Introduction à la théorie de l'information

Nicolas Sendrier

Chapitre 1

Systèmes de communication

1.1 Introduction

La théorie des communications s'intéresse aux moyens de transmettre une information depuis une source jusqu'à un utilisateur (cf. Figure 1.1). La nature de la *source* peut-être très variée. Il peut s'agir par exemple d'une voix, d'un signal électromagnétique ou d'une séquence de symboles binaires. Le *canal* peut être une ligne téléphonique, une liaison radio ou encore un support magnétique ou optique : bande magnétique ou disque compact. Le canal sera généralement perturbé par un *bruit* qui dépendra de l'environnement et de la nature canal : perturbations électriques, rayures, ... Le *codeur* représente l'ensemble des opérations effectuées sur la sortie de la source avant la transmission. Ces opérations peuvent être, par exemple, la modulation, la compression ou encore l'ajout d'une redondance pour combattre les effets du bruit. Elles ont pour but de rendre la sortie de la source compatible avec le canal. Enfin, le *décodeur* devra être capable, à partir de la sortie du canal de restituer de façon acceptable l'information fournie par la source.

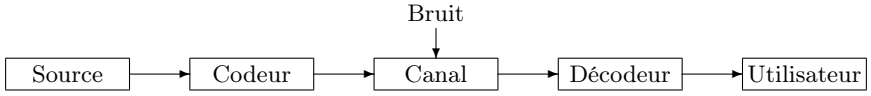


FIGURE 1.1 – Schéma d'un système de communication.

Dans les années 40, C. E. Shannon a développé une théorie mathématique appelée *théorie de l'information* qui décrit les aspects les plus fondamentaux des systèmes de communication. Cette théorie s'intéresse à la construction et à l'étude de modèles mathématiques à l'aide essentiellement de la théorie des probabilités. Depuis ce premier exposé publié en 1948 [1], la théorie de l'information s'est faite de plus en plus précise et est devenue aujourd'hui incontournable dans la conception tout système de communication, au sens le plus large de ce terme.

Dans ce cours, nous étudierons certains de ces modèles mathématiques, qui, bien que considérablement plus simple que les sources et les canaux physiques, permettent de donner une bonne intuition de leur comportement.

Dans le but de simplifier l'étude des systèmes de communication, nous étudierons séparément les modèles de sources et les modèles de canaux. Ceci peut se schématiser en séparant le codeur et le décodeur de la Figure 1.2 en deux parties. Le but du codeur de source est de représenter la sortie de la source, ou information, en une séquence binaire, et cela de la façon la plus économique possible. Le but du codeur de canal et de son décodeur est de reproduire le plus fidèlement possible cette séquence binaire malgré le passage à travers le canal bruité.

1. C. E. Shannon, *A Mathematical Theory of Communication*, Bell System Technical Journal, vol. 27, pages 379–423 et 623–656, 1948

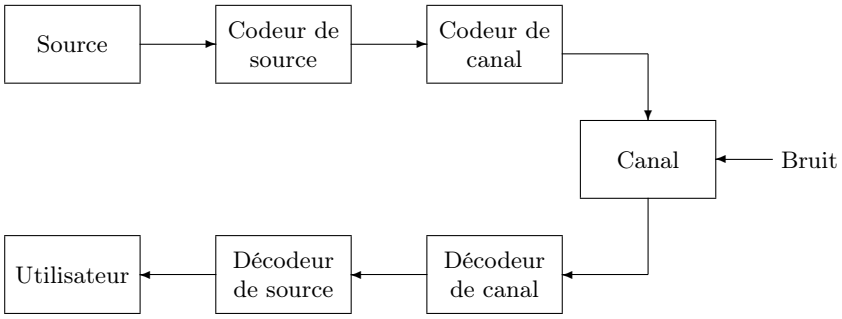


FIGURE 1.2 – Système de communication avec codeurs de source et de canal séparés.

Cette séparation rendra notre étude commode, car nous pourrons traiter indépendemment la source et le canal. De plus cette séparation n'implique aucune limitation sur les performances du système. Nous poursuivrons donc ce chapitre d'introduction par une brève description des différentes classes de modèles de sources et de canaux.

1.2 Sources et codage de source

1.2.1 Sources discrètes sans mémoire

Parmi les classes possibles de modèles de sources, nous nous intéresserons plus particulièrement aux *sources discrètes sans mémoire*. La sortie d'une telle source est une séquence de lettres tirées dans un alphabet fini [2] $\{a_1, \ldots, a_K\}$. Chaque lettre de la séquence est statistiquement indépendante des autre et obtenue d'après une loi de probabilité fixe $P(a_1), \ldots, P(a_K)$. Pour toute lettre a_k, $1 \leq k \leq K$, de la source, $P(a_K)$ est la probabilité pour que cette lettre soit produite, nous aurons donc $\sum_{k=1}^{K} P(a_k) = 1$.

Il peut sembler étonnant de modéliser une source d'information à l'aide d'un processus aléatoire. L'exemple suivant nous montre toutefois que l'utilisation des probabilités est indispensable pour coder une information de façon compacte.

Exemple: Soit une source d'information qui fournit comme information l'une des quatre lettres a_1, a_2, a_3 et a_4. Supposons que le codage transforme cette information en symboles binaires. Dans la Table 1.1, nous donnons deux codages différents de cette source. Dans la première

Méthode 1	Méthode 2
$a_1 \to 00$	$a_1 \to 0$
$a_2 \to 01$	$a_2 \to 10$
$a_3 \to 10$	$a_3 \to 110$
$a_4 \to 11$	$a_4 \to 111$

TABLE 1.1 – Deux codages d'un alphabet de quatre lettres.

méthode, deux symboles binaires sont générés pour chaque lettre émise, alors que dans la seconde le nombre de symboles est variable.

Si les quatre lettres sont équiprobables, alors la première méthode est la meilleure : 2 symboles par lettre en moyenne au lieu de 2,25. En revanche si l'on a

$$P(a_1) = \frac{1}{2}, \quad P(a_2) = \frac{1}{4}, \quad P(a_3) = P(a_4) = \frac{1}{8},$$

2. Nous réserverons la dénomination de « sources discrètes » aux alphabets finis. Lorsque l'alphabet est infini et dénombrable nous parlerons de source discrète infinie.

alors la méthode 1 nécessite toujours 2 symboles binaires par lettre en moyenne alors que la méthode 2 qui n'en nécessite que 1,75. Elle est dans ce cas la plus économique.

Il est donc important pour coder correctement une source de connaître son comportement statistique.

1.2.2 Entropie d'une source discrète sans mémoire

Il apparaît qu'il existe un lien entre l'information fournie par une source et la distribution de probabilité de la sortie de cette source. Plus l'événement donné par la source est probable, moins la quantité d'information correspondante est grande. Plus précisément, si une lettre a_k a une probabilité $P(a_k)$ d'être produite par la source, son *information propre* sera $I(a_k) = -\log_2 P(a_k)$. Cette définition parait conforme à l'idée intuitive que l'on peut se faire de l'information, et en particulier on a $I(a_k) = 0$ si $P(a_k) = 1$, c'est-à-dire que l'occurrence d'un événement certain ne peut fournir aucune information.

La valeur moyenne de l'information propre calculée sur l'ensemble de l'alphabet revêt une grande importance. Elle est appelée *entropie* de la source et vaut

$$\sum_{k=1}^{K} -P(a_k) \log_2 P(a_k).$$

L'entropie d'une source est le nombre moyen minimal de symboles binaires par lettre nécessaires pour représenter la source.

Par exemple, si un alphabet contient 2^L lettres équiprobables, il est immédiat que l'entropie de la source correspondante vaut L. Or il est bien clair que pour représenter 2^L lettres distinctes, L symboles binaires sont nécessaires.

L'entropie d'une source est parfois donnée en bits par seconde, si l'entropie d'une source discrète est H, et si les lettres sont émises toutes les τ_s secondes, son entropie en bits/s sera H/τ_s.

1.2.3 Autres modèles de sources

On peut citer également parmi les classes de modèles de source, les *sources discrètes stationnaires*, pour lesquelles une entropie peut être définie de façon analogue.

Enfin, les sources non discrètes, ou *sources continues*, ont une grande importance pour les applications. La sortie d'une telle source sera une fonction continue du temps, par exemple une tension, qu'il faut coder en une séquence binaire. La fonction continue devra être décrite le plus fidèlement possible par la séquence binaire générée par le codeur de source. Le problème essentiel dans ce cas consiste à minimiser le nombre de symboles transmis pour un niveau de distorsion donné.

1.3 Canaux et codage de canal

1.3.1 Canaux discrets

Pour modéliser correctement un canal de transmission il est nécessaire de spécifier l'ensemble des entrées et l'ensemble des sorties possibles. Le cas le plus simple est celui du *canal discret sans mémoire* : l'entrée est une lettre d'un alphabet fini $\{a_1, \ldots, a_K\}$, et la sortie est une lettre d'un autre alphabet fini, éventuellement identique, $\{b_1, \ldots, b_J\}$. Ces lettres sont émises en séquence, et pour que le canal soit sans mémoire, il faut que chaque lettre de la séquence reçue ne dépende statistiquement que de la lettre émise de même position. Ainsi un canal discret sans mémoire est entièrement décrit par la donnée des probabilités conditionnelles $P(b_j \mid a_k)$, pour tout a_k dans l'alphabet d'entrée et tout b_j dans l'alphabet de sortie.

Par exemple le canal binaire symétrique, représenté dans la Figure 1.3, est un canal discret sans mémoire, dont l'alphabet d'entrée et l'alphabet de sortie sont égaux tous deux à $\{0, 1\}$. La

probabilité pour qu'un symbole soit inchangé est $1-\epsilon$, et la probabilité pour qu'il soit transformé en son opposé est ϵ.

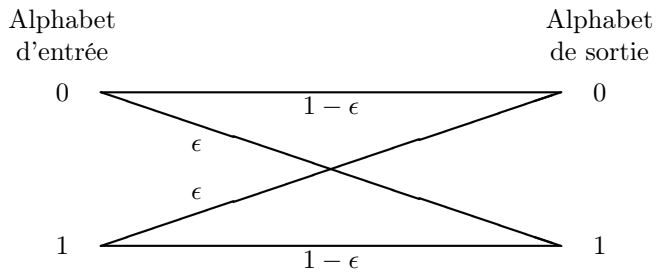


FIGURE 1.3 – Canal binaire symétrique.

On peut également considérer des canaux discrets à mémoire dans lesquels chaque lettre de la séquence de sortie peut dépendre de plusieurs lettres de la séquence d'entrée.

1.3.2 Canaux continus

Il existe une classe de modèles de canaux, appelé *canaux continus*, beaucoup plus proche des canaux physiques, dans lesquels l'entrée et la sortie sont des fonctions continues du temps.

Pour les canaux de cette classe, il est possible, et commode, de séparer le codeur et le décodeur en deux parties, comme le montre la figure 1.4. La première partie du codeur, que nous appellerons codeur de canal discret, transforme une séquence binaire en une séquence de lettres d'un alphabet fini $\{a_1, \ldots, a_K\}$, la seconde partie du codeur, le modulateur de données digitales (MDD) envoie pendant un temps τ_c sur le canal une des fonctions du temps prédéfinies $s_1(t), \ldots, s_K(t)$. La durée τ_c sera l'intervalle de temps séparant l'émission de deux lettre par le codeur de canal discret. L'ensemble de ces fonctions du temps mises bout à bout sera convertie à la sortie du canal par le démodulateur de données digitales (DDD) en une séquence de lettres d'un alphabet de sortie $\{b_1, \ldots, b_J\}$ au rythme, là encore, d'une lettre toutes les τ_c secondes.

On voit que de cette manière l'ensemble MDD-canal-DDD peut être considéré comme un canal discret dont l'alphabet d'entrée est $\{a_1, \ldots, a_K\}$ et l'alphabet de sortie $\{b_1, \ldots, b_J\}$. Si de plus le bruit est indépendant entre chaque intervalle de τ_c secondes, alors ce canal sera sans mémoire. L'étude des canaux discrets nous permettra donc de déduire des résultats sur les canaux continus.

1.3.3 Capacité d'un canal

L'un des paramètres les plus important d'un canal est sa *capacité*, nous verrons que cette capacité peut s'interpréter comme une mesure de la quantité d'information, exprimée en bits par seconde par exemple, pouvant être transmise à travers ce canal. L'intérêt de cette grandeur nous vient essentiellement du théorème sur le codage des canaux bruités : grossièrement, ce théorème dit que dans un canal de capacité C bits/s, si l'on veut faire transiter une quantité d'information à un *taux utile* de R bits/s, tel que $R < C$, alors il existe une procédure de codage et une procédure de décodage telles que le *taux d'erreur résiduel* soit arbitrairement faible.

La réciproque de ce théorème nous dit par contre que si $R > C$ alors pour toute procédure de codage et de décodage, le taux d'erreur résiduel est supérieur à une constante strictement positive qui dépend de R et C.

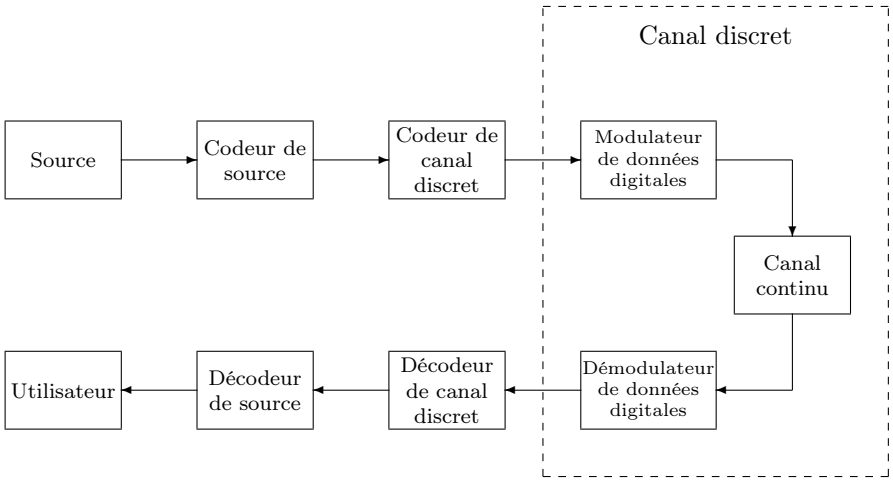


FIGURE 1.4 – Canal continu et canal discret.

Chapitre 2

Une mesure de l'information

Nous donnons dans ce chapitre une mesure de la quantité d'information adaptée à la description statistique des sources et des canaux. Les énoncés qui en résultent faisant largement appel aux probabilités discrètes, nous commencerons l'exposé par le rappel de quelques éléments de théorie des probabilités.

2.1 Espace probabilisé discret

Un *espace probabilisé discret* X est une expérience aléatoire décrite par la donnée d'un *espace des épreuves* $\mathcal{X}$ fini ou dénombrable (l'ensemble des résultats possibles de l'expérience), muni d'une mesure de probabilité $\Pr_X$. Un *évènement aléatoire* est une partie de $\mathcal{X}$

La mesure de probabilité est une application à valeur dans l'intervalle réel $[0, 1]$ définie pour toute partie $\mathcal{U}$ de $\mathcal{X}$ et qui vérifie

- $\Pr_X(\mathcal{X}) = 1$
- Si $\mathcal{U} \subset \mathcal{X}$ et $\mathcal{V} \subset \mathcal{X}$ avec $\mathcal{U} \cap \mathcal{V} = \emptyset$ alors $\Pr_X(\mathcal{U} \cup \mathcal{V}) = \Pr_X(\mathcal{U}) + \Pr_X(\mathcal{V})$

Un élément x de $\mathcal{X}$ est appelé *issue*, le singleton $\{x\}$ est un *évènement élémentaire* et nous posons $p_X(x) = \Pr_X[\{x\}]$. Pour toute partie $\mathcal{U}$ de $\mathcal{X}$, sa probabilité dans X sera égale à

$$\Pr_X[\mathcal{U}] = \sum_{x \in \mathcal{U}} p_X(x)$$

Dans le cas discret, la donnée d'une distribution de probabilité $p_X : \mathcal{X} \to \mathbf{R}$ telle que

$$p_X(x) \geq 0, \forall x \in \mathcal{X} \quad \text{et} \quad \sum_{x \in \mathcal{X}} p_X(x) = 1$$

est nécessaire et suffisante pour décrire la mesure de probabilité $\Pr_X$, nous parlerons de l'espace probabilisé discret $X = (\mathcal{X}, p_X)$.

2.1.1 Définitions

Soit $X = (\mathcal{X}, p_X)$ un espace probabilisé discret.

Une *variable aléatoire* d'un espace probabilisé est une application définie sur l'espace des épreuves. Pour toute variable aléatoire V, nous noterons

$$\Pr_X[V = v] = \Pr_X[\{x \in \mathcal{X} \mid V(x) = v\}],$$

C'est-à-dire que l'évènement « $V = v$ » est identifié avec l'ensemble des issues telle que $V(x) = v$. L'*espérance* (ou *moyenne*) d'une variable aléatoire réelle $V : \mathcal{X} \to \mathbf{R}$ est définie et notée par

$$E_X[V] = \sum_{x \in \mathcal{X}} p_X(x) V(x).$$

Probabilité conditionnelle. Dans certains cas, nous voudrons restreindre un espace probabilisé $(\mathcal{X}, p_X)$ à une partie $\mathcal{V}$ de $\mathcal{X}$ (telle que $\Pr_X[\mathcal{V}] \neq 0$). Dans le nouvel espace probabilisé $V = (\mathcal{V}, p_V)$, nous voulons que, pour $x \in \mathcal{V}$, la valeur de $p_V(x)$ soit proportionnelle à $p_X(x)$, pour conserver une distribution de probabilité, nous devons avoir $p_V(x) = p_X(x)/\Pr_X[\mathcal{V}]$.

Pour toute partie $\mathcal{U}$ de $\mathcal{X}$, nous définissons la *probabilité conditionnelle* de $\mathcal{U}$ sachant $\mathcal{V}$ dans X comme la probabilité de $\mathcal{U} \cap \mathcal{V}$ dans V. Nous noterons

$$\Pr_X[\mathcal{U} \mid \mathcal{V}] = \Pr_V[\mathcal{U} \cap \mathcal{V}] = \frac{\Pr_X[\mathcal{U} \cap \mathcal{V}]}{\Pr_X[\mathcal{V}]}.$$

La notation $\Pr_X[\mathcal{U} \mid \mathcal{V}]$ est commode mais abusive, la grandeur qu'elle définit ne pouvant pas être interprétée comme une probabilité dans X.

Soient deux parties $\mathcal{U}$ et $\mathcal{V}$ de $\mathcal{X}$, nous noterons $\Pr_X[\mathcal{U}, \mathcal{V}] = \Pr_X[\mathcal{U} \cap \mathcal{V}]$. Nous dirons que $\mathcal{U}$ et $\mathcal{V}$ sont *statistiquement indépendants*, si l'une des trois propriétés équivalentes ci-dessous est vérifiée

(i) $\Pr_X[\mathcal{U} \mid \mathcal{V}] = \Pr_X[\mathcal{U}]$

(ii) $\Pr_X[\mathcal{V} \mid \mathcal{U}] = \Pr_X[\mathcal{V}]$

(iii) $\Pr_X[\mathcal{U}, \mathcal{V}] = \Pr_X[\mathcal{U}]\Pr_X[\mathcal{V}]$

De même, deux variables aléatoires U et V sont dites indépendantes si pour tout u et tout v, nous avons $\Pr_X[U = u, V = v] = \Pr_X[U = u]\Pr_X[U = u]$.

2.1.2 Espace probabilisé joint

Nous parlerons d'espace probabilisé joint XY lorsque l'espace des épreuves est un produit cartésien $\mathcal{X} \times \mathcal{Y}$. La distribution de probabilité jointe est notée p_{XY}.

Nous noterons X et Y les variables aléatoires correspondant respectivement à la première et à la seconde coordonnées de l'issue. Nous définissons les lois marginales p_X et p_Y de p_{XY} par

$$p_X(x) = \Pr_{XY}[X = x] = \sum_{y \in \mathcal{Y}} p_{XY}(x, y), \quad \text{et} \quad p_Y(y) = \Pr_{XY}[Y = y] = \sum_{x \in \mathcal{X}} p_{XY}(x, y).$$

Notons que $(\mathcal{X}, p_X)$ et $(\mathcal{Y}, p_Y)$ définissent des espaces probabilisés, par commodité, nous noterons ces espaces X et Y, du même nom que les variables aléatoires qui leurs sont associées.

Lorsque $p_X(x) > 0$, la probabilité conditionnelle de « $Y = y$ » sachant « $X = x$ » est égale à

$$\Pr_{XY}[Y = y \mid X = x] = \frac{p_{XY}(x, y)}{p_X(x)}.$$

En effet les évènements « $Y = y$ » et « $X = x$ » correspondent aux parties $\mathcal{X} \times \{y\}$ et $\{x\} \times \mathcal{Y}$ de $\mathcal{X} \times \mathcal{Y}$ dont l'intersection est l'évènement élémentaire $\{(x, y)\}$. De même la probabilité conditionnelle de « $X = x$ » sachant « $Y = y$ » est égale à

$$\Pr_{XY}[X = x \mid Y = y] = \frac{p_{XY}(x, y)}{p_Y(y)}.$$

Les événements « $X = x$ » et « $Y = y$ » sont dit *statistiquement indépendants* si

$$p_{XY}(x, y) = p_X(x)p_Y(y).$$

Si cette égalité est vraie pour tout couple de $\mathcal{X} \times \mathcal{Y}$, alors X et Y sont dit statistiquement indépendants.

2.1.3 Simplification des notations

Dans les notations que nous utilisons, l'espace probabilisé est utilisé en indice. Dans toutes les utilisations que nous ferons des probabilités et des distributions de probabilité, cet indice peut-être omis sans créer d'ambiguïté.

Par exemple, considérons les espaces probabilisés X et Y sur $\mathcal{X}$ et $\mathcal{Y}$ ainsi que l'espace joint XY sur $\mathcal{X} \times \mathcal{Y}$. La probabilité $\Pr[\mathcal{U}]$ sera définie sur X, Y ou XY selon que $\mathcal{U}$ est une partie de $\mathcal{X}$, $\mathcal{Y}$ ou $\mathcal{X} \times \mathcal{Y}$. Nous utiliserons Pr au lieu de $\Pr_X$, $\Pr_Y$ et $\Pr_{XY}$, y compris pour les probabilités conditionnelles.

De la même manière, pour les distributions, p_X, p_Y ou p_{XY} il n'y a en général pas d'ambiguïté. Pour tout $(x, y) \in \mathcal{X} \times \mathcal{Y}$ nous noterons

$$\begin{array}{l} p(x) = p_X(x) \\ p(y) = p_Y(y) \\ p(x, y) = p_{XY}(x, y) \\ p(x \mid y) = \Pr_{XY}[X = x \mid Y = y] \\ p(y \mid x) = \Pr_{XY}[Y = y \mid X = x] \end{array}$$

2.2 Processus stochastiques

Un processus stochastique sur un ensemble discret $\mathcal{X}$ consiste à tirer aléatoirement et indéfiniment des éléments de $\mathcal{X}$. L'espace des épreuves correspondant est l'ensemble des suites $(x_1, \ldots, x_n, \ldots)$ d'éléments de $\mathcal{X}$. Pour tout entier $n \geq 1$, nous noterons X_n la variable aléatoire dont la valeur est le n-ème élément de la suite issue de l'expérience aléatoire. L'espace probabilisé que nous considérons n'est pas discret et la mesure de probabilité qui lui est associée ne peut être définie par une distribution de probabilité. Elle est définie par la donnée, pour tout $n > 0$ et tout $(x_1, \ldots, x_n) \in \mathcal{X}^n$, de

$$p(x_1, x_2, \ldots, x_n) = \Pr[X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n]$$

Processus sans mémoire. Si les variables aléatoires X_n sont deux à deux indépendantes et identiquement distribuées, nous dirons que le processus est *sans mémoire*, pour tout $n > 0$ et tout $(x_1, \ldots, x_n) \in \mathcal{X}^n$, nous avons

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i)$$

Processus stationnaire. Un processus stochastique sera dit *stationnaire* si pour tout $n > 0$

$$\Pr[X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n] = \Pr[X_2 = x_1, X_3 = x_2, \ldots, X_{n+1} = x_n].$$

Processus markovien. Un processus stochastique sera dit *markovien* (d'ordre 1) si pour tout $n > 0$,

$$\Pr[X_{n+1} = x_{n+1} \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n] = \Pr[X_{n+1} = x_{n+1} \mid X_n = x_n]$$

Un processus markovien est *invariant dans le temps* si $\Pr[X_{n+1} = x_{n+1} \mid X_n = x_n]$ est indépendant de n, nous noterons $p(x_{n+1} \mid x_n)$. Nous pouvons alors écrire

$$p(x_1, x_2, \ldots, x_n) = p(x_1)p(x_2 \mid x_1) \ldots p(x_n \mid x_{n-1})$$

Dans ce cas le processus est entièrement décrit par sa *distribution initiale* $(p(x))_{x \in \mathcal{X}}$ et sa *matrice de transition* $\Pi = (p(x_2 \mid x_1))_{(x_1, x_2) \in \mathcal{X}^2}$. Ce processus sera dit *irreductible* si sa matrice de transition admet un unique point fixe $(\lambda(x))_{x \in \mathcal{X}})$ appelé *distribution stationnaire*.

Si le processus considéré est stationnaire, alors il est invariant dans le temps et sa distribution initiale est égale à la distribution stationnaire.

Un processus *markovien* d'ordre ℓ est défini par

$$\Pr[X_{n+1} = x_{n+1} \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n] \\ = \Pr[X_{n+1} = x_{n+1} \mid X_{n-\ell+1} = x_{n-\ell+1}, \ldots, X_n = x_n]$$

$$p(x_1, x_2, \ldots, x_n) = p(x_1, \ldots, x_\ell) p(x_{\ell+1} \mid x_1, \ldots, x_\ell) \ldots p(x_n \mid x_{n-1}, \ldots, x_{n-\ell})$$

En pratique, en considérant le processus $(Y_i)_{i>0}$ défini sur $\mathcal{Y} = \mathcal{X}^\ell$ avec $Y_i = (X_i, X_{i+1}, \ldots, X_{i+\ell})$, on peut toujours se ramener à l'ordre 1.

2.3 Une définition de l'information

2.3.1 Incertitude et information

a - Description qualitative de l'information

Avant de trouver un moyen de mesurer quantitativement l'information, nous allons essayer de préciser le concept d'information.

Nous l'avons vu, la façon la plus appropriée de décrire un système de communication est d'en donner un modèle probabiliste. Qualitativement, fournir une information consiste à lever une partie de l'incertitude sur l'issue d'une expérience aléatoire. Par exemple, l'observation d'un '0' à la sortie d'un canal binaire bruité augmente la probabilité qu'un '0' ait été émis et diminue la probabilité qu'un '1' ait été émis.

D'une façon générale, considérons un espace probabilisé joint XY et les deux événements $X = x$ et $Y = y$. La probabilité conditionnelle $p(x \mid y)$ peut être interprétée comme la modification apportée à la probabilité $p(x)$ de l'événement x lorsque l'on reçoit l'information que l'événement y est réalisé. L'information « y est réalisé » modifie la probabilité de x, c'est-à-dire l'incertitude sur la réalisation de x, de $p(x)$ à $p(x \mid y)$. Plus précisément,

- si $p(x \mid y) \leq p(x)$, l'incertitude sur x augmente,
- si $p(x \mid y) \geq p(x)$, l'incertitude sur x diminue.

Une diminution de l'incertitude sur x doit être comprise comme un gain d'information sur x et inversement, une augmentation de l'incertitude sur x doit être comprise comme une perte d'information sur x. Cette quantité sear appelée *information mutuelle* de x et y, et peut être positive ou négative. Nous verrons par la suite que sa moyenne est toujours positive.

Le gain d'information maximal sur x sera obtenu lorsque $p(x \mid y) = 1$, c'est-à-dire essentiellement lorsque $x = y$. Cette quantité, fonction de la probabilité, sera appelée *information propre* de x notée $I(x) = f(p(x))$. Pour quantifier l'information, il nous faudra spécifier et définir cette fonction $f()$.

b - Description quantitative de l'information

L'information propre de x doit être une fonction de sa probabilité : $I(x) = f(p(x))$. Pour définir $f()$ nous admettrons :

- L'information propre de x est une fonction décroissante de $p(x)$: en effet un évènement certain n'apporte aucune information, alors qu'un évènement improbable en apportera beaucoup.
- L'information propre est une grandeur additive : si les évènements x et y sont statistiquement indépendants alors l'information totale qu'ils peuvent fournir est la somme des informations propres $f(p(x, y)) = f(p(x)p(y)) = f(p(x)) + f(p(y))$.

Nous devrons donc choisir une fonction de la forme $I(x) = \lambda \log p(x)$, avec $\lambda < 0$ pour assurer la décroissance par rapport à $p(x)$. Le choix de λ va dépendre de l'unité d'information que nous choisirons. Dans ce cours nous utiliserons le bit :

Un bit est égal à la quantité d'information fournie par le choix d'une alternative parmi deux équiprobables.

En clair, cela signifie que si X est un espace probabilisé dans l'espace des épreuves $\{0,1\}$ muni d'une loi uniforme (i.e. $p(0) = p(1) = 1/2$), alors la quantité d'information fournie par la réalisation de l'événement $X = 0$ (ou $X = 1$) est de 1 bit. On a

$$I(0) = \lambda \log p(0) = -\lambda \log 2 = 1,$$

donc $\lambda = -1/\log 2$ ce qui revient à choisir le logarithme en base 2 pour la définition de $I(x)$:

$$I(x) = -\log_2 p(x).$$

Le terme « bit » est une abréviation de « binary digit » mais ne doit pas être compris comme « symbole binaire » mais comme une unité de mesure de l'information. Il se trouve que pour représenter n bits d'information, il faudra utiliser (au moins) n symboles binaires en moyenne.

Exemple: Soit une source dont l'alphabet de sortie $\{a_0, \ldots, a_{15}\}$ comprend 16 lettres équiprobables, c'est-à-dire que pour tout k, $0 \leq k < 16$, $P(a_k) = 1/16$. L'information propre de l'une de ces sorties a_k sera égale à $I(a_k) = -\log_2(1/16) = 4$ bits.
Dans ce cas particulier, l'information va consister à choisir un entier k dans {0,1,...,15}, et pour représenter cette information il faut disposer de 4 symboles binaires.

Il faut prendre garde que ce résultat n'est vrai que parce que les lettres de la source sont équiprobables, en effet, si ce n'est pas le cas, l'information propre de l'événement a_k, $I(a_k) = -\log_2 p(a_k)$, sera généralement différente de 4, et nous verrons même plus loin que l'information propre moyenne peut être inférieure strictement à 4 bits.

2.3.2 Information mutuelle – Information propre

Nous considérons un espace probabilisé joint XY. Nous désirons ici donner une mesure quantitative de ce que nous apporte la réalisation d'un événement $y \in \mathcal{Y}$ sur la possibilité de réalisation d'un autre événement $x \in \mathcal{X}$; l'occurrence de l'événement y transforme la probabilité *a priori* $p(x)$ de l'événement x en la probabilité *a posteriori* $p(x \mid y)$.

Définition 2.1 (Information mutuelle) *L'information mutuelle entre les événements $x \in \mathcal{X}$ et $y \in \mathcal{Y}$ est définie par*[1]

$$I(x;y) = \log_2 \frac{p(x \mid y)}{p(x)}.$$

Remarquons que cette définition est symétrique, en effet, on a par définition de la probabilité conditionnelle, $p(x,y) = p(x \mid y)p(y) = p(y \mid x)p(x)$, et donc

$$I(x;y) = I(y;x) = \log_2 \frac{p(x,y)}{p(x)p(y)}.$$

Discussion sur le signe de $I(x;y)$

- $I(x;y) > 0$ signifie que si l'un des deux événements se réalise, alors la probabilité d'occurrence de l'autre augmente.
- $I(x;y) < 0$ signifie que si l'un des deux événements se réalise, alors la probabilité d'occurrence de l'autre diminue.
- $I(x;y) = 0$ signifie que les deux événements sont statistiquement indépendants.

Exemple: Considérons le canal binaire symétrique de probabilité de transition ϵ, avec des entrées a_1 et a_2 équiprobables (cf. Figure 2.1). Afin d'éviter une possible confusion, nous utiliserons les lettres a_1 et a_2 pour les entrées, et les lettres b_1 et b_2 pour les sorties au lieu des symboles binaires 0 et 1. Le canal binaire symétrique est défini par les probabilités conditionnelles

1. La notation $I(x,y)$ serait ambiguë car elle pourrait désigner l'information propre de l'événement (x,y).

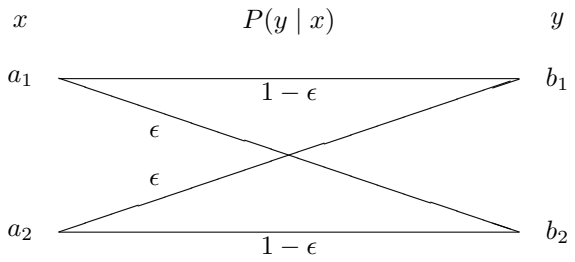


FIGURE 2.1 – Canal binaire symétrique

$p(b_1 \mid a_1) = p(b_2 \mid a_2) = 1 - \epsilon$, $p(b_1 \mid a_2) = p(b_2 \mid a_1) = \epsilon$. Puisque $p(a_1) = p(a_2) = 1/2$, on en déduit

$$p(a_1, b_1) = p(a_2, b_2) = \frac{1-\epsilon}{2} \quad \text{et} \quad p(a_1, b_2) = p(a_2, b_1) = \frac{\epsilon}{2},$$

et $p(b_1) = p(b_2) = 1/2$. Cela nous permet de calculer l'information mutuelle de chaque couple (a_k, b_j)

$$I(a_1; b_1) = I(a_2; b_2) = \log_2 2(1-\epsilon) \quad \text{et} \quad I(a_1; b_2) = I(a_2; b_1) = \log_2 2\epsilon.$$

On constate que pour $\epsilon < 1/2$, $I(a_1; b_1)$ est positif et $I(a_1; b_2)$ est négatif. Cela signifie que lorsqu'on observe à la sortie du canal la lettre b_1, la probabilité pour que a_1 ait été émise augmente. Et que au contraire si b_2 est observée, la probabilité pour que la lettre émise ait été a_1 diminue.

Enfin lorsque $\epsilon = 1/2$, toutes les informations mutuelles sont nulles, et donc les alphabets d'entrée et de sortie sont statistiquement indépendants, ce qui n'est évidemment pas souhaitable.

Considérons à présent à une autre grandeur intéressante ; la valeur maximale de $I(x; y)$ pour x fixé et y quelconque. Cette valeur est atteinte lorsque $p(x \mid y) = 1$ (en fait tout se passe comme si $y = x$). On a alors

$$I(x; y) = \log_2 \frac{p(x \mid y)}{p(x)} = \log_2 \frac{1}{p(x)}.$$

Il s'agit de la quantité maximale d'information que peut fournir l'événement x.

Définition 2.2 (Information propre) *L'information propre de l'événement $x \in \mathcal{X}$ est définie par*

$$I(x) = -\log_2 p(x).$$

L'information propre s'interprète comme la « quantité d'information fournie par la réalisation d'un événement ».

Notons que l'information propre est toujours positive ou nulle, et que plus un événement est improbable, plus son information propre est grande. À l'inverse, lorsque $p(x) = 1$, on a $I(x) = 0$, c'est-à-dire que la réalisation d'un événement certain n'apporte aucune information, ce qui semble conforme à l'intuition.

On peut également définir dans l'espace probabilisé joint XY l'information propre conditionnelle qui est égale à la quantité d'information fournie par un événement x sachant que l'événement y s'est réalisé.

Définition 2.3 *L'information propre conditionnelle de l'événement $x \in \mathcal{X}$, sachant $y \in \mathcal{Y}$ est définie par*

$$I(x \mid y) = -\log_2 p(x \mid y).$$

Cette dernière définition nous permet de donner une nouvelle interprétation de l'information mutuelle entre deux événements. En effet d'après la relation

$$I(x;y) = I(x) - I(x \mid y), \tag{2.1}$$

l'information mutuelle entre x et y est égale à la quantité d'information fournie par x moins la quantité d'information que fournirait x si y se réalisait.

2.4 Information mutuelle moyenne – Entropie

2.4.1 Définitions

Dans l'espace probabilisé joint XY, l'information mutuelle peut être considérée comme une variable aléatoire réelle. Sa moyenne peut donc être définie.

Définition 2.4 (Information mutuelle moyenne) *L'information mutuelle moyenne de X et Y dans l'espace probabilisé joint XY est définie par*

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

On peut également définir la moyenne de l'information propre de l'espace probabilisé X, cette moyenne porte le nom d'entropie.

Définition 2.5 (Entropie) *L'entropie d'un espace probabilisé X est défini par*

$$H(X) = \sum_{x\in\mathcal{X}} p(x) I(x) = \sum_{x\in\mathcal{X}} -p(x) \log_2 p(x).$$

Enfin, l'information propre conditionnelle est également une v.a. réelle, sa moyenne porte le nom d'entropie conditionnelle.

Définition 2.6 *L'entropie conditionnelle de X sachant Y dans l'espace probabilisé joint XY est définie par*

$$H(X \mid Y) = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} -p(x,y) \log_2 p(x \mid y).$$

L'équation (2.1) peut se réécrire en moyenne

$$I(X;Y) = H(X) - H(X \mid Y).$$

Cette relation nous sera utile pour interpréter la capacité d'un canal.

2.4.2 Propriétés de l'entropie

Théorème 2.7 *Soit X un espace probabilisé discret de cardinal K. Alors son entropie $H(X)$ vérifie*

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

avec égalité si et seulement si la loi de probabilité de X est uniforme.

preuve: La fonction $\log_2$ vérifie

$$\begin{array}{ll} \log_2 z < (z-1)\log_2 e & \text{si } z > 0, z \neq 1 \\ \log_2 z = (z-1)\log_2 e & \text{si } z = 1 \end{array} \tag{2.2}$$

Nous allons montrer que $H(X) - \log_2 K \leq 0$. On a

$$\begin{aligned} H(X) - \log_2 K &= \sum_x p(x) \log_2 \frac{1}{p(x)} - \sum_x p(x) \log_2 K \\ &= \sum_x p(x) \log_2 \frac{1}{Kp(x)}. \end{aligned}$$

En appliquant (2.2) à $z = 1/(Kp(x))$, on obtient

$$\begin{aligned} H(X) - \log_2 K &\leq \log_2 e \sum_x p(x) \left(\frac{1}{Kp(x)} - 1 \right) \\ &\leq \log_2 e \left(\sum_x \frac{1}{K} - \sum_x p(x) \right) = 0, \end{aligned}$$

avec égalité si et seulement si $1/(Kp(x)) = 1$ pour tout x.

L'inégalité est donc vérifiée, et il y a égalité si et seulement si $p(x) = 1/K$ pour tout x, c'est à dire si la loi de probabilité de X est uniforme. ◇

Cette proposition est fondamentale, en effet c'est elle qui nous dit que l'entropie d'un espace probabilisé de taille 2^L muni d'une loi uniforme est égale à L bits.

Nous verrons dans le chapitre sur le codage de source que cela nous donne une borne inférieure sur le nombre de symboles binaires nécessaires pour coder une source discrète à l'aide d'un code de taille fixe. Cette borne sera égale au plus petit entier supérieur ou égal à $\log_2 K$.

Théorème 2.8 *Soit un espace probabilisé joint discret XY. L'information mutuelle moyenne $I(X;Y)$ de X et de Y vérifie*

$$I(X;Y) \geq 0,$$

avec égalité si et seulement si X et Y sont statistiquement indépendants.

preuve: Montrons que $-I(X;Y) \leq 0$. On a, en utilisant (2.2)

$$\begin{aligned} -I(X;Y) &= \sum_{x,y} p(x,y) \log_2 \frac{p(x)p(y)}{p(x,y)} \\ &\leq \log_2 e \sum_{x,y} p(x,y) \left(\frac{p(x)p(y)}{p(x,y)} - 1 \right) \\ &\leq \log_2 e \left(\sum_{x,y} p(x)p(y) - \sum_{x,y} p(x,y) \right) \\ &\leq \log_2 e \left(\sum_x p(x) \sum_y p(y) - 1 \right) = 0, \end{aligned}$$

et il y a égalité si et seulement si $p(x)p(y)/p(x,y) = 1$ pour tout couple (x,y).

L'inégalité est donc prouvée et puisque x et y sont indépendants si et seulement si $p(x,y) = p(x)p(y)$, l'égalité $I(X;Y) = 0$ est vraie si et seulement si X et Y sont statistiquement indépendants. ◇

Ce résultat signifie essentiellement que en moyenne le fait de connaître la valeur de y dans Y diminue toujours l'incertitude sur X, sauf si X et Y sont indépendants, auquel cas aucune information n'est apportée.

Une autre version de cet résultat relie l'entropie et l'entropie conditionnelle.

Proposition 2.9 *Soit un espace probabilisé joint discret XY.*

$$H(X \mid Y) \leq H(X).$$

Interprété intuitivement ce résultat nous indique que « le conditionnement réduit l'entropie ».

Chapitre 3

Codage des sources discrètes

Nous nous intéresserons dans ce chapitre au codage de la sortie d'une source discrète en une séquence binaire. Ce codage devra permettre de retrouver la séquence de lettre de la source à partir du codage binaire. Nous verrons que le nombre minimal moyen de symboles binaires par lettre est égal à l'entropie de la source.

Dans de nombreux exemples pratique de codage de source, comme le code Morse, les lettres les plus fréquentes se voient attribuer les mots de code les plus courts. En Morse, la lettre 'e' est représentée par le mot '$\cdot$', alors que la lettre 'q', beaucoup moins fréquente, est codée par le mot '$\cdot\cdot--$'. De tels code sont dits de longueur variable. Notons que le code Morse n'est pas binaire, il existe un troisième symbole ' ' en plus de '$\cdot$' et '$-$' permettant de séparer les lettres (cet espace est même un peu plus long lorsqu'il sépare des mots).

3.1 Codage d'un alphabet discret

Nous considérerons un alphabet $\mathcal{X}$ discret (fini ou infini dénombrable). Nous désignerons par $\mathcal{X}^l$ l'ensemble des l-uplets de lettres de $\mathcal{X}$ et par $\mathcal{X}^* = \bigcup_{l\geq 1} \mathcal{X}^l$ l'ensemble des séquences finies non vides de lettres de $\mathcal{X}$. Nous désignerons par $\{0,1\}^*$ l'ensemble des séquences binaires finies. Pour tout $m \in \mathcal{X}^*$, nous noterons $|m|$ sa longueur.

Définition 3.1 *Un* codage *d'un alphabet discret est une procédure qui associe à chaque séquence finie de lettres une séquence binaire finie.*

Un codage est donc une application de $\mathcal{X}^*$ dans $\{0,1\}^*$.

Définition 3.2 *Un* code *d'un alphabet discret est une procédure qui associe à chaque lettre une séquence binaire appelée* mot de code*.*

Un code est donc une application φ de $\mathcal{X}$ dans $\{0,1\}^*$, qui à toute lettre x de $\mathcal{X}$ associe un mot de code. À un code donné, on peut associer le codage

$$\begin{array}{ccc} \mathcal{X}^* & \to & \{0,1\}^* \\ (x_1,\ldots,x_\ell) & \mapsto & \varphi(x_1) \,\|\, \cdots \,\|\, \varphi(x_\ell) \end{array} \tag{3.1}$$

où le symbole $\|$ représente la concaténation. En revanche, un codage n'est pas toujours associé à un code.

Définition 3.3 (Code régulier) *Un code sera dit* régulier *s'il est injectif (deux lettres distinctes sont codées à l'aide de deux mots de code distincts).*

Dans toute la suite nous ne considérerons que des codes réguliers. De plus pour un code régulier le terme de code sera également utilisé pour désigner l'ensemble des mots de code.

3.1.1 Codage d'une source discrète sans mémoire

Dans le cas général, une source discrète sera un processus stochastique sur un alphabet $\mathcal{X}$. Lorsque ce processus est sans mémoire une source discrète $X = (\mathcal{X}, p_X)$ sera un espace probabilisé discret, c'est-à-dire un espace des épreuves discret $\mathcal{X}$, nous parlerons ici d'*alphabet*, muni d'une distribution de probabilité p_X. Nous parlerons de code ou de codage d'une source pour désigner un code ou un codage de son alphabet.

Définition 3.4 *Soit $X = (\mathcal{X}, p_X)$ une source discrète sans mémoire d'entropie H et soit φ un code de X. La* longueur moyenne *de φ, notée $|\varphi|$, est le nombre de symboles binaires utilisés en moyenne pour coder une lettre de la source :*

$$|\varphi| = \sum_{x \in \mathcal{X}} |\varphi(x)| p_X(x),$$

*L'*efficacité *de φ est définie par*

$$E(\varphi) = \frac{H}{|\varphi|}.$$

Nous verrons que l'efficacité d'un code régulier ne peut excéder 1.

Efficacité d'un codage. Pour une source X sans mémoire et pour tout entier $L > 0$, nous noterons X^L la source d'alphabet $\mathcal{X}^L$ munie de la loi produit. Nous noterons également p_X cette loi. Soit Φ un codage de $\mathcal{X}$. Si la limite suivante

$$\mathcal{L}(\Phi) = \lim_{L \to \infty} \frac{1}{L} \sum_{(x_1, \ldots, x_L) \in \mathcal{X}^L} p_X(x_1, \ldots, x_L) \left| \Phi(x_1, \ldots, x_L) \right|,$$

existe, elle sera égale au nombre moyen de symboles binaires par lettre pour coder une suite de lettre de $\mathcal{X}$. Cette grandeur, qui mesure la performance du codage Ψ sera appelée *longueur moyenne par lettre*. L'efficacité du codage sera définie comme pour un code par

$$E(\Phi) = \frac{H(X)}{\mathcal{L}(\Phi)}.$$

Définie ainsi, l'efficacité d'un codage associé à un code par (3.1) est égale à l'efficacité de ce code.

3.1.2 Codes de longueur fixe

Définition 3.5 (Code de longueur fixe) *Un code de longueur fixe est un code dont tous les mots de code ont la même longueur. On parlera de longueur du code.*

Proposition 3.6 *Soit un alphabet $\mathcal{X}$ de cardinal K. Il existe un code régulier de $\mathcal{X}$ de longueur n telle que*

$$\log_2 K \leq n < 1 + \log_2 K. \tag{3.2}$$

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

preuve: Soit $\mathcal{X} = \{x_1, \ldots, x_K\}$, si nous choisissons de coder x_k par l'écriture de $k-1$ en base 2, cela nécessitera n symboles binaires, où n sera le plus petit entier tel que $2^n \geq K$. On en déduit facilement (3.2) en prenant le logarithme.

Réciproquement, soit $n < \log_2 K$, l'ensemble des séquences binaires de longueur n a pour cardinal $2^n < K$, il est donc exclus de faire correspondre à chaque élément de $\mathcal{X}$ une séquence différente. ◇

L'efficacité d'un code de longueur n d'une source X d'alphabet $\mathcal{X}$ est égale à $H(X)/n$. D'après le Théorème 2.7, on a $H(X) \leq \log_2 K$, donc d'après (3.2), on a $E \leq 1$. De plus $E = 1$ n'est possible qu'à deux conditions :

1. $H(X) = \log_2 K$, c'est-à-dire que les lettres de la source sont équiprobables.
2. $n = \log_2 K$, c'est-à-dire que le cardinal de l'alphabet est une puissance de 2.

Exemple: Soit une source dont l'alphabet est l'ensemble des chiffres décimaux $\mathcal{X} = \{0, 1, \ldots, 9\}$ muni de la loi de probabilité uniforme. Le code de longueur fixe d'une telle source a une longueur au moins 4. Par exemple

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

L'efficacité de ce code est égale à $H(X)/4 = (\log_2 10)/4 \approx 0,83$. Il est clair que ce code n'est pas optimal, puisque six mots binaires de longueur 4 sont inutilisés, donc ce code pourrait être utilisé pour une source ayant un cardinal 16.

Il est cependant possible d'améliorer l'efficacité du codage en considérant non plus des chiffres isolés mais des paires de chiffres. Ainsi la source peut être vue comme ayant pour alphabet l'ensemble $\mathcal{X}^2 = \{00, 01, \ldots, 99\}$ de cardinal 100. Si X est sans mémoire, la nouvelle source, notée X^2 reste munie d'une loi de probabilité uniforme, et son entropie vaut $H(X^2) = \log_2 100 = 2H(X)$.

La puissance de 2 immédiatement supérieure à 100 est $2^7 = 128$. Il existe donc un code régulier de X^2 de longueur 7. L'efficacité de ce code est cette fois égale à $H(X^2)/7 = (2\log_2 10)/7 \approx 0,95$ ce qui est meilleur.

En considérant la source X^3 de 1000 lettres, codées en 10 symboles binaires, on obtient une efficacité de $0,996$.

D'une façon générale, on peut considérer un code φ_L de longueur fixe de $\mathcal{X}^L$. D'après la Proposition 3.6 ce code permet d'obtenir une longueur moyenne par lettre

$$\frac{|\varphi_L|}{L} < \frac{1}{L} + \log_2 K.$$

C'est-à-dire que pour une source sans mémoire, on peut atteindre une efficacité arbitrairement proche de $H(X)/\log_2 K$. Si la source est uniforme ($H(X) = \log_2 K$), l'efficacité peut s'approcher de 1. Dans les autres cas ce n'est pas possible avec des codes de longueur fixe.

3.1.3 Codes de longueur variable

a - Codes à décodage unique - Codes préfixes

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

1. Utiliser un code de longueur fixe n, auquel cas la séquence binaire reçue est découpée en blocs de n symboles binaires qui seront décodés séparément.
2. Utiliser, comme pour le Morse, un symbole supplémentaire entre deux mots. Dans un tel cas, tout se passe comme si l'alphabet de sortie du codeur était augmenté d'une unité. Ainsi le code Morse peut être considéré comme un code ternaire et non binaire. Le troisième symbole servant de séparateur.

Le cas des *codes de longueur variable* est plus problématique en ce qui concerne le découpage des mots. Puisque deux mots consécutifs ont à priori des longueurs différentes, il faut concevoir des codes permettant la séparation sans ambiguïté des mots de code.

Définition 3.7 (Code déchiffrable) *Un code est dit* déchiffrable *ou* à décodage unique *si son codage associé est injectif.*

Cette définition signifie qu'il existe au plus une séquence de lettres de l'alphabet ayant permis d'obtenir une séquence binaire donnée.

Il existe une condition suffisante, dite du préfixe, pour qu'un code soit déchiffrable.

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

Définition 3.8 (Code préfixe) *Un code est dit* préfixe *s'il vérifie la condition du préfixe.*

Proposition 3.9 *Tout code préfixe est déchiffrable.*

Comme le montre l'exemple suivant, la condition du préfixe est une condition suffisante mais non nécessaire pour qu'un code soit déchiffrable.

Exemple: L ;alphabet $\{a_1, a_2, a_3\}$ est codée par $a_1 \to 1$, $a_2 \to 10$, $a_3 \to 100$. Les mots de codes peuvent être découpés en introduisant une séparation avant chaque '1' dans la séquence binaire, ce code est donc déchiffrable. Il n'est pourtant pas préfixe.

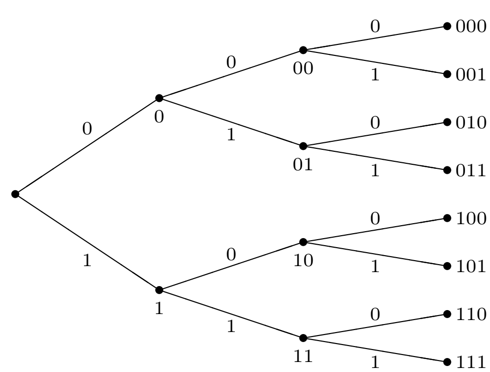
b - Représentation des codes préfixes par des arbres

Il est commode de représenter un code à l'aide d'un arbre binaire. Nous ne définirons pas ici les termes *branche, nœud, feuille, racine, fils, père*, qui auront leur signification usuelle. Les arbres que nous considérons ont les propriétés suivantes :

- chaque branche a pour attribut un des symboles binaires 0 ou 1,
- deux branches partant du même nœud ont des attributs différents,
- chaque nœud a pour attribut la concaténation des attributs des branches reliant la racine à ce nœud.

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

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

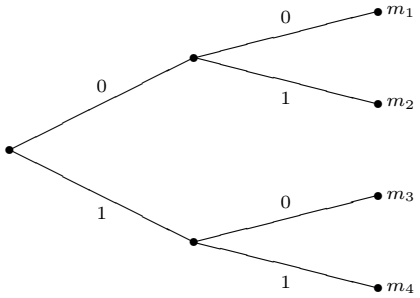


Pour représenter un code régulier, nous utiliserons le plus petit arbre contenant tous les mots de code, appelé *arbre du code*.

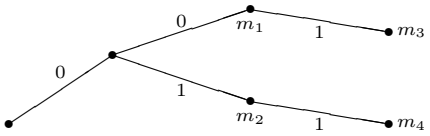
Exemple: Considérons l'alphabet $\{a_1, a_2, a_3, a_4\}$, et les trois codes suivants

	code 1	code 2	code 3
a_1	$m_1 = 00$	$m_1 = 00$	$m_1 = 0$
a_2	$m_2 = 01$	$m_2 = 01$	$m_2 = 10$
a_3	$m_3 = 10$	$m_3 = 001$	$m_3 = 110$
a_4	$m_4 = 11$	$m_4 = 011$	$m_4 = 111$

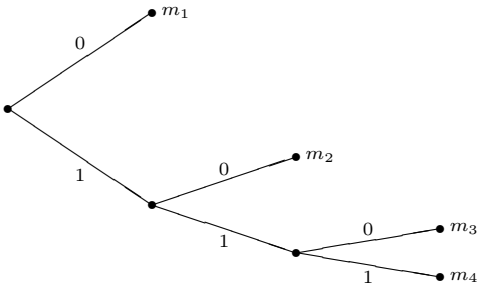
L'arbre du code 1 est donné par



L'arbre du code 2 est donné par



Et pour le code 3 l'arbre est



Nous constatons que pour les codes 1 et 3, qui sont préfixes, les mots de code sont exactement les feuilles de l'arbre, alors que pour le code 2 qui n'est pas préfixe, les mots de code m_1 et m_2 sont « à l'intérieur » de l'arbre.

Il existe une caractérisation simple des arbres des codes préfixes :

Proposition 3.10 *Un code est préfixe si et seulement si les feuilles de son arbre sont exactement ses mots de code.*

preuve: Dire qu'un mot de code est une feuille est équivalent à dire qu'il n'est pas le préfixe d'un autre mot de code. ◇

Définition 3.11 *Un code préfixe est dit* irréductible *si les nœuds de son arbre ont 0 ou 2 fils.*

Proposition 3.12 *Soit φ un code préfixe de $\mathcal{X}$. Il existe un code préfixe irréductible f tel que $|f(x)| \leq |\varphi(x)|$ pour tout $x \in \mathcal{X}$.*

Peuve en exercice.

c - Théorèmes de Kraft et McMillan

Il existe des conditions nécessaires et suffisantes pour l'existence de codes préfixes et déchiffrables. Pour les codes préfixes, il s'agit d'une propriété classique des arbres (inégalité de Kraft). Le résultat reste vrai pour un code déchiffrable (Théorème de Mac Millan).

Lemme 3.13 *Soit un arbre binaire dont les K feuilles ont pour ordre $n_1, \ldots, n_K$. Alors*

$$\sum_{k=1}^{K} 2^{-n_k} \leq 1.$$

preuve: Nous définissons le poids de chaque nœud de l'arbre de la façon suivante :

- le poids d'une feuille d'ordre i est égal à 2^{-i},
- le poids d'un nœud qui n'est pas une feuille est la somme des poids de ses fils.

1. Le poids d'un nœud d'ordre i est inférieur ou égal à 2^{-i}.
   Sinon, l'un de ses fils (d'ordre $i+1$) aurait un poids $> 2^{-(i+1)}$, et par récurrence, il existerait au moins une feuille de poids $> 2^h$, où h est l'ordre de cette feuille. Ceci serait contradictoire avec la définition du poids.
2. Le poids de la racine de l'arbre est égal à la somme des poids de ses feuilles.

La racine de l'arbre est un nœud d'ordre 0 et a donc un poids ≤ 1, ce qui nous donne l'inégalité recherchée. ◇

Lemme 3.14 *Soit un arbre binaire dont les K feuilles ont pour ordre $n_1, \ldots, n_K$ avec $m = \max(n_1, \ldots, n_K)$. Alors*

$$\left(1 - \sum_{k=1}^{K} 2^{-n_k} > 0\right) \Rightarrow \left(1 - \sum_{k=1}^{K} 2^{-n_k} \geq 2^{-m}\right)$$

preuve: En reprenant les notations de l'énoncé

$$\left(1 - \sum_{k=1}^{K} 2^{-n_k} > 0\right) \Rightarrow \left(2^m - \sum_{k=1}^{K} 2^{m-n_k} > 0\right)$$

Puisque pour tout k, $m \geq n_k$, le terme de gauche de l'inégalité de droite est entier, donc

$$\left(1 - \sum_{k=1}^{K} 2^{-n_k} > 0\right) \Rightarrow \left(2^m - \sum_{k=1}^{K} 2^{m-n_k} \geq 1\right) \Rightarrow \left(1 - \sum_{k=1}^{K} 2^{-n_k} \geq 2^{-m}\right).$$

◇

Théorème 3.15 (inégalité de Kraft) *Il existe un code préfixe de K mots de longueurs $n_1, \ldots, n_K$ si et seulement si l'inégalité*

$$\sum_{k=1}^{K} 2^{-n_k} \leq 1, \tag{3.3}$$

est satisfaite.

preuve:

1. Soit un code préfixe de K mots de longueurs $n_1, \ldots, n_K$, les mots de ce code sont exactement les feuilles de son arbre. D'autre part la longueur d'un mot est exactement égale à l'ordre du nœud le représentant dans l'arbre. Donc à l'aide du Lemme 3.13, on a (3.3).
2. Soient des entiers $1 \leq n_1 \leq \ldots \leq n_{K-1} \leq n_K$ vérifiant (3.3), montrons par récurrence sur K qu'il existe un arbre binaire dont les K feuilles ont pour ordre $n_1, \ldots, n_K$.
   (a) Si $K = 1$, l'arbre dégénéré suivant de profondeur n_1 convient :

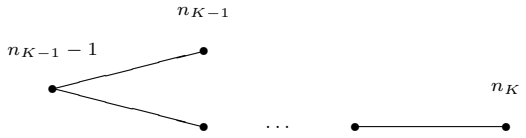
   (b) En enlevant la dernière longueur n_K, nous avons

   $$\sum_{k=1}^{K-1} 2^{-n_k} < 1$$

   donc, d'après le Lemme 3.14,

   $$\sum_{k=1}^{K-1} 2^{-n_k} + 2^{-n_{K-1}} = \sum_{k=1}^{K-2} 2^{-n_k} + 2^{1-n_{K-1}} \leq 1$$

On en déduit, par hypothèse de récurrence, qu'il existe un arbre dont les feuilles ont pour profondeurs $n_1, \ldots, n_{K-2}, n_{K-1} - 1$. Nous dotons la feuille à profondeur $n_{K-1} - 1$ de deux fils pour obtenir un arbre à K feuilles de profondeurs $n_1, \ldots, n_{K-2}, n_{K-1}, n_{K-1}$. Si $n_K > n_{K-1}$ il suffit de rallonger l'un des fils.



D'après la Proposition 3.10 cela suffit à montrer qu'il existe un code préfixe de K mots de longueur $n_1, \ldots, n_K$.

◇

Il existe un résultat similaire plus fort du à Mac Millan.

Théorème 3.16 (de Mac Millan) *Il existe un code déchiffrable de K mots de longueurs $n_1, \ldots, n_K$ si et seulement si l'inégalité*

$$\sum_{k=1}^{K} 2^{-n_k} \leq 1, \tag{3.4}$$

est satisfaite.

preuve: Si l'inégalité est satisfaite, alors il existe un code préfixe donc un code déchiffrable avec les bonnes longueurs.

Réciproquement, soit $\mathcal{X}$ un alphabet de cardinal K, soit φ un code déchiffrable de $\mathcal{X}$ et soit Φ son codage associé. Pour tout L-uplet $(x_1, \ldots, x_L) \in \mathcal{X}^L$, nous avons $\Phi(x_1, \ldots, x_L) = \varphi(x_1) \parallel \ldots \parallel \varphi(x_L)$ et donc $|\Phi(x_1, \ldots, x_L)| = |\varphi(x_1)| + \ldots + |\varphi(x_L)|$. Donc

$$\begin{aligned}
\left(\sum_x 2^{-|\varphi(x)|}\right)^L &= \sum_{x_1} \cdots \sum_{x_L} 2^{-|\varphi(x_1)|} \ldots 2^{-|\varphi(x_L)|} \\
&= \sum_{(x_1,\ldots,x_L)} 2^{-|\Phi(x_1,\ldots,x_L)|} \\
&= \sum_{i=1}^{mL} a_L(i) 2^{-i}
\end{aligned}$$

où $m = \max_{x \in \mathcal{X}} \ell(x)$ et $a_L(i)$ est le nombre de L-uplet de $\mathcal{X}^L$ dont le codage est de longueur i exactement. Le code φ est déchiffrable, donc pour tout i, nous avons $a_L(i) \leq 2^i$ (sinon il existerait au moins 2 L-uplets ayant la même image par Φ). Nous en déduisons que pour tout L

$$\sum_x 2^{-|\varphi(x)|} \leq (mL)^{1/L}.$$

D'où l'inégalité recherchée en faisant tendre L vers l'infini. ◇

Il est essentiel de noter que le Théorème 3.15 comme le Théorème 3.16 sont non constructifs. Ils nous donnent un résultat sur l'existence de codes dont les longueurs des mots de codes vérifient (3.4), mais ne prétendent pas que tout code dont les longueurs vérifient l'inégalité (3.4) est préfixe (ou déchiffrable).

3.2 Le premier théorème de Shannon pour une source discrète sans mémoire

Théorème 3.17 *Soit X une source discrète sans mémoire d'entropie H.*

1. *Tout code déchiffrable φ de X vérifie $|\varphi| \geq H$.*
2. *Il existe un code préfixe φ de X tel que $|\varphi| < H + 1$.*

preuve: Soit $X = (\mathcal{X}, p)$ une source discrète sans mémoire d'entropie H.

1. Soit φ un code déchiffrable de X. Montrons que $H - |\varphi| \leq 0$.

$$\begin{aligned} H - |\varphi| &= \sum_{x\in\mathcal{X}} p(x) \log_2 \frac{1}{p(x)} - \sum_{x\in\mathcal{X}} p(x)|\varphi(x)| \\ &= \sum_{x\in\mathcal{X}} p(x) \left(\log_2 \frac{1}{p(x)} + \log_2 2^{-|\varphi(x)|} \right) \\ &= \sum_{x\in\mathcal{X}} p(x) \log_2 \frac{2^{-|\varphi(x)|}}{p(x)} \end{aligned}$$

   D'après (2.2) page 15, nous en déduisons

$$H - |\varphi| = (\log_2 e) \left(\sum_{x\in\mathcal{X}} 2^{-|\varphi(x)|} - \sum_{x\in\mathcal{X}} p(x) \right) \leq 0$$

   en appliquant le Théorème 3.16 de Mac Millan.

2. Pour tout $x \in \mathcal{X}$, nous notons $\ell(x)$ l'unique entier vérifiant

$$2^{-\ell(x)} \leq p(x) < 2^{-\ell(x)+1}. \tag{3.5}$$

   En sommant l'inégalité de gauche nous obtenons

$$\sum_{x\in\mathcal{X}} 2^{-\ell(x)} \leq 1,$$

   ce qui nous assure d'après le Théorème 3.15 de Kraft qu'il existe un code préfixe dont les mots ont pour longueurs $\ell(x), x \in \mathcal{X}$. Soit φ un tel code, nous pouvons, sans perte de généralité, supposer que $|\varphi(x)| = \ell(x)$ pour tout x.
   En prenant le logarithme de l'inégalité de droite de (3.5), nous obtenons

$$\log_2 p(x) < -\ell(x) + 1,$$

   soit encore

$$|\varphi(x)| = \ell(x) < \log_2 \frac{1}{p(x)} + 1.$$

   En passant à la moyenne, nous obtenons

$$|\varphi| = \sum_x p(x)|\varphi(x)| < \sum_x p(x) \log_2 \frac{1}{p(x)} + 1 = H + 1.$$

◇

L'efficacité E du code préfixe produit par ce théorème vérifiera

$$1 - \frac{1}{H+1} < E \leq 1.$$

Cela nous montre que l'efficacité ne peut pas excéder 1. Cela ne suffit pourtant à prouver l'existence de codages dont l'efficacité s'approche arbitrairement de 1.

Pour obtenir de tels codages, il sera nécessaire, comme pour les codes de longueur fixe, d'effectuer le codage par blocs de lettres.

Théorème 3.18 (Premier théorème de Shannon) *Pour toute source discrète sans mémoire, il existe un codage déchiffrable dont l'efficacité est arbitrairement proche de 1.*

preuve: En terme mathématiques le théorème s'énonce :

> *Pour toute source discrète sans mémoire, pour tout réel $\varepsilon > 0$, il existe un codage déchiffrable dont l'efficacité est strictement supérieure à $1 - \varepsilon$.*

Soit X une source discrète sans mémoire. Soit $l > 0$, pour tout entier i tel que $l \geq i > 0$, nous appliquons le Théorème 3.17 à la source X^i pour obtenir un code préfixe φ_i de la source produit X^i tel que $iH \leq |\varphi_i| < iH + 1$.

Pour coder un mot $\mathbf{x}$ de $L = ml + r$ lettres, où $0 < r \leq l$, nous allons le découper en m blocs de l lettres de $\mathcal{X}$, notés $\mathbf{x}_1, \ldots, \mathbf{x}_{m+1}$ et un bloc de $r$lettres de $\mathcal{X}$, noté $\mathbf{x}_m$. Le codage du L-uplet sera

$$\Phi_l(\mathbf{x}) = \varphi_l(\mathbf{x}_1) \parallel \cdots \parallel \varphi_l(\mathbf{x}_m) \parallel \varphi_r(\mathbf{x}_{m+1}).$$

Ce codage est injectif (preuve laissée en exercice) et va nécessiter $|\varphi_l(\mathbf{x}_1)| + \cdots + |\varphi_l(\mathbf{x}_m)| + |\varphi_r(\mathbf{x}_{m+1})|$ symboles binaires pour les L lettres. En passant à la moyenne, le nombre de symboles binaires pour coder un L-uplet sera

$$m|\varphi_l| + |\varphi_r| \leq (lm + r)H + m + 1 = LH + 1 + \frac{L}{l},$$

et la longueur moyenne par lettre du codage Φ_l sera

$$\mathcal{L}(\Phi_l) = \lim_{L \to \infty} \frac{m|\varphi_l| + |\varphi_r|}{L} \leq H + \frac{1}{l}. \tag{3.6}$$

Et donc l'efficacité de ce codage vérifie

$$E(\Phi_l) \geq \frac{H}{H + 1/l} = 1 - \frac{1}{lH + 1}.$$

Pour $\varepsilon > 0$ fixé, dés que $l \geq 1/(\varepsilon H)$ nous aurons $E(\Phi_l) > 1 - \varepsilon$. ◇

3.3 Une procédure optimale de codage

3.3.1 Définition du code de Huffman

Un code de Huffman d'une source $X = (\mathcal{X}, p)$ est un code préfixe défini récursivement sur la taille K de son alphabet.

1. $K = 2$. Le code sera l'ensemble $\{0, 1\}$.
2. $K > 2$. Soit $\mathcal{X} = \{a_1, \ldots, a_{K-1}, a_K\}$ avec $p(a_1) \geq \ldots \geq p(a_{K-1}) \geq p(a_K)$. Un code de Huffman de X sera défini par

$$\left\{ \begin{array}{rcll} \varphi(a_k) & = & \psi(b_k) & \text{si } k \leq K-2 \\ \varphi(a_{K-1}) & = & \psi(b_{K-1}) \parallel 0 & \\ \varphi(a_K) & = & \psi(b_{K-1})) \parallel 1 & \end{array} \right.$$

où ψ est un code Huffman de la source $Y = (\mathcal{Y}, q)$, avec $Y = \{b_1, \ldots, b_{K-1}\}$ et

$$\left\{ \begin{array}{rcll} q(b_k) & = & p(a_k) & \text{si } k \leq K-2 \\ q(b_{K-1}) & = & p(a_{K-1}) + p(a_K) & \end{array} \right.$$

3.3.2 Le code de Huffman est optimal

Définition 3.19 *Un code déchiffrable d'une source est dit* optimal *s'il n'existe aucun code déchiffrable de cette source dont la longueur moyenne soit strictement inférieure.*

Lemme 3.20 *Soit f un code d'une source $X = (\mathcal{X}, p)$. Soient u et v deux lettres de $\mathcal{X}$, non nécessairement distinctes, telles que $p(u) \leq p(v)$ et $|f(u)| \leq |f(v)|$, alors le code g défini par $g(u) = f(v)$, $g(v) = f(u)$ et $g(x) = f(x)$, $x \notin \{u, v\}$, vérifie $|g| \leq |f|$.*

preuve: Nous avons

$$|f| - |g| = (p(v) - p(u))(|f(v)| - |f(u)|) \geq 0.$$

◇

Lemme 3.21 *Soit une source $X = (\mathcal{X}, p)$ telle que $\mathcal{X} = \{a_1, \ldots, a_{K-1}, a_K\}$ avec $p(a_1) \geq \ldots \geq p(a_{K-1}) \geq p(a_K)$. Il existe un code préfixe optimal f tel que $f(a_{K-1}) = m \parallel 0$ et $f(a_K) = m \parallel 1$.*

preuve: Soit g un code préfixe optimal de X. D'après la Proposition 3.12, on peut supposer qu'il est irréductible.

Soit m un prefixe d'un mot de code de longueur maximale, le code est irréductible, donc $m \parallel 0$ et $m \parallel 1$ sont des mots de code (de longueur maximale).

Soit u dans $\mathcal{X}$ tel que $g(u) = m \parallel 0$. Le code h est obtenu à partir de g en échangeant les images de u et a_{K-1}. Soit v dans $\mathcal{X}$ tel que $h(v) = m \parallel 1$. Le code f est obtenu à partir de h en échangeant les images de v et a_K. D'après le Lemme 3.20, nous avons $|f| \leq |h| \leq |g|$ donc f est optimal et nous avons bien $f(a_{K-1}) = m \parallel 0$ et $f(a_K) = m \parallel 1$. ◇

Proposition 3.22 *Le code de Huffman d'une source est optimal.*

preuve: Nous reprenons les notations de la description des codes de Huffman. Soit f un code préfixe optimal de X vérifiant le Lemme 3.21. Soit g le code de Y défini par

$$\begin{cases} g(b_k) &= f(a_k) \quad \text{si } k \leq K-2 \\ g(b_{K-1}) &= m \end{cases}$$

Nous avons $|f| - |g| = |\varphi| - |\psi| = q(b_{K-1})$. Si φ n'est pas optimal, alors $|f| < |\varphi|$, donc $|g| < |\psi|$, et donc ψ n'est pas optimal. Donc par contraposée, si ψ est optimal alors φ est optimal. Pour $K = 2$, le code de Huffman est optimal, donc, par récurrence, tout code de Huffman est optimal. ◇

3.4 Autres algorithmes de codage de source – Code arithmétique

Avant que Huffmann ne propose une méthode optimale, d'autres techniques avaient été proposées. Si elle ne sont pas aussi efficaces pour une source finie de petite taille, leur génése est néanmoins intéressante. En particulier, elles contiennent les idées qui ont conduit au codage arithmétique.

3.4.1 Préliminaires : développements 2-adiques

Tout nombre réel x de l'intervalle $[0, 1[$ peut s'écrire

$$x = \sum_{i=1}^{\infty} d_i \, 2^{-i}, \text{ où } d_i \in \{0, 1\}.$$

Cette écriture n'est pas unique car $2^{-j} = \sum_{i>j} 2^{-i}$. Si on impose que la suite des d_i contient une infinité de 0 (c'est-à-dire ne vaut pas 1 au delà d'un certain rang), alors l'écriture devient unique, on appellera *développement 2-adique* la suite correspondante. Nous noterons $D_\ell(x) = (d_1, \ldots, d_\ell)$ les ℓ premiers bits du développement 2-adique de x.

Lemme 3.23 *Soient u et v deux réels dans $[0,1[$, soit un entier $\ell > 0$. Si $D_\ell(u) = D_\ell(v)$ alors $|u-v| < 2^{-\ell}$.*

preuve: Soient

$$u = \sum_{i>0} u_i 2^{-i} \text{ et } v = \sum_{i>0} v_i 2^{-i}.$$

Si les développements à sont égaux à l'ordre ℓ, alors

$$|u-v| = \left| \sum_{i>\ell} (u_i - v_i) 2^{-i} \right| \leq \sum_{i>\ell} |u_i - v_i| \, 2^{-i} \leq \sum_{i>\ell} 2^{-i} = 2^{-\ell}.$$

L'égalité $|u-v| = 2^{-\ell}$ implique que tous les $u_i - v_i$ ont la même valeur (1 ou -1) et donc que l'une des suites terminerait par des 1, ce qui est interdit. Donc $|u-v| < 2^{\ell}$. ◇

3.4.2 Code de Shannon-Fano-Elias

Soit une source discrète sans mémoire $X = (\mathcal{X}, p)$. Nous supposerons que $p(x) > 0$ pour tout x. Soit φ un code déchiffrable de $\mathcal{X}$.

Si la longueur $|\varphi(x)|$ du mot de codant x est proche de l'information propre $I(x) = -\log_2 p(x)$ de x, alors la longueur moyenne $|\varphi|$ du code sera proche de la moyenne de l'information propre, c'est-à-dire l'entropie.

L'idée est de construire un code prefixe (donc déchiffrable) dans lequel le mot codant x aura une longueur « pas trop supérieure » à $-\log_2 p(x)$.

Nous munissons l'alphabet $\mathcal{X}$ d'une relation d'ordre total notée $<$. Pour tout x dans $\mathcal{X}$ nous noterons

$$S(x) = \sum_{x'<x} p(x'), \bar{S}(x) = \frac{p(x)}{2} + S(x) \text{ et } \ell(x) = \lceil -\log_2 p(x) \rceil .$$

Proposition 3.24 (code de Shannon-Fano-Elias) *Soit le code défini pour tout $x \in \mathcal{X}$ par $\varphi(x) = D_{\ell(x)+1}(\bar{S}(x))$.*

1. *Le code φ est préfixe.*
2. *$|\varphi| < H(X) + 2$.*

preuve:

1. Soient x et y distincts dans $\mathcal{X}$. Montrons que $\varphi(x)$ n'est pas un préfixe de $\varphi(y)$.
   Nous pouvons supposer que $\ell(x) \leq \ell(y)$ (sinon $\varphi(x)$ est plus long que $\varphi(y)$ et ne peut pas être son préfixe).
   Si $y > x$, nous avons

$$\begin{aligned}
\bar{S}(y) - \bar{S}(x) &= \frac{p(y)}{2} + \sum_{x'<y} p(x') - \frac{p(x)}{2} - \sum_{x'<x} p(x') \\
&= \frac{p(y)}{2} - \frac{p(x)}{2} + \sum_{x \leq x' < y} p(x') \\
&= \frac{p(y)}{2} + \frac{p(x)}{2} + \sum_{x < x' < y} p(x') > \max\left(\frac{p(y)}{2}, \frac{p(x)}{2}\right) \\
&> \max\left(2^{-\ell(y)-1}, 2^{-\ell(x)-1}\right)
\end{aligned}$$

   Si $x > y$, nous montrons de la même manière que $\bar{S}(x) - \bar{S}(y) > \max\left(2^{-\ell(y)-1}, 2^{-\ell(x)-1}\right)$.
   Et donc, puisque $\ell(x) \leq \ell(y)$

$$\left|\bar{S}(x) - \bar{S}(y)\right| > \max\left(2^{-\ell(x)-1}, 2^{-\ell(y)-1}\right) = 2^{-\ell(x)-1}.$$

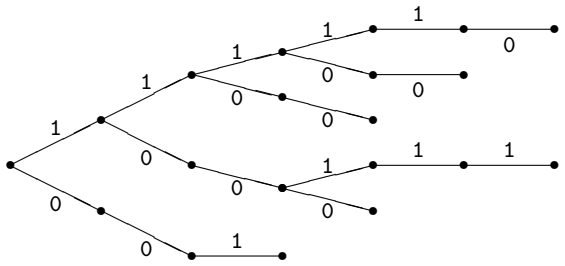
Le Lemme 3.23 nous permet de conclure que $\varphi(x)$ et $\varphi(y)$ ne coincident pas sur leurs $\ell(x)+1$ premiers bits. Donc $\varphi(x)$, qui est de longueur $\ell(x)+1$, n'est pas préfixe de $\varphi(y)$.

2. Pour tout x, nous avons $|\varphi(x)| = \ell(x) + 1 < -\log_2 p(x) + 2$. En passant à la moyenne, $|\varphi| < H(X) + 2$.

◇

Exemple:

x	$p(x)$	$\ell(x)$	$\bar{S}(x)$		$\varphi(x)$	Huffman
a	0.43	2	0.215	$0.0011011\ldots$	001	0
b	0.17	3	0.515	$0.1000001\ldots$	1000	110
c	0.04	5	0.620	$0.1001111\ldots$	100111	11111
d	0.22	3	0.750	$0.1100000\ldots$	1100	10
e	0.09	4	0.905	$0.1110011\ldots$	11100	1110
f	0.05	5	0.975	$0.1111100\ldots$	111110	11110



On constate que ce code n'est pas optimal. Il est bien préfixe, mais son arbre possède de nombreux nœuds avec un seul fils.

On notera également qu'en retirant le dernier symbole, le code n'est plus préfixe.

Il n'existe pas de transformation simple permettant de rendre ce code optimal. Toutefois, il existe une variante (probablement due à Shannon) qui améliore la longueur moyenne.

Proposition 3.25 *L'ordre sur $\mathcal{X}$ est celui des probabilités décroissantes. Soit le code défini pour tout $x \in \mathcal{X}$ par $\varphi(x) = D_{\ell(x)}(S(x))$.*

1. *Le code φ est préfixe.*
2. *$|\varphi| < H(X) + 1$.*

preuve:

1. Soient $x < y$ dans $\mathcal{X}$, nous avons $|\varphi(x)| = \ell(x) \leq \ell(y) = |\varphi(y)|$ car $p(x) \geq p(y)$. Montrons que $\varphi(x)$ n'est pas un préfixe de $\varphi(y)$.

$$S(y) - S(x) = p(x) + \sum_{x<x'<y} p(x') \geq p(x) \geq 2^{-\ell(x)}$$

Le Lemme 3.23 nous permet de conclure que $\varphi(x)$ et $\varphi(y)$ ne coincident pas sur leurs $\ell(x)$ premiers bits. Donc $\varphi(x)$, qui est de longueur $\ell(x)$, n'est pas préfixe de $\varphi(y)$.

2. Pour tout x, nous avons $|\varphi(x)| = \ell(x) < -\log_2 p(x) + 1$. En passant à la moyenne, $|\varphi| < H(X) + 1$.

◇

Exemple: avec la même source, nous obtenons

x	$p(x)$	$\ell(x)$	$S(x)$		$\varphi(x)$	Huffman
a	0.43	2	0.00	$0.0000000\ldots$	00	0
b	0.22	3	0.43	$0.0110111\ldots$	011	10
c	0.17	3	0.65	$0.1010011\ldots$	101	110
d	0.09	4	0.82	$0.1101001\ldots$	1101	1110
e	0.05	5	0.91	$0.1110100\ldots$	11101	11110
f	0.04	5	0.96	$0.1111011\ldots$	11110	11111

Ce code est meilleur, mais toujours pas optimal.

Chapitre 4

Les séquences typiques et l'AEP

4.1 Sources discrètes stationnaires

Dans le cas général, une source discrète produit des lettres d'un alphabet selon une loi de probabilité qui peut évoluer au cours du temps. Pour modéliser une source nous utiliserons un processus stochastique, c'est-à-dire une suite $X_1, X_2, \ldots, X_n, \ldots$ de variables aléatoires à valeur dans $\mathcal{X}$ (cf. §2.2).

Pour tout $L > 0$, nous pouvons définir l'entropie des L premières lettres par

$$H(X_1, \ldots, X_L) = \sum_{x_1,\ldots,x_L} -\Pr[X_1 = x_1, \ldots, X_L = x_L] \log_2 \Pr[X_1 = x_1, \ldots, X_L = x_L]$$

En divisant cette quantité par L, nous obtenons la quantité d'information moyenne fournie par chacune des L premières lettre de la source. Ce qui nous conduit à la définition suivante, valable pour tout processus stochastique.

Définition 4.1 *Soit un processus stochastique* $(X_1, \ldots, X_n, \ldots)$. *Son* entropie par lettre *est définie par*

$$H(\mathcal{X}) = \lim_{L\to\infty} \frac{1}{L} H(X_1, \ldots, X_L),$$

si cette limite existe.

En pratique nous nous intéresserons plus particulièrement aux sources stationnaires, c'est-à-dire que les probabilités du processus associé sont inchangées si on décale l'observation dans le temps :

$$\Pr[X_1 = x_1, X_2 = x_2, \ldots, X_L = x_L] = \Pr[X_{1+\delta} = x_1, X_{2+\delta} = x_2, \ldots, X_{L+\delta} = x_L].$$

Pour un processus stochastique nous pourrons noter p la distribution de probabilité et utiliser sans ambiguïté la notation

$$p(x_1, \ldots, x_L) = \Pr[X_1 = x_1, \ldots, X_L = x_L].$$

L'entropie conditionnelle sera définie par :

$$H(X_L \mid X_1, \ldots, X_{L-1}) = \sum_{x_1,\ldots,x_L} -p(x_1, \ldots, x_L) \log_2 p(x_L \mid x_1, \ldots, x_{L-1})$$

Cette grandeur représente la quantité moyenne d'information supplémentaire apportée par l'occurence de la L-ième lettre. Nous allons voir que cette quantité converge pour une source stationnaire et qu'elle est égale à l'entropie par lettre.

Proposition 4.2 *L'entropie par lettre de tout processus stochastique stationnaire* $((X_i)_{i>0}, p)$ *est définie, de plus elle est égale à*

$$H(\mathcal{X}) = \lim_{L\to\infty} H(X_L \mid X_1, \ldots, X_{L-1}).$$

preuve:

1. Montrons d'abord que $\lim_{L\to\infty} H(X_L \mid X_1, \ldots, X_{L-1})$ existe pour un processus stationnaire. D'après la Proposition 2.9, le conditionnement réduit l'entropie, donc pour tout L

$$H(X_L \mid X_1, \ldots, X_{L-1}) \leq H(X_L \mid X_2, \ldots, X_{L-1}) = H(X_{L-1} \mid X_1, \ldots, X_{L-2})$$

   Ainsi, la suite $(H(X_i \mid X_1, \ldots, X_{i-1}))_{i>0}$ est décroissante. Comme elle est positive, elle est convergente.

2. Pour tout $L > 0$, nous avons

$$\begin{aligned} p(x_1, \ldots, x_L) &= p(x_L \mid x_1, \ldots, x_{L-1}) p(x_1, \ldots, x_{L-1}), \\ p(x_1, \ldots, x_{L-1}) &= \sum_{x_L} p(x_1, \ldots, x_L). \end{aligned}$$

   Nous en déduisons que

$$H(X_1, \ldots, X_L) = H(X_L \mid X_1, \ldots, X_{L-1}) + H(X_1, \ldots, X_{L-1}),$$

   et donc

$$H(X_1, \ldots, X_L) = \sum_{i=1}^{L} H(X_i \mid X_1, \ldots, X_{i-1}).$$

   Un résultat classique d'analyse (lemme de Césaro) nous permet de conclure que comme $(H(X_i \mid X_1, \ldots, X_{i-1}))_{i>0}$ converge, c'est le cas aussi de $(\frac{1}{i} H(X_1, \ldots, X_i))_{i>0}$, de plus ces limites sont identiques.

◇

Entropie d'un processus markovien irreductible invariant dans le temps. Soit $((X_i)_{i>0}, p)$ un processus markovien irreductible invariant dans le temps d'ordre 1. Soit $\lambda(x), x \in \mathcal{X}$ sa distribution stationnaire. Son entropie par lettre est égale à

$$H(\mathcal{X}) = \lim_{L\to\infty} H(X_L \mid X_{L-1}) = \sum_{x,x'} -\lambda(x) p(x' \mid x) \log_2 p(x' \mid x).$$

Si le processus est stationnaire la situation est plus simple car alors sa distribution initiale est égale à la distribution stationnaire et donc $H(\mathcal{X}) = H(X_2 \mid X_1)$. Autrement dit, il suffit de remplacer $\lambda(x)$ par $p(x)$ dans la formule.

4.2 Les séquences typiques

Dans cette section, nous considérons un processus stochastique sur un alphabet $\mathcal{X}$.

Définition 4.3 (séquences typiques) *Soit un processus stochastique sur un alphabet $\mathcal{X}$ pour lequel l'entropie par lettre $\mathcal{H}$ est définie. Pour tout entier $n > 0$ et tout réel $\varepsilon > 0$ nous définissons*

$$A_\varepsilon^{(n)} = \left\{ (x_1, \ldots, x_n) \in \mathcal{X}^n, \left| \frac{1}{n} \log_2 \frac{1}{\Pr[X_1 = x_1, \ldots, X_n = x_n]} - \mathcal{H} \right| \leq \varepsilon \right\}.$$

Lorsque ε est petit et que n est grand, les éléments de $A_\varepsilon^{(n)}$ sont les n-uplets que l'on s'attend à voir produire par la source. Il faut noter que les séquences les plus probables ne sont pas forcément typiques.

Exemple: Si on considère une source binaire produisant des 0 avec une probabilité 2/3 et des 1 avec une probabilité 1/3, la séquence ne comportant que des 1 est peu probable et n'est pas typique, en revanche la séquence ne comportant que des 0, bien qu'étant la plus probable, n'est pas typique.

Les séquences typiques seront celles qui comportent environ 1/3 de 1.

Les séquences typiques sont un outil technique permettant de démontrer facilement des résultats difficiles de théorie de l'information. En pratique, pour une source ayant un comportement raisonnable (stationnaire, markovien, ...), la probabilité d'observer une séquence typiques (pour n suffisamment grand) tend vers 1. Cette propriété sur la mesure des ensembles de séquences typiques est appelée AEP (*Asymptotic Equipartition Property*). Il s'agit de la propriété minimale pour énoncer le premier théorème de Shannon.

Définition 4.4 (*Asymptotic Equipartition Property*) *Un processus stochastique (une source) vérifie l'AEP si pour tout* $\varepsilon > 0$

$$\lim_{n\to\infty} \Pr[A_\varepsilon^{(n)}] = 1.$$

Proposition 4.5 *Pour toute source vérifiant l'AEP*

(i) $\lim_{n\to\infty} \frac{1}{n} \log_2 \frac{1}{\Pr[X_1 = x_1, \ldots, X_n = x_n]} = \mathcal{H}$ *avec une probabilité 1.*

(ii) $\lim_{\varepsilon\to 0} \lim_{n\to\infty} \frac{1}{n} \log_2 |A_\varepsilon^{(n)}| = \mathcal{H}$

Autrement dit, pour n assez grand et $\varepsilon > 0$ assez petit

« Il existe $2^{n\mathcal{H}}$ *séquences typiques de longueur* n *qui sont équiprobables de probabilité* $2^{-n\mathcal{H}}$ *»*

preuve: (informelle)

(i) Soit $\varepsilon > 0$, pour tout n nous avons par définition

$$\left| \frac{1}{n} \log_2 \frac{1}{\Pr[X_1 = x_1, \ldots, X_n = x_n]} - \mathcal{H} \right| \leq \varepsilon$$

avec une probabilité $\Pr[A_\varepsilon^{(n)}]$. Si la source vérifie l'AEP, nous en déduisons que

$$\mathcal{H} - \varepsilon \leq \liminf_{n\to\infty} \frac{1}{n} \log_2 \frac{1}{\Pr[X_1 = x_1, \ldots, X_n = x_n]} \leq \limsup_{n\to\infty} \frac{1}{n} \log_2 \frac{1}{\Pr[X_1 = x_1, \ldots, X_n = x_n]} \leq \mathcal{H} + \varepsilon$$

avec une probabilité 1. En faisant tendre ε vers 0, nous obtenons la propriété souhaitée.

(ii) Soient $\varepsilon > 0$ et n un entier. Nous avons

$$|A_\varepsilon^{(n)}| 2^{-n(\mathcal{H}+\varepsilon)} \leq \sum_{(x_1,\ldots,x_n)\in A_\varepsilon^{(n)}} \Pr[X_1 = x_1, \ldots, X_n = x_n] = \Pr[A_\varepsilon^{(n)}] \leq 1,$$

et donc $|A_\varepsilon^{(n)}| \leq 2^{n(\mathcal{H}+\varepsilon)}$. Inversement

$$\Pr[A_\varepsilon^{(n)}] = \sum_{(x_1,\ldots,x_n)\in A_\varepsilon^{(n)}} \Pr[X_1 = x_1, \ldots, X_n = x_n] \leq |A_\varepsilon^{(n)}| 2^{-n(\mathcal{H}-\varepsilon)},$$

et donc $|A_\varepsilon^{(n)}| \geq \Pr[A_\varepsilon^{(n)}] 2^{n(\mathcal{H}+\varepsilon)}$. En prenant le logarithme

$$\mathcal{H} - \varepsilon + \frac{1}{n} \log_2 \Pr[A_\varepsilon^{(n)}] \leq \frac{1}{n} \log_2 |A_\varepsilon^{(n)}| \leq \mathcal{H} + \varepsilon$$

$$\mathcal{H} - \varepsilon \leq \liminf_{n\to\infty} \frac{1}{n} \log_2 |A_\varepsilon^{(n)}| \leq \limsup_{n\to\infty} \frac{1}{n} \log_2 |A_\varepsilon^{(n)}| \leq \mathcal{H} + \varepsilon.$$

De nouveau, le résultat est obtenu en faisant tendre ε vers 0.

◇

4.3 Théorème de Shannon

Soit Φ un codage d'une source sur $\mathcal{X}$ vérifiant l'AEP d'entropie par lettre $\mathcal{H}$. La longueur moyenne par lettre de Φ est définie comme précédemment comme la limite

$$\mathcal{L}(\Phi) = \lim_{n\to\infty} \frac{1}{n} \sum_{(x_1,\ldots,x_n)\in\mathcal{X}^n} \Pr[X_1 = x_1, \ldots, X_n = x_n] \, |\Phi(x_1, \ldots, x_n)| \,,$$

si celle-ci existe. Dans ce cas l'efficacité du codage Φ est égal à

$$E(\Phi) = \frac{\mathcal{H}}{\mathcal{L}(\Phi)}.$$

Théorème 4.6 *Soit une source vérifiant l'AEP.*

1. *Tout codage régulier de cette source à un efficacité* ≤ 1
2. *Il existe un codage déchiffrable de cette source d'efficacité arbitrairement proche de 1.*

preuve: Soit une source vérifiant l'AEP, nous notons $\mathcal{X}$ son alphabet et p sa distribution de probabilité.

1. Pour n fixé, d'après les résultats sur les sources sans mémoire, la restriction de tout codage régulier Φ de $\mathcal{X}$ à $\mathcal{X}^n$ doit vérifier

$$\frac{1}{n} \sum_{x_1,\ldots,x_n} \Pr[X_1 = x_1, \ldots, X_n = x_n] |\Phi(x_1, \ldots, x_n)| \geq \frac{1}{n} H(X_1, \ldots, X_n),$$

   en passant à la limite $n \to \infty$ nous obtenons $\mathcal{L}(\Phi) \geq \mathcal{H}$.

2. Soit $\varepsilon > 0$. Pour tout entier n, soient
   (a) F_n un code de longueur fixe minimale de $\mathcal{X}^n$
   (b) $G_{n,\varepsilon}$ un code de longueur fixe minimale de $A_\varepsilon^{(n)}$.
   (c) $\varphi_{n,\varepsilon}$ un code de $\mathcal{X}^n$ défini par

$$\varphi_{n,\varepsilon}(x_1, \ldots, x_n) = \begin{cases} 0 \parallel G_{n,\varepsilon}(x_1, \ldots, x_n) & \text{si } (x_1, \ldots, x_n) \in A_\varepsilon^{(n)} \\ 1 \parallel F_n(x_1, \ldots, x_n) & \text{sinon} \end{cases}$$

   Les longueurs moyenne de ces codes sont ($\lceil\cdot\rceil$ la partie entière par excès)

$$\begin{aligned} |F_n| &= \lceil n \log_2 |\mathcal{X}| \rceil \\ |G_{n,\varepsilon}| &= \lceil \log_2 |A_\varepsilon^{(n)}| \rceil \\ |\varphi_{n,\varepsilon}| &= 1 + \Pr[A_\varepsilon^{(n)}] |G_{n,\varepsilon}| + |(1 - \Pr[A_\varepsilon^{(n)}]) |F_n| \end{aligned}$$

   Soit $L = qn + r$ avec $0 \leq r < n$. Tout mot $(x_1, \ldots, x_L)$ de $\mathcal{X}^L$ est découpé en q mots de $\mathcal{X}^n$ noté $\mathbf{x}_1, \ldots, \mathbf{x}_q$ et un mot de $\mathcal{X}^r$ noté $\mathbf{x}_{q+1}$. L'image de $(x_1, \ldots, x_L)$ par $\Phi_{n,\varepsilon}$ de $\mathcal{X}$ est définie par

$$\Phi_{n,\varepsilon}(x_1, \ldots, x_L) = \varphi_{n,\varepsilon}(\mathbf{x}_1) \parallel \cdots \parallel \varphi_{n,\varepsilon}(\mathbf{x}_q) \parallel F_r(\mathbf{x}_{q+1})$$

   En faisant tendre L vers l'infini, nous obtenons $\mathcal{L}(\Phi_{n,\varepsilon}) = |\varphi_{n,\varepsilon}|/n$. Nous en déduisons que pour tout $\varepsilon > 0$

$$\lim_{n\to\infty} \mathcal{L}(\Phi_{n,\varepsilon}) \leq \mathcal{H} + \varepsilon.$$

   Par un choix approprié de ε puis de n, cela nous permet de définir un codage de longueur moyenne arbitrairement proche de $\mathcal{H}$ c'est-à-dire d'efficacité arbitrairement proche de 1.

◇

Toute source sans mémoire vérifie l'AEP et donc vérifie le théorème de Shannon (nous l'avons montré directement). Ce sera le cas également des processus markoviens.

Proposition 4.7 *Toute source markovienne irreductible invariante dans le temps vérifie l'AEP.*

Le cas le plus général dans lequel le théorème de Shannon sera vérifié, est celui des sources ergodiques. Nous n'en donnerons pas de définition formelle, mais intuitivement un processus stochastique sur $\mathcal{X}$ est ergodique si l'on peut complètement déterminer la loi de probabilité par l'observation d'une instance du processus (i.e. une suite infinie de lettre de $\mathcal{X}$). Un processus ergodique vérifie la loi des grand nombres, c'est-à-dire qu'en attendant suffisamment longtemps on observera un « comportement moyen » conforme à ce que l'on pouvait prévoir (i.e. toutes les séquences « observables » suffisamment longues sont typiques).

Chapitre 5

Codage de source universel

Le codage de source universel fait référence aux techniques de codage de source n'utilisant aucune connaissance *a priori* des statistiques de la source. Il peut s'agir de méthodes adaptatives (Huffmann, codage arithmétique), dans lesquelles le codage à un instant donné se fait en fonction des probabilités connues à ce stade, ou bien de méthodes dites « par dictionnaire » (toutes les variantes de Lempel-Ziv), dans lesquelles chaque mot est remplacé par son indice dans un dictionnaire (mis à jour en permanence).

Le chapitre ci-après n'est pas exhaustif. Sont décrits l'algorithme de Huffman adaptatif ansi que deux variantes de l'algorithme de Lempel-Ziv. Il existe de nombreux autres algorithmes : codage arithmétique adaptatif, transformée de Burrows-Wheeler (bzip), ou le LZMA (« Lempel-Ziv-Markov chain-Algorithm », 7-zip) qui ne seront pas décrits ici.

5.1 Algorithme de Huffmann adaptatif

Une première approche consisterait à imaginer un algorithme en deux passes, la première pour obtenir les statistiques sur les fréquences d'apparition des lettres dans le texte à compresser, et la seconde pour coder le texte à partir de l'arbre ainsi construit. Au delà des deux passes qui poseront souvent des problèmes, il sera nécessaire de transmettre en plus du fichier codé l'arbre ayant permis de coder le fichier, ce qui sera souvent inefficace. En pratique, on préfèrera coder la $n+1$-ème lettre d'un texte à l'aide d'un code de Huffman obtenu à partir des statistiques mesurées pour les n premières lettres.

Par exemple, supposons que nous ayons déjà lu n caractères dans le texte, correspondant à $K-1$ lettres distinctes.

- nous fabriquons X_n une source de K lettres formée des $K-1$ lettres déjà apparues auxquelles on attribue une probabilité proportionnelle au nombre d'occurences, et d'une K-ème lettre « fourre-tout » (vide) à laquelle on attribue la probabilité 0,
- on construit un code de Huffman T_n de cette source,
- la $n+1$-ème lettre est lue et est codée
  - par son mot de code s'il existe,
  - par le K-ème mot de code suivi du code ascii (par exemple) sinon.

Comme les statistiques utilisées ne dépendent que du passé, le décodeur pourra les connaître et ainsi avoir connaissance à tout moment du code utilisé pour chaque lettre.

5.1.1 Une description sommaire de l'algorithme

Les procédures de codage et de décodage lisent un flux d'entrée qui peut contenir des lettres, typiquement des octets (c'est d'ailleurs le seul type de donnée pour le codage), ou des mots de code (binaires en général). Pour simplifier la description la fonction `lire_mot_de_code` lit un mot de code dans le flux mais retourne la lettre qui lui est associée dans le code.

Codage

entrée : un flux de données F
  $T \leftarrow (\emptyset)$ // *l'arbre à une feuille vide*
  $\mathcal{A} \leftarrow \emptyset$
  tantque F non vide
    $x \leftarrow$ `lire_lettre`(F)
    si $x \in \mathcal{A}$
      `ecrire_mot_de_code`(x, T)
      $T \leftarrow$ `mise_à_jour`(x, T)
    sinon
      `ecrire_mot_de_code`$(\emptyset, T)$
      `ecrire_lettre`(x)
      $T \leftarrow$ `ajoute_feuille`(x, T)
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{x\}$

Décodage

entrée : un flux de données F
  $T \leftarrow (\emptyset)$ // *l'arbre à une feuille vide*
  $\mathcal{A} \leftarrow \emptyset$
  tantque F non vide
    $x \leftarrow$ `lire_mot_de_code`(T, F)
    si $x \in \mathcal{A}$
      `ecrire_lettre`(x)
      $T \leftarrow$ `mise_à_jour`(x, T)
    sinon
      $x \leftarrow$ `lire_lettre`(F)
      `ecrire_lettre`(x)
      $T \leftarrow$ `ajoute_feuille`(x, T)
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{x\}$

TABLE 5.1 – Algorithme de Huffman adaptatif

Comme il est évidemment exclus de recalculer l'arbre de Huffman à chaque lettre, il faut trouver une méthode efficace de mise à jour l'arbre (fonctions `mise_à_jour` et `ajoute_feuille`). La méthode qui suit est due à D. Knuth.

5.1.2 Mise en œuvre de l'algorithme de Huffman adaptatif

a - Préliminaires

Nous identifions un code préfixe d'une source discrète X à un arbre binaire à $|X|$ feuilles étiquetées par les lettres de X. Le code est dit complet si chaque nœud possède 0 ou 2 fils.

Définition 5.1 *Soit T un code prefixe d'une source discrète X, le poids relativement à (T, X) est défini de la façon suivante*
- *le* poids d'une feuille *est la probabilité de sa lettre*
- *le* poids d'un nœud *est la somme du poids de ses fils*

Définition 5.2 *Un* arbre de Huffman *de X est un code préfixe de X qui peut être obtenu par l'algorithme de Huffman.*

Définition 5.3 *Soit T un code prefixe complet d'une source discrète X de cardinal K. Un* ordre de Gallager *sur T relativement à X est un ordre $u_1, \ldots, u_{2K-1}$ sur les nœuds de T vérifiant*

1. *les poids des u_i relativement à (T, X) sont décroissants,*
2. *u_{2i} et u_{2i+1} sont frères dans T pour tout i, $1 \leq i < K$.*

Théorème 5.4 (Gallager) *Un code préfixe T (i.e. un arbre) d'une source X est un arbre de Huffman de X si et seulement si il existe un ordre de Gallager sur T relativement à X.*

Ce théorème sera admis. En particulier si un arbre associé à un code préfixe possède un ordre de Gallager sur ses nœuds, alors il est optimal.

b - Notations

Nous considérons un texte $(x_1, x_2 \ldots, x_n, \ldots)$ et une instance de l'algorithme de Huffman adaptatif sur ce texte. Nous notons X_n la source obtenue après lecture de la n-ème lettre et T_n un arbre de Huffman de cette source.

La probabilité d'une lettre de la source X_n est proportionnelle à son nombre d'occurence dans les n premières lettres du texte. Ainsi sans rien changer aux définitions (en particulier pour un ordre de Gallager), nous pourrons utiliser pour le *poids* le nombre d'occurence de chaque lettre (la lettre vide a une nombre d'occurences égal à 0). Notons que dans ce cas, le poids est entier et que le poids de la racine de T_n est n.

Soit x la $(n+1)$-ème lettre du texte. Soit $u_1, \ldots, u_{2K-1}$ un ordre de Gallager pour T_n relativement à X_n. Notons que u_1 est nécessairement la racine de T_n et u_{2K-1} est la feuille vide.

c - Mise à jour

Proposition 5.5 *Si $x \in X_n$ et si tous les nœuds $u_{i_1}, u_{i_2}, \ldots, u_{i_\ell}$ du chemin entre x et la racine de T_n sont les premiers de leur poids dans l'ordre de Gallager sur T_n relativement à X_n, alors T_n est un arbre de Huffman de X_{n+1}, et $u_1, \ldots, u_{2K-1}$ est un ordre de Gallager relativement à X_{n+1}.*

preuve: Si l'on pose $T_{n+1} = T_n$, le poids des nœuds reste inchangé, sauf pour les sommets $u_{i_1}, u_{i_2}, \ldots, u_{i_\ell}$ du chemin de x à la racine. Chacun de ces nœuds étant le premier de son poids dans un ordre de Gallager sur T_n relativement à X_n, l'ordre reste décroissant après incrémentation des poids. La condition sur les frères est toujours vérifiée, car ni l'arbre ni l'ordre n'ont changé, et nous obtenons bien un ordre de Gallager sur T_{n+1} relativement à X_{n+1}. ◇

Le cas ci-dessus est idéal et n'est pas vérifié en général. Toutefois, quite à faire quelques échanges de nœuds dans l'arbre, on peut toujours s'y ramener.

Pour décrire les algorithmes nous allons munir les nœuds de trois champs :
- `pere` un nœud,
- `poids` et `ordre` des entiers.

Initialement, pour tout nœud u de T_n
- le champ `pere` contient le père de u dans T_n (**nil** s'il s'agit de la racine),
- le champ `poids` contient le poids de u relativement à (T_n, X_n),
- le champ `ordre` contient l'indice de u dans un ordre de Gallager sur T_n relativement à X_n.

procédure `mise_à_jour`(x, T)
  $u \leftarrow$ `noeud`(x, T) // *la feuille de T dont la lettre associée est x*
  tantque $u \neq$ **nil**
    $\tilde{u} \leftarrow \min(v \mid v.\texttt{poids} = u.\texttt{poids})$ // *le nœud de plus petit* `ordre` *ayant le* `poids` *de u*
    `echanger`$(u.\texttt{pere}, \tilde{u}.\texttt{pere})$
    `echanger`$(u.\texttt{ordre}, \tilde{u}.\texttt{ordre})$
    $u.\texttt{poids} \leftarrow u.\texttt{poids} + 1$
    $u \leftarrow u.\texttt{pere}$

Proposition 5.6 *Si $x \in X_n$, alors à l'issue de la procédure* `mise_à_jour`(x, T_n)*, l'arbre T_{n+1} défini par les champs* `pere` *est un code préfixe de X_{n+1}. Les* `poids` *sont relatifs à (T_{n+1}, X_{n+1}) (à un coefficient multiplicatif près), et les champs* `ordre` *induisent un ordre de Gallager sur T_{n+1} relativement à X_{n+1}.*

preuve: (ébauche) Les propriétés suivantes sont des invariants de la boucle
- deux nœuds partageant le même `pere` ont des `ordre` consécutifs,
- l'`ordre` des nœuds est celui des `poids` décroissants.

De plus, puisque nous n'avons procédé qu'à des échanges de nœuds, à la fin de l'exécution, les champs `pere` définissent un arbre binaire complet (que nous notons T_{n+1}) possédant autant de feuilles que le cardinal de X_n (égal à celui de X_{n+1}). Il s'agit donc d'un code préfixe de X_{n+1}. La valeur des `poids` est relative à (T_{n+1}, X_{n+1}), car une seule feuille a vu son poids augmenter, celle correpondant à x, les autres nœuds dont le poids à été incrémenté l'ont été conformément à la définition du poids (le chemin de x à la racine). ◇

d - Ajouter une feuille

Si $x \notin X_n$, l'arbre doit augmenter de taille et aura $2K+2$ nœuds au lieu de $2K$. La nouvelle lettre va occuper la place de la lettre vide, puis nous ajouterons une feuille qui deviendra la nouvelle feuille vide.

procédure `ajoute_feuille`(x, T)
`mise_à_jour`$(\emptyset, T)$
$z \leftarrow$ `noeud`$(\emptyset, T)$ // *la feuille vide de* T
// *z devient un nœud interne, ses deux fils u et v sont crés ci-après*
$u \leftarrow$ **new** `feuille`(x, T) // *nouvelle feuille associée à x dans T*
u.`pere` $\leftarrow z$; u.`poids` $\leftarrow 1$; u.`ordre` $\leftarrow 2K$
$v \leftarrow$ **new** `feuille`$(\emptyset, T)$ // *nouvelle feuille vide de T*
v.`pere` $\leftarrow z$; v.`poids` $\leftarrow 0$; v.`ordre` $\leftarrow 2K+1$

Proposition 5.7 *Si $x \notin X_n$, alors à l'issue de la procédure* `ajoute_feuille`(x, T_n)*, l'arbre T_{n+1} défini par les champs* `pere` *est un code préfixe de X_{n+1}. Les* `poids` *sont relatifs à (T_{n+1}, X_{n+1}) (à un coefficient multiplicatif près), et les champs* `ordre` *induisent un ordre de Gallager sur T_{n+1} relativement à X_{n+1}.*

preuve: (ébauche) Juste après l'appel `mise_à_jour`$(\emptyset, T)$, d'après les résultats précédents, les propriétés sur l'ordre et sur les frères dans l'arbre sont vérifiées. L'ajout des feuilles u et v ne change pas ces propriétés, et les poids sont bien les poids relatifs à (T_{n+1}, X_{n+1}). ◇

e - Optimalité de l'algorithme

L'algorithme de Huffman adaptatif ne pourra être optimal que si la source est sans mémoire. Dans ce cas, conformément au but visé, les propositions 5.6, 5.7 et le théorème 5.4 nous permettent de montrer que « localement », le code utilisé est un code de Huffman « classique », et donc est optimal par rapport à ce qui est connu de la source à ce moment là.

5.2 Algorithme de Lempel-Ziv

Nous allons considérer plusieurs variantes de cet algorithme. Le principe est similaire dans tous les cas : on conserve un dictionnaire de tous les mots déjà apparus dans le texte, puis à chaque fois que l'on rencontre l'un de ces mots on le remplace par son indice dans le dictionnaire. Si le texte est redondant le dictionnaire sera (relativement) petit, et l'indice d'un mot aura une taille inférieure au mot lui-même. La difficulté est algorithmique, comment mettre en œuvre ce principe à un coût raisonnable.

5.2.1 Lempel-Ziv 78

Le texte à compresser est consitué de lettres d'un alphabet A et est découpé en une concaténation de mots

$$m_0, m_1, m_2, \ldots, m_i, \ldots, m_L$$

avec les propriétés suivantes :
- les m_i, $0 \leq i < L$, sont tous distincts,
- m_0 est le mot vide,
- pour tout i, $0 < i < L$, il existe un indice $j < i$ et une lettre $a \in A$ tels que $m_i = m_j \parallel a$,
- $m_L = m_j$ avec $0 \leq j < L$.

Proposition 5.8 *Pour tout texte constitué de lettre de A, le découpage ci-dessus est unique.*

preuve: Remarquons d'abord que dans tout découpage et pour tout m_i, tous les préfixes (stricts) de m_i sont des m_j avec $j < i$.

Soient deux découpages $(m_i)_{0\leq i\leq L}$ et $(m'_i)_{0\leq i\leq L'}$ distincts d'un même texte. Soit $i > 0$, le plus petit indice tel que $m'_i \neq m_i$. Les mots m'_i et m_i doivent avoir des longueurs distinctes. Supposons, sans perte de généralité, que m_i soit le plus long. Puisqu'il s'agit de deux découpages du même texte, alors m'_i est un prefixe strict de m_i. D'après la remarque du début de la preuve il existe un indice $j < i$ tel que $m'_i = m_j = m'_j$. Donc m'_i devrait être le dernier mot de son découpage, ce qui n'est pas possible (m_i est plus long, donc le texte n'est pas terminé après m'_i). ◇

Le dictionnaire sera composé des mots m_i du découpage. Une fois ce dernier effectué, on pourra remplacer chaque m_i dans le texte par le couple (j, a) avec j un indice $< i$ et a une lettre de A.

procédure `coder_LZ78`
entrée : un flux de données F
    $D_0 \leftarrow$ **nil** *// le mot vide*
    $n \leftarrow 1$ *// taille du dictionnaire*
    repéter
        // recherche du premier mot dans le flux n'appartenant pas à D, de la forme $D_i \parallel a$
        // avec $a =$ **nil** *s'il s'agit du dernier mot du texte*
        $(i, a) \leftarrow$ `lire_mot`(F, D)
        `écrire_indice`(i, n) ; `écrire_lettre`(a)
        $D_n \leftarrow D_i \parallel a$
        $n \leftarrow n + 1$
    tantque $a \neq$ **nil**

Dans le pseudo-code ci-dessus, l'appel `écrire_indice`(i, n) va écrire sur la sortie standard l'indice i sachant que celui-ci est compris entre 0 inclus et n exclus, en pratique on devra utiliser $\log_2(n)$ (arrondi supérieurement) bits pour écrire cet indice en base 2. Inversement, le décodeur lira un indice entre 0 inclus et n exclus à l'aide de l'appel `lire_indice`(F, n).

procédure `décoder_LZ78`
entrée : un flux de données F
    $D_0 \leftarrow$ **nil** *// le mot vide*
    $n \leftarrow 1$ *// taille du dictionnaire*
    repéter
        $i \leftarrow$ `lire_indice`(F, n)
        $a \leftarrow$ `lire_lettre`(F)
        $D_n \leftarrow D_i \parallel a$
        `écrire_mot`(D_n)
        $n \leftarrow n + 1$
    tantque $a \neq$ **nil**

5.2.2 Lempel-Ziv 77

Il s'agit de la variante mise en œuvre dans beaucoup de logiciels de compression (zip, gzip, ...). Elle est apparue en 1977 avant celle présentée précédemment (qui date de 1978). Elle est plus efficace, mais plus difficile à mettre en œuvre.

On suppose que n lettres ont été lues. On lit à partir de la $n + 1$-ème lettre le plus long mot (de longueur ℓ) qui soit un sous-mot commençant à la i-ème lettre avec $i \leq n$. Ce mot est codé par la paire (i, ℓ).

En pratique, on l'implémente à l'aide d'une fenêtre glissante de taille fixe W : le dictionnaire est l'ensemble des sous-mots de cette fenêtre. Chaque mot sera ainsi codé par un couple (i, ℓ) d'entiers compris entre 0 et $W - 1$. Notons que i sera uniformement distribué dans cet intervalle alors que la valeur moyenne de ℓ est beaucoup plus faible en moyenne (logarithmique en W).

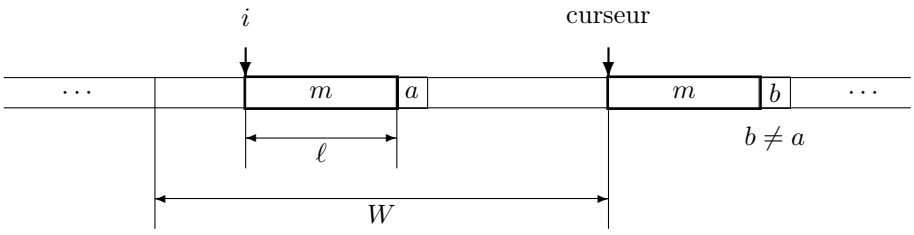


FIGURE 5.1 – Lempel-Ziv 77. Le mot m est le plus long à partir du curseur qui soit aussi un mot de la fenêtre, il est entièrement décrit par la paire (i, ℓ).

5.2.3 Optimalité de Lempel-Ziv

Les deux algorithmes exposés dans cette section sont optimaux, c'est-à-dire que pour toute source stationnaire ergodique d'entropie par lettre $\mathcal{H}$, la longueur moyenne pour coder un texte de n lettres tend vers $n\mathcal{H}$ bits lorsque n tend vers l'infini.

La preuve formelle, que nous présenterons pas ici, utilise les séquences typiques. L'intuition de la preuve est relativement simple, par exemple, pour l'algorithme de Lempel-Ziv 77, on montre, à l'aide des séquences typiques et de l'AEP, que la longueur moyenne ℓ du plus long mot reconnu est liée à la taille W de la fenêtre (lorsque celle-ci tend vers l'infini) par la relation $W = 2^{\ell\mathcal{H}}$. Ainsi chaque couple indice/longueur (i, ℓ) codera ℓ lettres de texte à l'aide d'environ $\ell\mathcal{H} + \lambda \log_2 \ell$ bits (λ une constante > 1 dépendant de la façon dont la longueur du mot est codée). Asymptotiquement nous arrivons à la valeur optimale de $\mathcal{H}$ bits par lettre.

Chapitre 6

Canaux discrets sans mémoire

Dans le chapitre précédent sur le codage de source, aucun élément extérieur ne venait modifier l'information. Au contraire lorsque l'information va transité dans le canal, elle sera perturbée par un bruit.

Le résultat principal de ce chapitre nous assure qu'il est possible de coder l'information de façon à ce que la détérioration soit négligeable. Cela se fera bien entendu au prix d'une redondance de l'information.

En fait pour tout canal de transmission, nous définirons une grandeur caractéristique appelée *capacité* et qui s'interprète comme la quantité maximale d'information pouvant transiter à travers le canal.

6.1 Généralités

6.1.1 Définition du canal discret

Pour définir un canal de transmission, il est nécessaire de décrire l'ensemble des entrées et des sorties possibles du canal, ainsi que le bruit qui perturbera la transmission.

Le canal discret est le modèle le plus simple de canal de transmission. L'ensemble des entrées comme celui des sorties seront des ensembles finis X et Y et le bruit se modélisera par la donnée d'une loi de probabilité conditionnelle de Y sachant X.

Définition 6.1 (Canal discret sans mémoire) *Un* canal discret sans mémoire *est défini par la donnée de :*

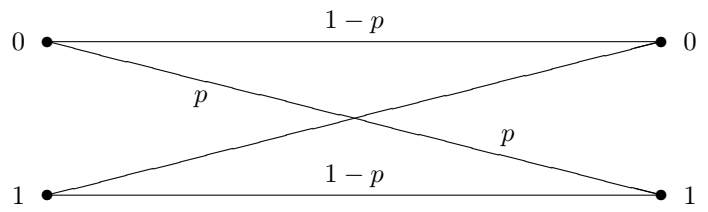
1. *Un alphabet d'entrée* $X = \{a_1, \dots, a_K\}$.
2. *Un alphabet de sortie* $Y = \{b_1, \dots, b_J\}$.
3. *Une loi de transition définies par les probabilité conditionnelles* $P(b_j \mid a_k)$.

La matrice $K \times J$ *suivante*

$$\Pi = \begin{pmatrix} P(b_1 \mid a_1) & \dots & P(b_J \mid a_1) \\ \vdots & \ddots & \vdots \\ P(b_1 \mid a_K) & \dots & P(b_J \mid a_K) \end{pmatrix}$$

est appelée matrice stochastique *du canal. Nous parlerons d'un canal* (X, Y, Π).

Exemple: Le canal binaire symétrique de probabilité de transition p représenté par le diagramme



a pour matrice stochastique

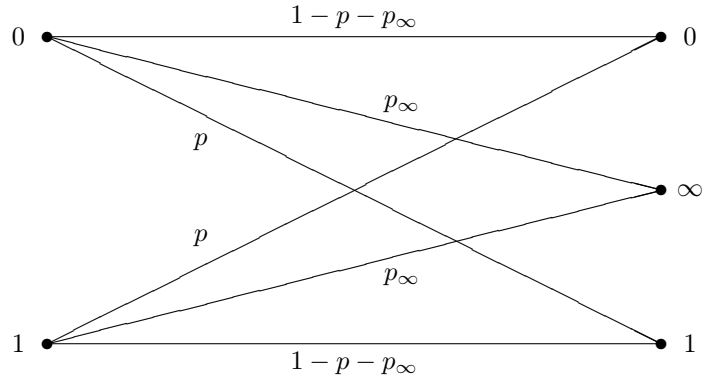
$$\begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}.$$

La probabilité de transition p du canal sera souvent appelée probabilité d'erreur du canal.

Définition 6.2 (Canal symétrique) *Un canal discret est dit* symétrique *si les lignes de sa matrice stochastique sont formées des mêmes éléments à l'ordre près.*

Un canal symétrique n'a pas forcément le même nombre d'entrées et de sorties.

Exemple: Le canal binaire symétrique à effacement est défini par le diagramme



et a pour matrice stochastique

$$\begin{pmatrix} 1-p-p_\infty & p_\infty & p \\ p & p_\infty & 1-p-p_\infty \end{pmatrix}.$$

6.1.2 Canal continu et canal discret

Nous avons vu dans le chapitre d'introduction que nous pouvions ramener l'étude des canaux continus à celle des canaux discrets.

En effet l'ensemble constitué du modulateur de données digitales, du canal continu et du démodulateur de données digitales dans la Figure 6.1 peut être considéré comme un canal discret.

La principale difficulté dans un tel modèle consiste à « traduire » le bruit du canal continu en un bruit pour le canal discret correspondant. Cette traduction dépendra évidemment du type de modulation choisi.

6.2 Capacité d'un canal

6.2.1 Capacité d'un canal

Soit un canal dont l'ensemble des entrées est X et l'ensemble des sorties Y. La donnée du canal nous fourni également la loi de transition, c'est à dire la loi de probabilité conditionnelle de Y sachant X. Cette loi sera notée P, pour tout x dans X et tout y dans Y, nous connaissons $P(y \mid x)$.

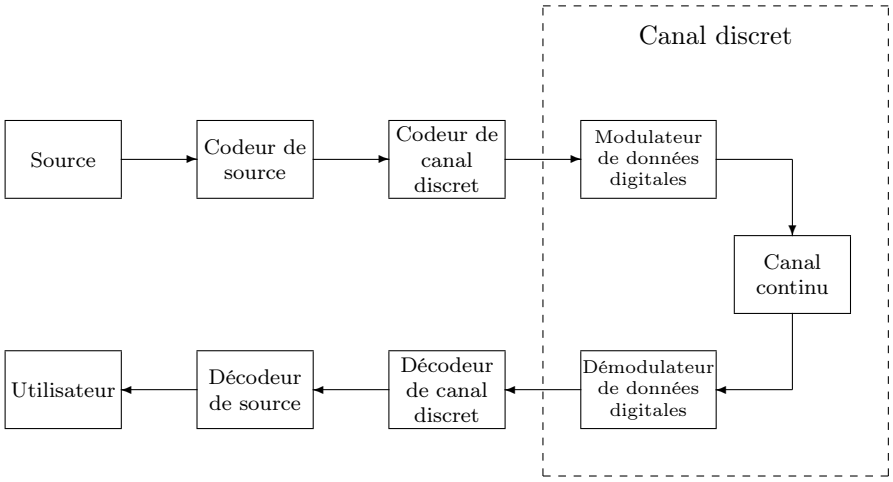


FIGURE 6.1 – Canal continu et canal discret.

Si nous connaissons également la loi d'émission, c'est-à-dire la loi de probabilité de X, alors nous sommes en mesure de calculer l'information mutuelle $I(X;Y)$ entre X et Y. En effet

$$I(X;Y) = \sum_x \sum_y P(y \mid x) P(x) \log_2 \frac{P(y \mid x)}{P(y)}$$

et $P(x,y)$ et $P(y)$ peuvent être calculés et valent

$$P(x,y) = P(y \mid x)P(x), \quad P(y) = \sum_x P(y \mid x)P(x).$$

L'information mutuelle entre X et Y s'interprète comme l'information transmise à travers le canal. La relation

$$I(X;Y) = H(X) - H(X \mid Y)$$

exprime alors que l'information transmise à travers le canal est égale à l'entropie de X, $H(X)$ diminuée de la quantité $H(X \mid Y)$ que l'on peut interpréter alors comme la perte d'information due au bruit.

Pour un canal donné, seule la loi de transition est connue. La loi d'émission ne dépend que de la source et du codeur de canal. La capacité du canal sera la plus grande valeur que peut atteindre l'information mutuelle de X et Y.

Définition 6.3 (Capacité d'un canal) *Soit un canal d'entrée X et de sortie Y. La* capacité *de ce canal est égal au maximum de l'information mutuelle entre X et Y pour toutes les lois d'émission possible.*

Cette définition ne nécessite pas un canal discret, et reste parfaitement applicable à un canal continu. Nous ne nous intéresserons cependant qu'à la capacité des canaux discret sans mémoire.

6.2.2 Capacité d'un canal discret sans mémoire

Si la définition de la capacité d'un canal peut sembler simple, son calcul pratique est d'une grande complexité en général. En effet si l'alphabet d'entrée est $X = \{a_1, \ldots, a_K\}$, et que la loi d'émission est $p_1, \ldots, p_K$, où $p_k = P(a_k)$ pour tout k, il faut calculer le maximum de $I(X;Y)$ sur l'ensemble des K-uplets $(p_1, \ldots, p_K)$ avec les contraintes $p_k \geq 0$ pour tout k, et $\sum p_k = 1$.

Ce calcul est cependant possible dans certains cas particuliers.

Capacité d'un canal symétrique.

Définition 6.4 (Canal fortement symétrique) *Un canal discret est dit* fortement symétrique *si les lignes et les colonnes de sa matrice stochastique sont égales à une permutation près.*

Définition 6.5 (Décomposition d'un canal) *Nous dirons que le canal $\mathcal{T} = (X, Y, \Pi)$ se décompose en une combinaison des canaux $\mathcal{T}_i = (X, Y_i, \Pi_i)_{1 \leq i \leq L}$, si les Y_i sont disjoints et s'il existe des nombres réels positifs $q_1 + q_2 + \ldots + q_L = 1$ tels que $\Pi = \left(\begin{array}{c|c|c} q_1\Pi_1 & \cdots & q_L\Pi_L \end{array} \right)$. Nous noterons formellement*

$$\mathcal{T} = \sum_{i=1}^{L} q_i \mathcal{T}_i$$

Proposition 6.6 *Un canal discret est symétrique si et seulement si il se décompose en une combinaison de canaux fortement symétriques.*

La preuve est laissée en exercice. L'énoncé ci-dessus est parfois utilisé comme définition.

Pour un canal symétrique, la matrice stochastique

$$\Pi = \begin{pmatrix} P(b_1 \mid a_1) & \ldots & P(b_J \mid a_1) \\ \vdots & \ddots & \vdots \\ P(b_1 \mid a_K) & \ldots & P(b_J \mid a_K) \end{pmatrix}$$

est telle que ses lignes sont toutes égales à l'ordre près. On pose pour tout couple (k, j), $\Pi_{k,j} = P(b_j \mid a_k)$.

L'information mutuelle moyenne s'écrit

$$I(X;Y) = H(Y) - H(Y \mid X),$$

où $H(Y \mid X)$ l'entropie conditionnelle de Y sachant X est égale à

$$\begin{aligned} H(Y \mid X) &= -\sum_k \sum_j P(a_k, b_j) \log_2 P(b_j \mid a_k) \\ &= -\sum_k \sum_j \Pi_{k,j} p_k \log_2 \Pi_{k,j} \\ &= -\sum_k p_k \sum_j \Pi_{k,j} \log_2 \Pi_{k,j}. \end{aligned}$$

Par définition d'un canal symétrique, le terme $-\sum_j \Pi_{k,j} \log_2 \Pi_{k,j}$ est indépendant de k. Notons le $H(\Pi)$. Nous avons alors

$$H(Y \mid X) = \sum_k p_k H(\Pi) = H(\Pi) \sum_k p_k = H(\Pi).$$

Donc pour maximiser $I(X;Y) = H(Y) - H(\Pi)$ il suffira de maximiser $H(Y)$.

Proposition 6.7 (Capacité d'un canal fortement symétrique) *La capacité d'un canal fortement symétrique (X, Y, Π) (avec $|Y| = J$) est égale à*

$$C = \log_2 J - H(\Pi).$$

preuve: Puisque $H(Y)$ est borné supérieurement par $\log_2 J$, la capacité d'un canal symétrique est bornée supérieurement par $\log_2 J - H(\Pi)$, avec égalité si et seulement si la loi de Y est uniforme.

Dans un canal fortement symétrique, on vérifie aisément que si la loi de X est uniforme, alors celle de Y l'est également. Donc la borne supérieure est atteinte. ◇

Cette borne supérieure ne sera pas toujours atteinte dans un canal symétrique. Toutefois, la capacité du canal est atteinte lorsque la loi d'émission est uniforme. En pratique le calcul de capacité d'un canal symétrique se fait à l'aide de la proposition suivante :

Proposition 6.8 *Soit $\mathcal{T}$ un canal symétrique dont la décomposition en canaux fortement symétrique s'écrit $q_1\mathcal{T}_1 + \ldots + q_L\mathcal{T}_L$. Sa capacité est atteinte lorsque la loi d'émission est uniforme et vaut*

$$C = \sum_{i=1}^{L} q_i C_i$$

où les C_i sont les capacités des canaux fortement symétriques $\mathcal{T}_i$.

Exemple: Considérons le canal binaire symétrique. Sa matrice stochastique vaut

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

et donc $H(\Pi) = -p\log_2 p - (1-p)\log_2(1-p)$.
La capacité du canal binaire symétrique de probabilité d'erreur p est donc égale à

$$C = 1 + p\log_2 p + (1-p)\log_2(1-p).$$

6.3 Théorème fondamental

Avant d'énoncer le second théorème de Shannon sur le codage des canaux bruité, nous allons décrire les façons par lesquelles nous pouvons opérer ce codage.

6.3.1 Codage de canal

D'après le hypothèses que nous avons admises jusqu'à présent, l'entrée du codeur de canal est égale à la sortie du codeur de source, c'est-à-dire qu'il s'agit d'une séquence binaire.

Définition 6.9 *Un* codage *d'un canal discret (X, Y, Π) est une procédure qui associe à chaque séquence binaire finie une séquence finie de lettres de X.*

Bien évidemment, il est hautement désirable qu'après passage dans le canal la séquence de lettres de Y reçue puisse être décodée pour retrouver la séquence binaire, et ce avec une probabilité la plus élevée possible.

Définition 6.10 *Un* code en bloc *d'un canal discret (X, Y, Π) est une procédure qui associe à chaque séquence binaire de longueur donnée l une séquence finie de n lettres de X appelée mot de code. L'entier n est appelé* longueur *du code, 2^l est le cardinal du code.*

Nous désignerons également par code l'ensemble des mots de codes pour tous les l-uplets binaires possibles.

À tout codage, il faudra évidemment associer une procédure de décodage. Cette procédure va pendre en entrée une séquence de lettres à la sortie du canal, et donner en sortie une séquence binaire. Ce décodage devra se faire de façon à maximiser la vraisemblance.

Définition 6.11 *Un* algorithme de décodage *d'un codage d'un canal (X, Y, Π) sera une procédure qui à toute séquence de lettres de Y associe une séquence binaire.*

Un algorithme de décodage *d'un code en bloc d'un canal (X, Y, Π) de longueur n et de cardinal 2^l sera une procédure qui à tout n-uplet de Y associe un l-uplet binaire.*

6.3.2 Canal binaire symétrique

Nous nous contenterons d'exposer le théorème fondamental de Shannon pour le codage de canal dans le cas d'un canal binaire symétrique. Il est aisé à partir de généraliser au cas q-aire, c'est-à-dire lorsque les alphabet d'entrée et de sortie du canal sont égaux et de cardinal q.

Comme nous l'avons déjà vu le canal binaire symétrique de probabilité de transition p à pour capacité $C = 1 + p\log_2 p + (1-p)\log_2(1-p)$.

Pour un tel canal, nous utiliserons des codes binaires, c'est-à-dire des sous ensemble de $\{0,1\}^n$ pour un certain entier n.

Définition 6.12 *Le taux d'un code binaire de longueur n et de cardinal M est égal à*

$$R = \frac{\log_2 M}{n}.$$

Un code aura un taux de 1 lorsque $M = 2^n$, c'est-à-dire qu'aucune redondance n'aura été ajoutée. Un code de taux ne 1/2 signifiera que $M = 2^{n/2}$, c'est-à-dire, si n est pair que le code double la longueur des séquences binaires.

Soit $\mathcal{C}$ un code en bloc de longueur n et de cardinal M. Soient $\vec{x}$ un mot code et $\vec{y}$ un n-uplet de lettres de Y.

Soit ϕ un algorithme de décodage nous noterons $\Pr(\phi(\vec{y}) = \vec{x})$ la probabilité pour que le mot $\vec{y}$ de Y^n soit décodé par le mot de code $\vec{x}$. Nous noterons $\Pr(\phi(\vec{y}) \neq \vec{x}) = 1 - \Pr(\phi(\vec{y}) = \vec{x})$ la probabilité de l'événement complémentaire.

La grandeur $\Pr(\phi(\vec{y}) \neq \vec{x})$ peut être considérée comme une v.a. réelle, donc il est possible de calculer sa moyenne.

Définition 6.13 (taux d'erreur résiduel) *Le* d'erreur résiduel *d'un algorithme de décodage est égale à la moyenne de la variable aléatoire* $\Pr(\phi(\vec{y}) \neq \vec{x})$ *lorsque la loi d'émission est uniforme.*

Si nous notons ce taux d'erreur résiduel P_e, nous avons

$$\begin{aligned} P_e &= \sum_{\vec{x},\vec{y}} P(\vec{x},\vec{y})\Pr(\phi(\vec{y}) \neq \vec{x}), \\ &= \sum_{\vec{x},\vec{y}} P(\vec{y} \mid \vec{x})P(\vec{x})\Pr(\phi(\vec{y}) \neq \vec{x}), \\ &= \sum_{\vec{x},\vec{y}} \frac{P(\vec{y} \mid \vec{x})}{M}\Pr(\phi(\vec{y}) \neq \vec{x}). \end{aligned}$$

La sommation s'effectue pour tout $\vec{x}$ dans $\mathcal{C}$ et tout $\vec{y}$ dans Y^n.

La taux d'erreur P_e est bien un taux d'échec moyen dans le cas ou la loi d'émission est uniforme. Ce n'est pas le cas en général.

Exemple: Considérons le code à répétition de longueur impaire $n = 2t + 1$ utilisé pour lutter contre le bruit dans un canal binaire symétrique sans mémoire de probabilité d'erreur p.

Un code à répétition est un code constitué de deux mots : $0 \ldots 0$ et $1 \ldots 1$ de longueur n qui codent respectivement les symboles '0' et '1'.

Un algorithme de décodage possible pour ce code est le décodage dit majoritaire : une séquence de n symboles binaires sera décodée par '0' si elle comporte une majorité de '0' dans son écriture, et par '1' sinon.

La probabilité pour que le mot $0 \ldots 0$ transmis sur le canal soit décodé par le symbole 1 est égale à

$$P_e = \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i},$$

c'est-à-dire la probabilité pour que le nombre de symboles ayant été transformé soit supérieur ou égal à t+1.

Il est possible de montrer que pour une probabilité d'erreur $p < 1/2$ fixée, la grandeur P_e peut être rendu arbitrairement petite.

6.3.3 Le second théorème de Shannon pour un canal binaire symétrique

Théorème 6.14 (Second théorème de Shannon) *Soit un canal binaire symétrique de capacité C. Pour tout $R < C$ il existe une suite $\mathcal{C}_n$ de codes de longueur n de taux R_n d'algorithme de décodage de taux d'erreur résiduel P_{en} telle que*

1. $\lim_{n\to\infty} R_n = R$
2. $\lim_{n\to\infty} P_{en} = 0$

Ce théorème signifie donc qu'il existe des codes en bloc permettant de réaliser un code dont le taux est aussi proche qu'on le désire de la capacité du canal.

Il existe un autre résultat qui est la réciproque de ce théorème.

Théorème 6.15 *Soit un canal binaire symétrique de capacité C. Soit un code de taux $R > C$ alors tout algorithme de décodage de ce code est tel que son taux d'erreur résiduel $P_e > K(R, C) > 0$, où $K(R, C)$ est une constante strictement positive ne dépendant que de R et de C.*

Ce résultat nous indique qu'il est inutile de chercher des codes de taux supérieur à la capacité du canal. La capacité C est donc bien le taux de codage maximal que l'on puisse atteindre pour faire transiter une information dans un canal donné.

Chapitre 7

Codes correcteurs d'erreurs

7.1 Définitions générales

7.1.1 Distance de Hamming

Définition 7.1 *Soient A^n l'ensemble des mots de longueur n sur A, $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1}) \in A^n$ et $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in A^n$.*

– *La* distance de Hamming *entre* $\mathbf{x}$ *et* $\mathbf{y}$ *est*

$$d_H(\mathbf{x}, \mathbf{y}) = |\{i \mid 0 \leq i \leq n-1,\ x_i \neq y_i\}|.$$

on vérifie que d_H est bien une métrique et on appelle alors espace de Hamming sur A *l'ensemble A^n muni de la métrique d_H.*

– *Si A est un groupe, le* poids de Hamming $w_H(\mathbf{x})$ *d'un mot $\mathbf{x} \in A^n$ est le nombre de ses coordonnées non nulles*

$$w_H(\mathbf{x}) = d_H(\mathbf{x}, \mathbf{0}),$$

où $\mathbf{0}$ est le mot de A^n ayant toutes ses coordonnées égales à l'élément neutre de A.

7.1.2 Codes en bloc – Codes linéaires

Définition 7.2 *Un* code C *sur A est une partie non vide de l'espace de Hamming A^n, n est appelé* longueur *du code, les éléments de C sont appelés* mots de code.

Lorsque $A = \{0, 1\}$ on parlera de code binaire.

Définition 7.3 *On appelle* distance minimale *d'un code C sur A, l'entier*

$$d = min\{d_H(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\},$$

si A est un groupe, on appelle poids minimal *d'un code C, l'entier*

$$min\{w_H(\mathbf{x}) \mid \mathbf{x} \in C, \mathbf{x} \neq \mathbf{0}\}.$$

Lorsque l'alphabet A peut être muni d'une structure de corps, l'espace A^n est un espace vectoriel. On a alors la définition suivante.

Définition 7.4 *Un code C est dit* linéaire *sur A, si A est un corps et C un sous-espace vectoriel de A^n. La dimension de C sur A est appelée la* dimension *du code C.*

Pour un code linéaire le *poids minimal* est égal à la *distance minimale.*

Les paramètres d'un code sont :

– son alphabet A de cardinal q,

– sa longueur n,
– son cardinal M (ou sa dimension k s'il est linéaire),
– sa distance minimale d.

Nous noterons $C(q; n, k, d)$ un code linéaire et $C[A; n, M, d]$ un code quelconque.

Définition 7.5 *Une* matrice génératrice G *d'un code linéaire C est une matrice $k \times n$ dont les lignes forment une base de C.*

Pour un même code il existe de nombreuses matrice génératrices. Parmi celles-ci certaines ont une forme pratique.

Proposition 7.6 *Tout code linéaire possède une matrice génératrice systématique, c'est-à-dire de la forme*

$$G = \left(\begin{array}{c|c} I_k & P \end{array} \right)$$

à une permutation des coordonnées près.

Définition 7.7 *Soit C un code linéaire de matrice génératrice G. Une* matrice de parité H *de C est une matrice $(n-k) \times n$ telle que $H\,{}^tG = 0$, où ${}^tG$ est la transposée de G.*

La matrice de parité H est telle que pour tout x dans C, $H\,{}^tx = 0$. C'est-à-dire que C est le noyau de l'endomorphisme de $\mathbf{F}_q^n$ dans $\mathbf{F}_q^{n-k}$ représenté par la matrice H.

Proposition 7.8 *Si G est une matrice génératrice systématique de C*

$$G = \left(\begin{array}{c|c} I_k & P \end{array} \right)$$

alors

$$H = \left(\begin{array}{c|c} -P & I_{n-k} \end{array} \right)$$

est une matrice de parité de C.

7.1.3 Décodage à vraisemblance maximale

Soit un canal de transmission (A, B, Π).

Définition 7.9 *Soit C un code de longueur n sur A. Un algorithme de décodage de C à* vraisemblance maximale *est une procédure de décodage $B^n \to C \subset A^n$ qui à tout élément de $\vec{y} \in B^n$ associe l'élément de $\vec{x} \in C$ réalisant le maximum de $P(\vec{y} \mid \vec{x})$.*

Proposition 7.10 *Dans un canal binaire symétrique de probabilité de transition $p < 1/2$, $P(\vec{y} \mid \vec{x})$ est une fonction croissante de $d_H(\vec{y}, \vec{x})$.*

preuve: Soient $\vec{x} = (x_1, \ldots, x_n)$ et $\vec{y} = (y_1, \ldots, y_n)$, on a

$$P(\vec{y} \mid \vec{x}) = \prod_{i=1}^{n} P(y_i \mid x_i) = p^{d_H(\vec{y},\vec{x})}(1-p)^{n-d_H(\vec{y},\vec{x})} = (1-p)^n \left(\frac{p}{1-p} \right)^{d_H(\vec{y},\vec{x})}.$$

◇

Proposition 7.11 *Soit ϕ un algorithme de décodage à vraisemblance maximale de C dans un canal binaire symétrique. Si $\phi(\vec{y}) = \vec{x}$ alors pour tout $\vec{x}'$ dans C on a $d_H(\vec{y}, \vec{x}) \leq d_H(\vec{y}, \vec{x}')$.*

Proposition 7.12 *Un code de distance minimale d peut corriger $(d-1)/2$ erreurs.*

7.2 Quelques classes de codes

7.2.1 Codes parfaits

Définition 7.13 *On appelle* code parfait *un code tel que l'ensemble des boules de rayon* $\lfloor \frac{d-1}{2} \rfloor$ *centrées en tous les éléments du code forment une partition de l'espace de Hamming* A^n.

Tous les codes parfaits sont connus.

Exemple:

1. Code à répétition de longueur impaire $n = 2t+1$. Ce code contient deux mots : $0\ldots0$ et $1\ldots1$. Sa matrice génératrice est

$$G = \begin{pmatrix} 1 & \ldots & 1 \end{pmatrix}$$

et sa matrice de parité est

$$H = \begin{pmatrix} 1 & & 0 & 1 \\ & \ddots & & \vdots \\ 0 & & 1 & 1 \end{pmatrix}$$

Ce code a pour distance minimale $d = n = 2t+1$, et tout mot de l'espace $\{0,1\}^n$ contient soit plus de $t+1$ '0', soit plus de $t+1$ '1', et se trouve donc à distance au plus $t = (d-1)/2$ d'un des deux mots de code.

2. Code de Hamming $\mathcal{H}_m$ est un code binaire linéaire de paramètres $(n = 2^m - 1, k = 2^m - m - 1, d = 3)$. La matrice de parité H_m de ce code est constituée de l'ensemble des $2^m - 1$ vecteurs colonne non nuls de $\mathbf{F}_2^m$. Par exemple :

$$H_3 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Le nombre de points d'une boule de rayon $(d-1)/2 = 1$ centrée en un mot du code $\mathcal{H}_m$ est $\binom{n}{0} + \binom{n}{1} = 2^m$. La réunion de toutes ces boules a donc un cardinal $2^k 2^m = 2^{k+m} = 2^n$, c'est-à-dire le cardinal de l'espace tout entier. Le code $\mathcal{H}_m$ est donc parfait.

7.2.2 Codes cycliques

a - Corps finis

Théorème 7.14 *Soit* F *un corps fini de cardinal* $q > 1$. *Alors*

1. $q = p^m$, *où* p *est un nombre premier et* m *un entier positif.*
2. F *est unique à isomorphisme près.*

- Lorsque p est premier le corps $\mathbf{F}_p$ est égal à $\mathbf{Z}/p\mathbf{Z}$.
- Lorsque $q = p^m$, avec $m > 1$, on dira que $\mathbf{F}_q$ est une extension de degré m de $\mathbf{F}_p$.

Proposition 7.15 *Soit* $\mathbf{F}_q$ *un corps fini. Soit* $p_m(x)$ *un polynôme irréductible de* $\mathbf{F}_q[x]$ *de degré* m. *Soit* $(p_m(x))$ *l'idéal engendré par le polynôme* $p_m(x)$, *c'est à dire l'ensemble des polynômes multiple de* $p_m(x)$.

1. *Le quotient* $\mathbf{F}_q[x]/(p_m(x))$ *est un corps fini de cardinal* q^m.
2. *Il existe toujours un élément* α *de* $\mathbf{F}_{q^m}$ *tel que* $p_m(\alpha) = 0$.

Définition 7.16 *Un polynôme* $p_m(x)$ *irréductible de degré* m *est* primitif *si l'ensemble des restes de la division de* x^i *par* $p_m(x)$ *sont tous distincts pour* $0 \leq i < q^m - 1$.
Un élement α *de* $\mathbf{F}_{q^m}$ *tel que* $p_m(\alpha) = 0$ *est dit primitif.*

Soit α un élement primitif de $\mathbf{F}_{q^m}$, on notera $\mathbf{F}_{q^m} = \mathbf{F}_q[\alpha]/(p_m(\alpha))$.
- Le corps fini à q^m élements est égal à l'ensemble $\{0, 1, \alpha, \alpha^2, \ldots, \alpha^{q^m-2}\}$.
- Le corps fini à q^m éléments est égal à l'ensemble des polynômes en α de degré $< m$.
- L'addition deux élements du corps sera l'addition de deux polynômes de $\mathbf{F}_q[\alpha]$.
- La multiplication de deux élements du corps sera le reste de la division par $p_m(\alpha)$ de la multiplication de deux polynômes de $\mathbf{F}_q[\alpha]$.

b - Codes BCH

L'alphabet est égal à $\mathbf{F}_q$, le corps à q éléments. Soit m un entier et $n = q^m - 1$. On note $\mathcal{R}_n = \mathbf{F}_q[X]/(X^n - 1)$ l'anneau des polynômes à coefficients dans $\mathbf{F}_q$ modulo l'idéal engendré par $X^n - 1$. Soit α un élément primitif de $\mathbf{F}_{q^m}$.

Définition 7.17 (codes cycliques) *Un* code cyclique *binaire de longueur n est un idéal de $\mathcal{R}_n$.*

Remarque :

1. Un code cyclique est linéaire.
2. Un code cyclique est invariant par permutation circulaire de ses coordonnées dans la base $(1, X, X^2, \ldots, X^{n-1})$de $\mathcal{R}_n$. La permutation circulaire d'une position à droite correspond à la multiplication par X modulo $X^n - 1$

$$\begin{aligned} & X\,(a_0 + a_1X + \ldots + a_{n-2}X^{n-2} + a_{n-1}X^{n-1}) \\ &= a_0X + a_1X^2 + \ldots + a_{n-2}X^{n-1} + a_{n-1}X^n \\ &= a_{n-1} + a_0X + a_1X^2 + \ldots + a_{n-2}X^{n-1} \mod X^n - 1. \end{aligned}$$

Théorème 7.18 *Soit C un code cyclique binaire de longueur n, on a*

i. *Il existe un unique polynôme unitaire $g(X)$ de degré minimal dans C.*

ii. *C est l'idéal engendré par $g(X)$, $g(X)$ est appelé* polynôme générateur *de C.*

iii. *$g(X)$ est un facteur de $X^n - 1$.*

iv. *Tout $c(X) \in C$ s'écrit de façon unique $c(X) = f(X)g(X)$ dans $\mathbf{F}_q[X]$. La dimension de C est $n - r$ où $r = \deg g(X)$.*

Définition 7.19 (codes BCH) *Soit C un code cyclique binaire de longueur $n = q^m - 1$ et soit $g(X)$ son polynôme générateur. C est un code BCH de* distance construite *δ si δ est le plus grand entier vérifiant*

$$\exists\, b \in \mathbf{Z},\ g(\alpha^b) = g(\alpha^{b+1}) = \ldots = g(\alpha^{b+\delta-2}) = 0,$$

où α est un élement primitif de $\mathbf{F}_{q^m}$.

Définition 7.20 *Un code est dit BCH* au sens strict *si $b = 1$. On notera $B(n, \delta)$ le code BCH binaire au sens strict de longueur n et de distance construite δ.*

Théorème 7.21 (borne BCH) *Soit un code BCH de distance construite δ, sa distance minimale d vérifie*

$$d \geq \delta.$$

Exemple: Le code de Hamming est un code BCH binaire de distance construite 3.

c - Code de Reed-Solomon

Définition 7.22 *Un code de Reed-Solomon est un code BCH de longueur $q - 1$ sur $\mathbf{F}_q$.*

7.2.3 Codes détecteurs d'erreur

Un code peut être utilisé pour detecter les erreurs au lieu de les corriger. Le problème que l'on souhaite résoudre est le suivant : un mot de code est transmis à travers un canal bruité et l'utilisateur souhaite savoir si ce mot a été modifié.

Pour qu'une erreur non detectée ait lieu, il faut qu'un mot de code soit modifié en un mot de code différent.

Proposition 7.23 *Un code de distance minimale d peut detecter $d-1$ erreurs.*

Exemple:

1. Le code de parité de longueur n est l'ensemble des séquences binaires de poids pair. Sa matrice génératrice est égale à

$$G = \begin{pmatrix} 1 & & 0 & 1 \\ & \ddots & & \vdots \\ 0 & & 1 & 1 \end{pmatrix}$$

   Le code est obtenu en ajoutant à une séquence de longueur $n-1$ un n-ième symbole égal à la somme des autres. Par exemple les 128 caractéres ASCII sont codés par des mots de longueur 8.

   Ce code permet de détecter une erreur.

2. code CRC (Cyclic Redundancy Check).

   Une séquence de k symboles binaires d'information est représentée par le polynôme $i(x)$. Soit $g(x)$ un polynôme de degré s. Le mot de code correspondant à $i(x)$ est égal à $c(x) = x^s i(x) + r(x)$ où $r(x)$ est le reste de la division de $x^s i(x)$ par $g(x)$.

   Ce code permet de detecter toute rafale (erreurs consécutives) d'erreurs de longueur inférieure ou égale à s.

7.3 Décodage des codes linéaires

7.3.1 Théorie algébrique du décodage

Dans toute cette section on considère un code linéaire $C(n,k,d)$ sur $\mathbf{F}_2$.

a - Position du problème

Etant donné
– que $\mathbf{x} \in C$ est le « message transmis »,
– que $\mathbf{x}$ est perturbé dans un canal bruité par l'erreur $\mathbf{e} \in \mathbf{F}_2^n$,
– que $\mathbf{y} = \mathbf{x} + \mathbf{e}$, le « message reçu », est le seul mot auquel le décodeur a accès,
le problème du décodage est de retrouver $\mathbf{x}$ à partir de $\mathbf{y}$.

Un algorithme de décodage de C devra donc prendre comme argument un élément de $\mathbf{F}_2^n$, et s'il se termine, rendre un élément du code C. Il devra également être déterministe, c'est-à-dire qu'un mot de l'espace $\mathbf{F}_2^n$ sera toujours décodé de la même manière. Nous proposons la définition suivante.

Définition 7.24 *Soit un code linéaire $C(n,k,d)$ sur $\mathbf{F}_2$, un* algorithme de décodage d'erreur de C *est une application*

$$\begin{array}{rccc} \gamma & : & \mathbf{F}_2^n & \longrightarrow & C \cup \{\infty\} \\ & & \mathbf{y} & \longmapsto & \gamma(\mathbf{y}) \end{array}$$

telle que $\forall\, \mathbf{x} \in C$, $\gamma(\mathbf{x}) = \mathbf{x}$. Le fait que $\gamma(\mathbf{y}) = \infty$ signifiant que $\mathbf{y}$ n'a pas été décodé.

Soit e un entier positif, on dira que γ est borné par *e, si*

$$\forall\, \mathbf{y} \in \mathbf{F}_q^n,\ \forall\, \mathbf{x} \in C,\quad d_H(\mathbf{x},\mathbf{y}) < \frac{e}{2} \Rightarrow \gamma(\mathbf{y}) = \mathbf{x}$$

où d_H est la métrique de Hamming sur $\mathbf{F}_q$, on dira que γ est borné strictement par *e si on a de plus*

$$\gamma(\mathbf{y}) = \mathbf{x} \neq \infty \Rightarrow d_H(\mathbf{x},\mathbf{y}) < \frac{e}{2}$$

b - Matrice de parité – Syndrome

Définition 7.25 *On appelle* matrice de parité *du code $C(n,k,d)$ une matrice H à $n-k$ lignes et n colonnes sur $\mathbf{F}_2$, telle que $C = \ker H = \{\mathbf{x} \mid H\mathbf{x}^t = 0\}$.*

La matrice H détermine le code C. Remarquons que par un théorème fondamental d'algèbre linéaire, les lignes de H forment une base de l'orthogonal de C dans $\mathbf{F}_2^n$ pour le produit scalaire usuel.

Définition 7.26 *Soit $\mathbf{y} \in \mathbf{F}_2^n$. Le* syndrome *de $\mathbf{y}$ est le vecteur de $\mathbf{F}_2^{n-k}$*

$$S(\mathbf{y}) = H\mathbf{y}^t.$$

l'application $S : \mathbf{F}_2^n \to \mathbf{F}_2^{n-k}$ ainsi définie est $\mathbf{F}_2$-linéaire.

H induit une relation d'équivalence sur $\mathbf{F}_2^n$

$$\mathbf{x}\ \mathcal{R}\ \mathbf{y} \Leftrightarrow H\mathbf{x}^t = H\mathbf{y}^t \Leftrightarrow H(\mathbf{x}-\mathbf{y})^t = 0 \Leftrightarrow \mathbf{x}-\mathbf{y} \in C.$$

Chacune des classes de cette relation d'équivalence est appelée « classe latérale » ou « translaté » de C.

Le code C est la classe de l'élément nul de $\mathbf{F}_2^n$. Plus généralement, deux éléments sont dans un même translaté si et seulement si ils ont le même syndrome.

Proposition 7.27 *Il existe au plus un mot de poids $< d/2$ dans un translaté donné.*

preuve: Soient $\mathbf{x},\mathbf{y} \in \mathbf{F}_2^n$ tels que $\mathbf{x}\ \mathcal{R}\ \mathbf{y}$, $w_H(\mathbf{x}) < d/2$ et $w_H(\mathbf{y}) < d/2$. Alors $\mathbf{x}-\mathbf{y} \in C$ et $w_H(\mathbf{x}-\mathbf{y}) = d(\mathbf{x},\mathbf{y}) \leq d(\mathbf{x},\mathbf{0}) + d(\mathbf{0},\mathbf{y}) < d/2 + d/2 = d$, en vertu de l'inégalité triangulaire. Donc $\mathbf{x} = \mathbf{y}$. ◇

c - Décodage des codes linéaires

A partir de la matrice de parité et des syndromes on peut définir l'algorithme de décodage γ suivant :

1. Dans tout translaté $\mathbf{y}+C$, on choisit un élément de plus petit poids, $e(\mathbf{y})$ que l'on met dans une table indexée par le syndrome de $\mathbf{y}$. Remarquons que $e(\mathbf{y}+\mathbf{x}) = e(\mathbf{y})$ pour tout $\mathbf{x} \in C$
2. Si $\mathbf{y}$ est le mot reçu, on calcule son syndrome $S(\mathbf{y})$, et on lit $e(\mathbf{y})$ dans la table.
3. l'algorithme retourne $\gamma(\mathbf{y}) = \tilde{\mathbf{x}} = \mathbf{y} - e(\mathbf{y})$.

Proposition 7.28 *γ est un algorithme à vraisemblance maximale, donc il est borné par sa distance minimale.*

preuve: Soit $\mathbf{y} \in \mathbf{F}_q^n$, $\tilde{\mathbf{e}}$ un élément de plus petit poids de la classe de $\mathbf{y}$
- $\gamma(\mathbf{y}) \in C$, en effet $\mathbf{y}\ \mathcal{R}\ \tilde{\mathbf{e}}$, donc $\gamma(\mathbf{y}) = \mathbf{y} - \tilde{\mathbf{e}}\ \mathcal{R}\ \mathbf{0}$.
- $\forall\, \mathbf{x} \in C$, $\mathbf{y}-\mathbf{x}\ \mathcal{R}\ \tilde{\mathbf{e}}$, donc $d_H(\mathbf{x},\mathbf{y}) = w_H(\mathbf{y}-\mathbf{x}) \geq w_H(\tilde{\mathbf{e}}) = d_H(\gamma(\mathbf{y}),\mathbf{y})$, donc $\gamma(\mathbf{y})$ est un élément du code réalisant le minimum de la distance à $\mathbf{y}$.

– L'algorithme est linéaire. En effet, pour tout $\mathbf{x} \in C$, $\mathbf{y} \in \mathbf{F}_q^n$

$$\gamma(\mathbf{y}+\mathbf{x}) = \mathbf{y}+\mathbf{x}-e(\mathbf{y}+\mathbf{x}) = \mathbf{y}+\mathbf{x}-e(\mathbf{y}) = \gamma(\mathbf{y})+\mathbf{x},$$

puisque $e(\mathbf{y}+\mathbf{x}) = e(\mathbf{y})$.

◇

Le décodage des codes linéaires se ramène donc à la recherche d'un mot de plus petit poids d'un translaté donné par son syndrome. Cette recherche est généralement difficile, mais dans certains cas, comme celui des codes BCH, on possède un algorithme qui fonctionne pour certains translatés.

7.3.2 Décodage des codes cycliques

Dans cette section les vecteurs de $\mathbf{F}_q^n$ sont notés en gras.

a - Code de Hamming binaire $d = 3$

Le code de Hamming binaire $\mathcal{H}_m$ est un code linéaire de paramètres $(n = 2^m-1, 2^m-m-1, 3)$, les colonnes de sa matrice de parité sont les $2^m - 1$ vecteurs distincts non nuls de $\mathbf{F}_2^m$ (c'est bien une matrice $m \times n$) :

$$H_m = \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & 1 & \dots & 1 \\ 0 & 1 & 1 & 0 & \dots & 1 \\ 1 & 0 & 1 & 0 & \dots & 1 \end{pmatrix}$$

le code $\mathcal{H}_m$ est l'ensemble des combinaisons linéaires nulles des colonnes de H_m.

Exemple: $m = 3$, le code de Hamming $\mathcal{H}_3$ a pour paramètres $(7, 4, 3)$, et pour matrice de parité :

$$H_3 = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Proposition 7.29 *Chaque coset du code de Hamming $\mathcal{H}_m$ possède un et un seul mot de poids 1.*

preuve: Nous savons qu'il existe au plus un mot de poids 1 dans chaque coset, de plus il y a exactement $2^m - 1$ mots de poids 1, et $2^m - 1$ cosets différents de $\mathcal{H}_m$. ◇

Nous pouvons en déduire que l'algorithme de décodage suivant est à vraisemblance maximale.
Soit $\mathbf{y} = (y_0, \dots, y_{n-1}) \in \mathbf{F}_2^n$.

1. On calcule $\mathbf{s} = H_m \mathbf{y}^t \in \mathbf{F}_2^m$
2. si $\mathbf{s} = 0$ alors $\mathbf{y} \in C$ et $\longrightarrow \mathbf{y}$
3. sinon $\mathbf{s}$ est la j-ème colonne de H_m et $\longrightarrow \mathbf{x} = (y_0, \dots, y_{j-1}, 1+y_j, y_{j+1}, \dots, y_{n-1}) \in C$

(Notons que l'ordre dans lequel les colonnes de H_m sont écrites n'est pas important)

Le code de Hamming cyclique

On peut décider d'écrire la matrice H_m d'une façon légèrement différente, considérons la matrice :

$$H_m = \begin{pmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \end{pmatrix}$$

où α est un élément générateur de $\mathbf{F}_{2^m}^*$. $\mathcal{H}_m$ est l'ensemble des vecteurs de $\mathbf{F}_2^n$ orthogonaux à H_m (les calculs sont effectués dans $\mathbf{F}_{2^m}$). Ce code est cyclique.

Proposition 7.30 *$\mathcal{H}_m$ est l'ensemble des polynômes de $\mathbf{F}_2[x]/(x^n-1)$ s'annulant en α. C'est aussi l'idéal engendré par le polynôme minimal de α dans $\mathbf{F}_2$.*

Exemple: $m=3$, on considère le code de Hamming de matrice de parité :

$$H_3 = \begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \end{pmatrix}$$

où α est un élément générateur de $\mathbf{F}_8$ tel que $\alpha^3+\alpha+1=0$ (par exemple). En écrivant H_3 dans la base $1, \alpha, \alpha^2$, on obtient :

$$H_3 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

qui est la même que celle décrite au début à une permutation des colonnes près.

b - Codes 2-correcteur binaires, $d=5$

Nous considerons à présent le code C suivant : l'ensemble des polynômes de $\mathbf{F}_2[x]/(x^n-1)$ s'annulant en α et en α^3. Il s'agit d'un code linéaire binaire de longueur $n=2^m-1$. Sa matrice de parité peut s'écrire :

$$H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \\ 1 & \alpha^3 & \alpha^6 & \dots & \alpha^{3(n-1)} \end{pmatrix}$$

Ce code a une distance minimale $d \geq 5$ et peut donc corriger 2 erreurs.

Décodage

Nous faisons les hypothèses suivantes :
- $a(x) \in \mathbf{F}_2[x]/(x^n-1)$, tel que $a(\alpha)=a(\alpha^3)=0$
- $e(x)=x^{i_1}+x^{i_2}$, avec $0 \leq i_j \leq n-1$, pour $j=1,2$ et $i_1 \neq i_2$
- on connaît le polynôme $b(x)=a(x)+e(x)$

comment déterminer les valeurs de i_1 et i_2 ?

On pose $X_1=\alpha^{i_1}$ et $X_2=\alpha^{i_2}$, qui seront les inconnues, et on connaît les valeurs de $\xi_1 = b(\alpha)=e(\alpha)$ et de $\xi_3=b(\alpha^3)=e(\alpha^3)$. On veut donc résoudre dans $\mathbf{F}_{2^m}$ le système :

$$\begin{cases} X_1+X_2 &= \xi_1 \\ X_1^3+X_2^3 &= \xi_3 \end{cases}$$

on a :

$$\xi_1^3 = X_1^3+X_2^3+X_1X_2(X_1+X_2) = \xi_3 + X_1X_2\xi_1$$

on peut donc réécrire le système :

$$\begin{cases} X_1+X_2 &= \xi_1 \\ X_1X_2 &= \frac{\xi_1^3+\xi_3}{\xi_1} \end{cases}$$

on se ramène donc à la résolution d'une équation du second degré dans $\mathbf{F}_{2^m}$. ◇

c - Codes BCH de distance construite δ

Soit α un élément primitif de $\mathbf{F}_{q^m}$, $n=q^m-1$.

On défini le code BCH primitif au sens strict sur $\mathbf{F}_q$ de distance construite δ comme l'ensemble des polynômes de $\mathbf{F}_q[z]/(z^n-1)$ s'annulant en $\alpha, \alpha^2, \dots, \alpha^{\delta-1}$. Ce code a pour matrice de parité :

$$H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \dots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{\delta-1} & \alpha^{2(\delta-1)} & \dots & \alpha^{(n-1)(\delta-1)} \end{pmatrix}$$

Le théorème de la borne BCH nous assure que ce code a une distance minimale supérieure ou égale à δ, ce code est donc au moins t-correcteur où :

$$t = \lfloor \frac{\delta - 1}{2} \rfloor$$

pour $t > 2$, il est difficile d'effectuer un calcul direct de l'erreur.

Par exemple dans le cas binaire pour $t = 3$, il faudrait résoudre le système :

$$\begin{cases} X_1 + X_2 + X_3 = \xi_1 \\ X_1^3 + X_2^3 + X_3^3 = \xi_3 \\ X_1^5 + X_2^5 + X_3^5 = \xi_5 \end{cases}$$

où ξ_1, ξ_3 et ξ_5 seraient connus, et X_1, X_2 et X_3 inconnus. Ce système semble à première vue peu pratique.

En fait on sait décoder les codes BCH à l'aide de l'algorithme d'Euclide étendu. Dans la section suivante nous présenterons les définitions utiles à l'exposé de l'algorithme proprement dit.

d - polynômes localisateur et évaluateur

Soit α une racine n-ème de l'unité, $\mathbf{F}_{q^m}$ la plus petite extension de $\mathbf{F}_q$ contenant α.

Soit $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1}) \in \mathbf{F}_q^n$, de poids w, les composantes non nulles de ce vecteur sont exactement :

$$a_{i_1}, a_{i_2}, \ldots, a_{i_w}$$

on associe à $\mathbf{a}$ les éléments suivants de $\mathbf{F}_{q^m}$:

$$X_1 = \alpha^{i_1}, X_2 = \alpha^{i_2}, \ldots, X_w = \alpha^{i_w}$$

appelés les *localisateurs* de $\mathbf{a}$, ainsi que les éléments suivants de $\mathbf{F}_q$:

$$Y_1 = a_{i_1}, Y_2 = a_{i_2}, \ldots, Y_w = a_{i_w}$$

qui sont les valeurs des coordonnées non nulles de $\mathbf{a}$.

Définition 7.31 *Le* polynôme localisateur *de* $\mathbf{a}$ *est le polynôme :*

$$\sigma(z) = \prod_{i=1}^{w}(1 - X_i z) = \sum_{i=0}^{w} \sigma_i z^i \in \mathbf{F}_{q^m}[z].$$

Les racines de $\sigma(z)$ *sont les inverses des localisateurs de* $\mathbf{a}$.

Les coefficients σ_i de $\sigma(z)$ sont les fonctions symétriques élémentaires des X_i :

$$\begin{cases} \sigma_1 &= -(X_1 + \ldots + X_w), \\ \sigma_2 &= X_1X_2 + X_1X_3 + \ldots + X_{w-1}X_w, \\ &\vdots \\ \sigma_w &= (-1)^w X_1 \ldots X_w. \end{cases}$$

Définition 7.32 *Le* polynôme évaluateur *de* $\mathbf{a}$ *est le polynôme :*

$$\omega(z) = \sum_{i=1}^{w} X_i Y_i \prod_{\substack{j=1 \\ j \neq i}}^{w}(1 - X_j z)$$

on a pour tout i, $1 \leq i \leq w$ *:*

$$Y_i = \frac{X_i^{-1}\omega(X_i^{-1})}{\prod_{j \neq i}(1 - X_j X_i^{-1})}.$$

La connaissance de $\sigma(z)$ *et* $\omega(z)$ *permet de déterminer entièrement le vecteur* $\mathbf{a}$.

On a $\deg\omega(z) < \deg\sigma(z) = w$, et $\sigma(z)$ et $\omega(z)$ sont premiers entre eux.

Définition 7.33 *Le* polynôme de Mattson-Solomon *associé à* $\mathbf{a}$ *est le polynôme suivant de* $\mathbf{F}_{q^m}[z]$ *:*

$$A(z) = \sum_{j=1}^{n} A_j z^{n-j}, \text{ où } A_j = \sum_{i=0}^{n-1} a_i \alpha^{ij}.$$

A_j est la valeur en α^j du polynôme $a(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1}$, notons que A_j est défini pour tout entier j.

on a donc :

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ \vdots \\ A_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \ldots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{n-1} & \alpha^{2(n-1)} & \ldots & \alpha^{(n-1)^2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Théorème 7.34 *Soit* $\mathbf{a} = (a_0, \ldots, a_{n-1}) \in \mathbf{F}_q^n$, $\sigma(z)$ *le polynôme localisateur de* $\mathbf{a}$, $\omega(z)$ *son polynôme évaluateur. Pour tout* $j = 1, \ldots, \infty$, $A_j = a(\alpha^j)$ *où* $a(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1}$. *On a :*

$$\omega(z) = S(z)\sigma(z)$$

où

$$S(z) = \sum_{j=1}^{\infty} A_j z^{j-1}.$$

preuve:

$$\begin{aligned} \frac{\omega(z)}{\sigma(z)} &= \sum_{i=1}^{w} \frac{X_i Y_i}{1 - zX_i}, \\ &= \sum_{i=1}^{w} Y_i X_i \sum_{j=1}^{\infty} (zX_i)^{j-1}, \\ &= \sum_{j=1}^{\infty} z^{j-1} \sum_{i=1}^{w} Y_i X_i^j, \\ &= \sum_{j=1}^{\infty} z^{j-1} \sum_{i=0}^{n-1} a_i \alpha^{ij}, \\ &= \sum_{j=1}^{\infty} A_j z^{j-1} = S(z). \end{aligned}$$

◇

e - l'algorithme d'Euclide étendu

Quel problème doit on résoudre ? Soit $\mathbf{a} \in C \longrightarrow \mathbf{b} = \mathbf{a} + \mathbf{e} \in \mathbf{F}_q^n$, connaissant $\mathbf{b}$ on veut déterminer $\mathbf{e}$, en supposant que le poids de $\mathbf{e}$ est suffisamment petit.

Le décodeur a accès à $H\mathbf{b}^t = H\mathbf{e}^t$, ce qui revient, pour un code BCH primitif au sens strict, à connaître les valeurs en $\alpha, \alpha^2, \ldots, \alpha^{\delta-1}$ du polynôme $e(x) = e_0 + e_1 x + \ldots + e_{n-1}x^{n-1}$ associé au vecteur $\mathbf{e}$. Si l'on se réfère à la section précédente, il suffit de calculer les valeurs des polynômes localisateur et évaluateur de $\mathbf{e}$.

Proposition 7.35 *Soient* $e(x) = e_0 + e_1 x + \ldots + e_{n-1}x^{n-1} \in \mathbf{F}_q[x]/(x^n - 1)$ *de poids* $w_H(e) \leq t = \lfloor \frac{\delta-1}{2} \rfloor$, *et* $S(z) = \sum_{i=1}^{\infty} e(\alpha^i) z^{i-1}$. *Les polynômes localisateur et évaluateur de* $e(x)$ *sont les seuls polynômes* $\sigma(z)$ *et* $\omega(z)$ *vérifiant :*

$$\begin{cases} \omega(z) \equiv \sigma(z)S(z) \mod z^{\delta-1} \\ \omega(z) \text{ et } \sigma(z) \text{ premiers entre eux} \\ \sigma(0) = 1 \\ \deg \omega(z) < t \\ \deg \sigma(z) \leq t \end{cases} \tag{7.1}$$

preuve: D'après la section précédente, les polynômes localisateur et évaluateur de e vérifient (7.1). Réciproquement soient $\tilde{\sigma}(z)$ et $\tilde{\omega}(z)$ vérifiant (7.1), on a :

$$\tilde{\omega}(z)\sigma(z) \equiv \tilde{\sigma}(z)S(z)\sigma(z) \equiv \tilde{\sigma}(z)\omega(z) \mod z^{\delta-1},$$

$$\tilde{\omega}(z)\sigma(z) - \tilde{\sigma}(z)\omega(z) \equiv 0 \mod z^{\delta-1},$$

or $\tilde{\omega}(z)\sigma(z) - \tilde{\sigma}(z)\omega(z)$ est de degré $< \delta - 1$, donc $\tilde{\omega}(z)\sigma(z) = \tilde{\sigma}(z)\omega(z)$, ce qui suffit à montrer le résultat. ◇

Description de l'algorithme

Théorème 7.36 (Algorithme d'Euclide) *Soient* $r_{-1}(z)$ *et* $r_0(z)$, *tels que* $\deg r_0 \leq \deg r_{-1}$, *on construit deux suites* $(r_i(z))_{i\geq 1}$ *et* $(q_i(z))_{i\geq 1}$ *telles que :*

$$\begin{array}{rcll} r_{-1}(z) & = & q_1(z)r_0(z) + r_1(z), & \deg r_1 < \deg r_0 \\ r_0(z) & = & q_2(z)r_1(z) + r_2(z), & \deg r_2 < \deg r_1 \\ r_1(z) & = & q_3(z)r_2(z) + r_3(z), & \deg r_3 < \deg r_2 \\ & \vdots & & \vdots \\ r_{j-2}(z) & = & q_j(z)r_{j-1}(z) + r_j(z), & \deg r_j < \deg r_{j-1} \\ r_{j-1}(z) & = & q_{j+1}(z)r_j(z). & \end{array}$$

alors $r_j(z)$, *le dernier reste non nul de ces divisions est le* pgcd *de* $r_{-1}(z)$ *et* $r_0(z)$.

Soient les polynômes $U_i(z)$ et $V_i(z)$ définis par :

$$\begin{array}{ll} U_{-1}(z) = 0, & U_0(z) = 1, \\ V_{-1}(z) = 1, & V_0(z) = 0, \end{array}$$

$$\begin{array}{l} U_i(z) = U_{i-2}(z) - q_i(z)U_{i-1}(z), \\ V_i(z) = V_{i-2}(z) - q_i(z)V_{i-1}(z). \end{array}$$

alors pour tout $i \geq 1$ on a :

$$\begin{cases} r_i(z) = V_i(z)r_{-1}(z) + U_i(z)r_0(z) \\ \deg U_i = \deg r_{-1} - \deg r_{i-1} \\ U_i(z) \text{ et } V_i(z) \text{ premiers entre eux} \end{cases} \tag{7.2}$$

c'est cette relation que nous utiliserons pour l'algorithme de décodage.

Algorithme de décodage

On veut résoudre (7.1) :

$$\begin{cases} \omega(z) \equiv \sigma(z)S(z) \mod z^{\delta-1} \\ \omega(z) \text{ et } \sigma(z) \text{ premiers entre eux} \\ \sigma(0) = 1 \\ \deg \omega(z) < t \\ \deg \sigma(z) \leq t \end{cases}$$

$S(z) \mod z^{\delta-1}$ étant connu.

Si on applique l'algorithme d'Euclide à $r_{-1}(z) = z^{\delta-1}$ et $r_0(z) = (S(z) \mod z^{\delta-1})$, on peut alors construire à l'aide de (7.2), des polynômes $r_i(z)$ et des $U_i(z)$ vérifiant :

$$r_i(z) \equiv U_i(z) r_0(z) \mod z^{\delta-1},$$

il existe un rang k pour lequel $\deg r_{k-1} \geq t$ et $\deg r_k < t$, on aura alors :

$$\begin{cases} \deg r_k < t, \\ \deg U_k = \deg r_{-1} - \deg r_{k-1} \leq t \end{cases}$$

à un coefficient multiplicatif près, on a donc résolu le problème. Soit λ tel que $\lambda U_k(0) = 1$, alors :

$$\begin{cases} \sigma(z) = \lambda U_k(z) \\ \omega(z) = \lambda r_k(z). \end{cases}$$

est la solution du système (7.1). ◇

interprétation pour le décodage Soit $a(x) \in \mathbf{F}_2[x]/(x^n - 1)$, tel que

$$a(\alpha) = a(\alpha^2) = \ldots = a(\alpha^{\delta-1}) = 0,$$

on considère le polynôme $b(x) = a(x) + e(x)$, où $e(x)$ est de poids $w < \delta/2$.

Connaissant $b(x)$, nous voulons déterminer $a(x)$ et $e(x)$. Puisque pour $i = 1, \ldots, \delta - 1$, on a $a(\alpha^i) = 0$, pour ces même valeurs de i, $e(\alpha^i) = b(\alpha^i)$ est connu. Le syndrome de $e(x)$, $S(z) = \sum_{i=1}^{\delta-1} e(\alpha^i) z^{i-1}$, nous permet à l'aide de l'algorithme d'Euclide étendu de déterminer la valeur de $e(x)$ par l'intermédiaire de ses polynômes localisateur et évaluateur.

Chapitre 8

Codes convolutifs

8.1 Définition

Définition 8.1 *Un codeur convolutif binaire (k, n, m) est un circuit linéaire sans feedback, d'ordre m, à k entrées et n sorties sur $\mathbf{F}_2$, et invariant dans le temps.*

- $\vec{X}(D) = (X_1(D), \ldots, X_k(D))$ le vecteur d'entrée,
  $X_i(D) = \sum_{j=t_0}^{\infty} x_{ij} D^j$ la transformée de Huffman de la i-ème entrée,
  $\vec{Y}(D) = (Y_1(D), \ldots, Y_n(D))$ le vecteur de sortie
- On a la relation
$$\vec{Y}(D) = \vec{X}(D) \cdot G(D)$$
où $G(D)$ est la matrice de transfert, $G(D) = (g_{ij}(D))_{1 \le i \le k, 1 \le j \le n}$ avec $g_{ij}(D) \in \mathbf{F}_2[D]$.
- Par définition l'ordre du codeur est égal à $m = \max_{1 \le i \le k, 1 \le j \le n} \deg g_{ij}(D)$
- On peut considérer $G(D)$ comme un polynôme sur l'anneau des matrices $k \times n$ sur $\mathbf{F}_2$:
$$G(D) = G_0 + G_1 D + \ldots + G_m D^m$$
avec $G_l = \big([D^l] g_{ij}(D)\big)_{1 \le i \le k, 1 \le j \le n}$ pour tout l, $0 \le l \le m$.

Définition 8.2 *Un* code convolutif *est l'ensemble des séquences produites par un codeur convolutif.*

Exemple: Le codeur binaire $(2, 1, 3)$ de matrice de transfert
$$G(D) = \begin{pmatrix} 1 + D^2 + D^3 & 1 + D + D^2 + D^3 \end{pmatrix}$$
peut être représenté par le circuit donné en figure 8.1.

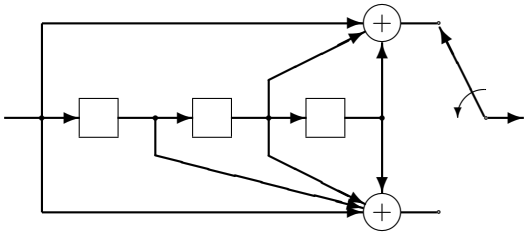


FIGURE 8.1 – Codeur convolutif $(2, 1, 3)$

Exemple: Le codeur binaire $(3, 2, 1)$ de matrice de transfert

$$G(D) = \begin{pmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{pmatrix}$$

peut être représenté par le circuit donné en figure 8.2.

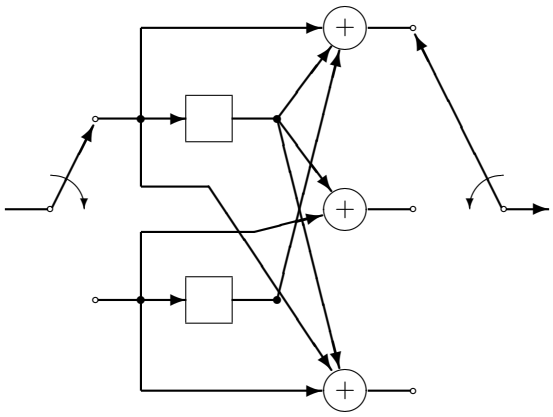


FIGURE 8.2 – Codeur convolutif $(3, 2, 1)$

8.2 Diagramme d'état – Treillis de codage

Définition 8.3 *La* mémoire *d'un codeur convolutif, notée K, est définie comme le nombre minimal de registres nécessaire à la réalisation du codeur.*

Pour le codeur de l'exemple 8.1 on a $K = 3$, et pour le codeur de l'exemple 8.1 on a $K = 2$.

L'état d'un codeur convolutif est défini comme la valeur de ses registres. Le nombre d'états possible pour un codeur binaire est donc 2^K.

8.2.1 Diagramme d'état

Le diagramme d'état d'un codeur est un graphe orienté ayant comme nœuds les états du codeur et comme branches les transitions d'un état vers un autre. Une séquence codée correspond à un chemin dans ce graphe.

Dans la figure 8.3 nous représentons le diagramme d'état du code de l'exemple 8.1. Ce codeur admet huit états, et à partir de chaque état deux transitions sont possibles.

Sur chacune des branches du graphe nous donnons également la valeur de l'entrée et la valeur de la sortie, ici l'entrée consiste en un seul bit, et la sortie en 2 bits.

8.2.2 Treillis de codage

Le treillis de codage d'un codeur convolutif est son diagramme d'état étendu dans le temps, c'est-à-dire que chaque unité de temps possède son propre diagramme d'état.

Exemple: Prenons un codeur $(3, 1, 2)$ de matrice de transfert

$$G(D) = \begin{pmatrix} 1+D & 1+D^2 & 1+D+D^2 \end{pmatrix}.$$

Pour un tel code le treillis de codage est donné par la figure 8.4. La valeur inscrite sur chaque branche est la valeur de sortie au temps considéré.

Dans la figure 8.4, nous partons de l'état 00, et à chaque étape nous ajoutons les branches correspondant aux transitions possibles.

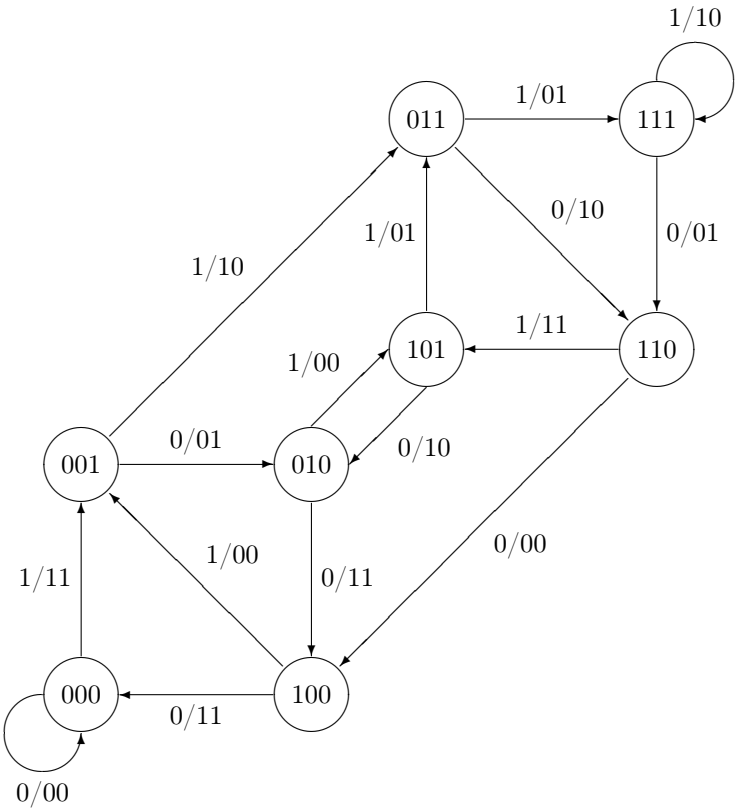


FIGURE 8.3 – Diagramme d'état d'un code $(2, 1, 3)$

8.3 Décodage

Le décodage d'un code convolutif va consister à trouver parmi les différents chemins du treillis de codage le chemin le plus probable pour aller d'un état donné au temps t_0 à un autre état donné au temps t.

8.3.1 Algorithme de Viterbi

L'algorithme de Viterbi suppose que le nombre d'états du codeur est faible.

On se place dans le cas d'un canal binaire symétrique, la métrique de Hamming permet alors de choisir la séquence la plus probable.

On considère la séquence X_t correspondant à un chemin quelconque s'arrêtant au temps t dans le treillis de codage, on pose $X_t = (x_{t_0}, \ldots, x_t)$, avec $x_i \in \mathbf{F}_2^n$.

Si $Y_t = (y_{t_0}, \ldots, y_t)$, avec $y_i \in \mathbf{F}_2^n$, est la séquence reçue jusqu'au temps t, on peut calculer facilement $d_H(X_t, Y_t)$ à partir de $d_H(X_{t-1}, Y_{t-1})$:

$$d_H(X_t, Y_t) = d_H(X_{t-1}, Y_{t-1}) + d_H(x_t, y_t).$$

(X_{t-1} et Y_{t-1} sont les séquences X_t et Y_t tronquées au temps $t-1$)

L'algorithme de Viterbi :
On suppose que pour chaque état du codeur au temps t, le meilleur chemin est connu ainsi que la distance de Hamming entre la séquence codée correspondant à ce chemin et la séquence reçue.

1. On calcule pour chaque état au temps t, tous ses successeur (il y en a 2^k).

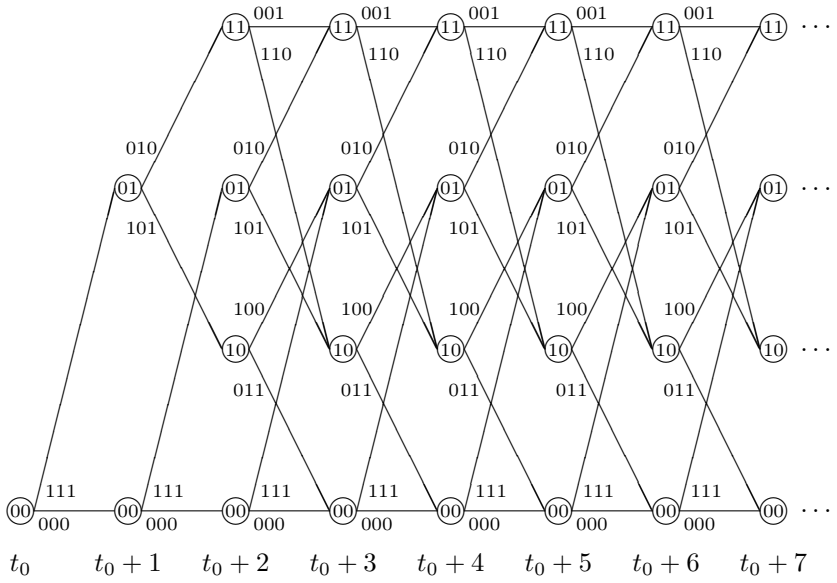


FIGURE 8.4 – Treillis de codage d'un code $(3, 1, 2)$

2. Chaque état à l'instant $t+1$ peut être atteint de 2^k façons, on conserve le chemin tel que la distance de Hamming à la séquence reçue soit minimale.

Exemple: Nous reprenons le codeur de l'exemple 8.2.2. Supposons que la séquence d'information soit $(1, 1, 1, 0, 1, 0, 0)$, avec une valeur des registres initialement nulle, la séquence émise est alors :

$$111 \quad 010 \quad 001 \quad 110 \quad 100 \quad 101 \quad 011 \ .$$

On convient généralement qu'une séquence codée doit ramener les registres à des valeurs nulles, donc la fin de la séquence d'information est constituée de 0. Après passage dans le canal nous recevons

$$111 \quad \underline{1}10 \quad 001 \quad 110 \quad 1\underline{11} \quad 101 \quad 011 \ .$$

La figure 8.5 nous donne l'ensemble des chemins possibles, il suffit alors de déterminer de proche en proche le plus probable.

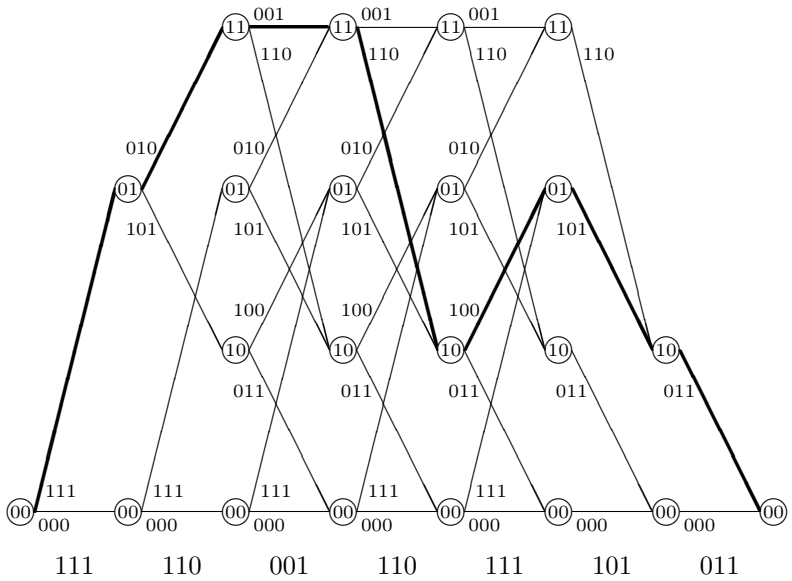


FIGURE 8.5 – Exemple de décodage

Table des matières