

Information, Calcul et Communication

Leçon 2.3: Compression de données (1ère partie)

O. Lévêque – Faculté Informatique et Communications

Introduction

Lors des deux leçons précédentes, nous avons répondu aux questions suivantes:

Comment représenter / capter la réalité physique avec des bits ?

Comment restituer cette réalité à partir de bits ?

Lors des deux leçons à venir, nous allons tenter de répondre aux questions suivantes:

Comment mesurer la quantité d'information présente dans des données ?

Comment stocker des données en utilisant le moins d'espace possible (sans ou avec pertes) ?

Introduction (2)

Pourquoi donc vouloir compresser des données ?

pour réduire l'espace-mémoire utilisé lors du stockage de données

pour réduire le temps de transmission et les problèmes de congestion lors de la transmission des données

Cependant, avec les progrès de la technique, ne suffit-il pas d'attendre un peu pour avoir de meilleures performances?

Certes, mais on veut toujours exploiter un système au maximum de ses capacités!

Introduction (3)

Le français est plein de redondance! Pour preuve, ce texte:

Selon une étude de l'Université de Cambridge, l'ordre des lettres dans un mot n'a pas d'importance, la seule chose importante est que la première et la dernière soient à la bonne place. Le reste peut être dans un désordre total et vous pouvez toujours lire sans problème. C'est parce que le cerveau humain ne lit pas chaque lettre elle-même, mais le mot comme un tout.

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 fautes d'orthographe...

Introduction (4)

On distingue deux types de compression:

la compression sans pertes, lorsqu'on désire retrouver l'intégralité des données stockées sous forme comprimée.

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...

Plan

- **Plan détaillé des deux leçons à venir:**
- **Aujourd'hui:**
 - notion d'entropie
 - compression sans pertes
 - algorithme de Shannon-Fano
- **La semaine prochaine (*possibles modifications*)**
 - algorithme de Shannon-Fano (bis)
 - analyse de performance – théorème de Shannon
 - compression avec pertes

Entropie

Voici une séquence de 16 lettres:

A B C D E F G H I J K L M N O P

Jeu n⁰ 1: Vous devez deviner quelle lettre j'ai choisi au hasard en posant un *nombre minimum de questions*, auxquelles je ne peux répondre que par **oui** ou par **non**.

Solution: 4 questions sont nécessaires et cela quelle que soit la lettre à trouver (leçon 1.1: algorithme de dichotomie). On dit que *l'entropie* de cette séquence est égale à 4.

Remarquer que $16 = 2^4$, autrement dit $4 = \log_2(16)$

Entropie (2)

Voici une autre séquence de 16 lettres (sans compter les espaces):

I L F A I T B E A U A I B I Z A

Jeu N° 2: Le jeu est le même qu'avant!

Remarques:

Je choisis la *position* de la lettre au hasard, de manière uniforme.

Vous ne devez deviner que la lettre elle-même, *pas sa position*.

Combien de questions binaires sont-elles nécessaires en moyenne pour deviner la lettre?

Entropie (3)

I L F A I T B E A U A I B I Z A

Solution: classer les lettres dans l'ordre décroissant du nombre d'apparitions dans la séquence:

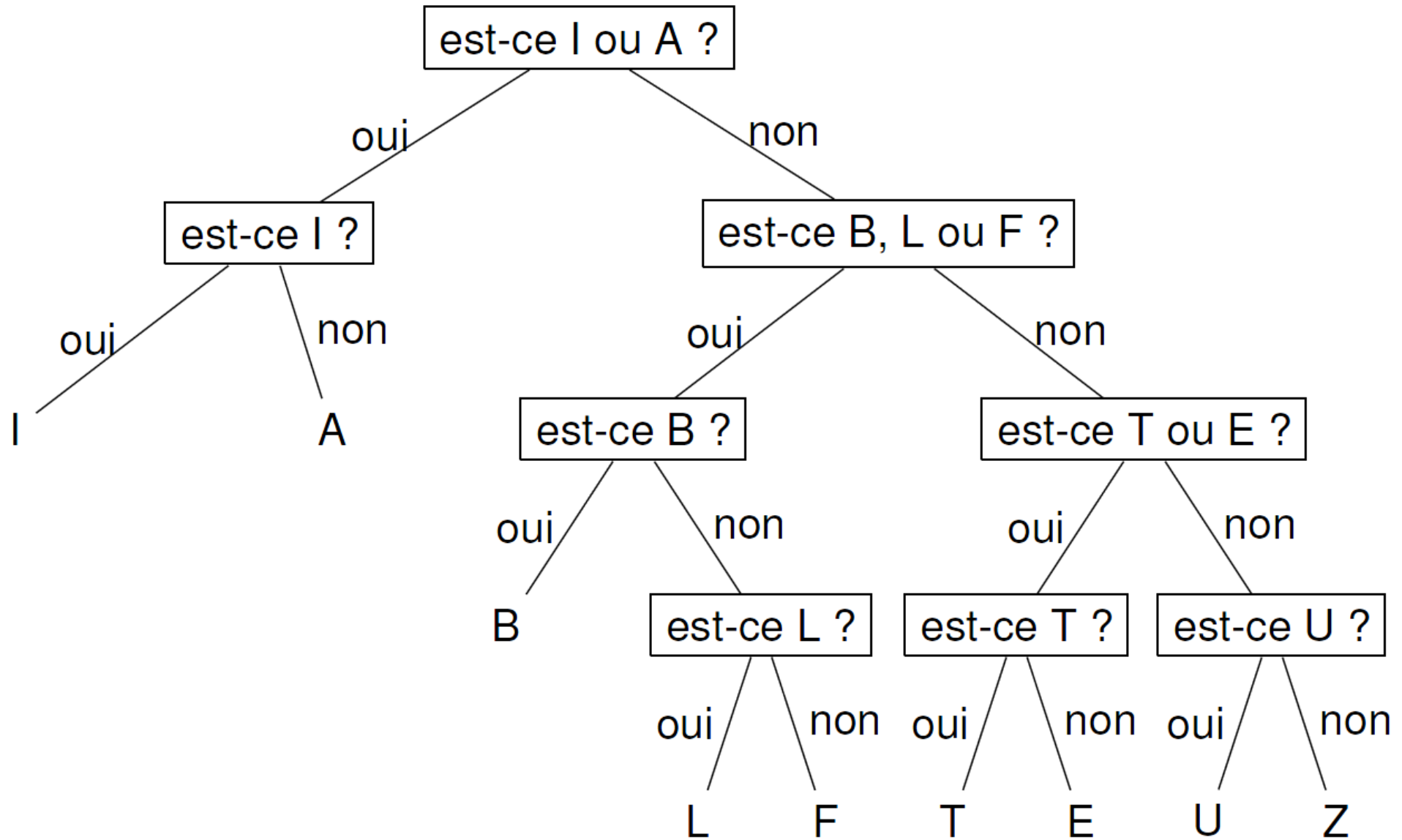
lettre	I	A	B	L	F	T	E	U	Z
nb d'apparitions	4	4	2	1	1	1	1	1	1

L'idée est la même que précédemment: on sépare l'ensemble des lettres en deux parties égales *en termes de nombre d'apparitions*, ce qui donne:

Question N° 1: est-ce que la lettre est un I ou un A?

Si la réponse est **oui**: **question n° 2:** est-ce que la lettre est un I?

Si la réponse est **non**: **question n° 2:** est-ce que la lettre est un B, un L ou un F? etc.



Entropie (4)

IL FAIT BEAU A IBIZA

lettre	I	A	B	L	F	T	E	U	Z
nb d'apparitions	4	4	2	1	1	1	1	1	1
nb de questions	2	2	3	4	4	4	4	4	4

Nombre de questions à poser en moyenne:

$$= \frac{2 \times 4}{16} \times 2 + \frac{1 \times 2}{16} \times 3 + \frac{6 \times 1}{16} \times 4 = \frac{16 + 6 + 24}{16} = \frac{46}{16} = 2.875$$

On dit que l'entropie de cette séquence est égale à 2.875.

Entropie (5)

Voici encore une autre séquence de 16 lettres:

A A A A A A A A A A A A A A A A

Jeu n⁰ 3: Le jeu est encore le même qu'avant.

Cette fois-ci, *aucune* question n'est nécessaire pour deviner la lettre choisie!

On dit que l'entropie de cette séquence est égale à 0.

Entropie (6) : lien avec la probabilité d'apparition d'une lettre

I L F A I T B E A U A I B I Z A

lettre	I	A	B	L	F	T	E	U	Z
nb d'apparitions	4	4	2	1	1	1	1	1	1
nb de questions	2	2	3	4	4	4	4	4	4

Pour deviner une lettre qui apparaît 1 fois sur 16 (i.e. probabilité de $1/16$), on a besoin de 4 questions. $4 = \log_2(16)$

Pour deviner une lettre qui apparaît 2 fois sur 16 (i.e. probabilité de $1/8$), on a besoin de 3 questions. $3 = \log_2(8)$

Pour deviner une lettre qui apparaît 4 fois sur 16 (i.e. probabilité de $1/4$), on a besoin de 2 questions. $2 = \log_2(4)$

AAAAAAAAAAAAAAAAAAAA

Pour deviner une lettre qui apparaît 16 fois sur 16 (i.e. probabilité de 1), on a besoin de 0 questions. $0 = \log_2(1)$

En résumé, pour deviner une lettre qui apparaît avec une probabilité p ,
on a besoin de $\log_2(1/p)$ questions.

Entropie (6') : cas particulier où la probabilité est de la forme 2^{-k}

IL FAIT BEAU A IBIZA

lettre	I	A	B	L	F	T	E	U	Z
probabilité p	2^{-2}	2^{-2}	2^{-3}	2^{-4}	2^{-4}	2^{-4}	2^{-4}	2^{-4}	2^{-4}
nb de questions	2	2	3	4	4	4	4	4	4

- Deviner une lettre ayant une probabilité 2^{-k} est de même nature que d'isoler **un** élément parmi 2^k éléments.
- La recherche par dichotomie nous a enseigné qui suffit de **k** divisions de l'ensemble de 2^k éléments pour trouver **un** élément particulier
- Cela revient à poser les **k** questions binaires pour trouver **un** élément particulier
- Une lettre de probabilité 2^{-k} a seulement besoins de **k** questions

Entropie (7)

Définition générale:

Soit X une séquence de lettres provenant d'un alphabet $A = \{a_1, \dots, a_n\}$. Soit p_j la probabilité d'apparition de la lettre a_j dans la séquence

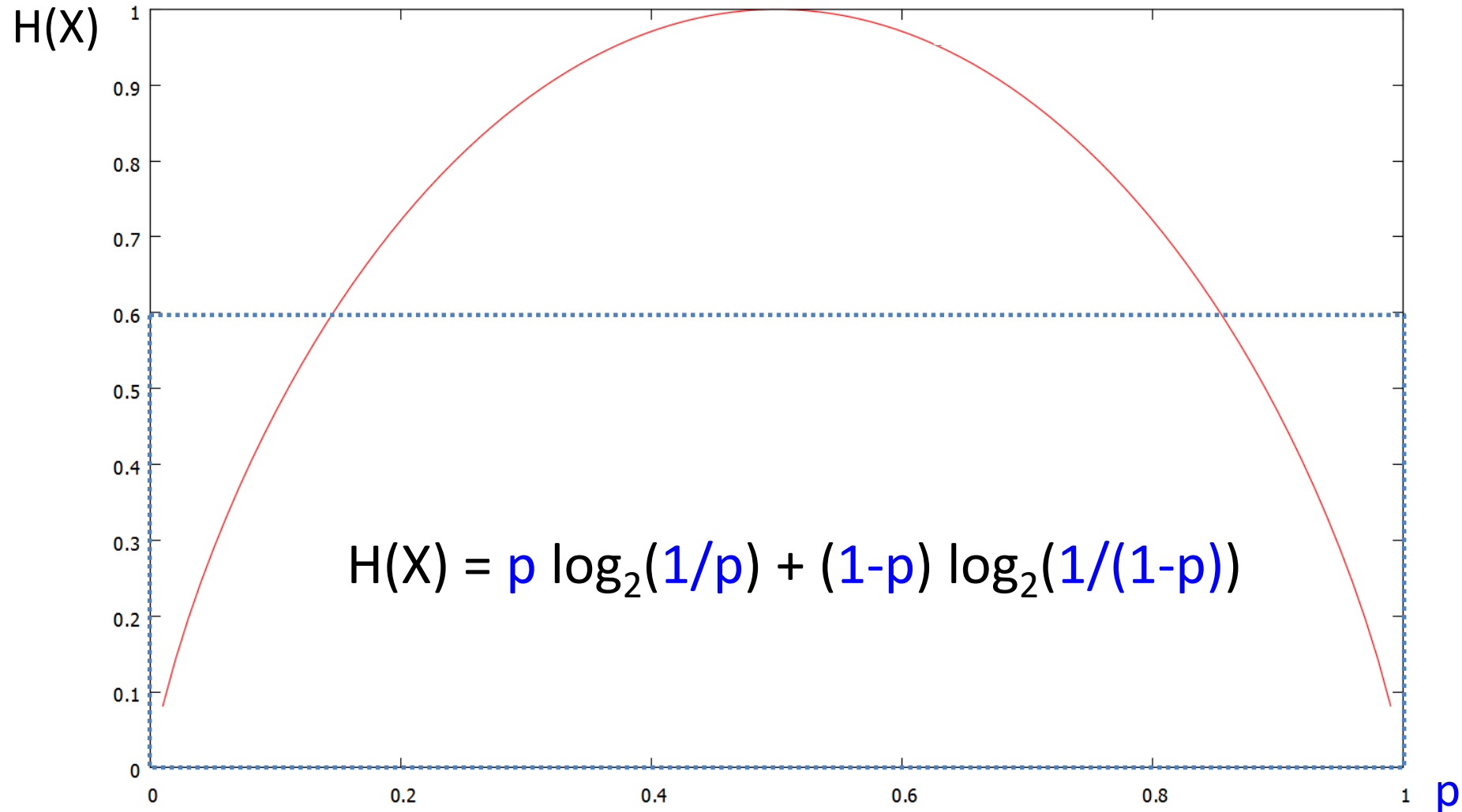
(remarquer que $0 \leq p_j \leq 1$ pour tout j et que $p_1 + \dots + p_n = 1$).

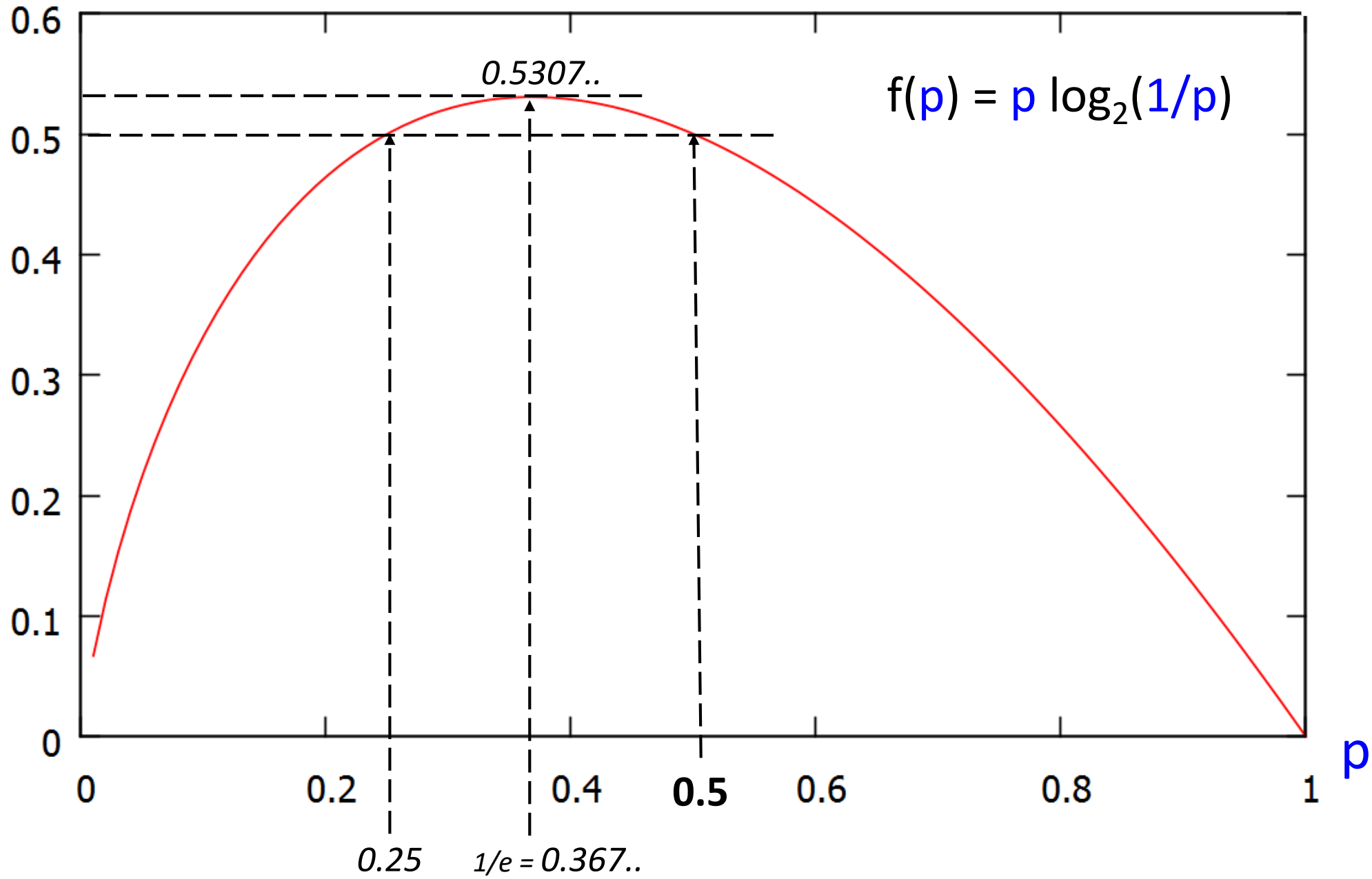
L'entropie de la séquence X est définie par

$$H(X) = p_1 \log_2 \left(\frac{1}{p_1} \right) + \dots + p_n \log_2 \left(\frac{1}{p_n} \right)$$

Par convention, si $p_j = 0$, alors on pose $p_j \log_2 \left(\frac{1}{p_j} \right) = 0$.

Exemple: entropie d'une séquence X de N lettres,
avec seulement DEUX lettres distinctes de probabilités respectives p et $(1-p)$
=> La formule générale de l'entropie $H(X)$ repose sur les probabilités d'apparition



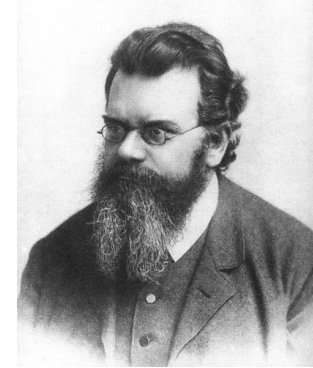


Entropie (8)

Origine en physique (Boltzmann, 1872):

L'entropie mesure le **désordre** dans un système physique.

Ludwig Eduard Boltzmann (1844-1905) ardent défenseur de l'existence des atomes
père de la physique statistique



Théorie de l'information (Shannon, 1948):

L'entropie mesure la **"quantité d'information"**
contenue dans un signal.

Claude Edwood Shannon (1916-2001) mathématicien, ingénieur électricien,
cryptologue, père de la théorie de l'information, jongleur...



Ou encore (comme on vient de le voir), a établi l'expression générale de l'entropie
basée sur les probabilités d'apparition des lettres.

Entropie (9)

Interprétation:

M A P J O F B G I L C K D E H N

$$H(X) = 4$$

I L F A I T B E A U A I B I Z A

$$H(X) = 2.875$$

A A A A A A A A A A A A A A A A

$$H(X) = 0$$

Plus il y a de lettres différentes, plus il y a de désordre, plus il y a de *variété* et donc “d’information” dans le message.

Plus il y a de lettres semblables, moins il y a de désordre, plus il y a de *redondance* et donc moins d’“information” dans le message.

SpeakUp room 21158 :

Which of the following statement is true about ENTROPIE and DESORDRE ?

- A. On ne peut pas comparer leur entropie car ces deux mots n'ont pas le même ensemble de lettres
- B. Ils ont la même entropie $H=3$ car ils ont le même nombre total de 8 lettres
- C. $H(\text{ENTROPIE}) > H(\text{DESORDRE})$
- D. $H(\text{DESORDRE}) > H(\text{ENTROPIE})$

Entropie (10)

Une remarque, et quelques propriétés de l'entropie:

$$H(X) = p_1 \log_2 \left(\frac{1}{p_1} \right) + \dots + p_n \log_2 \left(\frac{1}{p_n} \right)$$

Pour une probabilité d'apparition $0 \leq p \leq 1$ donnée, $\log_2(1/p) \geq 0$.
mais ça n'est pas forcément un nombre entier (ex: si $p = 1/3$).

$H(X) \geq 0$ en général, et $H(X) = 0$ si et seulement si l'**ordre** est total
c'est-à-dire si toutes les lettres sont les mêmes \Leftrightarrow leur probabilité d'apparition vaut 1

Si n est la taille de l'alphabet utilisé, $H(X) \leq \log_2(n)$ en général

$H(X) = \log_2(n)$ si et seulement si le **désordre** est total
c'est-à-dire si toutes les lettres sont différentes \Leftrightarrow leur probabilité d'apparition valant $1/n$ on a : $H(X) = n \cdot (1/n) \log_2(n) = \log_2(n)$.

$$H(X) \geq 0$$

Remarquer simplement que : $0 \leq p_j \leq 1$

D'où $1/p_j \geq 1$

D'où $\log_2(1/p_j) \geq \log_2(1)$

D'où $\log_2(1/p_j) \geq 0$

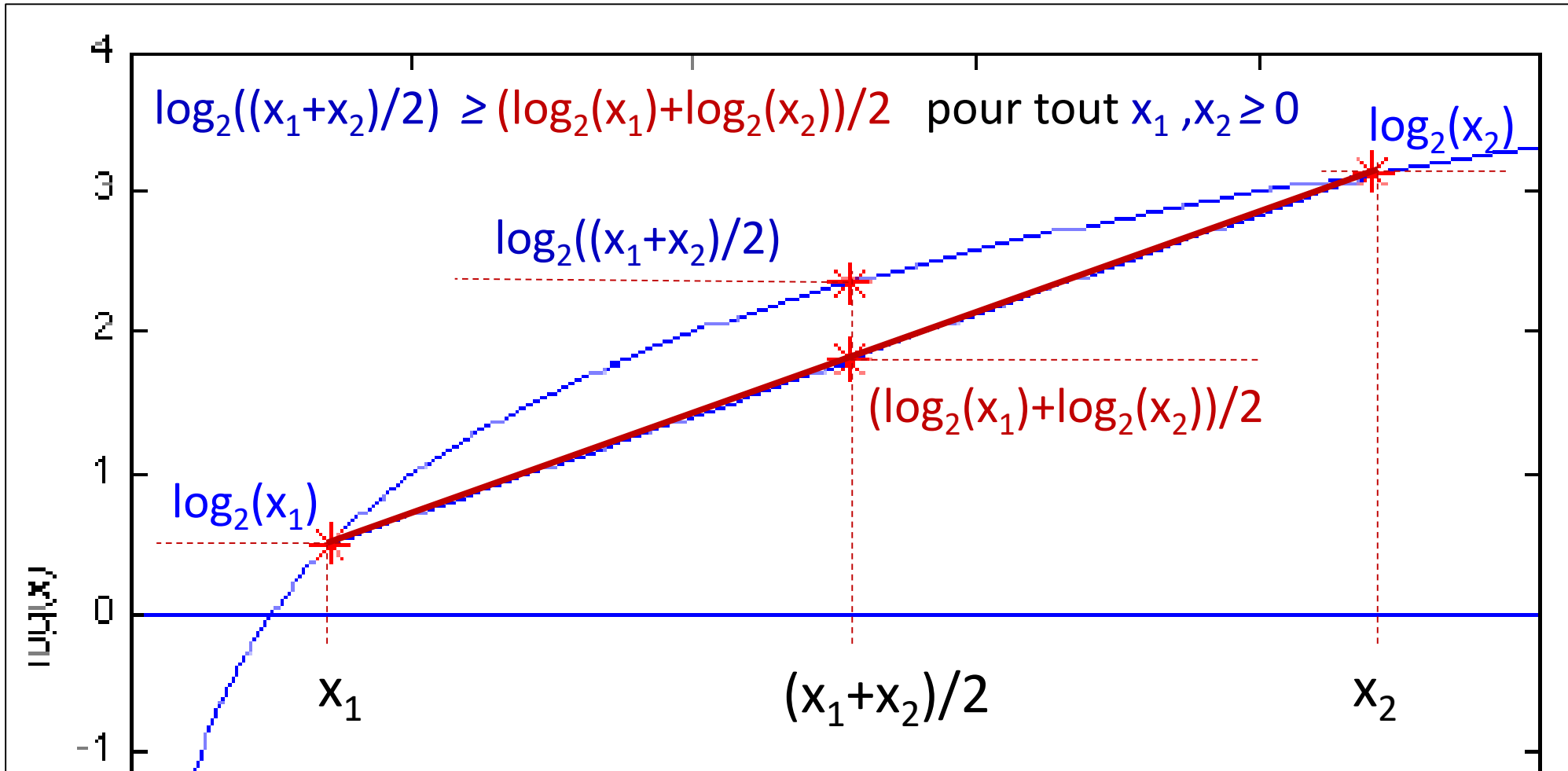
D'où $p_j \log_2(1/p_j) \geq 0$

Donc $H(X) \geq 0.$

Vérifications (2):

$$H(X) \leq \log_2(n)$$

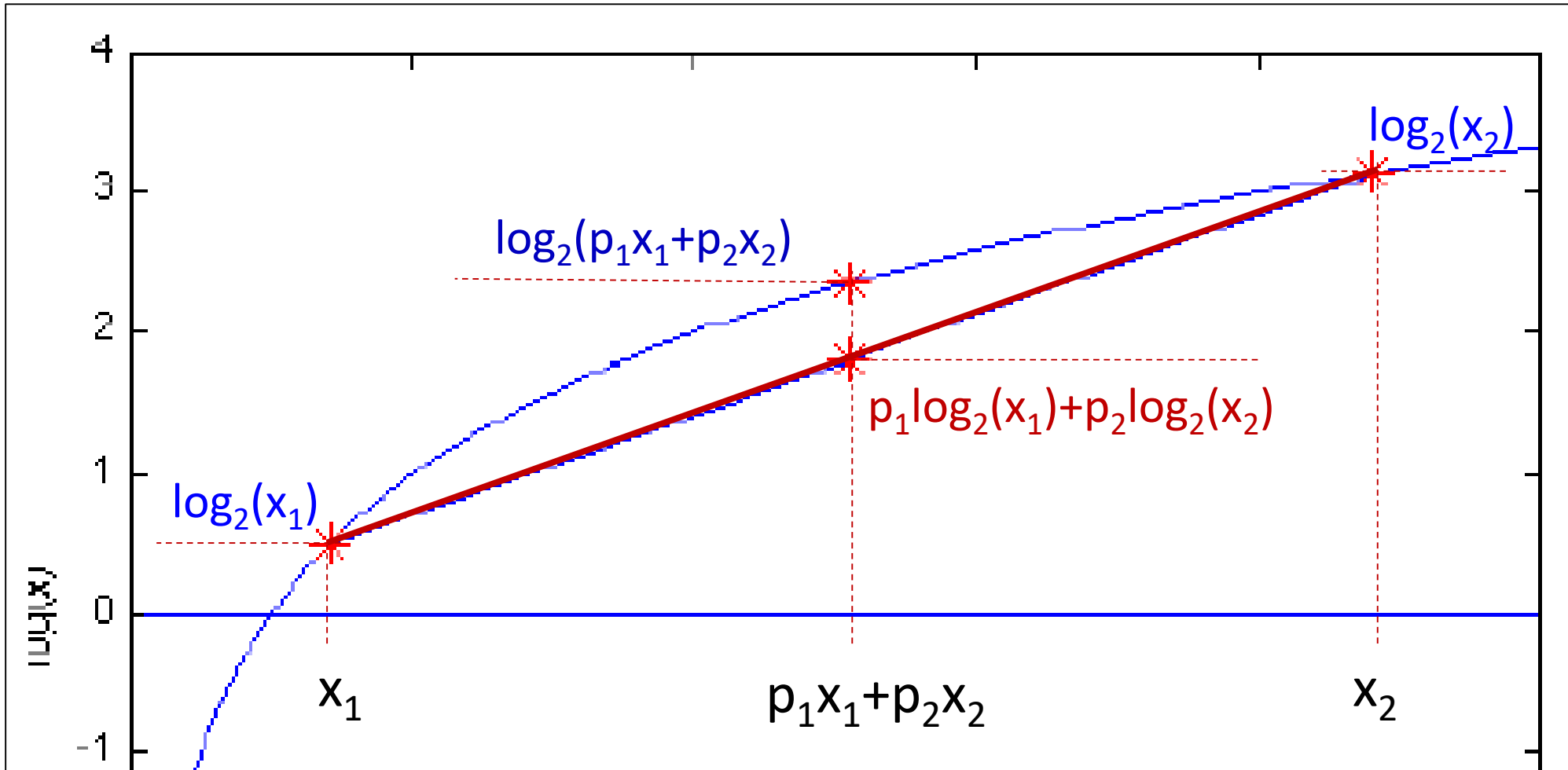
Remarquer que la fonction $f(x) = \log_2(x)$ est concave pour $x \geq 0$:



Vérifications (3):

Plus généralement, si $0 \leq p_1, p_2 \leq 1$ et $p_1 + p_2 = 1$, alors:

$$\log_2(p_1x_1+p_2x_2) \geq p_1\log_2(x_1)+p_2\log_2(x_2) \quad \text{pour tout } x_1, x_2 \geq 0$$

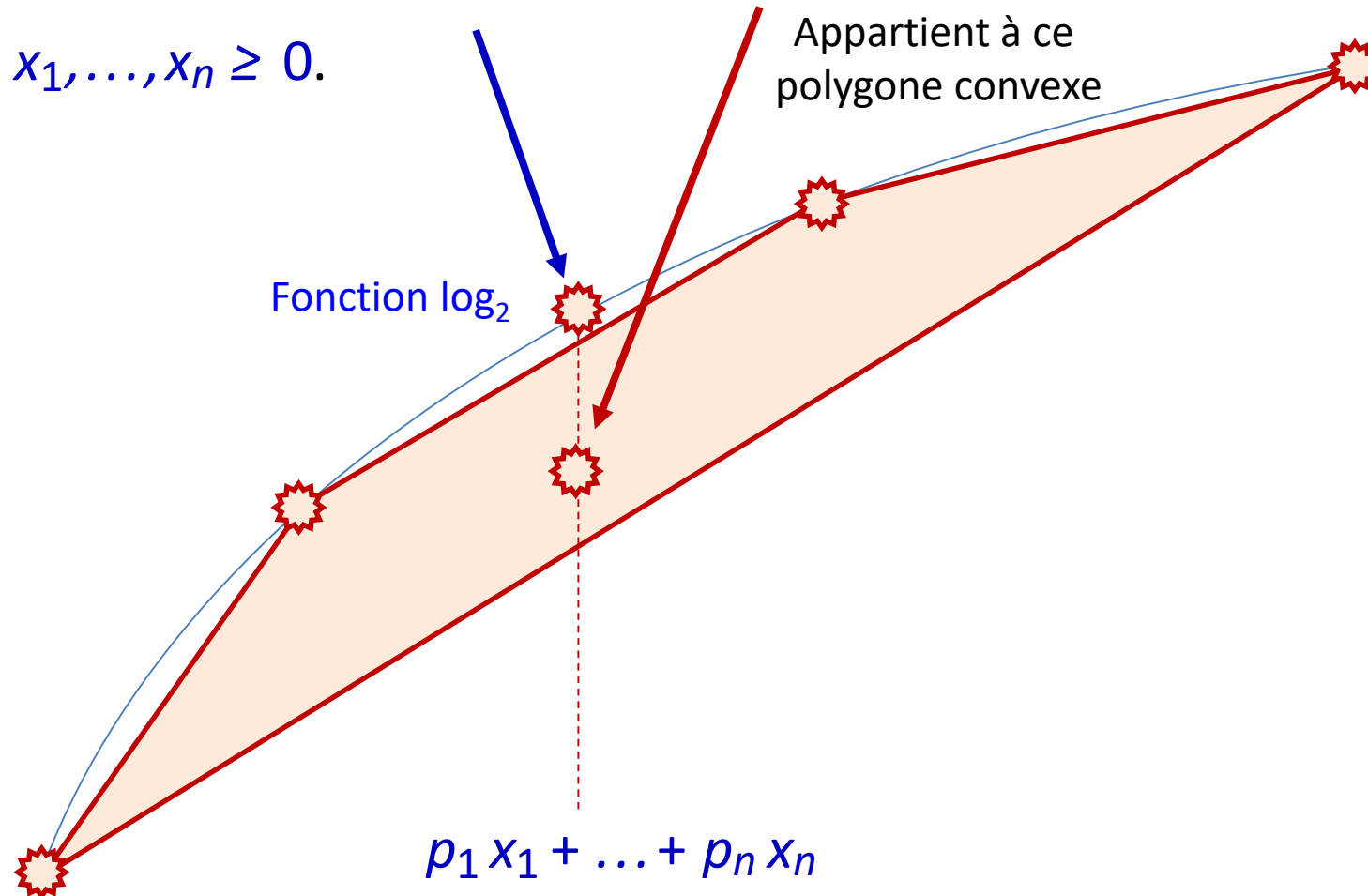


Vérifications: (4)

Plus généralement encore, si $0 \leq p_j \leq 1$ et $p_1 + \dots + p_n = 1$, alors

$$\log_2(p_1 x_1 + \dots + p_n x_n) \geq p_1 \log_2(x_1) + \dots + p_n \log_2(x_n)$$

pour tous $x_1, \dots, x_n \geq 0$.



Vérifications: (5)

Plus généralement encore, si $0 \leq p_j \leq 1$ et $p_1 + \dots + p_n = 1$, alors

$$\log_2(p_1 x_1 + \dots + p_n x_n) \geq p_1 \log_2(x_1) + \dots + p_n \log_2(x_n)$$

pour tous $x_1, \dots, x_n \geq 0$. (*inégalité de Jensen*)

En appliquant cette inégalité avec $x_j = 1/p_j$ on obtient finalement:

$$H(X) = p_1 \log_2(1/p_1) + \dots + p_n \log_2(1/p_n) \leq \log_2(p_1/p_1 + \dots + p_n/p_n)$$

$$H(X) \leq \log_2(1 + \dots + 1)$$

$$H(X) \leq \log_2(n)$$

Information, Calcul et Communication

Entropie

Mais à quoi tout cela peut-il bien servir?

Compression sans pertes

Comme on l'a vu dans l'introduction, la compression de données permet:

- de réduire l'espace-mémoire utilisé lors du stockage de données
- de réduire le temps de transmission et les problèmes de congestion lors de la transmission des données

Le principe de base derrière la **compression** de données est la **suppression de la redondance** contenue dans ces données; les lettres ou les mots qui reviennent souvent dans un message sont abrégés.

Lorsqu'on parle de compression **sans pertes**, on demande que le message d'origine puisse être récupéré dans son intégralité après le processus de compression.

L'opération doit être réversible

Compression sans pertes

Exemples d'algorithmes de compression sans pertes:

Langage SMS: “slt”, “tqt”, “lol”, etc; les mots qui reviennent souvent sont réduits à de courtes séquences de lettres.

Acronymes : EPFL, UNIL, MT, EL, etc...

Code Morse: “e” = . “a” = .- “s” = ... “t” = -

tandis que “x” = -.- “z” = -- ..

Le concept d'entropie permet de comprimer des données de manière *systematique et optimale*.

Compression sans pertes

Revenons sur la plage à Ibiza, et supposons que vous vouliez envoyer ce message par SMS à un ami:

I L F A I T B E A U A I B I Z A

Le but du jeu est maintenant de *minimiser le nombre de bits* nécessaires pour représenter ce message.

→ Il faut le coder plus efficacement que le code ASCII

Remarques importantes:

La personne qui reçoit votre message doit être capable de le **décoder**!

Avant l'envoi du message, l'expéditeur et le destinataire doivent se mettre d'accord sur un dictionnaire commun de motifs binaires pour chaque lettre:

Exemple: A='01', B='10', etc...

Compression sans pertes

IL FAIT BEAU A IBIZA

- **Solution N° 1:** Utiliser le code ASCII (cf. leçon 1.1): chaque lettre est représentée par 8 bits; $16 \times 8 = 128$ bits sont donc nécessaires pour représenter ce message.
- **Solution N° 2:** Remarquer que le message à représenter n'est composé que de 16 lettres: on n'a donc besoin que de 4 bits par lettre ($2^4 = 16$); et donc $16 \times 4 = 64$ bits au total.
- **Solution N° 3:** Remarquer que certaines lettres se répètent: il n'y a en fait que 9 lettres différentes. Et certaines sont plus fréquentes que d'autres...
Comment procéder?

Algorithme de Shannon-Fano

Pionnier de la théorie de l'information
 Robert Fano (1917-2016)
 Professeur au MIT



IL FAIT BEAU A IBIZA

Revenons à notre jeu de questions:

lettre	I	A	B	L	F	T	E	U	Z
nb d'apparitions	4	4	2	1	1	1	1	1	1
nb de questions	2	2	3	4	4	4	4	4	4

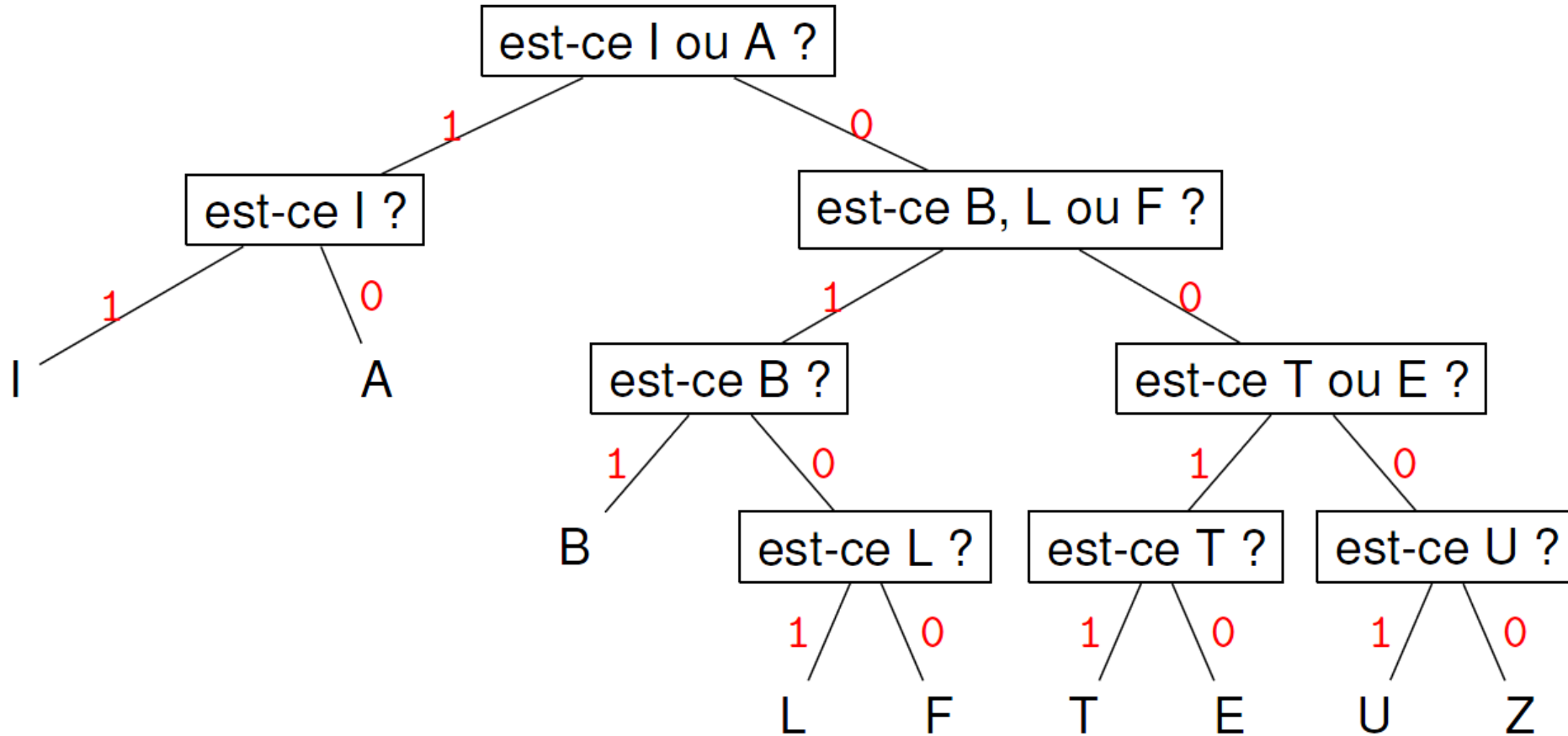
Pour attribuer une séquence de bits (un “mot de code”) à chaque lettre, nous suivons les deux règles ci-dessous:

Règle n⁰ 1: le nombre de bits attribués à chaque lettre est égal au nombre de questions nécessaires pour la deviner.

Règle n⁰ 2: les bits 0 ou 1 sont attribués en fonction des réponses obtenues aux questions. Plus précisément, le bit N^o j est égal à 1 ou 0 selon que la réponse à la j^e question était oui ou non.

IL FAIT BEAU A IBIZA

lettre	I	A	B	L	F	T	E	U	Z
nb d'apparitions	4	4	2	1	1	1	1	1	1



Algorithme de Shannon-Fano: cas particulier où p est de la forme 2^{-k}

IL FAIT BEAU A IBIZA

lettre	I	A	B	L	F	T	E	U	Z
probabilité p	2^{-2}	2^{-2}	2^{-3}	2^{-4}	2^{-4}	2^{-4}	2^{-4}	2^{-4}	2^{-4}
nb de questions = nb de bits	2	2	3	4	4	4	4	4	4

- Une lettre de probabilité 2^{-k} a seulement besoins de k questions

Selon la règle N°1 de l'algorithme de Shannon-Fano
une lettre de probabilité 2^{-k} est codée avec k bits

Algorithme de Shannon-Fano

lettre	I	A	B	L	F	T	E	U	Z
mot de code	11	10	011	0101	0100	0011	0010	0001	0000

Le message "IL FAIT BEAU A IBIZA" s'écrit ainsi:

11 0101 0100 10 ...

Observation N° 1:

Dans le dictionnaire, **aucun mot de code n'est le préfixe d'un autre.**
Donc le message reçu est parfaitement décodable, et ceci au fur et à mesure....

...sans attendre que la fin du message soit reçue

Performance de l'algorithme de Shannon-Fano

lettre	I	A	B	L	F	T	E	U	Z
mot de code	11	10	011	0101	0100	0011	0010	0001	0000

Observation n⁰ 2: Le nombre de bits utilisés est égal à:

$$8 \times 2 + 2 \times 3 + 6 \times 4 = 16 + 6 + 24 = 46$$

Par rapport à la représentation 'brute' qui nécessite 64 bits, on économise donc environ 25% de bits.

Vu que le message a 16 lettres, le nombre moyen de bits utilisés par lettre est égal à $46/16 = 2.875 = H(X) =$ entropie du message !

Nous montrerons la semaine prochaine que l'entropie est une *borne inférieure* au nombre moyen de bits par lettre dont on a besoin: aucun autre algorithme de codage ne permet de descendre plus bas.

C'est l'énoncé du premier théorème de Shannon.

Algorithme de Shannon-Fano

Avec cet algorithme, on obtient les performances suivantes:

Dans un texte:

15-25% de réduction avec les lettres, comme vu précédemment
jusqu'à 75% de réduction si on utilise les mots à la place des lettres!

Dans un fichier de données: 25-30% de réduction (p.ex:
fichiers archivés .zip pour l'envoi d'emails)

Conclusion temporaire

L'entropie mesure la “quantité d'information” présente dans un message.

Pour représenter un message par une séquence de bits, l'algorithme de Shannon-Fano utilise un nombre de bits par lettre égal à l'entropie du message (*dans un cas particulier seulement*).

La semaine prochaine:

algorithme de Shannon-Fano (bis) et algorithme de Huffman

analyse de performance – théorème de Shannon

compression avec pertes

Annexe: entropie - définitions plus générales

La formule de l'entropie reste valable même si les p_j ne sont pas estimés sur la séquence elle-même, mais proviennent d'une estimation sur un texte plus grand ou même de notre connaissance du procédé qui a généré un mot X.

Tout ce qui compte ce sont simplement les probabilités p_j de chaque lettre (peut importe d'où elles viennent).

L'entropie calculée dans ce cours est simplement celle du « jeu » consistant à tirer une lettre du « mot » X au hasard.

Pour de vraies séquences, en tant que telles, la définition de l'entropie est bien plus complexe et intègre toutes les dépendances entre sous-séquences de lettres, i.e. les probabilités de toutes les sous-séquences de n lettres. Mais ceci nécessite des outils mathématiques plus avancés (probabilités conditionnelles et limites).

Exercices

1. Considérons le mot:

D E S O R D R E

- a) Calculer l'entropie de ce mot.
- b) Trouver une représentation binaire de ce mot.
- c) Combien de bits sont-ils nécessaires pour représenter ce mot?

2. Considérons l'expression suivante (à nouveau, sans les espaces):

INFORMATION CALCUL ET COMMUNICATION

- a) Calculer l'entropie de cette séquence.
- b) Décrire la suite de questions optimales à poser pour deviner une lettre choisie au hasard dans cette séquence (attention, dans ce cas, toutes les probabilités ne sont pas des puissances inverses de 2).
- c) A l'aide de ces questions, trouver une représentation binaire de cette séquence. Combien de bits sont-ils nécessaires?