Information, Calcul et Communication: Cours 10

Résumé de la dernière leçon: entropie

si p_i est la probabilité d'apparition de la lettre a_i dans la séquence $\mathcal{X}$:

$$H(\mathcal{X}) = \sum_{i=1}^{n} p_i \log_2\left(\frac{1}{p_i}\right) \quad \in [0, \log_2(n)]$$

= mesure de la "quantité d'information" présente dans la séquence $\mathcal{X}$

Plan de cette leçon: compression de données

- compression sans pertes:
  - codages de Shannon-Fano et Huffman
    principe: ce qui revient souvent est abrégé
  - (premier) théorème de Shannon
- compression avec pertes

Introduction

Quel type de données peuvent être compressées?

- le langage
- le son (voix/musique), les images (photos), les vidéos
- tout type de données numériques

Le principe de base derrière la compression de données est la suppression de la redondance présente dans ces données.

Introduction

Le français est plein de redondance! Pour preuve, ces deux phrases:

~~Cette phrase dont les lettres ont été à moitié effacées est encore parfaitement lisible!~~

Si on liasse la permèire et la drneière ltertes à la bnone palce dnas cquahe mot, arols la prhsae est assui pafiatermnet lbiisle (ou pseruqe!)

Pourquoi donc tant de redondance dans la langue française ?

- pour pouvoir se comprendre, tout simplement!
- pour être capable de lire un texte même s'il contient des fotes d'orthografe...

Introduction

On distingue deux types de compression:

- la compression sans pertes: lorsqu'on désire retrouver l'intégralité des données stockées sous forme compressée.

  Exemples: billets pour un concert, déclaration d'impôts, bulletins de vote, articles scientifiques

- la compression avec pertes: lorsqu'on n'est pas tant à cheval que ça sur les détails et qu'on s'autorise un peu de distorsion.

  Exemples: émissions podcastées et morceaux de musique en format mp3, partage de photos sur le web, vidéos youtube...

Exemples d'algorithmes de compression sans pertes dans la vie de tous les jours:

- Langage SMS: “slt”, “tqt”, “mdr”, etc.
  les mots qui reviennent souvent sont abrégés.

- Code Morse: A = “.–” E = “.” S= “...” T= “–”
  tandis que X= “–..–” Z = “– –..”

- Acronymes: EPFL, UNIL, GC, MX, etc.

Compression sans pertes :

Codages de Shannon-Fano et Huffman

Pour expliquer ces deux systèmes d'encodage, nous les illustrerons sur la séquence de lettres :

AMPARAFARAVOLA (ville de Madagascar, 14 lettres)

Calculons tout d'abord l'entropie de cette séquence :

lettre	nb apparitions	probabilité
A	6	3/7
R	2	1/7
M	1	1/14
P	1	1/14
F	1	1/14
V	1	1/14
O	1	1/14
L	1	1/14

entropie :

$$H = \frac{3}{7}\log_2\left(\frac{7}{3}\right) + \frac{1}{7}\log_2(7) + 6\cdot\frac{1}{14}\cdot\log_2(14)$$

$$= \log_2(7) - \frac{3}{7}\log_2(3) + \frac{3}{7}$$

$$\simeq 2.55$$

- Si on désire maintenant représenter cette séquence de lettres sous forme de 0 et de 1, un premier choix s'offre à nous : il y a 8 lettres différentes ; on peut donc encoder chaque lettre avec 3 bits ($3 = \log_2 8$) : A → 000, R → 001, M → 010, etc. et donc la séquence entière avec 3×14 (nb total de lettres) = 42 bits.
- Cependant, cette méthode ne tient pas compte du fait que certaines lettres se répètent plus que d'autres dans la séquence ; en utilisant ce fait, on peut réduire de manière significative le nombre de bits à utiliser pour représenter la séquence.

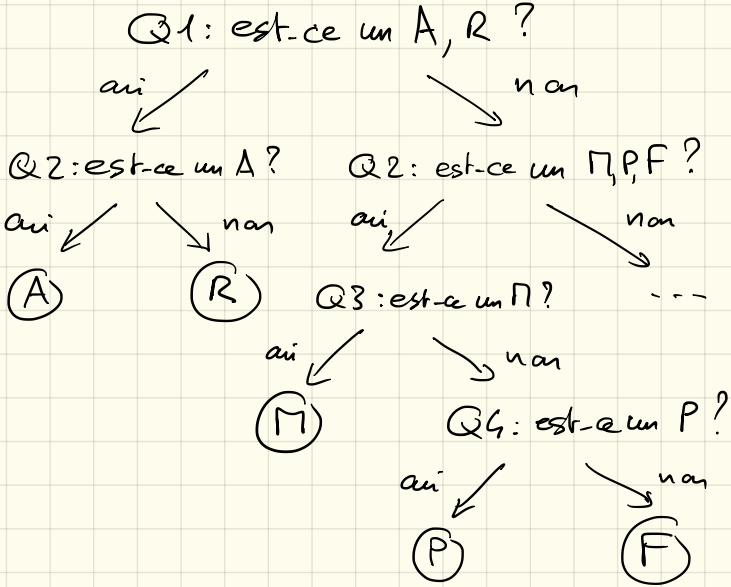
Codage de Shannon-Fano

Reprenons le jeu des questions de la semaine dernière, en ordonnant les lettres par nombre décroissant d'apparitions :

lettre	nb app.	nb questions
A	6	2
R	2	2
M	1	3
P	1	4
F	1	4
V	1	3
O	1	4
L	1	4

Q2 Q1 Q3 Q4 Q4 Q2 Q3 Q4

arbre



Remarques :

- Comme tout n'est plus parfaitement divisible par deux ici, on doit poser à chaque étape une question qui divise l'ensemble des possibilités en deux parties les plus égales possibles (tout en respectant l'ordre décroissant des nombres d'apparition : regrouper des lettres rares et fréquentes n'est pas une bonne idée pour ce qui va suivre) ; ceci nous amène parfois à devoir faire des choix !
- On n'a plus non plus une correspondance parfaite entre le nombre d'apparitions d'une lettre et le nombre de questions qu'il faut poser pour deviner celle-ci.

Règles d'encodage :

1. Nombre de questions à poser pour deviner une lettre
   = nombre de bits utilisés pour représenter cette lettre
2. Réponse oui ⟷ valeur 1 ; réponse non ⟷ valeur 0

Dans notre exemple :

lettre	nb apparitions	nb questions = nb bits	mot de code correspondant
A	6	2	11
R	2	2	10
M	1	3	011
P	1	4	0101
F	1	4	0100
V	1	3	001
O	1	4	0001
L	1	4	0000

= dictionnaire

(cf. branches de l'arbre)

Ainsi donc, on représente la séquence AMPARAFARAVOLA par 11011010111110...

- Nombre de bits utilisés : $8\times 2 + 2\times 3 + 4\times 4 = 16+6+16 = 38$ bits

  $\Rightarrow$ nb moyen de bits par lettre $= \frac{38}{14} \cong 2.71 < 3$; on gagne de la place !

- Par autant, la séquence de bits est-elle décodable ?

  (en supposant qu'on est en possession du dictionnaire)

  Remarquez : • tous les mots de code n'ont pas la même longueur, mais :

  • tous les mots de code sont différents

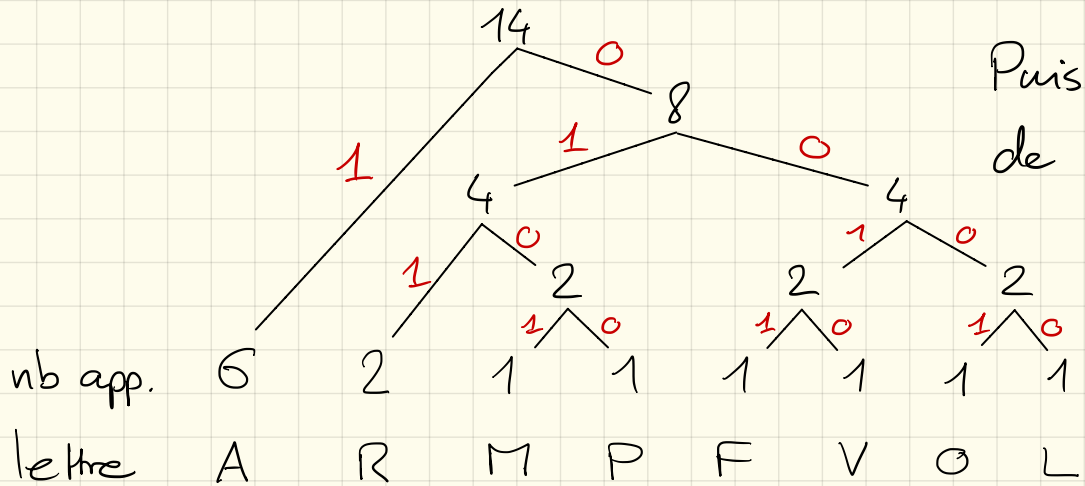
  • mieux : aucun mot de code n'est le préfixe d'un autre

$\Rightarrow$ on peut lire de gauche à droite sans problème : 11 011 0101 11110 ...

A M P A R

Codage de Huffman

Principe : Au lieu de construire l'arbre "depuis le haut" avec le jeu des questions, construire celui-ci "depuis le bas" en rejoignant les lettres les moins fréquentes au fur et à mesure :



Puis on parcourt l'arbre de haut en bas avec la règle :

{ gauche = 1
{ droite = 0

Ainsi on obtient un nouveau dictionnaire :

lettre	nb app.	nb bits	mot de code
A	6	1	1
R	2	3	011
M	1	4	0101
P	1	4	0100
F	1	4	0011
V	1	4	0010
O	1	4	0001
L	1	4	0000

dictionnaire (à nouveau sans préfixe)

Nombre de bits utilisés : $6\times1 + 2\times3 + 6\times4 = 6+6+24 = 36$

$\Rightarrow$ nombre moyen de bits par lettre : $\frac{36}{14} \simeq 2.57$; encore mieux !

(En général, ce mode de compression permet de réduire de 30% la taille d'un fichier (format zip).)

Remarques :

- entropie $\leq$ nb bits / lettre (Huffman) $\leq$ nb bits / lettre (Shannon-Fano) $\leq$ nb bits / lettre (encodage régulier)

  2.55 — 2.57 — 2.71 — 3

  Nous allons y revenir avec le (premier) théorème de Shannon !

- Que ce soit avec l'encodage de Shannon-Fano ou l'encodage de Huffman, on peut être amené à faire des choix :
  - Avec Shannon-Fano, ces choix peuvent impacter la structure de l'arbre et aussi le nombre total de bits utilisés.
  - Avec Huffman, ces choix peuvent également impacter la structure de l'arbre, mais le nombre total de bits utilisés reste le même dans tous les cas !

Le (premier) Théorème de Shannon

Définitions :

- Un code binaire est un ensemble C dont les éléments $c_1 \dots c_n$ (= mots de code) sont des suites de bits de longueur finie. (Ex : $C = \{11, 10, 011, 0101, \text{etc.}\}$)
- On note ℓ_j = la longueur du mot de code c_j
- Un code binaire est dit sans préfixe si aucun mot de code n'est le préfixe d'un autre.

  ⟶ tous les mots du code sont différents

  ⟶ une séquence formée de ces mots de code peut être lue de gauche à droite

- le code binaire $C = \{c_1, \dots, c_n\}$ peut être utilisé comme dictionnaire pour représenter une séquence x de lettres tirée d'un alphabet $A = \{a_1, \dots, a_n\}$.
(chaque lettre a_j est représentée par un mot de code c_j, de longueur ℓ_j). Exemple: $A \leftrightarrow 11$, $R \leftrightarrow 10$, $\Pi \leftrightarrow 011$, $P \leftrightarrow 0101$, etc.

- Si les lettres a_j apparaissent avec probabilités p_j dans la séquence x, alors la longueur moyenne du code est définie par: $L(C) = \sum_{j=1} p_j \ell_j$
(= nombre moyen de bits utilisés par lettre)
(= nombre total de bits / nombre total de lettres)

Théorème (Shannon)

Quel que soit le code binaire C sans préfixe utilisé pour représenter une séquence X donnée, l'inégalité suivante est vérifiée : $L(C) \geq H(X)$.

("On ne peut pas compresser une séquence au-delà de son entropie.")

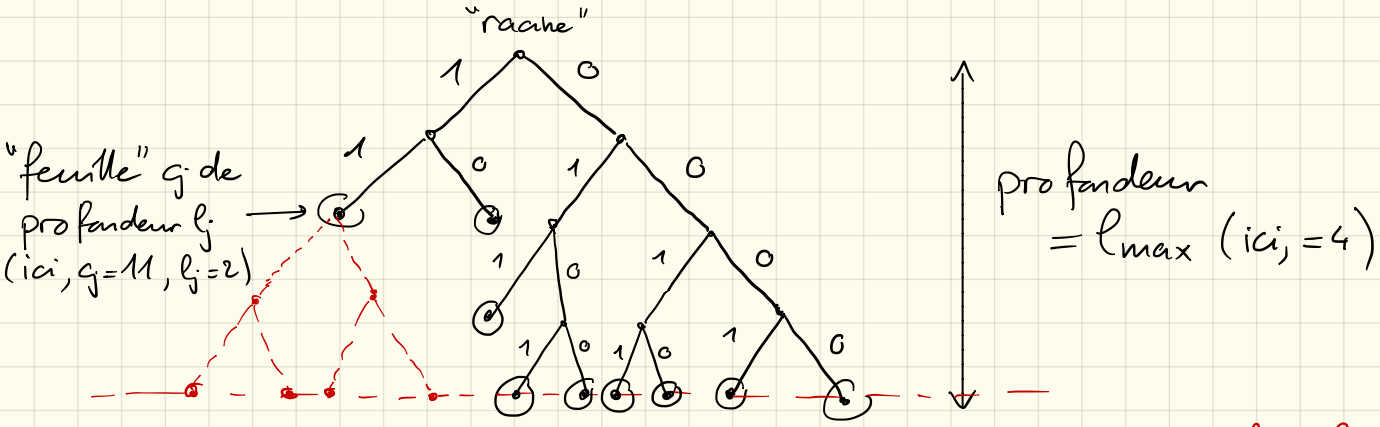
Lemme : inégalité de Kraft

Soit $C = \{c_1, \dots, c_n\}$ un code binaire sans préfixe.

Alors $\sum_{j=1}^{n} 2^{-\ell_j} \leq 1$ (où ℓ_j = longueur de c_j)

Preuve du lemme (illustration)

Représentation d'un code binaire en forme d'arbre :



- nb de descendants d'un mot de code c_j au niveau $\ell_{max} = 2^{\ell_{max} - \ell_j}$
- code sans préfixe $\Longrightarrow$ tous les descendants sont disjoints.
- nombre de descendants au niveau $\ell_{max} \leq 2^{\ell_{max}}$

$$\Longrightarrow \sum_{j=1}^{n} 2^{\ell_{max} - \ell_j} \leq 2^{\ell_{max}} \Longrightarrow \sum_{j=1}^{n} 2^{-\ell_j} \leq 1 \quad \#$$

Preuve du Théorème de Shannon

A var: $L(C) \geq H(X)$ i.e. $H(X) - L(C) \leq 0$?

$$H(X) - L(C) = \sum_{j=1}^{n} p_j \log_2\left(\frac{1}{p_j}\right) - \sum_{j=1}^{n} p_j \ell_j$$

$$= \sum_{j=1}^{n} p_j \left(\log_2\left(\frac{1}{p_j}\right) - \ell_j \right)$$

$$\log_2\left(\frac{1}{p_j}\right) + \log_2\left(2^{-\ell_j}\right) = \log_2\left(\frac{2^{-\ell_j}}{p_j}\right)$$

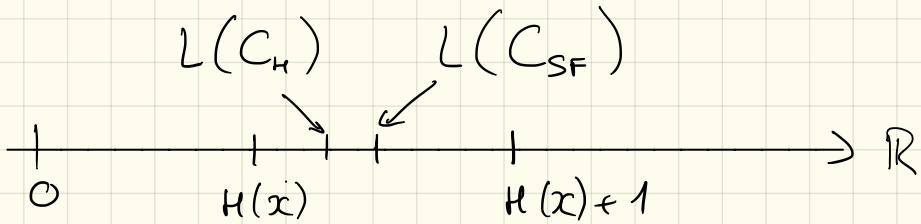
$$= \sum_{j=1}^{n} p_j \log_2\left(\frac{2^{-\ell_j}}{p_j}\right)$$

(concavité de $\log_2$: $\sum_{j=1}^{n} p_j \log_2(x_j) \leq \log_2\left(\sum_{j=1}^{n} p_j x_j\right) \quad \forall x_1, \dots, x_n > 0$)

$x_j = \frac{2^{-\ell_j}}{p_j}$: $H(X) - L(C) \leq \log_2\left(\sum_{j=1}^{n} p_j \frac{2^{-\ell_j}}{p_j}\right) = \log_2\left(\underbrace{\sum_{j=1}^{n} 2^{-\ell_j}}_{\leq 1 \text{ (lemme)}}\right) = 0$ #

On peut montrer de manière plus détaillée:

$$H(X) \leq L(C_{\text{Huffman}}) \leq L(C_{\text{Shannon-Fano}}) \leq H(X)+1$$



Et on peut montrer encore qu'on ne peut pas faire mieux que Huffman, même quand $L(C_H) > H(X)$.

Note: Et bien sûr, il existe des cas où "tout marche bien" ie., où $H(X) = L(C_H) = L(C_{SF})$.

Compression avec pertes

Est-il possible de passer en-dessous de la limite de Shannon?
Oui, mais pas n'importe comment!

Si par exemple on essaye d'utiliser le dictionnaire suivant
pour représenter la séquence AMPARAFARALOVA:

lettre	A	R	M	P	F	L	O	V
mot de code	1	0	11	10	01	00	111	110

Alors l'encodage de cette séquence est: 1111010...

mais son décodage est impossible: OPP? MMFR? AMPAR?

On est ici (encore une fois) victime de l'effet stroboscopique!

Compression avec pertes

En fait, nous avons déjà vu deux algorithmes de compression avec pertes:

- La représentation des nombres réels en virgule flottante
- Le filtrage des signaux avant échantillonnage

Pour le son, un célèbre algorithme de compression avec pertes a donné lieu au format mp3 (Brandenburg, 1993).

$\longrightarrow$ réduction de 90% de la taille d'un fichier audio
(par rapport à la taille d'un fichier "wave" enregistré à 44.1 kHz)

Compression avec pertes

En ce qui concerne les images, on peut citer:

- la pixellisation: procédé simple qui consiste à moyenner les couleurs sur des pixels de tailles plus ou moins grandes:

- le format jpeg ("Joint Photographic Experts Group", créé en 1992): savant mélange de plusieurs ingrédients...

Et finalement, pour la vidéo:

- le format mpeg ("Moving Expert Picture Group", créé en 1988)