LECTURE 2
SOURCE CODING THEOREM AND FIRST EFFICIENT COMPRESSION ALGORITHMS

Information Theory

Thomas Debris-Alazard

Inria, École Polytechnique

THE OBJECTIVE OF THE DAY

How to compress data?

- Ultimate data compression (source coding theorem)
- An algorithmic way to reach it (Huffman encoding)

How many bits are needed to describe the outcome of an experiment?

Compressing data from a source into L bits and recover the data reliably:

the source is at most L bits per symbol $\left(\textit{in average}\right)$

1. Optimal compression when small errors are allowed
   - Showing the possibility of non-trivial compression
   - First intuition/rigorous definition about the **A**symptotic **E**quipartition **P**rinciple (AEP)

     $\longrightarrow$ AEP pursued in Lecture 3
2. A first efficient algorithm to reach optimal compression: Huffman encoding

   $\longrightarrow$ It appears almost everywhere (`gzip`, `pkzip`, `winzip`, `bzip2`, `jpeg`, `png`, `mp3`)

SOURCE CODING THEOREM

How to compress outputs of $X : \Omega \to \mathcal{X}$?

$\longrightarrow$ Write elements of $\mathcal{X}$ with bit-strings! This will require strings of length

Raw bit content:

The raw bit content of $X : \Omega \to \mathcal{X}$ is defined as,

$$H_0(X) \stackrel{\text{def}}{=} \log_2 \sharp\mathcal{X}$$

But is it a good idea?

TO COMPRESS, WHY NOT?

How to compress outputs of $X : \Omega \to \mathcal{X}$?

$\longrightarrow$ Write elements of $\mathcal{X}$ with bit-strings! This will require strings of length

Raw bit content:

The raw bit content of $X : \Omega \to \mathcal{X}$ is defined as,

$$H_0(X) \stackrel{\text{def}}{=} \log_2 \sharp\mathcal{X}$$

But is it a good idea?

$\longrightarrow$ No! But it is non-trivial to overcome this. . .

Could we compress $\mathsf{X} = x \in \mathcal{X}$ to $c(x)$, and decompress $c(x)$ to x s.t $c(x)$ has less than $H_0(\mathsf{X})$ bits?

No! There are $2^{H_0(\mathsf{X})}$ possible outcomes...

Is it a dead-end? Remember your motto in this course (typical sequences)!

Could we compress $\mathsf{X} = x \in \mathcal{X}$ to $c(x)$, and decompress $c(x)$ to x s.t $c(x)$ has less than $H_0(\mathsf{X})$ bits?

No! There are $2^{H_0(\mathsf{X})}$ possible outcomes...

Is it a dead-end? Remember your motto in this course (typical sequences)!

- **Lossy Compressor**: compress by mapping some files to the same bit-string

$\longrightarrow$ It is ambiguous!

Lossy compressor efficiency:

Probability to map two different files to the same has to be small!

- **Lossless Compressor**: all mapping are different but

$\longrightarrow$ It imposes to compress sometimes with a larger number of bits

Lossless compressor efficiency:

Probability to be lengthened has to be small, and to be shortened has to be large!

Focus on lossy compressor!

Ambiguity: we don't care if the probability to happen is small!

Taking some risk?

Let $\mathcal{X} = \{a, b, c, d, e, f, g, h\}$ with associated distribution $\left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{3}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}\right\}$

1. Show that the raw bit content is 3 bits
2. Show that $\mathbb{P}(X \in \{a, b, c, d\}) = \frac{15}{16}$
3. Show that we can compress the source to 2 bits with risk $\frac{1}{16}$. What do you conclude?
4. Suppose that we accept a risk of $\frac{1}{2}$, how many bits are required to compress the source?

Ambiguity: risk δ to compress two data into the same bit-string

δ-sufficient subset $\mathcal{S}_\delta$ for compression:

$$\mathbb{P}(X \notin \mathcal{S}_\delta) \leq \delta$$

- Compression: define a one-to-one mapping[1] $S_\delta \mapsto \{0,1\}^{\log_2 \sharp S_\delta}$. Then, if $X = x \in S_\delta$ write $c(x)$ with $\log_2 \sharp\mathcal{S}_\delta$ bits, otherwise do $\perp$
- Decompression: inverse the one-to-one mapping

The decompression works with probability $\geq 1 - \delta$ and **it uses $\log_2 \sharp S_\delta$ bits**

(1) If $\sharp S_\delta$ not a power of two, use $\lceil \log_2 \sharp S_\delta \rceil$ bits.

Ambiguity: risk δ to compress two data into the same bit-string

δ-sufficient subset $\mathcal{S}_\delta$ for compression:

$$\mathbb{P}(X \notin \mathcal{S}_\delta) \leq \delta$$

- Compression: define a one-to-one mapping(1) $S_\delta \mapsto \{0,1\}^{\log_2 \sharp S_\delta}$. Then, if $X = x \in S_\delta$ write $c(x)$ with $\log_2 \sharp\mathcal{S}_\delta$ bits, otherwise do $\perp$
- Decompression: inverse the one-to-one mapping

The decompression works with probability $\geq 1-\delta$ and **it uses $\log_2 \sharp S_\delta$ bits**

For the best non-ambiguity: it motivates to consider

Smallest δ-sufficient subset $\mathcal{S}_\delta$ for compression:

$$\mathbb{P}(X \notin \mathcal{S}_\delta) \leq \delta \iff \mathbb{P}(X \in S_\delta) \geq 1-\delta$$

(1) If $\sharp S_\delta$ not a power of two, use $\lceil \log_2 \sharp S_\delta \rceil$ bits.

δ-sufficient subset $\mathcal{S}_\delta$ for compression:

$$\mathbb{P}(X \notin \mathcal{S}_\delta) \leq \delta \iff \mathbb{P}\left(X \in S_\delta\right) \geq 1 - \delta$$

- Compression: define a one-to-one mapping[2] $S_\delta \mapsto \{0, 1\}^{\log_2 \sharp S_\delta}$. Then, if $\mathbf{X} = x \in S_\delta$ write $c(x)$ with $\log_2 \sharp\mathcal{S}_\delta$ bits, otherwise do $\perp$

$\longrightarrow$ It compresses outcomes of $\mathbf{X}$ with $\log_2 \sharp S_\delta$ bits!

For the best non-ambiguity: it motivates to consider

Smallest δ-sufficient subset $\mathcal{S}_\delta$ for compression:

$$\mathbb{P}(X \notin \mathcal{S}_\delta) \leq \delta \iff \mathbb{P}\left(X \in S_\delta\right) \geq 1 - \delta$$

(2) If $\sharp S_\delta$ not a power of two, use $\lceil \log_2 \sharp S_\delta \rceil$ bits.

The essential bit content:

Given $\mathsf{X} : \Omega \to \mathcal{X}$ and $S_\delta \subseteq \mathcal{X}$ be the smallest δ-sufficient subset for X,

$$H_\delta\,(\mathsf{X}) \stackrel{\text{def}}{=} \log_2 \sharp S_\delta$$

Be careful: don't confuse H_δ and H_0 with the entropy H

Exercise:

Show that $H_\delta(\mathsf{X})$ is equal to $H_0(\mathsf{X})$ $\big($the raw bit content defined in Slide 5$\big)$ when $\delta = 0$

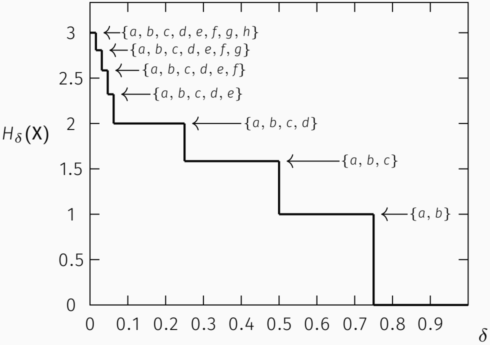
Optimal lossy compression:

The optimal lossy compression size, *i.e.*, the number of bits, with error δ is $H_\delta(\mathsf{X}) = \log_2 \sharp S_\delta$

A natural question: is $H_\delta(\mathsf{X})$ "strongly" function of δ?

Let $\mathcal{X} = \{a, b, c, d, e, f, g, h\}$ with associated distribution $\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{3}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}\}$

- $H_0(\mathsf{X}) = 3$
- $H_{1/16}(\mathsf{X}) = 2$
- $H_{3/4}(\mathsf{X}) = 0$



Consider the source: n independent flips $\mathbf{x} = (x_1, \dots, x_n) \in \{0,1\}^n$ where $\mathbb{P}(x_i = 1) = p$

$$\mathbb{P}(\mathbf{x}) = p^{|\mathbf{x}|}(1-p)^{n-|\mathbf{x}|} \quad \text{where } |\mathbf{x}| \stackrel{\text{def}}{=} \sharp\Big\{i \in \{1, \dots, n\} : \ x_i \neq 0\Big\}$$

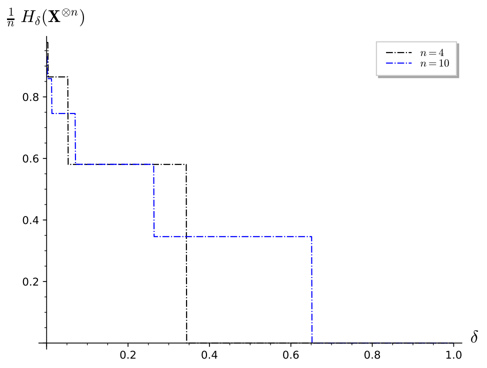
Let $\mathbf{X}^{\otimes n}$ denote this source $\Big($$n$ independent and identically distributed$\Big)$

What is the optimal compression size if we allow a probability of error δ?

Optimal lossy compression:

What is the size of $\log_2 \sharp S_\delta$ as function of δ?

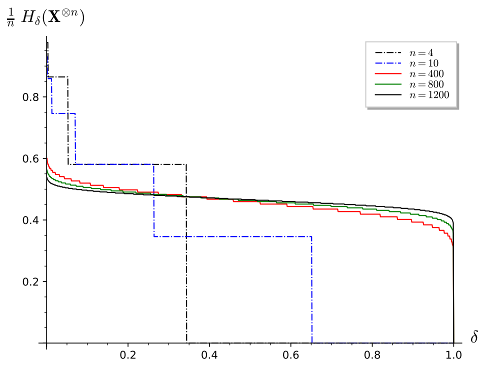
The behaviour depends “strongly‘” on δ **but also on n**



What does happen if n grows?

ANOTHER EXAMPLE (II)

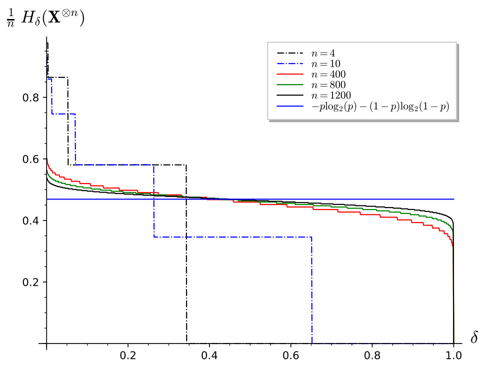
The behaviour depends “strongly‘” on δ **but also on n**



The curve becomes essentially flat! What do you conjecture?

ANOTHER EXAMPLE (II)

The behaviour depends “strongly‘” on δ **but also on n**



For all $\delta \in (0, 1)$, it tends toward the binary entropy $h(p) = -p \log_2 p - (1-p) \log_2(1-p)$

In particular: **it does not (extremely surprisingly!) depend on δ**

The probability that $\mathbf{x}$ has r 1's and $(n-r)$'s 0, *i.e.*, $|\mathbf{x}| = r$, is

$$\mathbb{P}(|\mathbf{x}| = r) = \binom{n}{r} p^r (1-p)^{n-r}$$

$\longrightarrow$ It is a Binomial distribution

The mean of $|\mathbf{x}|$ is np and its standard deviation is $\sqrt{np(1-p)}$,

- If $n = 100$ and $p = \frac{1}{10}$,

$$|\mathbf{x}| \sim 10 \pm 3 \quad \left(3/10 = 0.3\right)$$

- If $n = 1000$ and $p = \frac{1}{10}$,

$$|\mathbf{x}| \sim 100 \pm 10 \quad \left(10/100 = 0.1 < 0.3\right)$$

As n increases: distribution of $|\mathbf{x}|$ becomes more concentrated: possible values of $|\mathbf{x}|$ grows as n, the standard deviation of r only grows as $\sqrt{n}$

$\longrightarrow$ $|\mathbf{x}|$ is most likely to fall in a small range of values

But where does the distribution concentrate?

$\longrightarrow np$ 1's and $(n - np)$ 0's:

$$\log_2 \mathbb{P}(\mathsf{x})_{\text{typ}} = \log_2 p^{np}(1-p)^{n-np} = -nh(p) \quad \text{where } h(\cdot) \text{ binary entropy}$$

A remark:

The binary entropy $h(p)$ is the entropy of the Bernoulli distribution with parameter p

$\Big($recall that according to Lecture 1: $H(\mathsf{X}^{\otimes n}) = nH(\mathsf{X})\Big)$

It motivates to introduce (your new best friend!) the typical set:

$$T_{n\varepsilon} \overset{\text{def}}{=} \left\{ \mathsf{x} \in \mathcal{X}^n : \left| \frac{1}{n} \log_2 \frac{1}{\mathbb{P}\left(\mathsf{X}^{\otimes n} = \mathsf{x}\right)} - H(\mathsf{X}) \right| < \varepsilon \right\}$$
$$= \left\{ \mathsf{x} \in \mathcal{X}^n : 2^{-n(H(\mathsf{X})+\varepsilon)} < \mathbb{P}\left(\mathsf{X}^{\otimes n} = \mathsf{x}\right) < 2^{-n(H(\mathsf{X})-\varepsilon)} \right\}$$

Asymptotic Equipartition Property (AEP): "all" the distribution falls in $T_{n\varepsilon}$ where each events

happen with "equiprobable" probability

OUR HYPOTHESIS

We will consider in this lecture only i.i.d. sources with n drawing: $\mathsf{X}^{\otimes n}$

$$\mathbb{P}\left(\mathsf{X}^{\otimes n} = (x_1, \dots, x_n)\right) = \mathbb{P}(\mathsf{X} = x_1) \cdots \mathbb{P}(\mathsf{X} = x_n)$$

Be patient: a more general context in Lecture 3

Shannon source coding theorem:

Let $X : \Omega \to \mathcal{X}$, then for all $\delta \in (0,1)$,

$$\frac{1}{n} H_\delta \left(X^{\otimes n}\right) \xrightarrow[n \to +\infty]{} H(X)$$

Conclusion: the optimal lossy compression size for any error rate > 0 is the entropy of the source $H(X)$ **if we consider block of outputs with size large enough**

PROOF (I)

Start: use your best friend, the Asymptotic Equipartition Property $\left(\text{AEP}\right)$

Fundamental idea: to determine the typical set, apply the weak law of large number to the r.v.

Y whose outputs are $\frac{1}{n}\log_2\frac{1}{\mathbb{P}(\mathbf{x})}$ where $\mathbf{x}$ is drawn according to $\mathbf{X}^{\otimes n}$

$$\frac{1}{n}\log_2\frac{1}{\mathbb{P}(\mathbf{x})} \overset{\text{(indep)}}{=} \frac{1}{n}\sum_{\mathbf{x}\in\mathcal{X}^n}\log_2\frac{1}{\mathbb{P}(x_i)} \quad \text{where the } x_i\text{'s are i.d.d according to } \mathbf{X}$$

$\longrightarrow \mathrm{Y} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{Y}_i$ where the Y_i's are i.i.d. $\log_2\frac{1}{\mathbb{P}(x_i)}$ with x_i picked according to $\mathbf{X}$

Expectation and variance:

$$\mathbb{E}(\mathrm{Y}_i) = \sum_{x\in\mathcal{X}}\log_2\left(\frac{1}{\mathbb{P}(x)}\right)\mathbb{P}(x) = H(\mathbf{X}) \quad \text{and} \quad \sigma^2 \overset{\text{def}}{=} \mathbf{Var}(\mathrm{Y}_i)$$

By the weak law of large number:

$$\mathbb{P}\left(|\mathrm{Y}-H(\mathbf{X})|<\varepsilon\right) = \mathbb{P}_{\mathbf{x}}\left(\mathbf{x}\in\left\{\mathbf{y}:\ \left|\frac{1}{n}\log_2\frac{1}{\mathbb{P}(\mathbf{X}^{\otimes n}=\mathbf{y})} - H(\mathbf{X})\right| < \varepsilon\right\}\right) \geq 1-\frac{\sigma^2}{n\varepsilon^2}$$

where $\mathbf{x}$ is drawn according to $\mathbf{X}^{\otimes n}$

$$\mathbb{P}_{\mathsf{x}}\left(\mathsf{x} \in \left\{\mathsf{y} : \left|\frac{1}{n}\log_2 \frac{1}{\mathbb{P}(\mathsf{X}^{\otimes n} = \mathsf{y})} - H(\mathsf{X})\right| < \varepsilon\right\}\right) \geq 1 - \frac{\sigma^2}{n\varepsilon}$$

Typical set:

$$T_{n\varepsilon} \overset{\text{def}}{=} \left\{\mathsf{y} \in \mathcal{X}^n : \quad \left|\frac{1}{n}\log_2 \frac{1}{\mathbb{P}\left(\mathsf{X}^{\otimes n} = \mathsf{y}\right)} - H(\mathsf{X})\right| < \varepsilon\right\}$$

$$\longrightarrow \mathbb{P}_{\mathsf{x}}\left(\mathsf{x} \in T_{n\varepsilon}\right) \geq 1 - \frac{\sigma^2}{n\varepsilon^2}$$

Conclusion:

If n large enough, i.e., $\frac{\sigma^2}{n\varepsilon^2} \leq \delta$, then,

$$\sharp T_{n\varepsilon} \geq \sharp S_\delta = 2^{H_\delta\left(\mathsf{X}^{\otimes n}\right)}$$

as S_δ is the smallest subset such that $\mathbb{P}(\mathsf{x} \in S_\delta) \geq 1 - \delta$.

$\longrightarrow$ We will use $T_{n\varepsilon}$ to provide an upper-bound on $H_\delta\left(\mathsf{X}^{\otimes n}\right)$

PROOF (III): $\frac{1}{n}\log_2 H_\delta(X^{\otimes n}) \leq H(X) + \varepsilon$

$$T_{n\varepsilon} = \left\{ y \in \mathcal{X}^n : \quad \left| \frac{1}{n} \log_2 \frac{1}{\mathbb{P}\left(X^{\otimes n} = y\right)} - H(X) \right| < \varepsilon \right\}$$

$$\begin{aligned} 1 &= \sum_{x \in \mathcal{X}^n} \mathbb{P}(X^{\otimes n} = x) \\ &\geq \sum_{x \in T_{n\varepsilon}} \mathbb{P}(X^{\otimes n} = x) \\ &\overset{\text{(by def)}}{\geq} \sum_{x \in T_{n\varepsilon}} 2^{-n(H(X)+\varepsilon)} \\ &= \sharp T_{n\varepsilon}\, 2^{-n(H(X)+\varepsilon)} \end{aligned}$$

Conclusion:

$$\sharp T_{n\varepsilon} \leq 2^{n(H(X)+\varepsilon)}$$

If n large enough, i.e., $\frac{\sigma^2}{n\varepsilon^2} \leq \delta$, then,

$$\sharp S_\delta = 2^{H_\delta\left(X^{\otimes n}\right)} \leq \sharp T_{n\varepsilon} \leq 2^{n(H(X)+\varepsilon)}$$

PROOF (IV): $\frac{1}{n}\log_2 H_\delta(X^{\otimes n}) \geq H(X) - \varepsilon$

Proof by contradiction: suppose there exists an infinity of n's,

$$H_\delta\left(X^{\otimes n}\right) \leq n \cdot (H(X) - \varepsilon)$$

$$\longrightarrow \text{It exists } S \text{ such that: } \begin{cases} \sharp S \leq 2^{n(H(X)-\varepsilon)} \\ \mathbb{P}(x \in S) \geq 1 - \delta \end{cases}$$

$$\mathbb{P}(x \in S) = \mathbb{P}\left(x \in S \cap T_{n\varepsilon/2}\right) + \mathbb{P}\left(x \in S \cap \overline{T_{n\varepsilon/2}}\right) \leq \mathbb{P}\left(x \in S \cap T_{n\varepsilon/2}\right) + \mathbb{P}\left(x \notin T_{n\varepsilon/2}\right)$$

- Upper-bound on first term,

$$\begin{aligned} \mathbb{P}\left(x \in S \cap T_{n\varepsilon/2}\right) &= \sum_{x \in S \cap T_{n\varepsilon/2}} \mathbb{P}\left(X^{\otimes n} = x\right) \\ &\overset{\text{(by def)}}{\leq} \sum_{x \in S \cap T_{n\varepsilon/2}} 2^{-n(H(X)-\varepsilon/2)} \\ &\leq 2^{n(H(X)-\varepsilon)}\, 2^{-n(H(X)-\varepsilon/2)} = 2^{-n\varepsilon/2} \quad \left(\text{as } \sharp S \leq 2^{n(H(X)-\varepsilon)}\right) \end{aligned}$$

- Upper-bound on the second term,

$$\mathbb{P}\left(x \in T_{n\varepsilon/2}\right) \geq 1 - \frac{4\sigma^2}{n\varepsilon^2} \Longrightarrow \mathbb{P}\left(x \notin T_{n\varepsilon/2}\right) \leq \frac{4\sigma^2}{n\varepsilon^2}$$

PROOF (V): $\frac{1}{n} \log_2 \mathrm{H}_\delta(\mathrm{X}^{\otimes n}) \geq \mathrm{H}(\mathrm{X}) - \varepsilon$

Proof by contradiction: suppose there exists an infinity of n's,

$$H_\delta\left(\mathrm{X}^{\otimes n}\right) \leq n \cdot (H(\mathrm{X}) - \varepsilon)$$

Conclusion:

$$\longrightarrow \text{It exists } S \text{ such that } 1 - \delta \leq \mathbb{P}(\mathbf{x} \in S) \leq 2^{-n\varepsilon/2} + \frac{4\sigma^2}{n\varepsilon^2}$$

Contradiction: for n large enough $0 < 1 - \delta < 1 - \delta$, contradiction. . .

Conclusion:

For all $\varepsilon > 0$, it exists n_0 such that for all $n \geq n_0$,

$$n \cdot (H(\mathsf{X}) - \varepsilon) \leq H_\delta(\mathsf{X}^{\otimes n}) \leq n \cdot (H(\mathsf{X}) + \varepsilon)$$

$\longrightarrow$ We have proved Shannon's source coding theorem!

- $\frac{1}{n}H_\delta(X^{\otimes n}) < H(X) + \varepsilon$: even if the probability of error δ is extremely small

$$\frac{1}{n}H_\delta(X^{\otimes n}) \leq H(X) \quad \text{not} \quad \frac{1}{n}H_\delta(X^{\otimes n}) \approx 1$$

- $\frac{1}{n}H_\delta(X^{\otimes n}) > H(X) - \varepsilon$: even if we tolerate a lot of errors, *i.e.*, $\delta \approx 1$

$$\frac{1}{n}H_\delta(X^{\otimes n}) \geq H(X) \quad \text{not} \quad \frac{1}{n}H_\delta(X^{\otimes n}) \approx 0$$

Regardless of our allowance for error δ, the number of bits per symbol needed to specify X is $H(X)$

no more and no less

- We used that $X^{\otimes n}$ is i.i.d. X only to prove with the weak law of large number

$$\mathbb{P}\left(X^{\otimes n} \in T_{n\varepsilon}\right) \xrightarrow[n\to+\infty]{} 1$$

More generally: any sequence verifying this property can be compressed with $nH(X)$ bits

(see Lecture 3)

But is it the end of the story for compression?

No! The proof is non-constructive, how do we compress? If we use S_δ, do we know its description?

S_δ has exponential size $\approx 2^{nH(\mathbf{X})}$: deciding if $\mathbf{x}$ is in or not is potentially hard

Furthermore, we need n $\left(\text{number of outputs by the source}\right)$ to be large to reach the optimality!

$\longrightarrow$ In what follows: an efficient compression algorithm reaching the theoretical limits

even for small n

COMPRESSION WITH SYMBOL CODES

We defined a lossy compression using **fixed length block codes**: as n grows, we can **encode** n i.i.d. sources $(x_1, \ldots, x_n)$ into a block of $n \cdot (H(\mathsf{X}) + \varepsilon)$ bits with vanishing probability of error

$\longrightarrow$ We verified the **possibility** of compression, but the block coding defined did not give a practical algorithm :(

Now:

We study practical data compression algorithms but with **variable-length** compression for small block sizes and that are **not lossy**

Key idea:

Shorter compression to the more probable outcomes and longer compression to the less probable

- Implications if a compression is lossless? Some compressions are shortened other have to be lengthened but **by how much**? *Kraft inequality*...
- Making compression practical? The fastest compression and decompression algorithms
- Optimal compression? Is the best achievable compression while being efficient is the entropy?

Source coding theorem with symbol codes (informal):

Given a source X, we can efficiently compute a variable-length compression for which decompression is efficient and whose compression average length belongs to $[H(\mathsf{X}), H(\mathsf{X}) + 1)$

$\longrightarrow$ The compression we will exhibit is known as **Huffman encoding**!

$\mathcal{X}^+$ denotes the set of all strings of finite length composed of elements from $\mathcal{X}$

Symbol codes and codewords:

Mapping φ from $\mathcal{X}$ to $\{0,1\}^+$. Given $x \in \mathcal{X}$, then $\varphi(x)$ is called a codeword and $\ell(x)$ will denote its length

The extended code φ^+ is the mapping from $\mathcal{X}^+$ to $\{0,1\}^+$ obtained by concatenation, without punctuation, of the corresponding codewords:

$$\varphi^+(x_1, \ldots, x_L) \stackrel{\text{def}}{=} \varphi(x_1) \cdots \varphi(x_L)$$

Symbol codes are **variable-length**!

An example:

x_i	$\varphi(x_i)$
a	0
b	10
c	110
d	111

Key idea: shorter compression to the more probable outcomes and longer compression to the less probable

$\longrightarrow$ It is the case for the Morse coding! For instance `E` requires only one symbols, `Z` four

A	.—	N	—.
B	— . . .	O	— — —
C	—. — .	P	. — —.
D	—..	Q	. — —.—
E	.	R	. — .
F	.. — .	S	. . .
G	— — .	T	—
H	. . . .	U	..—
I	..	V	. . . —
J	. — ——	W	. — —
K	—.—	X	—..—
L	. — ..	Y	—. — —
M	——	Z	— — ..

Issue: ambiguity

It is impossible to distinguish `BAM` and `NIJ` $\left(- \, . \, . \, . \, . \, - \, - \, -\right)$. In practice: a void between transmission $\big($adapted for "human" conversations$\big)$

$\longrightarrow$ Morse is ternary coding: need to add a separator symbol $\{\cdot, -, \perp_{\text{sep}}\}$

Ambiguous codes imply loss of information. . .

Non-ambiguous code:

A symbol code φ is said to be non-ambiguous if under the extended code φ^+, no two distinct strings have the same encoding, *i.e.*,

$$\forall x, y \in \mathcal{X}^+, \; x \neq y \implies \varphi^+(x) \neq \varphi^+(y)$$

Non-ambiguous code: necessary for lossless compression

Non-ambiguous codes are uniquely decodable code:

For any non-ambiguous code,

$$\varphi^+(x) \longmapsto x$$

is a well-defined mapping which is called decoding

Prefix codes: no codeword is the beginning of another codeword

Prefix codes:

A symbol code is called a prefix code if no codeword is a prefix of any other codeword, *i.e.*,

$$\forall (x, y) \in \mathcal{X} : \quad \left(\exists \mathbf{t} \in \{0,1\}^+ \text{ s.t. } \varphi(x)\mathbf{t} = \varphi(y)\right) \Rightarrow \left(\varphi(x) = \varphi(y)\right)$$

Proposition:

A prefix code is uniquely and efficiently decodeable

Proof:

Decoding procedure: given $\varphi(x_1) \cdots \varphi(x_L)$, looking from left to right until identifying the first codeword $\varphi(x_1)$ and etc $\Big($it does not require a special marker between words as with Morse coding$\Big)$

$\mathcal{X}$	$\varphi_0(x_i)$	$\varphi_1(x_i)$	$\varphi_2(x_i)$	$\varphi_3(x_i)$	$\varphi_4(x_i)$
x_1	0	0	1	0	0
x_2	11	010	10	10	01
x_3	11	01	100	110	011
x_4	10	10	1000	111	111

- φ_0: ambiguous
- φ_1: ambiguous; not unique decoding: 010 is x_2 or x_1x_4 or x_3x_1?
- φ_2: unique decoding but not prefix code
- φ_3: prefix code
- φ_4: not prefix code but it can be uniquely decoded $\left(\text{why?}\right)$

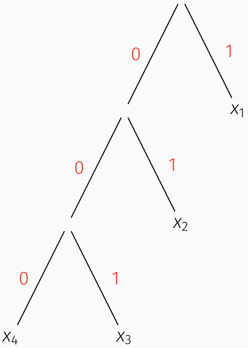
Prefix codes as tree:

Prefix codes can be represented on binary trees: codewords are given by leaves branches

Consider

$\mathcal{X}$	$\varphi(x_i)$
x_1	1
x_2	01
x_3	001
x_4	000

, then



Expected length:

Given a distribution of symbols $\mathsf{X} : \Omega \to \mathcal{X}$, the expected length of a symbol code $\varphi : \mathcal{X} \to \{0,1\}^+$ is

$$L(\varphi, \mathcal{X}) \stackrel{\text{def}}{=} \sum_{x \in \mathcal{X}} \ell(x)\, p(x) \quad \text{where } p(x) \stackrel{\text{def}}{=} \mathbb{P}(\mathsf{X} = x)$$

$\longrightarrow$ Expected length: **measure of efficiency**! We want it to be small

SOME EXAMPLES

$\mathcal{X}$	$\varphi_0(x_i)$	$p(x_i)$	$\ell(x_i)$	$\varphi_1(x_i)$	$p(x_i)$	$\ell(x_i)$	$\varphi_2(x_i)$	$p(x_i)$	$\ell(x_i)$
a	00	$\frac{1}{4}$	2	0	$\frac{1}{4}$	1	0	$\frac{1}{2}$	1
b	01	$\frac{1}{4}$	2	1	$\frac{1}{4}$	1	01	$\frac{1}{4}$	2
c	10	$\frac{1}{4}$	2	00	$\frac{1}{4}$	2	011	$\frac{1}{8}$	3
d	11	$\frac{1}{4}$	2	11	$\frac{1}{4}$	2	111	$\frac{1}{8}$	3

- $L(\varphi_0, \mathcal{X}) = 2$ is uniquely decodable $\big($prefix code$\big)$
- $L(\varphi_1, \mathcal{X}) = 1.75$ **but** it is not uniquely decodable: $\varphi_1(\mathrm{aa}) = \varphi_1(\mathrm{c})$
- $L(\varphi_2, \mathcal{X}) = 1.75$ is uniquely decodable **but** not prefix $\big($exercise$\big)$

What do you conclude?

$\mathcal{X}$	$\varphi_0(x_i)$	$p(x_i)$	$\ell(x_i)$	$\varphi_2(x_i)$	$p(x_i)$	$\ell(x_i)$
a	00	$\frac{1}{4}$	2	0	$\frac{1}{2}$	1
b	01	$\frac{1}{4}$	2	01	$\frac{1}{4}$	2
c	10	$\frac{1}{4}$	2	011	$\frac{1}{8}$	3
d	11	$\frac{1}{4}$	2	111	$\frac{1}{8}$	3

- $L(\varphi_0, \mathcal{X}) = 2$ is uniquely decodable
- $L(\varphi_2, \mathcal{X}) = 1.75$ is uniquely decodable

If we shorten $00 \mapsto 0$ in φ_0, then we keep unique decodability like in φ_2 if we we lengthen other codewords $\left(\textit{e.g. } 11 \mapsto 111\right)$

Constrained budget which can be spent on codewords, with shorter codewords being more expensive

What is the nature of this budget?

The code containing all the length 3 codewords has size 2^3

But what does happen if we want a prefix code with 0 and only length 3 codewords?

$\longrightarrow$ The only left possibility is $\{100, 110, 101, 111\}$ which has size $2^2 = \frac{2^3}{2}$

Give me your money:

You have 1 budget, codewords of length ℓ have cost $2^{-\ell}$. Codewords of length 3 $\left(\textit{resp. } 1\right)$ have cost $\frac{1}{8}$ $\left(\textit{resp. } \frac{1}{2}\right)$. If you spend more that your money, the code won't be uniquely decodable.

But, if

$$\sum_{x \in \mathcal{X}} 2^{-\ell(x)} \leq 1$$

is the code-uniquely decodable?

The code containing all the length 3 codewords has size 2^3

But what does happen if we want a prefix code with 0 and only length 3 codewords?

$\longrightarrow$ The only left possibility is $\{100, 110, 101, 111\}$ which has size $2^2 = \frac{2^3}{2}$

Give me your money:

You have 1 budget, codewords of length ℓ have cost $2^{-\ell}$. Codewords of length 3 $\left(\textit{resp.}\ 1\right)$ have cost $\frac{1}{8}$ $\left(\textit{resp.}\ \frac{1}{2}\right)$. If you spend more that your money, the code won't be uniquely decodable.

But, if

$$\sum_{x \in \mathcal{X}} 2^{-\ell(x)} \leq 1$$

is the code-uniquely decodable? **Yes: Kraft-McMillan inequalities**

McMillan's theorem:

There exists a uniquely decodable code with codewords of length $n_1, \ldots, n_K$ if and only if

$$\sum_{i=1}^{K} \frac{1}{2^{n_i}} \leq 1$$

Prefix codes are easy to decode, could we restrict our attention to prefix codes to simplify our life?

$$\Big(\text{prefix codes have a nice representation by trees} \Big)$$

Kraft inequality with prefix codes:

There exists a prefix code with codewords of length $n_1, \ldots, n_K$ if and only if

$$\sum_{i=1}^{K} \frac{1}{2^{n_i}} \leq 1$$

Proofs:

See Exercise Session

$\longrightarrow$ Uniquely decodable codes are not better than prefix codes!

Aim: minimizing over the code φ

$$L(\varphi, \mathcal{X}) = \sum_{x \in \mathcal{X}} \ell(x)\; p(x) \quad \text{where the } \; p(x) \overset{\text{def}}{=} \mathbb{P}(\mathsf{X} = x) \; \text{ are fixed}$$

Short codewords to the more probable symbols: reducing the expected length

But, Kraft inequality tells us that shortening some codewords necessarily lengthen others!

Shannon's theorem:

For any distribution $\mathsf{X} : \Omega \to \mathcal{X}$, there exists a prefix code φ with expected length satisfying

$$L(\varphi, \mathcal{X}) < H(\mathsf{X}) + 1$$

Furthermore, for any prefix code,

$$H(\mathsf{X}) \leq L(\varphi, \mathcal{X})$$

Proof:

- **Lower-bound**. First, define the distribution $q(x) \stackrel{\text{def}}{=} \frac{2^{-\ell(x)}}{z}$ where $z \stackrel{\text{def}}{=} \sum_{x'} 2^{-\ell(x')}$ which means that $\ell(x) = \log_2 1/q(x) - \log_2 z$. By Gibb's inequality,

$$\sum_x p(x) \log_2 1/q(x) \geq \sum_x p(x) \log_2 1/p(x)$$

Furthermore, by Kraft's inequality,

$$z = \sum_{x'} 2^{-\ell(x')} \leq 1$$

Therefore,

$$\begin{aligned} L(\varphi, \mathcal{X}) &= \sum_x p(x)\ell(x) = \sum_x p(x) \log_2 1/q(x) - \log_2 z \\ &\geq \sum_x p(x) \log_2 1/p(x) - \log_2 z \\ &\geq H(\mathsf{X}) \qquad \left(\text{as } z \leq 1\right) \end{aligned}$$

OPTIMAL CASE

Before, showing the upper-bound: some remarks

Remarks:

- Optimal source codelengths. $L(\varphi, \mathcal{X})$ is minimized and is equal to $H(\mathsf{X})$ if and only if the codelengths are equal to the Shannon information:

$$\sum_x 2^{-\ell(x)} = 1 \text{ and } \ell(x) = \log_2 1/p(x) \text{ but not necessarily an integer...}$$

- Implicit probabilities defined by codelengths. Conversely, any choice of codelengths implicitly defines a probability distribution

$$q(x) = \frac{2^{-\ell(x)}}{\sum_{x'} 2^{-\ell(x')}}$$

  for which those codelengths would be the optimal codelengths

Proof:

- **Upper-bound**. To mimic the proof of the lower-bound, set,

$$\ell(x) \stackrel{\text{def}}{=} \lceil \log_2 1/p(x) \rceil$$

where $\lceil \ell \rceil$ denotes the smallest integer greater than or equal to ℓ

By "Kraft inequality and prefix codes" (Slide 39), it exists a prefix code with these lengths as Kraft's inequality is satisfied,

$$\sum_x 2^{-\ell(x)} = \sum_x 2^{-\lceil \log_2 1/p(x) \rceil} \leq \sum_x 2^{-\log_2 1/p(x)} = \sum_x p(x) = 1.$$

Then we conclude,

$$L(\varphi, \mathcal{X}) = \sum_x p(x) \lceil \log_2 1/p(x) \rceil < \sum_x p(x) \left(\log_2 1/p(x) + 1 \right) = H(\mathbf{X}) + 1$$

OPTIMAL SOURCE CODING: HUFFMAN CODING

Proof of source coding theorem: the code we explicit $\Big($during the upper-bound$\Big)$ is not optimal!

Optimal code:

A uniquely decodeable coding φ_{opt} is said to be optimal if for all uniquely decodeable coding φ,

$$L(\mathcal{X}, \varphi_{\text{opt}}) \leq L(\mathcal{X}, \varphi)$$

Motivation:

An efficient algorithm to compute an optimal and prefix code with a given outcome distribution

$\longrightarrow$ Prefix codes enjoy an easy decoding algorithm via a "tree representation"!

Huffman coding algorithm: recursive construction

- If $L = \sharp\mathcal{X} = 2$, return $\varphi_L = \{0, 1\}$
- Otherwise, given $\mathcal{X} = \{x_1, \ldots, x_L\}$ and $p(x_1) \geq \cdots \geq p(x_L)$, build the new alphabet $\mathcal{Y} = \{x_1, \cdots, x_{L-2}, y_{L-1}\}$ with probability of outcomes $q(\cdot)$

$$q(x_i) = p(x_i) \quad \text{and} \quad q(y_{L-1}) = p(x_{L-1}) + p(x_L)$$

Let φ_{L-1} be a Huffman code for $\mathcal{Y}$, then build φ_L as:

- $\varphi_L(x_k) = \varphi_{L-1}(x_k)$, for $k = 1, \ldots, L-1$
- $\varphi_L(x_{L-1}) = \varphi_{L-1}(y_{L-1})\ 0$
- $\varphi_L(x_L) = \varphi_{L-1}(y_{L-1})\ 1$

Theorem:

Huffman coding is a prefix code which is optimal

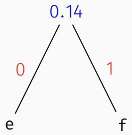
Furthermore, its average length belongs to $[H(\mathrm{X}), H(\mathrm{X}) + 1)$

$\longrightarrow$ In particular: if φ_{H} is the Huffman coding, then for all uniquely decodable code

$$L(\mathcal{X}, \varphi_{\mathrm{H}}) \leq L(\mathcal{X}, \varphi)$$

Huffman coding: use the probabilities of outcomes $p(x_i)$'s to build a tree **from the leaves**!
It equilibrates the probabilities at each level

x_i	$p(x_i)$	$-\log_2 p(x_i)$	$\varphi(x_i)$	$\ell(x)$
a	0.43	1.22	0	1
b	0.17	2.56	000	3
c	0.15	2.74	001	3
d	0.11	3.18	011	3
e	0.09	3.47	0100	4
f	0.05	4.32	0101	4

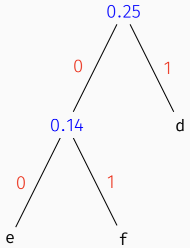


HUFFMAN CODING AS TREE

Huffman coding: use the probabilities of outcomes $p(x_i)$'s to build a tree **from the leaves**!

It equilibrates the probabilities at each level

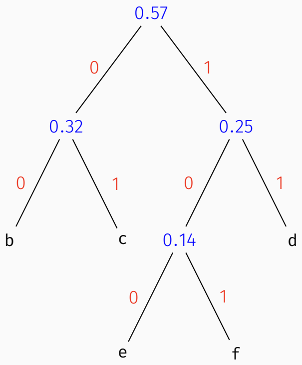
x_i	$p(x_i)$	$-\log_2 p(x_i)$	$\varphi(x_i)$	$\ell(x)$
a	0.43	1.22	1	1
b	0.17	2.56	000	3
c	0.15	2.74	001	3
d	0.11	3.18	011	3
e	0.09	3.47	0100	4
f	0.05	4.32	0101	4



Huffman coding: use the probabilities of outcomes $p(x_i)$'s to build a tree **from the leaves**!

It equilibrates the probabilities at each level

x_i	$p(x_i)$	$-\log_2 p(x_i)$	$\varphi(x_i)$	$\ell(x)$
a	0.43	1.22	1	1
b	0.17	2.56	000	3
c	0.15	2.74	001	3
d	0.11	3.18	011	3
e	0.09	3.47	0100	4
f	0.05	4.32	0101	4

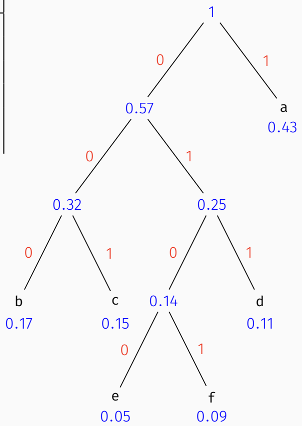


Huffman coding: use the probabilities of outcomes $p(x_i)$'s to build a tree **from the leaves**!

It equilibrates the probabilities at each level

x_i	$p(x_i)$	$-\log_2 p(x_i)$	$\varphi(x_i)$	$\ell(x)$
a	0.43	1.22	1	1
b	0.17	2.56	000	3
c	0.15	2.74	001	3
d	0.11	3.18	011	3
e	0.09	3.47	0100	4
f	0.05	4.32	0101	4

$H(X) = 2.248, \quad L(\varphi, \mathcal{X}) = 2.28$



Exercise:

Decompress: 010110010100

Huffman algorithm produces an optimal symbol code: but **not the end of the story**

We need to know beforehand the probabilities $p(x_i)$'s! And if we "make mistakes" by using another distribution q, then Huffman code compresses with rate $\approx H(p) + D_{KL}(p||q)$ instead of $H(p)$ $\big($see Exercise Session$\big)$

There is also an issue coming from a priori probabilities

When compressing ℓ symbols $x_1, \dots, x_\ell$, Huffman code consider them as independent

$\Big($not realistic in the case for instance of the language$\Big)$

An idea:

Pack symbols of $\mathcal{X}$ into symbols belonging to $\mathcal{X}^\ell$ for ℓ large enough:

- Don't take into account a priori probabilities but more realistic
- $\frac{L(\varphi,\mathcal{X}^\ell)}{H(X^{\otimes \ell})} \xrightarrow[\ell \to +\infty]{} 1$

However, memory complexity $\sharp\left(\mathcal{X}^\ell\right)$

Huffman codes were widely trumpeted as "optimal": **but** have many defects for practical purposes!

They are optimal **among symbol codes**: each $x \in \mathcal{X}$ is mapped to an integer number of bits

Could we design other codes than symbol codes?

$\longrightarrow$ Stream Codes: see arithmetic coding at Lecture 4 $\Big($Programming Session$\Big)$!

EXERCISE SESSION