Chapitre 3

Codage de sources discrètes

3.1. Introduction

On s'intéresse au codage d'une source discrète sans mémoire en une séquence binaire. Le décodage devra permettre de retrouver la séquence de lettres émises par la source à partir de la séquence codée binaire.

Nous verrons que le nombre minimal moyen de symboles binaires par lettre est égal à l'entropie de la source.

Un exemple célèbre et désuet de tel codage est le code Morse. En Morse la lettre e, très fréquente, est représentée par le mot «·», tandis que la lettre q, moins fréquente est représentée par un mot plus long «··−−». De tels codes sont dits codes à longueur variable. On les appelle aussi plus simplement code.

3.2. Les propriétés des codes

Soit une source discrète dont l'alphabet est $X = \{a_1,...,a_K\}$ et dont la loi de probabilité $P(a_1),...,P(a_k)$ est donnée.

3.2.1. Définitions

Un ***codage*** d'une source discrète est une procédure qui associe à chaque séquence finie de lettres de la source une séquence binaire finie.

Un codage est donc une application de X* dans {0,1}*, sachant X* représente l'ensemble de mots de l'alphabet X.

Un ***code*** d'une source discrète est une procédure qui associe, à chaque lettre de la source, une séquence binaire appelée mot de code.

Un code est donc une application de X dans $\{0,1\}^*$, qui à toute lettre a_k de X associe un mot de code m_k.

3.2.2. Efficacité d'un code

L'efficacité d'un codage d'une source X peut alors être définie comme pour un code par:

$$E = \frac{H(x)}{\bar{n}}$$

Où n est le nombre moyen de symboles binaires utilisés par lettre de la source, défini par:

$$\bar{n} = \sum_{k=1}^{K} n_k p(a_k)$$

Où n_k est la longueur du mot de code associé à a_k

2.2.3. Propriétés des codes

a. Régularité: Un code est dit régulier si deux lettres distinctes sont codées à l'aide de deux mots de code distincts.

b. Déchiffrabilité : Un code régulier est dit déchiffrable (ou à décodage unique) si pour toute suite m^1, …,m^n de mots de code il est possible de distinguer les mots m^i sans ambiguïté, et donc de retrouver les symboles s^i correspondants. Il existe différentes façons d'assurer cette propriété:

b. 1. Code à longueur fixe :

Un code de longueur fixe est un code dont tous les mots de code ont la même longueur. Si cette longueur est n, nous dirons que le code est de longueur n.

Proposition: Soit une source X dont l'alphabet a pour cardinal K. Il existe un code régulier de X de longueur n telle que:$\log_2 K \leq n < 1+\log_2 K$.

De plus, il n'existe aucun code régulier de longueur $n < \log_2 K$

Exemple :

Soit une source dont l'alphabet est A={0,1,...,9} munie de la loi de probabilité uniforme. On code cette source par un code en longueur fixe de longueur 4.

Lettre	0	1	2	3	4	5	6	7	8	9
Mot	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

L'efficacité du code est : $H(A)/4=(\log_2 10)/4 \approx 0,83$

Proposition : Soit X une source de cardinal K, soit X^L la source dont l'alphabet est l'ensemble des L-uplets d'éléments de X. Il existe un code régulier de X^L de longueur N telle que:

$$log_2 K \leq \frac{N}{L} < \frac{1}{L} + log_2 K$$

b.2. Codes avec séparateurs

Il existe deux méthodes aisées pour obtenir un code permettant de séparer deux mots de code consécutifs:

1. Utiliser un code de longueur fixe n, auquel cas la séquence binaire reçue est découpée en blocs de n symboles binaires qui seront décodés séparément.

2. Utiliser, comme pour le Morse, un symbole supplémentaire entre deux mots. Dans un tel cas, tout se passe comme si l'alphabet de sortie du codeur était augmenté d'une unité. Ainsi le code Morse peut être considéré comme un code ternaire et non binaire.

b.3. Codes avec préfixes

Un code vérifie la condition du préfixe si aucun mot de code n'est le début d'un autre mot de code.

Proposition : Tout code préfixe est déchiffrable.

c. Codes Irréductibles

Un code est dit irréductible s'il vérifie la condition du préfixe.Tout code irréductible est déchiffrable. On peut aussi dire qu'un code préfixe est irréductible si les nœuds de son arbre ont 0 ou 2 fils.

d. Code instantané

On dit qu'un code est à *décodage instantané* s'il est possible de décoder les mots de code dès lors que tous les symboles qui en font partie ont été reçus.

Exemple : Soit la source qui émis les symboles {a1,a2,a3} codés par :

a1→ 1, a2→ 10, a3→ 100.

Les mots de codes peuvent être découpés en introduisant une séparation avant chaque "1" dans la séquence binaire, ce code est donc déchiffrable. Il n'est pourtant pas irréductible.

2. 2.4.Représentation des codes irréductibles par des arbres

Notion utilisées: les termes branche, nœud, feuille, racine, fils, père.

Les arbres que nous considérons ont les propriétés suivantes :

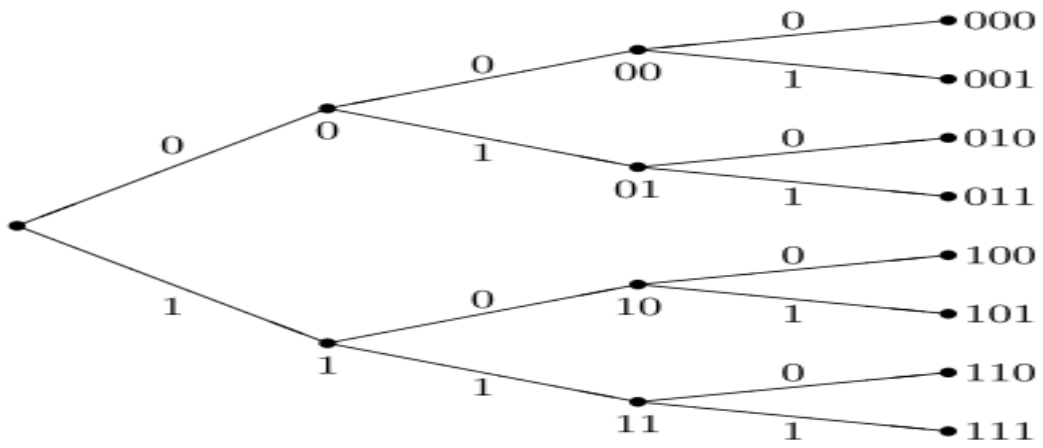
– Chaque branche a pour attribut un des symboles binaires 0 ou 1,

– Deux branches partant du même nœud ont des attributs différents,

– Chaque nœud a pour attribut la concaténation des attributs des branches reliant la racine à ce nœud.

– De plus, nous appellerons ordre d'un nœud ou d'une feuille le nombre de branches le séparant de la racine.

Exemple : Pour représenter l'ensemble des séquences binaire de longueur inférieure ou égale à 3 nous utilisons l'arbre :



Pour représenter un code régulier, nous utiliserons le plus petit arbre contenant tous les mots de code, appelé arbre du code.

Exemple : Considérons une source d'alphabet {a1,a2,a3,a4}, et les trois codages suivants:

	$P(a_i)$	Code 1	Code 2	Code 3
a1	1 /2	m1=00	m1=0	m1=00
a2	1 /4	m2=01	m2=10	m2=01
a3	1/8	m3=10	m3=110	M3=000
a4	1/8	m4=11	m4=111	M4=001

Les codes 1 et 2, sont irréductibles, les mots de code sont exactement les feuilles de l'arbre, alors que pour le code 3 n'est pas irréductible.

Pour le code 3 le mot de code m1 est à l'intérieur de l'arbre. Il existe une caractérisation simple des arbres des codes irréductibles :

Proposition: Un code est irréductible si et seulement si les feuilles de son arbre sont exactement ses mots de code.

Théorème de Kraft : Condition de McMillan: Soient $n_1,...,n_Q$ des longueurs de mots candidates pour coder une source Q-aire dans un alphabet. Alors l'inégalité de Kraft est une condition nécessaire et suffisante d'existence d'un code déchiffrable respectant ces longueurs de mots.

Application: Vérifier cette inégalité pour les codes 1 et 2 précédents

2.3. Algorithmes de codage statistiques

Le principe des algorithmes de codage statistique est d'utiliser les probabilités d'occurrences de chaque symbole dans une séquence de symboles émanant de la source.

2.3.1. Premier théorème de Shanon

Soit une source $X = \{a_1,...,a_K\}$ munie d'une loi de probabilité $P(a_1),...,P(a_K)$.

Théorème:

1. Pour toute source d'entropie H codée au moyen d'un code déchiffrable de longueur moyenne $\bar{n}$, on a: $\bar{n} \geq H(X)$

2. Pour toute source d'entropie H, il existe un code irréductible dont la longueur moyenne $\bar{n}$ est telle que: $H(X) \leq \bar{n} < H(X) + 1$

L'efficacité $E = \frac{H(X)}{\bar{n}}$ d'un code irréductible dont la longueur moyenne vérifie l'inégalité précédente sera telle que: $1 - \left(\frac{1}{H(X)+1}\right) < E \leq 1$

Premier théorème de Shannon: Pour toute source discrète sans mémoire, il existe un codage déchiffrable dont l'efficacité est arbitrairement proche de 1.

2.3.2. Algorithme de Shanon-Fano

L'algorithme comporte les étapes suivantes :

1) Les probabilités d'apparition de chaque symbole sont placées dans un tableau trié par ordre décroissant de probabilités.

2) Le tableau est coupé en deux groupes de symboles S0 et S1 dont la somme des probabilités de chaque groupe avoisine 0.5.

3) Le groupe S0 est codé par un "0" et S1 par un "1".

4) Si un groupe Si n'a qu'un seul élément, c'est une feuille terminale, sinon la procédure reprend récursivement à l'étape 2 sur le groupe Si.

Exemple : Soit l'alphabet source {A, B, C, D, E, F}, avec les probabilités :

Symbole S_i	A	B	C	D	E	F
$P(S_i)$	0,1	0,1	0,25	0,15	0,35	0,05

Après application de l'algorithme, on obtient les codes suivants :

Symbole S_i	A	B	C	D	E	F
Code	110	101	01	100	00	111

Décodage : Décoder la séquence 0110100

2.3.3.Code de Huffman

» Définition du code de Huffman : Le code de Huffmann d'une source X de cardinal K est un code irréductible qui se définit récursivement sur le cardinal de la source.

Méthode : L'algorithme opère sur une forêt. Une forêt est ici un ensemble d'arbres étiquetés complets: tout nœud interne (c'est-à-dire qui n'est pas une feuille) a deux fils non-vides.

La forêt initiale est formée d'un arbre à un nœud pour chaque lettre du langage-source, dont l'étiquette est la probabilité de cette lettre. La forêt finale est formée d'un unique arbre, qui est l'arbre de décodage du code.

L'algorithme est de type glouton: il choisit à chaque étape les deux arbres d'étiquettes minimales, soit x et y, et les remplace par l'arbre formé de x et y et ayant comme étiquette la somme de l'étiquette de x et de l'étiquette de y.

Exemple : Soit l'alphabet source S = {A, B, C, D, E, F}, avec les probabilités

Symbole S_i	**A**	**E**	**S**	**T**	**U**	**Y**
$P(S_i)$	21%	48%	12%	8%	6%	5%

Le code d'une lettre est alors déterminé en suivant le chemin depuis la racine de l'arbre jusqu'à la feuille associée à cette lettre en concaténant successivement un 0 ou un 1 selon que la branche suivie est à gauche ou à droite. On obtient ainsi les codes suivants :

Symbole S_i	A	E	S	T	U	Y
Code	01	1	001	0001	00001	00000

La langueur moyenne (L ou $\overline{n}$) = 2 ,13 bits/symbole.

H(S) = 2,10 bits/symbole

L'efficacité E = 2,10 /2,13 = 98,5%.

Définition: Un code déchiffrable de longueur moyenne $\overline{n}$ d'une source est dit *optimal* s'il n'existe aucun code déchiffrable de cette source dont la longueur moyenne des mots de code est strictement inférieure à $\overline{n}$.

Proposition: Le code de Huffman d'une source est optimal.

Démonstration :

Lemme : Pour toute source, il existe un code optimal instantané (de longueur moyenne minimale) qui satisfait les propriétés suivantes:

1. Si $p(S_j) > p(S_k)$ alors $n_j \leq n_k$,
2. Les deux mots les plus longs ont la même longueur,
3. Les deux mots les plus longs ne diffèrent que d'un bit, et correspondent aux deux mots les moins probables,

2.4. Algorithmes de codage arithmétiques

Contrairement aux deux algorithmes précédents, le code est associé à la séquence et non chaque symbole pris individuellement.

2.4.1. Codage de Lemplel-Ziv

Dans bien des cas pratiques, on ne connaît pas exactement la distribution de la source X. Dès lors, il est impossible de construire un code de Huffman. Sans connaître la distribution P_X, on peut néanmoins connaître son entropie H(X). On s'est alors posé la question de construire des codes de source "universels", susceptibles de coder toutes les sources dont l'entropie ne dépasse pas une borne fixée R en utilisant R bits par valeur à transmettre.

Une autre technique de codage universel a été développée pour les sources sur lesquelles on ne dispose d'aucune information. Cette technique, due à Lempel et Ziv (76), cumule de nombreux avantages :

• Les algorithmes de codage et de décodage sont très simples,

• Elle ne demande aucune connaissance sur la source,

• Elle est asymptotiquement optimale,

• Elle fonctionne même avec des sources à mémoire.

Par souci de simplicité, on suppose que X est une source binaire, éventuellement avec mémoire, à coder sur un alphabet binaire. L'algorithme procède en deux passes, sur une suite finie de bits x1, . . . ,xn. La première passe décompose le train de bits en segments de sorte que chaque nouveau segment n'ait pas été rencontré précédemment.

Sur la suite : 1011010100010 . . .cela donne la segmentation: 1|0|11|01|010|00|10| . . .

Il vient que chaque segment est composé d'un préfixe, correspondant à un segment déjà rencontré, plus un bit. A la fin de la première passe, on compte alors le nombre de segments, noté c(n).La deuxième passe procède au codage proprement dit. Elle numérote les segments par ordre d'apparition, ce qui demande $\lceil \log_2(c(n)) \rceil$ bits par segment, et représente chaque segment s à l'aide d'une paire (a, b) où a est le numéro du préfixe de s, et b le bit qui a été rajouté (le suffixe). Sur l'exemple précédent, on a 7 segments différents, que l'on va numéroter et coder sur 3 bits. La segmentation devient, sous forme codée :

(000, 1) (000, 0) (001, 1) (010, 1) (100, 0) (010,0) (001, 0) . . .

La longueur de la séquence codée est donc $c(n)[1+\lceil \log_2 c(n) \rceil]$. Le décodage se fait par le procédé inverse (il faut néanmoins connaître c(n)). Sur l'exemple ci-dessus, la séquence codée est plus longue que la séquence d'entrée du codeur... Mais il faut voir que pour des suites d'entrée longues, les segments que l'on identifie sont eux mêmes de plus en plus longs, et l'on y gagne beaucoup à les décrire sous la forme compacte (a, b).