M$ 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 Code-word 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 code-word length: Lower Bound**

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

*Proof*          We want to prove that:

$$H(s) - \sum_s p(s)l(s)$$

$$= -\sum_s p(s) \log p(s) - \sum_s p(s)l(s)$$

$$= -\sum_s p(s) \log p(s) - \sum_s p(s) \log 2^{l(s)}$$

$$= -\sum_s p(s) \log(p(s) \cdot 2^{l(s)}) \leq \dots$$

Therefore:

$$= \sum_s p(s) \log\left(\frac{1}{p(s)} 2^{-l(s)}\right)$$

$$\leq \sum_s p(s)\left(\frac{1}{p(s)} 2^{-l(s)} - 1\right) \cdot C$$

$$= \left(\sum_s 2^{-l(s)} - \sum_s p(s)\right) \cdot C$$

$$\leq 0$$

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

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

**Theorem**          **Theoreme 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*"

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

**Theoreme 14** *The average codeword-length of a uniquely decodable code $\Gamma$ for S must satisfy:*

$$H_D(S_1, S_2, \ldots, S_n) \le L((S_1, S_2, \ldots, S_n), \Gamma)$$

*And there exists a uniquely decodable code $\Gamma_{SF}$ satisfying:*

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

**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 $S_1, S_2, \ldots, S_n$ 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* $(s_1, S_2, \ldots, S_n) \in \mathcal{A}^n$

**Not independent**

**Definition 13** *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*
*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**

> **Theoreme 15** *The conditional entropy of a discrete random variable $X \in \mathcal{X}$ conditioned on $Y = y$ satisfies:*
>
> $$0 \leq H_D(X \mid Y = y) \leq \log_D |\mathcal{X}|$$
>
> *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 $Y$**

> **Theoreme 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 $y$.

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

**Conditioning Reduces Entropy**

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

**Theoreme 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*

$$H(X \mid Y) - H(X) =$$

$$= \sum_y p(y) \left( - \sum_x p(x \mid y) \log p(x \mid y) \right) + \sum_x p(x) \log p(x)$$

$$= \sum_{x,y} p(y) p(x \mid y) \log \frac{1}{p(x \mid y)} + \sum_{x,y} p(y \mid x) p(x) \log p(x)$$

$$= \sum_{x,y} p(x, y) \log \frac{p(x)}{p(x \mid y)}$$

$$\leq \sum_{x,y} p(x, y) \left( \frac{p(x)}{p(x \mid y} - 1 \right) \cdot \log e$$

$$= \sum_{x,y} (p(x) p(y) - p(x, y)) \log(e)$$

$$= \left( \left( \sum_y p(x) p(y) \right) - \left( \sum_x p(x) p(y) \right) \right)$$

**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:

> **Theoreme 18** *For any three discrete random variables $X, Y$ and $Z$,*
>
> $$H_D(X \mid Y, Z) \le H_D(X \mid Z)$$
>
> *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:

$$H_D(X \mid Y, Z) - H_D(X \mid Z) = \mathbb{E}\left[\log_D \frac{1}{p_{X|Y,Z}(X \mid Y, Z)}\right] + \mathbb{E}[\log_D p_{X|Z}(X \mid Z)]$$

$$= \mathbb{E}\left[\log_D \frac{p_{X|Z}(X \mid Z)}{p_{X|Y}(X \mid Y, Z)}\right]$$

$$= \mathbb{E}\left[\log_D \frac{p_{X|Z}(X \mid Z)p_{Y|Z}(Y \mid Z)p_Z(Z)}{niquesamere}\right.$$

───────── 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:

$$H(X \mid Y) = \sum_y p(y)H(X \mid Y = y)$$

$$= -\sum_y \sum_x p(x, y) \log p(x \mid y)$$

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:

$$0 \le H(X \mid Y = y) \le \log |\mathcal{X}|$$
$$0 \le H(X \mid Y) \le \log |\mathcal{X}|$$

And the inequality:

> $$H(X \mid Y) \le H(X)$$

*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 $L_1 = 1$:

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

What we are doing here is the joint distribution:

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

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:

$$H_D(X, Y) = -\sum_x p_X(x) \left( \sum_Y p_{Y|X}(y \mid x) \log_D (p_X(x) p_{Y|X}(y \mid x)) \right)$$

$$= -\sum_x p_X(x) \left( \sum_y p_{Y|X}(y \mid x) (\log_D p_X(x) + \log_D p_{Y|X}(y \mid x)) \right)$$

$$= -\sum_x p_X(x) \left[ \left( \sum_y p_{Y|X}(y \mid x) \log_D p_X(x) \right) + \left( \sum_y p_{Y|X}(y \mid x) \log_D p_{Y|X}(y \mid x) \right) \right]$$

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

---

**Theoreme 19**

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

---

*Professor remark*

Firstly:

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

Which is either proved by:

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

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**

**Theoreme 20** *Let $S_1, \ldots, S_n$ be discrete random variables. Then:*

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

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:

$$H(\underbrace{S_1, S_2, \ldots, S_{n-1}}_{=Z}, S_n)$$
$$= H(Z) + H(S_n \mid Z)$$
$$= H(\underbrace{S_1, S_2, \ldots, S_{n-2}}_{=Z'}, S_{n-1}) + H(S_n \mid S_1 \ldots, S_{n-1})$$
$$= H(Z') + H(S_{n-1} \mid Z') + H(S_n \mid S_{1,n-1})$$

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

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

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

**Theoreme 21** *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 iff, $S_1, \ldots, S_n$ are independent*

*Proof*
$$H(S_1, S_2, S_3) = H(S_1) + H(S_2 \mid S_1) + H(S_3 \mid S_1, S_2)$$
$$\leq H(S_1) + H(S_2) + H(S_3)$$

**Another way around**

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

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

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:

$$H(L_1, L_2) = 3.2744$$
$$H(L_1) = 0.6500$$
$$H(L_2) = 3.2188$$

We compute:

$$H(L_2 \mid L_1) = H(L_1, L_2) - H(L_1) = 3.2744 - 0.6500 = 2.6254$$
$$H(L_1 \mid L_2) = H(L_1, L_2)_H(L_2) = 3.2744 - 3.2188 = 0.056$$

And verify that indeed:

$$H(L_1 \mid L_2) \leq H(L-1) \leq H(L_1, L_2)$$
$$H(L_2 \mid L_1) \leq H(L_2) \leq H(L_1, L_2)$$

## 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:

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

However:

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

More generally:

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

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:

$$H(X \mid Y = y) = -\sum_x p(x \mid y) \log_D p(x \mid y)$$
$$H(X \mid Y = y) = -\sum_y \ldots$$

And we also saw that on average:

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

We also saw the chain rule:

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

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

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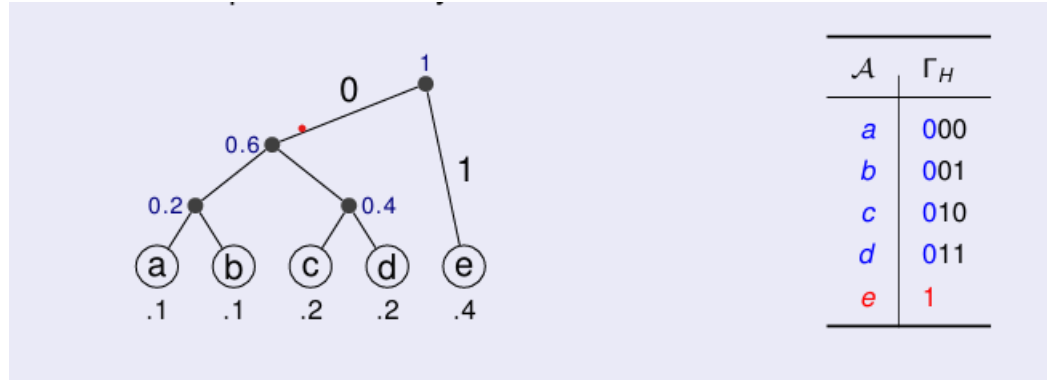>
> $$p(cat) = \frac{1}{2}$$
> $$p(dog) = \frac{1}{4}$$
> $$p(pony) = \frac{1}{4}$$
>
> We want to make it the best way, the question we should ask is:
>
> - is the animal a cat?
>
> $$X \quad a \quad b \quad c \quad d \quad e \text{ height}$$

We then do a Huffman tree:



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.
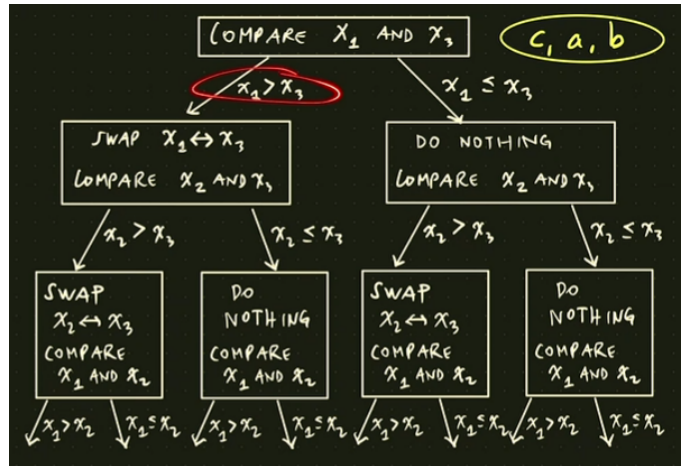
**Sorting via pairwise comparisons**

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 $p(x)$.

$$E \geq max_{p(x)} H_2(X) = \log_2 | \mathcal{X} |$$
$$= \log_2 n!$$

We already know a bounds on factorial:

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

Therefore:

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

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:

**Theoreme 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:

**Theoreme 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*

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

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

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
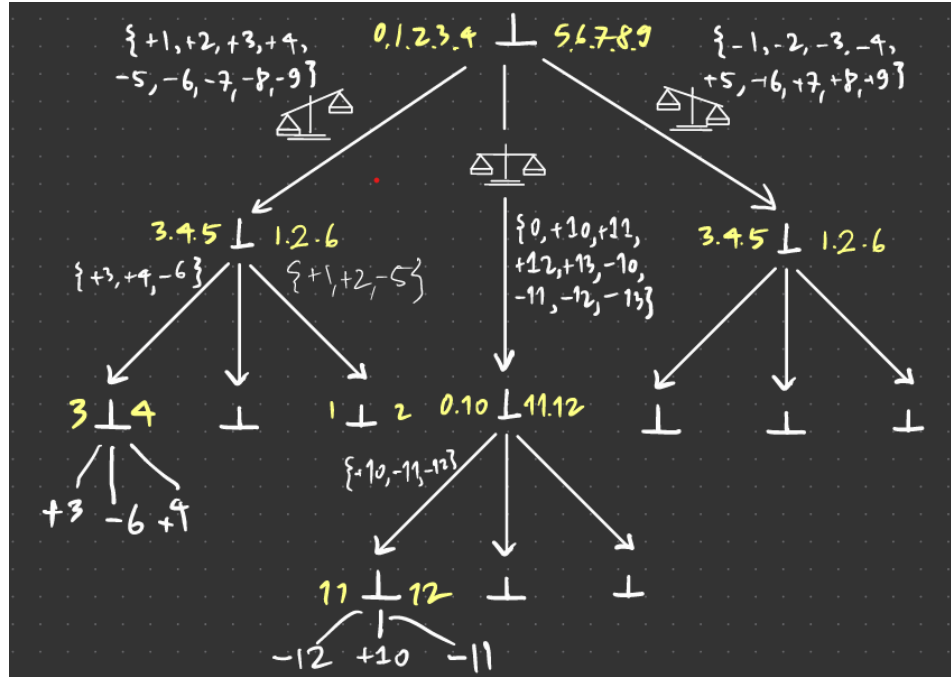
*Proof*

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

This is a *must*.

But also:

$$H_3(S_1) + H(S_2 \mid S_1) + H(S_3 \mid S_1, S_2) \leq H_3(S_1) + H(S_2) + H(S_3)$$
$$\leq \log_3 3 + \log_3 3 + \log_3 3$$

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?
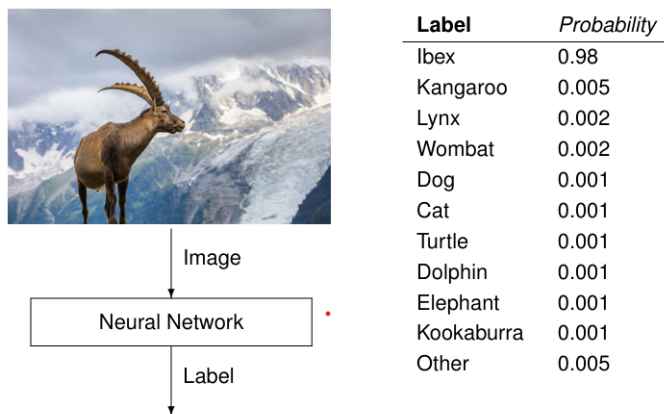From source compression, we can establish the following facts?

- 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*

$$L\{Q(x) \neq \text{Label}(x)\}$$

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

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**

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

- Given training data (image, label), for $i = 1, 2, \ldots, n$ we select $Q(\text{ label} \mid \text{image})$ 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** Takes a screen of the blackboard
**neural network**
- 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**     because why is juste binary we use:
**image $\mathcal{X}$**

**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**     Cross entropy loss:
**loss**

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

**Theoreme 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 $P = Q$.

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

*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:

- $0 \leq H_D(X) \leq \log_D |\mathcal{X}|$
- $H_D(X \mid Y) \leq H_D(X)$ 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?
>
> $$p(S_1, S_3) = \sum_{S_2} p(S_1, S_2, S_3)$$
> $$= \sum_{S_2} p(S_1)p(S_2 \mid S_1)p(S_3 \mid S_2)$$

**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:

$$L(P, Q) = -\sum_y P(y) \log_D Q(y)$$

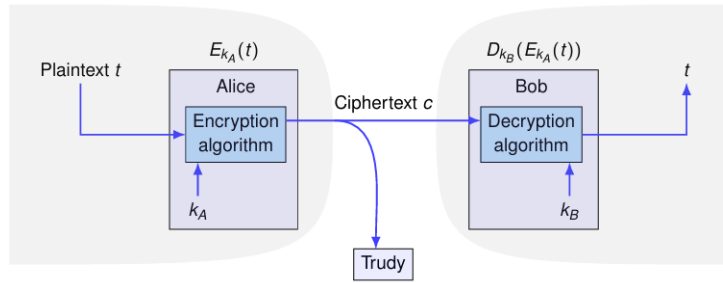———————————— March 18, 2025 — **Lecture 9 : Introduction to cryptography**

# 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 ci-*
*pher*

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 at-*
*tacks 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 (Vi-*
*genère's ci-*
*pher 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:
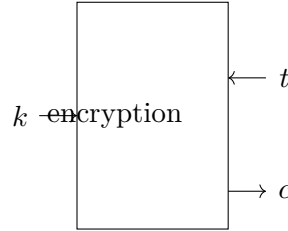


**Theoreme 25** *Perfect secrecy implies:*

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

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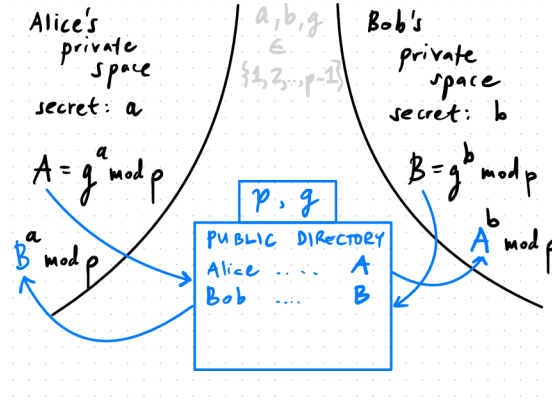
In 1976, Diffie and Hellman came up with a solution.

*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*

**Theoreme 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

**Asymmetric 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**
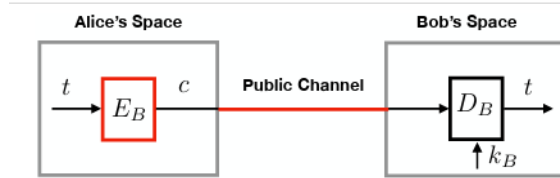
*Setup*

- 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** $x$.
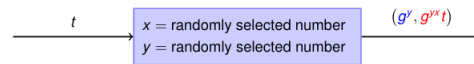


Given the trapdoor information $x$, we can invert the function as follows: Compute the inverse of $g(^{y})^x = g^{xy}$, multiply the result with $g^{xy}t$. The result is $t$.
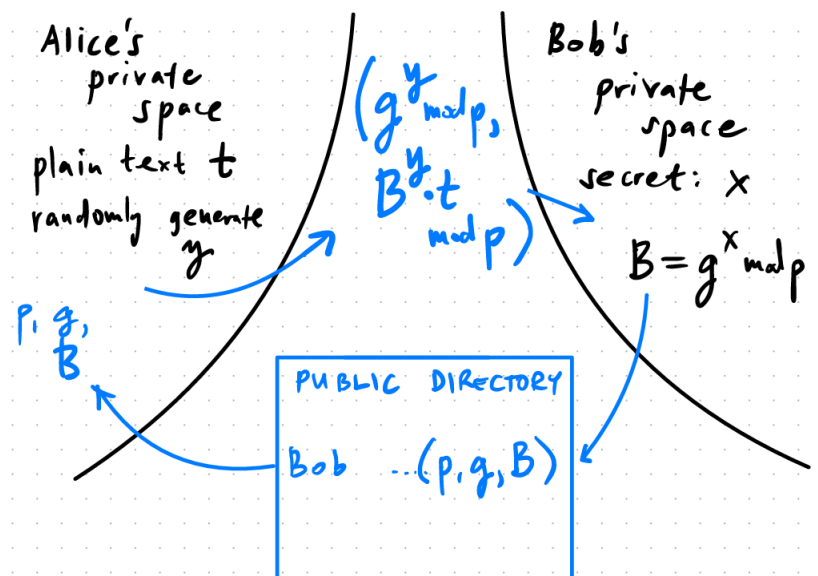
*More concretly*

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