Bases de théorie de l'information

Contents

Part I

Théorie de l'information

Introduction

Initiée par Claude Shannon en 1948, la théorie de l'information est une modélisation mathématique, essentiellement probabiliste, des problèmes liés à la mesure de la quantité d'information contenue dans un message, au stockage efficace et fiable d'un message (compression et protection).

Depuis son origine, la théorie de l'information est liée à celle de la communication. Quoique ce champ applicatif soit toujours parfaitement actuel, les outils et méthodes de la théorie de l'information trouvent leur utilité dans de nombreux autres domaines tels que l'estimation ou la finance.

Ce chapitre a pour objectif de fournir les notions théoriques de base pour la mesure quantitative de l'information (qu'est-ce qu'un bit d'information ?), sa représentation, son stockage, sa transmission, sa protection et sa dissimulation. Application à la compression sans perte, au codage correcteur, à la sécurité des transmissions et des contenus (stéganographie).

Il aborde les aspects suivants :

- Mesurer l'information. Incertitude et information. Entropies. L'entropie de Shannon et ses propriétés. Entropie de lois composées et transfert d'information.
- Structure d'une chaîne de communication. Sources d'information et compression. Canaux, capacité et codage de canal.
- Codage des sources discrètes. Equipartition asymptotique, notion de suite typique. Codes optimaux théoriques. Construction effective de codes optimaux.
- Transmettre et stocker l'information. Canal discret (canal binaire symétrique). Codage de canal et second théorème de Shannon.
- Codes détecteurs et correcteurs d'erreurs. Répétition et second théorème de Shannon. Codes détecteurs d'erreur. Codes correcteurs d'erreur. Codes en blocs linéaires. Distance de Hamming et distance Euclidienne. Décodage au sens du maximum de vraisemblance. Codes convolutifs et algorithme de Viterbi. Diagramme d'état, treillis et algorithme de Viterbi.
- Numérisation : échantillonnage et quantification.

Des domaines d'application seront évoqués pour illustrer l'intérêt des concepts abordés, citons :

- Stéganographie, tatouage, fuite d'information et sécurité.
- Compression d'un message. L'exploitation des propriétés statistiques d'un message peut permettre sa compression. Celle-ci s'avère utile dans de nombreux cas : transmission sur un canal à débit limité, stockage sur un support de capacité limitée ...
- Codes correcteurs d'erreurs. L'ajout d'une certaine redondance dans un message permet au lecteur de déceler un certain nombre d'anomalies (détection d'erreur) et parfois même de les corriger. Les codes correcteurs sont couramment utilisés : lecteurs de disques compacts par exemple.
- Capacité. Exploitation au mieux des caractéristiques d'un canal de transmission. La théorie de l'information fournie des bornes supérieures pour le débit d'information maximum qu'il est possible de faire passer au travers d'un canal physique donné.

Chapter 1

Incertitude, Entropie et Information

1.1 Notations

Au sens de Shannon, la théorie de l'information repose de façon essentielle sur l'existence d'une mesure objective de la quantité d'information contenue dans un message aléatoire. Bien que cette approche du problème de la mesure quantitative de l'information n'englobe pas tous les aspects du problème, elle satisfait à certaines attentes intuitives.

On considère une variable aléatoire discrète X prenant ses valeurs dans l'alphabet $\mathcal{A}$.
Pour tout $x \in \mathcal{A}$, on note $p(x) = p_X(x) = Pr[X = x]$ la probabilité de l'éventualité x.
On note les états $x_1, \cdots, x_N$ et leurs probabilités $Pr[X = x_j] = p(x_j) = p_j$.
$\mathcal{P} = \{p(x)\}_{x \in \mathcal{A}}$ est la loi de la v.a. X.
Les $\{p(x)\}_{x \in \mathcal{A}}$ sont des probabilités *a priori* sur les réalisations possibles de la v.a. X.

1.2 Des variables plus aléatoires que d'autres.

Dans une épreuve aléatoire, chaque éventualité se produit avec une fréquence connue *a priori*. Si l'on cherche à deviner l'issue de l'épreuve, les chances de succès dans cette tentative varient selon la distribution de probabilité sur l'ensemble des états possibles. La meilleure prédiction du résultat d'un tirage, au sens du minimum de la probabilité d'erreur, est l'état le plus probable. Si la probabilité de cet état est $p_{\max}$, la probabilité d'erreur est $1 - p_{\max}$.

Prenons l'exemple d'une loi de Bernoulli avec $p_1 = p$ et $p_2 = 1 - p$. Pour $p = 1$ on prédit l'état 1 et la probabilité d'erreur est nulle. Il en est de même pour $p = 0$: on prédit l'état 2 et la probabilité d'erreur est nulle. Entre ces deux extrêmes pour lesquels le résultat de l'expérience aléatoire est certain, le résultat est d'autant plus incertain (la probabilité d'erreur d'autant plus grande) que p est proche de $1/2$.

1.3 Définition de l'entropie de Shannon.

Incertitude de chacun des états. Il est assez naturel de définir l'incertitude $i(x)$ liée à la réalisation de l'état x comme étant une fonction de sa probabilité *a priori*.

$$i(x) = F[p(x)]$$

On peut raisonnablement imposer à F les propriétés suivantes :

- $i(x)$ est une quantité positive.
- F est une fonction décroissante, c'est-à-dire que l'incertitude est d'autant plus élevée que la probabilité *a priori* d'apparition de x est faible.

- Si X et Y sont deux v.a. indépendantes, l'incertitude liée à la réalisation du couple d'états (x, y) est la somme de l'incertitude liée à x et de celle liée à y :

$$F\left[p(x,y)\right] = F\left[p(x)p(y)\right] = F\left[p(x)\right] + F\left[p(y)\right]$$

L'utilisation de ces conditions conduit facilement à $F(x) = -\alpha \log(x)$ avec $\alpha > 0$.

Incertitude moyenne de l'ensemble des états. Pour définir une incertitude de la v.a. elle-même, il est naturel d'évaluer la moyenne des incertitudes sur l'ensemble des états. On définit de cette façon une quantité, l'entropie, attachée à l'expérience aléatoire elle-même. L'entropie est associée à la v.a. X, ou de façon équivalente à sa loi $\mathcal{P}$, les notations $H(X)$ ou $H(\mathcal{P})$ sont employées :

$$H\left(X\right) = H(\mathcal{P}) = -\alpha \sum_{x \in \mathcal{A}} p(x) \log p(x)$$

4 On adopte la convention $0 \log_2 0 = 0$ (prolongement par continuité de $x \log_2 x$ en 0).

Exercice : construction axiomatique de l'entropie. Montrer qu'une suite de fonctions $H_n(p_1, \cdots, p_n)$ symétriques qui vérifient les 3 propriétés suivantes :

1. $H_2(\frac{1}{2}, \frac{1}{2}) = 1$,
2. $H_2(p, 1-p)$ est une fonction continue de p,
3. $H_n(p_1, \cdots, p_n) = H_{n-1}(p_1 + p_2, p_3, \cdots, p_n) + (p_1 + p_2) H_2\left(\frac{p_1}{p_1+p_2}, \frac{p_2}{p_1+p_2}\right)$

est de la forme $H_n(p_1, \cdots, p_n) = -\sum_{j=1}^{n} p_j \log_2 p_j$

Cas de la loi de Bernoulli. L'expérience aléatoire la plus simple qui soit comporte deux issues possibles (pour une seule issue, l'expérience n'est pas aléatoire), c'est la loi de Bernoulli.

Les deux issues possibles $x_1 = 0$ et $x_2 = 1$, apparaissent avec probabilités $p(x_1) = p$ et $p\left(x_2\right) = 1 - p$. L'entropie s'écrit $H\left(X\right) = \alpha\left[-p \log\left(p\right) - (1-p) \log\left(1-p\right)\right]$. Cette entropie est maximale lorsque les deux issues de l'expérience sont équiprobables $p = 1-p = 1/2$. Choisir α, c'est choisir l'unité de mesure de l'incertitude. Le choix le plus couramment adopté attribue une incertitude de 1 bit à l'expérience aléatoire la plus simple qui soit : le pile ou face équitable. Avec ce choix du bit en tant qu'unité de mesure, l'entropie de la loi de Bernoulli prend la forme :

$$H\left(X\right) = -p \log_2\left(p\right) - (1-p) \log_2\left(1-p\right) \text{ bit(s)}$$

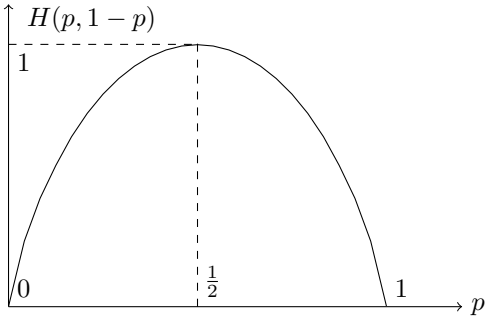


Figure 1.1: Entropie d'une loi de Bernoulli.

La fonction

$$H\left(X\right) = -p \log_2\left(p\right) - (1-p) \log_2\left(1-p\right) \text{ bit(s)}$$

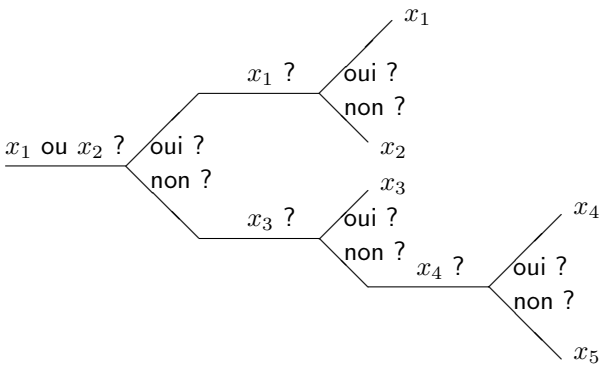
est

- continue en la variable p,
- nulle en $p = 0$ et $p = 1$ (loi déterministe),
- maximale en $p = 1/2$ (loi uniforme),
- strictement concave sur l'intervalle $[0, 1]$.

1.4 Applications à la construction d'un questionnaire

Exemple de bon questionnaire sous optimal

Considérons une v.a. X pouvant prendre cinq états $x_i, i = 1 \cdots 5$ avec les probabilités $p_1 = 0.3, p_2 = 0.2, p_3 = 0.2, p_4 = 0.15, p_5 = 0.15$. L'expérience X étant réalisée, on cherche à en déterminer le résultat à l'aide de questions binaires (à deux réponses). Pour cela, on construit un questionnaire. Par exemple:



Pour ce questionnaire, il faut 2 ou 3 questions pour déterminer celle des cinq issues qui s'est produite.

- Le nombre moyen de questions à poser vaut $N = 2(0.3 + 0.2 + 0.2) + 3(0.15 + 0.15) = 2.3$.
- L'entropie de X vaut quant à elle $H(X) = 2.27$.

Pour ce questionnaire, le nombre moyen de questions à poser est très proche de l'entropie. Cette caractéristique provient de la bonne conception du questionnaire : chaque question est choisie de telle sorte que les deux réponses possibles aient approximativement la même probabilité, c'est-à-dire de manière à maximiser l'entropie. Dans ces conditions chacune des réponses apporte le maximum d'information.

Nous verrons que l'entropie est un minorant du nombre moyen de questions binaires à poser pour déterminer le résultat d'une expérience aléatoire.

La notion d'entropie est importante dans de nombreux domaines, le plus immédiat est celui de la compression : si dans un message long, les différents caractères utilisés pour écrire le message apparaissent avec des fréquences différentes les unes des autres, cela signifie que les caractères qui composent le message n'apportent pas autant d'information que cela est théoriquement possible. Une réécriture du message avec une loi uniforme sur l'alphabet utilisé permettra une compression sans perte, aussi appelée codage entropique.

Exemple de questionnaire optimal

On lance une pièce jusqu'à obtenir face. La probabilité de face est noté p, celle de pile $q = 1 - p$. L'entropie de la pièce (loi de Bernoulli) vaut $H = -p \log_2(p) - q \log_2(q)$.

Le nombre de lancers est une v.a. X à valeurs dans $\mathbb{N}^*$ de loi géométrique :

$$P(X = n) = pq^{n-1}, n \in \mathbb{N}^*$$

L'entropie de X s'écrit :

$$\begin{aligned} H(X) &= -\sum_{n=1}^{+\infty} pq^{n-1}\log_2\left(pq^{n-1}\right) \\ &= -p\log_2(p)\sum_{n=1}^{+\infty} q^{n-1} - p\log_2(q)\sum_{n=1}^{+\infty} q^{n-1}(n-1) \\ &= -p\log_2(p)\underbrace{\sum_{n=1}^{+\infty} q^{n-1}}_{\frac{1}{1-q}=\frac{1}{p}} - p\log_2(q)\underbrace{\sum_{n=1}^{+\infty} nq^{n}}_{\frac{q}{(1-q)^2}=\frac{q}{p^2}} \\ &= \frac{-p\log_2(p) - q\log_2(q)}{p} \\ &= \frac{H}{p} \text{ bits.} \end{aligned}$$

Pour $p = 1/2$, $H(X) = 2$ bits.

Pour trouver le résultat de cette expérience, le questionnaire suivant est efficace car les réponses à chacune des questions binaires sont de mêmes probabilités :

- $X = 1$?
  - Si, oui, terminé en 1 question (avec probabilité $1/2$)
  - Si non, $X = 2$?
    - Si, oui, terminé en 2 questions (avec probabilité $1/4$)
    - Si non, $X = 3$?
      - Si, oui, terminé en 3 questions (avec probabilité $1/8$)
      - Si non, $X = 4$? $\cdots$

Réponse en une question avec probabilité $1/2$, en 2 questions avec probabilité $1/4$, en 3 questions avec probabilité $1/8$, etc. La longueur moyenne du questionnaire est exactement égale à l'entropie de X :

$$\sum_{n=1}^{\infty} n\left(\frac{1}{2}\right)^n = \frac{\frac{1}{2}}{\left(1-\frac{1}{2}\right)^2} = 2 = H(X)$$

1.5 Propriétés de l'entropie

1.5.1 Propriétés démontrées en TD

Maximisantes et minimisantes de l'entropie d'une v.a. à état fini.

- $0 \leq H(p_1, \cdots, p_N) \leq \log_2 N$
  - $H(p_1, \cdots, p_N) = 0$ lorsque la v.a. X est déterministe. Une variable déterministe est d'entropie minimale (désordre minimum).
  - $H(\frac{1}{N}, \cdots, \frac{1}{N}) = \log_2 N$: la loi uniforme est d'entropie maximale (désordre maximum). Pour un nombre d'états $N < +\infty$ l'entropie est maximale lorsque la distribution de probablité est uniforme sur l'ensemble des états. La loi uniforme discrète donne le même poids $1/N$ à chacune des N réalisations possibles. Du fait de la symétrie de la loi (par permutation des variables), l'incertitude moyenne (entropie) est égale à l'incertitude liée à chacune des réalisations. En particulier, lorsque $N = 2^k$, l'entropie est égale à k bits. On retrouve de cette façon l'usage courant de l'unité d'information. En

informatique, un octet comporte 8 bits d'information. Nous voyons ici que cette affirmation n'est exacte, au sens de la théorie de l'information, que lorsque les "bits" sont des v.a. de Bernoulli équilibrées statistiquement indépendantes (la loi est uniforme sur les 256 états). Nous sommes ainsi amenés à séparer la notion de bit, au sens de valeur binaire, de celle de bit d'information.

Inégalité de Gibbs.

- Soient $P = \{p_i\}_{i\in\{1,\cdots,N\}}$ et $Q = \{q_i\}_{i\in\{1,\cdots,N\}}$ deux lois de probabilité.

$$\sum_{i=1}^{N} p_i \log_2 \frac{q_i}{p_i} \leq 0 \text{ avec égalité lorsque les 2 lois sont identiques} \tag{1.1}$$

1.5.2 Propriétés laissées à titre d'exercice

- $H(p_1, \cdots, p_N)$ est une fonction positive, symétrique et continue des variables p_i.
- $H(p_1, \cdots, p_N)$ est une fonction strictement concave des p_i sur l'ensemble des lois de probabilité.
- L'association de plusieurs événements fait décroître l'entropie. Cette propriété est claire sur le cas limite qui consiste à grouper en un seul état l'ensemble des valeurs possibles d'une v.a. X. Inversement, la dissociation d'évènements accroît l'entropie.

1.6 Entropie conjointe

Considérons deux v.a. X à valeurs dans $x_1, \cdots, x_N$ et Y à valeurs dans $y_1 \cdots, y_M$. Il est possible de considérer le couple (X, Y) comme une seule v.a. Z pouvant prendre NM états $z_1, \cdots, z_{NM}$ en associant de manière bijective les z_k aux couples (x_i, y_j) par $z_{i+(j-1)N} = (x_i, y_j)$. D'où la définition naturelle $H(X,Y) = H(Z) = -\sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i, y_j)\log_2 p(x_i, y_j)$.

1.6.1 Définition de l'entropie conjointe.

Pour deux v.a. X, à valeurs dans l'alphabet $\mathcal{A}_X$, et Y, à valeurs dans $\mathcal{A}_Y$, l'entropie conjointe des v.a. X et Y est définie par :

$$H(X,Y) = -\sum_{x\in\mathcal{A}_X}\sum_{y\in\mathcal{A}_Y} p(x,y)\log_2 p(x,y) \tag{1.2}$$

1.6.2 Majoration de l'entropie conjointe.

Comparaison de l'entropie de la loi composée à l'entropie des lois marginales :

$$H(X,Y) \leq H(X) + H(Y) \text{ (égalité lorsque } X \text{ et } Y \text{ sont indépendantes).} \tag{1.3}$$

En effet

$$H(X) + H(Y) = -\sum_{x\in\mathcal{A}_X} p(x)\log_2 p(x) - \sum_{y\in\mathcal{A}_Y} p(y)\log_2 p(y)$$

or,

$$\begin{aligned} p(x) &= \sum_{y\in\mathcal{A}_Y} p(x,y) \\ p(y) &= \sum_{x\in\mathcal{A}_X} p(x,y) \end{aligned}$$

d'où

$$H(X) + H(Y) = -\sum_{x\in\mathcal{A}_X}\sum_{y\in\mathcal{A}_Y} p(x,y)\log_2 p(x)p(y)$$

La comparaison à $H(X,Y) = -\sum_{x\in\mathcal{A}_X}\sum_{y\in\mathcal{A}_Y} p(x,y)\log_2 p(x,y)$ résulte de (1.1).

$$\left\{\underbrace{-\sum_{x\in\mathcal{A}_X}\sum_{y\in\mathcal{A}_Y} p(x,y)\log_2 p(x,y)}_{H(X,Y)}\right\} - \left\{\underbrace{-\sum_{x\in\mathcal{A}_X}\sum_{y\in\mathcal{A}_Y} p(x,y)\log_2 p(x)p(y)}_{H(X)+H(Y)}\right\}$$

$$=$$

$$\sum_{x\in\mathcal{A}_X}\sum_{y\in\mathcal{A}_Y} p(x,y)\log_2\frac{p(x)p(y)}{p(x,y)} \leq 0$$

soit

$$H(X,Y) \leq H(X) + H(Y)$$

avec égalité lorsque X et Y sont indépendantes. □

1.7 Entropie conditionnelle

L'entropie d'une v.a. Y est une fonction des probabilités *a priori* $p(y)$. Si l'état d'une autre v.a. X a été observé, l'incertitude moyenne sur Y sachant que $X = x$, que nous noterons indifféremment $H(Y|X = x)$ ou $H(Y|x)$, peut être définie par :

$$H(Y|X = x) = H(Y|x) = -\sum_{y\in\mathcal{A}_Y} p(y|x)\log_2 p(y|x) \tag{1.4}$$

Cette quantité représente l'entropie *a posteriori* sur Y sachant que X s'est réalisée en x.

1.7.1 Définition de l'entropie conditionnelle

De façon plus générale, il est intéressant de chiffrer l'entropie *a posteriori* moyenne qu'il est possible de définir de la manière suivante :

$$H(Y|X) = \sum_{x\in\mathcal{A}_X} p(x)H(Y|x) \tag{1.5}$$

En explicitant $H(Y|X = x)$, cette entropie conditionnelle s'écrit :

$$H(Y|X) = -\sum_{x\in\mathcal{A}_X}\sum_{y\in\mathcal{A}_Y} p(x,y)\log_2 p(y|x) \tag{1.6}$$

L'entropie conditionnelle de Y par rapport à X est donnée par l'espérance de $-\log_2 p(Y|X)$ par rapport à la loi conjointe de X et de Y.

1.7.2 Propriétés de l'entropie conditionnelle

Règle de chaînage.

$$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y) \tag{1.7}$$

En effet

$$\begin{aligned} H(X,Y) &= -\sum_{x\in\mathcal{A}_X}\sum_{y\in\mathcal{A}_Y} p(x,y)\log_2 \overbrace{p(x,y)}^{p(x)p(y|x)} \\ &= -\sum_{x\in\mathcal{A}_X}\sum_{y\in\mathcal{A}_Y} p(x,y)\log_2 p(x) - \sum_{x\in\mathcal{A}_X}\sum_{y\in\mathcal{A}_Y} p(x,y)\log_2 p(y|x) \\ &= H(X)+H(Y|X) \end{aligned}$$

□

L'extension à n variables est immédiate :

$$H(X_1,\cdots,X_n)=\sum_{i=1}^{n} H(X_i|X_{i-1},\cdots,X_1) \tag{1.8}$$

Réduction de l'entropie par conditionnement.

De

$$H(X)+H(Y)\geqslant H(X,Y)=H(X)+H(Y|X)$$

on déduit la majoration de l'entropie conditionnelle par l'entropie *a priori* :

$$H(Y|X)\leqslant H(Y) \tag{1.9}$$

La variable X apporte de l'information et réduit l'incertitude sur Y. L'entropie *a posteriori* sur Y (après observation de X) est plus faible que l'entropie *a priori* (avant observation de X) .

Attention, ceci est un résultat en moyenne qui n'est pas exact pour le conditionnement par un évènement donné.

Deux cas particuliers utiles :

v.a. totalement dépendantes $Y=X$. $\Pr(X=\alpha|X=x)=\delta_{x-\alpha}$ (1 si $x=\alpha$ et 0 sinon), l'entropie de cette loi déterministe est nulle $H(X|x)=0$, d'où $H(X|X)=\sum_{x\in\mathcal{A}_X} p(x)H(X|x)=0$. Sachant X, il ne subsiste aucune incertitude sur X.

v.a. totalement indépendantes X et Y indépendantes. $H(X)+H(Y)=H(X,Y)=H(X)+H(Y|X)$ d'où $H(Y)=H(Y|X)$. Sachant X (indépendante de Y) l'incertitude sur Y est toujours $H(Y)$.

Asymétrie de l'entropie conditionnelle.

L'entropie conditionnelle n'est pas symétrique :

$$\left.\begin{array}{rcl} H(Y|X) &=& H(X,Y)-H(X) \\ H(X|Y) &=& H(X,Y)-H(Y) \end{array}\right\} \Rightarrow H(Y|X)\neq H(X|Y)$$

par contre :

$$\left.\begin{array}{rcl} H(Y)-H(Y|X) &=& H(Y)-H(X,Y)+H(X) \\ H(X)-H(X|Y) &=& H(X)-H(X,Y)+H(Y) \end{array}\right\} \Rightarrow H(Y)-H(Y|X)=H(X)-H(X|Y)$$

1.8 Divergence de Kullback et information mutuelle

1.8.1 Divergence de Kullback-Leibler ou entropie relative.

La divergence de Kullback, ou entropie relative, entre deux distributions p et q définies sur un même alphabet $\mathcal{A}$ est donnée par :

$$D(p||q)=\sum_{x\in\mathcal{A}} p(x)\log_2\frac{p(x)}{q(x)} \tag{1.10}$$

On adopte les conventions naturelles $0 \log_2 \frac{0}{0} = 0$, $0 \log_2 \frac{0}{q} = 0$ et $p \log_2 \frac{p}{0} = \infty$. D'après (1.1) la divergence de Kullback est non négative :

$$D(p||q) \geq 0 \tag{1.11}$$

avec égalité si et seulement si les lois p et q sont identiques.

Cette divergence mesure une proximité entre deux lois mais ce n'est pas une distance : elle n'est pas symétrique et ne satisfait pas l'inégalité triangulaire.

1.8.2 Information mutuelle.

L'information mutuelle (notée $I(X;Y)$) est la divergence de Kullback entre la loi conjointe $p(x,y)$ et le produit de ses marginales $p(x)p(y)$:

$$I(X;Y) = \sum_{x \in \mathcal{A}_X} \sum_{y \in \mathcal{A}_Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}$$

Elle mesure l'information apportée par une v.a. sur une autre. L'observation d'une v.a. Y réduit l'incertitude moyenne sur toute v.a. X statistiquement liée à Y. $I(X;Y)$ est la réduction d'incertitude sur X due à la connaissance de Y, on a :

$$I(X;Y) = H(X) - H(X|Y) \tag{1.12}$$

Preuve :

$$\begin{aligned} I(X;Y) &= \sum_{x \in \mathcal{A}_X} \sum_{y \in \mathcal{A}_Y} p(x,y) \log_2 \frac{\overbrace{p(x,y)}^{p(x|y)p(y)}}{p(x)p(y)} \\ &= \underbrace{\left\{ - \sum_{x \in \mathcal{A}_X} \sum_{y \in \mathcal{A}_Y} p(x,y) \log_2 p(x) \right\}}_{H(X)} - \underbrace{\left\{ - \sum_{x \in \mathcal{A}_X} \sum_{y \in \mathcal{A}_Y} p(x,y) \log_2 p(x|y) \right\}}_{H(X|Y)} \end{aligned}$$

□

Si $Y = X$, l'information mutuelle entre une v.a. X et elle-même se réduit à l'entropie :

$$I(X;X) = H(X) - H(X|X) = H(X)$$

Une propriété intéressante de cette information mutuelle est sa symétrie : l'information apportée par la v.a. Y sur la v.a. X est égale à la réduction d'incertitude moyenne sur X due à l'observation de Y.

$$I(X;Y) = I(Y;X) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

De la règle de chaînage de l'entropie conditionnelle (1.8), on déduit de manière directe celle de l'information mutuelle :

$$I(X_1, \cdots, X_n; Y) = \sum_{i=1}^{n} I(X_i; Y | X_{i-1}, \cdots, X_1)$$

1.9 Exercices

Maximum d'entropie sous contrainte de support

La v.a. X suit une loi discrète à nombre d'états N fini. Les probabilités des états sont $\{p_0, p_1, \cdots, p_{N-1}\}$.

1. Montrer que $H(p_0, \cdots, p_{N-1}) \geq 0$. Dans quels cas a-t-on $H(p_0, \cdots, p_{N-1}) = 0$?

2. Soient $\mathcal{P} = \{p_j\}_{j=0\cdots N-1}$ et $\mathcal{Q} = \{q_j\}_{j=0\cdots N-1}$ deux lois de probabilité à N états. Montrer :
$$D(\mathcal{P}||\mathcal{Q}) = \sum_{j=0}^{N-1} p_j \log_2 \frac{p_j}{q_j} \geq 0$$
Dans quel cas a-t-on égalité ?
3. En choisissant la loi $\mathcal{Q} = \{q_j\}_{j=0\cdots N-1}$, montrer que $H(p_0, \cdots, p_{N-1}) \leqslant \log_2 N$.
4. Quelle est la loi qui maximise l'entropie de la v.a. X ?
Quelle est l'entropie de cette loi ?

Maximum d'entropie sous contrainte de support

1.
   - $p_j \geq 0$ et $-\log_2 p_j \geq 0$ pour $p_j \in [0,1]$ d'où $H(p_0, \cdots, p_{N-1}) \geq 0$.
   - Les termes $-p_j \log_2 p_j$ de la somme $-\sum_{j=0}^{+\infty} p_j \log_2 p_j$ étant positifs ou nuls, on a $H(p_0, \cdots, p_{N-1}) = 0$ si tous sont nuls. Or $-p_j \log_2 p_j = 0$ pour $p_j \in \{0,1\}$. Comme $\sum_{j=0}^{+\infty} p_j = 1$, on a $H(p_0, \cdots, p_{N-1}) = 0$ si tous les p_j sauf un sont nuls, c'est-à-dire si X est déterministe.

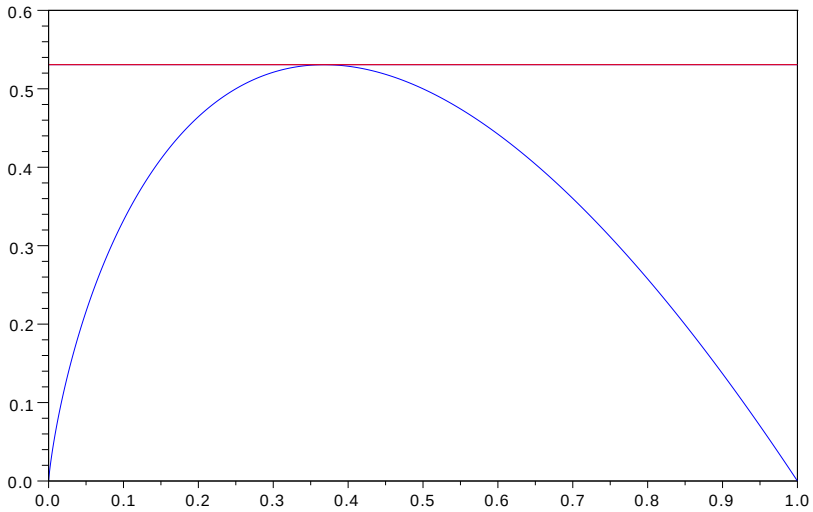


Figure 1.2: Allure de la fonction $-p \log_2$.

2. A la constante $\log(2)$ près, cette inégalité résulte directement de $\log(x) \leqslant x - 1$ (cf. figure 1.3).
$$\sum_{i=0}^{N-1} p_j \log \frac{q_j}{p_j} \leq \sum_{i=0}^{N-1} p_j \left(\frac{q_j}{p_j} - 1 \right) = \sum_{i=0}^{N-1} q_j - \sum_{i=0}^{N-1} p_j = 0$$
3. En prenant Q uniforme, il en résulte que l'entropie $H(P)$ de toute loi P est majorée par l'entropie de la loi uniforme Q : $H(p_0, \cdots, p_{N-1}) \leqslant \log_2 N$.
$$\sum_{i=0}^{N-1} p_j \log_2 \frac{1/N}{p_j} = -\sum_{i=0}^{N-1} p_j \log_2 p_j - \log_2(N) \sum_{i=0}^{N-1} p_j \leq 0$$

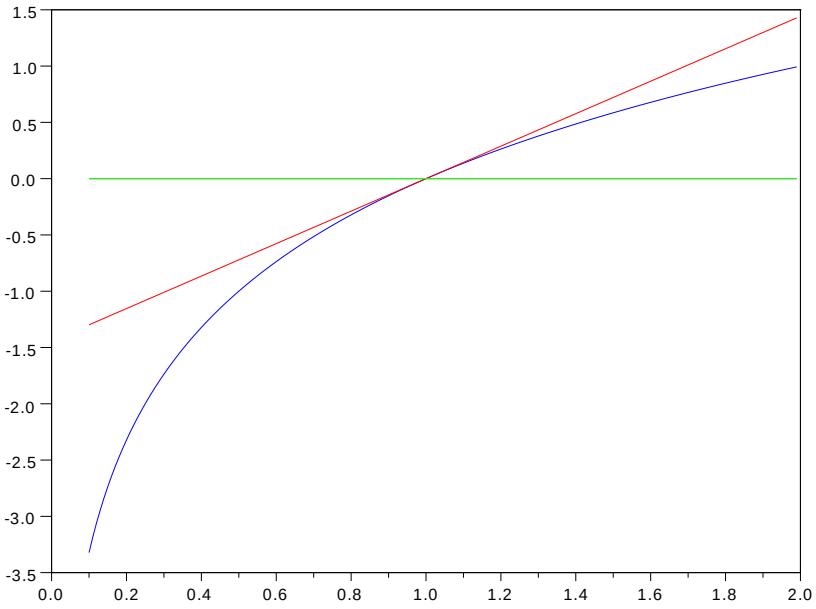


Figure 1.3: Majoration $\log_2(x) \leqslant (x-1)/\ln 2$.

d'où

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

4. Dans $H(X) \leq \log_2(N)$ l'égalité est atteinte pour $p_j = 1/N, j = 0 \cdots N-1$

Maximum d'entropie sous contrainte de moyenne

1. On rappelle que $\sum_{j=0}^{\infty} j\,\beta^j = \frac{\beta}{(1-\beta)^2}$ pour $0 < \beta < 1$.

   Montrer que $\left\{q_k = \alpha\beta^k\right\}_{k\in\mathbb{N}}$ est une loi de probabilité de moyenne μ pour $\beta = \frac{\mu}{\mu+1}$ et $\alpha = \frac{1}{1+\mu}$.

2. Calculer l'entropie de la loi $\left\{q_k = \alpha\beta^k\right\}_{k\in\mathbb{N}}$ de moyenne μ.

3. Montrer que la loi de probabilité discrète $\left\{q_k = \alpha\beta^k\right\}_{k\in\mathbb{N}}$ maximise l'entropie d'une v.a. X à valeurs entières non négatives sous contrainte de moyenne :

$$\mathbb{E}X = \sum_{k=0}^{\infty} k q_k = \mu$$

4. Si la contrainte de support est levée pour une contrainte de moyenne, la loi uniforme n'est plus la loi qui maximise l'entropie.

   Constuisons un exemple simple qui illustre ce point :

(a) Parmi les lois uniformes à support inclus dans $\mathbb{N}$, quelle sont celles dont la moyenne est égale à $\mu = 1$?

(b) Quelle est l'entropie maximale d'une loi uniforme de moyenne $\mu = 1$?

(c) Comparer l'entropie de la loi uniforme de moyenne 1 à celle de la maximisante de même moyenne trouvée à la première question.

Maximum d'entropie sous contrainte de moyenne

1. Pour contrainte de moyenne :

$$\alpha \frac{\beta}{(1-\beta)^2} = \mu$$

Une loi de probabilité somme à 1, d'où la deuxième contrainte :

$$\alpha \sum_{j=0}^{\infty} \beta^j = \frac{\alpha}{1-\beta} = 1$$

Finalement :

$$\begin{aligned} \beta &= \frac{\mu}{\mu+1} \\ \alpha &= \frac{1}{1+\mu} \end{aligned}$$

2. L'entropie de la loi $\left\{q_k = \alpha\beta^k\right\}_{j\in\mathbb{N}}$ sous contrainte de moyenne s'écrit :

$$\begin{aligned} -\sum_{k=0}^{\infty} q_k \log_2 q_k &= \log_2 -(\alpha) - \left(\underbrace{\sum_{k=0}^{\infty} q_k}_{1}\right) - \log_2(\beta) \left(\underbrace{\sum_{k=0}^{\infty} -k\, q_k}_{\mu}\right) \\ &= -\log_2 \alpha - \log_2 -(\beta)\mu \\ &= (\mu+1)\log_2(\mu+1) - -\mu \log_2 \mu \end{aligned}$$

3. **Solution 1.** En prenant $\left\{q_k = \alpha\beta^k\right\}_{k\in\mathbb{N}}$ dans l'inégalité

$$-\sum_{k=0}^{\infty} p_k \log_2 p_k \leq -\sum_{k=0}^{\infty} p_k \log_2 q_k$$

et en utilisant les deux contraintes, on a :

$$-\sum_{k=0}^{\infty} p_k \log_2 p_k \leq -\log_2(\alpha) \left(\underbrace{\sum_{k=0}^{\infty} p_k}_{1}\right) - \log_2(\beta) \left(\underbrace{\sum_{k=0}^{\infty} k\, p_k}_{\mu}\right)$$

Le second membre coincide avec l'entropie de la loi $\left\{q_k = \alpha\beta^k\right\}_{k\in\mathbb{N}}$: toute loi de probabilité $\{p_k\}_{k\in\mathbb{N}}$ qui vérifie la contrainte de moyenne a une entropie inférieure ou égale à celle de la loi $\left\{q_k = \alpha\beta^k\right\}_{k\in\mathbb{N}}$.

Remarque : si le support était $[k_0, +\infty[$, on aurait juste d'autres valeurs pour les constantes α et β :

$$\begin{aligned}\sum_{j=k_0}^{+\infty} \alpha\beta^j &= \alpha\frac{\beta^{k_0}}{1-\beta} = 1\\ \sum_{j=k_0}^{+\infty} \alpha j\beta^j &= \alpha\frac{\beta^{k_0}\left(\beta + k_0 - k_0\beta\right)}{\left(1-\beta\right)^2} = \mu\end{aligned}$$

d'où :

$$\begin{aligned}\beta &= \frac{(\mu - k_0)}{(\mu - k_0) + 1}\\ \alpha &= \frac{\left((\mu - k_0) + 1\right)^{k_0 - 1}}{\left(\mu - k_0\right)^{k_0}}\end{aligned}$$

Solution 2 - Lagrange. (a) On veut maximiser l'entropie $-\sum_{j=0}^{+\infty} p_j \log_2 p_j$ sous deux contraintes :

i. $\sum_{j=0}^{+\infty} p_j = 1$: p_j est une loi de probabilité.

ii. $\sum_{j=0}^{+\infty} j\, p_j = \mu$: la contrainte de moyenne.

$H(p)$ est concave, elle possède donc un maximum unique qui peut être trouvé par la méthode des multiplicateurs de Lagrange en annulant le gradient de :

$$-\sum_{j=0}^{+\infty} p_j \log_2 p_j + \lambda_1\left[\sum_{j=0}^{+\infty} p_j - 1\right] + \lambda_2\left[\sum_{j=0}^{+\infty} j\, p_j - \mu\right]$$

En dérivant par rapport à p_k :

$$-\log_2 p_k - 1 + \lambda_1 + \lambda_2 j = 0 \rightarrow p_k = 2^{\lambda_1 - 1}\left(2^{\lambda_2}\right)^k$$

En posant $\alpha = 2^{\lambda_1 - 1}$ et $\beta = 2^{\lambda_2}$, la loi est de la forme :

$$p_k = \alpha\beta^k$$

Les constantes α et β s'obtiennent en utilisant les contraintes :

$$\begin{aligned}\sum_{j=0}^{+\infty} p_j &= 1 \rightarrow \frac{\alpha}{1-\beta} = 1\\ \sum_{j=0}^{+\infty} j\, p_j &= \mu \rightarrow \frac{\alpha}{\left(1-\beta\right)^2} = \mu + 1\end{aligned}$$

En remplaçant la première égalité dans la seconde :

$$\beta = \frac{\mu}{\mu + 1}$$

La première donne ensuite :

$$\alpha = \frac{1}{\mu + 1}$$

Finalement :

$$p_k = \frac{1}{\mu + 1}\left(\frac{\mu}{\mu + 1}\right)^k, k \geq 0$$

4. 3 lois sont solution :
   (a) La loi qui concentre toute la probabilité sur la valeur 1, son entropie est nulle.
   (b) La loi uniforme sur 0 et 2, son entropie est $\log_2 2 = 1$
   (c) La loi uniforme sur $\{0;1;2\}$, son entropie vaut $\log_2 3$. En effet, une v.a. uniforme sur $\{0;1;\cdots,N-1\}$ a une moyenne égale à
   $$\mu = \frac{1}{N}\sum_{k=0}^{N-1} k = \frac{1}{N}\frac{N(N-1)}{2} = \frac{N-1}{2}$$
   Pour avoir $\mu = 1$, la seule solution est $N = 3$.
5. L'entropie de la loi uniforme de moyenne $\mu = 1$ vaut donc au maximum
   $$H_U = \log_2 3.$$
6. Cette entropie est à comparer à celle de la loi géométrique de même moyenne $\mu = 1$ qui vaut :
   $$H_G = (\mu+1)\log_2(\mu+1) - \mu\log_2\mu = 2\log_2 2 = \log_2 4 > \log_2 3 = H_U$$

Octet d'information

On considère un octet, c'est-à-dire un octuplet $(X_1,\cdots,X_8)$ composé de 8 v.a. binaires à valeurs dans $\{0;1\}$.

1. Quelle doit être la distribution des probabilités sur les configurations de l'octet pour qu'une v.a. à valeur octet soit la plus informative possible ? Quelle est alors l'entropie maximale ?
2. Pour un octet uniforme, quelle est la loi d'une composante binaire X_j ?
3. Si les composantes suivent toutes une loi uniforme sur $\{0;1\}$, peut-on en déduire que l'octet est uniforme sur ses 256 configurations ?
4. Pour un octet uniforme, montrer que deux composantes binaires X_n et $X_{m\neq n}$ sont indépendantes.
5. Montrer qu'en fait les 8 composantes sont mutuellement indépendantes.
6. Finalement, pour que l'octet soit d'entropie maximale, quelles sont les conditions requises sur ses composantes ?

Octet d'information

1. Il faut que les $2^8 = 256$ configurations soient équiprobables. L'entropie maximale est celle de la loi uniforme sur 256 états, soit $H = \log_2(2^8) = 8$ bits.
2. Il faut calculer la marginale de la loi conjointe en sommant sur les 7 autres variables :
   $$Pr[X_j = \alpha] = \sum_{X_{i\neq j}\in\{0;1\}, X_j=\alpha} Pr[X_1,\cdots,X_8] = 128.\frac{1}{256} = \frac{1}{2}$$
3. Non, contre-exemple : X_1 uniforme sur $\{0;1\}$ et $X_j = X_1$ pour $j \neq 1$ qui implique un octet uniforme sur seulement 2 états $(0,\cdots,0)$ et $(1,\cdots,1)$.

4. Il faut montrer que

$$Pr[X_n = \alpha, X_m = \beta] = Pr[X_n = \alpha] \times Pr[X_m = \beta] = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$$

$Pr[X_n = \alpha, X_m = \beta]$ s'obtient en marginalisant (somme sur les 64 configurations possibles des 6 composantes différentes de n et m) :

$$Pr[X_n = \alpha, X_m = \beta] = 64 \times \frac{1}{256} = \frac{1}{4}$$

5. Montrer qu'en fait les 8 composantes sont mutuellement indépendantes.
6. L'octet est d'entropie maximale si et seulement si les composantes sont mutuellement indépendantes et uniformément distribuées sur $\{0; 1\}$.

Maximum d'entropie sous contrainte de moment d'ordre 2

Remarques préliminaires :

- Pour cet exercice, on peut utiliser la méthode des multiplicateurs de Lagrange.
- Attention, la distribution cherchée ne peut pas être normalisée simplement.

1. Quelle est la loi de probabilité discrète qui maximise l'entropie d'une v.a. X à valeurs entières non négatives sous contrainte de moment d'ordre deux $\sum_{k=0}^{\infty} k^2 p_k = M^2$?
2. Quelle est l'entropie de cette loi ?

Maximum d'entropie sous contrainte de moment d'ordre 2

La contrainte porte sur le moment d'ordre deux.

1. Maximisation de l'entropie par la méthode des multiplicateurs de Lagrange avec deux contraintes :
   - $\sum_i q_i = 1$ (q est une loi de probabilité), et
   - $\sum_i i^2 q_i = M^2$ (le moment d'ordre deux est fixé).

$$\max_q \left\{ -\sum_i (q_i \log q_i) + \lambda_0 (\sum_i q_i) + \lambda_1 (\sum_i i^2 q_i) \right\}$$

En dérivant par rapport à q_k :

$$-\log(q_k) - 1 + \lambda_0 + \lambda_1 k^2 = 0$$

d'où :

$$q_k = \alpha \exp\left(\lambda_1 k^2\right)$$

On vérifie a posteriori que la solution est valide, c'est-à-dire :

- que les probabilités sont positives ($\alpha > 0$ contrainte non encore prise en compte),

- que la loi est sommable ($\lambda_1 < 0$)

- et qu'il n'existe pas une meilleure loi ayant le même moment d'ordre deux. Pour cela on utilise le fait que pour une fonction Φ dérivable convexe, on a (Bregman)

$$\Phi(x) - \Phi(y) - (x - y)\Phi'(y) \geq 0$$

En considérant deux jeux de probabilités p_i et q_i et en sommant sur i, on a donc :

$$\sum_i (\Phi(p_i) - \Phi(q_i) - (p_i - q_i)\Phi'(q_i)) \geq 0$$

En particulier pour $\Phi(x) = x \cdot \log(x)$: $-H(p) + H(q) - \sum_i (p_i - q_i)(\log(q_i) + 1) \geq 0$, soit :

$$-H(p) + H(q) - \sum_i (p_i - q_i)\log(q_i) \geq 0$$

En utilisant ce résultat, a posteriori, pour toutes lois p telle que $\sum_i i^2 q_i = \sum_i i^2 p_i$, on a :

$$-H(p) + H(q) - \sum_i (p_i - q_i)(\log(\alpha) + \lambda_1 i^2) \geq 0$$

soit

$$-H(p) + H(q) - \lambda_1 (\sum_i (i^2 p_i) - \sum_i (i^2 q_i)) \geq 0$$

donc $H(q) \geq H(p)$ (car p et q ont même moment d'ordre deux).

2. Il reste à déterminer la constante α telle que $\sum_i \alpha \exp\left(\lambda_1 i^2\right) = 1$:

$$\alpha = \frac{1}{\sum_i \exp\left(\lambda_1 i^2\right)}$$

(pas de forme explicite)

Trouver l'intrus, version Roberval

On dispose de 9 pièces, l'une d'elles est fausse et son poids diffère de celui des autres pièces (disons de 10%). On se pose deux questions :

Q1 trouver la fausse pièce sachant qu'elle est plus lourde que les autres,

Q2 trouver la fausse pièce et déterminer si elle est plus ou moins lourde que les autres.

Pour répondre à ces questions, le seul outil à disposition est une balance de type Roberval (figure 1.4) qui permet seulement des pesées comparatives (la balance est en équilibre, penche à gauche ou penche à droite).

Questions générales. -

1. Quelle est la quantité d'information maximale que peut apporter une pesée ?
2. Quelle première pesée choisir pour glaner le maximum d'information ?

Résolution de Q1. -

1. Quelle est l'entropie de la question Q1 ?
2. Est-il toujours possible de trouver la fausse pièce en une seule pesée ? Pourquoi ? Et en deux pesées ?
3. Comment isoler pratiquement la fausse pièce à la deuxième pesée ?

Résolution de Q2. -

[Une solution plus formelle avec entropies conditionnelles sera également possible si le cours est suffisamment avancé avant le TD.]

On souhaite maintenant non seulement identifier la fausse pièce mais également savoir si elle est plus ou moins lourde que les autres (question Q2).

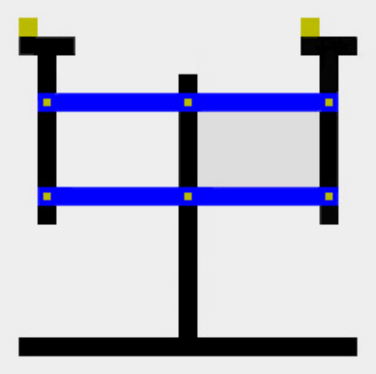


Figure 1.4: Balance de Roberval. Gilles Personne de Roberval (mathématicien et physicien français 1602-1675).

1. Quelle est l'entropie de la question Q2 ?
2. La première pesée utilisée pour Q1 étant optimale, elle est réutilisée pour Q2. Supposons qu'à cette première pesée, la balance penche à gauche : cela implique que la fausse pièce se trouve parmi les 6 pièces présentes sur les plateaux. 2 pesées sont elles suffisantes pour trouver une fausse pièce parmi 6 ET savoir si elle est plus lourde ou plus légère que les vraies pièces ?
3. Donner une procédure pratique en 3 pesées au maximum permettant de déterminer laquelle des pièces est fausse et de savoir si elle est plus ou moins lourde que les autres.

Trouver l'intrus, version Roberval

Questions générales. -

1. Une pesée peut donner trois résultats : la balance est en équilibre, penche à gauche ou penche à droite. L'entropie est maximale lorsque ces trois résultats sont équiprobables, elle vaut alors $\log_2 3 \approx 1.58$ bits. C'est la quantité d'information maximale que peut apporter une pesée.

   En général, l'entropie d'une pesée vérifie : $0 \leq H \leq \log_2 3$.

   Dans cet exercice le bit n'est pas l'unité de mesure naturelle, il serait plus commode d'utiliser H_3 qui donne une quantité homogène à un nombre de questions.
2. Si le nombre de pièces diffère entre les deux plateaux, il n'est pas possible d'avoir équilibre, l'expérience de pesée est déterministe : son entropie vaut 0.

   Il suffit donc de considérer les cas dans lesquels il y a le même nombre n de pièces sur chacun des plateaux : n pièces à gauche, n pièces à droite et $9-2n$ pièces hors de la balance.

   Les probabilités des trois états possibles de la balance sont :

   Penche à gauche $P_{\leftarrow}$. La fausse pièce est à gauche et elle est plus lourde que les autres (probabilité $n/18$) ou la fausse pièce est à droite et elle est plus légère que les autres (probabilité $n/18$). La probabilité pour que la balance penche à gauche est donc $P_{\leftarrow} = n/9$.

   Penche à droite $P_{\rightarrow}$. Par symétrie, la probabilité pour que la balance penche à droite vaut également $P_{\rightarrow} = n/9$.

   Équilibre $P_{\downarrow}$. La somme des probabilités valant 1, la probabilité d'équilibre vaut $P_{\downarrow} = (9-2n)/9$.

   L'entropie est maximale lorsque les trois états ont la même probabilité, c'est-à-dire pour $(9-2n)/9 = n/9$, soit $n = 3$.

Résolution de Q1. -

1. 9 résultats sont possibles, l'entropie de l'expérience vaut $\log_2 9 = 2\log_2 3 \approx 3.16$ bits : exactement l'information maximale que peuvent fournir deux pesées.
2. Est-il toujours possible de trouver la fausse pièce en une seule pesée ? Pourquoi ? Et en deux pesées ?

   En termes de codage, on a toujours $l_{\max} \geq \nu \geq H_D$, où $l_{\max}$ est la longueur du mot le plus long, $\nu = \sum p_j l_j$ la longueur moyenne des mots et $H_D = H/\log_D 2$ l'entropie en base D.

   Ici la longueur moyenne des mots (nombre moyen de pesées) joue le rôle de la compacité ν du code, les 3 résultats possibles de chaque pesée correspondent à un alphabet du code à $D = 3$ états. Cette question revient à demander s'il est possible que ν ($a\ fortiori\ l_{\max}$) soit inférieur à l'entropie $H_3 = H/\log_3 2 = 2$.

   La réponse négative à cette question ne dit pas qu'il n'existe pas de séries de pesées qui donnent parfois la solution en une seule pesée, mais ces séries possèdent un nombre moyen de pesées plus grand que la solution optimale.

   En 2 pesées, l'information maximale obtenue est $2\log_2 3$: elle est égale à l'entropie de la question $Q1$. Il n'est donc pas impossible qu'une solution existe.

   En terme de codage il faut $\nu = H_D$, avec ici $H_D = H(X)/\log_2 3 = 2$. En général, atteindre la borne est impossible car il faut pour cela que les mots soient de longueurs $-\log_D p_j$ et ces quantités ne sont en général pas entières. Ici en revanche, on a $-\log_D p_j = -\log_3(1/9) = 2, \forall j \in \{1 \cdots 9\}$. Les longueurs étant entières, il est possible d'atteindre la borne $\nu = H_D$.

   La contrainte pour trouver toujours la fausse pièce en 2 questions est plus forte encore, il faut que le mot de longueur maximale $l_{\max}$ soit de longueur H_D, cela n'est possible que pour $l_{\max} = \nu$, c'est-à-dire lorsque tous les mots sont de même longueur, c'est le cas ici.

   Est-ce que cela prouve pour autant qu'une solution existe ?
3. Une solution pratique est donnée par la figure 1.5 : on sait que la première pesée optimale utlise 3 pièces par plateau, la deuxième en découle de manière évidente. En termes de codage : toutes les

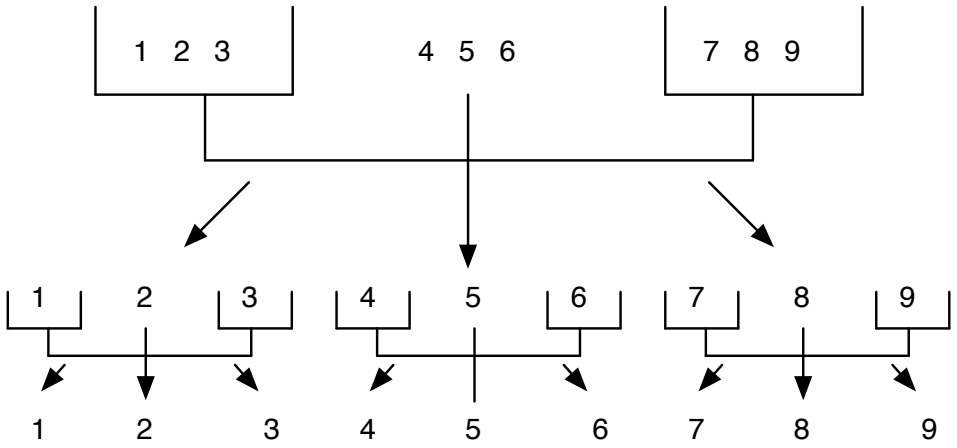


Figure 1.5: Résolution de la question Q1 en exactement 2 pesées.

   possibilités sont trouvées en exactement 2 pesées, le nombre moyen de pesée vaut donc $\nu = 2$ on a ici $\nu = H_D$. On sait que la condition pour atteindre la borne est $l_j = -\log_3 p_j$, ici les p_j sont des puissances entières de $1/D = 1/3$: $p_j = 1/9$ donc $l_j = -\log_3 1/9 = 2$.

Résolution de Q2. -

On souhaite maintenant non seulement identifier la fausse pièce mais également savoir si elle est plus ou moins lourde que les autres (question Q2).

1. Il y a 18 états possibles : 1 pièce parmi 9 pouvant être plus ou moins lourde que les vraies pièces. L'entropie de Q2 vaut $\log_2 18$.
2. Non, 2 pesées ne sont pas suffisantes pour trouver la fausse pièce parmi ces 6 et savoir si elle est plus lourde ou plus légère que les vraies pièces : l'expérience est à 12 états, il faut donc $\log_2 12$ bits alors que deux pesées en apporte au mieux $\log_2 9$.

   Celà étant, se ramener au problème précédent avec 6 pièces au lieu de 9 n'est pas une bonne approche, elle prive d'une partie de l'information obtenue lors de la première pesée (quelles pièces étaient dans quel plateau, quelles pièces étaient en dehors).

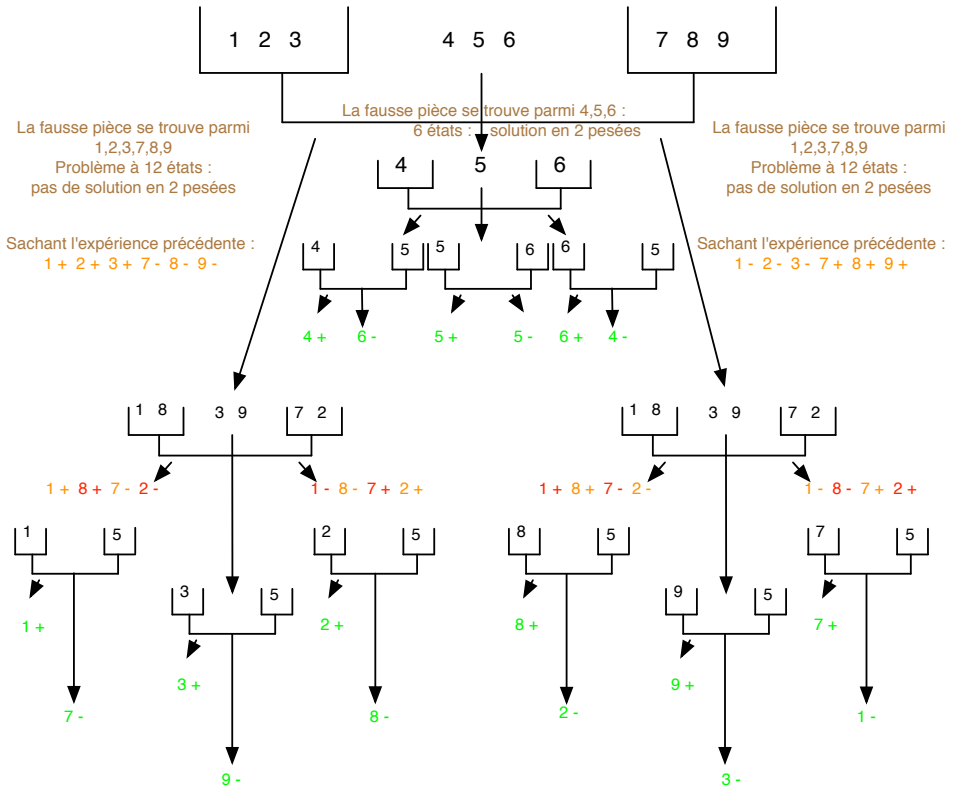


Figure 1.6: Résolution de la question Q2 en 3 pesées. En rouge les cas interdits par les pesées précédentes, en orange les cas possibles, en vert les cas avérés. + pour plus lourd et - pour moins lourd. A chaque pesée il est nécessaire de conserver la mémoire des résultats obtenus aux pesées précédentes, sans quoi la solution en 3 pesées est impossible.

3. Bien que la réponse à la question précédente soit négative, il est possible de trouver une solution en 2 pesées supplémentaires sachant le résultat de la première pesée : si la balance penche à gauche à la première pesée, il y a deux possibilités : la fausse pièce se trouve parmi 1,2,3 ET elle est plus lourde ou la fausse pièce se trouve parmi 7,8,9 ET elle est plus légère : il n'y a donc que 6 états possibles. On ne peut pas repartir de zéro à la deuxième expérience (mélanger les pièces), il faut garder la mémoire de la première (quelle pièce était dans quel plateau).

   Du point de vue des entropies : la première pesée est d'entropie $H(P_1)$, la deuxième d'entropie $H(P_2) = \log_2 12$ mais $H(P_2|P_1) = \log_2 6 < \log_2 9$ ainsi, sachant la première pesée, il n'est impossible de conclure en deux questions supplémentaires.

   Une solution pratique est donnée par la figure 1.6 : on sait que la première pesée optimale utlise 3 pièces par plateau, comme nous l'avons vu à la question précédente, la deuxième pesée n'en découle

Trouver l'intrus, version Columbo

On dispose maintenant d'une balance numérique qui fonctionne de 0 à 1 kg par pas de 0.1g.

La masse d'une pièce authentique est de 10 g.

5 sacs contiennent des pièces, dans l'un d'eux les pièces sont fausses et pèsent chacune 0.5 g de moins que les vraies pièces.

Pour utiliser la balance, on peut prendre le nombre de pièces souhaité dans chaque sac.

1. Donner un encadrement de l'entropie de la balance.
2. Les sacs contiennent chacun 37 pièces. Est-il possible d'identifier le sac de fausses pièces en une seule pesée ?
3. Même question si les sacs contiennent chacun 3 pièces.

Trouver l'intrus, version Columbo

1. La balance peut prendre les états 0 ; 0.1 ; 0;2 ; $\cdots$ 10000 : c'est-à-dire 10 001 états possibles, l'entropie est donc comprise entre 0 (la balance prend toujours le même état) et $\log_2 10001 \approx 13$ bits (les états sont pris avec la même probabilité)
2. Il faut être capable de construire 5 états discernables de la balance qui permettent d'identifier les 5 hypothèses possibles : le sac de fausse pièce est le sac 1, ..., le sac 5. Si les sacs contiennet suffisamment de pièces, de nombreuses solutions pratiques sont possibles. Par exemple : 1 pièce dans le premier sac, 2 dans le deuxième sac, ... Dans ce cas, si toutes les pièces étaient vraies, le poids serait $10*(1+2+3+4+5) = 150$ g, la différence avec le poids P affiché par la balance est un multiple de 0.5 qui donne le numéro du sac : $(150 - P)/0.5$ vaut 1 si une pièce est fausse (sac 1), 2 si 2 pièces sont fausses (sac 2) etc.
3. Lorsque les sacs contiennent peu de pièces, la question est de construire 5 états discernables, cela suppose que :

   i/ les sacs contiennent suffisamment de pièces,

   ii/ l'écart de poids entre une vraie pièce et une fausse pièce n'est pas trop grand (par exemple, si la fausse pièce pèse deux fois le poids d'une vraie pièce cela pose problème).

Complexité du tri en nombre de comparaisons

Soit E un ensemble totalement ordonné. On suppose que la seule opération possible sur E est de comparer deux éléments avec la comparaison $<_E$. La comparaison $a <_E b$ entre a et b s'apparente à une pesée.
Soit $T = [t_1, \ldots, t_n]$ un tableau de n éléments distincts; T est une variable aléatoire (v.a.) à valeurs un tableau de n éléments distincts.

On appelle tri de T la séquence (v.a.) $I_T = [i_1, \ldots, i_n]$ d'indices distincts définie par $\{i_1, \ldots, i_n\} = \{1, \ldots, n\}$ et $e_{i_1} <_E e_{i_2} <_E \ldots <_E e_{i_n}$.

1. *Montrer que l'entropie $H(I_T)$ de la v.a. I_T vérifie $H(I_T) \leq \log_2 n!$ bits.*
2. *Donner une distribution sur T telle que $H(I_T) = \log_2 n!$ bits.*

3. *Montrer qu'il n'existe pas d'algorithme qui trie un tableau de n éléments distincts de E en effectuant moins de $\log_2 n!$ comparaisons en pire cas.*
   N.B. Avec la formule de Striling

$$\log_2 n! \simeq_{n\to+\infty} \log_2\left(\sqrt{2\pi n} n^n e^{-n}\right) \simeq_{n\to+\infty} n\log_2 n - n\log_2 e.$$

4. *Justifier que le nombre de comparaisons $C(n)$ de l'algorithme récursif de tri par partition-fusion vérifie: $C(2) = 1$ et $C(n) = C(n/2) + C(n - n/2) + n - 1$. En déduire $C(n) \leq n\lceil\log_2 n\rceil$. Qu'en déduisez-vous ?*

5. *E est l'ensemble des n entiers entre $\{1, \ldots n\}$. Donner un algorithme qui trie un tableau d'éléments de E en faisant $O(n)$ opérations et 0 comparaisons. Est-ce contradictoire avec la question 3?*

Complexité du tri en nombre de comparaisons

1. Pour n éléments, il y a $n!$ valeurs de I_T possibles. Donc $H(I_T) \leq \log_2 n!$ bits.

2. Une distribution uniforme sur les $n!$ sorties possibles maximise l'entropie. Il y a plusieurs méthodes possibles pour construire une distribution sur l'entrée T qui donne une distribution uniforme sur I_T. Par exemple, on fixe n éléments distincts triés de E: $a_1 <_E \ldots <_E a_n$. Puis on tire uniformément une permutation $(i_1, \ldots, i_n)$ de $(1, \ldots, n)$ comme suit: i_1 choisi uniformément dans $\{1, \ldots, n\}$; puis i_2 uniformément parmi les $n-1$ indices de $\{1, \ldots, n\}$ différents de i_1, etc. Et on construit le tableau T par $T(i_j) = a_j$ pour $j = 1, \ldots, n$. Ainsi $I_T = [i_1, \ldots, i_n]$ suit bien une distribution uniforme.

3. Chaque comparaison apporte 1 bit d'information. Donc il faut au moins $\log_2 n!$ comparaisons pour obtenir la sortie quand l'entropie de I_T est maximale.

4. - Les équations donnant le nombre de comparaison se déduisent directement de l'algorithme : $C(2) = 1$ et $C(n) = C(n/2) + C(n - n/2) + n - 1$.
     Donc $C(n) \leq 2C(\lceil n/2\rceil) + n = \sum_{i=0}^{\lceil\log_2 n\rceil - 1} 2^i \frac{n}{2^i} = n\lceil\log_2 n\rceil$.
   - Détails, pour $n = 2^p$, on a $C(n) \leq 2C\left(\frac{n}{2}\right) + n - 1$
     - i/ Le premier terme $2C\left(\frac{n}{2}\right)$ est le coût du tri de deux tableaux de taille $n/2$.
       ii/ Le second terme est un majorant du coût de la fusion de deux tableaux triés. En effet, le pire cas est celui pour lequel l'élément s'insère juste après le premier élément de la liste : il faut 2 comparaisons pour insérer un seul élément, cela se produit $(n/2 - 1)$ fois, et quand il ne reste qu'un seul élément dans la liste de destination, 1 comparaison suffit, soit un total de $2 \times (n/2 - 1) + 1 = n - 1$ comparaison dans le pire cas.
     - En itérant $C(n) \leq 2C\left(\frac{n}{2}\right) + n - 1$, on a :

$$\begin{aligned}
C(n) &\leq 2\left(2C\left(\frac{n}{4}\right) + \frac{n}{2} - 1\right) + n - 1 \\
&= 2^2 C\left(\frac{n}{2^2}\right) + (n - 2^1) + (n - 2^0) \\
&\vdots \\
&\leq 2^{p-1} C\left(\frac{n}{2^{p-1}}\right) + (n - 2^{p-2}) + \cdots + (n - 2^0) \\
&= 2^{p-1} C\left(\frac{n}{2^{p-1}}\right) + n(p-1) - \sum_{k=0}^{p-2} 2^k \\
&= 2^{p-1} C\left(\frac{n}{2^{p-1}}\right) + n(p-1) + 1 - 2^{p-1}
\end{aligned}$$

En prenant $n = 2^p$, c'est-à-dire $p = \log_2 n$, et en utilisant $C(2) = 1$ on a :

$$C(n) \leq \frac{n}{2} + n(\log_2 n - 1) + 1 - \frac{n}{2} = n(\log_2 n - 1) + 1$$

5. Il suffit de faire pour $i = 1 \dots n$: $I[T[i]] = i$. Ce n'est pas contradictoire: les opérations effectuées sur E ne sont pas des comparaisons: une opération sur un indice de $\log_2 n$ bits apporte ici une information de $\log_2 n$ bits !

Détection de la langue par comptage des lettres

Un message $\mathbf{s}$ est une suite de N caractères pris dans un alphabet $\mathcal{A}$ composé de α caractères : $s_1, \cdots, s_N$ avec $s_j \in \mathcal{A}$ (par exemple $\mathcal{A} = \{A, B, ..., Z\}$, $\alpha = 26$, les espaces entre mots ne sont pas pris en compte dans cette approche.). L'espace Ω des messages de longueur N contient α^N éventualités dont la probabilité dépend de la langue. On observe un message $\mathbf{m} = m_1, \cdots, m_N$; l'objectif est de déterminer, avec un taux d'erreur minimum, la langue de ce message.

Le message est écrit dans une langue l avec $l \in \{0, 1\}$. Les différentes hypothèses pour la langue sont notées H_l : sous l'hypothèse H_l, le message est écrit dans la langue l et ses caractères ont des probabilités *a priori* $\left\{p_c^{(l)}\right\}_{c\in\mathcal{A}}$ supposées connues.

$f_c(\mathbf{m})$, la fréquence empirique du caractère c dans le message $\mathbf{m}$, est l'estimation de cette loi obtenue par comptage des différents caractères dans le message observé $\mathbf{m}$. La fréquence empirique de chaque caractère $c \in \mathcal{A}$ est donnée par :

$$f_c(\mathbf{m}) = \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}_{m_j = c}\,,\ c \in \mathcal{A}$$

Remarques :

- Lorsque la longueur de message tend vers l'infini, pour un message $\mathbf{m}$ écrit dans la langue l, cette estimation $\{f_c(\mathbf{m})\}_{c\in\mathcal{A}}$ tend vers $\left\{p_c^{(l)}\right\}_{c\in\mathcal{A}}$ (loi des grands nombres).
- $f_c(\mathbf{m})$ est le nombre d'occurrences du caractère c dans le message $\mathbf{m}$ divisé par N.

La probabilité *a priori* (avant observation d'un message) pour que le message soit écrit dans la langue l est notée $\mathcal{P}_{H_l}$ (probabilité *a priori* de la langue l). Les $\mathcal{P}_{H_l}$ sont supposées connues (sans *a priori* on choisit $\mathcal{P}_{H_0} = \mathcal{P}_{H_1} = 1/2$).

On note $\widehat{H_\lambda}$ l'hypothèse décidée sur la langue et H_l l'hypothèse effective pour le message observé.

On note $\Pr\left[\widehat{H_\lambda}|H_l\right]$ la probabilité de décider la langue λ (décider l'hypothèse H_λ) sachant que la langue l est utilisée (hypothèse H_l effective). La probabilité d'erreur s'écrit :

$$P_e = \mathcal{P}_{H_0} \Pr\left[\widehat{H_1}|H_0\right] + \mathcal{P}_{H_1} \Pr\left[\widehat{H_0}|H_1\right] \quad \text{(Probabilité d'erreur.)} \tag{1.13}$$

Le but de cet exercice est de répondre à deux questions :

- Dans un message donné, comment détecter la langue (avec un taux d'erreur minimum) à partir de la fréquence empirique d'apparition des différents caractères ?
- Quelle est la fiabilité (probabilité d'erreur P_e) de cette décision ?

L'idée consiste à partitionner l'espace Ω en 2 régions de décision D_0 et D_1 de telle sorte que :

- si le message $\mathbf{m}$ observé appartient à D_j on décide l'hypothèse H_j.

- les régions D_j sont choisies de sorte que la probabilité d'erreur P_e soit minimale. $\{D_j\}_{j\in\{0,1\}}$ est l'ensemble des messages pour lesquels on décide H_j.

Pour simplifier la démarche, on modélise les caractères comme des variables aléatoires indépendantes de même loi.

En notant $\Pr[\mathbf{z}|H_l] = P_l(\mathbf{z})$ la probabilité du message $\mathbf{z}$ sous l'hypothèse H_l :

$$\Pr\left[\widehat{H_\lambda}|H_l\right] = \sum_{\mathbf{z}\in D_\lambda} \Pr[\mathbf{z}|H_l].$$

1. Proposer une solution pratique pour la détection de la langue qui n'utilise que l'histogramme empirique des caractères dans le message et les histogrammes *a priori* des deux langues en concurence.
2. Exprimer la probabilité d'erreur P_e en fonction de $\mathcal{P}_{H_0}$, $\mathcal{P}_{H_1} = 1 - \mathcal{P}_{H_0}$, $P_0(\mathbf{z})$ et $P_1(\mathbf{z})$ en sommant sur les ensembles D_0 et D_1 ; puis seulement sur l'ensemble D_1.
3. En déduire que la région D_1 à choisir pour que la probabilité d'erreur soit minimale est
$$D_1 = \{\mathbf{z} \in \Omega / \mathcal{P}_{H_0} P_0(\mathbf{z}) - \mathcal{P}_{H_1} P_1(\mathbf{z}) < 0\}.$$
4. Combien y-a-t-il d'éléménts dans Ω ? On considère un algorithme qui précalcule $\mathcal{D}_1$ pour tester ensuite l'appartenance d'un mot à $\mathcal{D}_1$. Quel est son coût? Qu'en pensez-vous ?
5. Les caractères du message étant supposés indépendants, exprimer le test d'appartenance du message $\mathbf{m}$ observé à la région D_1 en fonction de $\mathcal{P}_{H_0}$, $\mathcal{P}_{H_1}$ et des $p^{(0)}_{m_k}$, $p^{(1)}_{m_k}$?
6. Réécrire le test établi à la question précédente en fonction des fréquences empiriques $f_c(\mathbf{m})$ des caractères dans le message observé $\mathbf{m}$.
7. Réécrire le test en fonction des deux divergences de Kullback $D\left(f_c(\mathbf{m})||p^{(1)}_c\right)$ et $D\left(f_c(\mathbf{m})||p^{(0)}_c\right)$.
   Commentez le cas sans *a priori* $\mathcal{P}_{H_1} = \mathcal{P}_{H_0}$ et la manière dont l'*a priori* impacte le test.
8. Donner un algorithme implémentant ce test et donner son nombre d'opération arithmétiques. Comparer au coût du test précédent (question 3).
9. Que devient le test lorsque deux langues ont des caractères distribués selon la même loi de probabilité *a priori* ? Et si, de plus, $\mathcal{P}_{H_1} = \mathcal{P}_{H_0}$?
10. En supposant les caractères indépendants, montrer que la probabilité d'erreur s'écrit
$$P_e = \frac{1}{2} - \frac{1}{2}\sum_{\mathbf{z}\in\Omega}\left|\mathcal{P}_{H_1}\prod_{c\in\mathcal{A}} {p^{(1)}_c}^{N f_c(\mathbf{z})} - \mathcal{P}_{H_0}\prod_{c\in\mathcal{A}} {p^{(0)}_c}^{N f_c(\mathbf{z})}\right|$$
    Que peut-on dire de l'utilisation pratique de ce résultat ?

Détection de la langue par comptage des lettres

1. Par exemple, calculer la somme des modules des écarts de fréquence entre la loi empirique et chacune des deux lois *a priori* et choisir la langue dont la distribution est la plus proche en ce sens de la loi empirique.
   Le choix d'une mesure de distance est arbitraire et les performances en probabilité d'erreur n'ont pas de raison d'être optimales.
   Cette approche ne dit pas comment prendre en compte une information *a priori* quant à la probabilité que le message soit écrit dans une langue plutôt qu'une autre.

$$\begin{aligned} P_e &= \mathcal{P}_{H_0} \sum_{\mathbf{z}\in D_1} P_0(\mathbf{z}) + \mathcal{P}_{H_1} \sum_{\mathbf{z}\in D_0} P_1(\mathbf{z}) \\ &= \mathcal{P}_{H_0} \sum_{\mathbf{z}\in D_1} P_0(\mathbf{z}) + \mathcal{P}_{H_1} \left(1 - \sum_{\mathbf{z}\in D_1} P_1(\mathbf{z})\right) \\ &= \mathcal{P}_{H_1} + \sum_{\mathbf{z}\in D_1} \left[\mathcal{P}_{H_0} P_0(\mathbf{z}) - \mathcal{P}_{H_1} P_1(\mathbf{z})\right] \end{aligned}$$

3. Choisir
$$D_1 = \{\mathbf{z} \in \Omega / \mathcal{P}_{H_0} P_0(\mathbf{z}) - \mathcal{P}_{H_1} P_1(\mathbf{z}) < 0\}$$
c'est sélectionner les éléments de Ω qui réduisent la probabilité d'erreur et écarter ceux qui l'augmente. C'est-à-dire sélectionner les z tels que $\mathcal{P}_{H_0} P_0(\mathbf{z}) - \mathcal{P}_{H_1} P_1(\mathbf{z}) < 0$

4. Pour tous les mots z de ω, si $\mathcal{P}_{H_0} P_0(z) - \mathcal{P}_{H_1} P_1(z) < 0$ alors ajouter z à $\mathcal{D}_1$. Le coût est donc $\Omega\left(|\mathcal{A}|^N\right)$. Ce coût est exponentiel en la taille N du mot en entrée, prohibitif pour $|\mathcal{A}| = 26$ et $n = 100$[1].

5. Si les caractères sont indépendants, la probabilité d'un message est le produit des probabilités marginales pour chacun des ses caractères :
$$P_l(\mathbf{z}) = \prod_{k=1}^{N} p_{z_k}^{(l)}$$
Ainsi, le test d'appartenance du message $\mathbf{m}$ observé à la région D_1 s'écrit :
$$\frac{\prod_{k=1}^{N} p_{m_k}^{(1)}}{\prod_{k=1}^{N} p_{m_k}^{(0)}} > \frac{\mathcal{P}_{H_0}}{\mathcal{P}_{H_1}}$$

6. On peut écrire ce test en fonction des fréquences empiriques des caractères :
$$\log \frac{\prod_{c\in\mathcal{A}} {p_c^{(1)}}^{N f_c(\mathbf{m})}}{\prod_{c\in\mathcal{A}} {p_c^{(0)}}^{N f_c(\mathbf{m})}} > \log \frac{\mathcal{P}_{H_0}}{\mathcal{P}_{H_1}}$$
$$\sum_{c\in\mathcal{A}} N f_c(\mathbf{m}) \log p_c^{(1)} - \sum_{c\in\mathcal{A}} N f_c(\mathbf{m}) \log p_c^{(0)} > \log \frac{\mathcal{P}_{H_0}}{\mathcal{P}_{H_1}}$$
$$N \sum_{c\in\mathcal{A}} f_c(\mathbf{m}) \log \frac{p_c^{(1)}}{p_c^{(0)}} > \log \frac{\mathcal{P}_{H_0}}{\mathcal{P}_{H_1}}$$

7. Ce test s'écrit également :
$$N \sum_{c\in\mathcal{A}} f_c(\mathbf{m}) \log \frac{p_c^{(1)}}{p_c^{(0)}} \frac{f_c(\mathbf{m})}{f_c(\mathbf{m})} > \log \frac{\mathcal{P}_{H_0}}{\mathcal{P}_{H_1}}$$
C'est-à-dire :
$$N \left[D\left(f_c(\mathbf{m}) || p_c^{(0)}\right) - D\left(f_c(\mathbf{m}) || p_c^{(1)}\right)\right] > \log \frac{\mathcal{P}_{H_0}}{\mathcal{P}_{H_1}}$$

[1] 26^{100} est très supérieur au nombre estimé en 2010 de changements de spin dans l'univers observable depuis le Big Bang, de l'ordre de 10^{125}.

Autrement dit, on calcule les divergences de Kullback entre les fréquences empiriques d'une part et les lois des langues 0 et 1 d'autre part. L'écart entre ces 2 divergences est comparé à un seuil qui dépend de l'a priori.

Ce seuil vaut zéro dans le cas où l'a priori ne privéllégie aucune des deux langues, dans ce cas le choix est celui de la langue dont la loi est la plus proche des fréquences empiriques (au sens de la divergence de Kullback).

8. L'algorithme est direct en remarquant que $Nf_c(m) =$ nombre d'occurences de c dans m:

   somme $= 0$; POUR $i = 1 \dots N$ FAIRE somme $+= \log \frac{p^{(1)}_{m_i}}{p^{(0)}_{m_i}}$; RETURN (somme $> \log \frac{\mathcal{P}_{H_0}}{\mathcal{P}_{H_1}}$) ;

   Le coût est $O(N)$ en utilisant les deux tables $p_c^{(j)}$, donc linéaire en la taille du mot en entrée. Le langage est détecté en temps réel $O(N)$ avec une probabilité d'erreur minimum.

9. On décide H_1 si $\mathcal{P}_{H_1} > \mathcal{P}_{H_0}$, c'est-à-dire l'hypothèse qui est la plus probable *a priori*. Si les deux langues ont même probabilité *a priori* les deux hypothèses sont équiprobables, on ne peut pas trancher.

10. Le codage des extensions de la source : considérer comme source les extensions d'ordre 3 permet de prendre en compte les syllabes et d'appréhender beaucoup mieux la structure (dépendance entre les caractères) de la langue.

11.
$$\begin{aligned} P_e &= \mathcal{P}_{H_0} \sum_{\mathbf{z}\in D_1} P_0(\mathbf{z}) + \mathcal{P}_{H_1} \sum_{\mathbf{z}\in D_0} P_1(\mathbf{z}) \\ &= \mathcal{P}_{H_1} + \sum_{\mathbf{z}\in D_1} [\mathcal{P}_{H_0} P_0(\mathbf{z}) - \mathcal{P}_{H_1} P_1(\mathbf{z})] \\ &= \mathcal{P}_{H_0} + \sum_{\mathbf{z}\in D_0} [-\mathcal{P}_{H_0} P_0(\mathbf{z}) + \mathcal{P}_{H_1} P_1(\mathbf{z})] \end{aligned}$$

   La demi somme :

$$\begin{aligned} P_e &= \frac{1}{2} - \frac{1}{2} \sum_{\mathbf{z}\in D_0} \overbrace{[\mathcal{P}_{H_0} P_0(\mathbf{z}) - \mathcal{P}_{H_1} P_1(\mathbf{z})]}^{\geq 0 \text{ pour } z\in D_0} \\ &- \frac{1}{2} \sum_{\mathbf{z}\in D_1} \underbrace{[-\mathcal{P}_{H_0} P_0(\mathbf{z}) + \mathcal{P}_{H_1} P_1(\mathbf{z})]}_{\geq 0 \text{ pour } z\in D_1} \end{aligned}$$

   d'où
$$P_e = \frac{1}{2} - \frac{1}{2} \sum_{\mathbf{z}\in\Omega} |\mathcal{P}_{H_1} P_1(\mathbf{z}) - \mathcal{P}_{H_0} P_0(\mathbf{z})|$$

   En utilisant $P_0(\mathbf{z}) = \prod_{k=1}^N p^{(0)}_{z_k}$ et $P_1(\mathbf{z}) = \prod_{k=1}^N p^{(1)}_{z_k}$, on a finalement :
$$P_e = \frac{1}{2} - \frac{1}{2} \sum_{\mathbf{z}\in\Omega} \left| \mathcal{P}_{H_1} \prod_{k=1}^N p^{(1)}_{z_k} - \mathcal{P}_{H_0} \prod_{k=1}^N p^{(0)}_{z_k} \right|$$

   ou, en utilisant les fréquences empiriques :
$$P_e = \frac{1}{2} - \frac{1}{2} \sum_{\mathbf{z}\in\Omega} \left| \mathcal{P}_{H_1} \prod_{c\in\mathcal{A}} {p_c^{(1)}}^{Nf_c(\mathbf{z})} - \mathcal{P}_{H_0} \prod_{c\in\mathcal{A}} {p_c^{(0)}}^{Nf_c(\mathbf{z})} \right|$$

   Trouvez cette probabilité impose de sommer sur tous les éléments de Ω, ce calcul se fait hors ligne, une seule fois. Pour autant, il est très lourd et souvent pratiquement non réalisable.

Chapter 2

Compressibilité et entropie.

Lorsque l'on cherche à réécrire un message dans un alphabet donné de la manière la plus brève possible, une question essentielle est la détermination d'une borne inférieure à cette compacité. L'existence d'une telle borne permet de situer les performances de tout algorithme pratique de compression par rapport à cette performance optimale.

L'objectif principal de ce chapitre est de trouver une telle borne en montrant que la longueur moyenne minimale du message codé est liée à l'entropie.

2.1 Définition d'un codage de source

Soit X une v.a. prenant ses valeurs x dans l'ensemble fini $\mathcal{A}_X = \{x_1, \cdots, x_N\}$ avec les probabilités $\{p(x_1) = p_1, \cdots, p(x_N) = p_N\}$.

2.1.1 Codage d'un seul état

Définition d'un code source. Un code source C est une application de $\mathcal{A}_X$ dans $\mathcal{D}^*$ où $\mathcal{D}^*$ est l'ensemble des séquences de longueur finie écrites dans un alphabet à D caractères $\{a_1, \cdots, a_D\}$.

$$\begin{array}{rccc} C: & \mathcal{A}_X & \rightarrow & \mathcal{D}^* \\ & x & \rightarrow & C(x) \end{array}$$

Le code source C associe le mot de code $C(x)$ à l'état x de la v.a. X.

Ainsi, chacun des N états possibles $x_1, \cdots, x_N$ (alphabet de la source) de la v.a. X est décrit à l'aide d'une séquence finie de caractères de l'alphabet du code $a_1, \cdots, a_D$. Chacune de ces séquences s'appelle un mot-code ; l'ensemble des mots du code s'appelle le code.

Remarque. Ce problème est le même que celui de la théorie des questionnaires (évoqué au chapitre 1). Les caractères des mots du code peuvent être considérés comme les réponses à des questions destinées à déterminer le résultat de l'expérience aléatoire.

Exemples de codes.

Code binaire. Ecriture d'un message alphabétique écrit avec l'alphabet $\{a, b, \cdots, z\}$ à l'aide d'un code binaire d'alphabet $\{0, 1\}$: $D = 2$, $a_1 =$ "0" et $a_2 =$ "1". L'alphabet du code étant choisi, il existe de nombreuses manières de choisir les mots-code. Un exemple en informatique est le code ASCII.

Code morse. Le code morse est basé sur l'utilisation d'un alphabet-code composé de quatre symboles : le trait, le point et deux séparateurs l'un pour les caractères l'autre pour les mots. Le code morse est construit de façon à faire correspondre les mots les plus courts aux lettres de l'alphabet les plus probables. Le "e", lettre

la plus fréquente, est codée à l'aide d'un seul point, le mot code le plus court. Cette procédure permet de réduite la longueur moyenne des mots du code. Elle n'est pas particulière au code morse. Nous verrons dans des cas plus généraux qu'il s'agit d'une propriété essentielle d'un code optimal.

Compacité. La longueur du mot de code $C(x)$ est notée $l(x)$ (ou $l(x_j) = l_j$). La longueur moyenne des mots, aussi appelée compacité du code, s'écrit :

$$\nu = \sum_{x \in \mathcal{A}_X} p(x) l(x) = \sum_{j=1}^{N} p_j l_j$$

L'espérance est prise par rapport à la loi de X : les symboles peu probables n'accroissent que peu la taille moyenne des messages codés alors que les symboles très probables contribuent fortement à la longueur moyenne et doivent de ce fait être les plus courts possible.

Code non singulier. Un code est dit non singulier si l'application C est injective, c'est-à-dire si deux états différents sont codés par deux séquences différentes ; autrement dit si un même mot de code ne peut pas correspondre à plusieurs états de la v.a. X.

Pour un code non singulier, il est possible de remonter sans ambiguïté d'un mot du code à la valeur encodée.

2.1.2 Codage d'un message composé d'une suite d'états

Source simple. Une source faite de copies indépendantes d'une même v.a. X est dite simple. Une telle source délivre des messages composés de réalisations indépendantes d'une v.a. X[1].

Dans la suite, seules des sources simples sont considérées.

Message source. Une suite de n états de la source constitue un message, c'est-à-dire une suite de caractères appartenant à l'alphabet de la source (qui compte N caractères $x_1, \cdots, x_N$).

Codage d'une source simple. L'objectif du codage de source est de transformer un message source en un message codé, c'est-à-dire en une suite de mots du code, eux-mêmes composés de caractères appartenant à l'alphabet[2] du code $a_1, \cdots, a_D$.

Extension du codage C d'un unique état au codage C^* d'un message source. On note $x_{i_k} \in \{x_1, \cdots, x_N\}$ le $k^{\text{ème}}$ état généré par la source, de sorte qu'un message source de longueur n s'écrit $x_{i_1} \cdots x_{i_n}$.

L'extension du codage C d'un unique état de la source en un codage C^* d'un message composé de n caractères (suite de n états de la source) est telle que :

$$\begin{array}{rccc} C^* : & (\mathcal{A}_X)^n & \longrightarrow & \mathcal{D}^* \\ & \underbrace{x_{i_1} \cdots x_{i_n}} & \longrightarrow & \underbrace{C(x_{i_1}) \cdots C(x_{i_n})} \\ & \text{Message source} & & \text{Message codé} \end{array}$$

$C(x_{i_1}) \cdots C(x_{i_n})$ désigne la concaténation des mots $C(x_{i_1})$ à $C(x_{i_n})$.

Codes déchiffrables. Un code est déchiffrable si à chaque message codé correspond au plus un message source. Pour que tout message codé soit déchiffrable, il faut que C^* soit non singulier. Un code déchiffrable est aussi qualifié de séparable du fait que la déchiffrabilité résulte de la possibilité de séparer les mots successifs à la lecture du message codé.

[1] Des sources plus complexes introduisent des dépendances entres les valeurs générées.

[2] Cet alphabet correspond par exemple au type de lettres qu'il est possible de stocker sur un support (0 ou 1 dans une mémoire informatique par exemple) ou aux lettres qu'il est possible de transmettre au travers d'un canal donné.

Exemple de code non déchiffrable. La déchiffrabilité constitue naturellement une propriété essentielle en pratique, elle n'est pas automatique pour autant : codons une source à quatre états par les mots $M_1 = 0, M_2 = 010, M_3 = 01, M_4 = 10$. A la lecture du message codé 010, il n'est pas possible de remonter de façon unique à la séquence qui lui a donné naissance. En effet, ce message peut être interprété comme provenant de M_1M_4 ou de M_2 ou bien encore de M_3M_1. A une seule séquence codée correspondent plusieurs messages de la source : le code n'est pas déchiffrable.

Assurer la déchiffrabilité. Les méthodes classiques qui permettent d'assurer la déchiffrabilité sont :

Longueur unique des mots. La façon la plus évidente d'assurer la déchiffrabilité d'un code consiste à attribuer la même longueur à tous les mots-code. Ainsi, il suffit de compter les caractères pour séparer les mots du code. Les codes ASCII et UTF, par exemple, utilisent cette procédure. Celle-ci peut être naturelle lorsque les données sont stockées par paquet (typiquement octet ou ensemble d'octets).

Séparateur. Une autre manière de procéder consiste à utiliser un caractère supplémentaire destiné à identifier la fin d'un mot-code. Cette technique dégrade bien évidemment les performances du code. Exemple : en morse, un intervalle de temps est placé entre les lettres.

Condition du préfixe. Une condition suffisante de déchiffrabilité plus intéressante, appelée condition du préfixe, consiste à imposer aux mots d'un code la propriété suivante : aucun mot-code ne doit être le préfixe d'un autre mot-code.

Les codes qui vérifient la condition du préfixe sont dit instantanés (ou parfois irréductible). Cette appellation est due au fait qu'il est possible d'effectuer le décodage pas à pas (sans avoir à attendre d'autres caractères pour prendre une décision quant au mot-code lu) : dès qu'un mot est reconnu, il est possible de le séparer de la suite du message sans attendre de lire les caractères qui suivent. Les codes instantanés ont une grande importance pratique.

Un code instantané est déchiffrable mais tous les codes déchiffrables ne sont pas instantanés. A titre d'exemple, considérons le code $\{M_1 = 0, M_2 = 00001\}$. Ce code n'est pas instantané puisque M_1 est le préfixe de M_2. Il est néanmoins déchiffrable. A lecture du message 000001, il n'est pas possible de décider que le premier 0 correspond au mot M_1 puisque ce 0 peut aussi n'être que le début de M_2. Cependant, à la lecture du cinquième 0, on sait de façon certaine que le premier 0 correspondait à M_1. Pour décider M_1 il a fallu attendre pour s'assurer qu'il ne s'agissait pas d'un autre mot-code. C'est ici que se situe l'origine du terme non instantané.

2.2 CNS d'existence d'un code instantané

La taille N de l'alphabet de la source (nombre d'états de la v.a. X), la taille D de l'alphabet du code et les longueurs $\{l_i = l(x_i)\}_{i=1\cdots N}$ des mots du code étant fixées, il n'est pas toujours possible de construire un code. Les mots du code doivent avoir une longueur suffisante pour permettre de représenter toutes les éventualités et satisfaire la contrainte de déchiffrabilité. Dans le cas des codes instantanés, une CNS d'existence est donnée par l'inégalité de Kraft.

Inégalité de Kraft. Si D désigne la taille de l'alphabet utilisé pour le codage, un code instantané composé de mots de longueurs $l_1, l_2 \cdots, l_N$ existe si et seulement si

$$\sum_{i=1}^{N} D^{-l_i} \leqslant 1$$

Notons $l_1 \leq l_2 \leq \cdots \leq l_N$ la longueur des mots-code associés aux états x_i de X. Le nombre maximum de mots de longueur l_N qu'il est possible de former à l'aide d'un alphabet à D lettres est de D^{l_N}.

Supposons qu'il existe un code instantané, l'utilisation du mot de longueur l_i élimine $D^{l_N - l_i}$ possibilités (condition du préfixe). On ne peut pas éliminer plus de point terminaux qu'il n'en existe, c'est-à-dire $\sum_{i=1}^{N} D^{l_N - l_i} \leqslant D^{l_N}$, l'inégalité de Kraft en découle en divisant par D^{l_N}.

Réciproquement, si l'inégalité de Kraft est vérifiée, il est possible de construire un code instantané en choisissant successivement les mots de longueurs $l_1, l_2, \cdots, l_N$.

□

Ce résultat s'étend à l'ensemble des codes déchiffrables, il porte alors le nom de théorème de Mac Millan. Cette extension est d'une grande importance pratique et montre que l'on peut se cantonner à utiliser des codes instantanés sans être pénalisé du point de vue des performances en compression. En effet, s'il existe un code déchiffrable avec des mots de longueurs l_i, ce code satisfait la condition de Mac-Millan et il existe un code instantané pour les mêmes longueurs l_i.

2.3 Codes optimaux théoriques.

Ce paragraphe démontre deux résultats qui permettront ensuite de construire effectivement des codes optimaux :

1. Il existe une borne inférieure pour la longueur moyenne des mots d'un code.
2. Il est possible de s'approcher aussi près qu'on le souhaite de cette borne.

2.3.1 Efficacité d'un code.

On cherche à compresser au mieux un message sans perdre d'information, c'est-à-dire en étant capable de restituer exactement le message original à partir de sa version compressée. Cela revient à réécrire le message avec un nombre de caractères-code aussi réduit que possible. Pour cela, un des critères possibles est celui de la longueur moyenne minimale pour les mots qui composent le code.

La mesure d'incertitude H permet de caractériser le taux de compression qu'il est possible d'espérer pour un message donné et rend possible une mesure effective de l'efficacité d'une procédure de codage.

L'entropie de la source d'information étant $H(X)$ et la longueur moyenne des mots-code ν ; l'entropie moyenne par caractère-code (après codage) est égale à $H(X)/\nu$, cette quantité est majorée par l'entropie de la loi uniforme sur l'ensemble des caractères du code. Ainsi $H(X)/\nu \leq \log_2(D)$. Un code est d'autant plus efficace qu'il approche cette borne, ceci conduit à définir l'efficacité d'un code par :

$$E = \frac{H(X)}{\nu \log_2(D)}$$

Des conditions d'optimalité commencent à apparaître : si ν est un entier, il existe D^ν mots de longueur ν dont les caractères appartiennent à un alphabet de taille D. L'information est maximale lorsque les mots sont équiprobables, elle vaut alors $\log_2[D^\nu] = \nu \log_2[D]$. Cette borne supérieure est atteinte lorsque les caractères qui composent le mot sont indépendants et que chacun d'eux est distribué uniformément sur les D caractères de l'alphabet du code.

Les deux racines de redondance sont là : dépendance entre caractères et distribution non uniforme.

2.3.2 1er théorème de Shannon : la compacité est minorée par l'entropie

Une caractéristique essentielle d'un code réside dans la longueur moyenne des mots qu'il emploie, sa compacité. Cette compacité dépend de la nature de la source (des messages) à coder. Lorsque la source délivre toujours le même message (l'un des symboles est de probabilité un), son entropie est nulle, et le codage est immédiat (un seul mot code, le plus court possible, $i.e.$ de longueur 1). Lorsque la source délivre des symboles équiprobables, l'entropie est maximale et l'on conçoit qu'il n'est pas de compression possible. La prise en considération de ces cas extrêmes fait pressentir le rôle de l'entropie de la source sur la compacité du code. Nous allons maintenant formaliser cette intuition. Le résultat obtenu, appelé premier théorème de Shannon, montre qu'il existe une borne inférieure pour la compacité et précise que cette borne peut être atteinte asymptotiquement.

On note $H_D(X)$ l'entropie en base D de la v.a. X :

$$H_D = -\sum_{i=1}^{N} p_i \log_D p_i = \frac{H(X)}{\log_2 D}$$

La compacité est minorée par l'entropie.

$\nu \geqslant H_D(X)$, la borne est atteinte pour $p_i = D^{-l_i^*} \leftrightarrow l_i^* = -\log_D p_i$.

Pour le montrer, calculons la différence entre la longueur moyenne des mots du code et l'entropie :

$$\nu - H_D(X) = \sum_{i=1}^{N} p_i l_i - \left(-\sum_{i=1}^{N} p_i \log_D p_i \right)$$

en insérant $l_i = -\log_D D^{-l_i}$:

$$\begin{aligned}
\nu - H_D(X) &= \sum_{i=1}^{N} p_i \log_D p_i - \sum_{i=1}^{N} p_i \log_D D^{-l_i} \\
&= \sum_{i=1}^{N} p_i \log_D \left[\frac{p_i}{\left(\frac{D^{-l_i}}{\sum_{i=1}^{N} D^{-l_i}} \right) \sum_{i=1}^{N} D^{-l_i}} \right] \\
&= \underbrace{\log_D \frac{1}{\sum_{i=1}^{N} D^{-l_i}}}_{\geq 0} + \underbrace{\sum_{i=1}^{N} p_i \log_D \left[\frac{p_i}{\left(\frac{D^{-l_i}}{\sum_{i=1}^{N} D^{-l_i}} \right)} \right]}_{D(p||q) \geq 0}
\end{aligned}$$

Le second terme est la divergence de Kullback $D\left(p||q\right)$ entre la loi $\{p_i\}_{i=1\cdots N}$ et la loi $\left\{ q_i = \frac{D^{-l_i}}{\sum_{i=1}^{N} D^{-l_i}} \right\}_{i=1\cdots N}$ tandis que le premier terme est positif ou nul d'après l'inégalité de Kraft. On en déduit :

$$\nu - H_D(X) \geq 0$$

L'égalité est atteinte si et seulement si les deux conditions suivantes sont vérifiées :

1. $D\left(p||q\right) = 0$, c'est-à-dire $p_i = \frac{D^{-l_i}}{\sum_{i=1}^{N} D^{-l_i}}$
2. $\sum_{i=1}^{N} D^{-l_i} = 1$

Autrement dit $\nu = H_D(X)$ si et seulement si $p_i = D^{-l_i}$.

Ceci est possible si et seulement si $l_i = -\log_D p_i$ est entier pour tout $i \in \{1 \cdots N\}$. Ce ceci se produit lorque la distribution des probabilités est de la forme $p_i = D^{-n_i}$ avec n_i entier, une telle distribution est dite D-adic.

□

En pratique, la borne est atteinte à au plus 1 bit près.

Le choix $p_i = D^{-l_i}$ conduit aux longueurs $l_i = -\log_2(p_i)/\log_2(D)$ non-nécessairement entières. Le mieux qu'il est possible de faire est d'arrondir à l'entier supérieur, ainsi :

$$-\log_2(p_i)/\log_2(D) < l_i < -\log_2(p_i)/\log_2(D) + 1$$

En multipliant par p_i et en sommant sur l'ensemble des symboles, on obtient le meilleur encadrement pour la compacité :

$$H_D(X) \leqslant \nu \leqslant H_D(X) + 1$$

Ce résultat dit simplement que le cas le plus défavorable est celui pour lequel tous les mots ont une longueur optimale très légèrement supérieure à un entier.

Remarque. Nous avons ainsi démontré que l'entropie en base 2 représente le nombre moyen minimum de questions qu'il est nécessaire de poser pour déterminer le résultat d'une expérience.

2.4 Codage par bloc

Pour approcher plus finement la borne inférieure, l'idée est de ne plus coder individuellement chacun des états possibles de la source mais de grouper les symboles par bloc pour ensuite coder ces blocs.

Ce groupement peut être vu de la façon suivante.

2.4.1 Extension d'ordre s de la source X.

On appelle extension d'ordre s de la source X la source qui groupe s v.a. successives de type X : $Y = (X_1, \cdots, X_s)$.

L'extension d'ordre s d'une source simple est également une source simple, cette extension peut prendre N^s états.

Par exemple le message $x_{i_1}, \cdots, x_{i_s}, x_{i_{s+1}}, \cdots, x_{i_{2s}}$ issu de X correspond au message y_{i_1}, y_{i_2} issu de Y avec $y_{i_1} = x_{i_1}, \cdots, x_{i_s}$ et $y_{i_2} = x_{i_{s+1}}, \cdots, x_{i_{2s}}$.

2.4.2 Le codage par bloc permet de mieux approcher la borne inférieur H_D.

L'encadrement de la compacité pour l'extension d'ordre s de la source X s'écrit :

$$H_D(Y) \leqslant \nu_s \leqslant H_D(Y) + 1$$

où ν_s est la longueur moyenne des mots pour la source Y, c'est-à-dire pour un groupe de s états de la source X.

Les symboles délivrés par la source X étant indépendants (source simple), l'entropie de $(X_1, \cdots, X_s)$ est la somme des entropies des composantes X_j, les X_j étant des copies indépendantes de X, on a $H(Y) = sH(X)$ d'où

$$H_D(X) \leqslant \nu \leqslant H_D(X) + \frac{1}{s}$$

En codant des extensions de plus en plus longues de la source, il est possible d'approcher la borne optimale. La borne est approchée en introduisant un retard (il faut attendre que s symboles sources soient réunis avant d'effectuer le codage) et en augmentant la complexité du codeur.

2.4.3 Exemple — illustration du gain d'un codage par bloc

Soit X une source qui délivre deux symboles $x_1 =' A'$ et $x_2 =' B'$ avec les probabilités $p(A) = 0.8$ et $p(B) = 0.2$.

L'entropie de la source vaut

$$H(X) = -0.8 \log_2(0.8) - 0.2 \log_2(0.2) = 0.72 \text{ bits}$$

Codage direct de la source. Les longueurs de mot optimales, au sens de la compacité, $l_1^* = -\log_2 0,8 = 0,32$ et $l_2^* = -\log_2 0,2 = 2,32$ permettent d'atteindre la borne inférieure (entropie de la source), elles sont non entières : longueur très courte pour le mot le plus probable, plus longue pour l'autre mot. Arrondir à l'entier supérieur (1 et 3) ne permet pas de construire le meilleur code : le mot de longueur 3 est inutilement long puisqu'il suffit de le choisir de longueur 1 avec comme caractère de code celui qui n'est pas utilisé par le mot de longueur 1 qui code l'état 'A' : dans le tableau ci-dessous, l'état 'A' de la source est codé par le mot 1 et l'état 'B' par le mot 0, la longueur moyenne des mots du code est de $\nu = 1$ caractère.

Finalement, le codage est évident : la source possède 2 états, les mots du code sont choisis aussi courts que possible (longueur 1) et l'alphabet le plus réduit possible ($D = 2$ caractères $a_1 =' 0'$ et $a_2 =' 1'$).

Les longueurs des mots diffèrent des longueurs optimales $l_i^* = -\log_2 p_i$. On vérifie l'encadrement :

$$0.72 = H(X) \leqslant \nu = 1 \leqslant H(X) + 1 = 1.72$$

Symbole	$p(x_i)$	l_i^*	l_i	Mots
A	0.8	0.32	1	1
B	0.2	2.32	1	0

On constate, sur ce cas particulier, que l'impossibilité de choisir des mots de longueur non entière augmente la longueur moyenne de moins de 1 caractère (0.28 dans cet exemple, soit un surcoût non négligeable de 39% sur la longueur moyenne des messages codés).

Arrondir les longueurs optimales à l'entier supérieur donne une longueur moyenne de $0.8 \times 1 + 0.2 \times 3 = 1.4$ qui vérifie également l'encadrement

$$0.72 = H(X) \leqslant 1.4 \leqslant H(X) + 1 = 1.72$$

Codage des extensions d'ordre 2 de la source. Les longueurs optimales des mots ($l_1^* = -\log_2 0,8 = 0,32$ et $l_2^* = -\log_2 0,2 = 2,32$) laissent penser qu'un codage par bloc est intéressant. Considérons l'extension d'ordre $s = 2$ de la source, c'est-à-dire la v.a. Y pouvant prendre les quatre états $y_1 =' AA'$, $y_2 =' AB'$, $y_3 =' BA'$ et $y_4 =' BB'$ avec les probabilités $p_{AA} = p(A)p(A)$, $p_{AB} = p_{BA} = p(A)p(B)$ et $p_{BB} = p(B)p(B)$. On peut alors, par exemple, construire le code suivant :

Symbole	$p(y_i)$	l_i^*	l_i	Mots
AA	0.64	0.643	1	0
AB	0.16	2.643	2	10
BA	0.16	2.643	3	110
BB	0.04	4.643	3	111

Le codage des extensions d'ordre deux de la source conduit à la longueur moyenne $\nu_2 = 1.56$ caractère par paire de caractères de X. Autrement dit $\nu = 0.78$ caractère par symbole de la source X. On vérifie à nouveau l'inégalité :

$$0.72 = H(X) \leqslant \nu = 0.78 \leqslant H(X) + 1/2 = 1.22$$

On constate sur cet exemple l'efficacité d'un codage bloc : la perte est maintenant inférieure à $1/2$, ici elle vaut 0.06 (contre 0.28 pour le codage direct de la source) soit un surcoût de seulement 8% en terme de longueur moyenne des messages codés contre 39% pour le codage directe de la source.

Remarque :

2.4.4 Codage de sources non simples.

Le codage des extensions est certes utile pour que la compacité approche la borne H_D lors du codage d'une source simple mais, le gain reste en général faible et ce d'autant plus que les mots sont longs en moyenne. En pratique, il est le plus souvent inutile de coder des extensions d'ordre $s > 2$.

En revanche, le codage par bloc apporte des gains très importants pour le codage de sources non simples, il s'agit d'une façon simple et efficace de prendre en compte la dépendance des caractères générés par la source. En français par exemple, les lettres forment des syllabes composées 2 ou 3 lettres, ainsi le codage des extensions d'ordre 3 intègre au moins partiellement cette structure forte des messages et la compression croît considérablement (puisque la distribution des probabilités est très inégale sur les suites de 3 lettres, aaa,ztz n'apparaissent jamais par exemple, alors que tre est une sous séquence fréquente)

2.5 Exercices

Codage d'un couple de variables aléatoires

Soient deux v.a. indépendantes X_1 et X_2 (copies indépendantes d'une unique v.a. X d'alphabet $A_X = \{-1; 1\}$), toutes deux de loi uniforme $P_X(-1) = p_{-1} = P_X(+1) = p_{+1} = \frac{1}{2}$. A partir des symboles X_1 et X_2, nous allons former et étudier les propriétés informatives de deux types de mots :

Mots de type 1. $\mathbf{X} = (X_1, X_2)$,

Mots de type 2. $\mathbf{S} = (S_+, S_-)$ avec $S_+ = X_1 + X_2$ et $S_- = X_1 - X_2$.

Etude des mots de type 1 (couples de 2 symboles). Calculer

1. les entropies $H(X_1)$ et $H(X_2)$,
2. l'entropie conditionnelle $H(X_2|X_1)$,
3. l'entropie conjointe $H(\mathbf{X}) = H(X_1, X_2)$
4. l'information mutuelle $I(X_1; X_2)$.

Etude des mots de type 2 (Somme et différence de 2 symboles) .

1. Caractérisation séparée de S_+ et de S_-.
   (a) Déterminer l'alphabet et le jeu de probabilités de la v.a.. S_+. Préciser pourquoi S_- est de même loi que S_+.
   (b) En déduire l'entropie $H(S_+)$.
   (c) Comparer l'entropie de S_+ à celle de X_1 et à celle du mot $\mathbf{X} = (X_1, X_2)$.
2. Caractérisation de $\mathbf{S} = (S_+, S_-)$.
   (a) Déterminer le tableau des probabilités conditionnelles $Pr(S_-|S_+)$.
   (b) Déterminer le tableau des probabilités conjointes $Pr(S_+, S_-)$.
   (c) Calculer les entropies $H(\mathbf{S}) = H(S_+, S_-)$, $H(S_-|S_+)$ ainsi que $I(S_+; S_-)$.
   (d) Commenter chaque valeur obtenue en c) par rapport à la situation initiale (mot simple $\mathbf{X} = (X_1, X_2)$).
   (e) Dire pourquoi il est naturel de définir l'efficacité d'une source simple (v.a. X à N états) par $\mathcal{E} = H(X)/\log_2 N$? (et donc la redondance par $R = 1 - \mathcal{E}$).
   Comment peut-on interpréter la quantité négative ou nulle $H(X) - \log_2 N$?
   (f) Au regard de la dimension potentielle $\dim S_+ \times \dim S_-$ (où $\dim S_+$ représente le cardinal de l'alphabet S_+), calculer la redondance du mot $\mathbf{S}$, et comparer à la redondance du mot $\mathbf{X}$.
   Cette redondance peut-elle avoir une utilité ?
   Comparer à la redondance de S_+ seule (ou S_- seule) (une des composantes de $\mathbf{S}$).
3. On considère la source qui génère une suite d'éléments ternaires telle que : $S_+^1 S_-^1 S_+^2 S_-^2 \cdots S_+^n S_-^n \cdots$. Chaque paire $S_+^j S_-^j$ code par leur somme et leur différence 2 valeurs binaires X_1^j, X_2^j d'une suite de v.a. binaires i.i.d. $X_1^1, X_2^1 X_1^2, X_2^2 \cdots$.
   (a) Quelle est la longueur moyenne minimale des mots pour un codage binaire de cette source (codage symbole par symbole, chaque symbole étant de type S_+ ou de manière équivalente S_-)?
   (b) Quelle est la longueur moyenne minimale des mots pour un codage binaire de ses extensions d'ordre 2 ? Commenter le rôle que joue le choix de l'indice du premier symbole codé (début du processus de codage au symbole S_+^1 ou S_-^1)
   (c) En supposant que le codage débute par le symbole S_+^1, est-il utile de coder les extensions d'ordre 4 ?

Lien entre un symbole et la somme (puis entre mot simple et mot composé). .

1. Après avoir indiqué les tableaux de probabilités nécessaires, calculer $H(X_1, S_+)$, $H(X_1|S_+)$ et $I(X_1; S_+)$.
2. Indiquer sans faire le calcul quelles entropies il faudrait tester et leurs valeurs attendues pour montrer que la connaissance du mot somme/différence $\mathbf{S}$ est équivalente à la connaissance du mot simple $\mathbf{X}$?

Liens entre 3 v.a. ou plus. .

1. Les v.a. X_1 et X_2 sont elles dépendantes sachant leur somme S_+ ? Donner le tableau des probabilités conditionnelles $Pr(X_1, X_2|S_+)$ et argumenter.
2. En déduire (sans calcul) si l'information mutuelle conditionnelle $I(X_1; X_2|S_+)$ doit est supérieure, inférieure ou égale à l'information mutuelle simple $I(X_1, X_2)$.
3. Calculer $H(X_1, X_2|S_+)$, comparer à $H(X_1|S_+)$ et commenter.
4. Calculer $H(X_1, X_2, S_+)$, à partir de la règle de chaînage de 2 v.a..
   Que vaut $H(S_+|X_1, X_2)$?
   Vérifier vos résultats en développant aussi $H(X_1, X_2, S_+)$ à partir de la règle de chaînage de 3 v.a..

Etude des mots de type 1 (couples de 2 symboles). Calculer

1. $H(X_1) = H(X_2) = 1$ bit : chacune des v.a. prise seule apporte 1 bit d'information : elle suit une loi uniforme binaire.
2. $H(X_2|X_1) = 1$: X_1 est indépendante de X_2 donc connaître X_1 ne réduit pas l'incertitude sur X_2, $H(X_2|X_1) = H(X_2) = 1$ bit.
3. $H(\mathbf{X}) = H(X_1) + H(X_2) = 2$ bits : les v.a. X_1 et X_2 sont indépendantes, elles ne partagent pas d'information, l'information apportée par le couple (X_1, X_2) est la somme des informations apportées par X_1 et X_2,
4. $I(X_1; X_2) = 0$: il n'y a aucun partage d'information entre X_1 et X_2 en raison de leur indépendance, $I(X_1; X_2) = H(X_2) - H(X_2|X_1) = H(X_2) - H(X_2) = 0$.

Etude des mots de type 2 (Somme et différence de 2 symboles) .

1. Caractérisation séparée de S_+ et de S_-.
   (a) S_+ et S_- ont le même alphabet que nous notons A_S. $S_+ \in A_S = \{-2; 0; +2\}$ avec les probabilités respectives $P_{S_+} = \{1/4; 1/2; 1/4\}$ puisque

$$\begin{aligned} p_{S_+}(-2) &= p_X(-1)p_X(-1) = \frac{1}{4} \\ p_{S_+}(0) &= p_X(+1)p_X(-1) + p_X(-1)p_X(+1) = \frac{1}{2} \\ p_{S_+}(+2) &= p_X(+1)p_X(+1) = \frac{1}{4} \end{aligned}$$

   (b) L'entropie de S_+ vaut $H(S_+) = -2 \times \frac{1}{4}\log_2(1/4) - \frac{1}{2}\log_2(1/2) = 1.5$ bits.
   (c) Quantité d'information (moyenne) de S_+ supérieure à celle du symbole X_1 ($H(S_+) = 1,5 > H(X_1) = 1$) mais inférieure à celle du mot simple $\mathbf{X}$ de 2 symboles ($H(S_+) \leq H(\mathbf{X}) = 2$).
   On a besoin de la deuxième combinaison (différence) pour avoir des informations équivalentes au mot simple $\mathbf{X}$.
2. Caractérisation de $\mathbf{S} = (S_+, S_-)$.
   (a) Tableaux de probabilités conditionnelles $Pr(S_-|S_+)$.

$P(S_-\|S_+)$	$S_- = -2$	$S_- = 0$	$S_- = +2$
$S_+ = -2$	0	1	0
$S_+ = 0$	1/2	0	1/2
$S_+ = +2$	0	1	0

(b) Tableaux de probabilités conjointes $Pr(S_+, S_-)$:

$P(S_+, S_-)$	$S_- = -2$	$S_- = 0$	$S_- = +2$
$S_+ = -2$	0	1/4	0
$S_+ = 0$	1/4	0	1/4
$S_+ = +2$	0	1/4	0

(c) $H(S_-, S_+) = 2$ bits : même quantité d'info que $\mathbf{X}$.
$H(S_-|S_+) = -1/4 \log_2(1/2) - 1/4 \log_2(1/2) = 0.5$ bit.
$I(S_+; S_-) = H(S_-) - H(S_-|S_+) = 1.5 - 0.5 = 1$

(d) Cette fois les 2 symboles sont dépendants, contrairement au cas précédent.
$I(S_+; S_-) = H(S_-) - H(S_-|S_+) = 1.5 - 0.5 = 1$
Chaque symbole (somme ou différence) porte 1,5 bits, mais ils partagent 1 bit.

(e) L'entropie d'une source à N états vérifie : $0 \leq H(X) \leq \log_2 N$, c'est-à-dire

$$0 \leq \frac{H(X)}{\log_2 N} \leq 1$$

Plus ce ratio est proche de 1, plus la source est proche de la source la plus efficace qui apporte $\log_2 N$ bits d'information.
La divergence de Kullback entre la loi de X $(p_j, j = 1 \cdots N)$ et la loi uniforme $(u_j = 1/N, i = 1 \cdots N)$ s'écrit :

$$D(P||U) = \sum_{j=1}^{N} p_j \log \frac{p_j}{1/N} = \log_2(N) - H(X) \geq 0$$

soit $H(X) \leq \log_2 N$

(f) Redondance : on a $\dim S_+ \times \dim S_- = 9$ mots possibles si chaque lettre utilise un alphabet de taille 3. D'où la redondance
$R(\mathbf{S}) = 1 - H(\mathbf{S})/\log_2 9 = 1 - 2/\log_2(3^2) = 1 - 1/\log_2 3 \approx 36.9$ %.
$\mathbf{S}$ est fortement redondante alors que la redondance du mot $\mathbf{X}$ était nulle.
La redondance apporte souvent une certaine forme de robustesse : lorsqu'un symbole est perdu toute, ou une partie, de l'information qu'il porte peut être retrouvée dans d'autres symboles.
Le calcul de la redondance pour S_+ seul donne :
$R(S_+) = 1 - H(S_+)/\log_2 3 = 1 - 1.5/\log_2(3) \approx 5$ %.
La redondance du couple $\mathbf{S} = (S_+, S_-)$ est importante alors que ses composantes (marginales du couple $\mathbf{S}$) S_+ et S_- sont faiblement redondantes : la redondance provient plus de la dépendance entre les v.a. S_+ et S_- que de la non uniformité des marginales S_+ et S_-.

3. La v.a. $\mathbf{S}$ étant redondante, une suite de réalisations de $\mathbf{S}$ est compressible.
S_+ et S_- ont même loi, $S_+^1 S_-^1 \cdots S_+^n S_-^n$ est une suite de v.a. de même loi. La longueur moyenne minimale des mots pour un codage binaire est $\nu = H(S_+) = 1.5$
Les symboles de l'extension d'ordre deux sont de deux formes possibles :

(a) Les symboles $(S_+^k S_-^k)$ sont indépendants entre eux (pour différents k) : l'extension d'ordre deux est une source simple. Chaque symbole est composé de deux sous-symboles dépendants d'entropie conjointe 2 bits, une compression optimale de cette source simple est possible.

(b) Les symboles $S_-^k S_+^{k+1}$ sont dépendants entre eux, l'extension d'ordre deux n'est pas une source simple. Chaque symbole est composé de deux sous-symboles indépendants d'entropie conjointe 3 bits. La compression n'est pas optimale dans ce cas.

(c) Le codage des extensions d'ordre 2, même si le processus de codage débute à la position optimale, bute sur le caractère non entier des longueurs optimales des mots, le codage des extensions d'ordre 4 limite cet effet. Ici les probabilités sont des puissances entières de 1/2, cette extension est inutile. On peut aussi considérer la convolution de la suite binaire i.i.d. par la réponse $\delta_0 + \delta_1$. Dans ce cas la suite ternaire n'est pas séparable, augmenter la taille de l'extension améliore les performances. Idem avec une réponse du type $\delta_0 + \frac{1}{2}\delta_1$ pour laquelle la suite résultante est quaternaire.

Lien entre un symbole et la somme (puis entre mot simple et mot composé).

1. Probabilités conditionnelles et conjointes

$P(X_1\|S_+)$	$X_1=-1$	$X_1=+1$
$S_+=-2$	1	0
$S_+=0$	1/2	1/2
$S_+=+2$	0	1

et

$P(X_1,S_+)$	$X_1=-1$	$X_1=+1$
$S_+=-2$	1/4	0
$S_+=0$	1/4	1/4
$S_+=+2$	0	1/4

D'où :
$H(X_1,S_+)=H\{1/4,1/4,1/4,1/4\}=2$ bits.
$H(X_1|S_+)=-1/4\log_2(1/2)-1/4\log_2(1/2)=0.5$ bit $>$ incertitude sur le symbole sachant la somme,
$I(X_1,S_+)=H(X_1)-H(X_1|S_+)=1-0.5=0.5$ bit $>$ information partagée par la somme et le symbole élémentaire.

2. Pour montrer que la connaissance du mot somme/différence $\mathbf{S}$ est équivalente à celle du mot simple $\mathbf{X}$, on pourrait vérifier que $H(\mathbf{X}|\mathbf{S})=H(\mathbf{S}|\mathbf{X})=0$ ou encore que $I(\mathbf{X},\mathbf{S})=H(\mathbf{X})=H(\mathbf{S})=2$.

Liens entre 3 v.a. ou plus. .

1. Alors que les v.a. X_1 et X_2 sont indépendantes à l'origine, elles ne le sont plus sachant la somme S_+, puisque $X_1=S_+-X_2$.
   En témoigne aussi le tableau de Probabilités conditionnelles $Pr(X_1,X_2|S_+)$:

$Pr(X_1,X_2\|S_+)$	$(-1,-1)$	$(-1,+1)$	$(+1,-1)$	$(+1,+1)$
-2	1	0	0	0
0	0	1/2	1/2	0
$+2$	0	0	0	1

   Si on s'intéresse à la ligne sachant $S_+=0$, on voit que $Pr(X_1=-1,X_2=+1|S_+=0)=1/2\neq[Pr(X_1=-1|S_+=0)=\frac{1}{2}]\times[Pr(X_2=+1|S_+=0)]=\frac{1}{2}\times\frac{1}{2}=\frac{1}{4}$,
   Ce qui confirme bien la non indépendance entre X_1 et X_2 lorsque S_+ est connue.
2. $I(X_1,X_2|S_+)>0$ (non indépendance conditionnelle) alors que $I(X_1,X_2)=0$.
3. D'après le tableau précédent $Pr(X_1,X_2|S_+)$ et $P(S_+)=\{1/4;1/2;1/4\}$ d'où $H(X_1,X_2|S_+)=-1/4\log_2(1/2)-1/4\log_2(1/2)=0.5$ bit. L'incertitude sur le mot simple $\mathbf{X}$ sachant la somme est la même que l'incertitude d'un seul symbole sachant la somme. Ceci tient de la dépendance conditionnelle $H(X_1,X_2|S_+)=H(X_1|S_+)$.
   Ou encore $H(X_1,X_2|S_+)=H(X_1|S_+)+H(X_2|X_1,S_+)$ mais $H(X_2|X_1,S_+)=0$.
4. $H(X_1,X_2,S_+)=H(S_+)+H(X_1,X_2|S_+)=1.5+0.5=2$ bits.
   Ainsi $H(S_+|X_1,X_2)=H(X_1,X_2,S_+)-H(X_1,X_2)=0$ bit aucune incertitude sur la somme si on connaît les 2 symboles de départ (ce qui aurait pu aussi se déterminer directement à partir du tableau des probabilités $H(S_+|X_1,X_2)$ qui a seulement 4 valeurs non-nulles égales à 1.
   A partir de la règle de chaînage de 3 v.a. on vérifie bien $H(X_1,X_2,S_+)=H(X_1)+H(X_2|X_1)+H(S_+|X_1,X_2)=1+1+0=2$ bit, ou bien encore $=H(S_+)+H(X_1|S_+)+H(X_2|X_1,S_+)=1.5+0.5+0=2$ bit.

Chapter 3

Algorithmes de compression

On admet le résultat suivant : si un code instantané est optimal alors il est également optimal dans la classe (plus grande) des codes déchiffrables. C'est la raison pour laquelle nous nous restreignons aux seuls codes instantanés.

Les algorithmes sont présentés par défaut dans le cas binaire ($D = 2$), mais peuvent être généralisés au cas $D > 2$ ($D = 3$, code ternaire par exemple).

3.1 Conditions nécessaires d'optimalité

Avant de présenter les deux codes classiques que sont le code de Huffman et le code de Fano-Shannon, donnons quelques conditions nécessaires d'optimalité.

Soit C un code dont les mots ont les longueurs l_i avec les probabilités $p_1 \geq \cdots \geq p_N$ (Lorsque plusieurs mots ont la même probabilité, on les classe par ordre de taille croissante). Les conditions suivantes sont nécessaires pour que le code soit optimal :

1. $p_j > p_k \rightarrow l_j \leq l_k$.

   Si cette condition n'est pas vérifiée, il suffit de permuter les mots l_j et l_k pour obtenir un code plus performant et C ne serait pas optimal.

   Preuve. Si C est un code optimal (de compacité ν) et C' le code (de compacité ν') obtenu en permutant les mots j et k, on a :

$$\begin{aligned} l'_i &= l_i \text{ pour } i \neq j, i \neq k \\ l'_j &= l_k \\ l'_k &= l_j \end{aligned}$$

   d'où

$$\begin{aligned} \nu' - \nu &= \left[\sum p_i l'_i\right] - \left[\sum p_i l_i\right] \\ &= \left[p_j l'_j + p_k l'_k + \sum_{i \neq j, i \neq k} p_i l_i\right] - \left[p_j l_j + p_k l_k + \sum_{i \neq j, i \neq k} p_i l_i\right] \\ &= \left[p_j l'_j + p_k l'_k\right] - \left[p_j l_j + p_k l_k\right] \\ &= \left[p_j l_k + p_k l_j\right] - \left[p_j l_j + p_k l_k\right] \\ &= \left(p_j - p_k\right)\left(l_k - l_j\right) \end{aligned}$$

   D'après les deux hypothèses $p_j > p_k$ ($p_j - p_k > 0$) et C optimal ($\nu' - \nu \geq 0$), $\nu' - \nu \geq 0$ on a $l_k \geq l_j$ $\square$

2. $l_N = l_{N-1}$: les deux mots les plus longs ont la même longueur.

   C'est une conséquence directe de la construction d'un code préfixé dans la preuve de l'inégalité de Kraft.

   D'après le résultat précédent, les probabilités étant classées par ordre décroissant $p_1 \geq \cdots \geq p_N$, les longueurs des mots le sont par ordre croissant $l_1 \leq \cdots \leq l_N$ et les longueurs des deux plus longs mots sont l_{N-1} et l_N.

   Par construction, il reste au moins un noeud non utilisé à la profondeur l_{N-1} (sinon il ne serait pas possible de construire des mots plus longs) et ce noeud est le préfixe de mots de longueur l_N. Mais ce mot de longueur l_N étant le dernier à construire, il est inutile (et néfaste) de choisir $l_N > l_{N-1}$, ce noeud est un meilleur choix, et pour ce choix $l_N = l_{N-1}$.

3. Parmi les mots de longueur l_N, au moins deux ne diffèrent que par le dernier caractère puisque s'il n'en est pas ainsi, la suppression de celui-ci, conduit à un code plus performant.

   Conséquence directe du point précédent.

3.2 Code optimal de Huffman et code de Fano-Shannon

Deux codes classiques :

Le code de Fano-Shannon. Très naturelle, la construction du code de Fanon-Shannon reprend l'idée déjà utilisée pour la construction de bons questionnaires. La loi uniforme maximise l'entropie, pour déterminer laquelle des N réalisations possibles de la source s'est effectivement produite, imaginons que l'on pose des questions binaires (deux réponses possibles). Le meilleur choix, pour une question donnée, consiste à équilibrer la probabilité des deux réponses possibles, ce choix maximise l'information. Telle est l'idée du codage de Fano-Shannon : diviser l'ensemble des réalisations possibles de la source en deux sous-ensembles de probabilités aussi voisines que possible puis renouveler l'opération sur chacun des sous-ensembles.

Si cette manière de procéder est bien localement optimale, l'enchaînement de ces traitements localement optimaux n'est pour autant pas globalement optimal.

Ceci se comprend du fait qu'une très bonne question (même probabilité pour les réponses oui et non ou, en termes de codage, même probabilité pour chacun des deux symboles possibles) peut impliquer dans les étapes ultérieures des questions très sous optimales. Dans ce cas, il peut être préférable de relaxer la contrainte sur l'optimalité de cette question particulière pour perdre moins ensuite et globalement gagner en longueur moyenne.

Le code de Huffmann. L'idée du code de Huffmann consiste à grouper les deux événements les moins probables en un unique événement et à renouveler l'opération avec le nouvel ensemble d'événements ainsi obtenu.

L'algorithme de Huffman (1952) est un algorithme glouton, c'est-à-dire un algorithme qui enchaîne des procédures localement optimales en vue d'un résultat global optimal. Cet algorithme construit un code instantané optimal. En pratique, il conduit à des réductions de longueur de l'ordre de 20 à 90% et il peut être associé à d'autres formes de codage.

Il modélise le code par une forêt composée d'arbres qui possèdent des noeuds dont le poids est la somme des poids de leurs enfants (le poids d'un arbre est celui de sa racine).

Initialement, les arbres (à 1 seul noeud) sont les états de la source, le poids de chacun de ces arbres est la probabilité de l'état associé.

A chaque étape, l'algorithme groupe deux des arbres de plus faible poids en un arbre unique dont ils sont les enfants.

Au final, l'algorithme produit un arbre unique de poids 1.

Soit X une source à N états

$$x_1 = y_1^N \quad x_2 = y_2^N \quad \cdots \quad x_N = y_N^N$$

de probabilités $\pi_i^N = p\left(y_i^N\right)$ classées par ordre décroissant :

$$\pi_1^N \geq \quad \pi_2^N \geq \quad \cdots \geq \quad \pi_N^N$$

On note C^n le code de Huffman d'une source à n états.

Le code de Huffman C^2 d'une source à 2 états (y_1^2 de probabilité π_1^2 et y_2^2 de probabilité π_2^2) est défini par :

$$\begin{array}{rcl} C^2\left(y_1^2\right) & = & 0 \\ C^2\left(y_2^2\right) & = & 1 \end{array}$$

C^2 est optimal pour une source à 2 états.

Supposons que C^{n-1} est un code de Huffman binaire pour

$$\begin{array}{ccccc} y_1^{n-1} & y_2^{n-1} & & \cdots & y_{n-1}^{n-1} \\ \pi_1^{n-1} \geq & \pi_2^{n-1} \geq & \cdots \geq & & \pi_{n-1}^{n-1} \end{array}$$

alors C^n le code de Huffman à n états est construit comme :

$$\begin{array}{rcl} C^n\left(y_1^n\right) & = & C^{n-1}\left(y_1^{n-1}\right) \\ \vdots & \vdots & \vdots \\ C^n\left(y_{n-2}^n\right) & = & C^{n-1}\left(y_{n-2}^{n-1}\right) \\ C^n\left(y_{n-1}^n\right) & = & C^{n-1}\left(y_{n-1}^{n-1}\right)||0 \\ C^n\left(y_n^n\right) & = & C^{n-1}\left(y_{n-1}^{n-1}\right)||1 \end{array}$$

Les $n-1$ états de C^{n-1} sont obtenus en groupant les deux états de plus faible probabilité parmi les n états de C^n.

Non optimalité du code Fano-Shannon, un contre-exemple.

Considérons une source X à 4 états de probabilités $\pi_1^4 = 0.45, \pi_2^4 = 0.4, \pi_3^4 = 0.1, \pi_4^4 = 0.05$.

L'entropie de la source vaut :

$$\begin{array}{rcl} H(X) & = & -\sum_{i=1}^{4} \pi_i^4 \log_2 \pi_i^4 \\ & = & -0.45 \log_2 0.45 - 0.4 \log_2 0.4 - 0.1 \log_2 0.1 - 0.05 \log_2 0.05 \\ & = & 1.6 \text{ bit} \end{array}$$

Aucun code ne peut avoir une longueur moyenne inférieure à 1.6 caractères par mot. Nous allons vérifier sur cet exemple que le code de Fano-Shannon est sous-optimal.

Les longueurs optimales théoriques valent :

$$\begin{array}{rcl} l_1^* & = & 1.15 \\ l_2^* & = & 1.32 \\ l_3^* & = & 3.32 \\ l_4^* & = & 4.32 \end{array}$$

En arrondissant à l'entier supérieur le jeu de longueurs devient $l_1 = 2, l_2 = 2, l_3 = 4, l_4 = 5$, soit une longueur moyenne de $0.45 \times 2 + 0.4 \times 2 + 0.1 \times 4 + 0.05 \times 5 = 2.35$.

Le code de Fano-Shannon (figure 3.1) n'a que des mots de longueur 2, sa longueur moyenne vaut $\nu_{FS} = 2$. Il est déjà meilleur qu'un simple arrondi à l'entier supérieur.

Considérons maintenant le code de Huffman (figure 3.2), ses mots sont de longueurs variables, $\nu_H = 0.45 \times 1 + 0.4 \times 2 + 0.1 \times 2 + 0.05 \times 2 = 1.7$: $\nu_H < \nu_{FS}$, le code de Huffman est meilleur que celui de Fano-Shannon.

Ce contre exemple démontre que la procédure de Fano-Shannon n'est pas optimale.

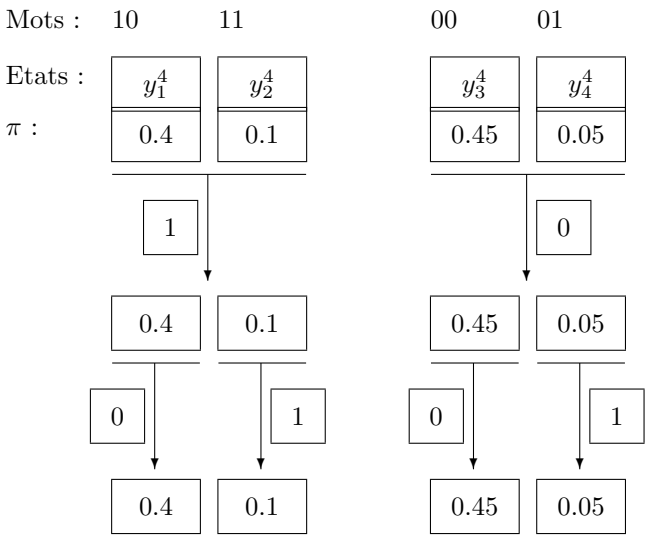


Figure 3.1: Codage de la source de jeu de probabilités $\pi_1^4 = 0.45, \pi_2^4 = 0.4, \pi_3^4 = 0.1, \pi_4^4 = 0.05$. Le code de Fano-Shannon est sous optimal, sa longueur moyenne vaut $\nu_{FS} = 2$.

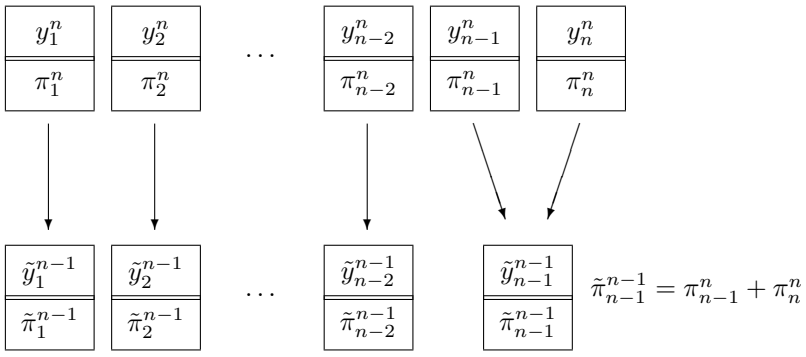
Optimalité du code de Huffman.

En cours, on montre que la réduction de Huffman, qui permet de passer de n à $n-1$ états, correspond à une variation de longueur moyenne égale à la somme des probabilités des deux états fusionnés. Ainsi, le groupement des deux états les moins probables donne la variation de longueur la plus faible possible.

On montre également que l'optimalité au rang $n-1$ équivaut à l'optimalité au rang n et que le code optimal pour deux états consiste à coder l'un d'eux par le mot 0 et l'autre par 1.

Ces résultats démontrent l'optimalité du codage de Huffman.

Réduction de Huffman. Considérons la réduction de Huffman, passage du rang n au rang $n-1$ par fusion des deux états les moins probables :



Calculons la variation de longueur moyenne lors de la réduction. Pour $i = 1 \cdots n-2$, on a $l\left(y_i^n\right) = l\left(\tilde{y}_i^{n-1}\right)$ avec $\pi_i^n = \tilde{\pi}_i^{n-1}$ et pour $i = n-1, n$, $l\left(y_{n-1}^n\right) = l\left(y_n^n\right) = l\left(\tilde{y}_{n-1}^{n-1}\right) + 1$.

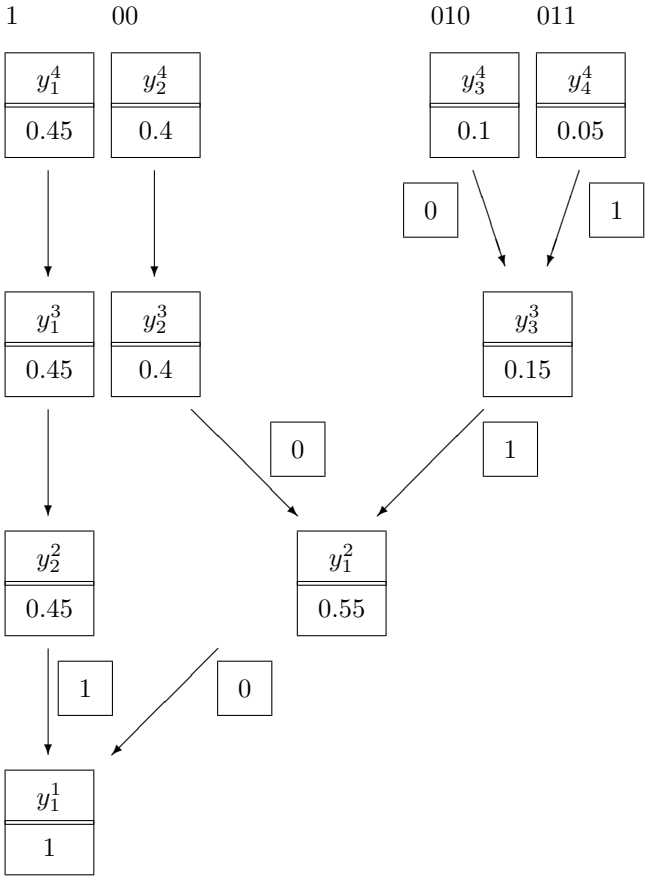


Figure 3.2: Codage de la source de jeu de probabilités $\pi_1^4 = 0.45, \pi_2^4 = 0.4, \pi_3^4 = 0.1, \pi_4^4 = 0.05$. Le code de Huffman est optimal, sa longueur moyenne vaut $\nu_H = 0.45 \times 1 + 0.4 \times 2 + 0.1 \times 2 + 0.05 \times 2 = 1.7$.

D'où la différence de longueur moyenne entre la profondeur n et la profondeur $n-1$:

$$\begin{aligned}
\nu_n - \nu_{n-1} &= \left[\sum_{i=1}^{n-2} \pi_i^n l\left(y_i^n\right) + \pi_{n-1}^n l\left(y_{n-1}^n\right) + \pi_n^n l\left(y_n^n\right)\right] \\
&- \left[\sum_{i=1}^{n-2} \pi_i^n l\left(y_i^n\right) + \left[\pi_{n-1}^n + \pi_n^n\right]\left\{l\left(y_{n-1}^n\right) - 1\right\}\right] \\
&= \pi_{n-1}^n + \pi_n^n
\end{aligned}$$

En groupant les deux noeuds de plus faibles probabilités, la réduction de Huffman est le choix qui augmente le moins la longueur moyenne entre la profondeur $n-1$ et la profondeur n.

L'optimalité à l'odre de n équivaut à l'optimalité à l'ordre $n-1$. Appelons $C^{*(n)}$ un code optimal canonique (probabilités classées par ordre décroissant) pour le jeu de probabilités à la profondeur n (de longueur moyenne $\nu^{*(n)}$) et $C^{*(n-1)}$ un code optimal pour le jeu de probabilités à la profondeur $n-1$ (de longueur moyenne $\nu^{*(n-1)}$).

On construit le code de profondeur n à partir du code de profondeur $n-1$ en créant 2 mots par prolongement par 0 ou 1 du mot de probabilité la plus faible ($\pi_{n-1}^{n-1} = \pi_{n-1}^{n} + \pi_{n}^{n}$). La longueur moyenne $\nu^{(n)}$ du code obtenu vaut

$$\nu^{(n)} = \nu^{*(n-1)} + \pi_{n-1}^{n} + \pi_{n}^{n}$$

Inversement, le code à la profondeur $n-1$ obtenu par fusion des 2 états les moins probables à la profondeur n a pour longueur moyenne :

$$\nu^{(n-1)} = \nu^{*(n)} - \left[\pi_{n-1}^{n} + \pi_{n}^{n}\right]$$

En sommant ces deux égalités, on a

$$\nu^{(n)} + \nu^{(n-1)} = \nu^{*(n-1)} + \nu^{*(n)}$$

soit

$$\underbrace{\left[\nu^{(n)} - \nu^{*(n)}\right]}_{\geq 0} + \underbrace{\left[\nu^{(n-1)} - \nu^{*(n-1)}\right]}_{\geq 0} = 0$$

Les termes entre crochets étant positifs (les codes optimaux ne peuvent pas être plus longs que les autres), ils doivent être nuls.

Ainsi l'optimalité du code à la profondeur $n-1$ ($\nu^{(n-1)} = \nu^{*(n-1)}$) implique celle du code à la profondeur n ($\nu^{(n)} = \nu^{*(n)}$) (et inversement l'optimalité à la profondeur n implique celle à la profondeur $n-1$)

Code optimal pour une source à 2 états. A la profondeur $n = 2$, on fusionne 2 états de probabilités π_1^2 et π_2^2 en un seul de probabilité $\pi_1^2 + \pi_2^2 = 1$. Un code $C^{*(2)}$ optimal pour ces 2 états (y_1^2 de probabilité π_1^2 et y_2^2 de probabilité π_2^2) est donné par :

$$\begin{aligned} C^{*(2)}\left(y_1^2\right) &= 0 \\ C^{*(2)}\left(y_2^2\right) &= 1 \end{aligned}$$

$C^{*(2)}$ est le code de Huffman pour 2 états.

Code optimal pour n états. Finalement, en partant du code de Huffman optimal $C^{*(2)}$, on construit récursivement un code optimal au rang n.

3.3 Autres types de codes — code arithmétique

Ce paragraphe présente une autre méthode de compression : le codeur arithmétique.

On considère $x_{i_1} \cdots x_{i_n}$ (avec $i_j \in \{1, \cdots, N\}$), la séquence de longueur n à coder .

Le codeur arithmétique procède de la manière suivante.

Intervalle initial. On réalise tout d'abord une partition de l'intervalle $[0; 1)$ en N sous intervalles $I_j = [\alpha_j; \beta_j)$ de longueurs p_j, $j = 1 \cdots N$.

Intervalle courant. L'algorithme procède à un codage par intervalle au cours duquel l'intervalle initial est réduit en fonction de la suite des symboles à coder.

L'intervalle courant $[a_k; b_k)$ est de longueur notée $w_k = b_k - a_k$.

Initialement $a_0 = 0$, $b_0 = 1$ et $w_0 = 1$.

Codeur. Encoder le $(k+1)$ème caractère $x_{i_{k+1}}$ ($k \geq 0$) (associé à l'intervalle $I_{i_{k+1}} = [\alpha_{i_{k+1}}; \beta_{i_{k+1}})$) se fait par réduction de l'intervalle courant $C_k = [a_k; b_k)$ en $[a_{k+1}; b_{k+1})$ selon :

$$\begin{aligned} a_{k+1} &= a_k + w_k\, \alpha_{i_{k+1}} && (3.1) \\ b_{k+1} &= a_k + w_k\, \beta_{i_{k+1}} && (3.2) \\ & && (3.3) \end{aligned}$$

Décodeur. Les intervalles sont associés de manière bijective aux suites de caractères. Le nombre n_1 qui code le message complet appartient à un intervalle qui détermine le premier caractère de ce message. Pour retrourver la suite des caractères qui composent le message, le décodeur procède de la manière suivante :

$$n_1 = \text{le nombre sorti du codeur} \tag{3.4}$$

$$n_{k+1} = \frac{n_k - \alpha_{i_k}}{p_{i_k}} \tag{3.5}$$

$$\tag{3.6}$$

A chaque itération, le nombre n_{k+1} appartient à un intervalle qui détermine le caractère $k+1$.

Exemple de codeur/décodeur

On considère une source à $N = 3$ états notés x_1 ='A', x_2 ='M', x_3 ='I' de probabilités respectives $p_1 = 0.44$, $p_2 = 0.16$ et $p_3 = 0.4$.

On veut procéder au codage arithmétique de la séquence, de longueur $n = 3$, $x_{i_1}x_{i_2}x_{i_3}$='MAI' ($i_1 = 2$, $i_2 = 1$, $i_3 = 3$) issue de cette source.

Codage

Initialisation. $a_0 = 0$, $b_0 = 1$ et $w_0 = 1$.

La partition initiale de $[0; 1)$ est

$I_1 = [\alpha_1 = 0; \beta_1 = 0.44)$, $I_2 = [\alpha_2 = 0.44; \beta_2 = 0.6)$, $I_3 = [\alpha_3 = 0.6; \beta_3 = 1)$.

Codage du caractère numéro 1 **(**$k = 0$**,** $i_1 = 2$**)**

$$\begin{aligned} a_1 = a_0 + w_0\,\alpha_{i_1} &= 0 + 1 \times 0.44 = 0.44 \\ b_1 = a_0 + w_0\,\beta_{i_1} &= 0 + 1 \times 0.6 = 0.6 \end{aligned}$$

L'intervalle qui code la séquence d'un caractère 'M' est $C_1 = [0.44; 0.6]$ de longueur $w_1 = 0.16$.

Codage du caractère numéro 2 **(**$k = 1$**,** $i_2 = 1$**)**

$$\begin{aligned} a_2 = a_1 + w_1\,\alpha_{i_2} &= 0.44 + 0.16 \times 0 = 0.44 \\ b_2 = a_1 + w_1\,\beta_{i_2} &= 0.44 + 0.16 \times 0.44 = 0.5104 \end{aligned}$$

L'intervalle qui code la séquence 'MA' est $C_2 = [0.44; 0.5104]$ de longueur $w_2 = 0.0704$.

Codage du caractère numéro 3 **(**$k = 2$**,** $i_3 = 3$**)**

$$\begin{aligned} a_3 = a_2 + w_2\,\alpha_{i_3} &= 0.44 + 0.0704 \times 0.6 = 0.4822 \\ b_3 = a_2 + w_2\,\beta_{i_3} &= 0.44 + 0.0704 \times 1 = 0.5104 \end{aligned}$$

L'intervalle qui code la séquence 'MAI' est $C_3 = [0.4822; 0.5104]$ de longueur $w_3 = 0.0282$.

Finalement, le codage de la séquence 'MAI' est tout nombre de l'intervalle $C_3 = [0.4822; 0.5104]$, prenons $n = 0.5$.

Décodage

Partant du nombre $n = 0.5$, les étapes du décodage sont :

Initialisation. $n_1 = 0.5$.

Décodage du caractère numéro 1 $(k = 0)$.

$n_1 = 0.5 \in I_2$: le premier caractère appartient à l'intervalle du 'M' $(i_1 = 2)$ de probabilité $p_{i_1} = p_2 = 0.16$.

Décodage du caractère numéro 2 $(k = 1)$.

$$n_2 = \frac{n_1 - \alpha_{i_1}}{p_{i_1}} = (0.5 - 0.44)/0.16 = 0.375$$

$n_2 = 0.375 \in I_1$: le deuxième caractère appartient à l'intervalle du 'A' $(i_2 = 1)$ de probabilité $p_{i_2} = 0.44$.

Décodage du caractère numéro 3 $(k = 2)$.

$$n_3 = \frac{n_2 - \alpha_{i_2}}{p_{i_2}} = (0.375 - 0)/0.44 = 0.85$$

$n_3 = 0.85 \in I_3$: le dernier caractère appartient à l'intervalle du 'I' $(i_3 = 3)$ de probabilité $p_{i_3} = 0.4$.

Remarque sur la mise en œuvre du codage arithmétique

En pratique, utiliser des nombre en virgule flottante peut poser problème : il est préférable de représenter l'intervalle initial sous la forme $0, 1, \cdots, N_{\max}$ et de ne manipuler que des entiers. Dans les deux cas, l'algorithme bute sur le caractère fini de la représentation et un nombre ne peut coder qu'une suite de longueur finie L qu'il est nécessaire de déterminer avant de procéder au codage par bloc de longueur L des messages.

3.4 Aspects pratiques du codage entropique

En pratique, plusieurs solutions sont possibles :

Loi connue a priori. Si le codeur et le décodeur disposent a priori du jeu de probabilité de la source et si les fréquences empiriques (les probabilités estimées à partir du fichier lui-même) coïncident avec ces probabilités a priori utilisées pour le codage et le décodage, la procédure de codage est optimale. En général bien sûr, cela n'est pas le cas et le déajustement entre la loi a priori et la loi empirique induit une sous optimalité dont le coût par caractère peut être évalué : il est donné par la divergence de Kullback entre les deux lois.

Loi inconnue a priori. Pour éviter ce problème de désajustement entre la loi empirique et celle qui est utilisée par le codeur et le décodeur, il est possible d'utiliser les fréquences empiriques pour le codage, mais alors, cette loi doit être transmise au décodeur en supplément du message compressé lui-même.

Pour choisir entre ces deux solutions, il faut comparer ces deux pénalités. Typiquement, pour des messages courts, le surcoût lié à la transmission au décodeur de la loi empirique ne compense en général pas le gain de codage alors que pour des messages longs la pénalité due au déajustement l'emporte et embarquer le jeu des probabilités empiriques avec le message devient une meilleure option.

3.5 Suites typiques

Ce paragraphe donne une idée intuitive de la raison pour laquelle la non maximalité de l'entropie d'une source simple permet de réduire l'ensemble des messages pour lesquels une compression est nécessaire.

La possibilité de réécrire de manière plus compacte les messages délivrés par une source simple (c'est-à-dire qui génère des v.a. i.i.d.) peut être comprise intuitivement grâce à la notion de suite typique : lorsque l'entropie

d'une source n'est pas maximale seule une faible partie de l'ensemble des suites possibles se réalisent avec une probabilité significative. Dans la limite des longues suites, l'ensemble des messages peut ainsi être divisé en deux sous-ensembles : l'un comprend les suites typiques qui apparaissent toutes avec la même probabilité et l'autre des suites dont la probabilité d'occurrence est négligeable. Il est ainsi possible de n'associer des étiquettes (mots de code) qu'aux seules suites typiques, leur nombre étant très inférieur au cardinal de l'ensemble de toutes les suites possibles, l'écriture des messages est plus compacte que celle qui résulte d'un étiquetage indifférencié de toutes les suites possibles.

Une suite s_n de n symboles est un ensemble ordonné de n réalisations indépendantes d'une v.a. X pouvant prendre N états $x_1, \cdots, x_N$ avec les probabilités $p_i = p(x_i)$.

Soit $\varepsilon > 0$ et k tels que $1/k^2 < \varepsilon/N$. Une suite est dite typique si elle vérifie la condition suivante pour tous ses symboles :

$$\frac{|f_i(s_n) - np_i|}{\sqrt{np_i(1-p_i)}} < k, \forall i \in \{1, \cdots, N\}$$

$f_i(s_n)$ représente le nombre d'occurrences du symbole i dans la suite s_n de longueur n, c'est une variable aléatoire binomiale de moyenne np_i et d'écart type $\sqrt{np_i(1-p_i)}$. Une suite est donc dite typique lorsque la fréquence attendue np_i de chacun des symboles i est proche de la fréquence empirique $f_i(s_n)$ effectivement observée dans la suite s_n. Ici, le terme proche signifie simplement que l'écart est de l'ordre de $\sqrt{n}$ alors que le nombre d'éléments de la suite vaut n.

Point 1 : L'ensemble des suites non-typiques est asymptotiquement négligeable.

$$P\left(s_n \notin T\right) = P\left(\frac{\left|f_i\left(s_n\right) - np_i\right|}{\sqrt{np_i\left(1-p_i\right)}} > k \text{ pour au moins un des } x_i\right)$$

En majorant la probabilité de l'union des évènements par la somme de leurs probabilités, on peut écrire :

$$P\left(s_n \notin T\right) \leqslant \sum_{i=1}^{N} P\left(\frac{\left|f_i\left(s_n\right) - np_i\right|}{\sqrt{np_i\left(1-p_i\right)}} > k\right)$$

D'après l'inégalité de Tchebychev, la probabilité pour qu'une v.a. s'écarte de sa moyenne de plus de k fois son écart type est inférieure à $1/k^2$, ainsi :

$$P\left(s_n \notin T\right) \leqslant \sum_{i=1}^{N} \frac{1}{k^2} = \frac{N}{k^2} < \varepsilon$$

La probabilité pour que la suite ne soit pas typique est donc négligeable.

Point 2 : chaque suite typique a une probabilité voisine de 2^{-nH}. Par définition, pour chaque symbole i d'une suite typique s_n, on a :

$$np_i - k\sqrt{np_i\left(1-p_i\right)} \leqslant f_i(s_n) \leqslant np_i + k\sqrt{np_i\left(1-p_i\right)}$$

Posons $A = -k\sum_{i=1}^{N} \log\left(p_i\right)\sqrt{p_i\left(1-p_i\right)}$. En multipliant l'inégalité précédente par $-\log_2 p_i > 0$ et en sommant sur l'ensemble des N symboles possibles :

$$nH - A\sqrt{n} \leqslant -\sum_{i=1}^{N} f_i(s_n)\log(p_i) \leqslant nH + A\sqrt{n}$$

En remarquant maintenant que la probabilité de la suite s_n vaut $p(s_n) = p_1^{f1(s_n)} p_2^{f_2(s_n)} \cdots p_N^{f_N(s_n)}$, on a :

$$-\log P(s_n) = -\sum_{i=1}^{N} f_i(s_n)\log(p_i)$$

Finalement, la probabilité $p(s_n)$ peut être encadrée comme suit :

$$2^{-nH-A\sqrt{n}} \leqslant P(s_n) \leqslant 2^{-nH+A\sqrt{n}}$$

Pour n grand, chaque suite typique s_n possède une probabilité voisine de 2^{-nH} .

Point 3 : Le nombre de suites typiques est de l'ordre de 2^{nH} Notons ν le nombre de suites typiques de longueur n . L'inégalité précédente étant valable pour toute suite typique, une sommation sur l'ensemble des suites typiques permet d'écrire : $\nu 2^{-nH-A\sqrt{n}} \leqslant p(s_n \in T) \leqslant \nu 2^{-nH+A\sqrt{n}}$. Or $1-\varepsilon \leqslant p(s_n \in T) \leqslant 1$. De ces deux inégalités, on tire l'encadrement :

$$(1-\varepsilon)2^{nH-A\sqrt{n}} \leqslant \nu \leqslant 2^{nH+A\sqrt{n}}$$

Pour n grand, le nombre de suites typiques est donc de l'ordre de 2^{nH}.

En résumé, pour des messages longs (n grand), on compte environ 2^{nH} suites typiques qui apparaissent toutes avec une même probabilité (voisine de 2^{-nH}). Les autres suites n'apparaissent pas avec une probabilité suffisamment élevée pour que leur compression soit nécessaire.

Lien entre codage bloc et suites typiques. En codant des extensions d'ordre s de la source, avec s grand, le nombre de suites typiques de longueur s est de l'ordre de $2^{sH(X)} = D^{sH_{D(X)}}$. La probabilité pour qu'une suite soit typique étant de l'ordre de un, on peut ne coder que les suites typiques. Avec un alphabet de D caractères, il faut que la longueur moyenne des mots du code soit égale à $sH_{D(X)}$. Autrement dit, la longueur moyenne des mots par caractère source tend vers $H_D(X)$ lorsque s tend vers l'infini.

3.6 Exercices

Mauvais code source

Lesquels des codes suivants ne peuvent pas être des codes de Huffman ?

1. Code composé des 3 mots $0, 10, 11$.
2. Code composé des 4 mots $00, 01, 10, 110$.
3. Code composé des 2 mots $01, 10$

Mauvais code source

1. Code qui respecte la condition du préfxe, sature Kraft. Potentiellement un code de Huffaman pour peu que les probabilités soient affectées correctement aux mots.
2. Le mot 110 peut être raccourci à 11, ce code n'est pas optimal, il ne peut donc pas être un code de Huffman.
3. Les mots 01 et 10 sont inutilement longs, ils peuvent être raccourcis à 0 et 1, ce code n'est pas optimal, il ne peut donc pas être un code de Huffman.

Soit une source simple (sans mémoire) S sur un alphabet A_S composé de 5 lettres, $A_S = \{s_1, s_2, s_3, s_4, s_5\}$.

1. Soit le code binaire $\{00, 11, 010, 111, 1010\}$:
   (a) Est-il à décodage unique (déchiffrable) ?
   (b) Est-il instantané ?
2. Soit le code binaire $\{00, 11, 010, 011, 101\}$:
   (a) Est-il instantané ?
   (b) Existe-t-il un code binaire instantané plus court ? (Justifier)
3. On s'intéresse maintenant à un codage ternaire de la source. On suppose que le jeu de probabilités des symboles de S est $P_S = \{1/3, 1/3, 1/9, 1/9, 1/9\}$.
   (a) Pour un codage ternaire à mots de longueur fixe l, quel serait le choix de l ?
   (b) Calculer l'entropie et la redondance de la source.
   (c) En déduire la longueur moyenne minimale d'un code source ternaire instantané.
   (d) Pour la source S :
      i. Proposer un code de Huffman (donner l'arbre).
      ii. Calculer sa longueur moyenne et son efficacité. Commenter les résultats obtenus.
      iii. Peut-on en déduire une propriété de la suite des symboles après codage?

Codage de source binaire et ternaire

Soit une source simple (sans mémoire) S sur un alphabet A_S composé de 5 lettres, $A_S = \{s_1, s_2, s_3, s_4, s_5\}$.

1. Soit le code binaire $\{00, 11, 010, 111, 1010\}$:
   (a) Le code binaire $\{00, 11, 010, 111, 1010\}$ n'est pas à décodage unique (exemple : $s_2 s_5$ ou $s_4 s_3$)
   (b) N'étant pas à décodage unique, il n'est donc pas instantané ($C2$ est le préfixe de $C4$ par exemple) : la condition du préfixe est une condition suffisante de déchifrabilité.
2. Le code binaire $\{00, 11, 010, 011, 101\}$ est instantané, mais on peut obtenir un code plus court instantané en raccourcissant $C5$ à 10 au lieu de 101 (avec les autres mots inchangés). Le code résultant est toujours instantané (la condition préfixe est vérifiée).
   Remarque, on sature alors l'inégalité de Kraft Mac-Millan.
3. (a) Code ternaire à longueur fixe, nécessiterait l = 2 symboles.
   (b) Entropie $= 2/3 \log_2 3 + 3/9 \log_2(3^2) = \frac{4}{3} \log_2 3$.
   $H(S) = 2.1133$ bit/lettre.
   Redondance $R(S) = 1 - \frac{H(S)}{\log_2 5} = 1 - 2.1133/2.3219 = 8.98\%$.
   (c) D'après le premier théorème de Shannon:
   $\mathrm{L}_{\min} = H(S)/\log_2(3) = 2.1133/1.585 = 4/3 = 1.333$ symbole/caractère.
   (d) Pour la source S :
      i. $C1 = 0$, $C2 = 1$, $C3 = 20$, $C4 = 21$, $C5 = 22$.

ii. $L = L_{\min} = 4/3$, étant donné que le jeu de probabilité est tel que $\log_3(1/p(x_i))$ sont des entiers qui peuvent donc être choisis (sans arrondi) comme longueurs des mots du code, correspondant aux longueurs d'un code optimum absolu.

iii. On sait ainsi que les symboles $\{0, 1, 2\}$ observés en sortie de codage sont équiprobables et indépendants.

Longueurs des mots d'un code de Huffman

Soit une v.a. X qui prend 4 valeurs $\{1; 2; 3; 4\}$ avec les probabilités $\left\{\frac{1}{3}; \frac{1}{3}; \frac{1}{4}; \frac{1}{12}\right\}$.

1. Donner un code de Huffman pour cette v.a..
2. Montrer qu'il existe deux ensembles de longueurs de mots de code optimaux qui sont : $(1, 2, 3, 3)$ et $(2, 2, 2, 2)$.
3. En conclure qu'il existe des codes optimaux avec des longueurs de mots de code qui dépasse la longueur de code de Shannon $\left\lceil \log_2 \frac{1}{p(x)} \right\rceil$ pour certains symboles.

Longueurs des mots d'un code de Huffman

1.

Code	Symbole	Probabilité			
0	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	1
11	2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	
101	3	$\frac{1}{4}$	$\frac{1}{3}$		
100	4	$\frac{1}{12}$			

2. La compacité est égale à 2.

   Le code 00,01,10,11 a une longueur moyenne (compacité) de 2 également.

   Les deux jeux de longueurs vérifient l'inégalité de Kraft et donnent la même longueur moyenne (2) pour les mots de code. Les deux sont optimaux.

3. Le mot de code 101 associé au symbole 3 (de probabilité $1/4$) est de longueur 3 alors que pour cet état de la v.a. X la longueur de Shannon vaut $\left\lceil \log_2 \left\lceil \frac{1}{1/4} \right\rceil \right\rceil = 2$. Pour un symbole particulier, le mot d'un code de Huffman peut être plus grand que la longueur de Shannon, néanmoins en moyenne le code de Huffman ne peut pas être plus long que celui de Shannon.

Coût d'un codage inapproprié

Soient X une v.a. à 5 états $\{1, 2, 3, 4, 5\}$, deux distributions de probabilité $p(x)$ et $q(x)$ pour X et deux codes, l'un C_p adapté à la loi p, l'autre C_q à la loi q.

Etat	1	2	3	4	5
Loi p	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
Code C_p	0	10	110	1110	1111
Loi q	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$
Code C_q	0	100	101	110	111

1. Calculer $H(p)$, $H(q)$, $D(p||q)$ et $D(q||p)$.

2. Vérifier que la compacité de C_p sous la loi p est égale à l'entropie $H(p)$ et qu'ainsi C_p est optimal pour cette loi.

   Rappeler les deux conditions qui assurent qu'un code est absolument optimal (compacité égale à l'entropie).

   Interpréter sur la représentation en arbre du code le saturation de l'inégalité de Kraft et la caractère préfixé du code.

3. Même question pour le code C_q et la loi q.

4. Si le code C_q est utilisé alors que la distribution de la source est p, quelle est la longueur moyenne des mots de code. De combien dépasse t-on l'entropie de la loi p ?

5. Monter que cette perte est donnée par la divergence de Kullback entre les lois p et q et préciser les conditions particulières qui font qu'il en est ainsi.

Coût d'un codage inapproprié

1. $$H(p) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{8}\log_2\frac{1}{8} - 2\frac{1}{16}\log_2\frac{1}{16} = \frac{15}{16} = 1,875$$

   $$H(q) = -\frac{1}{2}\log_2\frac{1}{2} - 4\frac{1}{8}\log_2\frac{1}{8} = 2$$

   $$D(p||q) = \sum p_i \log_2\frac{p_i}{q_i} = \frac{1}{2}\log_2 1 + \frac{1}{4}\log_2 2 + \frac{1}{8}\log_2 1 + 2\frac{1}{16}\log_2\frac{1}{2} = \frac{1}{8}$$

   $$D(q||p) = \sum q_i \log_2\frac{q_i}{p_i} = \frac{1}{2}\log_2 1 + \frac{1}{8}\log_2\frac{1}{2} + \frac{1}{8}\log_2 1 + 2\frac{1}{8}\log_2 2 = \frac{1}{8}$$

   La divergence de Kullback n'est pas symétrique, en général $D(p||q) \neq D(q||p)$.

2. $$C_p = \frac{1}{2}\times 1 + \frac{1}{4}\times 2 + \frac{1}{8}\times 3 + 2\frac{1}{16}\times 4 = \frac{15}{8} = H(p)$$

   La compacité est égale à l'entropie car i/ l'inégalité de Kraft est saturée, ii/ Les longueurs l_i des mots sont $l_i = -\log_2 p_i$.

   Sur l'arbre qui représente le code, la saturation de Kraft se traduit par le fait que les feuilles sont des mots.

   Rappel : pour un code préfixé, les mots sont des feuilles.

3. $$C_q = \frac{1}{2}\times 1 + 4\frac{1}{8}\times 3 = 2 = H(q)$$

   La compacité est égale à l'entropie pour les mêmes raisons qu'à la question précédente.

4. Calculons la longueur moyenne des mots du code adapté à la loi q lorsque la loi est p :

   $$C_q^p = \frac{1}{2}\times 1 + \left(\frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16}\right)\times 3 = 2$$

   On dépasse de $1/8$ la longueur moyenne du code adapté à la loi p.

5. $1/8 = D(p||q)$ et ce n'est pas un coïncidence car $C_q^p = \sum_i p_i l_i^q$ avec $l_i^q = -\log_2 q_i$ d'où

   $$C_q^p = -\sum_i p_i \log_2 q_i$$

et donc

$$C_q^p - C_p^p = -\sum_i p_i \log_2 \frac{q_i}{p_i} = D(p||q)$$

La divergence de Kullback donne l'excès de longueur lié à l'utilisation d'une loi non adaptée.

Attention : ce résultat n'est vrai que lorsque les codes adaptés à chacune des deux lois sont absolument optimaux (atteignent la borne inférieure — entropie).

Pour la compression de messages longs, cette perte peut-être trop importante ; pour éviter cette perte, il peut alors être préférable de transmettre la loi utilisée.

Chapter 4

Vue d'ensemble d'une chaîne classique de l'information

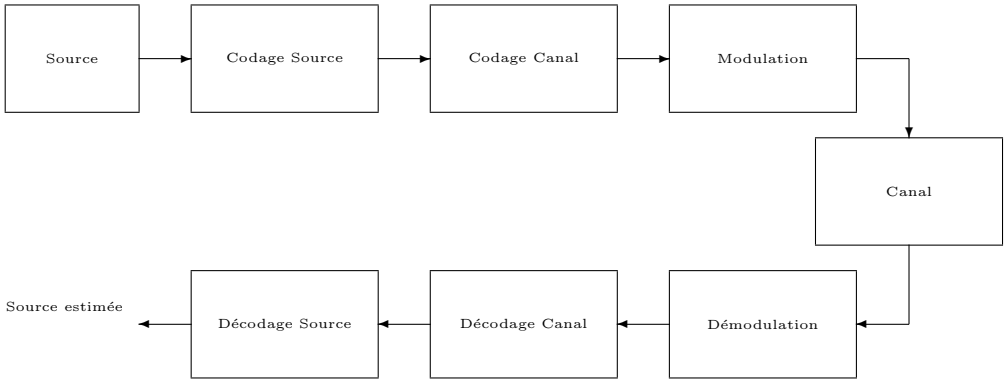


Figure 4.1: Structure générique d'une chaîne de transmission de l'information

4.1 Éléments d'une chaîne de l'information

Une chaîne de transmission de l'information vise à transmettre de manière rapide et fiable un message d'un expéditeur vers un destinataire. Son schéma général est donné par la figure (4.1).

Les différents éléments qui composent cette chaîne sont les suivants :

Le canal. Il peut s'agir

- d'un medium de transmission tel qu'une ligne de téléphone, un câble coaxial, une fibre optique ou une liaison radio (téléphone mobile, wifi, bluetooth, wimax, ...)
- d'un système physique de stockage de l'information, par exemple un disque magnétique ou optique, de la mémoire flash, de la RAM ou même d'un simple papier sur lequel est imprimé un message (code barre, code QR, filigrane).

Une caractéristique essentielle du canal réside dans sa non fiabilité, c'est-à-dire dans le fait que des perturbations de nature aléatoire affectent les messages qui y transitent. Le type de perturbation dépend du

canal : bruit d'origine électronique, interférences entre utilisateurs lors d'une transmission radio, poussières, rayures ou taches sur un support de stockage optique.

La source. La source délivre un message informatif. On distingue classiquement deux types de sources :

1. les sources numériques telles qu'un fichier ou une image numérique.
2. les sources analogiques telles que la voix ou la musique. Les sources analogiques sont pratiquement toujours converties en sources numériques plus faciles à manier techniquement, le passage du monde analogique au monde numérique est appelé *numérisation* et se compose le plus souvent de deux étapes nommées *échantillonnage* et *quantification*.

La source numérique (obtenue directement ou par numérisation) peut être modélisée de manière probabiliste, le modèle le plus simple est celui d'une suite de variables aléatoires indépendantes identiquement distribuées (i.i.d.), c'est la source simple.

Le codeur de source. Le rôle du codeur de source est de représenter le message issu de la source de façon aussi concise que possible. On distingue deux types de codeurs de source :

1. les codeurs avec perte : les codeurs de ce type assurent des taux de compression qui peuvent être très élevés, le prix à payer pour atteindre une forte compression est que le codage n'est pas réversible, c'est-à-dire qu'il n'est pas possible de revenir de manière exacte du message compressé à l'original. Le codage avec perte est intéressant pour des domaines dans lesquels une certaine dégradation de la qualité est acceptable. Les applications communes concernent la parole, la musique, l'image et la vidéo et des codeurs classiques pour ces cas sont typiquement le CELP (parole), le MP3 (musique), le JPEG (image) et le MPEG (vidéo).
2. les codeurs sans perte : les codeurs de ce type assurent des taux de compression plus modestes que les codeurs avec perte mais le codage est réversible. Cette propriété est indispensable dans certain cas, par exemple lorsque le message est un fichier exécutable.

Le codeur canal. Alors que le codeur de source a à charge d'éliminer, ou tout au moins de réduire, la redondance "incontrôlée" du message initial, le codeur de canal introduit de la redondance contrôlée en vue de protéger le message lors de sa transmission sur un canal non fiable. Fondamentalement, le principe de tout codeur canal consiste à répéter l'information transmise afin de permettre au récepteur de détecter voire de corriger les erreurs introduites lors du passage au travers du canal.

Si le fait que le taux d'erreur tende vers zéro lorsque le nombre de répétitions tend vers l'infini (mais au prix d'un débit qui tend lui aussi vers zéro) est assez intuitif (loi des grands nombres), l'un des résultats les plus remarquables de la théorie de l'information est qu'il est également possible d'assurer un taux d'erreur aussi faible que souhaité sans réduire à zéro le débit pourvu que celui-ci soit inférieur à une certaine limite liée aux caractéristiques du canal.

Le modulateur. Le rôle du modulateur consiste à adapter le message numérique à la nature physique du canal. Par exemple en transformant une suite binaire en une séquence d'allumages et d'extinction d'une source lumineuse ou en une tension variant au fil du temps.

Le démodulateur. Dual du modulateur assurant idéalement (pour un canal parfait) la restitution de la séquence numérique en entrée du modulateur à partir de l'ensemble des observations effectuées en sortie du canal.

Le décodeur canal. Dual du codeur de canal assurant la détection ou la correction, dans la mesure du possible, des erreurs introduites par le canal. Idéalement, le décodeur corrige toutes les erreurs produites par un canal non fiable dans certaines limites théoriques liées au niveau de perturbation du canal, à la quantité d'information produite par la source et à la taille du message transmis.

Le décodeur source. Dual du codeur de source visant à la restitution du message en entrée du codeur de source de manière exacte (codeur sans perte) ou approchée (codeur avec perte). Le codeur de source assure une compression, le décodeur de source une décompression.

4.2 Quelques mots sur la modulation.

L'entrée d'un canal physique (tel qu'un coaxial ou une ligne téléphonique par exemple) est un signal, c'est-à-dire une quantité physique (lumière, électricité, onde sonore, ...) porteuse d'information. Les signaux sont typiquement des fonctions du temps mais ils peuvent également dépendre de l'espace (image, code barre) ou de tout autre paramètre. Dans ce paragraphe, le signal est une fonction du temps.

Appelons T la durée de la transmission, combien de fonctions orthogonales est-il possible de construire sur cette durée T ? Il est clair que la réponse est une infinité, par exemple la base de Fourier

$$\left\{\exp\left[i2\pi\nu\frac{n}{T}t\right]\right\}_{n\in\mathbb{Z}}$$

pour le produit scalaire

$$\langle f,g\rangle = \frac{1}{T}\int_0^T f(t)g^*(t)dt$$

Les canaux physiques sont de bande passante B limitée, cela signifie que seules les fréquences $\nu\in[0,B]$ passent au travers du canal. Ainsi, le nombre des fonctions de base qui passent est $\frac{B}{1/T}=BT$ et pour transmettre un ensemble de BT valeurs $\{c_n\}_{n=1\cdots BT}$, on transmet la fonction

$$f(t)=\sum_{n=1}^{BT} c_n \exp\left[i2\pi\nu\frac{n}{T}t\right]$$

C'est la modulation ; transformation d'un ensemble discret de valeurs en une fonction.

Un modèle de canal élémentaire est tel que la sortie $s(t)$ du canal est une version bruitée par un bruit $b(t)$ de son entrée :

$$s(t)=f(t)+b(t)$$

En calculant le produit scalaire de $s(t)$ avec chacune des fonctions $\exp\left[i2\pi\nu\frac{n}{T}t\right]$, on obtient BT valeurs :

$$\begin{array}{rcl} s_1 &=& c_1+b_1\\ \vdots &\vdots& \vdots\\ s_N &=& c_N+b_N \end{array}$$

avec $b_n=\left\langle b(t),\exp\left[i2\pi\nu\frac{n}{T}t\right]\right\rangle$. C'est la démodulation, opération duale de la modulation qui permet de passer du monde continu des fonctions au monde discret des suites.

En l'absence de bruit, le couple (modulateur, démodulateur) est transparent : la sortie du démodulateur est égale à l'entrée du modulateur.

Les opérations de modulation/démodulation (pratiquement groupées au sein d'un mo-dem) transforment le canal physique en BT canaux élémentaires à entrée scalaire et sortie scalaire.

Modulation et démodulation réalisent le lien entre les modèles de canaux classiques en théorie de l'information et le monde réel.

Chapter 5

Canal et Capacité

La source en entrée du canal génère des symboles, elle possède une certaine entropie et distille une certaine quantité d'information.

Lors de son passage au travers du canal, l'information est affectée par des perturbations et la question est de savoir quelle part de l'information transmise peut être récupérée en sortie du canal. La quantité d'information qui passe dans le canal peut être vue comme la différence entre l'incertitude quant à la source avant observation de la sortie du canal et l'incertitude conditionnelle à cette observation. Cette différence dépend de la loi d'entrée du canal, une maximisation sur cette loi d'entrée conduit à la notion de capacité. L'existence de cette notion théorique de capacité est essentielle du fait du deuxième théorème de Shannon qui montre qu'une transmission peut être fiable (par un moyen appelé codage) à condition que l'entropie de la source soit inférieure à la capacité du canal.

5.1 Notion de canal

Pratiquement un "canal de transmission" est par exemple un milieu physique au travers duquel il est possible de faire passer de l'information. La démarche adoptée ici ne se préoccupe en aucun cas de l'origine physique des propriétés du canal, elle se contente d'en donner une description purement probabiliste. Le canal est alors considéré comme un système probabiliste qui accepte des symboles porteurs d'information en entrée et restitue en sortie d'autres symboles. Les alphabets d'entrée et de sortie sont en général différents.

5.1.1 Canal discret sans mémoire et invariant

Notons $\{x_1, \cdots, x_N\}$ les N lettres de l'alphabet d'entrée du canal et $\{y_1, \cdots, y_M\}$ les M lettres de son alphabet de sortie.

En général, la séquence $y_{j_1}, \cdots, y_{j_n}$ observée en sortie d'un canal discret dépend de son entrée $x_{i_1}, \cdots, x_{i_n}$ et d'un état interne du canal. Du point de vue probabiliste, cela se traduit par le fait que la loi de probabilité entrée-sortie est de la forme :

$$P_n\left(y_{j_1}, \cdots, y_{j_n}; x_{i_1}, \cdots, x_{i_n};\ \text{état}\right)$$

où les y_i appartiennent à l'alphabet de sortie du canal et les x_i à l'alphabet d'entrée. Cette relation signifie que le symbole présent à un instant donné en sortie du canal dépend de l'ensemble de symboles présents à l'entrée et de l'état propre du canal avant l'application de la première entrée x_1. Le canal est dit sans mémoire lorsque cette relation entrée-sortie peut être réduite à :

$$\begin{aligned} P_n\left(y_{j_1}, \cdots, y_{j_n}; x_{i_1}, \cdots, x_{i_n};\ \text{état}\right) &= P_n\left(y_{j_1}, \cdots, y_{j_n}; x_{i_1}, \cdots, x_{i_n}\right) \\ &= P_1\left(y_{j_1}|x_{i_1}\right) \cdots P_n(y_{j_n}|x_{i_n}) \end{aligned}$$

La première égalité signifie que le canal ne possède pas d'état interne et la seconde que le symbole observé en un indice donné en sortie ne dépend que de l'entrée au même indice. Dans le cas d'un canal sans mémoire, le lien probabiliste entre l'entrée et la sortie est complètement décrit par la donnée des jeux de probabilités de transition $P_k(y_{j_k}|x_{i_k})$, $k = 1 \cdots n$. Si les caractéristiques du canal sont invariantes $P_k(y_{j_k}|x_{i_k})$ ne dépend pas de k et ce jeu de probabilités est représentable sous forme matricielle : la matrice de transition $N \times M$ d'un canal discret sans mémoire est alors définie par :

$$\mathbf{\Pi} = [\Pi_{ij}] = [p(y_j|x_i)] \quad \begin{matrix} i \in \{1 \cdots N\} \\ j \in \{1 \cdots M\} \end{matrix}$$

Remarques :

- Chaque ligne de la matrice de transition $\mathbf{\Pi}$ contient un jeu de probabilités, d'où $\sum_{j=1}^{M} p\left(y_j|x_i\right) = 1, \forall i \in \{1, \cdots, N\}$. Ce n'est pas le cas pour les colonnes.
- La loi de probabilité de la sortie Y du canal s'obtient simplement à partir de celle de l'entrée X et de la matrice de transition[1] :

  $$\mathcal{P}_Y = \mathbf{\Pi}^T \mathcal{P}_X$$

  avec

  $$\begin{aligned} \mathcal{P}_Y^T &= [p(y_1), \cdots, p(y_M)] \\ \mathcal{P}_X^T &= [p(x_1), \cdots, p(x_N)] \end{aligned}$$

Quelques structures particulières utiles :

Canal uniforme par rapport à l'entrée. Pour un tel canal, les symboles sont tous affectés de la même manière par les erreurs : les lignes sont identiques à une permutation près. Une conséquence importante de cette uniformité en entrée est que l'entropie conditionnelle ne dépend pas de la loi de l'entrée $H(Y|X) = H(Y|X = x_i), \forall i \in \{1, \cdots, N\}$.

Canal uniforme par rapport à la sortie. Les colonnes sont identiques à une permutation près. Une conséquence importante de cette uniformité en sortie est qu'une loi uniforme en entrée induit une loi uniforme en sortie.

Canal symétrique. C'est un canal uniforme en entrée et en sortie avec $N = M$.

5.1.2 Canaux élémentaires

Deux canaux très simples jouent un rôle fondamental : le canal binaire symétrique et le canal à bruit additif gaussien.

Le Canal Binaire Symétrique (CBS)

Le canal binaire symétrique (cf. figure 5.1) est à entrées binaires (notées par exemple 0 et 1, mais cela n'a pas d'importance, ce ne sont que des étiquettes) et à sorties binaires (également notées 0 et 1) tel que :

- Le canal est sans mémoire : la sortie en k dépend uniquement de l'entrée en k.
- Binaire : 2 entrées possibles (0, 1) et 2 sorties possibles (0, 1).
- Symétrique : les entrées 0 et 1 sont affectées de manière égale par les erreurs (probabilité d'erreur p). Les probabilités de transition, et donc les performances du canal, sont complètement déterminées par un seul paramètre p :

  $$\begin{aligned} p &= Pr(y_k = 1|C_k = 0) = Pr(y_k = 0|C_k = 1) \\ 1-p &= Pr(y_k = |C_k = 1) = Pr(y_k = 0|C_k = 0) \end{aligned}$$

[1] Cette relation est la forme vectorielle de $p(y_j) = \sum_{i=1}^{N} p\left(y_j|x_i\right) p\left(x_i\right)$

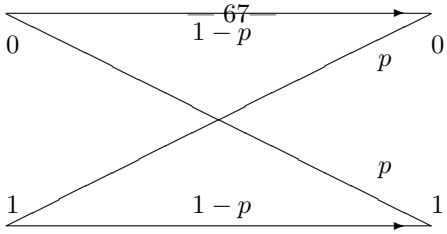


Figure 5.1: Canal binaire symétrique.

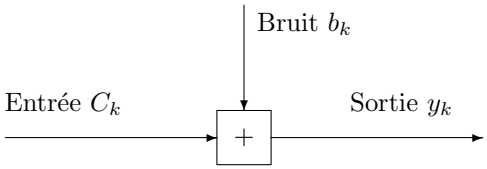


Figure 5.2: Canal à bruit additif blanc gaussien. A chaque utilisation canal, la sortie y_k est la somme de l'entrée C_k et d'une perturbation b_k distribuée selon une loi normale de moyenne nulle et de variance σ^2. La suite des v.a. b_k est i.i.d. : le canal est sans mémoire.

La matrice de transition $\mathbf{\Pi} = [\Pi_{ij}] = [p(y_j|x_i)]_{i,j\in\{1;2\}}$ d'un canal binaire symétrique est une matrice 2×2 donnée par :

$$\mathbf{\Pi} = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$$

Vraisemblance des observations en sortie de canal. Si l'entrée est un message constitué d'une suite de n valeurs binaires $C_1, \cdots, C_n$ que nous groupons dans un vecteur $\boldsymbol{C}$, la distance de Hamming entre le vecteur $\boldsymbol{y} = [y_1, \cdots, y_n]^T$ composé des n valeurs binaires observées en sortie du canal et le vecteur $\boldsymbol{C}$ composé des valeurs binaires placées en entrée est le nombre de positions qui diffèrent entre ces deux vecteurs

$$d_H(\boldsymbol{y}, \boldsymbol{C}) = |\{j | 0 \leq j \leq n, y_j \neq C_j\}|$$

La vraisemblance des observations $\boldsymbol{y}$ s'écrit :

$$p(\boldsymbol{y}|\boldsymbol{C}_{rs}) = (1-p)^n \left(\frac{p}{1-p}\right)^{d_H(\boldsymbol{y},\boldsymbol{C}_{rs})}$$

d'où

$$\log p(\boldsymbol{y}|\boldsymbol{C}_{rs}) = d_H(\boldsymbol{y}, \boldsymbol{C}_{rs}) \log\left(\frac{p}{1-p}\right) + n \log(1-p)$$

$n \log(1-p)$ est une constante et $\log\left(\frac{p}{1-p}\right) < 0$ (car $0 < p < 1/2$), ainsi maximiser la vraisemblance revient à minimiser $d_H(\boldsymbol{y}, \boldsymbol{C}_{rs})$.

Canal à bruit additif gaussien

Le canal gaussien (cf. figure (5.2)) est à entrée réelle et à sortie réelle tel que :

- Le canal est sans mémoire : la sortie dépend uniquement de l'entrée actuelle.

- La sortie est la superposition de l'entrée et d'une v.a. gaussienne centrée de variance σ^2 (le bruit gaussien) de densité de probabilité

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

- Les performances du canal sont complètement déterminées par un seul paramètre σ^2.

Bien que le canal gaussien ne soit pas à état fini en entrée ni en sortie, il est présenté ici car il joue un rôle important en pratique lorsque l'on s'intéresse à l'aspect physique d'un stockage ou d'une transmission de l'information.

Vraisemblance des observations en sortie de canal. Si une valeur C_k est placée en entrée d'un canal gaussien, sa sortie y_k suit une loi normale de variance σ^2 centrée en C_k que nous pouvons noter $p(y_k|C_k)$ ou $p_{C_k}(y_k)$:

$$p(y_k|C_k) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{|y_k - C_k|^2}{2\sigma^2}\right)$$

Si maintenant, l'entrée est un message constitué d'une suite de n valeurs $C_1, \cdots, C_n$ que nous groupons dans un vecteur $\boldsymbol{C} = [C_1, \cdots, C_n]^T$.

La distance euclidienne entre le vecteur $\boldsymbol{y} = [y_1, \cdots, y_n]^T$ composé des n valeurs observées en sortie du canal et le vecteur $\boldsymbol{C}$ composé des valeurs placées en entrée est la somme des carrés des erreurs entre ces deux vecteurs

$$d_E^2(\boldsymbol{y}, \boldsymbol{C}) = \|\boldsymbol{y} - \boldsymbol{C}\|^2 = \sum_{j=1}^{n} |y_j - C_j|^2$$

Lorsque la suite des perturbations b_k est une suite de v.a. indépendantes de même loi, la vraisemblance des observations $\boldsymbol{y}$ s'écrit :

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{C}) &= \prod_{j=1}^{n} \left[\frac{1}{\sigma\sqrt{2\pi}}\right] \exp\left(-\frac{|y_j - C_j|^2}{2\sigma^2}\right) \\ &= \left[\frac{1}{\sigma\sqrt{2\pi}}\right]^n \exp\left(-\frac{\|\boldsymbol{y} - \boldsymbol{C}\|^2}{2\sigma^2}\right) \\ &\propto \exp\left(-\frac{d_E^2(\boldsymbol{y}, \boldsymbol{C})}{2\sigma^2}\right) \end{aligned}$$

La fonction $\log$ étant monotone, maximiser le logarithme de la vraisemblance équivaut à maximiser la vraisemblance elle-même et conduit à des calculs plus simples :

$$\log p(\boldsymbol{y}|\boldsymbol{C}) = -n \log\left[\sigma\sqrt{2\pi}\right] - \frac{d_E^2(\boldsymbol{y}, \boldsymbol{C})}{2\sigma^2}$$

Ainsi, maximiser la vraisemblance à variance connue est équivalent à minimiser la distance euclidienne et il est possible d'estimer, au sens du maximum de vraisemblance (MV), l'entrée $\boldsymbol{C}$ à partir des sorties $\boldsymbol{y}$ en minimisant une distance Euclidienne.

La canal binaire vu comme approximation du canal gaussien

Supposons un canal gaussien à entrée binaire $C_k = \pm 1$. Lorsque $C_k = +1$, la sortie du canal suit une loi normale de moyenne $+1$; lorsque $C_k = -1$, une loi normale de moyenne -1.

La règle de décision optimale au sens du MV consiste à prendre le signe de la sortie du canal gaussien et ainsi à transformer celle-ci en une valeur binaire. On créé de cette manière un canal binaire symétrique entre les entrées binaires du canal gaussien et les décisions binaires prises en sortie (cf. figure 5.3).

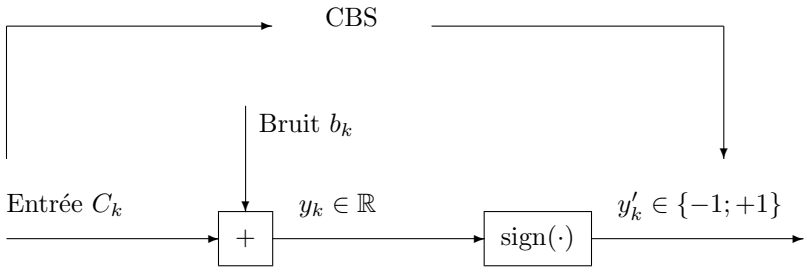


Figure 5.3: Un canal à bruit additif blanc gaussien à entrées binaires avec décisions binaires en sortie est un canal binaire symétrique (CBS) de paramètre $p = Q\,(1/\sigma)$.

En ce sens, le canal binaire symétrique est une approximation du canal gaussien avec :

$$p = \frac{1}{\sigma\sqrt{2\pi}} \int_1^{+\infty} \exp\left(-\frac{x^2}{2\sigma^2}\right) dx = Q\left(\frac{1}{\sigma}\right)$$

avec

$$Q(x) = (2\pi)^{-1/2} \int_x^{+\infty} \exp\left(-t^2/2\right) dt$$

Cette approximation fait perdre la fiabilité des décisions : pour une observation positive proche de 0, les hypothèses ± 1 sont presque aussi vraisemblables l'une que l'autre alors que pour une observation positive très supérieure à 0 l'hypothèse $+1$ est beaucoup plus vraisemblable que -1. Une fois la décision prise, cette information est perdue.

Les hypothèses faites dans la suite sont les suivantes :

- l'entrée du canal est une suite de réalisations indépendantes d'une v.a. X. Cette hypothèse est naturelle puisque l'un des buts du codage de source est précisément de rendre les caractères après compression indépendants les uns des autres.
- le canal est invariant et sans mémoire de sorte que, pour une entrée X, sa sortie Y est liée seulement à X.

5.2.1 Définition de la capacité

L'observation de la sortie Y diminue l'incertitude sur l'entrée X ; elle apporte de l'information. Avant observation de Y, l'incertitude *a priori* sur l'entrée X est $H(X)$. Après observation de Y, elle est plus faible et vaut $H(X|Y) \leq H(X)$. La réduction d'incertitude est la quantité d'information apportée par Y sur X, elle vaut :

$$I(X;Y) = \overbrace{H(X)}^{\text{Incertitude } a\ priori} - \overbrace{H(X|Y)}^{\text{Incertitude } a\ posteriori} \geq 0$$

Étant donné que le conditionnement réduit l'incertitude ($0 \leq H(X|Y) \leq H(X)$), on a $0 \leq I(X;Y) \leq H(X)$.

Cette information mutuelle est positive ou nulle, elle dépend de la loi de X et de la nature du canal. Puisque qu'il est impossible de modifier la nature physique du canal, la maximisation de l'information qui traverse le canal se fait en choisissant au mieux la loi $\mathcal{P}_X$ de l'entrée, d'où la définition de la capacité C par l'information maximale qui peut passer au travers de ce canal :

$$C = \max_{\mathcal{P}_X} I(X;Y)$$

L'information mutuelle est une fonction convexe de la loi de l'entrée (exercice, le montrer), ainsi la recherche du maximum de l'information mutuelle se réduit à celle d'un extremum.

Signification des différents termes impliqués dans la capacité :

$H(X)$ Entropie de la source.

$H(X|Y)$ Incertitude résiduelle sur X sachant Y (Cette quantité est parfois appelée équivoque.). De l'incertitude subsiste du fait que le canal est bruité.

$I(X;Y)$ L'information qui passe entre X et Y ; c'est-à-dire, l'information mutuelle étant symétrique, l'information partagée par les v.a. X et Y.

En général, le calcul de la capacité d'un canal est difficile. Le paragraphe suivant traite d'un cas simple mais réaliste : le canal symétrique.

5.2.2 Capacité d'un canal symétrique.

Le calcul de la capacité d'un canal symétrique peut être mené à bien sans difficulté. Rappelons que l'information mutuelle est symétrique $I(X;Y) = I(Y;X) = H(Y) - H(Y|X)$. Le calcul se décompose en plusieurs étapes :

1. Monter que l'entropie conditionnelle $H(Y|X)$ est indépendante de la loi d'entrée. Ce résultat permet de ramener la maximisation de l'information mutuelle à celle de $H(Y)$.

   Preuve. On a

   $$H(Y|X) = \sum_{k=1}^{N} p(x_k) H(Y|X = x_k)$$

   avec

   $$H(Y|X = x_k) = -\sum_{j=1}^{M} p(y_j|x_k) \log p(y_j|x_k)$$

Les quantités $p(y_j|x_k)$ représentent les éléments de la k-ième ligne de la matrice de transition du canal. Le canal étant symétrique, *a fortiori* uniforme en entrée, le jeu de probabilités est le même sur toutes les lignes et l'entropie $H(Y|X=x_k)$ est indépendante de k, d'où

$$\begin{aligned} H(Y|X) &= \sum_{k=1}^{N} p(x_k)\, H(Y|X=x_k) \\ &= H(Y|X=x_i)\left[\sum_{k=1}^{N} p(x_k)\right],\ \forall i = 1\cdots N \\ &= H(Y|X=x_i),\ \forall i = 1\cdots N \end{aligned}$$

Finalement l'entropie conditionnelle

$$H(Y|X) = -\sum_{j=1}^{M} p(y_j|x_i)\log p(y_j|x_i),\ \forall i = 1\cdots N$$

ne dépend que des probabilités de transition du canal $p(y_j|x_i)$, elle est indépendante de la loi d'entrée. □

2. Montrer que $H(Y)$ est maximum lorsque la loi d'entrée est uniforme.

   Preuve. $H(Y)$ est maximale lorsque Y suit une loi uniforme. Il suffit donc de déterminer la loi de X qui rend Y uniforme. Montrons que la loi uniforme pour X est solution. Si X est uniforme, on a $p(x_i) = 1/N$ pour tout i. La probabilité du symbole y_j en sortie est donnée par

   $$p(y_j) = \sum_{i=1}^{N} p(x_i, y_j) = \sum_{i=1}^{N} p(x_i)p(y_j|x_i) = \frac{1}{N}\sum_{i=1}^{N} p(y_j|x_i)$$

   La somme $\sum_{i=1}^{N} p(y_j|x_i)$ représente la somme des éléments de la colonne j : elle est indépendante de j car le canal est symétrique, *a fortiori* uniforme en sortie. Ainsi, une entrée de loi uniforme maximise l'information mutuelle et permet d'atteindre la capacité. □

3. Calculer la capacité

   Valeur de la capacité du canal symétrique. Il suffit d'évaluer l'information mutuelle pour une loi d'entrée uniforme. On a

   $$C = H(Y) - H(Y|X) = \log(M) + \sum_{j=1}^{M} p(y_j|x_i)\log p(y_j|x_i)$$

   Cette formule est valable pour un canal doublement uniforme (en entrée et en sortie). Si le canal est symétrique on a aussi $N = M$. □

Cas du canal binaire symétrique. Pour un CBS, la connaissance de la seule quantité scalaire p détermine complètement la matrice de transition du canal et sa capacité ne dépend que cette probabilité d'erreur p :

$$C(p) = 1 + (1-p)\log(1-p) + p\log(p)$$

En notant $H(p, 1-p)$ l'entropie d'une loi binominale, on a $C = 1 - H(p, 1-p)$.

Cas sans bruit. Pour p nul ou égal à un, la capacité est maximale, le canal est sans bruit.

Un canal sans bruit est idéal pour la transmission ou le stockage de l'information.

Canal de capacité nulle. Pour $p = 1/2$, la capacité est nulle. En effet, un symbole en sortie du canal peut provenir avec la même probabilité de l'un ou de l'autre des symboles d'entrée. Autrement dit, l'observation de la sortie ne renseigne en rien l'entrée : l'information apportée par cette observation est nulle.

Un canal de capacité nulle est idéal pour la dissimulation de l'information.

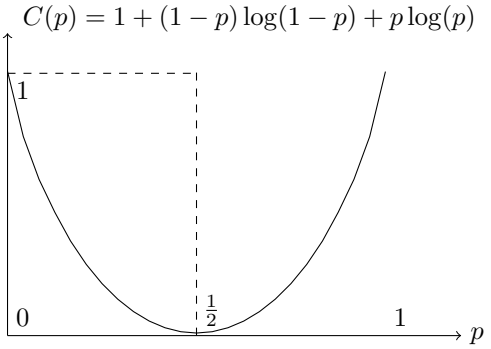


Figure 5.4: Capacité d'un CBS.

La capacité d'un canal binaire symétrique (CBS) est maximale en $p = 0$ et $p = 1$ (canal sans bruit), elle est nulle en $p = 1/2$ (canal de capacité nulle). Le pire cas (pour la transmission) est donc $p = 1/2$ et non pas 1, valeur pour laquelle il suffit d'inverser chacune des sorties pour restaurer parfaitement l'entrée.

5.3 Second théorème de Shannon

L'importance de la capacité tient à un résultat essentiel dû à C. Shannon qui affirme que, pour peu que l'entropie de la source placée en entrée du canal soit strictement inférieure à la capacité de celui-ci, il est possible, sous certaines conditions, de récupérer parfaitement l'entrée à partir de la sortie. Ce résultat peu intuitif permet de construire des systèmes de transmission ou de stockage fiables sur des canaux qui ne le sont fondamentalement pas.

Le moyen pratique qui permet d'atteindre cet objectif de fiabilité est le codage canal (aussi appelé codage détecteur et correcteur d'erreurs). Contrairement au codage de source qui vise à décrire les messages délivrés par la source de la manière la plus concise possible, le codage de canal augmente la longueur des messages par adjonction de redondance en vue de permettre une correction des erreurs dues aux imperfections du canal lors de l'estimation de l'entrée du canal à partir de sa sortie.

5.3.1 Code à répétition

Commençons par un exemple très simple destiné à illustrer le caractère étonnant du deuxième théorème de Shannon.

Soit un canal binaire symétrique, de probabilité d'erreur p, à entrées et sorties dans $\{0, 1\}$. On cherche à réduire cette probabilité d'erreur en ajoutant au message une certaine redondance. La méthode la plus simple consiste à répéter plusieurs fois chaque symbole en entrée du canal pour procéder ensuite à une décision majoritaire en sortie.

Supposons que le même symbole x (valant 0 ou 1) soit émis $n = 2s + 1$ fois (s entier), la décision peut procéder de la règle suivante :

- si le nombre de x reçu est supérieur ou égal à $s + 1$, on décide que x a été émis,
- sinon, on décide $1 - x$.

Illustration pour $n = 3$. La figure (5.5) illustre les mots d'un code à répétition pour $n = 3$. Les mots du code (000) et (111) sont représentés par les grandes sphères.

- Si l'un des mots du code est reçu, on décide que c'est bien ce mot qui est présent en entrée. Cette décision, bien que non certaine, est très fiable lorsque les erreurs qui affectent les valeurs transmises sont indépendantes. En effet, si (000) est en entrée et que, par exemple, la probabilité d'erreur vaut $p = 10^{-2}$, la probabilité d'observer ce même mot (000) en sortie vaut $(1 - 10^{-2})^3 \approx 0.97$ alors que celle d'observer le mot (111) vaut $(10^{-2})^3 = 10^{-6}$.
- De même lorsqu'une séquence autre que (000) ou (111) est observée, il est plus probable qu'elle provienne de celui des deux mots du code qui se trouve à distance de Hamming minimale et décider qu'elle provient effectivement de ce mot est la règle de décodage qui minimise la probabilité d'erreur.

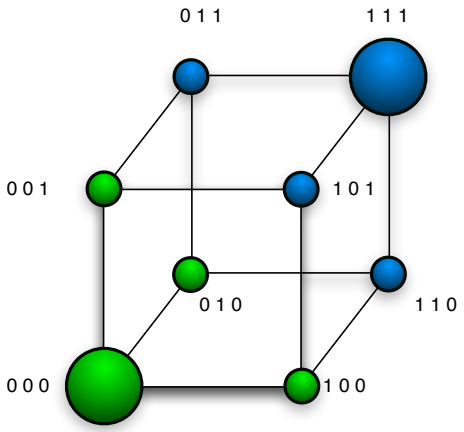


Figure 5.5: k=1 bit d'information, 2 mots de code (de longueur n=3): (000) (111).

Par exemple, si l'on observe 001, la probabilité pour que cette observation provienne du mot d'entrée 000 (1 erreur en position 3) vaut $(1-10^{-2})^2 10^{-2} \approx 10^{-2}$ alors que la probabilité pour qu'elle provienne du mot 111 (2 erreurs aux positions 1 et 2) vaut $(1-10^{-2})\left(10^{-2}\right)^2 \approx 10^{-4}$. L'hypothèse '$000$ en entrée' est environ cent fois plus vraisemblable que l'hypothèse '111 en entrée'.

Probabilité d'erreur par mot. Le canal étant symétrique, la probabilité d'erreur est la même pour le 0 et pour le 1. Ainsi, il est possible de calculer la probabilité d'erreur en ne considérant que le cas d'une entrée est égale à 0.

Pour le code à répétition, cette entrée 0 est transmise $2s+1$ fois : l'entrée est le mot de code $C_0, \cdots, C_{2s}$ à $n = 2s+1$ caractères, tous égaux à 0. Pour chacun des 0 du mot d'entrée, la sortie y_j qui correspond vaut 1 avec probabilité p.

Notons $S_{2s+1} = \sum_{j=0}^{2s} y_j$ le nombre de 1 reçus lorsque le mot composé de $2s+1$ valeurs 0 est transmis. La probabilité d'erreur après prise de décision majoritaire (probabilité d'erreur par mot) est donnée par :

$$Pr\left(S_{2s+1} \geqslant s+1\right)$$

Remarque préliminaire. La probabilité d'erreur par mot s'écrit :

$$Pr\left(S_{2s+1} \geqslant s+1\right) = Pr\left(\frac{S_{2s+1}}{2s+1} \geqslant \frac{s+1}{2s+1}\right)$$

Lorsque $n \to \infty$, le terme $\frac{s+1}{2s+1}$ tend vers $1/2$ alors que, d'après la loi des grands nombres, la moyenne empirique $\frac{S_{2s+1}}{2s+1}$ tend vers $p < 1/2$, ainsi $\lim_{n\to\infty} Pr\left(S_{2s+1} \geqslant s+1\right) = 0$.

Calcul de la probabilité d'erreur après décodage. La probabilité pour que la somme $S_{2s+1} = \sum_{j=0}^{2s} y_j$ soit égale à k (entier compris entre 0 et $2s+1$) vaut :

$$Pr\left(S_{2s+1} = k\right) = C_{2s+1}^{k} p^k \left(1-p\right)^{2s+1-k}$$

La probabilité d'erreur est la probabilité de décider 1, cela se produit lorsque le nombre de 1 reçu est supérieur à $s+1$:

$$Pr\left(S_{2s+1} \geq s+1\right) = \sum_{k=s+1}^{2s+1} C_{2s+1}^{k} p^k \left(1-p\right)^{2s+1-k}$$

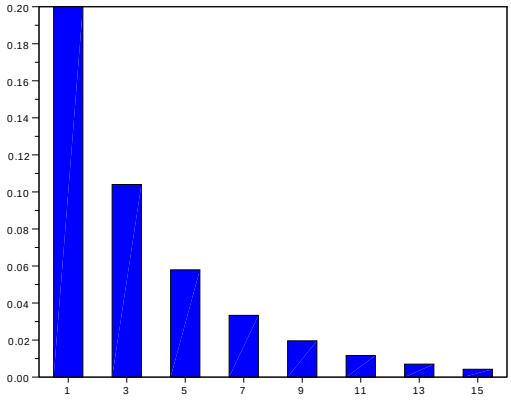


Figure 5.6: Performance d'un code à répétition sur un CBS pour $p = 1/5$ et une longueur des mots de code variant de 1 à 15.

La figure (5.6) représente cette probabilité d'erreur en fonction de la longueur des mots de code pour $p = 1/5$.

Ainsi, un code à répétition permet de réduire autant qu'on le souhaite la probabilité d'erreur. Le prix à payer est le suivant. Supposons que le canal puisse transmettre au maximum un symbole par seconde. Initialement, le débit d'information était de 1 bit/s. Après codage répétitif, il n'est plus que de $1/n = 1/(2s + 1)$ bit/s. Pour réduire à une valeur arbitrairement petite la probabilité d'erreur, il a donc fallu réduire dans les mêmes proportions le débit de transmission.

Contrairement à ce que pourrait laisser penser l'exemple du code à répétition, il n'est pas nécessaire de réduire à 0 le débit pour réduire à 0 la probabilité d'erreur.

Une procédure de codage plus générale que celle utilisée par le code à répétition consiste à associer à toute séquence de longueur k un mot de code de longueur $n > k$. Le rendement d'un tel codage est $R = k/n$.

Une première approche intuitive est la suivante : parmi les 2^n séquences de longueur n seules 2^k sont des mots de code. La densité des mots de code dans l'espace des séquences de longueur n est donc

$$\frac{2^k}{2^n} = 2^{k-n} = 2^{n\left(\frac{k}{n}-1\right)}$$

Comme $n > k$, le rendement R du code vérifie $R = k/n < 1$ d'où $R - 1 < 0$ et finalement

$$\frac{2^k}{2^n} = 2^{n(R-1)} \quad \xrightarrow{n \to \infty} \quad 0$$

A rendement constant, l'espace des séquences de longueurs n est de plus en plus vide au fur et à mesure que la taille des mots croît. Ainsi, les possibilités de détection et de correction des erreurs croissent avec n.

La différence fondamentale entre cette approche et le codage à répétition est qu'ici ce n'est pas k qui est fixé mais le rendement $R = k/n$. Ainsi, pour $n \to \infty$, la densité des mots de code tend vers 0 à rendement constant (alors que pour le code à répétition, le rendement $R = 1/n$ tend vers 0 à la même vitesse que la densité des mots).

Le résultat étonnant établit par Shannon est que, sous réserve que le débit reste inférieur à la capacité du canal, il est possible de réduire à une valeur arbitrairement petite la probabilité d'erreur sans réduire le débit de transmission. Le moyen permettant d'arriver à ce résultat s'appelle le codage canal.

Bien que, en général, les codes qui permettent d'arriver à ce genre de résultats soient beaucoup plus complexes qu'un code à répétition, tous utilisent d'une manière ou d'une autre une forme de répétition.

Notations

- X_n l'ensemble des suites $\mathbf{x}$ de longueur n en entrée du canal (écrites dans un alphabet d'entrée I).
- Y_n l'ensemble des suites $\mathbf{y}$ de longueur n en sortie du canal (écrites dans un alphabet de sortie J).
- $p(\mathbf{y}|\mathbf{x})$ la probabilité de transition du canal pour les séquences de longueur n (probabilité d'observer la séquence $\mathbf{y}$ lorsque la séquence $\mathbf{x}$ est en entrée).

Codeur canal. En entrée du canal, on utilise un sous-ensemble de $M = 2^k$ mots dans l'ensemble X_n des séquences de longueurs n ($k < n$). A chacun de ces M états, le codeur canal associe un mot de code composé d'une suite de n caractères (les caractères de l'alphabet d'entrée du canal, ici typiquement 0 ou 1).

Hypothèses

1. La source d'information est simple et de loi uniforme (copies indépendantes d'une v.a. uniformément distribuée sur l'alphabet utilisé).

   Si la source possède 2 états, son extension d'ordre k est une v.a. uniforme à $M = 2^k$ états et les k v.a. sont indépendantes et toutes de loi uniforme.

   Avant leur entrée dans le canal, ces k valeurs sont transformées, par une opération déterministe (codeur canal), en une suite plus longue de $n > k$ valeurs. Ainsi, chacun de ces n valeurs porte en moyenne $R = k/n$ bits d'information. R est le rendement du code.

2. Le canal est sans mémoire à entrées et sorties discrètes. Pour un canal sans mémoire $p(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{n} p(y_j|x_j)$.

Décodeur au sens du maximum du vraisemblance. A chacune des séquences $\mathbf{y}$ de longueur n en sortie du canal, le décodeur canal associe l'un des mots du code, c'est-à-dire l'un des états de la source étendue. La règle de décision qui minimise la probabilité d'erreur est celle du maximum de vraisemblance (cf. TD 8) :

On décide le mot $\mathbf{x_m}$ si

$$p\left(\mathbf{y}|\mathbf{x_m}\right) > p\left(\mathbf{y}|\mathbf{x_{m'}}\right) \forall m' \neq m$$

Le second théorème de Shannon ou théorème du codage canal donne une limite sur l'entropie de la source en dessous de laquelle le codage peut garantir un taux d'erreur aussi faible que nécessaire pour peu que la longueur des mots du code soit suffisamment grande.

La démonstration de ce résultat repose sur l'évaluation des performances de codes par bloc en utilisant la technique dite du "codage aléatoire".

Majoration de la probabilité d'erreur pour un mot Le mot m étant en entrée, la probabilité d'erreur $P_{e(m)}$ est la probabilité de décider $m' \neq m$, c'est-à-dire la probabilité qu'il existe un m' tel que $p\left(\mathbf{y}|\mathbf{x_{m'}}\right) > p\left(\mathbf{y}|\mathbf{x_m}\right)$.

Notons

$$\phi_m\left(\mathbf{y}\right) = \begin{cases} 1 & \text{s'il existe un } m' \text{ tel que } p\left(\mathbf{y}|\mathbf{x_{m'}}\right) > p\left(\mathbf{y}|\mathbf{x_m}\right). \\ 0 & \text{sinon.} \end{cases}$$

La probabilité d'erreur lorsque le mot m est en entrée s'écrit :

$$P_{e(m)} = \sum_{\mathbf{y} \in Y_n} P\left(\mathbf{y}|\mathbf{x}_m\right) \phi_m\left(\mathbf{y}\right)$$

Majoration pour tout $s > 0$:

$$\phi_m\left(\mathbf{y}\right) \leq \left[\frac{\sum_{m' \neq m} P\left(\mathbf{y}|\mathbf{x}_{m'}\right)^{\frac{1}{1+s}}}{P\left(\mathbf{y}|\mathbf{x}_m\right)^{\frac{1}{1+s}}} \right]^s$$

Le second membre étant positif (probabilités positives) l'inégalité est immédiate pour $\phi_m\left(\mathbf{y}\right) = 0$.

Pour $\phi_m\left(\mathbf{y}\right) = 1$, il y a erreur de détection, c'est-à-dire qu'il existe un m' tel que $p\left(\mathbf{y}|\mathbf{x_{m'}}\right) > p\left(\mathbf{y}|\mathbf{x_m}\right)$, le numérateur est donc supérieur au dénominateur et le second membre est supérieur à 1. □

De cette majoration de $\phi_m\left(\mathbf{y}\right)$ découle celle de la probabilité d'erreur :

$$P_{e(m)} \leq \sum_{\mathbf{y} \in Y_n} \left\{ P\left(\mathbf{y}|\mathbf{x}_m\right)^{\frac{1}{1+s}} \left[\sum_{m' \neq m} P\left(\mathbf{y}|\mathbf{x}_{m'}\right)^{\frac{1}{1+s}} \right]^s \right\}, s > 0$$

Introduction du codage aléatoire. L'idée consiste à considérer un ensemble probabilisé de codes pour lequel une majoration simple de la probabilité d'erreur moyenne peut être obtenue pour ensuite en déduire l'existence d'un code au sein de cet ensemble dont la probabilité d'erreur est également inférieure à cette borne.

Les mots du code sont issus de tirages indépendants selon une loi $P(\mathbf{x})$ (définie sur l'espace des séquences d'entrée).

Lorsque les mots de code sont aléatoires, les quantités $P\left(\mathbf{y}|\mathbf{x}_m\right)$ qui en dépendent sont des variables aléatoires indépendantes (pour différentes valeurs de m).

Majoration de la probabilité d'erreur pour un ensemble probabilisé de codes

$$\begin{aligned}
\bar{P_e} = \mathbb{E}P_{e(m)} &\leq \mathbb{E}\sum_{\mathbf{y}\in Y_n}\left\{P\left(\mathbf{y}|\mathbf{x}_m\right)^{\frac{1}{1+s}}\left[\sum_{m'\neq m}P\left(\mathbf{y}|\mathbf{x}_{m'}\right)^{\frac{1}{1+s}}\right]^s\right\}, 0<s \\
\text{(Linéarité intégrale)} &= \sum_{\mathbf{y}\in Y_n}\mathbb{E}\left\{P\left(\mathbf{y}|\mathbf{x}_m\right)^{\frac{1}{1+s}}\left[\sum_{m'\neq m}P\left(\mathbf{y}|\mathbf{x}_{m'}\right)^{\frac{1}{1+s}}\right]^s\right\}, 0<s \\
\text{(Mots indépendants)} &= \sum_{\mathbf{y}\in Y_n}\left\{\mathbb{E}\left[P\left(\mathbf{y}|\mathbf{x}_m\right)^{\frac{1}{1+s}}\right]\mathbb{E}\left[\left(\sum_{m'\neq m}P\left(\mathbf{y}|\mathbf{x}_{m'}\right)^{\frac{1}{1+s}}\right)^s\right]\right\}, 0<s
\end{aligned}$$

Pour $0<s<1$ la fonction $f(x)=x^s$ est concave et l'inégalité de Jensen permet d'écrire la majoration :

$$\begin{aligned}
\bar{P_e} &\leq \sum_{\mathbf{y}\in Y_n}\left\{\mathbb{E}\left[P\left(\mathbf{y}|\mathbf{x}_m\right)^{\frac{1}{1+s}}\right]\left[\mathbb{E}\sum_{m'\neq m}P\left(\mathbf{y}|\mathbf{x}_{m'}\right)^{\frac{1}{1+s}}\right]^s\right\}, 0<s<1 \\
\text{(Linéarité intégrale)} &= \sum_{\mathbf{y}\in Y_n}\left\{\mathbb{E}\left[P\left(\mathbf{y}|\mathbf{x}_m\right)^{\frac{1}{1+s}}\right]\left[\sum_{m'\neq m}\mathbb{E}\left(P\left(\mathbf{y}|\mathbf{x}_{m'}\right)^{\frac{1}{1+s}}\right)\right]^s\right\}, 0<s<1
\end{aligned}$$

La quantité

$$\mathbb{E}\left(P\left(\mathbf{y}|\mathbf{x}_{m'}\right)^{\frac{1}{1+s}}\right)=\sum_{\mathbf{x}\in X_n}P(\mathbf{x})P\left(\mathbf{y}|\mathbf{x}\right)^{\frac{1}{1+s}}$$

est la même pour tous les mots m' d'où la majoration de la probabilité d'erreur moyenne pour un ensemble probabilisé de codes :

$$\bar{P_e}\leq(M-1)^s\sum_{\mathbf{y}\in Y_n}\left[\sum_{\mathbf{x}\in X_n}P(\mathbf{x})P\left(\mathbf{y}|\mathbf{x}\right)^{\frac{1}{1+s}}\right]^{1+s}, 0<s<1$$

Cette majoration permet de déduire qu'il existe un code pour lequel cette borne est également un majorant de sa probabilité d'erreur.

Nous allons maintenant montrer que, pour peu que $R<C$, cette borne tend vers 0 lorsque la longueur des mots croît.

Borne pour un canal sans mémoire et une source simple. Pour trouver une forme simple de cette borne très générale, prenons maintenant en compte les hypothèses particulières qui sont faites ici :

- Le canal est sans mémoire :
$$p\left(\mathbf{y}|\mathbf{x}\right)=\prod_{j=1}^{n}p\left(y_j|x_j\right)$$
- La source est simple :
$$p\left(\mathbf{x}\right)=\prod_{j=1}^{n}p\left(x_j\right)$$

Sous ces hypothèses simplificatrices, la majoration devient :

$$\bar{P_e}\leq(M-1)^s\sum_{\mathbf{y}\in Y_n}\left[\sum_{\mathbf{x}\in X_n}\prod_{j=1}^{n}p\left(x_j\right)p\left(y_j|x_j\right)^{\frac{1}{1+s}}\right]^{1+s}, 0<s<1$$

Soit, en écrivant les sommes sur les alphabets d'entrée et de sortie du canal :

$$\bar{P}_e \leq (M-1)^s \left\{ \sum_{j\in J} \left[\sum_{k\in I} p_k p(j|k)^{\frac{1}{1+s}} \right]^{1+s} \right\}$$

En utilisant la majoration $M-1 < M = 2^{nR}$ ($R = k/n$ le rendement du code), on a :

$$\bar{P}_e \leq 2^{-n[-sR+E_0(s,\{p_k\})]}$$

avec

$$E_0\left(s,\{p_k\}\right) = -\log_2 \left[\sum_{j\in J} \left[\sum_{k\in I} p_k p(j|k)^{\frac{1}{1+s}} \right]^{1+s} \right]$$

Ainsi, il existe un code dont la probabilité d'erreur par mot P_e est majorée comme suit :

$$P_e \leq 2^{-nE(R)} \text{ avec } E(R) = \max_{s,\{p_k\}} \left[-sR + E_0\left(s,\{p_k\}\right)\right]$$

Nous allons maintenant calculer cette borne dans le cas particulier du canal binaire symétrique et montrer qu'elle décroît pour s'annuler en $R = C$.

Calcul explicite de la borne pour un canal binaire symétrique. Pour simplifier la maximisation, nous supposons que la loi d'entrée est uniforme[2] de sorte que la maximisation ne porte que sur le paramètre s.

$$E_0\left(s,\{p_k\}\right) = -\log_2 \left[\sum_{j=0}^{1} \left[\sum_{k=0}^{1} p_k p(j|k)^{\frac{1}{1+s}} \right]^{1+s} \right]$$

Pour $p_0 = p_1 = 1/2$:

$$E_0\left(s,\{p_k\}\right) = -\log_2 \left[\left[\frac{1}{2} p(0|0)^{\frac{1}{1+s}} + \frac{1}{2} p(0|1)^{\frac{1}{1+s}} \right]^{1+s} + \left[\frac{1}{2} p(1|0)^{\frac{1}{1+s}} + \frac{1}{2} p(1|1)^{\frac{1}{1+s}} \right]^{1+s} \right]$$

Pour un CBS : $p(1|1) = p(0|0) = 1-p$ et $p(1|0) = p(0|1) = p$:

$$\begin{aligned} E_0\left(s,\{p_k\}\right) &= -\log_2 \left[\frac{1}{2^s} \left[p^{\frac{1}{1+s}} + (1-p)^{\frac{1}{1+s}} \right]^{1+s} \right] \\ &= s - (1+s)\log_2 \left[p^{\frac{1}{1+s}} + (1-p)^{\frac{1}{1+s}} \right] \end{aligned}$$

Pour éviter le calcul analytique fastidieux, il est possible de calculer numériquement le maximum

$$E(R) = \max_{0<s<1} \left\{ s(1-R) - (1+s)\log_2 \left[p^{\frac{1}{1+s}} + (1-p)^{\frac{1}{1+s}} \right] \right\}$$

et de tracer son évolution en fonction de R : la figure (5.7) représente l'évolution de la fonction $E(R)$ en fonction de R pour différentes valeurs de la capacité du canal[3].

[2] Cette hypothèse est naturelle puisque l'on sait que, pour un canal binaire symétrique, la capacité est atteinte pour une loi d'entrée uniforme. En maximisant seulement sur s, on trouvera ainsi une borne qui, même si elle n'était pas la plus fine, permet de conclure sur ce cas particulier.

[3] On a $-sR + E_0\left(s,\{p_k\}\right) = s(1-R) - sH_{\frac{1}{1+s}}$ où $H_\lambda = (1-\lambda)^{-1}\log_2 \sum_k p_k^\lambda$ est l'entropie de Rényi ; H_λ est une fonction décroissante de λ avec $H_0 = \log_2 2 = 1$ et $H_\infty = -\log_2 \max(p, 1-p)$. En $R = C$, $-sR + E_0\left(s,\{p_k\}\right)|_{R=C} = s\left(H_1 - H_{\frac{1}{1+s}}\right)$, le minimum de $H_{\frac{1}{1+s}}$ est atteint en $s = 0$, point en lequel la fonction $-sR + E_0$ s'annule.

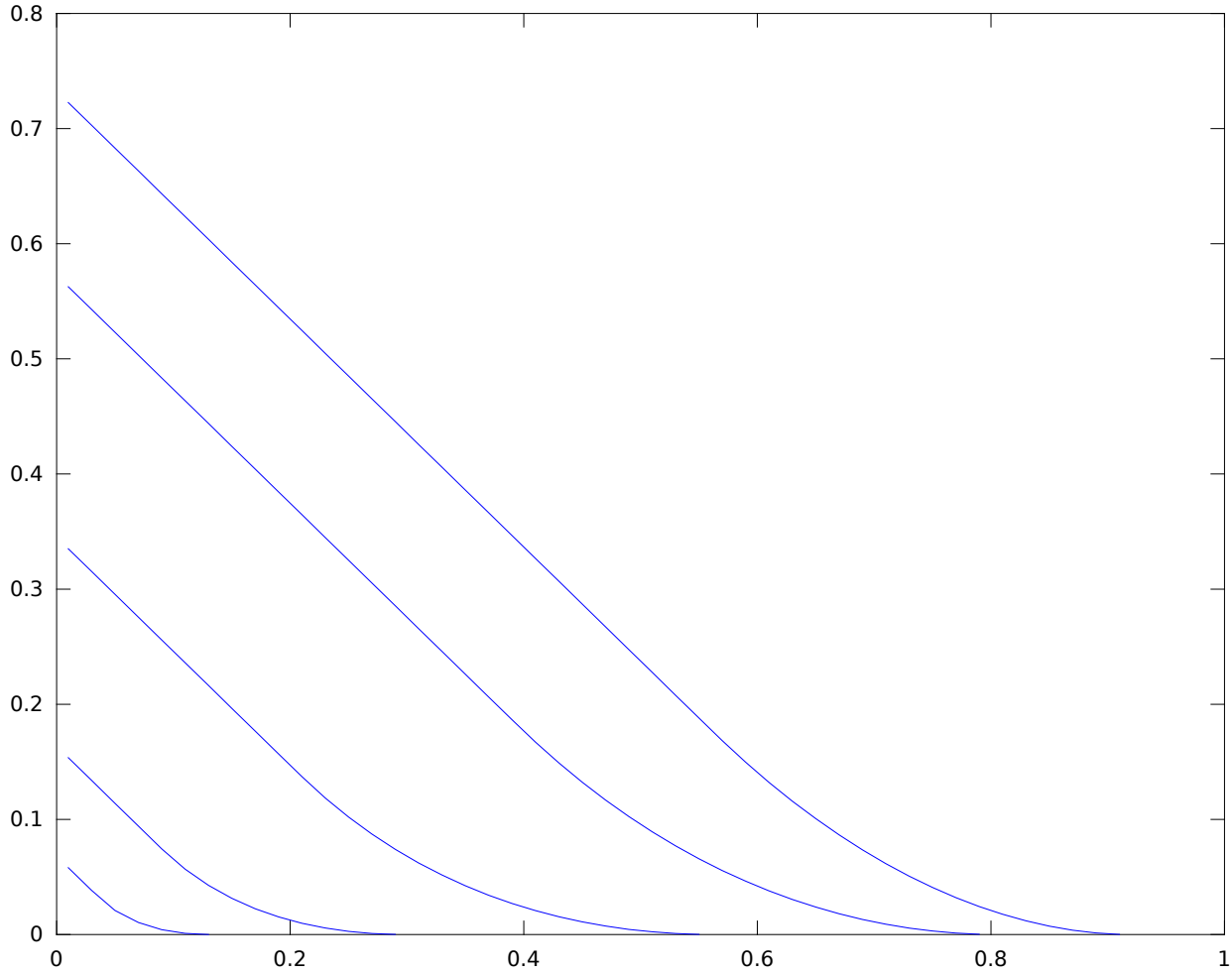


Figure 5.7: Borne $E(R)$ pour un canal binaire symétrique et les valeurs de capacités $C = 0.13$; 0.3 ; 0.56 ; 0.8 ; 0.92. La fonction $E(R)$ est positive, décroissante et s'annule en $R = C$ valeur du rendement au delà de laquelle la probabilité d'erreur par mot ne tend plus vers zéro lorsque la taille des mots de code croît.

Code octave pour tracer la figure (5.7).

```
% Merci à Rodrigo Cabral pour ce morceau de code
p=0.01:0.02:0.99;
s=0.01:0.02:0.99;
r=0.01:0.02:0.99;
invps=1./(1+s);
expon=zeros(length(p),length(r),length(s));
max_exp=zeros(length(p),length(r));
for i=1:length(p)
    for j=1:length(r)
        for k=1:length(s)
        expon(i,j,k)=s(k)*(1-r(j))
                    -(1+s(k))*log2((p(i)^invps(k))+((1-p(i))^(invps(k))));
        end
    max_exp(i,j)=max(expon(i,j,:));
    end
end
max_exp(max_exp<=0)=NaN;
capacity=1+(p.*log2(p)+(1-p).*log2(1-p));
f=max_exp';
hold;
plot(r,f(:,1)); capacity(1)
plot(r,f(:,2)); capacity(2)
plot(r,f(:,5)); capacity(5)
plot(r,f(:,10)); capacity(10)
plot(r,f(:,15)); capacity(15)
```

Commentaire sur le second théorème de Shannon. Le résultat remarquable établi par Shannon garantit que pour $R < C$ il existe un code dont la probabilité d'erreur est aussi faible que souhaité pour peu que la longueur des mots n soit suffisante.

Lorsque n croît, $k = nR$ croît de manière proportionnelle, cela signifie qu'il est possible de réduire la probabilité d'erreur à une valeur arbitrairement faible à rendement constant, ce qui est très différent de ce qui se produit pour un simple code à répétition pour lequel le rendement décroît lorsque la longueur des mots croît.

5.4 Exercices

Canal binaire symétrique

On écrit de l'information x binaire (deux symboles notés 0 et 1) sur un support non fiable. Lors de la relecture, on observe y qui n'est pas toujours égal à x. La probabilité pour qu'un 1 soit relu comme un 0 vaut p, on suppose que la probabilité pour qu'un 0 se transforme en 1 vaut également p. Un tel modèle aléatoire élémentaire de perturbation est appelé Canal Binaire Symétrique.[4]

On suppose que les probabilités *a priori* du 0 et du 1 dans le message (entrée du canal) sont connues, elles sont notées :

$$\begin{aligned} Pr\,(1) &= \pi \\ Pr\,(0) &= 1-\pi \end{aligned}$$

1. Calculer la probabilité d'erreur binaire moyenne (P_e) en sortie du canal.

   Dire comment cette probabilité dépend de la loi d'entrée du canal ? (Pourquoi ?)

[4] Remarque : la perturbation des valeurs stockées peut s'écrire $y = x \oplus b$ où $\oplus$ désigne la somme modulo 2 (ou exclusif) et b la perturbation binaire (loi de Bernoulli).

2. Exprimer la loi de probabilité de la sortie y en fonction de p et de π.
3. Probabilités *a posteriori*.
   (a) Quelle est la probabilité *a posteriori* $p(x=1|y=1)$?
   (b) En déduire $p(x=0|y=1)$
   (c) En déduire $p(x=0|y=0)$ et $p(x=1|y=0)$.
   (d) Pour quelle valeur de p cette probabilité *a posteriori* $p(x=1|y=1)$ est-elle identique à la probabilité *a priori* π ?
   Comment interprétez-vous ce résultat ?
   (e) Que valent les probabilités *a posteriori* $p(x=1|y=1)$ et $p(x=0|y=1)$ lorsque l'entrée est équidistribuée sur $\{0;1\}$?
   (f) Quelle est la qualité d'un système de stockage pour lequel $p=1$?
4. Calcul et maximisation de l'information mutuelle entre l'entrée et la sortie.
   (a) Exprimer l'entropie de la sortie en fonction de π et p et vérifier que cette entropie est maximale pour une loi d'entrée uniforme.
   (b) Exprimer l'entropie conditionnelle de la sortie sachant l'entrée en fonction de p.
   (c) En déduire l'information mutuelle maximale (par rapport à la loi de l'entrée) entre l'entrée et la sortie.
   (d) Comment interprétez-vous maintenant le résultat de la question 3.d. par rapport à l'information mutuelle entre l'entrée et la sortie du canal ?

Canal binaire symétrique

1. La probabilté d'erreur s'écrit :

$$\begin{aligned} P_e &= p(y=1|x=0)p(x=0) + p(y=0|x=1)p(x=1) \\ &= p\cdot(1-\pi) + p\cdot\pi \\ &= p \end{aligned}$$

   Ainsi, p représente aussi la probabilité d'erreur binaire moyenne (P_e) en sortie du canal, et ce, quelque soient les probabilités *a priori*.
   Cette propriété résulte de la symétrie du canal (Canal Binaire Symétrique).
2. La sortie du canal est binaire, sa loi de probabilité est le jeu de deux probabilités $\{p(y=0), p(y=1)\}$. On a :

$$\begin{aligned} p(y=1) &= \sum_{x\in\{0,1\}} p(y=1|x)p(x) = p\,(1-\pi) + (1-p)\,\pi \\ p(y=0) &= \sum_{x\in\{0,1\}} p(y=0|x)p(x) = (1-p)\,(1-\pi) + p\pi \end{aligned}$$

3. (a)

$$p(x=1|y=1) = \frac{p(y=1|x=1)p(x=1)}{p(y=1)} = \frac{(1-p)\pi}{p(1-\pi)+(1-p)\pi}$$

   (b)

$$p(x=0|y=1) + p(x=1|y=1) = 1$$

   d'où

$$p(x=0|y=1) = 1 - p(x=1|y=1).$$

(c) Par symétrie :

$$\begin{aligned} p(x=0|y=0) &= p(x=1|y=1) \text{ en permutant } \pi \text{ par } 1-\pi \\ p(x=1|y=0) &= p(x=0|y=1) \text{ en permutant } \pi \text{ par } 1-\pi \end{aligned}$$

(d) $p(x=1|y=1) = p(x=1)$ pour $\frac{(1-p)\pi}{p(1-\pi)+(1-p)\pi} = \pi$, c'est-à-dire $p = 1/2$.
Dans ce cas, la relecture ne change rien par rapport à la connaissance *a priori* du message, le support est totalement illisible. L'entrée et la sortie du canal sont des v.a. indépendantes.

(e)

$$\begin{aligned} p(x=1|y=1) &= \left.\frac{(1-p)\pi}{p(1-\pi)+(1-p)\pi}\right|_{\pi=\frac{1}{2}} = 1-p \\ p(x=0|y=1) &= p \end{aligned}$$

On retrouve les probabilités de transition du canal.

(f)

$$\begin{aligned} p(x=0|y=1) &= 1 \\ p(x=1|y=0) &= 1 \end{aligned}$$

Il est parfait mais il faut le savoir et inverser toutes les valeurs lues.

4. (a)

$$H(Y) = -p(y=0)\log_2\left[p(y=0)\right] - p(y=1)\log_2\left[p(y=1)\right]$$

avec

$$\begin{aligned} p(y=1) &= p\,(1-\pi) + (1-p)\,\pi \\ p(y=0) &= (1-p)\,(1-\pi) + p\pi \end{aligned}$$

La sortie suit une loi uniforme lorsque l'entrée elle-même est uniformément distribuée :

$$p\,(1-\pi) + (1-p)\,\pi = (1-p)\,(1-\pi) + p\pi \rightarrow p + \pi - 2p\pi = \frac{1}{2}$$

Cette égalité est vérifiée pour $\pi = 1/2$ quel que soit p.
Pour $\pi = 1/2$, Y est uniforme et $H(Y) = 1$

(b)

$$H(Y|X) = \sum_{x\in\{0,1\}} p(x) H(Y|X=x)$$

$H(Y|X=x) = -\sum_{y\in\{0,1\}} p(y|x)\log_2 p(y|x)$ dépend de la loi *a posteriori* or le jeu de probabilités est le même pour $x=0$ et $x=1$ (seul l'ordre change) :

$$\begin{aligned} p(y=1|x=0) &= p & \qquad p(y=1|x=1) &= 1-p \\ p(y=0|x=0) &= 1-p & \qquad p(y=0|x=1) &= p \end{aligned}$$

ainsi $H(Y|X=x)$ ne dépend pas de la loi de l'entrée x. On a :

$$\begin{aligned} H(Y|X=x) &= H(Y|X=0) = H(Y|X=1) \\ &= -p\log_2(p) - (1-p)\log_2(1-p) \end{aligned}$$

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= 1 + p\log_2(p) + (1-p)\log_2(1-p) \end{aligned}$$

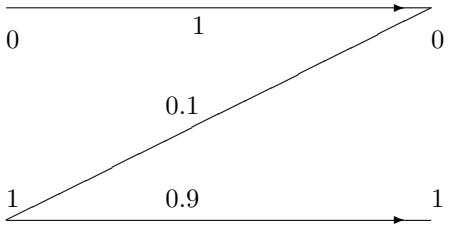
La capacité d'un canal binaire symétrique (CBS) est maximale en $p = 0$ et $p = 1$ (canal sans bruit), elle est nulle en $p = 1/2$ (canal de capacité nulle).

Le pire cas (pour la transmission) est donc $p = 1/2$ et non pas 1, valeur pour laquelle il suffit d'inverser chacune des sorties pour restaurer parfaitement l'entrée.

(d) L'information mutuelle entre l'entrée et la sortie du canal est nulle pour $p = 1/2$, ainsi la relecture n'apporte aucune information sur le message, le support est totalement illisible.

Capacité du canal en Z.

Calculer la capacité du canal défini par le diagramme de transition suivant :



Capacité du canal en Z.

La matrice de transition du canal est :

$$\mathbf{\Pi} = [\Pi_{ij}] = [p(y_j|x_i)] \quad \begin{matrix} i \in \{1 \cdots N = 2\} \\ j \in \{1 \cdots M = 2\} \end{matrix} = \begin{bmatrix} 1 & 0 \\ 0.1 & 0.9 \end{bmatrix}$$

Pour la loi de l'entrée, on note

$$\begin{aligned} p(x=1) &= p_1 \\ p(x=0) &= 1 - p_1 \end{aligned}$$

Pour la loi de la sortie, d'après la formule des probabilités totales, on a :

$$\begin{aligned} p(y=0) &= 1 \times (1-p_1) + 0.1 \times p_1 = 1 - 0.9p_1 \\ p(y=1) &= 0 \times (1-p_1) + 0.9 \times p_1 = 0.9p_1 \end{aligned}$$

L'entropie de la sortie s'écrit :

$$H(Y) = -(0.9p_1)\log_2(0.9p_1) - (1-0.9p_1)\log_2(1-0.9p_1)$$

L'entropie conditionnelle $H(Y|X)$

$$H(Y|X) = \sum_{x \in \{0;1\}} p(x) H(Y|X=x)$$

avec

$$H(Y|X=x) = -\sum_{y\in\{0;1\}} p(y|x)\log_2 p(y|x)$$

c'est-à-dire

$$H(Y|X) = -\sum_{x\in\{0;1\}}\sum_{y\in\{0;1\}} \underbrace{p(x)p(y|x)}_{p(x,y)}\log_2 p(y|x)$$

Avec les valeurs :

$$\begin{array}{lcl} x=0,y=0 & : & p(y=0|x=0)p(x=0)\log_2 p(y=0|x=0) = 1\times(1-p_1)\log_2 1 \\ x=0,y=1 & : & p(y=1|x=0)p(x=0)\log_2 p(y=1|x=0) = 0\times(1-p_1)\log_2 0 \\ x=1,y=0 & : & p(y=0|x=1)p(x=1)\log_2 p(y=0|x=1) = 0.1\times(p_1)\log_2 0.1 \\ x=1,y=1 & : & p(y=1|x=1)p(x=1)\log_2 p(y=1|x=1) = 0.9\times(p_1)\log_2 0.9 \end{array}$$

d'où

$$H(Y|X) = -\underbrace{1\times p_0\log_2 1}_{0 \text{ car } p(0,0)=1} - \underbrace{0\times p_0\log_2 0}_{0 \text{ car } p(0,1)=0} - \underbrace{p_1\times 0.1\log_2 0.1}_{p(0,1)=p(0|1)p(1)=0.1p_1} - \underbrace{p_1\times 0.9\log_2 0.9}_{p(0,1)=}$$

$$\begin{array}{rcl} H(Y|X) & = & -\underbrace{1\times p_0\log_2 1}_{0 \text{ car } p(0,0)=1} - \underbrace{0\times p_0\log_2 0}_{0 \text{ car } p(0,1)=0} - \underbrace{p_1\times 0.1\log_2 0.1}_{p(0,1)=p(0|1)p(1)=0.1p_1} - \underbrace{p_1\times 0.9\log_2 0.9}_{p(0,1)=} \\ & = & -p_1\times 0.1\log_2 0.1 - p_1\times 0.9\log_2 0.9 \\ & = & 0.469p_1 \end{array}$$

Le canal n'est pas uniforme, cette entropie conditionnelle dépend de la loi de l'entrée.

L'information mutuelle s'écrit :

$$I(X;Y) = H(Y) - H(Y|X) = -(0.9p_1)\log_2(0.9p_1) - (1-0.9p_1)\log_2(1-0.9p_1) - 0.469p_1$$

Pour maximiser l'information mutuelle, on calcule la dérivée par rapport à p_1 :

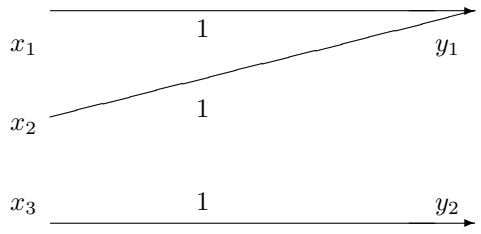
$$\frac{\partial I}{\partial p_1} = 0.9\times\log_2\left[\frac{1-0.9p_1}{0.9p_1}\right] - 0.469$$

En annulant cette dérivée, on trouve $p_1 = 0.457$ et $p_0 = 0.543$

Pour cette loi d'entré, l'information mutuelle est la capacité du canal, on trouve $C = 0.76$.

Capacité du canal déterministe.

Calculer la capacité du canal défini par le diagramme de transition suivant :



la matrice de transition est donnée par

$$\mathbf{\Pi} = [\Pi_{ij}] = [p(y_j|x_i)] \begin{array}{l} i \in \{1 \cdots 3\} \\ j \in \{1 \cdots 2\} \end{array} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Toutes les lignes contiennent le même jeu de probabilités, ce canal est uniforme en entrée et l'entropie conditionnelle se simplifie :

$$H(Y|X) = \sum_{i=1}^{3} p(x_i) \overbrace{H(Y|X=x_i)}^{\text{indépendant de } i} = H(Y|X=x_i) \overbrace{\sum_{i=1}^{3} p(x_i)}^{1 \,:\, p \text{ est une loi}} = H(Y|X=x_i), \forall i$$

Calculons cette entropie conditionnelle pour la première ligne :

$$H(Y|X=x_1) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

Pour ce canal, sachant l'entrée la sortie est certaine, le canal est déterministe. L'information mutuelle entre l'entrée est la sortie se réduit à

$$I(X;Y) = H(Y) = -p(y_2) \log_2 p(y_2) - p(y_1) \log_2 p(y_1)$$

avec

$$\begin{aligned} p(y_1) &= p(x_1) + p(x_2) \\ p(y_2) &= p(x_3) \end{aligned}$$

L'information mutuelle est maximale, égale à 1, lorsque Y est uniforme, c'est-à-dire :

$$\begin{aligned} p(x_1) + p(x_2) &= 1/2 \\ p(x_3) &= 1/2 \end{aligned}$$

Finalement, la capacité de ce canal déterministe est de 1 bit (c'est l'information maximale que peut apporter une sortie à deux états)

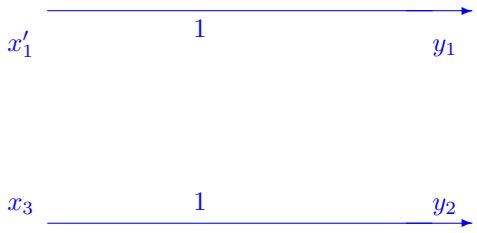
Remarques :

- Pour un canal à N entrée et M sorties, on a pour la capacité :

  $$0 \leq C \leq \log_2 \min\{N, M\}$$

  Ici $N = M = 2$ et $0 \leq C \leq \log_2 2 = 1$. Aucun canal à 2 entrées et 3 sorties ne peut avoir une capacité supérieure à 1 bit.

- En groupant les entrées x_1 et x_2 en un nouvel état x_1', ce qui revient à satisfaire l'égalité $p(x_1)+p(x_2) = 1/2$ avec le choix $p(x_2) = 0$ le canal est de la forme



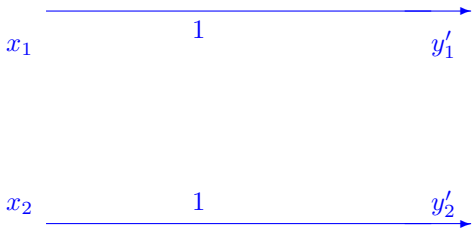
et sa capacité est clairement égale à 1.

Proposer plusieurs méthodes pour calculer la capacité du canal de matrice de transition :

$$\mathbf{\Pi} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

Canal sans équivoque

En regroupant les sorties y_1 et y_2 en y'_1 et les sorties y_3 et y_4 en y'_2, le canal est de la forme



Sa capacité est égale à 1 (obtenue pour une loi d'entrée uniforme).

Capacité et résolution de l'observation

On considère un canal sans mémoire à entrée $X \in \{-1, +1\}$ binaire et à bruit B additif quaternaire à valeurs dans $\{-1.5, -0.5, +0.5, +1.5\}$ avec les probabilités respectives $\{1/8, 3/8, 3/8, 1/8\}$. On note $Y = X + B$ la sortie du canal.

1. Donner la matrice de transition de ce canal.
2. Calculer l'entropie conditionnelle $H(Y|X)$.
3. Calculer l'information mutuelle entre l'entrée et la sortie pour une loi d'entrée uniforme. (On peut vérifier que cette loi est celle qui maximise l'information mutuelle)
4. Si l'on ne considérait que le signe $Z = \text{sign}(Y)$ de la sortie, quelle serait la capacité du canal qui relie X à Z ?

   Commenter l'intérêt de disposer d'une sortie Y plus fine que Z.

Chapter 6

Codage canal en pratique

Tout codage canal repose sur la répétition. Celle-ci permet de rendre la détection fiable et l'on sait, d'après le second théorème de Shannon, que cette fiabilité peut ne pas se faire au détriment du rendement, elle impose seulement que les mots du code soient de longueur suffisante.

Ce chapitre est consacré aux principes du codage par bloc.

F_2. Les codes abordés ici sont binaires : les mots du code sont écrits dans l'alphabet $F_2 = \{0, 1\}$. On munit F_2 de 2 opérations :

$\boxed{\oplus}$ le OU EXCLUSIF (addition modulo 2) :

$\oplus$	0	1
0	0	1
1	1	0

$\boxed{\& \textbf{ ou } \cdot}$ le ET (produit) :

&	0	1
0	0	0
1	0	1

$(F_2, \oplus)$ est une groupe abélien, $(F_2, \oplus, \cdot)$ est un corps commutatif.

6.1 Codage par bloc

Un code de rendement $R = k/n$ produit, à partir de k bits d'information, un mot de code composé de $n > k$ valeurs binaires.

Le codage est une application de F_2^k dans F_2^n qui à toute séquence binaire $U \in F_2^k$ de longueur k associe une séquence binaire $V \in F_2^n$ de longueur n.

La répartition des 2^k mots du code dans l'espace des séquences de longueur n, et plus particulièrement la distribution des distances entre mots (le spectre des distances) conditionnent les performances du code. La distance minimale entre les mots (distance minimale du code) est particulièrement importante.

Exemple. Pour le code à répétition $n = 3$ fois, le code compte $2^{k=1} = 2$ mots (000) et (111) parmi les $2^{n=3} = 8$ séquences possibles et le décodage au sens du maximum de vraisemblance s'effectue selon la règle :

- (000), (001), (010), (100): détection et correction vers (000)
- (111), (110), (101), (011): détection et correction vers (111)

La distance minimale $d_{\min}$ de ce code vaut $d_{\min} = 3$, c'est la distance de Hamming entre les mots (000) et (111).

Les séquences (000), (001), (010), (100) sont détectées de manière exacte en (000), leur distance de Hamming par rapport à (000) est inférieure ou égale à $1 = \lfloor (d_{\min} - 1)/2 \rfloor$.

Il en est de même pour les séquences (111), (110), (101), (011) par rapport au mot (111).

Ce code corrige au plus $\lfloor d_{\min} - 1)/2 \rfloor = 1$ erreur.

Un code en bloc est linéaire si les 2^k mots du code forment un sous espace vectoriel de F_2^n.

Ce sous espace vectoriel est de dimension k dans F_2^n de dimension n.

On note $C(n,k)$ un code bloc linéaire avec des mots de longueur n construits à partir de k bits informatifs.

6.2.1 Matrice génératrice.

Un code linéaire est caractérisé par sa matrice génératrice G.

Soit $\mathbf{m} \in F_2^k$ une séquence de k bits informatifs[1], on peut la décomposer dans une base $\{\mathbf{e}_i\}_{i=1..k}$ de F_2^k : $\mathbf{m} = \oplus_{i=1}^{k} m_i \mathbf{e}_i$. Pour un code linéaire, les mots sont une fonction g linéaire de $\mathbf{m}$: $g(\mathbf{m}) = \oplus_{i=1}^{k} m_i g(\mathbf{e}_i)$. Les k séquences $g(\mathbf{e}_i)$ peuvent être décomposées dans une base $\{\mathbf{e'}_j\}_{j=1..n}$ de F_2^n : $\mathbf{e}_i = \oplus_{j=1}^{n} g_{ij} \mathbf{e'}_j$. Les mots du code sont générés par :

$$\mathbf{c} = g(\mathbf{m}) = \mathbf{mG}$$

où

$$\mathbf{G} = \begin{bmatrix} g_{11} & \cdots & g_{1n} \\ \vdots & & \\ g_{k1} & \cdots & g_{kn} \end{bmatrix}$$

est appelée matrice génératrice du code.

Remarques :

- Les lignes de $\mathbf{G}$ sont des mots du code. Tout mot du code est combinaison linéaire des lignes.
- La matrice $\mathbf{G}$ n'est pas unique, elle dépend des bases choisies pour F_2^k et F_2^n.
- Il est toujours possible d'écrire la matrice génératrice sous une forme dite systématique :

  $$\mathbf{G} = \begin{bmatrix} \mathbf{I}_{k\times k} & \mathbf{P}_{k\times(n-k)} \end{bmatrix}$$

  de sorte que les mots du code s'écrivent :

  $$\mathbf{c} = \mathbf{mG} = \mathbf{m} \begin{bmatrix} \mathbf{I}_{k\times k} & \mathbf{P}_{k\times(n-k)} \end{bmatrix} = \begin{bmatrix} \mathbf{m} & \mathbf{mP} \end{bmatrix}$$

  Pour une telle forme, les k premières valeurs d'un mot sont les bits porteurs d'information tandis que les $n-k$ valeurs restantes constituent la redondance.

6.2.2 Matrice de contrôle de parité.

La matrice de contrôle de parité d'un code bloc linéaire est la matrice $(n-k)\times n$ définie par

$$\mathbf{GH}^T = \mathbf{0}_{k\times(n-k)}$$

Si $\mathbf{G}$ est sous forme systématique :

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}^T_{k\times(n-k)} & \mathbf{I}_{(n-k)\times(n-k)} \end{bmatrix}$$

En effet

$$\begin{aligned} \mathbf{GH}^T &= \begin{bmatrix} \mathbf{I}_{k\times k} & \mathbf{P}_{k\times(n-k)} \end{bmatrix} \begin{bmatrix} \mathbf{P}^T_{k\times(n-k)} & \mathbf{I}_{(n-k)\times(n-k)} \end{bmatrix}^T \\ &= \mathbf{P}_{k\times(n-k)} \oplus \mathbf{P}_{k\times(n-k)} \\ &= \mathbf{0}_{k\times(n-k)} \end{aligned}$$

Les mots de code $\mathbf{c} = \mathbf{mG}$ appartiennent au noyau de la matrice de parité : $(\mathbf{mG})\,\mathbf{H}^T = \mathbf{0}$.

[1] La coutume en codage consiste à représenter les mots par des vecteurs ligne contrairement à l'usage en algèbre.

6.2.3 Syndrome.

Il est facile de détecter des erreurs en calculant le produit $\mathbf{s} = \mathbf{r}\mathbf{H}^T$ de la séquence reçue $\mathbf{r}$ par la matrice de parité, le résultat de cette opération est appelé le syndrome, c'est une séquence de longueur $n-k$.

L'entrée du canal est un mot du code $\mathbf{c}$, la sortie diffère de l'entrée en les positions des erreurs. Notons $\mathbf{p}$ la séquence composée d'un 1 aux positions des erreurs et de 0 ailleurs. La séquence $\mathbf{r}$ en sortie du canal est la somme modulo 2 du mot $\mathbf{c}$ et de la séquence des erreurs : $\mathbf{r} = \mathbf{c} \oplus \mathbf{p}$, le calcul du syndrome donne

$$\mathbf{r}\mathbf{H}^T = [\mathbf{c} \oplus \mathbf{p}]\,\mathbf{H}^T = \mathbf{c}\mathbf{H}^T \oplus \mathbf{p}\mathbf{H}^T = \mathbf{p}\mathbf{H}^T$$

Le syndrome est la somme des colonnes de la matrice de parité d'indices égaux aux positions des erreurs.

- Si le syndrome est nul, la séquence reçue est un mot du code (pas nécessairement le bon).
- Si le syndrome est non nul, il est certain qu'une erreur s'est produite.
- Des structures algébriques adaptées permettent de remonter du syndrome aux positions les plus probables d'un certain nombre d'erreurs et ainsi de les corriger avec un fort taux de succès. Une condition nécessaire pour cela est que le nombre de configurations possibles du syndrome (2^{n-k}) soit supérieur ou égal au nombre de configurations possibles du motif des erreurs $\mathbf{p}$ que l'on veut corriger. Par exemple, si l'on veut corriger une erreur sur un mot de longueur $n = 7$, il faut $n - k \geq 3$.

6.2.4 Distance minimale

La détection et la correction de certaines configurations d'erreurs sont possibles du fait que seule une fraction des séquences binaires de longueur n appartient à l'ensemble des mots du code.

La procédure de détection qui minimise la probabilité d'erreur par mot étant la recherche du mot de code à distance minimale de la séquence observée en sortie du canal, plus les mots sont espacés les uns des autres meilleure est la capacité de correction d'un code. Lorsque le canal est peu perturbé, la probabilité d'erreur est dominée par les confusions entre les mots les plus proches ; ainsi, la distance minimale joue un rôle essentiel et constitue une caractéristique très importante d'un code.

Cette distance minimale s'écrit :

$$d_{\min} = \min_{i \neq j} d_H \ \ (\mathbf{c}_i, \mathbf{c}_j) = \min_{i \neq j} \sum_{k=1}^{n} \mathbf{c}_{i_k} \oplus \mathbf{c}_{j_k}$$

où $\mathbf{c}_{i_k}$ désigne la composante k du mot i.

Ce calcul se simplifie en remarquant que, l'espace des mots du code étant un sous espace vectoriel de F_2^n, la somme de deux mots est également un mot. Ainsi, $\mathbf{c}_{i_k} \oplus \mathbf{c}_{j_k}$ est la composante k du mot $\mathbf{z} = \mathbf{c}_i \oplus \mathbf{c}_j$.

On définit le poids de Hamming $P_{\mathbf{z}}$ d'une séquence binaire $\mathbf{z} = [z_1, \cdots, z_n]$ comme le nombre de bits à 1 dans la séquence :

$$P_{\mathbf{z}} = \sum_{k=1}^{n} z_k$$

Ainsi, pour calculer la distance minimale, il suffit de calculer le poids de Hamming de l'ensemble des mots du code et de prendre le minimum.

6.3 Quelques exemples de codes élémentaires.

Contrôle de parité $C(3,2)$**.** Partant de $k = 2$ bits informatifs, ce code ajoute un bit de parité égal à leur somme modulo 2 pour produire des mots $\left[\begin{array}{ccc} c_0 & c_1 & c_2 \end{array}\right]$ de longueur $n = 3$ tels que $c_2 = c_0 \oplus c_1$:

$$\begin{array}{ccc} 00 & \longrightarrow & 000 \\ 01 & \longrightarrow & 011 \\ 10 & \longrightarrow & 101 \\ 11 & \longrightarrow & 110 \end{array}$$

En choisissant $\mathbf{e}_0 = [10]$ et $\mathbf{e}_1 = [01]$ pour base de F_2^2 et $\mathbf{e}'_0 = [100]$, $\mathbf{e}'_1 = [010]$, $\mathbf{e}'_2 = [001]$ pour base de F_2^n, la matrice génératrice devient :

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{I}_{k\times k}=\mathbf{I}_{2\times 2}} & \overbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}^{\mathbf{P}_{k\times(n-k)}=\mathbf{P}_{2\times 1}} \end{bmatrix}$$

Pour ce code, la matrice de contrôle de parité est :

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \underbrace{\begin{bmatrix} 1 & 1 \end{bmatrix}}_{\mathbf{P}^T_{2\times 1}} & \overbrace{\begin{bmatrix} 1 \end{bmatrix}}^{\mathbf{I}_{1\times 1}} \end{bmatrix}$$

et les mots du code vérifient $\mathbf{cH}^T = \mathbf{0}$. C'est-à-dire :

$$\begin{bmatrix} c_0 & c_1 & c_2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 0 \leftrightarrow c_0 \oplus c_1 \oplus c_2 = 0 \leftrightarrow c_2 = c_0 \oplus c_1$$

Performance de ce code de contrôle de parité. Si la séquence en sortie de canal est $\mathbf{r}$, on calcule le syndrome $\mathbf{s} = \mathbf{rH}^T$ et

- Si le syndrome $\mathbf{s}$ est non nul, il est certain qu'une erreur s'est produite.
- Si le syndrome $\mathbf{s}$ est nul, $\mathbf{r}$ est un mot du code, probablement celui qui a été placé en entrée du canal mais pas certainement. Par exemple, si le mot 011 est en entrée et que 2 erreurs se produisent, l'une en première position et l'autre en deuxième position, la sortie du canal est 101 qui est aussi un mot du code. Dans une configuration de ce type, le syndrome est nul et il n'existe aucun moyen de détecter la présence d'erreurs.

6.4 Exercices

Principes du codage décodage

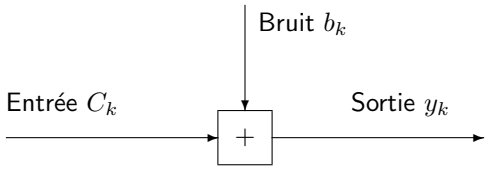
Préambule (rappel) : Soit X une variable aléatoire réelle de densité de probabilité f :

$$\Pr(X \geq a) = \int_a^{+\infty} f(t)dt = 1 - \Pr(X \leq a) \text{ et } \Pr(X \in [a,b]) = \Pr(X \in]a,b[) = \int_a^b f(t)dt.$$

La loi normale $\mathcal{N}(\mu, \sigma^2)$ de moyenne μ et de variance σ^2 a pour densité $f_{\mu,\sigma}(t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2\right\}$.

Décision optimale au sens du MV, canal gaussien

On se place en sortie d'un canal gaussien dont l'entrée C_k vaut ± 1.



A chaque utilisation canal k, l'observation y_k est la somme de l'entrée C_k et d'une perturbation b_k distribuée selon une loi normale de moyenne nulle et de variance σ^2. Les v.a. b_k sont indépendantes : le canal est sans mémoire.

On note $\widehat{C_k}$ l'hypothèse décidée pour C_k.

1. On place une seule valeur $C = c \in \{-1, +1\}$ en entrée; la sortie correspondante $Y|C = c$ est une variable aléatoire à valeur réelle. Quelle est la *vraissemblance*, *i.e.* la densité de probabilité, notée $f_c(y)$, de $Y|C = c$?

2. Les probabilités *a priori* (avant observation de y) des entrées ± 1 sont notées $\mathcal{P}_\pm$. On partitionne $\mathbb{R}$ en deux éléments Z_+ et Z_-. On décide que l'entrée était $+1$ si $y \in Z_+$ et -1 sinon.

   Ecrire la probabilité d'erreur en fonction de $\mathcal{P}_+$, Z_+ et de $f_{+1}(y)$ et $f_{-1}(y)$.

3. Quelle est la partition Z_-, Z_+ qui minimise la probabilité d'erreur ? Exprimer cette partition en fonction du rapport de vraisemblance logarithmique
$$\log \frac{f_{+1}(y)}{f_{-1}(y)}$$

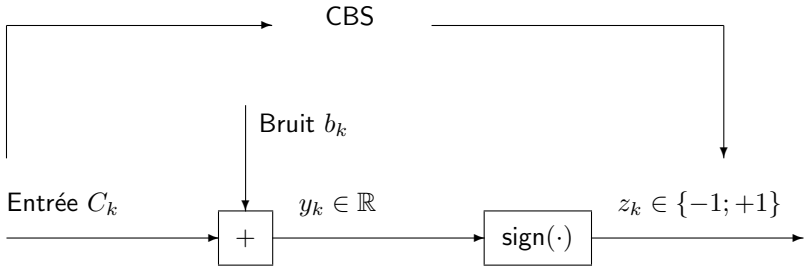
4. A partir de maintenant, on suppose la loi d'entrée uniforme $\mathcal{P}_+ = \mathcal{P}_- = 1/2$. Montrer que la règle de décision établie à la question précédente se réduit alors à $\widehat{C} = \text{sign}(y)$, c'est-à-dire que l'on décide $+1$ lorsque la valeur observée est positive, -1 si elle est négative.

5. Montrer que cette règle de décision revient à choisir celle des 2 valeurs possibles en entrée qui se trouve à distance euclidienne minimale de l'observation y.

6. On suppose maintenant que le code émis comporte 2 mots binaires de longueur n (parmi les 2^n séquences binaires possibles). On émet un mot-code binaire $\mathbf{C}$ de longueur n choisi dans ce dictionnaire de 2 mots-codes équiprobables : $\mathbf{C^0} = \left(C_0^0, \cdots, C_{n-1}^0\right)$ et $\mathbf{C^1} = \left(C_0^1, \cdots, C_{n-1}^1\right)$. Le rapport de vraisemblance logarithmique impliqué dans la définition de la règle de décision (qui minimise la probabilité d'erreur) devient
$$\log \frac{f_{\mathbf{C}^1}(\mathbf{y})}{f_{\mathbf{C}^0}((\mathbf{y}|)} \quad = \quad \frac{1}{2\sigma^2}\left[d_E(\mathbf{y}, \mathbf{C}^0) - d_E(\mathbf{y}, \mathbf{C}^1)\right]$$

   Commenter et interpréter ce résultat.

Décision optimale au sens du MV, CBS



On se place maintenant après la décision, le canal qui lie z_k à C_k est binaire. On suppose la loi d'entrée uniforme $\mathcal{P}_+ = \mathcal{P}_- = 1/2$.

1. Exprimer la probabilité de la sortie $z_k = +1$ sachant $C_k = -1$ en fonction de la variance σ^2 ; en déduire que le canal est symétrique (CBS) et donner sa probabilité de transition (*i.e.* d'erreur) p.

2. Pour réduire la probabilité d'erreur après décodage, on ajoute de la redondance en répétant $n = 2s + 1$ fois ($s \in \mathbb{N}^*$) chaque symbole en entrée du canal.

   On construit ainsi le codeur de rendement $1/n$:

$$\begin{array}{ccc} & & \overbrace{0 \cdots\cdots 0}^{n=2s+1 \text{ fois}} \\ 0 & \rightarrow & \\ 1 & \rightarrow & \underbrace{1 \cdots\cdots 1}_{n=2s+1 \text{ fois}} \end{array}$$

   (a) Montrer que la règle de décision qui minimise la probabilité d'erreur est une décision majoritaire en sortie, c'est-à-dire que le décodeur associé est :

$$\begin{array}{ccc} \overbrace{z_0 \cdots\cdots z_{2s}}^{\sum z_j < s+1} & \rightarrow & 0 \\ \underbrace{z_0 \cdots\cdots z_{2s}}_{\sum z_j \geq s+1} & \rightarrow & 1 \end{array}$$

   (b) Exprimer cette probabilité d'erreur en fonction de la taille du mot de code n (le nombre de répétition) et de la fiabilité du canal p

Principes du codage décodage

Décision optimale au sens du MV, canal gaussien

1. La vraisemblance s'écrit (en densité):

$$f_c(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{|y-c|^2}{2\sigma^2}}$$

   La loi de la sortie du canal Y est obtenue en décalant de c la moyenne de la loi du bruit.

2. $P_+ = \Pr(C = +1)$, $P_- = \Pr(C = -1)$. Sachant la sortie du canal y, on note $\Pr(\widehat{C} = +1|y)$ la probabilité de décider que l'entrée C est égale à $+1$ et $\Pr(\widehat{C} = -1|y)$ la probabilité de décider que l'entrée C est égale à -1.

   Avec ces notations, la probabilité d'erreur P_e s'écrit :

$$\begin{aligned} P_e &= \Pr(\widehat{C} = +1|C = -1)P_- + \Pr(\widehat{C} = -1|C = +1)P_+ \\ &= P_- \Pr(Y \in Z_+|C = -1) + P_+ \Pr(Y \in Z_-|C = +1) \\ &= P_- \int_{Z_+} f_{-1}(y)dy + P_+ \int_{Z_-} f_{+1}(y)dy \end{aligned}$$

   en supposant que $f_{-1}(y)$ et $f_{+1}(y)$ sont intégrables respectivement sur Z_+ et Z_- (justifié question suivante). En utilisant $Z_+ \cup Z_- = \mathbb{R}$ et $Z_+ \cap Z_- = \emptyset$:

$$P_e = P_- \int_{Z_+} f_{-1}(y)dy + P_+ \left(1 - \int_{Z_+} f_{+1}(y)dy\right)$$

   c'est-à-dire :

$$P_e = P_+ + \int_{Z_+} \{f_{-1}(y)P_- - f_{+1}(y)P_+\}\, dy$$

3. On construit la région Z_+ de sorte à minimiser la probabilité d'erreur : Z_+ est l'ensemble des y tels que $f_{-1}(y)P_- - f_{+1}(y)P_+ < 0$ d'où

$$Z_+ = \left\{ y \in \mathbb{R} : \log \frac{f_{+1}(y)}{f_{-1}(y)} > \log \frac{P_-}{P_+} \right\}$$

On a donc $y \in Z_+ \iff \frac{1}{2\sigma^2}\left((y+1)^2 - (y-1)^2\right) > \log \frac{1-P_+}{P_+} \iff y > \frac{\sigma^2}{2} \log \frac{1-P_+}{P_+}$ d'où

$$Z_+ = \left[\frac{\sigma^2}{2} \log \frac{1-P_+}{P_+}, +\infty \right[\text{ et } Z_- = \left] -\infty, \frac{\sigma^2}{2} \log \frac{1-P_+}{P_+} \right[$$

ce qui justifie l'hypothèse f_{+1} et f_{-1} intégrables sur respectivement Z_-et Z_+.
Choisir l'hypothèse de vraisemblance maximale minimise la probabilité d'erreur.

4. Pour $\mathcal{P}_+ = \mathcal{P}_- = 1/2$, la région Z_+ est telle que :

$$Z_+ = \left\{ y \in \mathbb{R} : \log \frac{f_{+1}(y)}{f_{-1}(y)} > 0 \right\}$$

Le rapport de vraisemblance logarithmique impliqué dans la définition de Z_+ s'écrit :

$$\log \frac{f_{+1}(y)}{f_{-1}(y)} = \log \frac{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{|y-1|^2}{2\sigma^2}}}{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{|y+1|^2}{2\sigma^2}}} = \frac{2}{\sigma^2} y$$

et la définition de Z_+ se réduit à $Z_+ = \{y \in \mathbb{R} : y > 0\}$, c'est-à-dire que l'on décide $+1$ lorsque $y > 0$ et -1 lorsque $y \leq 0$, autrement dit $\widehat{C} = \text{sign}(y)$.

5. Le rapport de vraisemblance logarithmique impliqué dans la définition de Z_+ s'écrit :

$$\begin{aligned} \log \frac{f_{+1}(y)}{f_{-1}(y)} &= -\frac{|y-1|^2}{2\sigma^2} + \frac{|y+1|^2}{2\sigma^2} \\ &= \frac{1}{2\sigma^2} \left\{ d_E(y,-1) - d_E(y,+1) \right\} \end{aligned}$$

On décide $+1$ lorsque $y \in Z_+$ c'est-à-dire lorsque $d_E(y,-1) - d_E(y,+1) > 0$: la distance euclidienne de y à $+1$ est plus faible que la distance de y à -1.

6. On note y_j la sortie du canal pour l'entrée j (C_j^0 ou C_j^1) et on groupe les sorties qui correspondent dans le vecteur $\mathbf{y} = [y_0, \cdots, y_{n-1}]$. On note $\mathbf{C}^{0,1}$ le vecteur pouvant prendre les valeurs $\mathbf{C}^0$ ou $\mathbf{C}^1$. La vraisemblance des observations est la densité, notée $f_{\mathbf{C}^{0,1}}$, de la variable aléatoire $\mathbf{Y}|\mathbf{C} = \mathbf{C}^{0,1}$:

$$\begin{aligned} f_{\mathbf{C}^{0,1}}(\mathbf{Y}) &= \prod_{j=0}^{n-1} \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{\left| y_j - C_j^{0,1} \right|^2}{2\sigma^2} \right] \\ &= \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \exp\left(-\frac{\sum_{j=0}^{n-1} \left| y_j - C_j^{0,1} \right|^2}{2\sigma^2} \right) \end{aligned}$$

Minimiser la probabilité d'erreur revient toujours à choisir l'hypothèse de vraisemblance maximale (même raisonnement que pour le cas scalaire des questions précédentes). La région Z_1 de décision pour l'hypothèse $\mathbf{C}^1$ est donnée par :

$$Z_1 = \left\{ \mathbf{y} \in \mathbb{R}^n : \log \frac{f_{\mathbf{C}^1}(\mathbf{y})}{f_{\mathbf{C}^0}(\mathbf{y})} > 0 \right\}$$

Le rapport de vraisemblance logarithmique se réduit toujours à la comparaison de deux distances euclidiennes :

$$\begin{aligned}\log \frac{f_{\mathbf{C}^1}(\mathbf{y})}{f_{\mathbf{C}^0}((\mathbf{y}|)} &= \frac{1}{2\sigma^2}\left[-\sum_{j=0}^{n-1}\left|y_j - C_j^1\right|^2 + \sum_{j=0}^{n-1}\left|y_j - C_j^0\right|^2\right] \\ &= \frac{1}{2\sigma^2}\left[d_E(\mathbf{y},\mathbf{C}^0) - d_E(\mathbf{y},\mathbf{C}^1)\right]\end{aligned}$$

Décision optimale au sens du MV, CBS

1.

$$\begin{aligned}\Pr(z_k = +1|C_k = -1) &= \frac{1}{\sigma\sqrt{2\pi}}\int_0^{+\infty}\exp\left[-\frac{(x+1)^2}{2\sigma^2}\right]dx \\ &= \frac{1}{\sigma\sqrt{2\pi}}\int_1^{+\infty}\exp\left[-\frac{x^2}{2\sigma^2}\right]dx \\ &= \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{-1}\exp\left[-\frac{(-x)^2}{2\sigma^2}\right]dx \\ &= \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{0}\exp\left[-\frac{(x-1)^2}{2\sigma^2}\right]dx \\ &= \Pr(z_k = -1|C_k = +1)\end{aligned}$$

le canal est symétrique de probabilité de transition $p = \frac{1}{\sigma\sqrt{2\pi}}\int_1^{+\infty}\exp\left[-\frac{x^2}{2\sigma^2}\right]dx$.

2. (a) L'entrée C est une suite de $n = 2s+1$ valeurs binaires $C_0, \cdots, C_{n-1}$ que nous groupons dans un vecteur $\boldsymbol{c}$, la distance de Hamming entre le vecteur $\boldsymbol{z} = [z_0, \cdots, z_{n-1}]^T$ composé des n valeurs binaires en sortie du canal et le vecteur $\boldsymbol{v}$ composé des valeurs binaires en entrée est le nombre de positions qui diffèrent entre ces deux vecteurs

$$d_H(\boldsymbol{z},\boldsymbol{c}) = |\{j|0 \le j < n, z_j \ne C_j\}|$$

La vraisemblance des observations $\boldsymbol{z}$ est la densité $f_{\boldsymbol{C}}$ de $\mathbf{Z}|\boldsymbol{C} = \boldsymbol{c}$ et s'écrit :

$$f_{\boldsymbol{c}}(\boldsymbol{z}) = (1-p)^n\left(\frac{p}{1-p}\right)^{d_H(\boldsymbol{z},\boldsymbol{c})}$$

d'où

$$\log f_{\boldsymbol{c}}(\boldsymbol{z}) = d_H(\boldsymbol{z},\boldsymbol{c})\log\left(\frac{p}{1-p}\right) + n\log(1-p)$$

$n\log(1-p)$ est une constante et $\log\left(\frac{p}{1-p}\right) < 0$ (car $0 < p < 1/2$), ainsi maximiser la vraisemblance revient à minimiser $d_H(\boldsymbol{z},\boldsymbol{c})$.

Une décision au sens du maximum de vraisemblance choisit le mot à distance de Hamming minimale par rapport aux observations, c'est-à-dire celui qui diffère d'elles en un nombre minimum de positions, ceci équivaut à une décision majoritaire. En effet, s'il y a plus de 0 que de 1 en sortie du canal, l'observation est plus proche du mot $0\cdots0$, à l'inverse si le nombre de 1 est supérieur au nombre de 0 l'observation est plus proche du mot $1\cdots1$.

(b) Le canal étant symétrique, on calcule la probabilité d'erreur par la probabilité pour que l'on décide le mot $\overbrace{1\cdots\cdots1}^{n=2s+1 \text{ fois}}$ alors que le mot $\overbrace{0\cdots\cdots0}^{n=2s+1 \text{ fois}}$ est en entrée du canal.
La probabilité pour que la somme $S_{2s+1}=\sum_{j=0}^{2s} z_j$ soit égale à k (entier compris entre 0 et $2s+1$) vaut :

$$\Pr\left(S_{2s+1}=k\right)=C_{2s+1}^{k}p^{k}\left(1-p\right)^{2s+1-k}$$

La probabilité d'erreur est la probabilité de décider le mot $\overbrace{1\cdots\cdots1}^{n=2s+1 \text{ fois}}$, cela se produit lorsque le nombre de 1 reçu est supérieur à $s+1$:

$$\Pr\left(S_{2s+1}\geq s+1\right)=\sum_{k=s+1}^{2s+1}C_{2s+1}^{k}p^{k}\left(1-p\right)^{2s+1-k}$$

Ainsi, un code à répétition permet de réduire autant qu'on le souhaite la probabilité d'erreur. Le prix à payer est le suivant. Supposons que le canal puisse transmettre au maximum un symbole par seconde. Initialement, le débit d'information était de 1 bit/s. Après codage répétitif, il n'est plus que de $1/n=1/(2s+1)$ bit/s. Pour réduire à une valeur arbitrairement petite la probabilité d'erreur, il a donc fallu réduire dans les mêmes proportions le débit de transmission.

Le résultat essentiel de Shannon (second théorème de Shannon) est que, dans la limite des grands mots de code, cette réduction à une valeur arbitrairement petite de la probabilité d'erreur peut être réalisée sans réduire le débit.

Codes de Hamming

Les codes de Hamming datent de 1950, ils forment une famille de codes simples et néanmoins toujours suffisants pour certaines applications telles que les mémoires RAM ECC [2].

Comme tout code bloc linéaire binaire, à une séquence de longueur k, un code de Hamming associe un mot de code de longueur $n>k$ qui comprend $m=n-k$ 'bits' de redondance. Pour un code de Hamming :

$$\begin{aligned} n &= 2^{m}-1 \\ k &= 2^{m}-m-1 \end{aligned}$$

1. Quel est le rendement du code de Hamming ?
2. Pour $k=4$, on a $n=7$ et la matrice génératrice de code Hamming $(7,4)$ est donnée par :

$$\mathbf{G}=\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}_{4,7}$$

(a) Le code défini par $\mathbf{G}$ est-il sous forme systématique ?

[2] Dans ce contexte, on utilise plutôt un code de Hamming étendu doté d'un bit de parité supplémentaire et d'une distance minimale 4, permettant 1 correction et 2 détections.

$$\mathbf{H}=\left(\begin{array}{cccc|cccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right)_{4,8} \text{ et } \mathbf{G}=\left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right)_{4,8}.$$

(b) Donner sa matrice de contrôle de parité.

3. Quels sont les mots du code ?

4. Quelle est la distance minimale de ce code ?

5. Quelle est sa capacité de détection, quelle est sa capacité de correction ?

6. En supposant qu'un mot de code subit une unique erreur, comment trouver la position de cette erreur et comment la corriger ?

Codes de Hamming

1. Le rendement vaut

$$R = k/n = 1 - \frac{m}{2^m - 1}$$

Tous les codes de Hamming ont la même capacité de correction et de détection d'erreur, le rendement approche 1 lorsque m croît. Bien sûr il n'est pas possible de choisir m arbitrairement grand car le rendement doit rester inférieur à la capacité.

2. (a) Le code est sous forme systématique $\mathbf{G} = \left[\begin{array}{cc} \mathbf{I}_{k\times k} & \mathbf{P}_{k\times(n-k)} \end{array} \right]$ avec

$$\mathbf{P} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}_{4,3}.$$

(b) Ainsi, sa matrice de contrôle de parité est donnée par :

$$\mathbf{H} = \left[\begin{array}{cc} \mathbf{P}^T_{k\times(n-k)} & \mathbf{I}_{(n-k)\times(n-k)} \end{array} \right]$$

Ici :

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}_{3,7}.$$

Le rendement vaut $R = k/n = 4/7 \approx 0.571$.

3. Les mots du code sont les combinaisons linéaires des lignes de $\mathbf{G}$ avec des coefficients de combinaison binaires : $\mathbf{c} = \mathbf{mG}$ où $\mathbf{m}$ est une séquence binaire de longueur k. Ici, $\mathbf{G}$ a 4 lignes et il y a 16 mots (1 pour chacune des séquences de longueur $k = 4$), le code étant systématique les $k = 4$ premières valeurs de chaque mot sont celles de la séquence qu'il code :

$$
\begin{array}{lcl}
0000 & \longrightarrow & 0000|000 \\
0001 & \longrightarrow & 0001|111 \\
0010 & \longrightarrow & 0010|011 \\
0011 & \longrightarrow & 0011|100 \\
0100 & \longrightarrow & 0100|101 \\
0101 & \longrightarrow & 0101|010 \\
0110 & \longrightarrow & 0110|110 \\
0111 & \longrightarrow & 0111|001 \\
1000 & \longrightarrow & 1000|110 \\
1001 & \longrightarrow & 1001|001 \\
1010 & \longrightarrow & 1010|101 \\
1011 & \longrightarrow & 1011|010 \\
1100 & \longrightarrow & 1100|011 \\
1101 & \longrightarrow & 1101|100 \\
1110 & \longrightarrow & 1110|000 \\
1111 & \longrightarrow & 1111|111
\end{array}
$$

A noter quelques mots particuliers : le mot nul et les lignes de G.

4. La distance minimale est la distance entre les deux mots les plus proches l'un de l'autre.

   Il faut calculer toutes les distances par paires ($(15 \times 16)/2 = 120$ distances) et trouver la plus petite.

   Comme les mots forment un SEV, la somme de deux mots est un mot et la distance de Hamming entre 2 mots est le poids de Hamming d'un autre mot (nombre de bits à 1 dans ce mot). La distance minimale est donc le poids du mot de poids de Hamming le plus faible (en excluant le mot 0000000 qui est la somme terme à terme modulo 2 d'un mot avec lui-même).

   D'après la liste des mots établie à la question précédente, la distance minimale est de 3 (il n'existe pas de mots avec moins de trois 1).

5. Les deux mots les plus proches sont à distance 3 : aucun motif de 2 erreurs ne peut faire passer d'un mot à un autre mot, toute combinaison de deux erreurs est détectable. Par contre il existe des motifs de 3 erreurs qui conduisent à observer en sortie du canal un mot autre que celui en entrée.

   La plus petite distance entre mots étant égale à 3, si une unique erreur se produit la séquence reçue appartient à la boule de rayon 1 et cette boule n'intersecte aucune des boules de même rayon centrées sur les autres mots. Les codes de Hamming peuvent donc corriger une seule erreur en associant les séquences reçues au mot le plus proche.

   A noter que les boules pavent l'espace : aucune séquence n'est en dehors d'une sphère, toute séquence appartient à une unique sphère.

6. Si $\mathbf{c}$ est un mot du code, le mot affecté d'une erreur est de la forme $\mathbf{c} \oplus \mathbf{p}$ où $\mathbf{p}$ est une séquence non nulle seulement à l'emplacement de l'erreur. Le produit $(\mathbf{c} \oplus \mathbf{p})\mathbf{H}^T$ (syndrome) vaut :

$$(\mathbf{c} \oplus \mathbf{p})\mathbf{H}^T = \mathbf{c}\mathbf{H}^T \oplus \mathbf{p}\mathbf{H}^T = \mathbf{p}\mathbf{H}^T$$

   Le syndome (de longueur 3) est la ligne de $\mathbf{H}^T$ (colonne de $\mathbf{H}$) qui correspond à la position de l'erreur.

La position de l'erreur étant connue et les valeurs étant binaires, il suffit de commuter la valeur de la séquence erronée en la position de l'erreur pour réaliser la correction.

Les colonnes de la matrice $\mathbf{H}$ codent en base 2 les entiers de 1 à 7

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ 6 & 5 & 3 & 7 & 4 & 2 & 1 \end{pmatrix}_{3,7} .$$

Le syndrome donne donc directement la position de l'erreur.

Pour simplifier encore, il est possible de classer les colonnes de H de 1 à 7, ainsi le syndrome donne directement la position de l'erreur.

Utilisation et performances du code de Hamming

Soit X une source simple d'alphabet $\{0, 1\}$ avec une probabilité du 1 égale à 0.135.

1. Quelques questions de révision :
   (a) Quelles sont les deux redondances possibles d'une source ? La dépendance et la non uniformité.
   (b) Quelle est l'entropie de la source X ?
   (c) Quelle est la longueur minimale des mots d'un code de compression entropique pour un codage binaire instantané de cette source ?
   (d) Comment procéder pour approcher pratiquement cette longueur moyenne minimale à l'aide d'un codage de Huffman ?
   (e) Le codage de source étant réalisé de manière à être très près de la borne, quelle est approximativement la statistique des caractères en sortie du codeur ?
   (f) On considère un Canal Binaire Symétrique (CBS) de probabilité de transition $p = 0.08$.
   Quelle est la capacité de ce canal ?

2. Dans la suite, on suppose qu'un code de Hamming $(7, 4)$ est utilisé. On souhaite étudier ce codage de Hamming dans le contexte posé par les questions précédentes.
   (a) On suppose le codage source utilisé très proche de la borne minimale théorique et l'on place un codeur canal en sortie du codeur de source. Quelle est l'entropie moyenne par symbole binaire après codage canal (n, k) ?
   (b) Quel est le rendement maximum d'un codeur canal qui assure que l'hypothèse de validité du second théorème de Shannon est vérifiée ? Que dit alors le second théorème de Shannon ?
   (c) En pratique, pourquoi est-il important que la loi soit uniforme sur l'ensemble des mots du code ?
   (d) On utilise le code de Hamming $(7, 4)$, qu'en conclure par rapport aux questions précédentes à propos de la validité du second théorème de Shannon ? Qu'en conclure par rapport à sa probabilité d'erreur par mot ?
   (e) Quelle est la probabilité pour qu'un mot de code soit décodé de manière erronée ? Evaluer numériquement cette probabilité pour $p = 0,08$, $p = 10^{-3}$ et $p = 10^{-6}$.
   Que dire sur la probabilité d'erreur binaire ?

Utilisation et performances du code de Hamming

1. Quelques questions de révision :

   (a) La dépendance et la non uniformité.

   (b) $H \approx 0.571$

   (c) $\nu \geq H$: la longueur minimale est $H \approx 0.571$.
   Evidement, un codage direct ne peut pas donner une longueur inférieure à 1 : le codage binaire d'une source binaire impose le recours au codage des extensions.

   (d) Codage des extensions de la source, plus l'ordre est élevé plus on approche de la borne.

   (e) Les valeurs 0 et 1 sont équiprobables, elles sont également indépendantes. Ceci équivaut à dire que pour un bloc de longueur k toutes les 2^k configurations possibles du bloc sont équiprobables.
   Attention, le codeur source ne doit pas troquer une forme de redondance pour une autre, par exemple en rendant la loi uniforme sur $\{0,1\}$ mais au prix de l'introduction d'une dépendance entre les symboles successifs.

   (f) $C = 1 + p\log_2 p + (1-p)\log_2(1-p) \approx 0.5978$

2. (a) En sortie du codeur source un bloc de longueur k contient k bits d'information car (i) les k v.a. sont indépendantes donc l'entropie du k-uplet est la somme des k entropies des compsantes (ii) l'entropie de chacune des composantes vaut 1 (entropie d'une loi uniforme sur deux états).
   Les 2^k mots du code sont liés aux 2^k blocs de longueur k possibles en entrée du codeur canal par une application bijective, l'entropie d'un bloc de longueur n (mots du code) en sortie du codeur vaut donc également k, c'est-à-dire que, en moyenne, il y a k/n bits d'information par symbole en sortie du codeur de canal.

   (b) L'entropie maximale d'un bloc de k valeurs en entrée du codeur canal est de k bits (c'est à peu de choses près le cas lorsque le codage source est proche de la borne inférieure entropique).
   Il faut que l'entropie moyenne par valeur binaire après le codeur canal soit inférieure à la capacité du canal C. $k/n < C$ garantit cette condition dans le pire cas (entropie maximale en entrée du codeur canal)
   Pour toute probabilité d'erreur par mot $\epsilon > 0$ fixée, il existe alors, pour une longueur de mots suffisante, un code qui assure une probabilité d'erreur après décodage plus faible que ϵ.

   (c) Cela assure que le décodage optimal, au sens de la probabilité d'erreur par mot, est donné par la recherche du mot de code à distance minimale de la séquence de longueur n observée en sortie du canal.

   (d) Son rendement est inférieur à la capacité, les conditions du second théorème de Shannon sont donc vérifiées MAIS la probabilité d'erreur n'est pas pour autant aussi faible qu'on le souhaite (il faut augmenter n à k/n constant pour faire tendre la probabilité d'erreur vers 0).
   On va la calculer dans les questions suivantes

   (e) Les jeux de plus de 1 erreur sur la taille du mot ne peuvent pas être corrigés, ainsi la probabilité de faux décodage d'un mot est la probabilité pour que le nombre d'erreurs soit au moins de 2.

   $$P_e = 1 - (1-p)^n - n\,(1-p)^{n-1}\,p$$

   Pour le code (7,4) : $P_e = 1 - (1-p)^7 - 7\,(1-p)^6\,p$.
   .
   Remarque : ce calcul donne un résultat exact car : pour le code de Hamming 16 des 128 séquences de longueur 7 sont des mots. Une sphère de rayon 1 est centrée sur chacun des mots, chaque sphère compte 8 séquences (Le mot central et 7 non mots) et $16 \times 8 = 128$: autrement dit toute séquence appartient à une unique sphère et les sphères pavent l'espace. Par conséquent, lorsqu'il y a plus d'une

erreur, on se retrouve nécessairement dans une sphère autre que celle de départ et le mot choisi est faux.

.

Insister sur l'évolution de P_e en fonction de p : pour p petit, le code permet de gagner beaucoup ($P_e = 2.10^{-5}$ pour $p = 10^{-3}$) pour $p = 0,08$ le rendement est à peine inférieur à la capacité et le gain est très faible ($P_e = 0,1$) et pour p tel que le rendement soit supérieur à la capacité le codage dégrade le taux d'erreur.

Ceci s'explique par le fait que, lorsque le rendement approche la capacité, les erreurs en nombre supérieur à la capacité de correction du code deviennent nombreuses et le code est souvent dépassé.

On ne peut pas comparer directement P_e et p : P_e est une probabilité d'erreur par mot alors que p est un taux d'erreur binaire (TEB). Choisir un mauvais mot n'implique pas qu'il existe des erreurs sur les bits d'information.

Dans le cas $p = 0,08$ le TEB est inférieur à $0,08$ et le système codé est légèrement meilleur que le système non codé.

Que ce n'est pas immédiat ... juste pour insister sur le fait que toute la théorie repose sur la probabilité d'erreur par mot et pas sur la probabilité d'erreur binaire.

Chapter 7

Quelques mots sur l'approche algorithmique

7.1 Complexité de Lempel-Ziv

On considère

- un alphabet fini A, par exemple $A = \{0; 1\}$ ou $A = \{a; b; c; \cdots z\}$.
- Une séquence composée de n caractères $s = s(1), \cdots s(n)$ avec $s(i) \in A$.

On note

- $l(s)$ la longueur de la séquence s,
- Λ la séquence vide, de longueur $l(\Lambda) = 0$.
- $s(i, j)$ la sous séquence $s(i), \cdots s(j)$ avec la convention $s(i, j) = \Lambda$ pour $i > j$

Une séquence r est un préfixe de s s'il existe $j < l(s)$ tel que $r = s(1, j)$.

7.1.1 Reproductibilité

Une séquence $s = s(1, n)$ est reproductible à partir du préfixe $s(1, j)$ (avec $1 \leq j < n$) si elle peut être obtenue par copie récursive en commençant à une position p située dans le préfixe ($1 \leq p \leq j$), autrement dit si $s(j+1, n)$ est une sous séquence de $s(1, n-1)$. Pour que cela soit possible, il faut qu'il existe une position p dans le préfixe ($p < j$) telle que $s(j + 1, n) = s(p, p + l(j + 1, n) - 1)$.

On note $s(1, j) \to s$ le fait que s est reproductible à partir de $s(1, j)$.

Exemple : $0.0.1. \to 0.0.1.0.1.0.1.0.$ Considérons la séquence

$s = 0.0.1.0.1.0.1.0$ de longueur $l(s) = 8$ et de son préfixe de longueur 3 :

$r = 0.0.1.$

On cherche à générer les 5 caractères qui suivent le préfixe (l'extension $s(4, 8)$ en commençant une copie à une position p située dans le préfixe ($1 \leq p \leq 3$).

En partant de $p = 1$: le quatrième caractère $s(4) = 0$ peut être obtenu par copie du premier $s(1)$ puisque $s(4) = 0 = s(1)$. Poursuivre signifierait obtenir $s(5) = 1$ par copie de $s(2)$, cette opération est impossible car $s(5) \neq s(2)$. s ne peut pas être générée en partant de la position $p = 1$. Le même test doit être réitéré pour toutes les positions p comprises entre 1 et 3 (dans le préfixe).

En partant de $p = 2$, on a bien

$s(4) = 0 = s(2)$, puis

$s(5) = 0 = s(3)$, ici la copie a atteind la fin du préfixe (—$s(3)$—), néanmoins elle peut se poursuivre de manière récursive en copiant s(4) (qui était déjà une copie), ainsi :

$s(6) = 0 = s(4)$, puis

$s(7) = 0 = s(5)$, puis finalement

$s(8) = 0 = s(6)$.

Il est donc possible de générer l'extension $s(4, 8)$ en débutant une copie dans le préfixe en position $p = 2$: $s(1, 3) \to s(1, 8)$.

Exemple : $0.1. \nrightarrow 0.1.1.0.$ Considérons la séquence

$s = 0.1.1.0.$ de longueur $l(s) = 4$ et de son préfixe de longueur 2 :

$r = 0.1.$

On cherche à générer les 2 caractères qui suivent le préfixe (l'extension $s(3, 4)$ en commençant une copie à une position p située dans le préfixe ($1 \leq p \leq 2$).

En partant de $p = 1$: le troisième caractère $s(3) = 1$ diffère de $s(1) = 0$, aucune copie n'est possible.

En partant de $p = 2$: le troisième caractère $s(3) = 1 = s(2)$ peut être généré par copie, autrement dit $0.1. \to 0.1.1.$.

Cependant, la copie ne peut pas se poursuivre car $s(4) = 0 \neq s(3) = 1$.

Il n'existe donc aucune position p dans le préfixe qui permette de générer la séquence complète : $s = 0.1.1.0.$ n'est pas reproductible à partir du préfixe $r = 0.1$, on note $0.1. \nrightarrow 0.1.1.0.$.

7.1.2 Productibilité

Une séquence $s = s(1, n)$ est productible à partir du préfixe $s(1, j)$ (on note $s(1, j) \Rightarrow s(1, n)$ avec $j < n$) si $s = s(1, n - 1)$ est reproductible à partir du même préfixe.

Par définition, une séquence reproductible est donc productible mais toutes les séquences productibles ne sont pas reproductibles.

Exemple : $0.1. \Rightarrow 0.1.1.0.$ Considérons à nouveau la séquence

$s = 0.1.1.0.$ de longueur $l(s) = 4$ et de son préfixe de longueur 2 :

$r = 0.1.$

Nous avons vu que $0.1. \nrightarrow 0.1.1.0.$. (non reproductible) mais $0.1. \to 0.1.1.$ donc $0.1. \Rightarrow 0.1.1.0.$.

7.1.3 Histoire exhaustive

Pour toute séquence s, le premier caractère peut être produit à partir de la séquence vide :

$\Lambda \Rightarrow s(1, 1)$

Ce processus de production peut toujours se poursuivre :

$s(1, 1) \Rightarrow s(1, l_1)$ avec $l_1 = l_{\star 1} + 1$ où $l_{\star 1}$ est le plus grand l tel que $s(1, 1) \to s(1, l)$.

$s(1, l_j) \Rightarrow s(1, l_{j+1})$ avec $l_{j+1} = l_{\star j} + 1$ où $l_\star j$ est le plus grand l tel que $s(1, l_j) \to s(1, l_{j+1})$.

Comme on a toujours $s(1, j) \Rightarrow s(1, j + 1)$, le nombre d'étape pour produire toute séquence s est inférieur ou égal à $l(s)$.

A toute séquence, on peut associer une histoire (non unique) des productions qui la génère :

$\Lambda \Rightarrow s(1, 1) \Rightarrow s(1, l_1) \cdots s(1, l_{t-1}) \Rightarrow s(1, l_t)$

Chaque histoire correspond à une décomposition de la séquence s en t sous séquences :

$s = s(1, l_1).s(l_1 + 1, l_2) \cdots s(l_{t-1} + 1, l_t)$

A chaque étape, la copie qui génère le plus long prolongement, est dite exhaustive. En choisisant à chaque étape la version exhaustive, l'histoire obtenu est appelée histoire exhaustive, c'est la décomposition la plus courte (avec le moins de composantes).

Exemple.

Le nombre de composante de l'histoire exhaustive est une mesure possible de la complexité de la séquence s.

Complexité moyenne d'une séquence i.i.d.

7.2 Codage de Lempel-Ziv

Les techniques de copier-coller évoquées ci-dessus pour établir une mesure de complexité conduisent directement à des procédures de codage.

Pour quelques détails voir la page "Complexité de Lempel-Ziv" sur fr.wikipedia.org.

Chapter 8

Chiffrement

Les probabilités et les statistiques sont des outils adaptés à beaucoup d'opérations de maniement de l'information. Donnons quelques indices par des exemples en cryptographie et en cryptanalyse.

La cryptographie a pour objectif de réécrire un message sous une forme non déchiffrable pour une personne qui ignore le secret de son chiffrage (une clef).

La cryptanalyse vise à casser le chiffrement.

Pour des codes par substitution, l'analyse des fréquences d'apparition des différents caractères qui composent le message est une méthode de cryptanalyse. Par exemple, en français, le E représente environ 17% des lettres d'un texte, suivi du A (7%), du S etc. Une estimation fiable de ces fréquences dépend de la taille et de la représentativité de l'échantillon utilisé pour cette estimation. Dans un texte court, des déviations non négligeables peuvent apparaître. Un exemple d'estimation sur un texte de petite taille est donné par la figure (??)

8.0.1 Le chiffre de César.

Le chiffre de César procède par décalage circulaire dans l'alphabet $\{A, \cdots, Z\}$. Par exemple, pour un décalage de 2 : $A \to C, B \to D \cdots Y \to A, Z \to B$.

Ce chiffre doit son nom à Jules César qui l'utilisait entre autre (avec un décalage de 3) pour écrire à Ciceron.

Ce chiffrement par substitution (chaque lettre est remplacée par une autre lettre) à distance fixe dans l'alphabet est très faible du fait du peu de clefs possibles (26). Pour le décoder, il suffit de tester les 26 décalages.

Chiffrement. Le chiffrement est une substitution monoalphabétique.

La constante utilisée pour le décalage est aussi la clef.

La méthode de déchiffrement est la même (avec décalage opposé).

Cryptanalyse La méthode brute est suffisante : essayer chacune des 26 clefs.

Le plus souvent, une analyse des fréquences des lettres du message déchiffré avec les différentes clefs suivie d'un calcul de distance avec la fréquence des lettres de la langue du message suffit à l'identification de la clef.

Quelques cas particuliers célèbres :

Décalage de 1. C'est le chiffre d'Auguste.

Décalage de 2. C'est le code Hélène (LN) : pour $L \to N$, le décalage vaut 2.

Décalage de 13. ROT13 est le nom donné au chiffre de César pour un décalage de 13. Pour l'alphabet Latin, ROT13 est son propre inverse (en décalant à nouveau de 13, on retrouve l'original, $13 + 13 = 26 = 0[26]$).

ROT13 est utilisé sur des forums pour que des solutions d'énigmes ne sautent pas aux yeux. C'est un peu l'équivalent Usenet de l'écriture tête en bas des solutions dans les revues papiers.

```
Homme libre, toujours tu chériras la mer!
La mer est ton miroir; tu contemples ton âme
Dans le déroulement infini de sa lame,
Et ton esprit n'est pas un gouffre moins amer.

Tu te plais à plonger au sein de ton image;
Tu l'embrasses des yeux et des bras, et ton coeur
Se distrait quelquefois de sa propre rumeur
Au bruit de cette plainte indomptable et sauvage.

Vous êtes tous les deux ténébreux et discrets:
Homme, nul n'a sondé le fond de tes abîmes;
Ô mer, nul ne connaît tes richesses intimes,
Tant vous êtes jaloux de garder vos secrets!

Et cependant voilà des siècles innombrables
Que vous vous combattez sans pitié ni remords,
Tellement vous aimez le carnage et la mort,
Ô lutteurs éternels, ô frères implacables!
```

```
E. 12.3 %
S. 9.8 %
T. 9.1 %
A. 6.8 %
R. 6.7 %
O. 6.7 %
N. 6.5 %
U. 5.8 %
I. 5.6 %
L. 5.1 %
M. 4.7 %
D. 3.5 %
C. 2.3 %
P. 2.1 %
B. 1.9 %
V. 1.4 %
F. 1.1 %
G. 1.1 %
H. 0.7 %
X. 0.7 %
Q. 0.5 %
Z. 0.4 %
J. 0.4 %
Y. 0.2 %

Deux lettres manquent :
K. 0 %
W. 0 %
```

Figure 8.1: Estimation de fréquences sur un texte court

8.0.2 Le chiffre de Vigenère.

Attribué à Blaise de Vigenère (diplomate français, 1523-1596), ce système de substitution polyalphabétique remplace chaque lettre par une autre (variable) obtenue à partir d'un ensemble de décalages qui constitue la clef.

Chiffrement. La clef est une suite de lettres, en utilisant l'équivalence lettre-chiffre

A	B	C	···	Z
0	1	2	···	25

, c'est aussi une suite de décalages. On chiffre la première lettre du message en lui ajoutant la première lettre de la clef (modulo 26), la deuxième lettre du message en lui ajoutant la deuxième lettre de la clef, etc. Si la longueur de la clef est inférieure à celle du message, on reprend la clef à son début après avoir utilisé son dernier caractère.

Exemple avec le message `QUEL BEAU CHIFFRE` **et la clef** `IMAG` $(8, 12, 0, 6)$

```
Message original : QUEL BEAU CHIFFRE
Clef périodisée  : IMAG IMAG IMAGIMA
Message chiffré  : YGER JQAA KTILNDE
```

En pratique, le chiffrement s'effectue avec une table dont les colonnes correspondent aux caractères à coder et les lignes aux caractères de la clef.

Déchiffrement. Il suffit de soustraire la clef au message en utilisant la même méthode que pour le chiffrement.

Cryptanalyse. Casser le chiffre de Vigenère est d'autant plus difficile que la clef est longue : lorsque la longueur de la clef est égale à celle du texte, cela est impossible, c'est le chiffre de Vernam, le seul système de cryptographie parfaitement sûr. A l'opposé, pour une clef de longueur 1, Le chiffre de Vigenère se réduit à celui de César dont la cryptanalyse est évidente. Entre ces deux extrêmes, une piste réside dans le fait que les lettres espacées de la longueur de la clef subissent le même décalage. Ainsi, si la longueur de la clef était connue, il suffirait de réaliser une analyse des fréquences (cf. cryptanalyse du chiffre de César) pour chacun des sous-messages obtenus en sous échantillonnant le message initial à la longueur de la clef. La question préalable est donc de déterminer la longueur de la clef, pour cela plusieurs pistes peuvent être explorées, par exemple :

Indice de coincidence. En français, la probabilité pour que deux lettres tirées au hasard dans un texte coïncident vaut environ 0.074 (indice de coïncidence 0,065 pour l'anglais.). En testant différentes longueurs de clef pour trouver celles pour lesquelles l'indice de coïncidence se rapproche le plus de cette valeur, on peut estimer la longueur de la clef.

Méthode de Kasiski. La méthode recherche les répétitions (typiquement d'au moins 3 lettres) dans le message chiffré, celles-ci peuvent provenir de deux occurrences d'une même séquence originale chiffrées avec la même portion de clef. L'écart entre les deux séquences est alors un multiple de la longueur de la clef.

Si l'on repère deux séquences différentes qui se répètent à distances d_1 et d_2, les diviseurs communs de d_1 et d_2 sont des longueurs possibles pour la clef.

Chapter 9

Codes convolutifs et algorithme de Viterbi

Pour un codeur convolutif, comme pour un code bloc linéaire, chaque bloc de k bits est transformé en un bloc de longueur n MAIS ces n valeurs binaires dépendent non seulement des k bits d'information en entrée mais aussi des entrées précedentes.

En pratique, cela implique que la séquence à coder est découpée en blocs de longueurs k.

9.1 Structure d'un code convolutif

Notons $\boldsymbol{I}_j = (I_{j1} \cdots I_{jk})$ le $j^{\text{ème}}$ paquet de k bits informatifs et $\boldsymbol{C}_j = (C_{j1} \cdots C_{jn})$ un paquet de n bits codés. L'opération de codage d'une suite semi infinie de bits informatifs $\boldsymbol{I} = (\boldsymbol{I}_0 \boldsymbol{I}_1 \cdots)$ s'écrit $\boldsymbol{C} = \boldsymbol{IG}$, $\boldsymbol{C}$ est la suite semi-infinie des bits codés $\boldsymbol{C} = (\boldsymbol{C}_0 \boldsymbol{C}_1 \cdots)$.

La matrice $\boldsymbol{G}$ d'un code convolutif dépend de $K = M + 1$ matrices $k \times n$ notées $\{\boldsymbol{G}_i\}_{i=0..M}$. K est appelé la longueur de contrainte du code. On a :

$$G = \begin{pmatrix} \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_M & \boldsymbol{0}_{k\times n} & \cdots \\ \boldsymbol{0}_{k\times n} & \boldsymbol{G}_0 & \cdots & \boldsymbol{G}_{M-1} & G_M & \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots \\ & \vdots & \boldsymbol{0}_{k\times n} & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \\ & & \vdots & \ddots & \boldsymbol{G}_0 & \ddots \\ \vdots & & & & \boldsymbol{0}_{k\times n} & \ddots \\ & & & & & \ddots \end{pmatrix} \quad \text{soit} \quad \begin{array}{lcl} \boldsymbol{C}_0 & = & \boldsymbol{I}_0\boldsymbol{G}_0 \\ \boldsymbol{C}_1 & = & \boldsymbol{I}_0\boldsymbol{G}_1 + \boldsymbol{I}_1\boldsymbol{G}_0 \\ \vdots & & \\ \boldsymbol{C}_M & = & \boldsymbol{I}_0\boldsymbol{G}_M + \boldsymbol{I}_1\boldsymbol{G}_{M-1} + \cdots + \boldsymbol{I}_M\boldsymbol{G}_0 \\ \vdots & & \\ \boldsymbol{C}_j & = & \boldsymbol{I}_{j-M}\boldsymbol{G}_M + \cdots + \boldsymbol{I}_j\boldsymbol{G}_0 \text{ pour } j \geq M \\ \vdots & & \end{array}$$

Avec la convention $\boldsymbol{I}_i = 0$ pour $i < 0$, la relation d'encodage $\boldsymbol{C} = \boldsymbol{IG}$ est une convolution :

$$\boldsymbol{C}_j = \sum_{l=0}^{M} \boldsymbol{I}_{j-l}\boldsymbol{G}_l. \tag{9.1}$$

Pour un bloc $\boldsymbol{I}$ de bits informatifs de taille finie, seuls $L < +\infty$ paquets de k bits sont non nuls : $\boldsymbol{I} = (\boldsymbol{I}_0 \cdots \boldsymbol{I}_{L-1})$ et $\boldsymbol{C} = (\boldsymbol{C}_0 \cdots \boldsymbol{C}_{L-1+M})$. Cette séquence codée tronquée est générée par un code en bloc linéaire dont la matrice génératrice est la sous matrice[1] $kL \times n(L + M)$ des éléments situés en haut à gauche de $\boldsymbol{G}$. La structure particulière de la matrice de ce grand code conduit à une procédure de décodage particulièrement efficace.

[1] Bien que cette troncature réduise le rendement à $R' = R\frac{L}{L+M}$, le rendement est pratiquement très voisin de $R = \frac{k}{n}$ du fait que L est grand devant M. Nous appelerons toujours rendement le rapport $R = \frac{k}{n}$.

9.2 Mise en œuvre par registre à décalage

Notons $g_{\alpha\beta}^{(l)}$ l'élément générique de la matrice $\boldsymbol{G}_l$. En explicitant les n composantes $C_{j1}, \cdots, C_{jn}$ de $\boldsymbol{C}_j$ dans la relation $\boldsymbol{C}_j = \sum_{l=0}^{M} \boldsymbol{I}_{j-l}\boldsymbol{G}_l$, nous pouvons écrire :

$$\boldsymbol{C}_j = [C_{j1}, \cdots, C_{jn}] = \left[\sum_{l=0}^{M}\sum_{\alpha=1}^{k} I_{j-l,\alpha} g_{\alpha 1}^{(l)}, \cdots, \sum_{l=0}^{M}\sum_{\alpha=1}^{k} I_{j-l,\alpha} g_{\alpha n}^{(l)}\right]$$

Le bit codé $C_{j\beta} = \sum_{\alpha=1}^{k}\sum_{l=0}^{M} I_{j-l,\alpha} g_{\alpha\beta}^{(l)}$ ne dépend que de l'entrée présente $\boldsymbol{I}_j$ et des M valeurs passées de l'entrée $\boldsymbol{I}_{j-1}, \cdots, \boldsymbol{I}_{j-M}$. Ainsi, le calcul des bits codés $C_{j\beta}$ peut être réalisé pratiquement en mémorisant M valeurs passées de l'entrée dans des registres à décalage (un registre $\alpha \in 1 \cdots k$ par bit du mot de longueur k présent en entrée). Pour le registre α d'une telle construction, seules sont connectées à l'additionneur $\beta \in 1 \cdots n$ les cases mémoire pour lesquelles $g_{\alpha\beta}^{(l)} = 1$.

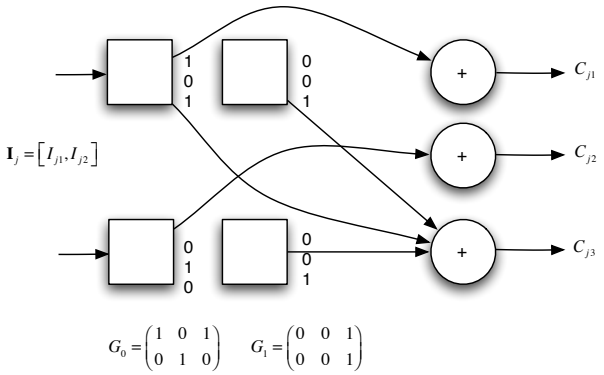


Figure 9.1: Réalisation d'un code convolutif de rendement 2/3

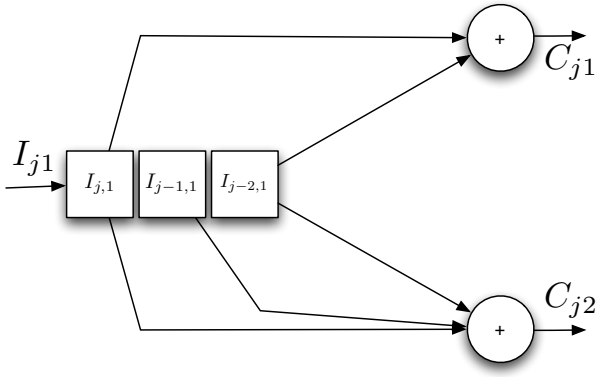


Figure 9.2: Code convolutif de rendement $1/2$ ($k = 1$, $n = 2$).

Deux exemples simples de codes convolutifs :

Code de rendement $2/3$**.** La figure (9.1) correspond à un code convolutif de rendement $2/3$ avec $k = 2$, $n = 3$, $K = 2$, $\boldsymbol{G}_0 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ et $\boldsymbol{G}_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$.

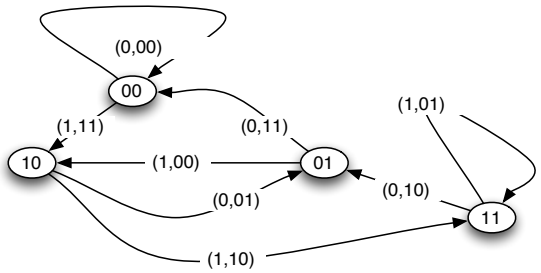


Figure 9.3: Machine à état fini associée au code convolutif de la figure (9.2).

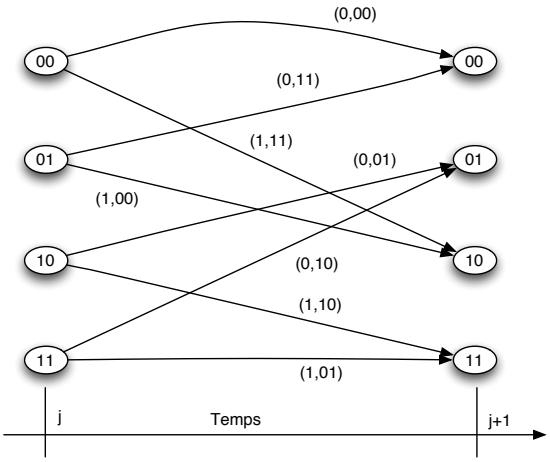


Figure 9.4: Couches j et $j+1$ du treillis associé au code de la figure (9.2).

Code de rendement $1/2$**.** Dans la suite de ce chapitre, nous utilisons le code représenté figure (9.2). Ce code est de rendement $1/2$ ($k=1$, $n=2$), il est défini par $\boldsymbol{G}_0 = \begin{pmatrix} 1 & 1 \end{pmatrix}$, $\boldsymbol{G}_1 = \begin{pmatrix} 0 & 1 \end{pmatrix}$ et $\boldsymbol{G}_2 = \begin{pmatrix} 1 & 1 \end{pmatrix}$.

9.3 Représentation par automate fini

L'état du codeur est défini par le contenu des registres à décalage. A chaque étape, les valeurs sont décalées d'une case vers la droite, le contenu de la case la plus à droite est perdue (mémoire finie de taille M) tandis qu'une nouvelle entrée $\boldsymbol{I}_j$ est affectée à la case la plus à gauche.

9.3.1 Diagramme d'état

Ainsi, le codeur peut être représenté par une machine à état fini pilotée par les bits à encoder. L'état peut prendre 2^{kM} valeurs distinctes. Sous l'action de l'une des 2^k entrées $\boldsymbol{I}_j$ possibles, chaque état à l'instant j peut évoluer vers 2^k nouvelles positions possibles à l'instant $j+1$. Les flèches des transitions entre les états seront indexées par le couple entrée-sortie $(\boldsymbol{I}_j, \boldsymbol{C}_{j+1})$.

La figure (9.3) représente le diagramme de transition de la machine à état fini associée au code de paramètre $\boldsymbol{G} = [\boldsymbol{G}_0, \boldsymbol{G}_1, \boldsymbol{G}_2] = [110111], k=1, n=2, K=3$ de la figure (9.2). L'état peut prendre $2^{kM} = 2^{1\times 2} = 4$ valeurs différentes.

9.3.2 Diagramme en treillis

Le temps n'apparaît pas dans le diagramme de transition. Le diagramme en treillis est une variante du diagramme de transition constituée d'une infinité de répliques d'un module de base représentant les valeurs possibles de l'état à un instant donné, les transitions possibles entre un instant et le suivant ainsi que les nouvelles valeurs possibles de l'état à l'instant suivant.

La figure (9.4) donne le module de base du treillis associé au code de la figure (9.2).

9.3.3 Chemins dans le treillis

A chaque séquence en entrée du codeur correspond un chemin unique dans le treillis (et inversement). Ainsi, estimer la séquence d'entrée revient à estimer un chemin dans le diagramme en treillis. L'automate part d'un état

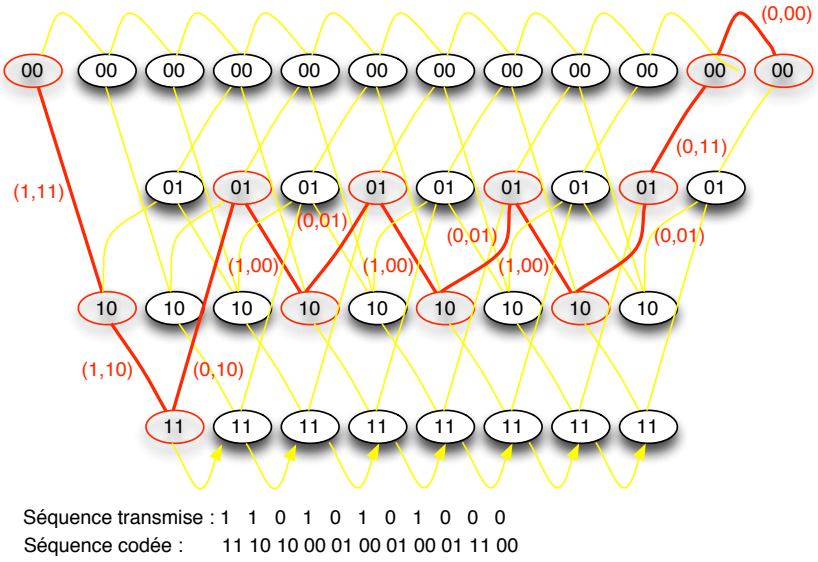


Figure 9.5: Treillis du code de la figure (9.2) pour une séquence d'entrée débutant en 00 et finissant en 00.

initial connu, parcourt le treillis en étant piloté par la séquence à encoder avant d'être finalement ramené à l'état initial.

Par exemple, pour le code de la figure (9.2) et l'état initial 00, la séquence d'entrée[2] 1 1 0 1 0 1 0 1 0 [0 0] engendre la séquence codée 11 10 10 00 01 00 01 00 01 [11 00]. Le parcours du treillis associé à cette entrée est représenté sur la figure (9.5).

9.4 Algorithme de Viterbi

Etant donnée une observation $\boldsymbol{y}$ de la sortie $\boldsymbol{C}$ du codeur entachée d'erreurs, comment est-il possible d'estimer l'entrée du canal en commettant aussi peu d'erreurs de décision que possible ?

Pour une séquence de L mots de k bits, la sortie du codeur se compose de $L+M$ mots binaires de longueur n. En supposant le canal sans mémoire, le décodeur observe $L+M$ paquets $\boldsymbol{y}_j$ de n valeurs. Notons cette observation $\boldsymbol{y} = (\boldsymbol{y}_0, \cdots, \boldsymbol{y}_{L+M-1})$ avec $\boldsymbol{y}_j = (y_{j1}, \cdots, y_{jn})$.

9.4.1 Modèle de canal et critère de détection

Le critère du maximum de vraisemblance est celui qui minimise la probabilité d'erreur lors de la détection : la séquence $\boldsymbol{C}$ optimale est celle qui maximise $P(\boldsymbol{y}|\boldsymbol{C})$ ou de façon équivalente $\log P(\boldsymbol{y}|\boldsymbol{C})$ puisque la fonction $\log$ est monotone.

Le problème à résoudre est très simple : parmi toutes les séquences possibles $\boldsymbol{C}$, il faut trouver celle dont la vraisemblance $P(\boldsymbol{y}|\boldsymbol{C})$ est maximale ; pour cela il suffit de calculer la vraisemblance pour toutes les séquences possibles.

Malheureusement, cette approche est inutilisable en pratique du fait du trop grand nombre de séquences à tester (ce nombre croît exponentiellement avec la longueur de la séquence à coder).

[2]Notons que 2 zéros sont ajoutés à la séquence à encoder afin de ramener l'état du codeur à sa valeur initiale 00.

9.4.2 Mise en œuvre efficace par l'algorithme de Viterbi

L'algorithme de Viterbi fournit un moyen efficace d'estimer la séquence présente en entrée selon le critère du maximum de vraisemblance.

Etant donnée la bijection entre mots de code et chemin dans le treillis, la log-vraisemblance en sortie d'un canal sans mémoire factorise :

$$\log P(\boldsymbol{y}|\boldsymbol{C}) = \sum_{j=0}^{L+M-1} \log P\left(\boldsymbol{y}_j|\boldsymbol{C}_0 \cdots \boldsymbol{C}_j\right) = \sum_{j=0}^{L+M-1} \log P\left(\boldsymbol{y}_j|s_j s_{j+1}\right)$$

où $s_j s_{j+1}$ désigne la branche du treillis (entre l'état s_j au temps j et l'état s_{j+1} au temps $j+1$) associée au mot $\boldsymbol{C}_j$. La quantité $l_j(s_j, s_{j+1}) = \log P\left(\boldsymbol{y}_j|s_j s_{j+1}\right)$ est appelée métrique de branche.

Pour aller plus loin, il faut exprimer la vraisemblance en fonction du modèle de canal. Pour les deux cas classiques, canal binaire symétrique et canal gaussien, cela donne :

Canal binaire symétrique. En notant $d_H\left(\boldsymbol{y}, \boldsymbol{C}_{rs}\right)$ la distance de Hamming entre un mot reçu $\boldsymbol{y}$ et la sortie du codeur $\boldsymbol{C}_{rs}$ associée à la branche $r \to s$, on sait que maximiser la vraisemblance revient à minimiser la distance de Hamming $\sum_{j=0}^{L+M-1} d_H\left(\boldsymbol{y}_j, \boldsymbol{C}_{s_j s_{j+1}}\right)$, $l_j(s_j, s_{j+1}) = d_H\left(\boldsymbol{y}_j, \boldsymbol{C}_{s_j s_{j+1}}\right)$ est la métrique de branche adaptée au canal binaire symétrique.

Canal gaussien. De même, la métrique adaptée au canal gaussien est la distance euclidienne $l_j(s_j, s_{j+1}) = d_E\left(\boldsymbol{y}_j, \boldsymbol{C}_{s_j s_{j+1}}\right)$.

D'après ce qui précède, la vraisemblance est fonction du chemin suivi dans le treillis. Pour trouver le chemin qui maximise la vraisemblance, l'algorithme de Viterbi procède de la façon suivante.

1. Pour chaque état s de la couche j du treillis, l'algorithme calcule un chemin $p_j(s_0, s)$ débutant en s_0 et finissant en s à la couche j qui minimise la métrique cumulée (aussi appelée métrique de chemin) $w_j(s) = \sum_{l=0}^{j-1} l(s_l, s_{l+1})$.

2. Le passage d'une couche à la suivante suivante $(j \to j+1)$ s'effectue selon :

$$w_{j+1}(s) = \min_{r \in E(s)} \left[w_j(r) + l_j(r, s)\right]$$

avec $E(s)$ l'ensemble des antécédents de s.

9.5 Illustration du fonctionnement de Viterbi

A titre d'exemple, illustrons maintenant le fonctionnement de l'algorithme de Viterbi pour le codeur de la figure (9.2) dont le diagramme d'état est donné en figure (9.3).

Le codeur est initialement à l'état initial 00.

Soit la séquence informative 001.

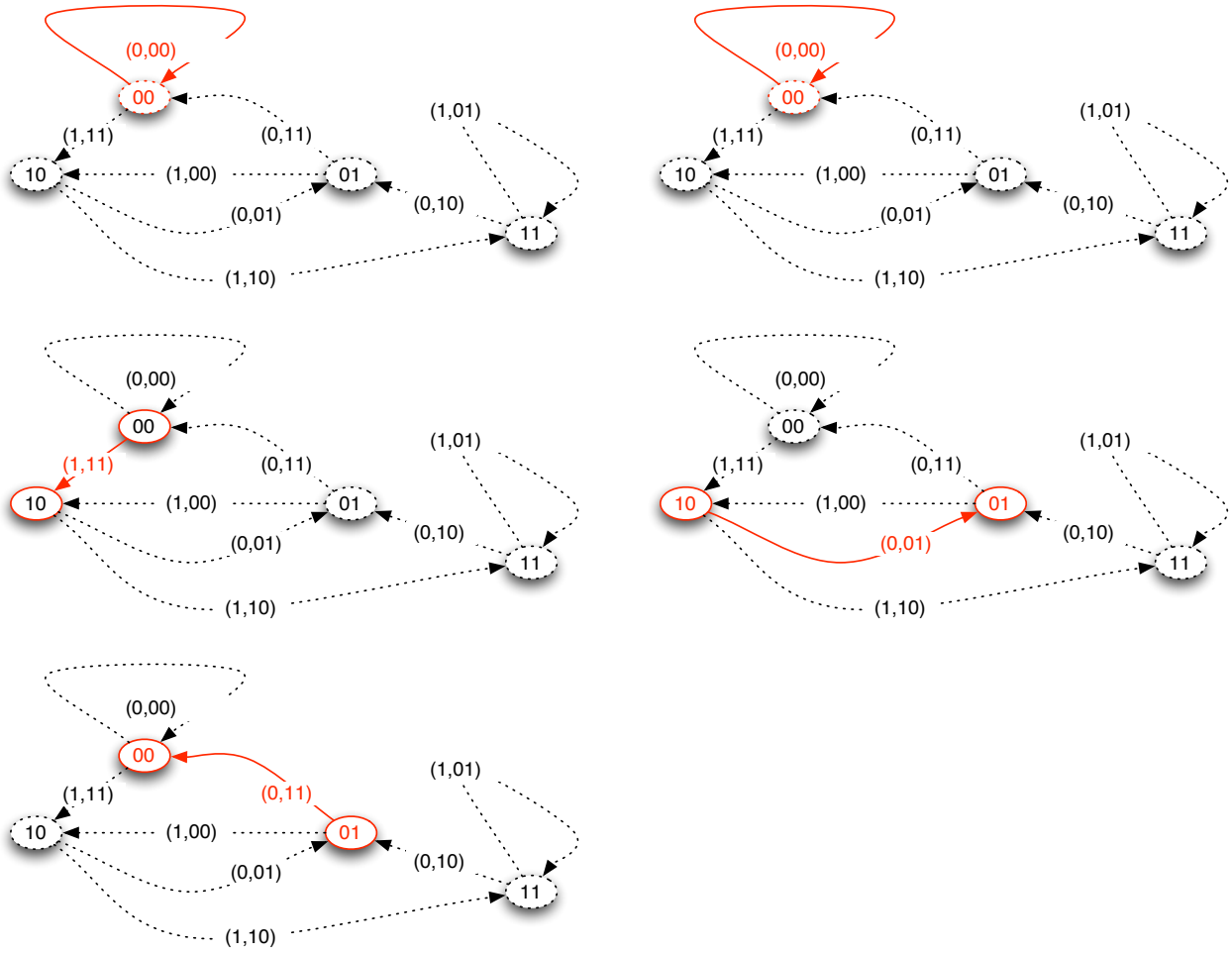
9.5.1 Codage par la machine à état fini.

La table (9.1) illustre le processus de codage par la machine à état fini.

- La première valeur qui entre dans le codeur est $\boldsymbol{I}_0 = 0$. D'après le diagramme de la figure (9.3), cette entrée active la transition indexée par le couple $(0, 00)$: l'entrée 0 engendre la sortie $\boldsymbol{C}_0 = (00)$. Cette transition fait passer de l'état 00 à l'état 00.

- De même, l'entrée suivante $(\boldsymbol{I}_1 = 0)$ active la transition $(0, 00)$ qui génère la sortie $\boldsymbol{C}_1 = (00)$ et place à nouveau le système dans l'état 00.

Table 9.1: Codage par la machine à état fini.



- Le dernier bit informatif 1 active la transition $(1, 11)$, génère la sortie $\boldsymbol{C}_2 = (11)$ et place le système dans l'état 10.

Deux zéros sont alors introduits en entrée pour ramener le codeur à l'état initial.

- Le premier zéro active la transition $(0, 01)$, génère la sortie $\boldsymbol{C}_3 = (01)$ et place à nouveau le système dans l'état 01.

- Le dernier zéro active la transition $(0, 11)$, génère la sortie $\boldsymbol{C}_4 = (11)$ et place à nouveau le système dans l'état 00.

En résumé la séquence informative 001 est codée en une séquence :

$$[\boldsymbol{C}_0, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3, \boldsymbol{C}_4] = [00, 00, 11, 01, 11]$$

9.5.2 Perturbation par le canal

Supposons que la séquence codée passe sur un canal binaire symétrique et qu'elle soit affectée par deux erreurs de transmission, de telle sorte que la séquence reçue soit :

$$[\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_4] = [10, 01, 11, 01, 11]$$

9.5.3 Décodage par l'algorithme de Viterbi

Illustrons maintenant le fonctionnement de l'algorithme de Viterbi pour cette séquence reçue. Le schéma global du fonctionnement de l'algorithme est celui de la figure (9.6).

Détaillons les différentes étapes de ce fonctionnement. Le détail du fonctionnement de l'algorithme est donnée par les tables (9.2) et (9.3).

Entrée $\boldsymbol{y}_0 = 10$. Le codeur est initialement dans son état initial 00. Pour la première transition, l'entrée de l'algorithme de Viterbi est constituée des deux premières valeurs 10 de la séquence reçue 1001110111. Deux branches sont possibles (cf. tab.(9.2) ligne 1, colonne 1) à partir de l'état 00. L'effet de chacune des deux entrées possibles est le suivant :

0 active la transition $(0, 00)$, la sortie correspondante 00 diffère en un emplacement de la séquence 10 reçue.

1 active la transition $(1, 11)$, la sortie correspondante 11 diffère en un emplacement de la séquence 10 reçue.

La métrique de branche vaut donc 1 pour chacune des branches (cf. tab.(9.2) ligne 1, colonne 2). Etant donné, qu'une seule branche aboutie à chacun des nœuds 00 et 10, leurs labels sont égaux aux métriques de chemin, elles-mêmes égales à la métrique de branche (cf. tab.(9.2) ligne 1, colonne 3).

Entrée $\boldsymbol{y}_1 = 01$. Le codeur peut maintenant partir de l'état 00 ou de l'état 10. Deux branches partent de chacun de ces états (cf. tab.(9.2) ligne 2, colonne 1), pour chaque branche l'algorithme compare l'entrée et la deuxième partie du couple qui indexe la branche et porte la métrique de branche sur le graphe (cf. tab.(9.2) ligne 2, colonne 2). La métrique de chemin est égale à la somme de cette métrique de branche avec la métrique de chemin jusqu'au nœud précédent (cf. tab.(9.2) ligne 2, colonne 3).

Entrée $\boldsymbol{y}_2 = 11$. Quatre nœuds de départ sont possibles, deux branches partent de chaque nœud (cf. tab.(9.2) ligne 3, colonne 1). Ainsi, contrairement aux cas particuliers des deux entrées précédentes, les nœuds de la couche suivante sont atteints par plusieurs branches. L'algorithme de Viterbi calcule la métrique de tous les chemins (cf. tab.(9.2) ligne 3, colonne 2) et, pour chaque nœud de la couche suivante, ne conserve que les chemins de poids minimal (cf. tab.(9.2) ligne 3, colonne 3).

Entrées suivantes. L'algorithme procède de la même manière jusqu'à la fin de la séquence reçue (cf. tab.(9.3)) pour finalement sélectionner le chemin de métrique minimale.

La lecture de message décodé est alors réalisée en suivant l'un des chemins optimaux. Ici, les index du chemin optimal sont (0,00), (0,00), (1,11), (0,01), (0,11). Les bits informatifs sont donnés par la première partie des couples, d'où le message décidé par l'algorithme de Viterbi : 00100 : les erreurs de transmissions ont été corrigées.

Remarques.

- L'algorithme impose la mémorisation des chemins survivants du début à la fin du fonctionnement. En fait, il suffit de remonter jusqu'au point où les chemins optimaux fusionnent. Ce décalage entre le point de fusion et le point de fonctionnement courant est une variable aléatoire, en pratique la séquence est estimée avec un retard fixé (3 ou 4 fois la longueur de contrainte du code) pour simplifier la mise en oeuvre tout en ayant un taux d'échec très faible.
- L'algorithme de Viterbi s'applique à toute situation dans laquelle il est nécessaire de décider de l'entrée d'une machine à état fini dont on observe une sortie bruitée, son domaine d'application est beaucoup plus large que le seul domaine du codage.

Table 9.2: Décodage par l'algorithme de Viterbi. Les branches de transition sont indexées par des couples (entrée,sortie). Les métriques des branches sont encerclées, les métriques des chemins sont placées sur les branches, le label d'un nœud correspond à la plus petite métrique de chemin jusqu'à ce nœud. Chaque colonne décrit les phases de l'algorithme pour l'entrée d'une donnée. Les lignes correspondent au message reçu : 10 01 11 (01 11)

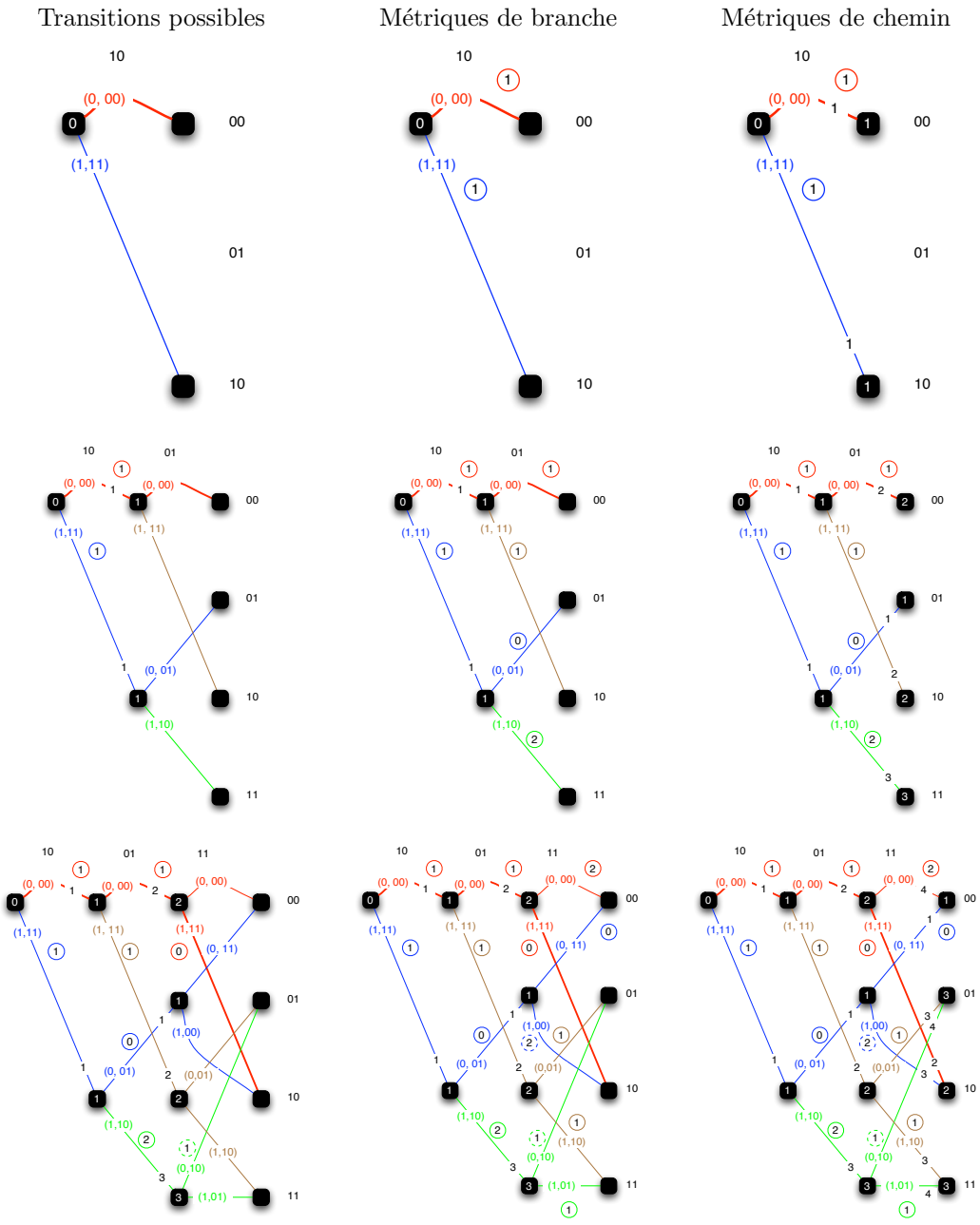
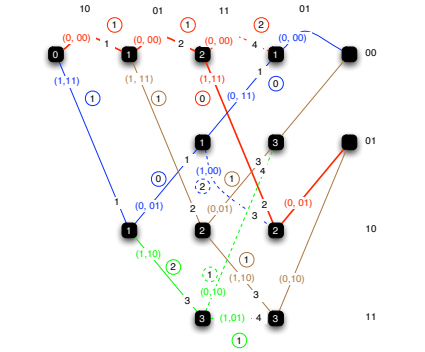
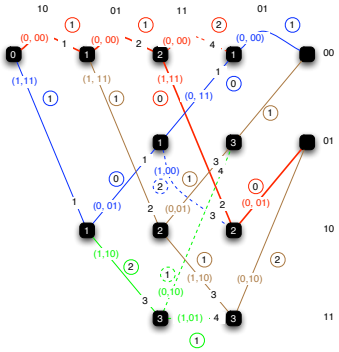


Table 9.3: Décodage par l'algorithme de Viterbi. Suite ...

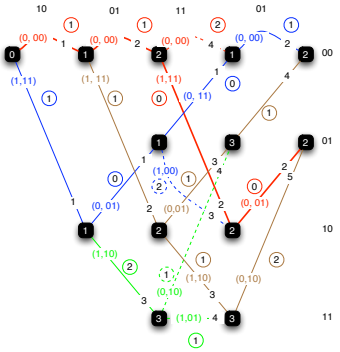
Transitions possibles

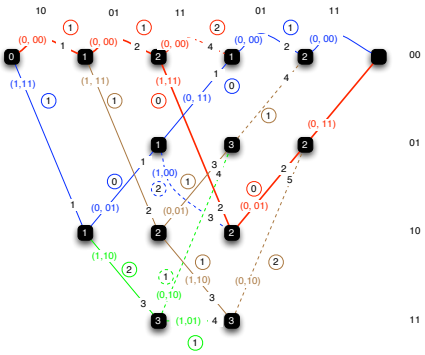


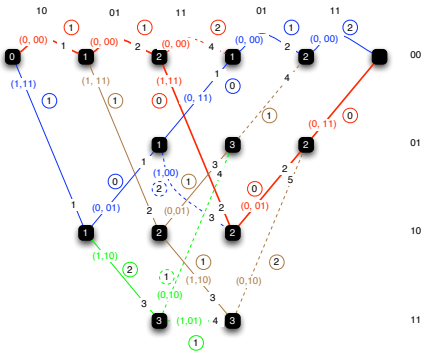
Métriques de branche

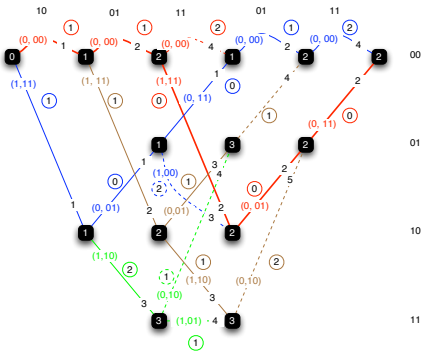


Métriques de chemin









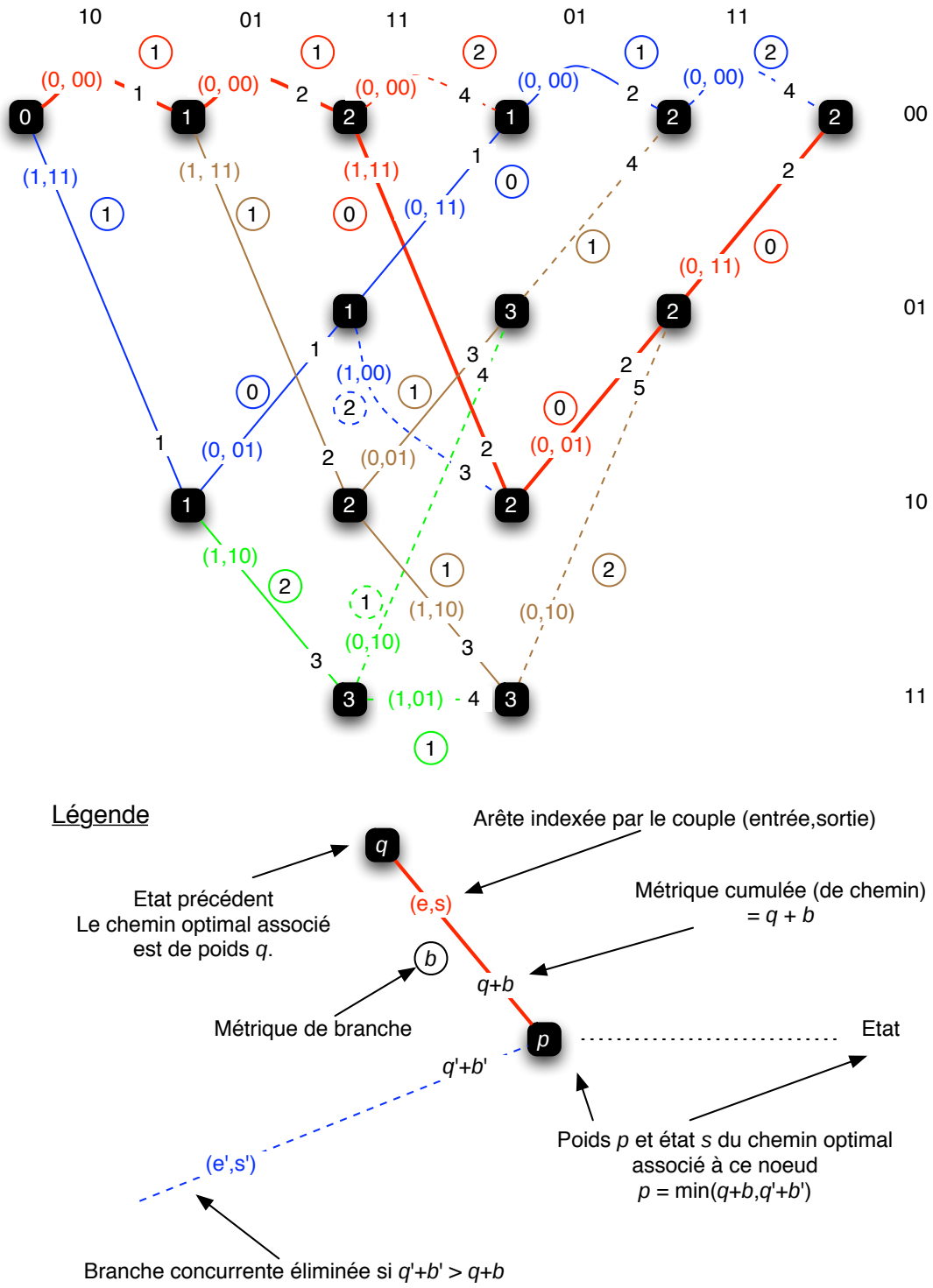


Figure 9.6: Illustration du fonctionnement de l'algorithme de Viterbi.

Chapter 10

Graphes factoriels et codes correcteurs

10.0.1 Graphes factoriels

L'idée d'une représentation graphique d'un code est déjà présente dans les travaux de Gallager (1963). Tanner (1981) a généralisé cette approche en adoptant une description par un graphe biparti d'une généralisation des codes de Gallager dans laquelle les variables (mots) sont visibles. Wiberg (1995) ajoute la possibilité de variables cachées ainsi que l'interprétation Bayésienne.

La première partie de ce chapitre présente la notion de graphe factoriel associé à une fonction multivariable produit de sous-fonctions multivariables plus simples en illustrant la manière dont le graphe encode non seulement la factorisation mais aussi l'algorithme de calcul d'une marginale. Cette approche permet d'introduire l'algorithme somme-produit pour le calcul d'une marginale unique puis de l'étendre en vue du calcul simultané de l'ensemble des marginales. Les graphes factoriels peuvent être utilisés essentiellement de deux manières dans la modélisation des systèmes, soit pour décrire un ensemble de configurations valides pour les variables d'état de ce système soit pour une description probabiliste.

La deuxième partie du chapitre illustre comment ces notions s'appliquent à la modélisation des codes et des canaux de transmission et englobent dans une même approche de nombreux algorithmes classiques tels que l'algorithme de Viterbi, l'algorithme BCJR (MAP) et l'algorithme de Kalman pour ne citer que quelques exemples.

Notion de graphe factoriel

Soit une fonction de n variables $g(x_1, \cdots, x_n) : S = A_1 \times \cdots \times A_n \to \mathbb{R}$ qui admet une factorisation de la forme :

$$g(x_1, \cdots, x_n) = \prod_{j \in J} f_j(X_j)$$

où J est une collection de sous-ensembles de variables.

Il est possible d'associer un graphe à une telle fonction en associant les facteurs (fonctions) à des noeuds de type fonction — de forme carrée sur les figures — et les variables à des noeuds de type variable — en forme de cercle sur les figures — et en connectant par une arête chaque fonction aux variables dont elle dépend.

Exemple de graphe factoriel. Le graphe factoriel associé à la fonction :

$$g(x_1, x_2, x_3, x_4) = f_a(x_1) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_2, x_4)$$

est celui de la figure (10.1).

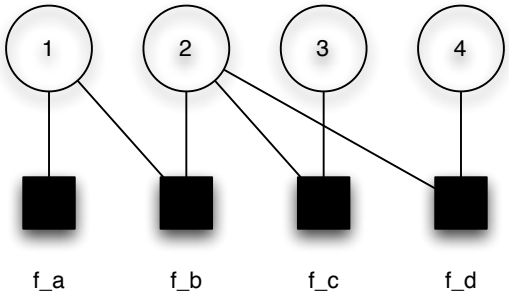


Figure 10.1: Graphe factoriel de la fonction $f_a(x_1) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_2, x_4)$.

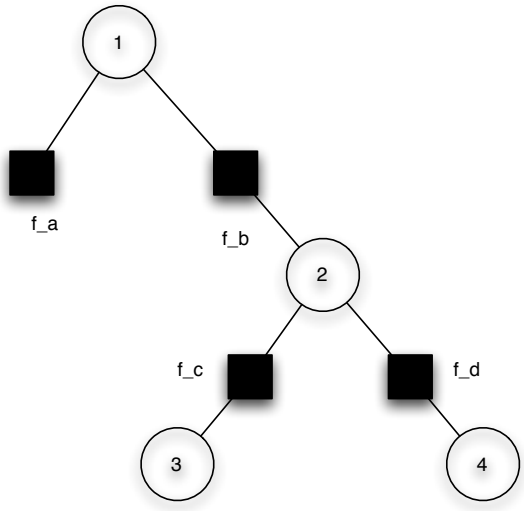


Figure 10.2: Graphe factoriel enraciné.

Marginalisation d'un produit de fonctions et arbre de calcul

Les travaux de Aji et McEliece (1997, 2000) ont, les premiers, montré que beaucoup d'algorithmes ont pour rôle essentiel de résoudre un unique problème : le calcul d'une, de plusieurs, voire de toutes les marginales d'une fonction de plusieurs variables qui admet une factorisation. Ils ont baptisé leur approche GDL (pour Generalized Distributive Law).

Une alternative très semblable est celle des graphes factoriels (factor graph). Cette approche est plus proche de l'existant et conduit à des algorithmes applicables même lorsque la solution exacte est d'une complexité rédhibitoire. Des liens très forts existent avec d'autres approches classiques telles que les champs de Markov aléatoires (MRF — Markov Random Fields) et les réseaux Bayésiens (Comme nous allons le voir, l'algorithme BP — Belief Propagation — est un cas particulier de l'algorithme SP — Sum-Product.)

Marginalisation.

Calculer une marginale d'une fonction consiste à intégrer cette fonction sur toutes ses variables sauf une :

$$g_i(x_i) = \sum_{\neg x_i} g(x_1, \cdots, x_n), i = 1 \cdots n$$

La notation $\sum_{\neg x_i}$ signifie que la sommation s'effectue sur toutes les variables sauf x_i, la marginale est parfois appelée résumé de g pour la variable x_i.

Lorsque la fonction factorise, l'algorithme de calcul de marginales se doit, pour être efficace, d'exploiter les factorisations et de réutiliser les sommes partielles. Dans ce qui suit, nous considérons que le graphe associé à la factorisation est un arbre.

Arbre de calcul.

Pour un arbre, le graphe encode la factorisation mais aussi l'algorithme de calcul des marginales. Celui-ci peut-être vu de deux manières : soit comme un algorithme récursif qui fonctionne du haut vers le bas, soit comme un calcul du bas vers le haut.

Les transformations élémentaires qui permettent de passer du graphe à l'arbre de calcul sont représentées sur la figure (10.3).

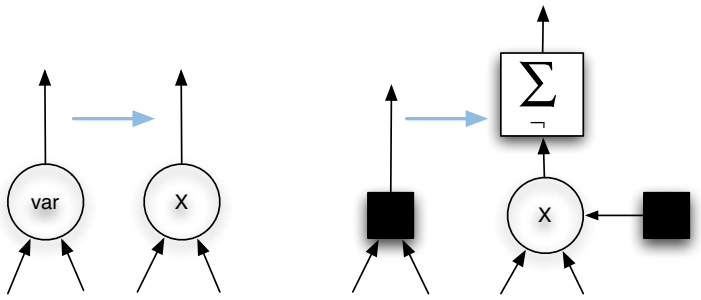


Figure 10.3: Tranformations élémentaires qui permettent de passer du graphe factoriel à l'algorithme de calcul.

L'arbre de calcul obtenu par application directe de ces transformation admet quelques simplifications courantes : les nœuds de type variable de degré deux ne font rien ; le résumé est superflu pour les fonctions de une seule variable. La figure (10.5) décrit la version simplifiée du graphe de la figure (10.4).

Exemple.

La marginale en la variable x_1 de la fonction représentée par le graphe de la figure (10.1) s'écrit :

$$\begin{aligned} g(x_1) &= \sum_{\neg x_1} f_a(x_1) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_2, x_4) \\ &= f_a(x_1) \sum_{x_2} f_b(x_1, x_2) \sum_{x_3} f_c(x_2, x_3) \sum_{x_4} f_d(x_2, x_4) \\ &= f_a(x_1) \sum_{\neg x_1} f_b(x_1, x_2) \sum_{\neg x_2} f_c(x_2, x_3) \sum_{\neg x_2} f_d(x_2, x_4) \end{aligned}$$

Pour calculer la marginale en x_1, l'arbre est tout d'abord enraciné en x_1 comme l'indique la figure (10.2). La figure (10.4) présente le résultat d'une application des transformations décrites par la figure (10.3) au graphe (10.2). Après simplification, l'arbre de calcul est celui de la figure (10.5).

10.0.2 Algorithme somme-produit.

Ce qui précède indique un moyen pratique d'exploiter la factorisation d'une fonction de plusieurs variables pour en calculer une marginale. Ce processus peut être visualisé de la manière suivante.

Vision imagée du fonctionnement : Les nœuds sont des processeurs qui communiquent entre eux par des canaux (arêtes). Les messages décrivent des marginales.

Pour un seul noeud :

Choix d'une racine : Le nœud i est pris comme racine.

Initialisation au niveau des feuilles :

- Chaque variable feuille envoie la fonction identité à son père.
- Chaque fonction feuille envoie la description de f à son père.

Propagation vers la racine : Les nœuds attendent les messages de tous leurs enfants pour calculer le message à destination du père :

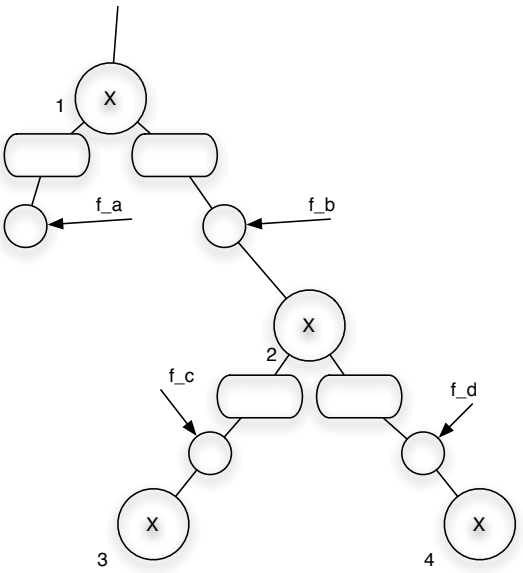


Figure 10.4: Graphe de calcul de l'expression $f_a(x_1)\, f_b(x_1, x_2)\, f_c(x_2, x_3)\, f_d(x_2, x_4)$.

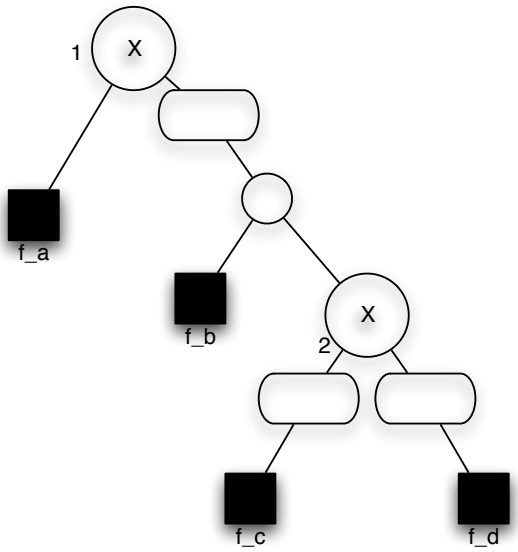


Figure 10.5: Graphe de calcul simplifié.

- Un nœud de type variable envoie le produit des messages provenant de ses enfants.
- Un nœud de type fonction envoie vers son père x le résumé en x du produit par f des messages provenant de ses enfants.

Pour plusieurs noeuds.

L'application de l'algorithme mono-nœud à l'ensemble des variables est (très) redondante. Si l'on souhaite calculer l'ensemble des marginales, le déploiement successif de l'arbre autour de chacun des noeuds, jouant successivement le rôle de la racine, conduit à calculer deux messages par arête.

Le message d'un nœud v vers une arête e est le produit de la fonction locale en v (Id pour nœud de type variable) par le résumé en e des messages reçus par les autres arêtes.

D'où une formule de calcul unique :

$$m_{f\to x}(x) = \sum_{\neg x} \left(f(X) \prod_{w\in n(f)\setminus\{x\}} m_{w\to f}(w) \right)$$
$$X = n(f)$$

qui se simplifie dans le cas d'un message allant d'une variable vers une fonction.

$$m_{x\to f}(x) = \prod_{w\in n(x)\setminus\{f\}} m_{h\to x}(x)$$

Ce fonctionnement est illustré par la figure (10.6).

La figure (10.7) illustre la chronologie des étapes du fonctionnement de l'algorithme sur un exemple simple.

10.1 Graphes et modélisation.

Deux approches essentielles et non exclusives peuvent être utilisées pour modéliser un système à l'aide d'un graphe :

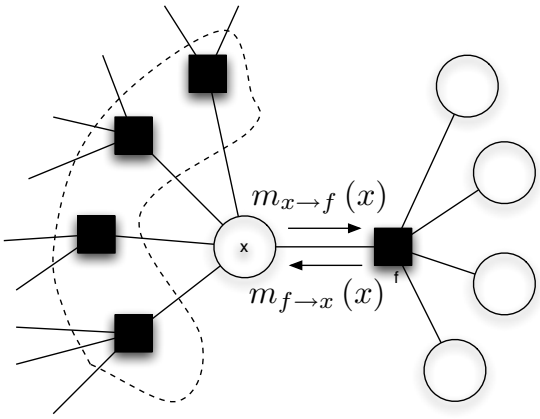


Figure 10.6: L'algorithme somme-produit propage deux messages sur chaque arête et sur un exemple simple.

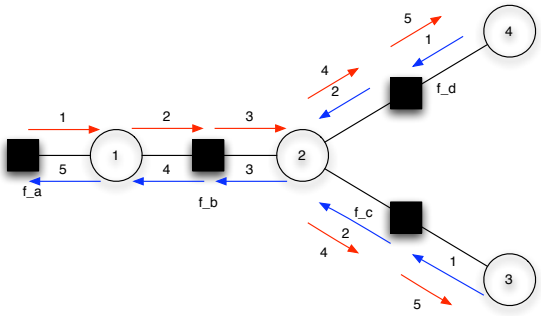


Figure 10.7: les différentes étapes du fonctionnement de l'algorithme somme-produit illustrées

$$m_{x\to f}(x) = \prod_{w\in n(x)\setminus\{f\}} m_{h\to x}(x)$$

$$m_{f\to x}(x) = \sum_{\neg x}\left(f(X)\prod_{w\in n(f)\setminus\{x\}} m_{w\to f}(w)\right)$$

Approche comportementale. Configurations valides des variables.

Approche probabiliste. Représentation d'une probabilité conjointe.

Une combinaison de ces deux approches est possible, l'exemple le plus parlant est celui du codage canal.

10.1.1 Approche comportementale.

L'approche comportementale consiste à décrire à l'aide d'un graphe l'ensemble des configurations valides des variables d'état d'un système.

Exemple du codage. Le cas des codes correcteurs est particulièrement adapté à l'illustration de cette approche : un code correcteur peut être représenté par sa fonction indicatrice :

$$1_B(x_1,\cdots,x_n) = [(x_1,\cdots,x_n)\in B]$$

B désigne l'ensemble des mots du code, l'indicatrice vaut 1 si le mot appartient au code et 0 sinon. La vérification de l'appartenance à l'ensemble des mots du code peut souvent être décomposée en la vérification d'une série de prédicats, soit, du fait de la définition de l'indicatrice du code :

$$[P_1\wedge P_2\wedge\cdots\wedge P_n] = [P_1][P_2]\cdots[P_n]$$

La figure (10.8) illustre le cas du code dont l'indicatrice se factorise comme suit :

$$1_{code}(x_1,\cdots,x_n) = [x_1\oplus x_2\oplus x_5 = 0][x_2\oplus x_3\oplus x_6 = 0][x_1\oplus x_3\oplus x_4 = 0]$$

Exemple des treillis. Le treillis déploie le fonctionnement d'une machine à état fini en fonction du temps, chacune des sections représente le diagramme d'état de la machine. Ce diagramme décrit les configurations possibles d'un ensemble de trois variables : l'état s_{i-1} dans lequel se trouve la machine instant $i-1$, son état à l'instant suivant s_i et une (ou plusieurs) variable d'entrée externe x_i. Ainsi, le comportement local est défini par une fonction de contrôle indicatrice $T_i(s_{i-1},x_i,s_i)$.

Définition du comportement par un treillis.

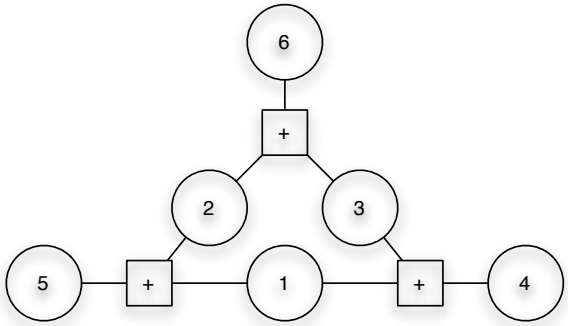


Figure 10.8: Graphes factoriel associé au code $1_{code}(x_1, \cdots, x_n) = [x_1 \oplus x_2 \oplus x_5 = 0][x_2 \oplus x_3 \oplus x_6 = 0][x_1 \oplus x_3 \oplus x_4 = 0]$.

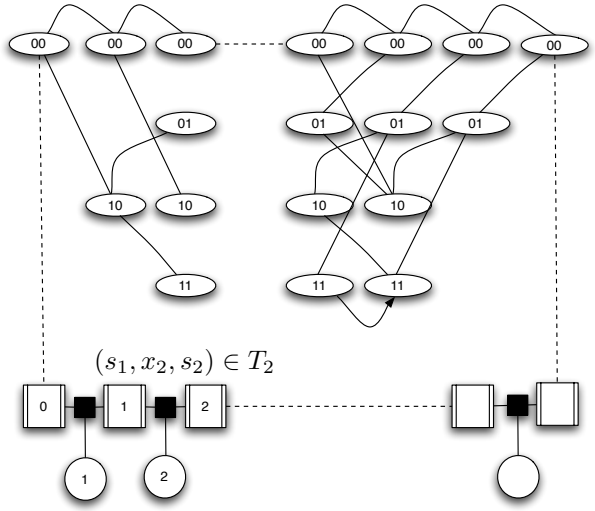


Figure 10.9: Treillis.

Section i :
- L'arête de (i) vers $(i-1)$ est une variable visible.
- Comportement local est défini par $T_i(s_{i-1}, x_i, s_i)$.
- Les contrôles sont les indicatrices des comportements locaux.

Le treillis décrit le comportement dans l'espace s,x.

10.1.2 Approche probabiliste.

Dans une approche probabiliste, le graphe est une représentation des distributions de probabilités qui exprime des indépendances et des indépendances conditionnelles. Exemples :

- Calcul des probabilités a posteriori en codage.
- Chaînes de Markov cachées.

Une loi conjointe de la forme :

$$f(x_1, x_2, \cdots, x_n)$$

peut toujours se factoriser à l'aide de la formule de Bayes :

$$f(x_1, x_2, \cdots, x_n) = \prod_{j=1}^{n} f(x_j | x_1, \cdots, x_{j-1})$$

Si $x_1, \cdots, x_n$ forment une chaîne de Markov, cette expression se simplifie considérablement :

$$f(x_1, x_2, \cdots, x_n) = \prod_{j=1}^{n} f(x_j | x_{j-1})$$

La figure (10.10) illustre les graphes associés à ces différentes possibilités.

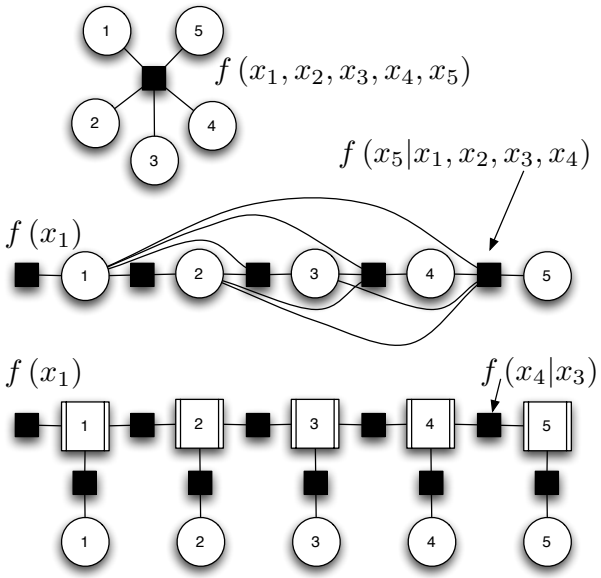


Figure 10.10: Factorisation de la densité de probabilité : forme générale (en haut), formule de Bayes (au centre) et cas markovien (en bas).

10.2 Application de l'algorithme somme-produit aux chaînes de Markov.

Quelques cas particuliers célèbres : algorithme SP en signal, IA, communication

Sur des chaînes :
- Algorithme forward/backward
- Algorithme de Viterbi (solution bidirectionnelle)
- Lisseur de Kalman

Sur les arbres :
- Algorithme Belief Propagation de Pearl.
- Décodeurs itératifs (turbo, LDPC). Instance de l'algorithme BP sur un graphe à cycles longs.
- Certains algorithmes de FFT

10.2.1 Algorithme MAP, BCJR, forward/backward.

Le problème du décodage d'un code dont le fonctionnement est décrit par un treillis se prête parfaitement à un modèle mixte comportemental/probabiliste du type de celui de la figure (10.11). Le comportement local est défini par $T_i\left(s_{i-1}, x_i, u_i, s_i\right)$, les variable du haut u_i sont les entrées, la deuxième ligne représente le caractère markovien de l'état s_i, les variables du bas sont les sorties x_i et les fonctions pendantes expriment la dépendance statistique des observations y_i par rapport aux variables de sortie x_i.

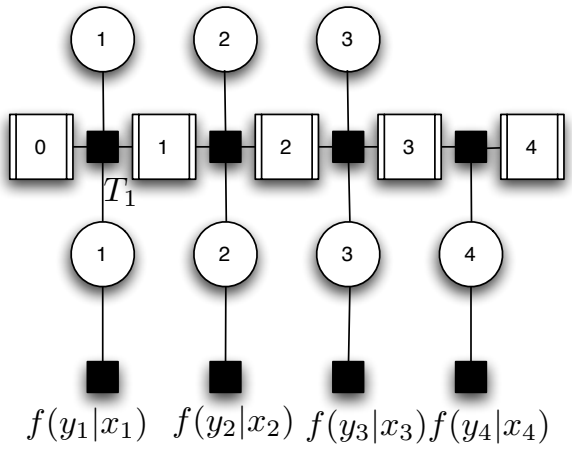


Figure 10.11: Graphe factoriel pour l'algorithme MAP.

La loi conjointe sachant l'observation :

$$g_y\left(u, s, x\right) = \prod_{i=1}^{n} T_i\left(s_{i-1}, x_i, u_i, s_i\right) \prod_{i=1}^{n} f\left(y_i|x_i\right)$$

Les probabilités *a posteriori* sont proportionnelles aux fonctions marginales :

$$p\left(u_i|y\right) \propto \sum_{\neg u_i} g_y\left(u, s, x\right)$$

il est possible d'utiliser l'algorithme somme-produit pour résoudre le problème.

En général, l'algorithme somme-produit effectue :

$$\begin{aligned} & m_{f\to x}\left(x\right) = \sum_{\neg x}\left(f\left(X\right) \prod_{w\in n(f)\setminus\{x\}} m_{w\to f}\left(w\right)\right) \\ & X = n\left(f\right) \end{aligned}$$

Les notations classiques pour ce problème sont (algorithme MAP) :

$$m_{T_i\to u_i}\left(u_i\right) = \delta\left(u_i\right) \quad m_{x_i\to T_i}\left(x_i\right) = \gamma\left(x_i\right) \quad \begin{array}{l} m_{s_i\to T_{i+1}}\left(s_i\right) = \alpha\left(s_i\right), \text{forward} \\ m_{s_i\to T_i}\left(s_i\right) = \beta\left(s_i\right), \text{backward} \end{array}$$

Sa spécialisation au cas présent donne :

$$\begin{aligned} \alpha\left(s_i\right) &= \sum_{\neg s_i} T_i\left(s_{i-1}, x_i, u_i, s_i\right) \alpha\left(s_{i-1}\right)\gamma\left(x_i\right) \\ \beta\left(s_{i-1}\right) &= \sum_{\neg s_{i-1}} T_i\left(s_{i-1}, x_i, u_i, s_i\right) \beta\left(s_i\right)\gamma\left(x_i\right) \end{aligned}$$

$$\delta\left(u_i\right) = \sum_{\neg u_i} T_i\left(s_{i-1}, x_i, u_i, s_i\right) \alpha\left(s_{i-1}\right)\gamma\left(x_i\right)$$

Etape 1 : initialisation. 1. Initialisation des feuilles.

2. Les nœuds de type variable d'ordre 2 ne font que transmettre.
3. Deux récurrences parallèles : directe et rétrograde.

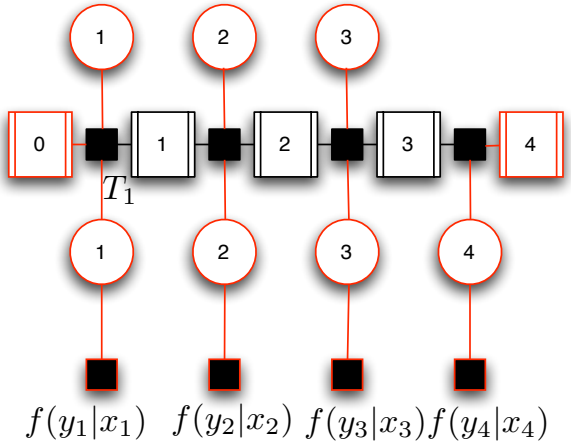


Figure 10.12: Etape 1.

Etape 2 : propagation des messages directs et rétrogrades

$$\alpha(s_i) = m_{s_i \to T_{i+1}}(s_i) = \sum_{e \in E_i(s_i)} \alpha(e)\gamma(e)$$
$$\beta(s_{i-1}) = m_{s_i \to T_i}(s_i) = \sum_{e \in E_i(s_{i-1})} \beta(e)\gamma(e)$$

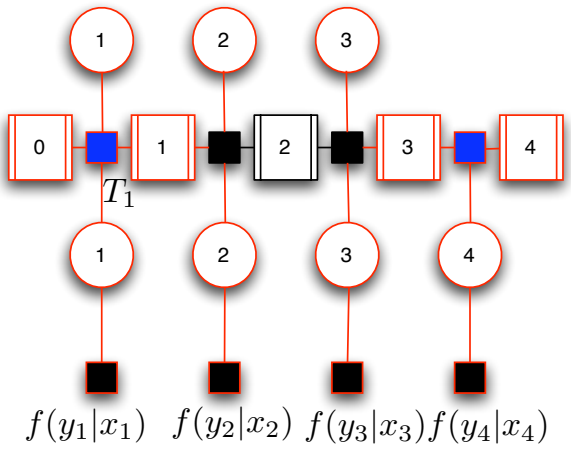


Figure 10.13: Etape 2.

Ainsi, l'algorithme somme-produit permet de retrouver le célèbre algorithme MAP utilisé en codage canal.

10.2.2 Algorithme de Viterbi.

Voyons maintenant comment l'algorithme de Viterbi lui aussi peut être obtenu de manière simple et directe à l'aide de cette même approche.

Semi-anneau Distributivité :

$$x.(y+z) = (x.y) + (x.z)$$

Exemple : max-produit Pour des valeurs réelles non négatives :

$$x\left(\max\left(y,z\right)\right) = \max\left(xy,xz\right)$$

$$\max g\left(x_1,\cdots,x_n\right) = \sum_{\neg\{\}} g\left(x_1,\cdots,x_n\right)$$

En -log produit devient somme max devient un min

10.3 Réseaux Bayésiens. Représentation parcimonieuse des lois conjointes.

Bayes : Toute loi conjointe peut être représentée par un réseau Bayésien :

$$\mathbb{P}\left(v_1,\cdots,v_N\right) = \mathbb{P}\left(v_1\right)\mathbb{P}\left(v_2|v_1\right)\cdots\mathbb{P}\left(v_N|v_1,\cdots,v_{N-1}\right)$$

Factorisation : un nœud est une v.a. de parents $P\left(\nu_i\right)$

$$\mathbb{P}\left(v_1,\cdots,v_N\right) = \mathbb{P}\left(v_1\right)\mathbb{P}\left(v_2|v_1\right)\cdots\mathbb{P}\left(v_N|v_1,\cdots,v_{N-1}\right)$$

Outils important dans de nombreux domaines : systèmes experts, codage, ... Conversion d'un réseau Bayésien en un factor graph :

1. Introduire d'un nœud facteur pour chaque terme.
2. Tracer les arêtes de ce nœud à vi et à ses parents.

Application directe de l'algorithme SP au factor graph associé.

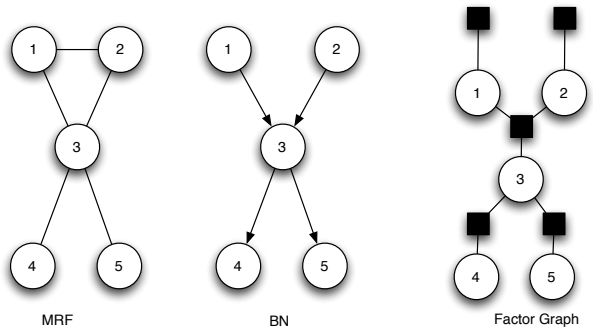


Figure 10.14: Changement de représentation sur un exemple simple : champ de Markov (à gauche), réseau Bayésien (au centre) et graphe factoriel (à droite).

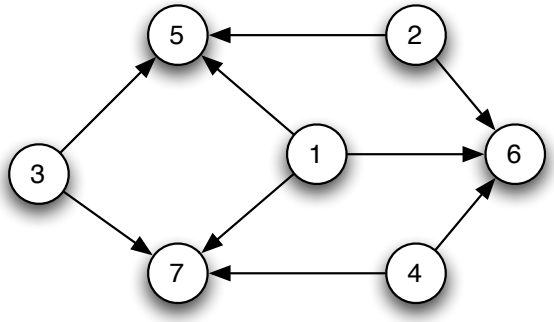


Figure 10.15: Graphe factoriel pour l'algorithme MAP.

Exemple 1 : Réseau Bayésien d'un code de Hamming. L'exemple de la figure (10.15).

Contraintes de parité :

$$\begin{cases} X_5 = X_1 \oplus X_2 \oplus X_3 \\ X_6 = X_1 \oplus X_2 \oplus X_4 \\ X_7 = X_1 \oplus X_3 \oplus X_4 \end{cases}$$

Factorisation induite :

$$\mathbb{P}(X_1, \cdots, X_7) = \mathbb{P}(X_1)\,\mathbb{P}(X_2)\,\mathbb{P}(X_3)\,\mathbb{P}(X_4) \times \mathbb{P}(X_5|X_1, X_2, X_3)\,\mathbb{P}(X_6|X_1, X_2, X_4)\,\mathbb{P}(X_7|X_1, X_3, X_4)$$

Observation en sortie d'un canal gaussien :

$$\mathbb{P}(X_1, \cdots, X_N; Y_1, \cdots, Y_N) = \mathbb{P}(X_1, \cdots, X_N) \prod_{n=1}^{N} \mathbb{P}(Y_n|X_n)$$

Exemple 2 : Forme bits-contraintes du réseau d'un code bloc linéaire. L'exemple de la figure (10.16).

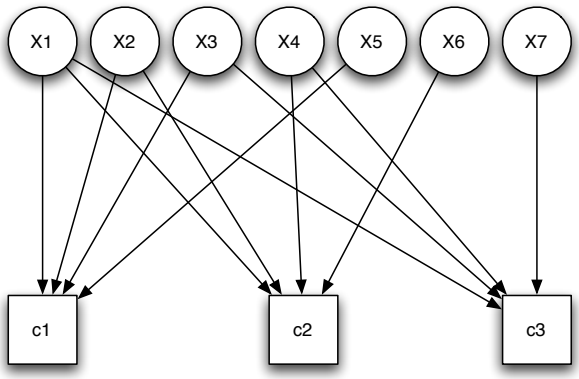


Figure 10.16: Graphe factoriel pour l'algorithme MAP.

Forme systématique

$$\mathbb{P}(X_1, \cdots, X_N) = \mathbb{P}(X_1) \cdots \mathbb{P}(X_K) \prod_{n=K+1}^{N} \mathbb{P}(X_n|P(X_n))$$

$$\mathbb{P}(X_1, \cdots, X_N) \propto \mathbb{P}(X_1) \cdots \mathbb{P}(X_N) \prod_{i=1}^{N} \mathbb{P}(c_i = 0 | P(c_i))$$

$$\mathbb{P}(c_i = 0 | P(c_i)) = \begin{cases} 1 \text{ pour } \oplus_{X_i \in P(c_i)} X_i = 0 \\ 0 \text{ sinon} \end{cases}$$

Suites de l'histoire : Le filtrage optimal et ses implantations Mixtures de gaussiennes, analyses multi-résolutions, filtrage particulaire.

Algorithmes adaptatifs sur des arbres

10.4 Turbo-codes et autres approches itératives.

Le principe "turbo" a été introduit pour le codage. Il s'agit en fait d'une approche très générale applicable à de nombreux problèmes : égalisation, synchronisation, détection multi-utilisateurs, ...

Supposons que le mot de code $X = (X_1, \cdots, X_n)$ soit transmis, le mot associé en sortie du canal s'écrit $Y = (Y_1, \cdots, Y_n)$. A chaque valeur reçue peut-être allouée une information de fiabilité $\alpha = (\alpha_1, \cdots, \alpha_n)$. Le mot transmis diffère du mot reçu d'un mot-erreur $Z^m = Y \oplus X^m$ de poids $W(Z^m) = \sum_{i=1}^{n} Z_i^m$.

Décodeur incomplet. Le décodeur incomplet trouve le mot $X^m = (X_1^m, \cdots, X_n^m)$ le plus proche de $Y = (Y_1, \cdots, Y_n)$ pour la distance de Hamming tel que $W(Z^m) \leq \left\lfloor \frac{^d\text{min}^{-1}}{2} \right\rfloor$.

- - one word if $W(Z^m) \leq \left\lfloor \frac{^d\text{min}^{-1}}{2} \right\rfloor$, sinon aucun.
- La décision est correcte si le nombre d'erreurs est inférieur à $\left\lfloor \frac{^d\text{min}^{-1}}{2} \right\rfloor$

Décodeur complet. $\min_m W(Y \oplus X^m)$ peut donner un mot de code même si le nombre d'erreurs est plus grand que $\left\lfloor \frac{^d\text{min}^{-1}}{2} \right\rfloor$.

Décodeur complet souple. $\min_m W_\alpha(Y \oplus X^m)$ où $W_\alpha(Z^m) = \sum_{i=1}^{n} \alpha_i Z_i^m$ représente le poids analogique.

Sur l'exemple de la figure (), un décodage dur donne le mot X_A.

Idée : utilisation d'un décodeur dur pour trouver un petit nombre de mots candidats et sélectionner celui qui se trouve à distance Euclidienne minimale du message reçu.

$$\Lambda(d_j) = \log\left(\frac{P(a_j=+1|R)}{P(a_j=-1|R)}\right)$$

$$P(a_j = \pm 1 | R) = \sum_{C^i \in S_j^{\pm 1}} P\left(E = C^i | R\right) \text{ avec } S_j^{\pm 1} \text{ set of word } / \; c_j^i = \pm 1$$

$$\Lambda(d_j) = \log\left(\frac{\sum_{C^i \in S_j^{+1}} P\left(R \middle| E=C^i\right)}{\sum_{C^i \in S_j^{-1}} P(R|E=C^i)}\right) \text{ avec } P\left(R \middle| E = C^i\right) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{\left|R-C^i\right|^2}{2\sigma^2}\right)$$

$$\Lambda(d_j) \approx \frac{1}{2\sigma^2}\left(\left|R - C^{-1(j)}\right|^2 - \left|R - C^{+1(j)}\right|^2\right)$$

$C^{\pm 1(j)}$ mots de codes dans $S_j^{\pm 1}$ à distance Euclidienne minimale de R

$$R = (r_1, \cdots, r_n) = E + G$$
$$E : \text{transmittedcodeword}$$
$$G : \text{gaussiannoise}$$

Concatenation of SISO (soft input soft output) algorithms can be extended to all blocks (equalization, multi-user detection, synchronization, ...)

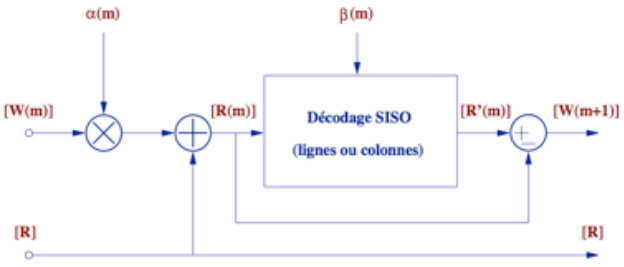


Figure 10.17: Structure pour une demi-itération du processus turbo.

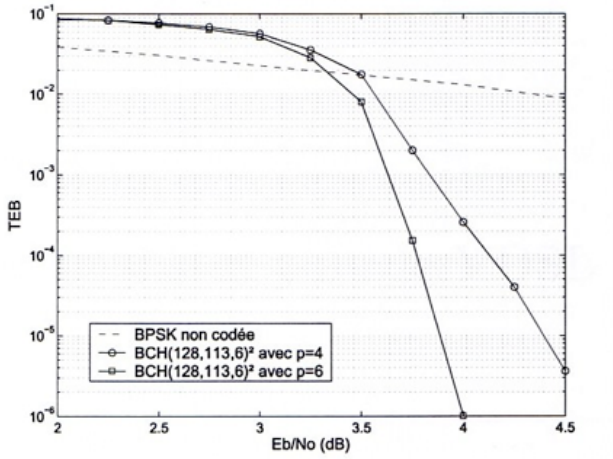


Figure 10.18: Performances d'un décodeur turbo.

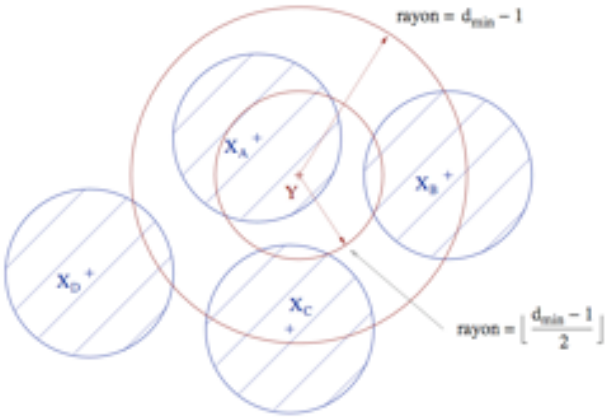


Figure 10.19: Mots trouvés par les différents décodeurs.

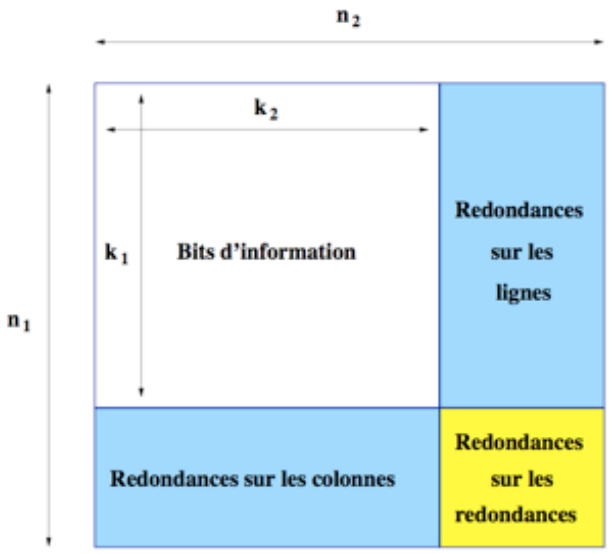


Figure 10.20: Enodage réalisé par un code produit.

Bibliography

[1] R.B. Ash. Information Theory. Dover, 1990.

[2] H. Atlan. L'organisation biologique et la théorie de l'information. La librairie du XXIème siècle - Seuil, février 2006.

[3] L. Brillouin. La science et la théorie de l'information. Paris, Masson, 1959. (Reprint, 1988, 314 p., Broché, ISBN 2-87647-036-5.) Calcul des probabilités, 1966, suivi de Introduction à la théorie de l'information, 1966 , Reprint, 1992, ISBN 2-87647-082-9.

[4] T.M. Cover, J.A. Thomas. Elements of information theory. Second edition, 2006. Wiley.

[5] G. Dubertret. Initiation à la cryptographie. Vuibert 2002 (3ème édition)

[6] D. K. Fadeev. Zum Begriff der Entropie einer endlichen Wahrseheinlichkeiitsschemas, Arbeiten zur Informationstheorie I, Berlin, Deutscher Verlag der Wissensehaften, 1957, pp. 85-90.

[7] R.P. Feynman. Leçons sur l'informatique. Odile Jacob, 1996.

[8] S. Kullback. Information Theory and Statistics. Dover, 1997

[9] D. J.C. MacKay. Information Theory, Inference, and Learning Algorithms. Copyright Cambridge University Press 2003. On-screen viewing permitted. Printing not permitted. http://www.cambridge.org/0521642981. See http://www.inference.phy.cam.ac.uk/mackay/itila/ for links.

[10] A. Rényi. Measures of information and entropy, in Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability 1960, pp. 547–561.

[11] E. Roubine. Introduction à la théorie de la communication. Tome 3 : Théorie de l'information. MASSON 1970.