Chapitre **3**

Codages **E**ntropiques

1. Sources discrètes

- ***Source discrète sans mémoire :*** la probabilité d'apparition d'un symbole ne dépend pas des symboles précédents

$$p\,(x_n \setminus x_{n-1}, x_{n-2}, \ldots, x_1) = p\,(x_n) \qquad \textbf{(1)}$$

- ***Source discrète avec mémoire*** **:** les symboles fournis par ce type de source sont statistiquement dépendants. On utilise un processus stochastique pour sa modélisation.

$$p\,(x_n \setminus x_{n-1}, x_{n-2}, \ldots, x_1) = p\,(x_n \setminus x_{n-1}) \qquad \textbf{(2)}$$

- ***Source discrète stationnaire :*** la probabilité d'apparition des différents symboles est indépendante du temps.

$$p\,(x_n) = p\,(x_{n+k}) \qquad \textbf{(3)}$$

2. Objectifs de codage d'une source discrète

Une source discrète est codée pour trois raisons principales :

- Compresser les données, c'est-à-dire réduire la longueur moyenne des mots de code qui représentent les symboles d'une source. Ceci est possible par suppression, au maximum, de la redondance des messages.
- Le codage contribue à élever le degré d'immunité du message contre le bruit, par conséquent, à assurer une bonne qualité de transmission en présence de bruit. Ceci est possible par ajout de la redondance afin de pouvoir corriger les messages bruités.
- Un dernier aspect du codage est de garantir une certaine confidentialité en rendant le message non déchiffrable par des lecteurs non autorisés.

Dans la suite nous allons examiner le premier aspect du codage, à savoir coder par soucis de compression.

3. Codage de source

Soit une source discrète **X, *sans mémoire,*** à ***n*** valeurs possibles $\{x_1, x_2, \ldots, x_n\}$ dans un alphabet fini $A = \{a_1, a_2, \ldots, a_M\}$. Les statistiques de cette source sont caractérisées par une distribution de probabilité $p(x)$ sous la forme $\{p_1, p_2, \ldots, p_M\}$, où p_i est la probabilité d'occurrence du symbole $x_i \in A$.

3.1. Code

Un code de source **C** pour la variable aléatoire ***X*** est une fonction de *A (*l'ensemble des valeurs possibles de ***X*****)** vers D*, l'ensemble des chaînes de symboles d'un alphabet D-aire. $\boldsymbol{c}_i$ est le ***mot de code*** correspondant au symbole $x_i \in A$, et L_i est sa longueur. Le code est l'ensemble des mots de codes $\{c_1, \ldots, c_M\}$.

i. Code à longueur fixe

Un code à longueur fixe est tel que tous des mots de code possèdent la même longueur en bits, ***L***. Le taux de codage (coding rate) ***R***, qui représente le nombre moyen de bits par symbole de source, est égal à :

$$R = \log_2 M \ \textbf{(bits)} \qquad \textbf{(4)}$$

Si une source a pour cardinal ***M,*** il est possible de la coder avec un code de longueur fixe ***L*** tel que

$$\log_2(M) \leq L \leq 1 + \log_2(M) \qquad \textbf{(5)}$$

L'efficacité, ***E***, d'un code de longueur ***L*** est:

$$E = \frac{\mathbf{H}(X)}{L} \quad \text{et} \quad E \leq 1 \qquad \textbf{(6)}$$

Remarque : $E = 1$ ssi :

- $\mathbf{H}(X) = \log_2 M$, c'est-à-dire les lettres de la source sont équiprobables,
- $L = \log_2 M$, c'est-à-dire le cardinal de la source est une puissance de 2.

Exemple 1

1. $\mathbf{X} = \{x_1, x_2, \ldots, x_8\}$; $x_i \in A = \{0, 1, 2, 3\}$; $p(x) = ¼.$; $\boldsymbol{M} = 4$; $\mathbf{C} = \{00, 01, 10, 11\}$

$R = L = \log_2(4) = 2$ bits/symbole

2. $\mathbf{X} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, munie de la loi de probabilité uniforme, c.à.d $p(x) = 1/10$.

lettre	0	1	2	3	4	5	6	7	8	9
mot de code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

$\mathbf{H}(X) = \log_2(10) \approx 3.32 \ bits$; $E = \mathbf{H} / 4 \approx 0.83$; Ce code n'est pas optimal.

ii. Code à longueur variable

Un code à longueur variable est tel que tous les mots de code ne possèdent pas la même longueur, en bits. Un mot sera d'autant plus long que sa probabilité d'apparition sera petite. Le taux de codage, R, est égal à la longueur moyenne, $\overline{L}$, du code tel que :

$$R = \overline{L} = \sum_{i=1}^{M} p_i . L_i \ \textbf{(bits)} \qquad \textbf{(7)}$$

Un code est d'autant plus efficace en compression que R est petit.

L'efficacité, E, du code est définie par:

$$E = \frac{\mathbf{H}(X)}{\overline{L}} \qquad \textbf{(8)}$$

Exemple 2

1. $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$; $x_i \in A = \{0, 1, 2, 3\}$;

x_i	p_i	c_i	***R***	***H***	***E= H/R***
0	0.5	**0**	$R = \bar{L} = 0.5 \times 1$	$\mathbf{H}(X) = -0.5 \times \log_2(0.5)$	**1 Code optimal**
1	0.25	**10**	$+ 0.25 \times 2$	$- 0.25 \times \log_2(0.25)$	
2	0.125	**110**	$+ 0.125 \times 3$	$- 0.125 \times \log_2(0.125)$	
3	0.125	**111**	$+ 0.125 \times 3$	$- 0.125 \times \log_2(0.125)$	
			$= 1.75\ bits$	$= 1.75\ bits$	

2. Reprise de l'exemple 8 du chapitre 2 (Rappels de le théorie de l'information). Soit une source ***S*** qui produit deux symboles x_1 et $x_2 \in \{A,B\}$ avec les probabilités respectives de 4/5 et 1/5 à une cadence de 80 symboles par minute.

- Le codage le plus simple qu'on puisse imaginer est :

Symboles x_i *de* **S**	$p(x_i)$	Symboles c_i *du code*
A	0.8	0
B	0.2	1

La longueur moyenne des mots de code, exprimée en nombre de symboles {0,1}du code **C** par symbole $x_i \in \{A,B\}$de la source ***S,*** est

$$\bar{L} = \sum_{x_i \in \{A,B\}} p(x_i).L(x_i) = 0.8\times1 + 0.2\times1 = 1 \text{ bit}$$

$\mathbf{H}(S) = 0.72$ bit; $E = 0.72$

Une suite typique de 30 symboles de la source est codée ainsi:

A	B	A	A	A	A	B	A	A	A	A	A	B	A	B	A	A	A	A	A	A	A	A	B	A	B	A	A	A	A
0	1	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0

Si l'on considère un canal binaire sans perte qui transmet 2 symboles 0 et 1 à une cadence de 1 symbole par seconde. La capacité du canal est évidemment de 1bit/s. avec ce code, on génère 80 symboles/min, ce que le canal ne peut pas absorber.

- Ce code n'est pas optimal. Il est possible d'améliorer son efficacité en considérant non plus des symboles isolés mais les paires de symboles. Ceci revient à changer la source **S** en $\mathbf{S^2}$.

Symboles x_i *de* $\mathbf{S^2}$	$p(x_i)$	Symboles c_i *du code*
AA	0.64	0
AB	0.16	10
BA	0.16	110
BB	0.04	111

La longueur moyenne des mots de code, en nombre de symboles par paire de lettres de la source:

$$\bar{L} = \sum_{x_i \in \{AA,\, AB,\, BA\, BB\}} p(x_i).L(x_i) = 0.64\times1 + 0.16\times2 + 0.16\times3 + 0.04\times3 = 1.56$$

Soit 1.56 / 2 = 0.78 bit/lettre de la source

$\mathbf{H}(S^2) = 2\ \mathbf{H}(S) = 1.44$ bits; $E = 1.44 / 1.56 \approx 0.92$

La même suite de 30 symboles de la source est maintenant codée avec 24 bits:

AB	AA	AA	BA	AA	AA	BA	BA	AA	AA	AA	AB	AB	AA	AA
10	0	0	110	0	0	110	110	0	0	0	10	10	0	0

L'encodeur produit 0.7 x 80 = 62.4 symboles/min, ce qui est encore trop haut pour le canal.

- On peut améliorer davantage le code en changeant la source **S** en $\mathbf{S^3}$:

Symboles x_i *de* $\mathbf{S^3}$	$p(x_i)$	Symboles c_i *du code*
AAA	0.512	0
AAB	0.128	100
ABA	0.128	101
BAA	0.128	110
ABB	0.032	11100
BAB	0.032	11101
BBA	0.032	11110
BBB	0.008	11111

La longueur moyenne des mots de code est 2.184 bits par 3 lettres de la source, donc 0.728 bit par lettre.

$\mathbf{H}(S^3) = 3\ \mathbf{H}(S) = 2.16$ bits; $E = 2.16 / 2.184 \approx 0.99$

La même suite de 30 symboles de la source est maintenant codée avec 22 bits:

ABA	AAA	BAA	AAA	BAB	AAA	AAA	AAB	ABA	AAA
101	0	110	0	11101	0	0	100	101	0

L'encodeur produit 0.728 x 80 = 58.24 symboles/min, ce qui est suffisant pour le canal.

3.2. Décodabilité d'un code

i. Code non-singulier (non-ambigu)

Un code d'une source discrète est *non-singulier* lorsque deux symboles différents de la source correspondent à deux mots de code différents. Ainsi :

$$x_i \neq x_j \Rightarrow c_i \neq c_j$$

ii. Code uniquement décodable

C* est le code étendu de **C** tel que : $C^*(x_1, x_2, \ldots, x_n) = \mathbf{C}(x_1)\,\mathbf{C}(x_2)\ldots\mathbf{C}(x_n) = c_1 c_2 \ldots c_n$

On dit que le code **C** est *uniquement décodable* si son extension **C*** est non singulier.

De manière générale, on dit que le code d'une source discrète est *uniquement décodable* si et seulement si chaque séquence (de longueur finie) de mots de code ne correspond qu'à un seul message de la source.

Exemple 3

Considérons la source composée des trois symboles a, b et c. Ses messages peuvent être n'importe quelle séquence de ces symboles. On considère les codes suivants:

Symboles x_i *de la source*	Code 1	Code 2
a	1	1
b	00	00
c	11	10

- Le code 1 est non-ambigu, mais non uniquement décodable. Par exemple, une fois codés, il n'y a aucun moyen de distinguer le message «**aaaa**» du message «**cc**». En effet, tous deux sont codés «**1111**».
- Le code 2 est non-ambigu et uniquement décodable. Par exemple, la séquence «**10000**» se décode «**abb**» et la séquence «**1000**» se décode «**cb**».

3.3. Code préfixe

i. préfixe

On dit qu'une séquence ***z*** de longueur n ($n \geq 1$) est un *préfixe* d'une autre séquence ***z′*** si et seulement si les n premiers symboles de ***z′*** forment exactement la séquence ***z***. Par exemple, abba est un préfixe de abbabc.

ii. Code préfixe

Un code *préfixe* est un code pour lequel aucun mot de code n'est le préfixe (début) d'un autre mot de code. ***Il est clair que tout code préfixe est uniquement décodable***.

Cependant, tous les codes à décodage unique ne satisfont pas forcément la condition du préfixe. Voir le Code 2 de l'exemple 10.

iii. *Code instantané*

Un code est dit *instantané* si et seulement si chaque mot de code dans toute chaîne de mots de code peut être décodé dès que l'on a atteint sa fin (chaque symbole est décodé sans référence aux symboles suivants).

Un code est instantané si et seulement si aucun mot de code n'est le préfixe d'un autre mot de code.

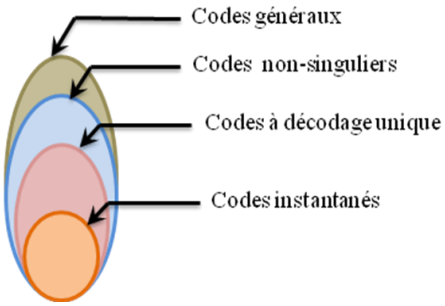

Fig.1 – Relations entre les différents types de codes.

Exemple 4

1. Considérons la source composée de quatre symboles a, b, c et d dont les codes respectifs sont 0, 10, 110 et 111.
   - Ce code est instantané et uniquement décodable.
   - Codage du message **adbccab** : **011110110110010**
   - Décodage de la séquence **1001101010** : **bacbb**

2. Ces codes sont-ils préfixes ? Uniquement décodables ? Instantanés ?

   a. Code1 : $c_1 = 00$, $c_2 = 10$, $c_3 = 01$, $c_4 = 11$: Préfixe, uniquement décodable et instantané

   b. Code2 : $c_1 = 0$, $c_2 = 1$, $c_3 = 01$: Non préfixe, non uniquement décodable et non instantané

   c. Code3 : $c_1 = 1$, $c_2 = 101$: Non préfixe, uniquement décodable et non instantané

iv. *Inégalité de Kraft*

Théorème 1: *Il existe un code instantané D-aire de N mots de code* $\{c_1, \ldots, c_N\}$ *et dont les longueurs des mots de code sont les entiers positifs* $L_1, L_2, \ldots, L_N$ *si et seulement si :*

$$\sum_{k=1}^{N} D^{-L_k} \leq 1 \qquad \textbf{(9)}$$

Lorsque l'égalité se réalise dans l'équation (20)*, le code instantané correspondant est complet.*

Remarque

Le théorème 1 nous apprend uniquement *quand* un code instantané peut exister, mais il ne répond absolument pas à la question ‘’est-ce qu'un code donné (avec telles longueurs de ses mots de code) est instantané ?’’

Exemple 5

1. Considérons la source composée des trois symboles a, b et c. On considère les codes binaires (D = 2) suivants: **C1** = {1, 00, 10} et **C2** = {0, 10, 11}

Le code **C1** n'est pas instantané. Toutefois la somme correspondante $\sum_{k=1}^{3} 2^{-L_k} = 2^{-1} + 2^{-2} + 2^{-2} = 1$. Le piège à éviter est que le théorème 1 ne nous dit

pas que ce code est instantané, **mais** qu'il existe un code instantané avec les *mêmes longueurs* de mots de code. Un tel code est par exemple le code **C2**.

2. Existe-t-il un code binaire instantané avec des longueurs de mots de code 1, deux fois 2, 3, et 4 ?

La réponse est non, étant donné que $\sum_{k=1}^{5} 2^{-L_k} = 2^{-1} + 2^{-2} + 2^{-2} + 2^{-3} + 2^{-4} = \frac{19}{16} \succ 1$

3.4. Codage et optimisation – Théorème fondamental (1er Théorème de Shannon)

On considère des sources discrètes sans mémoire.

Théorème 2: *Etant donné une source discrète A d'entropie H(A), et étant donné un alphabet D-aire pour le code, il est possible de coder chaque lettre de la source de manière instantanée et telle que la longueur moyenne* $\overline{L}$ *des mots de code satisfasse:*

$$\frac{\mathbf{H}(A)}{\log_2(D)} \leq \overline{L} \leq \frac{\mathbf{H}(A)}{\log_2(D)} + 1 \qquad \textbf{(10)}$$

Si D = 2 (alphabet binaire), on a:

$$\mathbf{H}(A) \leq \overline{L} \leq \mathbf{H}(A) + 1 \qquad \textbf{(11)}$$

Les relations (21) et (22) assignent des bornes pour la longueur moyenne des mots d'un code. Dans le cas d'un alphabet D-aire, nous aurons donc certainement besoin d'au moins **H**(*A*)/ $\log_2$D symboles de code en moyenne pour spécifier *A*.

Ces inégalités sont à l'origine du terme *codage entropique*, pour le codage sans perte uniquement décodable.

La double inégalité dans (22) donne lieu à une interprétation du théorème de Shannon qui stipule qu'il est possible de coder sans dégradation une source d'information d'entropie **H** bits en utilisant **H** + ε bits par échantillon où ε est un nombre arbitrairement petit. D'après ce théorème, l'entropie d'une source constitue la plus petite longueur moyenne des mots d'un code binaire. Plus l'entropie est petite, plus la longueur moyenne du code optimal est petite et plus la possibilité de compression est grande.

Exemple 6

Considérons une source *X* composée des symboles {0, 1, 2, 3} aux probabilités respectives {0.5, 0.25, 0.125, 0.125}, et dont les codes respectifs sont {**0, 10, 110, 111**}. Ce code est optimal, car:

$$\mathbf{H}(X) = \overline{L} = 1.75 \text{ bits / symbole}$$

4. Codage entropique

Un codage entropique est basé sur la notion de redondanc due à la non uniformité de distribution de probabilités, car certains échantillons apparaissent plus fréquemment que d'autres. Dans ce cas, un code à longueur variable est plus approprié en ce sens qu'on assigne les mots de code les plus courts aux symboles les plus fréquents. L'implémentation d'un tel code nécessite la connaissance préalable de la distribution des probabilités de l'ensemble des symboles émis par la source.

4.1. Code de Shannon-Fano

C'est un code binaire à longueur variable qui est effectué conformément aux étapes suivantes:

1. Arrangement des symboles émis par la source dans l'ordre décroissant de leurs probabilités d'occurrece.
2. Division des symboles en deux groupes de telle sorte que les sommes des probabilités des deux groupes soient aussi voisines que possible. Un bit **0** ou **1** est alors arbitrairement assigné à chaque groupe.
3. Division, à nouveau, de chaque ensemble en deux sous-ensembles, séparés avec **0** et **1**.
4. Itération sur les sous-ensebles résultants jsqu'à obtention de deux probabilités globales.
5. Finalement, les mots de code appropriés sont obtenus en joignant les bits associés à chaque probabilité en partant de la gauche vers la droite.

Exemple 7

Considérons une source avec 8 symboles telle que :

x_i	$p(x_i)$	*Processus de réduction*				c_i
x_1	0.4	0	0			**00**
x_2	0.18		1			**01**
x_3	0.1	1	0	0		**100**
x_4	0.1			1		**101**
x_5	0.07		1	0	0	**1100**
x_6	0.06				1	**1101**
x_7	0.05			1	0	**1110**
x_8	0.04				1	**1111**

Exemple de codage de Shanon-Fano

H(X) = 2.55 bits /symbole; $\overline{L}$ = 2.64 bits / symbole; L'efficacité du codage est E = 2.55 / 2.64 ≈ 96.6%

4.2. Code de Huffman

Un code est optimal si la longueur moyenne des mots de code est minimale et si le décodage est unique. Le code optimal le plus connu est celui de huffman. Le procédé est le suivant:

1. Arranger les symboles émis par la source dans l'ordre décroissant de leurs probabilités d'occurrece.
2. Isoler les deux symboles les moins probables et les distinguer avec 0 et 1.
3. Réunir ces deux symboles en un seul nouveau, en additionnant leurs probabilités, que l'on place avant le symbole ayant une probabilité moins grande. Dans le cas où la nouvelle probabilité est égale à l'une des probabilités restantes, l'approche de Huffman compte deux variantes selon que cette somme est placée à la position la plus haute ou la plus basse.
4. Le processus est répété jusqu'à ce que deux symboles aient la somme de leurs probabilités égale à l'unité.

Remarque : On peut également construire le code de Huffman D-aire.

Propriétés

1. Le code de Huffman est optimal.
2. Les deux plus longs mots de code ont la même longueur.
3. Les deux plus longs mots de code ne diffèrent que par leur dernier bit et correspondent aux deux symboles les moins probables.

Exemple 8

Considérons la même source de l'exemple 13. Avec le code de Huffman on obtient :

$\overline{L}$ = 2.61 bits / symbole; L'efficacité du codage est E = 2.55 / 2.61 ≈ 97.8%

x_i	$p(x_i)$	Processus de réduction						c_i
x_1	0.4	0.4	0.4	0.4	0.4	0.4	0.6 0	1
x_2	0.18	0.18	0.18	0.19	0.23	0.37 0	0.4 1	001
x_3	0.1	0.1	0.13	0.18	0.19 0	0.23 1		011
x_4	0.1	0.1	0.1	0.13 0	0.18 1			0000
x_5	0.07	0.09	0.1 0	0.1 1				0100
x_6	0.06	0.07 0	0.09 1					0101
x_7	0.05 0	0.06 1						00010
x_8	0.04 1							00011

Exemple de codage de Huffman

5. Mesures des performances d'une compression

Pour évaluer correctement les performances d'un algorithme de codage donné, on utilise souvent trois critères, à savoir :

- Le taux de compression,
- La distorsion,
- Le temps de calcul de l'algorithme de compression et de décompression.

5.1. Taux de compression

Le taux de compression de bits, noté T_C, est défini par :

$$T_C = \frac{Nombre\ de\ bits\ avant\ compression}{Nombre\ de\ bits\ après\ compression} \qquad (12)$$

5.2. Distorsion du signal

Dans le cas d'une compression irréversible, il devient nécessaire de mesurer le degré de distorsion introduite dans le signal reconstruit. Plusieurs mesures peuvent être utilisées.

L'erreur quadratique moyenne, MSE (en anglais Mean Square Error) entre le signal original X et le signal reconstruit après codage, Y.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left[X(i)-Y(i)\right]^2 \qquad \textbf{(13)}$$

5.3. Temps de traitement

Le temps nécessaire à la machine pour exécuter le programme de codage et décodage constitue une contrainte limitative dans un système de transmission à l'inverse du stockage et de l'archivage.