Théorie de l'information

Cours 6 - Introduction au codage source

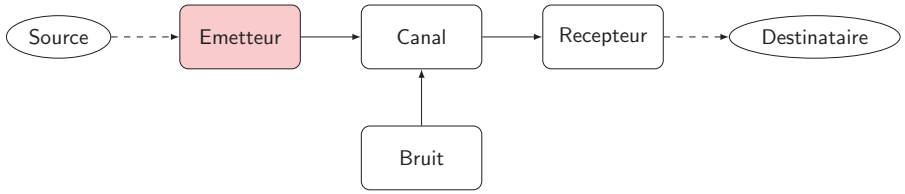
Laurent Oudre
laurent.oudre@univ-paris13.fr

Université Paris 13, Institut Galilée
Master Ingénierie et Innovations en Images et Réseaux - 1ère année
2017-2018

Sommaire

Sommaire

Emetteur



L'émetteur prend ce message numérique et réalise les étapes suivantes :

- **Codage source :** compression des données pour qu'elles prennent le moins de place possible. Cela revient à remplacer le message à envoyer par un message le plus court possible, souvent représenté sous forme d'une série de 0 et de 1.
- **Codage canal :** rajout de bits d'information supplémentaires dans le message pour permettre de corriger les éventuelles erreurs de transmission
- Transformer le message numérique en un signal physique (onde électromagnétique, signal électrique, etc...) qui puisse être transmis sur le canal de transmission

Travaux de Shannon

La théorie de l'information et les travaux de Shannon ont permis de répondre à deux questions fondamentales sur les sytèmes de communication :

- Codage source : étant donnée une source, à quel point peut-on compresser les données lors du codage, tout en faisant en sorte que le destinataire puisse toujours déchiffrer les messages que l'on envoie ?
  Premier théorème de Shannon
- Codage canal : étant donné un canal de communication bruité, jusqu'à quel débit d'information peut-on envoyer les données en conservant une probabilité d'erreur à la sortie qui soit limitée ?
  Deuxième théorème de Shannon

Rappel : cadre du cours

Dans le cadre de ce cours, on considérera uniquement :

- Des sources discrètes sans mémoire où les symboles envoyés successivement sont indépendants et identiquement distribués
- Des canaux discrets sans mémoire
- Un système de communication sera donc entièrement déterminé par :
  - Les alphabets $\mathcal{X}$ et $\mathcal{Y}$ des symboles d'entrée et de sortie
  - La loi de probabilité $p_X(x)$ de la source
  - La loi de probabilité conditionnelle $p_{Y|X}(y|x)$: matrice de transfert du canal

Codage source

Considérons une source discrète sans mémoire, modélisée par une variable aléatoire discrète X à valeurs dans un alphabet $\mathcal{X} = \{x_1, \dots, x_M\}$.

- $\mathcal{X}$ est appelé l'**alphabet source**
- On va définir un deuxième alphabet $\mathcal{C}$ appelé **alphabet code**, qui va nous servir à coder les messages envoyés par la source. Dans la majorité des cas (et dans la suite de ce cours), on va choisir $\mathcal{C} = \{0, 1\}$
- Chaque symbole de l'alphabet source $\mathcal{X}$ va être associé à un symbole ou une suite de symboles de l'alphabet de code $\mathcal{C}$, que l'on appelle **mot**
- Le but du codage source est, qu'une fois codés dans $\mathcal{C}$, les mots aient la longueur la plus petite possible. Il s'agit donc d'introduire la notion de **compression**.

Codage source

Codage d'une source alphanumérique

Considérons une source ayant comme alphabet $\mathcal{X} = \{A, B, C\}$ et un message à envoyer $AAAABAAAAC$. On considère deux codes différents ayant le même alphabet code $\mathcal{C} = \{0, 1\}$:

Code n1	Code n2
$A \longrightarrow 0$	$A \longrightarrow 11$
$B \longrightarrow 10$	$B \longrightarrow 1$
$C \longrightarrow 11$	$C \longrightarrow 00$

- Avec le code n1, le message codé est : 000010000011. Il contient 12 bits
- Avec le code n2, le message codé est : 1111111111111111100. Il contient 19 bits. Il y a également un problème supplémentaire : si on considère les deux premiers bits reçus 11, on ne peut pas savoir s'il s'agit du symbole A ou du groupe de symboles BB...
- Le symbole A est ici beaucoup plus fréquent que les deux autres : si l'on veut compresser correctement le message, on a intêret à lui associer un mot de longueur faible.

Principe du codage source

- Le but du codage source est, à partir d'une source X , de trouver une correspondance symbole/mot qui minimise le nombre de bits utilisés pour coder un message
- Pour cela, on va utiliser les probabilités d'apparition des symboles et faire en sorte que les mots associés aux symboles très fréquents soient de longueur faible.
- Néanmoins, ceci n'est pas suffisant, car nous verrons que pour que le message une fois codé puisse être décodé, on ne peut pas choisir n'importe quels mots.

Sommaire

Codage par bloc

Étant donnée une source ayant comme alphabet source $\mathcal{X} = \{x_1, \dots, x_M\}$

- Le principe du **codage par bloc** consiste à découper le message de la source en blocs de taille fixe k et à les coder indépendamment les uns des autres par une suite de taille variable de symboles de l'alphabet de code.
- Chaque bloc de taille k sera associé à un mot. Sachant qu'il y a M symboles dans l'alphabet source, cela fait M^k possibilités pour les blocs.
- Si $k = 1$, cela revient à associer un mot à chaque symbole de la source.

Codage par bloc

Codage d'une source alphanumérique

Considérons une source ayant comme alphabet $\mathcal{X} = \{A, B, C\}$ et un message à envoyer

$$ABACCABC$$

- On peut soit envoyer chaque symbole séparément. Dans ces cas là, on définira 3 mots, un pour chaque symbole.

$$A \quad B \quad A \quad C \quad C \quad A \quad B \quad C$$

- Soit on peut transmettre le message en faisant des paquets de 2 symboles. Dans ces cas là, on aura $3^2 = 9$ mots, un pour chaque groupe de symboles.

$$AB \quad AC \quad CA \quad BC$$

- C'est virtuellement comme si l'on considérait une source ayant non pas un alphabet à 3 symboles, mais à 9 symboles.
- Dans le deuxième cas, on dit que l'on travaille non pas sur la source X, mais sur son extension d'ordre 2, que l'on note $X^{[2]}$

Extension d'une source

Extension d'une source et codage par bloc

Étant donnée une source discrète X à valeurs dans $\mathcal{X} = \{x_1, \ldots, x_M\}$, on appelle **extension d'ordre** k ou **extension de degré** k de la source X et on note $X^{[k]}$ la source émettant des paquets de k symboles de la source X.

$$\underbrace{x^{(1)}x^{(2)}\ldots x^{(k)}}_{k \text{ symboles}} \quad \underbrace{x^{(k+1)}x^{(k+2)}\ldots x^{(2k)}}_{k \text{ symboles}} \quad \underbrace{x^{(2k+1)}x^{(2k+2)}\ldots x^{(3k)}}_{k \text{ symboles}} \quad \ldots$$

Extension d'une source

Codage d'une source alphanumérique

Considérons une source ayant comme alphabet $\mathcal{X} = \{A, B, C\}$ et un message à envoyer

$$ABAC$$

- Chaque symbole peut être associé à un mot :

Code n1
$A \longrightarrow 0$
$B \longrightarrow 10$
$C \longrightarrow 11$

  Dans ce cas le message codé sera : 010011

- Ou alors le codage se fait à partir d'un groupe de 2 symboles :

Code n3		
$AA \longrightarrow 0$	$BA \longrightarrow 01$	$CA \longrightarrow 011$
$AB \longrightarrow 010$	$BB \longrightarrow 10$	$CB \longrightarrow 11$
$AC \longrightarrow 111$	$BC \longrightarrow 100$	$CC \longrightarrow 1$

  Dans ce cas, le message codé sera : 010111

- Dans le premier cas, on travaille directement sur la source X, et dans le deuxième cas sur son extension d'ordre 2 notée $X^{[2]}$.

Code non singulier

- Un code en bloc est **non singulier** si les mots utilisés pour coder les symboles sont tous différents.
- Si la taille de l'alphabet source est M, et que l'on utilise un code en bloc de taille k, il faut donc définir M^k mots différents.

Codage d'une source alphanumérique (suite)

Les codes n1 et n2 présentés précédemment sont tous deux des codes non singuliers : les messages associés à tous les symboles sont bien différents.

Code en bloc déchiffrable

- Un code non singulier est **déchiffrable** si toute suite de mots ne correspond qu'un seul message.
- Cela signifie qu'un message codé sous forme de 0 et 1 ne peut se déchiffrer que d'une seule façon.

Codage d'une source alphanumérique (suite)

Code n1	Code n2
$A \longrightarrow 0$	$A \longrightarrow 11$
$B \longrightarrow 10$	$B \longrightarrow 1$
$C \longrightarrow 11$	$C \longrightarrow 00$

- n1 est bien déchiffrable car quelque soit le mot, il n'y a qu'une seule façon de le déchiffrer
- A l'inverse n2 n'est pas déchiffrable. Si l'on reçoit 11, on ne peut pas savoir si cela correspond à A ou à BB...

Code instantané

- Un code déchiffrable est **instantané** si aucun des mots n'est le préfixe d'un autre mot.
- On peut donc le déchiffrer de façon instantanée, sans utiliser de renseignements fournis par la suite du texte.

Codage d'une source alphanumérique (suite)

Code n1	Code n4
$A \longrightarrow 0$	$A \longrightarrow 0$
$B \longrightarrow 10$	$B \longrightarrow 01$
$C \longrightarrow 11$	$C \longrightarrow 011$

- n1 est instantané
- n4 n'est pas instantané (par exemple ici le mot associé à A est le début du mot associé à B), mais il est en revanche tout de même déchiffrable car le bit 0 sert d'indicateur de début de mot.

Construction d'un code instantané binaire

Supposons que l'on souhaite construire un code instantané binaire et que l'on a 5 symboles à coder $\{x_1, \dots, x_5\}$

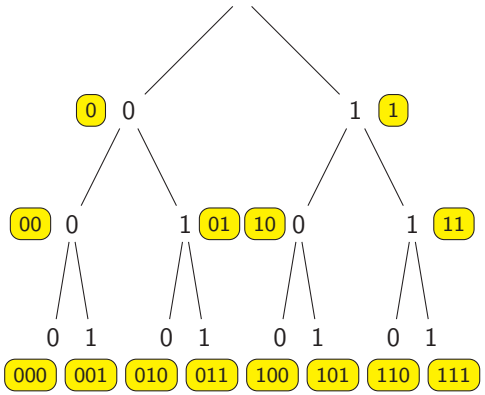
- Comme le code est instantané, il est également non singulier, donc il va falloir définir 5 mots différents.
- Si l'on souhaite utiliser le mot 0, on sait d'avance que tous les autres mots doivent commencer par 1.

$$x_1 \rightarrow 0 \qquad x_2, x_3, x_4, x_5 \rightarrow 1\dots$$

- Je ne peux pas choisir le mot 1 pour x_2 car sinon je n'ai plus aucune possibilité pour les autres
- Je peux par exemple choisir le mot 10 pour x_2, mais dans ces cas là tous les autres mots doivent commencer par 11

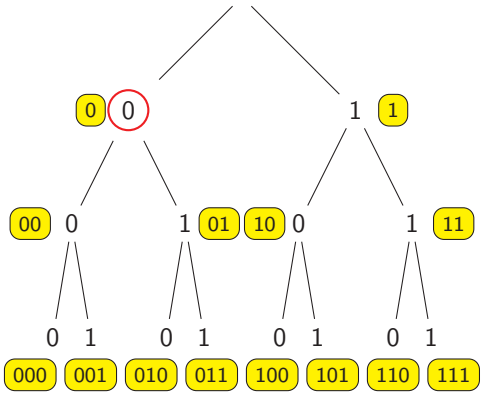
$$x_2 \rightarrow 10 \qquad x_3, x_4, x_5 \rightarrow 11\dots \qquad \text{etc...}$$

Construction d'un code instantané binaire



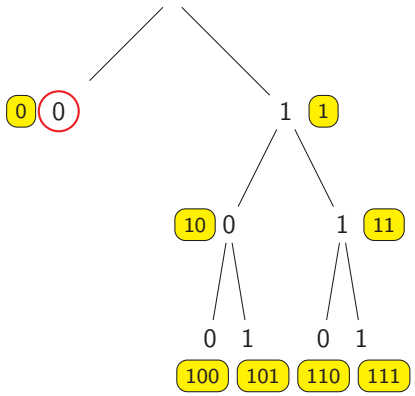
- On peut tout résumer en considérant un arbre binaire (ici à 3 niveaux).
- Les mots se lisent en commençant à la racine de l'arbre et en descendant à la feuille considérée
- A chaque fois que l'on choisit un mot, comme le code est instantané il faudra effacer les branches de niveaux inférieurs qui partent de cette feuille.

Construction d'un code instantané binaire



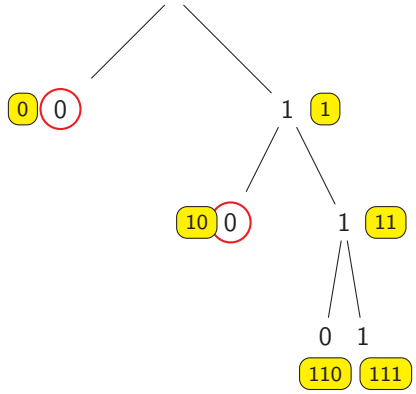
- Si je choisis le mot 0, tous les mots des niveaux inférieurs sont interdits pour la suite.

Construction d'un code instantané binaire



- Si je choisis le mot 0, tous les mots des niveaux inférieurs sont interdits pour la suite.

Construction d'un code instantané binaire



- Si pour le deuxième mot, je choisis 10, je dois enlever les branches correspondantes, etc...

Liens entre les codes

- Etant donné un code en bloc, on a les relations suivantes :

$$\text{instantané} \implies \text{déchiffrable} \implies \text{non singulier}$$

- Dans ce cours, nous allons nous focaliser sur les codes instantanés, qui permettent de pouvoir déchiffrer de façon unique le message, mais également de pouvoir le faire à la volée en temps réel.
- On verra dans la suite comment créer un code instantané permettant de compresser au mieux les données.
- Trouver un *bon* code instantané revient à supprimer les *bonnes* branches dans l'arbre binaire

Cadre du cours

- Dans la suite du cours on n'utilisera que des codes déchiffrables ou instantanés
- On ne traitera ici que le codage avec un alphabet de code $\mathcal{C} = \{0, 1\}$, c'est à dire que tous les mots seront des séries de 0 et de 1
- On ne traitera également que le codage à bloc de taille $k = 1$, c'est à dire que chaque symbole de l'alphabet source $\mathcal{X} = \{x_1, \ldots, x_M\}$ sera associé à un mot. On aura donc exactement M mots différents (car les codes déchiffrables et instantanés sont des codes non singuliers).

Sommaire

Longueur moyenne d'un code

Longueur moyenne d'un code

Étant donnée une source discrète sans mémoire, modélisée par une variable aléatoire discrète X à valeurs dans un alphabet $\mathcal{X} = \{x_1, \dots, x_M\}$, et un code déchiffrable binaire comportant M mots de longueur $\{l_1, \dots, l_M\}$, on appelle **longueur moyenne** du code la quantité :

$$\bar{L} = \sum_{i=1}^{M} P_X(X = x_i) l_i$$

- La longueur moyenne d'un code peut être vue comme l'espérance de la variable aléatoire L correspondant à la taille des mots.

Longueur moyenne d'un code

- Plus la longueur moyenne du code $\bar{L}$ est petite, plus on aura réussi à compresser les données
- Un des buts en codage source, est de construire des codes ayant la longueur moyenne la plus petite possible
- Ceci implique de choisir avec soin les mots à associer à chacun des symboles.
- Intuitivement, il faudra réserver les mots de petite taille aux symboles les plus probables si l'on veut minimiser la longueur moyenne

Longueur moyenne d'un code

Exemple

Considérons une source X ayant comme alphabet $\mathcal{X} = \{0, 1, 2, 3\}$

X	$p_X(x)$	Code 1	Code 2
$X = 0$	$\frac{1}{4}$	10	01
$X = 1$	$\frac{1}{8}$	110	00
$X = 2$	$\frac{1}{8}$	111	11
$X = 3$	$\frac{1}{2}$	0	11

- $\bar{L}_{code1} = \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 + \frac{1}{2} \times 1 = 1.75$ bits
- $\bar{L}_{code2} = \frac{1}{4} \times 2 + \frac{1}{8} \times 2 + \frac{1}{8} \times 2 + \frac{1}{2} \times 2 = 2$ bits
- Même s'il y a des mots plus longs dans le code 1, la longueur est moyenne est plus petite car on a affecté un mot très court (0) au symbole le plus probable.

Inégalité de Kraft

Inégalité de Kraft

Il existe un code binaire instantané contenant M mots de longueurs $\{l_1, \ldots, l_M\}$, si et seulement si :

$$\sum_{i=1}^{M} 2^{-l_i} \leq 1$$

- Attention, ce n'est pas parce qu'un code vérifie cette inégalité qu'il est instantané !
- En revanche, un code instantané doit vérifier cette propriété

Démonstration de l'inégalité de Kraft

Démontrons que c'est une condition nécessaire. On suppose donc que l'on considère un code binaire instantané.

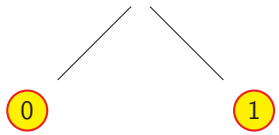
- On suppose qu'il contient M mots (il est instantané donc non singulier).
- Notons N_j le nombre de mots de longueur j du code, et m la longueur maximale des mots du code
- On a donc :

$$\sum_{j=1}^{m} N_j = M$$

- Nous allons, en reprenant une construction par arbre binaire, conjecturer du nombre maximal de mots de longueur j que l'on peut définir.

Démonstration de l'inégalité de Kraft

Cas $j = 1$. Quel est le nombre maximal de mots de longueur 1 que l'on peut définir ?

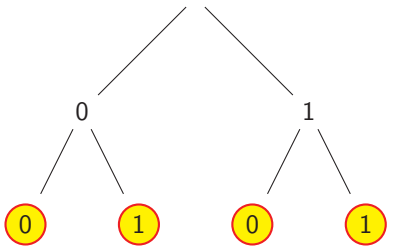


- Il y a au maximum deux mots de longueur 1 que l'on peut définir : 0 ou 1
- On a donc forcément :

$$N_1 \leq 2$$

Démonstration de l'inégalité de Kraft

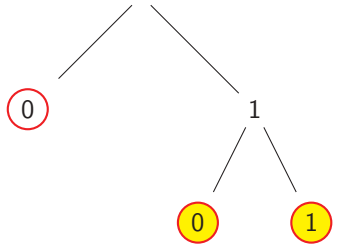
Cas $j = 2$. Quel est le nombre maximal de mots de longueur 2 que l'on peut définir ?



- Si $N_1 = 0$, tous les mots de longueur 2 sont potentiellement possibles
- Il y a au maximum quatres mots de longueur 2 que l'on peut définir : 00, 01, 10 et 11

Démonstration de l'inégalité de Kraft

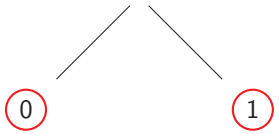
Cas $j = 2$. Quel est le nombre maximal de mots de longueur 2 que l'on peut définir ?



- En revanche si $N_1 = 1$, une moitié de l'arbre n'est plus accessible, et on ne peut donc définir que 2 mots de taille 2
- Dans l'exemple ci-contre, on a supposé que le mot 0 avait été choisi : dans ce cas on a uniquement deux mots de taille 2 possibles : 10 et 11

Démonstration de l'inégalité de Kraft

Cas $j = 2$. Quel est le nombre maximal de mots de longueur 2 que l'on peut définir ?



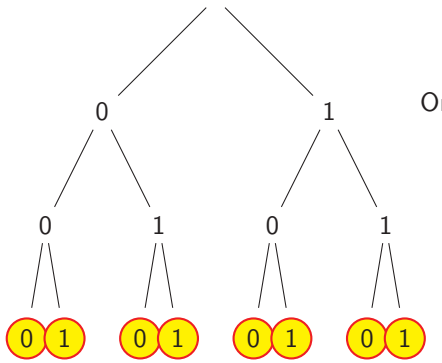
- Si $N_1 = 2$ il est impossible de définir un mot de longueur 2
- On peut résumer les trois étapes précédentes par la formule :

$$N_2 \leq 4 - 2N_1$$

$$N_2 \leq 2^2 - 2^1 N_1$$

Démonstration de l'inégalité de Kraft

Considérons le cas $j = 3$ Quel est le nombre maximal de mots de longueur 3 que l'on peut définir ?

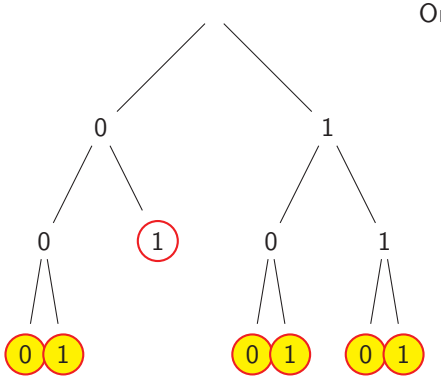


On suppose que $N_1 = 0$

- Si $N_2 = 0$, on a 8 mots de longueur 3 possibles : 000, 001, 010, 011, 100, 101, 110 et 111

Démonstration de l'inégalité de Kraft

Considérons le cas $j = 3$ Quel est le nombre maximal de mots de longueur 3 que l'on peut définir ?

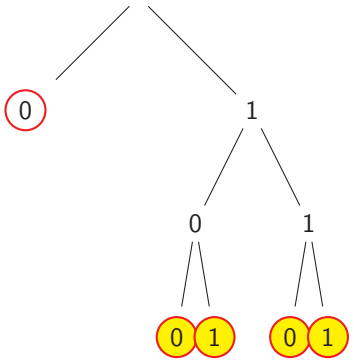


On suppose que $N_1 = 0$

- Si $N_2 = 1$, on a 2 mots en moins de longueur 3 possibles
- Dans l'exemple ci-contre, on a supposé que le mot 01 avait été choisi : dans ce cas les mots 010 et 011 ne sont plus possibles
- Dans le cas général, si N_2 est non nul, on enlève donc $2N_2$ mots de longueur 3

Démonstration de l'inégalité de Kraft

Considérons le cas $j = 3$ Quel est le nombre maximal de mots de longueur 3 que l'on peut définir ?



On suppose maintenant que N_1 est non nul

- Si $N_2 = 1$, on a 4 mots en moins de longueur 3 possibles
- Dans l'exemple ci-contre, on a supposé que le mot 0 avait été choisi : dans ce cas les mots 000, 001, 010 et 011 ne sont plus possibles
- Dans le cas général, si N_1 est non nul, on enlève donc $4N_1$ mots de longueur 3
- On peut résumer les étapes précédentes en :

$$N_3 \leq 2^3 - 2^2 N_1 - 2N_2$$

Démonstration de l'inégalité de Kraft

- En continuant ce raisonnement, on pourrait démontrer que :

$$N_m \leq 2^m - 2^{m-1} N_1 - 2^{m-2} N_2 - \ldots - 2N_{m-1}$$

- En divisant cette expression par 2^m, on a :

$$2^{-m} N_m + 2^{-(m-1)} N_{m-1} + \ldots + 2^{-2} N_2 + 2^{-1} N_1 \leq 1$$

- Parmi les longueurs $\{l_1, \ldots, l_M\}$, il y en a N_1 égales à 1, N_2 égales à 2, etc..., ce qui fait qu'on retrouve immédiatement la relation :

$$\sum_{i=1}^{M} 2^{-l_i} \leq 1$$

Démonstration de l'inégalité de Kraft

- La démonstration de l'autre inclusion est plus calculatoire et sera donc admise. Il s'agit de faire le raisonnement dans l'autre sens et de prouver qu'on peut dans ce cas construire un arbre binaire approprié.
- A noter que ce théorème existe dans une version portant sur les codes déchiffrables. On l'appelle dans ce cas l'**inégalité de Mac Millan**.

Propriétés d'un code binaire déchiffrable

Propriétés d'un code binaire déchiffrable

- Étant donnée une source discrète sans mémoire, modélisée par une variable aléatoire discrète X et un code binaire déchiffrable (ou instantané) de longueur moyenne $\bar{L}$, on a :

$$\bar{L} \geq H(X)$$

- On appelle **rendement** (ou efficacité) d'un code la quantité

$$\nu = \frac{H(X)}{\bar{L}} \qquad \text{On a : } \nu \in [0, 1]$$

- On appelle **redondance** d'un code la quantité

$$\rho = 1 - \nu \qquad \text{On a : } \rho \in [0, 1]$$

Démonstration : $\bar{L} \geq H(X)$

- Considérons un code binaire déchiffrable (ou instantané) de longueur moyenne $\bar{L}$. Montrons que $\bar{L} \geq H(X)$
- Notons
  - $p_i = P_X(X = x_i)$
  - $Q = \sum_{i=1}^{M} 2^{-l_i}$
  - $q_i = \frac{2^{-l_i}}{Q}$
- On a
  - $\sum_{i=1}^{M} q_i = 1$
  - $\bar{L} = \sum_{i=1}^{M} p_i l_i$
  - $Q \leq 1$ d'après Mac Millan (ou Kraft dans le cas instantané)

Démonstration : $\bar{L} \geq H(X)$

$$
\begin{aligned}
D_{KL}(p||q) \geq 0 &\iff \sum_{i=1}^{M} p_i \log_2\left(\frac{p_i}{q_i}\right) \geq 0 \\
&\iff \sum_{i=1}^{M} p_i \log_2(p_i) + \sum_{i=1}^{M} p_i \log_2\left(\frac{1}{q_i}\right) \geq 0 \\
&\iff \sum_{i=1}^{M} p_i \log_2\left(\frac{1}{q_i}\right) \geq H(X) \\
&\iff \sum_{i=1}^{M} p_i \log_2\left(\frac{Q}{2^{-l_i}}\right) \geq H(X) \\
&\iff \bar{L} + \log_2(Q) \geq H(X) \\
&\iff \bar{L} \geq H(X) \text{ car } \log_2(Q) \leq 0
\end{aligned}
$$

Démonstration : $\bar{L} \geq H(X)$

- Au passage, on peut remarquer qu'on a égalité si et seulement si $p_i = q_i$, c'est à dire

$$P_X(X = x_i) = 2^{-l_i} \iff l_i = -\log_2(P_X(X = x_i))$$

- Dans ce cas on a $Q = 1$, et donc $\bar{L} = H(X)$.
- Il faut dans ce cas là, que toutes les quantités l_i soient entières, donc que **toutes les probabilités d'apparition des symboles soient des puissances (négatives) de 2**.
- On dit dans ce cas que le code est **absolument optimum**, car on atteint la borne inférieure pour la longueur moyenne. Le rendement d'un code absolument optimum est égal à 1, et sa redondance à 0.

Démonstration : $\bar{L} \geq H(X)$

Exemple de code absolument optimum

On considère une source X à valeurs dans $\mathcal{X} = \{1, 2, 3\}$ dont les lois de probabilités sont données par le tableau suivant :

	$p_X(x)$	Code
1	$\frac{1}{2}$	1
2	$\frac{1}{4}$	01
3	$\frac{1}{4}$	00

- On a $H(X) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) = 1.5$ bits
- On a $\bar{L} = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2 = 1.5$ bits
- Toutes les probabilités d'apparition sont des puissances de 2, on peut donc trouver un code absolument optimal et l'on atteint la borne inférieure pour la longueur moyenne.

Sommaire

Premier théorème de Shannon

Premier théorème de Shannon ou Théorème du codage de source

Étant donnée une source discrète sans mémoire, modélisée par une variable aléatoire discrète X, il existe un code binaire déchiffrable (ou instantané) de longueur moyenne $\bar{L}$ tel que :

$$H(X) \leq \bar{L} < H(X) + 1$$

Démonstration

- Notons comme précédemment $p_i = P_X(X = x_i)$
- On a vu précédemment que si les probabilités d'apparition des symboles sont des puissances de 2, on peut atteindre la borne inférieure de la longueur minimale en prenant $l_i = -\log_2(p_i)$.
- Mais dans le cas général, on ne peut pas utiliser directement ceci car les l_i doivent être des entiers. Considérons les longueurs

$$l_i = \lceil -\log_2(p_i) \rceil$$

et montrons qu'on peut construire un code déchiffrable (ou instantané) ayant ces longueurs.
- On a :

$$l_i \geq -\log_2(p_i) \iff 2^{-l_i} \leq p_i$$

- Donc

$$\sum_{i=1}^{M} 2^{-l_i} \leq \sum_{i=1}^{M} p_i = 1$$

- D'après Mac Millan (ou Kraft), il existe donc un code déchiffrable (ou instantané) avec les longueurs $l_i = \lceil -\log_2(p_i) \rceil$

Démonstration

- Considérons donc un code déchiffrable (ou instantané) ayant les longueurs

$$l_i = \lceil -\log_2(p_i) \rceil$$

- On a :

$$-\log_2(p_i) \leq l_i < -\log_2(p_i) + 1$$

- Donc :

$$-\log_2(p_i)\, p_i \leq l_i p_i < -\log_2(p_i)\, p_i + p_i$$

- En sommant :

$$-\sum_{i=1}^{M} \log_2(p_i)\, p_i \leq \sum_{i=1}^{M} l_i p_i < -\sum_{i=1}^{M} \log_2(p_i)\, p_i + \sum_{i=1}^{M} p_i$$

- Et finalement :

$$H(X) \leq \bar{L} < H(X) + 1$$

Conséquences du théorème

- La démonstration du théorème nous a permis non seulement de prouver le résultat, mais également de construire un code vérifiant cette double inégalité.
- En effet, étant donnée une source à M symboles ayant comme probabilités d'apparition $p_1, \ldots, p_M$, on sait qu'en considérant les longueurs

$$l_i = \lceil -\log_2(p_i) \rceil$$

on sera capable de construire un code instantané dont la longueur moyenne sera proche de l'entropie de la source.

- Un code binaire déchiffrable (ou instantané) dont la longueur moyenne $\bar{L}$ vérifie

$$H(X) \leq \bar{L} < H(X) + 1$$

est appelé un code **compact**.

- Par construction, un code absolument optimum est un code compact.

Conséquences du théorème

Exemple de construction de code

On considère une source X à valeurs dans $\mathcal{X} = \{1, 2, 3\}$ dont les lois de probabilités sont données par le tableau suivant :

	$p_X(x)$
1	$\frac{1}{3}$
2	$\frac{1}{3}$
3	$\frac{1}{3}$

- Grâce à la démonstration précédente, on sait qu'en prenant $l_1 = l_2 = l_3 = \lceil -\log_2(\frac{1}{3}) \rceil = 2$ bits, on va pouvoir construire un code déchiffrable (ou instantané) qui sera compact.
- En prenant par exemple $1 \rightarrow 00$, $1 \rightarrow 01$ et $2 \rightarrow 10$, on a bien un code déchiffrable de longueur moyenne $\bar{L} = 2$ bits
- Or $H(X) = \log_2(3) = 1.6$ bits, donc on a bien : $H(X) \leq \bar{L} < H(X) + 1$

Deuxième version ?

- Le premier théorème de Shannon existe dans une deuxième version, qui donne des informations sur les propriétés asymptotiques des codes déchiffrables (ou instantanés)
- On a vu qu'on pouvait définir un code déchiffrable tel que :

$$H(X) \leq \bar{L} < H(X) + 1$$

- En réalité, on peut s'approcher aussi proche que l'on veut de l'entropie... à condition de ne pas coder directement la source, mais de coder son extension de degré k

Conséquences sur l'extension d'une source

- Considérons une source discrète sans mémoire X à valeurs dans $\mathcal{X} = \{x_1, \ldots, x_M\}$ et son extension d'ordre k $X^{[k]}$. D'après le premier théorème de Shannon, il existe un code déchiffrable (ou instantané) de longueur moyenne $\bar{L}$ tel que :

$$H\left(X^{[k]}\right) \leq \bar{L} < H\left(X^{[k]}\right) + 1$$

- Pour mieux comprendre ce que cela implique, nous allons calculer $H\left(X^{[k]}\right)$ et démontrer que :

$$\boxed{H\left(X^{[k]}\right) = k\, H(X)}$$

Démonstration de $H\left(X^{[k]}\right) = k\ H(X)$

- Nous allons démontrer cette propriété par récurrence.
- Etape 1 : $k = 1$
  Il est aisé de voir que cela est vrai pour $k = 1$: l'extension d'ordre 1 de la source X est exactement la source X !
- Etape 2 : Récurrence. Supposons que la propriété est vraie pour $k-1$, et montrons qu'elle est vraie pour k

Démonstration de $H\left(X^{[k]}\right) = k\ H(X)$

- Considérons une réalisation $x^{(1)}x^{(2)}\dots x^{(k)}$ de $X^{[k]}$. On peut la voir comme la réalisation simulatanée de $x^{(1)}x^{(2)}\dots x^{(k-1)}$ (qui correspond à une extension d'ordre $k-1$ de la source X) et de $x^{(k)}$ (extension d'ordre 1 de la source X).
- On peut donc dire que

$$H\left(X^{[k]}\right) = H\left(X^{[k-1]}, X\right)$$

- On a donc :

$$H\left(X^{[k]}\right) = H\left(X^{[k-1]}\right) + H(X) - I\left(X^{[k-1]}; X\right)$$

- Comme la source est sans mémoire, $X^{[k-1]}$ et X sont indépendantes, donc $I\left(X^{[k-1]}; X\right) = 0$
- Par récurrence, on a donc bien $H\left(X^{[k]}\right) = kH(X)$

Conséquences sur l'extension d'une source

- D'après la propriété démontrée précédemment, étant donnée une source X et un entier k, il existe donc un code déchiffrable (ou instantané) de longueur moyenne $\bar{L}$ tel que :

$$H\left(X^{[k]}\right) \leq \bar{L} < H\left(X^{[k]}\right) + 1$$

$$kH(X) \leq \bar{L} < kH(X) + 1$$

- Sauf que si l'on code une extension d'ordre k, la longueur moyenne $\bar{L}$ correspond à la longueur moyenne pour coder un bloc de k symboles. Donc en réalité, la longueur moyenne du code $\bar{L}_{sym}$ pour coder un seul symbole est

$$\bar{L}_{sym} = \frac{\bar{L}}{k}$$

- On a donc finalement

$$H(X) \leq \bar{L}_{sym} < H(X) + \frac{1}{k}$$

Premier théorème de Shannon (bis)

Premier théorème de Shannon ou Théorème du codage de source (bis)

Soit une source discrète sans mémoire, modélisée par une variable aléatoire discrète X. Pour tout entier $k > 0$, il existe un code binaire instantané (ou déchiffrable) de longueur moyenne pour coder un seul symbole $\bar{L}_{sym}$ tel que :

$$H(X) \leq \bar{L}_{sym} < H(X) + \frac{1}{k}$$

Conséquences

$$H(X) \leq \bar{L}_{sym} < H(X) + \frac{1}{k}$$

- Quand $k \to +\infty$ on a $\bar{L}_{sym} = H(X)$
- Considérons une source sans mémoire X. Le premier théorème de Shannon nous indique donc qu'il existe un code déchiffrable (ou instantané) de longueur moyenne par symbole $\bar{L}_{sym}$ aussi près que l'on veut de l'entropie $H(X)$, à condition de coder des blocs de taille k suffisamment grande.
- Plus on code des blocs de grande taille, plus on arrive à compresser les données

Sommaire

Quelques codes courants

- Nous allons maintenant étudier deux codes courants : Shannon-Fano et Huffman
- Ce ne sont pas nécessairement les codes réalisant la meilleure compression (ils sont relativement anciens), mais ils ont été largement utilisés (par exemple au début de la création du format ZIP)
- Le principe de ces deux codes est en gros le meme :
  - Etant donnés des symboles dont la probabilité d'apprition est **connue**, il s'agit d'associer de façon intelligente les mots aux symboles pour que la longueur moyenne soit la plus petite possible.
  - Chaque symbole de la source sera affecté à un mot différent de façon statique.
- Dans le cours de M2, d'autres codes plus efficaces seront présentés, qui seront basés sur des principes légèrement différents (approches par dictionnaires, utilisation de la redondance de la source, adaptation au type de donnée envoyée etc...)

Code de Shannon-Fano

- Ce code a été créé par Robert Fano, en utilisant le résultat du premier théorème de Shannon
- On considère une source à M symboles, dont on connait les probabilités d'apparition $p_1, \ldots, p_M$

Code de Shannon-Fano

Algorithme

1. On classe les probabilités d'apparition des symboles dans un tableau par ordre décroissant
2. On divise le tableau en deux groupes S_0 et S_1 pour que la somme des probabilités dans chaque groupe fasse environ la moitié de la somme totale des probabilités.
3. On code le groupe S_0 par un 0 et le groupe S_1 par un 1
4. On réitère les étapes 2 et 3 sur les sous groupes jusqu'à ce que l'on tombe sur un symbole unique

Code de Shannon-Fano : déroulé

On considère une source X à valeurs dans $\{x_1, x_2, x_3, x_4, x_5\}$, avec les probabilités d'apparition $\{0.1, 0.3, 0.2, 0.1, 0.3\}$

- **Etape 1 : Former le tableau**

x_2	0.3
x_5	0.3
x_3	0.2
x_1	0.1
x_4	0.1

Code de Shannon-Fano : déroulé

- **Etapes 2 & 3 : Faire les groupes et associer le bit 0 ou 1**

x_2	0.3
x_5	0.3
x_3	0.2
x_1	0.1
x_4	0.1

- En bleu : groupe S_0, en rouge : groupe S_1
- Aucun groupe ne contient qu'un symbole donc on continue

Code de Shannon-Fano : déroulé

- **Etapes 2 & 3 : Faire les groupes et associer le bit 0 ou 1 (suite)**

x_2	0.3	0.3
x_5	0.3	0.3
x_3	0.2	0.2
x_1	0.1	0.1
x_4	0.1	0.1

- En bleu : groupe S_0, en rouge : groupe S_1
- Il reste un groupe à séparer

Code de Shannon-Fano : déroulé

- **Etapes 2 & 3 : Faire les groupes et associer le bit 0 ou 1 (suite)**

x_2	0.3	0.3	
x_5	0.3	0.3	
x_3	0.2	0.2	
x_1	0.1	0.1	0.1
x_4	0.1	0.1	0.1

- En bleu : groupe S_0, en rouge : groupe S_1
- $x_1 \to 110$, $x_2 \to 00$, $x_3 \to 10$, $x_4 \to 111$, $x_5 \to 01$

Code de Shannon-Fano : déroulé

x_1	110
x_2	00
x_3	10
x_4	111
x_5	01

- On a $H(X) \approx 2.1710$ bits et $\bar{L} = 2.2$ bits
- Le code construit est donc un code compact car on a bien :

$$H(X) \leq \bar{L} < H(X) + 1$$

Code de Huffman

- Ce code a été créé en 1952, et a été extremement utilisé depuis dans de nombreuses applications (son, image, vidéo)
- Il est souvent utilisé en deuxième couche après un premier codage source dédié au type de média (cf cours M2)
- Si l'on suppose que les probabilités d'apparition des symboles sont connues et fixes, que l'on veut coder chaque symbole par un mot différent de façon statique, et que l'on veut avoir un code instantané, alors le code de Huffamn est celui qui minimise la longueur moyenne.
- Le résultat est toujours un code compact.
- Comme pour le code précedent, on considère une source à M symboles, dont on connait les probabilités d'apparition $p_1, \ldots, p_M$

Code de Huffman

Algorithme

1. On classe les probabilités d'apparition des symboles dans un tableau par ordre décroissant
2. On regroupe les deux symboles de probabilités les plus faibles pour en faire un nouveau symbole dont la probabilité est la somme des probabilités de ces deux symboles. On classe ce nouveau symbole parmi les autres, toujours par ordre décroissant de probabilité
3. On représente cette fusion sous forme d'un arbre. On itère cette opération et à chaque étape le nombre de symboles diminue et l'arbre se construit.
4. On affecte le bit 0 à toutes les branches de gauche et 1 à toutes les branches de droite (ou l'inverse). On lit les mots associés en partant de la racine et allant jusqu'à la feuille considérée.

Code de Huffman : déroulé

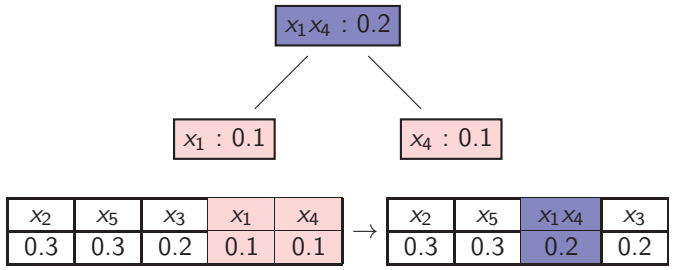
On considère une source X à valeurs dans $\{x_1, x_2, x_3, x_4, x_5\}$, avec les probabilités d'apparition $\{0.1, 0.3, 0.2, 0.1, 0.3\}$

Etape 1 : On classe les symboles par probabilité d'apparition décroissante

x_2	x_5	x_3	x_1	x_4
0.3	0.3	0.2	0.1	0.1

Code de Huffman : déroulé

Etape 2 & 3 : On fusionne les 2 symboles avec les probabilités les plus faibles. On construit l'arbre



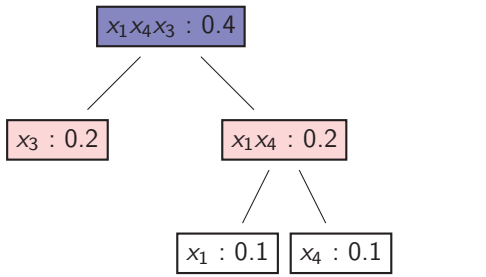
x_2	x_5	x_3	x_1	x_4
0.3	0.3	0.2	0.1	0.1

$\rightarrow$

x_2	x_5	x_1x_4	x_3
0.3	0.3	0.2	0.2

Code de Huffman : déroulé

Etape 2 & 3 : On fusionne les 2 symboles avec les probabilités les plus faibles. On construit l'arbre



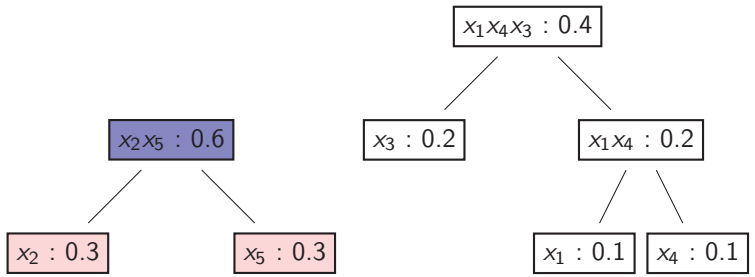
x_2	x_5	x_1x_4	x_3
0.3	0.3	0.2	0.2

$\rightarrow$

$x_1x_4x_3$	x_2	x_5
0.4	0.3	0.3

Code de Huffman : déroulé

Etape 2 & 3 : On fusionne les 2 symboles avec les probabilités les plus faibles. On construit l'arbre



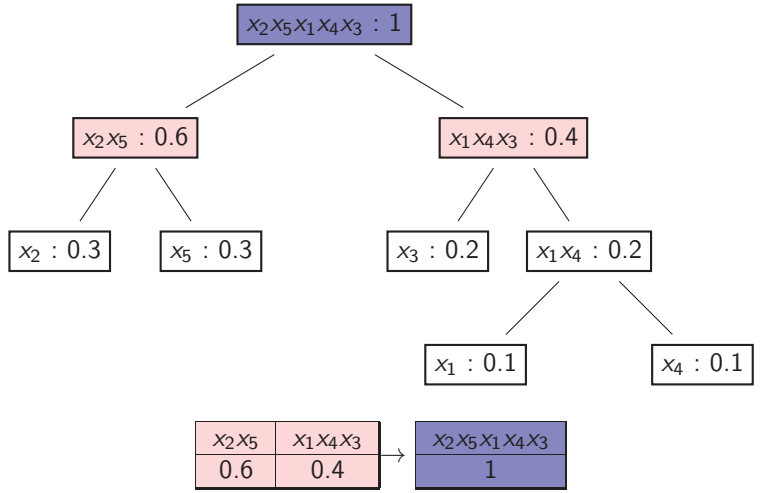
$x_1x_4x_3$	x_2	x_5
0.4	0.3	0.3

$\rightarrow$

x_2x_5	$x_1x_4x_3$
0.6	0.4

Code de Huffman : déroulé

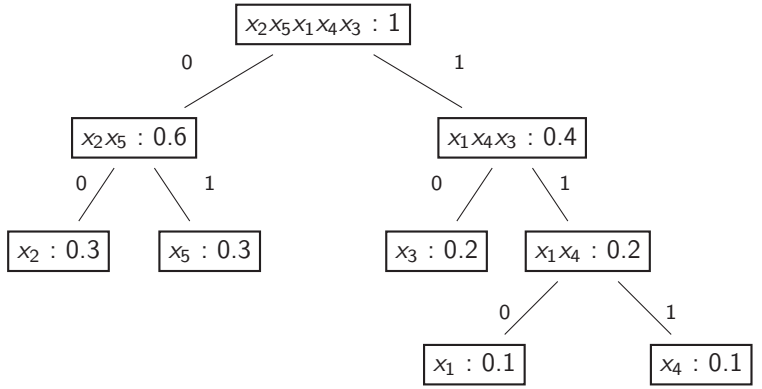
Etape 2 & 3 : On fusionne les 2 symboles avec les probabilités les plus faibles. On construit l'arbre



x_2x_5	$x_1x_4x_3$	→	$x_2x_5x_1x_4x_3$
0.6	0.4		1

Code de Huffman : déroulé

Etape 4 : On forme le code en affectant le bit 0 aux branches de gauche et le bit 1 aux branches de droite



x_1	x_2	x_3	x_4	x_5
110	00	10	111	01

Code de Huffman : déroulé

x_1	110
x_2	00
x_3	10
x_4	111
x_5	01

- On a $H(X) \approx 2.1710$ bits et $\bar{L} = 2.2$ bits
- Le code construit est donc un code compact car on a bien :

$$H(X) \leq \bar{L} < H(X) + 1$$

Bilan

- L'approche du codage de Shannon-Fano est descendante : l'algorithme part de l'ensemble des symboles et divise cet ensemble récursivement jusqu'à arriver à des parties ne contenant qu'un seul symbole. L'inconvénient de cette approche est que, lorsqu'il n'est pas possible de séparer un ensemble de symboles et deux sous-ensembles de probabilités à peu près égales (c'est-à-dire lorsque l'un des sous-ensembles est beaucoup plus probable que l'autre), les codes produits ne sont pas optimaux.
- Le codage de Huffman a une approche ascendante : l'algorithme part des symboles et regroupe ceux ayant la probabilités la plus faible, jusqu'à avoir regroupé tous les symboles. Cette approche permet d'obtenir systématiquement un code optimal au niveau du symbole, dans le pire cas de la même longueur que le code de Shannon-Fano équivalent, dans tous les autres cas plus court.