THEORIE DE L'INFORMATION

Table des matières

Bibliographie

[0] **Anya Désilles.** Théorie de l'information. *Polycopié de cours*, 2010.
[1] **Robert B. Ash.** Information theory. *Dover Publications Inc.*, 1965.
[2] **David J.C. MacKay.** Information Theory, Inference and Learning Algorithms. *Cambridge University Press*, 2003. Disponible au téléchargement gratuit sur http://www.inference.phy.cam.ac.uk/mackay/itila/book.html.
[3] **John R. Pierce.** An Introduction to Information Theory. Symbols, Signals and noise. *Dover Publications Inc*, NY 1980.
[4] **Fazlollah M. Reza.** An introduction to information theory. *Dover Publications Inc.*, 1994.
[5] **Julien Salomon.** Traitement numérique du signal. *Polycopié de cours*, 2009.
[6] **Olivier Rioul.** Théorie de l'information et du codage. *Hermes Lavoisier*, 2007.

Ce polycopié s'inspire de celui d'Anya Désilles [0] que je remercie très chaleureusement.

Introduction

La théorie de l'information trouve ses origines dans les débuts des communications électriques. Le premier télégraphe élaboré par S. Morse entre 1832 et 1838 a permis de communiquer n'importe quel texte à l'aide de signaux électriques.

Dans le code de Morse chaque lettre de l'alphabet est représentée par une séquence de :
– points (courant électrique de courte durée);
– traits (courant de longue durée);
– espaces (absence de courant);

On peut remarquer que dans la version définitive du code la lettre "e", la plus fréquente dans l'anglais, est représentée par la séquence la plus courte : un seul point. A cette époque, S. Morse n'avait pas fait d'analyse théorique pour arriver à cette conclusion, mais plutôt des observations empiriques. Son but en effet était de concevoir le code tel que la saisie d'un texte par un opérateur soit, en moyenne, la plus rapide possible. Aujourd'hui, la théorie moderne de communication a montré qu'il est possible de gagner à peine 15 % de temps de saisie par rapport au code de Morse.

Quelques années plus tard, une première ligne télégraphique est installée entre Washington et Baltimore. En enterrant les câbles, S. Morse rencontre une nouvelle difficulté : le milieu dans lequel ces derniers se trouvent influe sur la qualité de transmission. En particulier, il remarque que si l'opérateur saisit trop vite son code, le signal reçu est indéchiffrable. Les points, les traits, les espaces sont confondus dans un seul signal d'intensité moyenne. Ainsi apparaît le problème de perturbations dues aux conditions physiques de transmission de l'information et, comme condition de bonne réussite, la nécessité de limiter le débit d'émission de symboles.

Bien plus tard, dans les années 1940, l'ingénieur et mathématicien C. Shannon et le mathématicien Warren Weaver, ont formalisé le processus de communication et donné les outils mathématiques permettant de répondre aux questions posées empiriquement par les premiers télégraphes. Ils proposent en particulier, une représentation schématique du processus de communication, connue sous le nom de paradigme de Shannon.

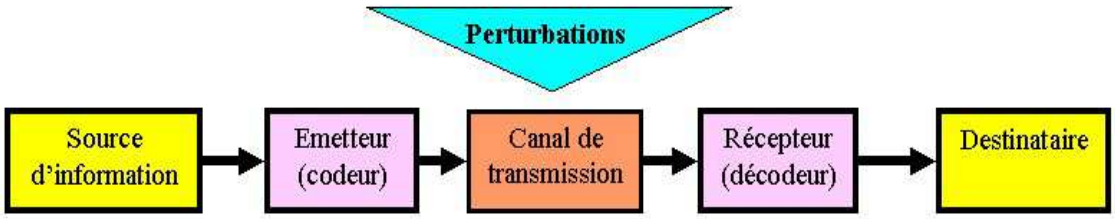


Figure 1. *Paradigme de Shannon*

Ce modèle est une *approximation linéaire de processus de communication* qui met l'accent sur les aspects purement techniques de transmission d'un message. On peut résumer ce modèle de la façon suivante :

1. La source d'information choisit un message M parmi un certain nombre de messages possibles.
2. L'émetteur transforme le message en signal S compatible physiquement avec le mode de transmission choisi. On dit qu'il *encode* le message.
3. Le signal S est alors soumis à l'entrée d'un canal de transmission.
4. Lors de la transmission des perturbations peuvent intervenir et transformer le signal envoyé. On parle alors de *bruit de canal* B.
5. A la sortie du canal, le signal $\tilde{S}$ éventuellement entaché d'erreurs dues au bruit, est soumis au décodeur qui le transforme en message $\tilde{M}$ lisible par le destinataire.

Voici un exemple pour illustrer ce schéma. Imaginez qu'un ami vous envoie une carte postale du lieu de ses vacances. La carte voyage sans incident jusqu'à la ville où vous habitez. Le jour où le facteur doit enfin vous l'apporter, il la fait tomber par mégarde. Malheur ! Il pleut. L'enveloppe a pris l'eau. C'est ainsi que vous retrouvez les quelques lignes écrites par votre ami à moitié illisibles. Allez-vous pouvoir reconstituer le contenu complet du message ?

Dans cet exemple, la *source d'information* est votre ami, et plus précisément, son cerveau. C'est encore lui qui joue ici le rôle de l'*émetteur*, en transformant ses idées en mots et ensuite en écrivant les mots à l'aide de lettres de l'alphabet. Le *canal de transmission* est ici représenté par les services postaux. Les dégâts causés par l'eau à la carte postale représentent le *bruit de canal*. Les questions que l'on pourrait poser dans cette situation sont les suivantes :

1. Y a-t-il un moyen de préparer le message de façon à éviter les désagréments causés par les erreurs de transmission et garantir à l'arrivée la lisibilité du message ? Par exemple, pourrait-on écrire avec une encre indélébile ? C'est le problème de **codage de source**.

2. Y a-t-il un moyen de transporter le signal dans le canal de manière à limiter les perturbations ? Par exemple, protéger le courrier dans des enveloppes imperméables ? C'est le problème de **codage de canal**.

3. Comment évaluer l'incertitude que le récepteur a sur le contenu réel du message reçu ? C'est le problème de **mesure de l'information**.

Ce modèle, certes très simplifié, de la communication, a servi de base dans le développement de la théorie de l'information à partir des années 1940. Il s'inscrit dans une description plus générale, donnée par Warren Weaver, des 3 niveaux de problèmes de communication :

Niveau 1 : Technique : Avec quelle précision peut-on transmettre les symboles de la communication ?
Niveau 2 : Sémantique : Dans quelle mesure les symboles véhiculent-ils la signification ?
Niveau 3 : Efficacité : Dans quelle mesure la signification reçue influence-t-elle le comportement et l'action du destinataire ?

La théorie de l'information s'intéresse uniquement aux problèmes du 1er niveau de communication. En particulier, le sens des messages traités n'a aucune importance.

L'information quantifiable associée à un message donné n'est pas dans son sens sémantique mais dans sa rareté. Un message, même très significatif, connu à l'avance par le récepteur ne lui apporte aucune information au sens technique du terme. Par contre, la réception d'un message très improbable mais démuni de tout sens est dans ce cadre considérée comme événement porteur d'une grande quantité d'information.

Les chapitres qui suivent ont pour objectif d'introduire les concepts principaux de la théorie de l'information et les théorèmes fondamentaux. Ces derniers établissant les limites théoriques en matière de codage de l'information et de transmission. Ensuite, des applications seront consacrées à l'étude de quelques algorithmes de base de codage de source.

1. Modélisation mathématique d'une source d'information

1. Représentation mathématique d'une source

Soit une source (un émetteur) produisant des symboles d'un alphabet fini :

$$\Omega = \{\omega_1, \cdots, \omega_m\}.$$

Nous supposons que chaque symbole ω_i, $i = 1, 2, \cdots, m$ est émis aléatoirement avec une probabilité connue p_i de telle sorte que :

$$\sum_{i=1}^{m} p_i = 1.$$

On considère alors l'expérience aléatoire consistant à observer un symbole émis. A cette expérience nous associons une variable aléatoire, notée X, à valeurs dans Ω.
C'est une variable aléatoire discrète, dont la loi est définie par l'ensemble des probabilités d'émission des symboles de l'alphabet :

$$P[X = \omega_i] = p(\omega_i) = p_i \qquad i = 1, \cdots, m$$

Exemple 1.1. Soit X la variable associée à une source binaire, produisant des symboles 0 et 1 avec les probabilités respectives p et $q = 1 - p$.
Nous avons ici l'alphabet constitué de 2 symboles $\Omega = \{0, 1\}$ et $P[X = 0] = p$, $P[X = 1] = q$.

Exemple 1.2. Soit Y la variable associée à une source disposant d'un vocabulaire de 20 mots :
Ω = {*le, la, pomme, fruit, est, un, arbre, pommier, jardin, ombre, fait, soleil, vent, pluie, grandit, quand, ne, pas, mais, oiseau*} et tous les mots sont supposés *équiprobables*.
Ainsi, n'importe quel mot $M_i \in \Omega$ du vocabulaire est choisi avec la probabilité $1/20$ et :

$$P[X = M_i] = \frac{1}{20} \qquad i = 1, \cdots, 20.$$

Dans la suite nous allons considérer des événements associés à la variable aléatoire X comme des sous-ensembles de Ω et la mesure de probabilité définie pour un événement $A \subset \Omega$ par :

$$P[A] = P[X \in A]$$

Si on considère l'émission de plusieurs symboles successifs, à des instants précis $\{t_1, t_2, \cdots, t_n\}$, on peut y associer la suite de variables aléatoires $\{X_1, X_2, \cdots, X_n\}$, chacune correspondant à l'émission d'un symbole $s(t) \in \Omega,\ t \in \{t_1, t_2, \cdots, t_n\}$. Nous supposons pour l'instant que tous les symboles successifs sont émis de façon indépendante les uns des autres et avec les mêmes probabilités d'émission :

$$p_i = P[s(t) = \omega_i]$$

Cela signifie que les variables aléatoires X_k, $k = 1, \cdots, n$ sont *deux à deux indépendantes et identiquement distribuées*. On parle alors d'une *source stationnaire et sans mémoire*. Nous allons préciser la signification formelle de ces termes plus loin.

2. Information propre

Pour introduire l'ensemble des axiomes qui définissent l'information propre d'un événement $A \subset \Omega$, nous allons nous servir du 1er exemple de source binaire précédent. Soit donc une source binaire d'alphabet $\Omega = \{0, 1\}$ et la variable aléatoire X associée avec les probabilités données :

$$p(0) = P[X = 0] = p \quad \text{et} \quad P(1) = P[X = 1] = q = 1 - p$$

Dans ce qui suit, nous allons construire de façon axiomatique, une fonction

$$h:\ \mathrm{P}(\Omega) \to \mathbf{R}$$

associant à un événement $A \subset \Omega$, la mesure de la quantité d'information contenue dans A. Nous l'appellerons **"information propre d'un événement A"**.

Nous avons déjà indiqué de façon intuitive que la quantité d'information apportée par un événement devrait être d'autant plus grande que l'événement est rare, improbable. Nous pouvons donc postuler que la quantité d'information d'un événement doit être de la forme :

$$h(A) = f\left(\frac{1}{P(A)}\right)$$

où f est une fonction croissante.

En particulier, dans notre exemple de source binaire $A = \{\omega\}$, avec $\omega \in \Omega$. On doit avoir :

$$h(\omega) = f\left(\frac{1}{P(\omega)}\right) \qquad \omega \in \{0,1\}$$

Si un événement A est certain, c'est-à-dire si $P[A] = 1$, il n'apporte aucune information. La fonction f que l'on recherche doit donc vérifier la condition limite suivante :

$$\lim_{x \to 1} f(x) = 0$$

Par ailleurs, on doit s'attendre à ce que l'information apportée par 2 événements indépendants, A et B, soit égale à la somme des informations apportées par chacun de ces événements (*rappel :* 2 événements indépendants A et B sont tels que $P(A \cap B) = P(A) \cdot P(B)$) :

$$h(A \cap B) = h(A) + h(B)$$

et donc

$$h(A \cap B) = f\left(\frac{1}{P(A \cap B)}\right) = f\left(\frac{1}{P(A)} \cdot \frac{1}{P(B)}\right) = f\left(\frac{1}{P(A)}\right) + f\left(\frac{1}{P(B)}\right)$$

Ainsi, la fonction f doit vérifier la propriété suivante :

$$f(xy) = f(x) + f(y)$$

L'unique fonction (à une constante multiplicative près) qui vérifie toutes ces propriétés est la fonction *logarithme*. Ainsi, nous définissons l'information propre d'un événement de la façon suivante :

Définition 1.1. Information propre. Soient $\Omega = \{\omega_1, \cdots, \omega_m\}$, un alphabet discret et X la variable aléatoire associée. Pour tout événement $A \subset \Omega$, la quantité d'information propre de A est définie par

$$h(A) = -\log_2(P(A))$$

Remarque 1.1. Dans la suite nous allons considérer le logarithme de base 2. L'unité de mesure de l'information est alors le *Shannon*. Le changement de base du logarithme provoque la modification de la constante multiplicative car

$$\log_a(x) = \log_a(b) \cdot \log_b(x)$$

Voici quelques propriétés élémentaires de la fonction $h(A)$ qui sont les conséquences immédiates des propriétés de la fonction *logarithme*.

Proposition 1.1. Propriétés de l'information propre.

1. La quantité d'information est toujours une grandeur *positive* : $\forall A \subset \Omega \quad h(A) \geq 0$

2. Elle est nulle si et seulement si l'événement A est *certain* : $h(A) = 0 \quad \Leftrightarrow \quad P(A) = 1$

3. $\lim_{P(A) \to 0} h(A) = +\infty$

3. Information conditionnelle

Nous avons déjà évoqué dans la section précédente la question d'évaluation de la quantité d'information apportée par la réalisation conjointe de 2 événements A et B. Nous avons postulé que dans le cas de 2 événements *indépendants*, on a :

A et B *indépendants* : $h(A \cap B) = h(A) + h(B)$

Or, dans le cas où A et B *ne sont pas indépendants*, on peut utiliser la notion de probabilité conditionnelle :

Probabilité de A *sachant* B (ou encore *Probabilité de* A *si* B) définie par : $P(A|B) = \dfrac{P(A \cap B)}{P(B)}$

d'où les relations :

$$P(A \cap B) = P(B)P(A|B) = P(A)P(B|A)$$

Alors, la quantité d'information propre de $A \cap B$ est :

$$h(A \cap B) = -\log_2(P(A \cap B)) = -\log_2(P(A)P(B|A)) = -\log_2(P(A)) - \log_2(P(B|A)) = h(A) - \log_2(P(B|A))$$

Définition 1.2. Information conditionnelle. On appelle *information conditionnelle de* B *sachant* A la quantité :

$$h(B|A) = -\log_2(P(B|A))$$

Remarque 1.2. On peut donc écrire $h(A \cap B)$ de la façon suivante :

$$h(A \cap B) = h(A) + h(B|A) \quad \rightarrow \quad h(B|A) = h(A \cap B) - h(A)$$

On peut interpréter la quantité d'information conditionnelle comme suit : $h(B|A)$ représente l'information apportée par l'observation de l'événement B *qui n'a pas déjà été apportée* par l'observation de A.

On a la relation duale :

$$h(A \cap B) = h(B) + h(A|B) \quad \rightarrow \quad h(A|B) = h(A \cap B) - h(B)$$

4. Information mutuelle

Définition 1.3. Information mutuelle. On appelle *information mutuelle* de 2 événements A et B la quantité :

$$i(A;B) = h(A) - h(A|B) = \log_2 \frac{P(A|B)}{P(A)} = \log_2 \frac{P(A \cap B)}{P(A)P(B)}$$

En particulier, $i(A;B)$ est nulle lorsque les 2 événements A et B sont indépendants.

5. Information et incertitude

La définition de l'information associée à un événement A implique que cette quantité est d'autant plus grande que l'événement est rare (c'est-à-dire que sa probabilité est proche de zéro). Ainsi, on peut assimiler l'information apportée par l'observation de A et l'incertitude que nous avons sur sa réalisation, ou encore la difficulté de prévoir A.

Considérons par exemple l'expérience de lancé d'un dé équilibré à 6 faces. Soit X la Variable Aléatoire (VA) associée :

$$\forall \omega \in \{1, 2, 3, 4, 5, 6\} \qquad P[X = \omega] = \frac{1}{6}$$

Imaginons que le résultat soit $X = 2$. Soit A l'événement associé. On a $P(A) = \frac{1}{6}$.

Si on essaie de deviner ce résultat, on a 1 chance sur 6 de trouver la réponse correcte. L'information associée est :

$$h(A) = -\log_2\left(\frac{1}{6}\right) = \log_2(6) \approx 2.585 \text{ } \textit{Shannon}$$

Imaginons que l'on affirme que le résultat du lancé est un nombre pair. Soit B l'événement associé. On a $P(B) = \frac{1}{2}$.

Alors, la quantité d'information que nous apporte ce renseignement est

$$h(B) = -\log_2\left(\frac{1}{2}\right) = \log_2(2) = 1 \text{ } \textit{Shannon}$$

Cette information augmente notre chance de réussite. En effet, nous avons maintenant la probabilité $P(A|B) = \frac{1}{3}$ de deviner le résultat. Et nous retrouvons la décomposition de l'information associée à $A \cap B$ comme somme de l'information apportée par l'observation de B et de l'information conditionnelle $h(A|B)$:

$$h(A \cap B) = h(B) + h(A|B) = -\log_2\left(\frac{1}{2}\right) - \log_2\left(\frac{1}{3}\right) = \log_2(2) + \log_2(3) = \log_2(6)$$

et la quantité d'information associée à $h(A \cap B)$ est identique à celle de $h(A)$.

6. Entropie d'une source

On considère maintenant une source stationnaire et sans mémoire, produisant à des instants précis $\{t_1, t_2, \cdots, t_k\}$, des symboles $s(t) \in \Omega_X$, $t \in \{t_1, t_2, \cdots, t_k\}$ d'un alphabet Ω_X donné.

L'*entropie* d'une telle source représente la *quantité moyenne d'information propre* associée à l'observation de chacun des symboles possibles.

Supposons que l'alphabet de la source $\Omega_X = \{s_i,\ i = 1, \cdots, n\}$ comporte n symboles et que la Variable Aléatoire (VA) associée à l'observation d'un symbole émis, X, a la distribution de probabilité donnée $\{p_i = P[X = s_i],\ i = 1, \cdots, n\}$. L'entropie de la source est alors la fonction des n probabilités $H(p_1, p_2, \cdots, p_n)$.

Tout comme pour l'information propre, l'entropie peut être définie par l'ensemble d'axiomes suivant :

1. Si tous les symboles sont équiprobables l'entropie $H(1/n, 1/n, \cdots, 1/n) = f(n)$ est une fonction de n, la taille de l'alphabet. Nous souhaitons qu'elle soit la mesure de l'incertitude associée à la source, ou encore de la difficulté à prédire le symbole émis. Il est alors naturel de supposer que l'entropie augmente avec la taille de l'alphabet. Par exemple, il est plus difficile de deviner le résultat du lancé d'un dé à 6 faces que celui d'une pièce de monnaie. Ainsi, la fonction $f(n) = H(1/n, 1/n, \cdots, 1/n)$ ci-dessus doit être *croissante*.

2. Considérons une source qui émet des couples de symboles puisés chacun dans un alphabet indépendant de tailles respectives l et m. La taille de l'alphabet produit est alors $l \cdot m$. Or, l'observation de chaque couple apporte la somme d'informations liées à chaque symbole. Ainsi, la fonction f doit vérifier la propriété d'*additivité* : $f(l \cdot m) = f(l) + f(m)$.

3. La fonction $H(p_1, p_2, \cdots, p_n)$ est *continue* selon chacune de ses variables.

4. Propriété de groupe. Si l'on divise l'alphabet donné Ω_X en 2 parties disjointes : $A = \{s_1, \cdots, s_r\}$ et $B = \{s_{r+1}, \cdots, s_n\}$, et dont la réunion est Ω_X (A et B réalisent une partition de Ω_X), chacune peut être considérée comme alphabet. Notons $p = P(A) = \sum_{k=1}^{r} p_k$ et $q = 1 - p = P(B) = \sum_{k=r+1}^{n} p_k$. On a : $p + q = 1$. On peut alors définir sur A et sur B, les distributions de probabilités respectives : $\left\{\frac{p_1}{p}, \cdots, \frac{p_r}{p}\right\}$ et $\left\{\frac{p_{r+1}}{q}, \cdots, \frac{p_n}{q}\right\}$. Alors l'entropie associée à l'alphabet Ω_X se décompose de la façon suivante :

$$H(X) = H_{\Omega_x} = H(p_1, \cdots, p_r, p_{r+1}, \cdots, p_n) = H(p, 1-p) + pH\left(\frac{p_1}{p}, \cdots, \frac{p_r}{p}\right) + (1-p)H\left(\frac{p_{r+1}}{1-p}, \cdots, \frac{p_n}{1-p}\right) = H(p,q) + pH_A + qH_B$$

On remarquera que $P_A = \left\{\frac{p_1}{p}, \cdots, \frac{p_r}{p}\right\}$ et $Q_B = \left\{\frac{p_{r+1}}{1-p}, \cdots, \frac{p_n}{1-p}\right\}$ définissent des distributions de probabilités respectivement sur *A* et *B*.

On peut interpréter ainsi cette propriété : L'incertitude moyenne (*l'entropie*) sur X est composée de :
1. l'incertitude sur le choix de l'une des 2 parties A et B : c'est $H(p,q)$
2. la moyenne des incertitudes associées à chacune des 2 parties séparément : c'est $pH_A + qH_B$

5. *Symétrie.* Il est naturel de supposer que la quantité moyenne d'information associée à un alphabet Ω_X ne dépend pas de l'ordre (de numérotation) des symboles. Ainsi, la permutation de 2 arguments de la fonction d'entropie ne la modifie pas :

$$H(p_1, \cdots, p_i, \cdots, p_j, \cdots, p_n) = H(p_1, \cdots, p_j, \cdots, p_i, \cdots, p_n)$$

La fonction vérifiant ces 4 axiomes est unique à une constante multiplicative près. Après la publication en 1948 par C. Shannon de son article *"The Mathematical Theory of Communication"*, de nombreux mathématiciens ont travaillé à l'élaboration de la définition axiomatique de l'entropie. On peut citer Khinchin (1957), Fadeev (1956), Kullback (1958), Rényi (1962). En sélectionnant des ensembles d'axiomes différents, tous ont abouti à la même fonction. D'où la définition qui suit.

Définition 1.4. Entropie d'une source. Soient $\Omega = \{\omega_1, \cdots, \omega_m\}$ l'alphabet fini d'une source et X la variable aléatoire associée telle que $P[\omega_i] = p_i, \quad i = 1, \cdots, m$. On appelle **entropie** ou encore **quantité moyenne d'information** de la source la quantité :

$$H(X) = H(p_1, p_2, \cdots, p_n) = E[h(\omega)] = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

L'unité de mesure de cette quantité est le *"bit par symbole"*.

L'entropie représente la quantité d'information que l'on obtient en moyenne, en observant les symboles en sortie de la source pendant suffisamment longtemps ou encore, la valeur de l'information moyenne obtenue en observant simultanément les symboles en sortie d'un grand nombre de sources identiques.
Elle mesure également le nombre de bits moyen nécessaires pour coder chaque symbole de la source. C'est une limite théorique. Nous verrons plus tard comment en pratique approcher cette limite. Les différents algorithmes de codage donnent plus ou moins satisfaction.

Exemple 1.3. Entropie d'une source binaire. Soit une source binaire émettant des symboles 0 avec la probabilité p et 1 avec la probabilité $q = 1 - p$. Alors l'entropie de cette source est :

$$H_2(p) = -p\log_2(p) - (1-p)\log_2(1-p)$$

Lorsque, par exemple, les 2 symboles sont équiprobables, on a $p = 1/2$ et alors

$$H_2(1/2) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = \log_2(2) = 1 \text{ } bit / symbole$$

Nous pouvons interpréter ce résultat ainsi :

Lorsque les symboles d'une source binaire sont équiprobables, il faut un bit par symbole en moyenne pour coder un symbole.

Il s'agit de la *valeur maximale* possible *de l'entropie* d'une source binaire car *l'incertitude est maximale* (**équiprobabilité des symboles**). Nous allons dans la suite utiliser cette fonction pour introduire ou illustrer les propriétés fondamentales de l'entropie.

Proposition 1.2. Propriétés de l'entropie d'une source binaire.
Soit $H_2(p), \quad p \in [0,1]$, l'entropie d'une source binaire définie dans l'exemple ci-dessous. En tant que fonction de p elle a les propriétés suivantes :

1. $H_2(p)$ est une *fonction continue* sur $]0,1[$ telle que $\lim_{p \to 0^+} H_2(p) = 0 = \lim_{p \to 1^-} H_2(p)$

2. $H_2(p)$ est *positive* sur $p \in [0,1]$

3. $H_2(p)$ est *symétrique* par rapport à $p_0 = 0.5$ et atteint son maximum en p_0 tel que $H_2(p_0) = 1$.

4. $H_2(p)$ est *strictement concave* :

$$\forall (p_1, p_2) \in [0,1],\ p_1 \neq p_2,\ \forall \lambda \in]0,1[\quad H_2(\lambda p_1 + (1-\lambda)p_2) > \lambda H_2(p_1) + (1-\lambda)H_2(p_2)$$

Toutes ces propriétés sont faciles à vérifier avec les méthodes simples de l'analyse des fonctions d'une variable réelle. Elles sont résumées par la représentation graphique de $H_2(p)$ donnée sur la *figure* 1.1.

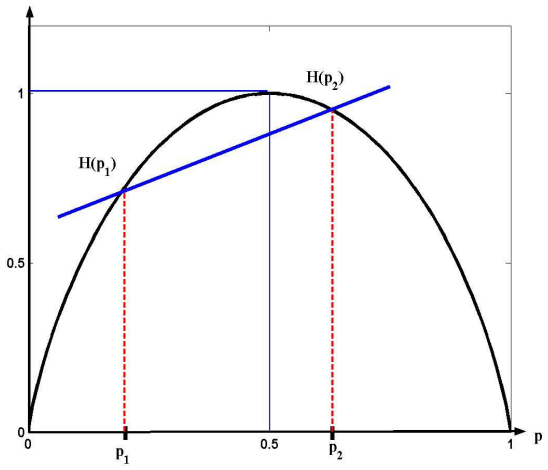


Figure 1.1. *Fonction d'entropie d'une source binaire :* $H_2(p)$ *y est plus simplement notée* $H(p)$

6.1. Interprétation de la fonction d'entropie

Considérons l'exemple d'une source S dont l'alphabet est composé de 5 symboles $\Omega = \{a, b, c, d, e\}$ et dont la distribution de probabilité est la suivante :

X	a	b	c	d	e
$P(X)$	0.3	0.2	0.2	0.15	0.15

L'entropie de cette source, calculée selon la définition est :

$$H(X) = -\sum_{i=1}^{5} p_i \log_2(p_i) = -0.3\log_2(0.3) - 2\cdot 0.2\log_2(0.2) - 2\cdot 0.15\log_2(0.15) \approx 2.27\, bits/symbole$$

Entropie comme mesure d'information. Nous avons déjà mentionné que l'entropie représente la quantité moyenne d'information apportée par l'observation des symboles de la source. On peut également l'interpréter comme la difficulté moyenne de prédire chaque symbole à la sortie : *l'entropie, ou quantité d'information moyenne, révèle le degré d'incertitude, d'aléa, quant à la prédiction des symboles émis par la source.*

Entropie comme mesure du nombre de bits de codage. Imaginons que l'on essaie de deviner le symbole observé à la sortie de la source. Pour ce faire on peut poser des questions à un automate qui ne répond que par **"oui"** ou par **"non"**. Voici un schéma représentant le déroulement du jeu :

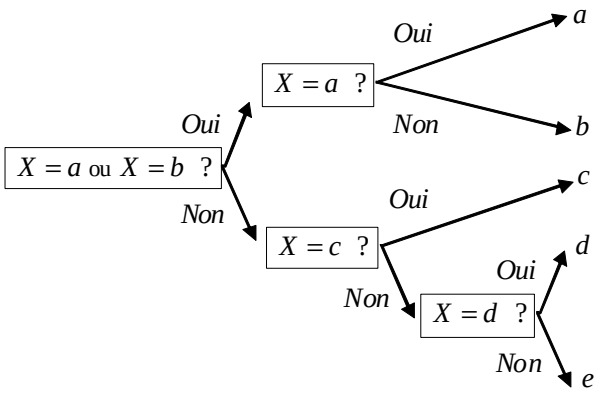


Figure 1.2. *Questions*

Il est facile de voir que le nombre de questions nécessaire pour deviner le symbole est une variable aléatoire N définie sur Ω de la façon suivante :

X	a	b	c	d	e
$P(X)$	0.3	0.2	0.2	0.15	0.15
N	2	2	2	3	3

Alors le nombre moyen de questions $\overline{N}$ (encore noté $< N >$) nécessaires pour deviner le symbole est :

$$\overline{N} = 2\cdot(0.3 + 0.2 + 0.2) + 3\cdot(0.15 + 0.15) = 2.3$$

Si l'on utilise 1 bit pour coder la réponse à chaque question (arbitrairement, bit 1 pour *Oui*, bit 0 pour *Non*), cela nous permet de coder les symboles de cette source :

a	*Oui – Oui*	11
b	*Oui – Non*	10
c	*Non – Oui*	01
d	*Non – Non – Oui*	001
e	*Non – Non – Non*	000

Nous étudierons plus loin l'un des théorèmes fondamentaux de la théorie de l'information établissant que l'entropie d'une source est le plus petit nombre moyen de bits par symbole nécessaire pour coder les messages produits par cette source. Dans cet exemple, nous avons bien : $[\overline{N} = 2.3] > [2.27 = H(X)]$.

6.2. Propriétés générales de la fonction d'entropie

Proposition 1.3. Positivité. Soit S une source sans mémoire et stationnaire d'alphabet $\Omega = \{\omega_i\}_{i=1}^n$. Soit X la variable aléatoire associée de distribution de probabilité P donnée $P[X = \omega_i] = p_i$, $i = 1, \cdots, n$. Notons $H(p_1, p_2, \cdots, p_n)$ sa fonction d'entropie. Alors

$$H(p_1, p_2, \cdots, p_n) \geq 0 .$$

En plus, l'égalité a lieu uniquement si l'une des probabilités p_i est égale à 1 et les autres nulles.

Preuve de la Proposition 1.3.

Sachant que : $\forall i = 1, \cdots, n \quad 0 \leq p_i \leq 1$, on a : $-p_i \log_2(p_i) \geq 0$.

Si l'un des symboles de l'alphabet est certain ($p_i = 1$) alors tous les autres sont forcément impossibles : ($\forall k \neq i, \ p_k = 0$).

On a alors : $-p_i \log_2(p_i) = 0$ et $\forall k \neq i, \ p_k \log_2(p_k) = 0$ (par continuité) et donc $H = 0$. (Rappel : $\log_2(1) = 0$)

Lemme 1.1. Inégalité de Gibbs. Soient $P = (p_i)_{i=1}^n$ et $Q = (q_i)_{i=1}^n$ deux distributions de probabilités sur le même univers fini $\Omega = \{\omega_i\}_{i=1}^n$. Alors

$$\sum_{i=1}^{n} p_i \log_2\left(\frac{p_i}{q_i}\right) \geq 0$$

et *l'égalité a lieu si et seulement si* $\forall i = 1, \cdots, n \quad p_i = q_i$.

Théorème 1.1. Maximum de la fonction d'entropie.

$$H(p_1, p_2, \cdots, p_n) \leq \log_2(n)$$

et *l'égalité a lieu si et seulement si* $\forall i = 1, \cdots, n \quad p_i = \frac{1}{n}$.

Preuve du Théorème 1.1.

Soient $P = (p_i)_{i=1 \cdots n}$, une distribution de probabilité sur l'univers fini $\Omega = \{\omega_i\}_{i=1}^n$, associée à une source et $Q = \left(\frac{1}{n}\right)_{i=1 \cdots n}$, la distribution uniforme sur le même univers. Appliquons l'inégalité de Gibbs à P et Q :

$$\sum_{i=1}^{n} p_i \log_2\left(\frac{p_i}{q_i}\right) \geq 0 \ \Leftrightarrow \ \sum_{i=1}^{n} p_i \log_2(np_i) \geq 0$$

d'où

$$0 \leq \sum_{i=1}^{n} p_i(\log_2 n + \log_2 p_i) \ \Rightarrow \ \log_2 n \cdot \sum_{i=1}^{n} p_i \geq -\sum_{i=1}^{n} p_i \log_2 p_i \ \Rightarrow \ \log_2 n \geq H(p_1, p_2, \cdots, p_n)$$

du fait que $\sum_{i=1}^{n} p_i = 1$.

D'après le lemme précédent, *l'égalité a lieu si et seulement si* $\forall i = 1, \cdots, n \quad p_i = q_i = \frac{1}{n}$.

Remarque 1.3. Nous pouvons interpréter ce résultat en disant que l'incertitude sur le symbole observé à la sortie d'une source est maximale lorsque tous les symboles sont équiprobables. Nous retrouvons ici le cas particulier de l'entropie d'une source binaire $H_2(p)$ étudiée dans la proposition **1.2**.

Proposition 1.4. Concavité. Soient $P=(p_i)_{i=1}^n$ et $Q=(q_i)_{i=1}^n$ deux distributions de probabilités sur le même univers fini $\Omega=\{\omega_i\}_{i=1}^n$ associé à une source S. Alors $\forall\lambda\in[0,1]$

$$H(\lambda P+(1-\lambda)Q)\geq\lambda H(P)+(1-\lambda)H(Q)$$

ou encore

$$H(\lambda p_1+(1-\lambda)q_1,\cdots,\lambda p_n+(1-\lambda)q_n)\geq\lambda H(p_1,\cdots,p_n)+(1-\lambda)H(q_1,\cdots,q_n)$$

De plus, l'égalité a lieu si et seulement si

$$P=Q$$

Remarque 1.4. Nous pouvons interpréter cette propriété comme suit : *l'entropie d'une moyenne de deux sources* est *supérieure* à la *moyenne de leur entropie.*

Exemple 1.4. Considérons trois sources binaires S_1, S_2 et S_3 sur l'alphabet $\Omega=\{a,b\}$. Supposons que les distributions de probabilités de S_1 et S_2 sont les suivantes :

$$P_1(a)=p_1=\frac{1}{3},\qquad P_1(b)=1-p_1=\frac{2}{3}$$
$$P_2(a)=p_2=\frac{1}{2},\qquad P_2(b)=1-p_2=\frac{1}{2}$$

Supposons que la distribution de S_3 soit la *moyenne* des deux premières :

$$P_3(a)=p_3=\frac{p_1+p_2}{2}=\frac{5}{12},\qquad P_3(b)=1-p_3=\frac{7}{12}$$

Calculons les entropies des trois sources :

$$H(S_1)=H_2(p_1)=-p_1\log_2(p_1)-(1-p_1)\log_2(1-p_1)=-\frac{2}{3}+\log_2(3)\approx 0.92\ \textit{bit/symbole}\qquad \textit{Rappel:}\ \log_2(2)=1$$
$$H(S_2)=H_2(p_2)=-p_2\log_2(p_2)-(1-p_2)\log_2(1-p_2)=1\ \textit{bit/symbole}$$
$$H(S_3)=H_2(p_3)=-p_3\log_2(p_3)-(1-p_3)\log_2(1-p_3)\approx 0.98\ \textit{bit/symbole}$$

On constate bien que l'entropie de la troisième source est supérieure à la moyenne des entropies des deux premières :

$$\left(H(S_3)\approx 0.98\,bit/symbole\right)\ >\ \left(\frac{H(S_1)+H(S_2)}{2}\approx 0.96\,bit/symbole\right)$$

———————

TD 1. Modélisation mathématique d'une source d'information

Partie I. Entropie d'une source. Définitions et propriétés.

Exercice 1. Entropie. Soit une source d'alphabet $\Omega = \{1,2,3,4,5\}$. Calculer son *entropie* pour les distributions de probabilités suivantes.

1. $P_1 = \{0.2, 0.2, 0.2, 0.2, 0.2\}$ *distribution uniforme*
2. $P_2 = \{0.05, 0.05, 0.05, 0.05, 0.8\}$
3. $P_3 = \{0.1, 0.2, 0.3, 0.15, 0.25\}$

Exercice 2. Information moyenne.

1. On lance une pièce dont les deux cotés sont identiques : *pile*. Quelle est *l'entropie* associée à cette expérience ?
2. On lance un dé équilibré à 6 faces. Quelle est *l'information moyenne* apportée par l'observation de la *parité* du résultat ?
3. Un jeu de cartes contient 3 piques, 4 trèfles, 2 cœurs et 1 carreau. On tire une carte au hasard. Quelle est *l'entropie* de l'observation de la *couleur* (pique ♠ , trèfle ♣ , cœur ♥ ou carreau ♦) de la carte ?

Exercice 3. Propriété de groupe. L'objectif de cet exercice est de vérifier sur un exemple, une propriété importante de la fonction d'entropie. Soit X une source d'alphabet $\Omega_X = \{x_1, \cdots, x_n\}$ et de distribution de probabilités $P_X = \{p_1, \cdots, p_n\}$. Soit $1 \le r < n$. On divise l'alphabet en deux sous-ensembles $A = \{x_1, \cdots, x_r\}$ et $B = \{x_{r+1}, \cdots, x_n\}$ de telle sorte que $\Omega_X = A \cup B$ et $A \cap B = \varnothing$ (A et B forment une partition de Ω_X). Soit $p = P(A)$ et $q = 1 - p = P(B)$. Alors on a la relation suivante :

$$H(X) = H(p_1, \cdots, p_r, p_{r+1}, \cdots, p_n) = H(p, 1-p) + pH\left(\frac{p_1}{p}, \cdots, \frac{p_r}{p}\right) + (1-p)H\left(\frac{p_{r+1}}{1-p}, \cdots, \frac{p_n}{1-p}\right) = H(p,q) + pH_A + qH_B$$

On remarquera que $P_A = \left\{\frac{p_1}{p}, \cdots, \frac{p_r}{p}\right\}$ et $Q_B = \left\{\frac{p_{r+1}}{1-p}, \cdots, \frac{p_n}{1-p}\right\}$ définissent des distributions de probabilités respectivement sur *A* et *B*.

On peut interpréter ainsi cette propriété : L'incertitude moyenne (*l'entropie*) sur X est composée de :
1. l'incertitude sur le choix de l'une des 2 parties A et B : c'est $H(p,q)$
2. la moyenne des incertitudes associées à chacune des 2 parties séparément : c'est $pH_A + qH_B$

Soit X une source d'alphabet $\Omega = \{1,2,3,4,5\}$ de distribution de probabilité $P = \{0.1, 0.2, 0.3, 0.15, 0.25\}$. Soient $A = \{1,3,5\}$ et $B = \{2,4\}$.

1. Calculer *l'entropie* de X.
2. Calculer $p = P(A)$ et $q = P(B)$. En déduire $H(p,q)$.
3. Définir les *distributions de probabilités* sur A et sur B comme indiqué ci-dessus et les *entropies* associées. Vérifier ensuite la *propriété de groupe*.

Exercice 4. Vers le codage de Shannon. Soit la source X d'alphabet $\Omega = \{1, 2, 3, 4, 5\}$ de distribution de probabilité $P = \{0.1, 0.2, 0.3, 0.15, 0.25\}$. Supposons que l'on souhaite deviner le symbole émis par la source. On a le droit de poser des questions binaires (réponses possibles "oui" et "non"). On cherche à construire la stratégie qui, en moyenne, permet de trouver la réponse en un *nombre minimal de questions*.

Remarquons qu'une question binaire induit sur l'ensemble Ω, une partition en deux sous-ensembles A et B correspondant aux réponses "oui" et "non". Par exemple, si l'on demande "est-ce que le chiffre est impair ?", la partition qui en découle sera $A = \{1, 3, 5\}$ et $B = \{2, 4\}$. Soit $p = P(A)$ la probabilité de la réponse "oui" à une question donnée.

1. Calculer *l'entropie* de X.
2. Quelle est *l'information moyenne* obtenue par la réponse à une question binaire ?
3. Quelle est *l'information moyenne maximale* ? Et pour quelle valeur de p est-elle atteinte ?
4. Quel est alors le *meilleur choix de première question* à poser ?
Indication : il faut minimiser l'incertitude, ne pas poser de question dont on connaît la réponse et n'apportant donc pas d'information.
5. Appliquer le même raisonnement *récursivement* pour choisir la meilleure deuxième question selon la réponse à la première. Continuer jusqu'à l'identification de chaque symbole. Construire un *arbre* représentant la stratégie obtenue.
6. Si on associe *arbitrairement* le bit 1 à la réponse "oui" et le bit 0 à "non", cet arbre définit un *code*. Donner la *table de code* associée.
7. Calculer le *nombre moyen de questions* (la *longueur moyenne des mots de code*). Comparer à l'*entropie* de X.

———————

2. Cryptographie

Ce chapitre n'a pas pour ambition de faire un exposé complet sur la cryptographie. Nous allons ici montrer en quoi les principaux concepts de la théorie de l'information que nous venons d'introduire sont utiles dans ce domaine.
En 1949 C. Shannon a publié un article intitulé *"Communication Theory of Secrecy Systems"* dans la revue Bell System Technical Journal. Dans cet ouvrage C. Shannon a montré comment mesurer, à l'aide de la notion d'entropie, la sécurité d'un système de communications crypté. Il a introduit la notion de *cryptosystème parfaitement sûr* et a mis en évidence les faiblesses d'un tel système dues aux difficultés de sa mise en pratique. Nous suivrons son raisonnement pour arriver à montrer l'importance du concept d'entropie dans ce domaine.
Pour cela nous commencerons par une brève présentation des notions élémentaires de la cryptographie. Ensuite, nous allons exposer la théorie de C. Shannon sur la sécurité d'un cryptosystème.

1. Quelques définitions

La *cryptographie* est une science qui étudie les méthodes permettant de transmettre des messages de façon confidentielle.

On va appeler **message clair** un ensemble de données (texte, image,...) que l'on souhaite transmettre.

Le **chiffrement** (ou **cryptage**, ou encore **codage** ou aussi **chiffrage**) est un procédé permettant de transformer le message clair de telle sorte qu'il soit incompréhensible par qui que ce soit d'autre que l'auteur du message et le destinataire.

Le **chiffré** (ou **crypté**) est le résultat du chiffrement.

Le **déchiffrement** (ou **décryptage** ou encore **décodage** ou aussi **déchiffrage**) est le procédé permettant de retrouver le message clair à partir du chiffré.

La **clé** de chiffrement est un paramètre qui est utilisé dans le procédé de chiffrement ou de déchiffrement.

Un **cryptosystème** est un ensemble de clés K, de messages clairs possibles M, de messages chiffrés possibles C associé à un algorithme de chiffrement, représenté par une fonction $\mathrm{E}:\mathrm{K}\times\mathrm{M}\to\mathrm{C}$, et un procédé de déchiffrement, représenté par une fonction $\mathrm{D}:\mathrm{K}\times\mathrm{C}\to\mathrm{M}$. On suppose que pour tout $m\in\mathrm{M}$, il existe une paire de clés de chiffrement et de déchiffrement telles que la relation

$$\mathrm{D}(k_D,\mathrm{E}(k_E,m))=m$$

est assurée. On note $(\mathrm{K},\mathrm{M},\mathrm{C},\mathrm{E},\mathrm{D})$ un cryptosystème.

Remarque 2.1. Principe de Kerckhoffs. De nos jours, la plus grande partie des méthodes de cryptographie reposent sur ce principe, énoncé par Auguste Kerckhoffs à la fin XIXe siècle (publié dans un article au Journal des sciences militaires, vol. IX, pp. 5-38, Janvier 1883, pp. 161-191, Février 1883. *Source : Wikipédia*).

La sécurité d'un cryptosystème ne doit pas reposer sur la non divulgation de la fonction de cryptage mais uniquement sur la non divulgation de la clé.

Un autre énoncé de ce principe, connu sous le nom de *maxime de Shannon* a été proposé par Claude Shannon, au milieu du XXe sciècle : *L'adversaire connaît le système*.

Deux modèles principaux de chiffrement existent actuellement :

Chiffrement symétrique

On appelle aussi ce modèle "chiffrement à clé secrète". C'est l'approche classique. La même clé est utilisée pour chiffrer et déchiffrer un message. Cette clé doit alors rester secrète. Dans les systèmes de communication à grande échelle cela représente la principale faiblesse de ces chiffrements. En effet la clé doit être communiquée au destinataire et dans de nombreux cas elle est envoyée par le même canal de transmission que le message lui même. Si elle est interceptée à ce moment là, toute la communication est découverte !

Chiffrement asymétrique

On l'appelle aussi chiffrement à clé publique. La clé utilisée pour le chiffrement est publique et elle est différente de la clé de déchiffrement, qui est secrète. Ainsi le destinataire du message choisit la clé de déchiffrement et la clé de chiffrement associée. Il peut envoyer à l'émetteur la clé de chiffrement (publique). Et la clé de déchiffrement, secrète, ne circule pas dans les canaux de transmission.

1.1. Quelques exemples élémentaires

1.1.1. Chiffrement de César (50 av. J.C.)

Le chiffre de César utilisait le principe de décalage cyclique de l'alphabet. Soit en effet l'alphabet latin A de 26 lettres. Un message $m = m_1 m_2 \cdots m_n$ est chiffré en remplaçant chaque lettre m_i par la lettre $\sigma(m_i)$ obtenue par décalage cyclique par exemple de 3 positions dans l'alphabet. Par exemple, la lettre "a" sera remplacée par la lettre "d", la lettre "z" par "c", … On peut généraliser ce principe avec un décalage de k caractères, $k = 1, \cdots, 26$. Le paramètre k de décalage est alors la clé secrète. Ce système est facile à casser (en force brute, i.e. on essaie toutes les clés possibles) car l'espace des clés est très réduit : ici il n'y a que 26 clés possibles. Il suffit alors de tester toutes les 26 possibilités jusqu'à trouver un texte intelligible.

Il est possible de rendre le chiffrage par substitution plus complexe. Au lieu d'utiliser les décalages cycliques, on peut utiliser une permutation quelconque $\sigma : \mathrm{A} \to \mathrm{A}$ des lettres du message. Alors la fonction de chiffrement se présente de la façon suivante. Soit un message $m = m_1 m_2 \cdots m_n$

$$\mathrm{E}(m) = \mathrm{E}(\sigma, m) = \sigma(m_1)\sigma(m_2)\cdots\sigma(m_n)$$

Exemple : avec $\sigma = 2$, le message clair « robert » donne le chiffré (anagramme) « bertro » (la permutation se fait avec une lettre décalée de σ positions vers la droite).

La clé secrète est alors la permutation σ et l'ensemble K de toutes les clés est celui de toutes les permutations possibles sur l'alphabet de 26 caractères (périodicité des permutations). On a donc $Card(\mathrm{K}) = 26\,! \approx 4 \cdot 10^{26}$. Il est donc raisonnable de supposer qu'une attaque par force brute (qui consiste à tester toutes les clés possibles) ne soit pas réalisable (temps calcul rédhibitoire).

Il est néanmoins possible de casser un tel code par analyse statistique. En effet, si l'on connaît la langue du message clair, on connaît la table de fréquences des caractères. Par exemple, en français, la lettre "e" est la plus fréquente. On peut également utiliser les tables de fréquences d'ordres supérieurs qui établissent les probabilités d'occurrence de couples de caractères. En français par exemple, le caractère "u" a 95% de chances d'être trouvé après "q". Alors, en analysant le chiffré seul et en calculant les fréquences d'occurrence des différents caractères dans le chiffré on peut retrouver la permutation et donc le texte clair. C'est un exemple d'attaque à chiffré seul.

1.1.2. Substitution polyalphabétique

Une variante de la méthode de substitution consiste à utiliser un nombre fini de permutations au lieu d'une seule. On découpe alors le texte clair en blocs de p caractères et on choisit p permutations d'alphabet : $\sigma_1\sigma_2 \cdots \sigma_p$. Alors pour chaque bloc de p caractères de texte on applique la transformation

$$\mathrm{E}(m_1 m_2 \cdots m_p) = \sigma_1(m_1)\sigma_2(m_2)\cdots\sigma_p(m_p)$$

Par exemple, le chiffrement de la machine *Enigma* utilisée pendant la 2nde guerre mondiale était un chiffrement de ce type avec des décalages cycliques pour chaque permutation. La clé était alors une suite de nombres dont la longueur indiquait la longueur des blocs et chaque nombre représentait le décalage à appliquer au caractère correspondant du bloc.
Ce chiffrement, bien que plus complexe que les précédents peut aussi être cassé par les méthodes statistiques basées sur l'étude des fréquences d'apparition des caractères.

1.1.3. Chiffrement de Vernam

Cette méthode a été proposée en 1917. Elle a été longtemps utilisée pour le chiffrement des messages télégraphiques. Les messages clairs sont des suites de bits de longueur n. L'espace des messages, de taille 2^n, est alors $\mathrm{M} = \{0,1\}^n$. Les clés sont les suites binaires de même longueur que les messages : $\mathrm{K} = \mathrm{M} = \{0,1\}^n$. La fonction de chiffrage consiste à ajouter, bit à bit modulo 2, la clé au message clair (opérateur ou exclusif $\oplus$). Pour tout message $m = m_1 m_2 \cdots m_n$ et pour toute clé $k = k_1 k_2 \cdots k_n$, on obtient le chiffré c :

$$c = \mathrm{E}(k,n) = m \oplus k = m_1 \oplus k_1, m_2 \oplus k_2, \cdots m_n \oplus k_n$$

Nous allons expliquer dans la section suivante pourquoi ce chiffrement est *parfaitement sûr* au sens de Shannon. Remarquons pour l'instant ses faiblesses (sa principale qualité résidant dans sa simplicité due au fait qu'il s'agit d'un code à clé secrète – clé identique pour le chiffrement et le déchiffrement). Ce chiffrement est vulnérable devant une attaque à clair connu : si quelqu'un découvre ne serait-ce qu'un bloc de message clair m, il peut immédiatement obtenir la clé en la calculant :

$$k = m \oplus c$$

du fait de la propriété de l'opérateur ou exclusif : $c = m \oplus k$ et $c \oplus k = m \oplus k \oplus k = m \oplus \mathbf{0} = m$

Il est alors indispensable de changer de clé à chaque bloc. C'est pour cela qu'on appelle cette méthode "masque jetable". Si l'on prend en compte le fait que les clés sont de même longueur que les blocs et qu'il faut les transmettre par un canal sécurisé avant de transmettre le chiffré il devient évident que la mise en pratique de cette méthode est difficile. Ce code a tout de même été utilisé pour le téléphone rouge. La clé alors été envoyée par porteur car il était préférable d'envoyer une nouvelle clé si elle a été interceptée que de courir le risque que le message secret soit intercepté et décodé.

2. La théorie de C. Shannon sur la sécurité entropique

Nous allons maintenant nous intéresser à la signification que prend le concept d'entropie dans le contexte de cryptographie.
Rappelons que l'entropie d'une variable aléatoire mesure l'incertitude associée au résultat d'une expérience que la variable représente. Nous allons donc donner une interprétation probabiliste au problème de chiffrement en nous plaçant à la place du cryptanalyste, celui qui cherche par analyse à retrouver le message clair à partir du chiffré. De son point de vue, le choix d'un message clair et celui d'une clé sont des résultats d'un tirage aléatoire. Le message chiffré est alors aussi aléatoire. Soient donc les variables aléatoires M, C et K représentant respectivement le choix d'un message clair dans l'ensemble M, le choix d'un chiffré dans l'ensemble C et celui d'une clé dans l'ensemble K.
C. Shannon définit un cryptosystème parfaitement sûr comme étant un système dans lequel la connaissance du chiffré n'apporte pas d'information sur le message clair.

Définition 2.1. Cryptosystème parfaitement sûr. Soit un cryptosystème $(\mathrm{K}, \mathrm{M}, \mathrm{C}, \mathrm{E}, \mathrm{D})$. Soient M et C les variables aléatoires représentant le choix d'un message clair et d'un chiffré. Le système est dit parfaitement sûr si et seulement si

$$H(M|C) = H(M)$$

Proposition 2.1. La sécurité parfaite du chiffre de Vernam. Supposons que l'espace des clés est le même que l'espace des messages clairs : $\mathrm{M} = \mathrm{K}$ et que toutes les clés sont équiprobables. Alors le chiffre de Vernam est parfaitement sûr.

Preuve de la Proposition 2.1.

Nous allons supposer sans perte de généralité que l'alphabet utilisé est binaire et que les messages et les clés sont des séquences de bits de longueur connue n. Alors l'espace des messages clairs et des clés est

$$\mathrm{M} = \mathrm{K} = \{0,1\}^n$$

Comme toutes les clés sont supposées équiprobables, on a :

$$\forall k \in \mathrm{K}, \quad P[K = k] = \frac{1}{2^n} \qquad \text{car il y a } 2^n \text{ clés possibles}$$

Pour établir l'égalité des entropies, il suffit de montrer l'égalité des probabilités : (*Rappel :* $h(A) = -\log_2(P[A])$)

$$\forall m \in \mathrm{M}, \quad \forall c \in \mathrm{C}, \quad P[M = m | C = c] = P[M = m]$$

Par définition de probabilité conditionnelle on a :

$$P[M = m | C = c] = \frac{P[M = m, C = c]}{P[C = c]}$$

On sait que dans le chiffrement de Vernam les 3 variables K, M et C sont liées : $C = K \oplus M$. Alors :

$$C = c \wedge (M = m) \Leftrightarrow K = (c - m) \wedge (M = m) \quad \text{en fait} \quad k = (c - m) \text{ modulo } 2 \quad \text{car} \quad k = 0 \text{ ou } 1$$

Donc on a

$$P[M = m, C = c] = P[M = m, K = c - m] = P[M = m]P[K = c - m]$$

car les variables aléatoires K et M sont indépendantes. De plus :

$$P[K = c - m] = \frac{1}{2^n} \rightarrow P[M = m, C = c] = \frac{P[M = m]}{2^n}$$

On calcule maintenant la probabilité

$$P[C = c] = \sum_{x \in \mathrm{M}} P[C = c, M = x] = \sum_{x \in \mathrm{M}} \frac{P[M = x]}{2^n} = \frac{1}{2^n}$$

On a donc bien $P[M | C] = P[M]$

Le théorème suivant permet de montrer qu'un système parfaitement sûr doit avoir des clés au moins aussi longues que les messages clairs.

Théorème 2.1. Dans un système cryptographique parfaitement sûr on a

$$H(K) \geq H(M)$$

En particulier, si tous les messages et toutes les clés sont équiprobables, les clés sont de longueur au moins égale à celle des messages *(il doit y avoir plus d'information dans les clés que dans les messages).*

Il n'est donc pas possible de trouver un chiffrement parfaitement sûr ayant des clés moins longues que celles du chiffre de Vernam (pour lequel tous les messages de même taille sont équiprobables, et toutes les clés de même taille le sont aussi).

Dans ses travaux C. Shannon s'est aussi intéressé aux systèmes à clé courte, qui ne sont pas parfaits. La question que l'on peut alors poser est la suivante. Etant donné un système cryptographique dans lequel toutes les clés et tous les messages clairs sont équiprobables, de combien de chiffrés faut il disposer pour être capable de retrouver de façon unique la clé ? Pour répondre à la question nous allons introduire quelques notions supplémentaires.

Définition 2.2. Soit un langage composé de mots de longueur donnée N sur un alphabet donné A. On associe la variable aléatoire M au choix au hasard d'un mot du langage. On appelle taux du langage ou taux d'entropie, la quantité

$$r = \lim_{N \to \infty} \frac{H(M)}{N}$$

Si le nombre de caractères de l'alphabet est L on appelle taux maximal du langage la quantité

$$R = \log_2 L$$

Enfin on appelle redondance du langage la différence

$$D = R - r$$

Dans cette définition, le taux du langage représente la quantité moyenne d'information par caractère de message, le taux maximal correspond à l'entropie maximale d'un caractère, c'est à dire au cas où tous les messages sont équiprobables. Enfin, la redondance représente la différence entre le nombre de bits maximal et ce qui est tout juste nécessaire pour coder un caractère. Par exemple, le taux de l'anglais a été évalué par des méthodes statistiques à près de 1.5.
Nous allons maintenant introduire la mesure permettant de quantifier la quantité d'information dont il est faut disposer pour pouvoir déterminer la clé d'un chiffrement.

Définition 2.3. Distance d'unicité. Soit un cryptosystème $(\mathrm{K}, \mathrm{M}, \mathrm{C}, \mathrm{E}, \mathrm{D})$. On appelle distance d'unicité le plus petit nombre de messages chiffrés qu'il est nécessaire de disposer pour que l'incertitude résiduelle sur la clé sachant ces chiffrés soit nulle :

$$n_0 : \quad H\left(K \middle| C_1, \cdots, C_{n_0}\right) = 0$$

On peut montrer la proposition suivante.

Proposition 2.2. La distance d'unicité d d'un cryptosystème est égale à

$$d = \frac{H(K)}{D} = \frac{H(K)}{R - r}$$

où D est la redondance du langage.

Exemple 2.1. On a établi par analyse statistique que le taux d'entropie du français est d'environ $r \approx 3.97$ et que le taux maximal est $R = 4.67$. Ainsi la redondance est $D = 0.7$. On peut donc calculer la distance d'unicité pour un chiffrement par substitution. L'espace des clés est alors l'ensemble de toutes les permutations possibles de l'alphabet latin et, si on considère que les clés sont équiprobables, on a :

$$d = \frac{H(K)}{D} = \frac{\log_2(26!)}{0.7} \approx 126$$

Il faut donc en moyenne 126 caractères de texte chiffré pour être en mesure de déterminer la clé de façon unique.

Remarque 2.2. Il s'agit bien d'une condition nécessaire à la possibilité de découvrir la clé. Il faut interpréter ce résultat dans le sens suivant : si l'on ne dispose pas de longueur suffisante de message chiffré donnée par la distance d'unicité, il est impossible de déterminer la clé avec certitude.
Ce résultat ne permet surtout pas de se prononcer sur la puissance de calcul nécessaire pour découvrir la clé ni sur les moyens d'y parvenir.

Remarque 2.3. Il est important d'observer que la distance d'unicité est inversement proportionnelle à la redondance des messages clairs. Plus l'ensemble des messages clairs est redondant moins il faut d'informations pour déterminer la clé. Ainsi la sécurité du système dépend aussi de la qualité intrinsèque des messages.
On en déduit que, pour améliorer la distance d'unicité, il est préférable de traiter en amont les messages de façon à réduire la redondance. Par exemple, une compression sans pertes par codage de *Huffman* ou autre peut être utilisée avant le chiffrement. Nous allons aborder ces techniques dans les chapitres suivants.

———————

TD 2. Cryptographie

Partie I. Entropie : un jeu d'espion !

Le chiffrement de César consiste à décaler l'alphabet de k positions de façon cyclique et remplacer chaque lettre d'un message clair par une lettre correspondante de l'alphabet décalé. La clé secrète de ce chiffre est un entier $1 \le k \le 26$ qui représente le décalage de l'alphabet. Nous allons dans un premier temps étudier la sécurité de ce chiffre, et ensuite mettre en pratique la méthode d'analyse des fréquences qui a permis de le casser.

Exercice 1. Rendons à César ce qui est à César : son chiffre !

1. Montrer que pour les messages de longueur $l = 1$, le chiffre de César est parfaitement sûr au sens de Shannon. Pour cela montrez que :

$$\forall m \in \mathrm{M}, \quad \forall c \in \mathrm{C}, \quad P[M = m | C = c] = P[M = m]$$

2. Montrer que pour les messages de longueur $l \ge 2$, le chiffre n'est plus parfaitement sûr. Pour cela analyser l'exemple suivant. Soient le message $m = AB$ et le chiffré $c = DM$. Montrer que $P[M = m | C = c] = 0$ tandis que $P[M = m] \neq 0$.

Exercice 2. Cryptanalyse du chiffre de César.

La méthode d'analyse des fréquences a été inventée par le savant AL-Kindi au IX-ème siècle. On suppose que l'on connaît la langue du texte clair et que l'on dispose du message chiffré. Dans le cas du chiffre de César, on cherche à déterminer le paramètre k, clé du chiffre. La méthode consiste à comparer l'histogramme d'occurrences des caractères du chiffré avec la table des fréquences d'occurrence des caractères de la langue du texte clair. Voici la table des fréquences de la langue française.

Lettre	*Fréquence (%)*
A	8.4
B	1.06
C	3.03
D	4.18
E	17.26
F	1.12
G	1.27
H	0.92
I	7.34
J	0.31
K	0.05
L	6.01
M	2.96

Lettre	*Fréquence (%)*
N	7.13
O	5.26
P	3.01
Q	0.99
R	6.55
S	8.08
T	7.07
U	5.74
V	1.32
W	0.04
X	0.45
Y	0.3
Z	0.12

1. Le travail à réaliser consiste à déchiffrer le message issu du chiffrement du clair :

« BELLE MARQUISE VOS YEUX ME FONT MOURIR D AMOUR ».

Pour *chiffrer* ce message, utiliser l'applet Java : http://www.bibmath.net/crypto/substi/cryptcesar.php3

Pour *déchiffrer* : par exemple, sous *Word*, compter le nombre d'occurrences de chaque symbole, et sous *Excel*, tracer l'allure de la table des occurrences obtenue et comparer à la courbe issue de la table des fréquences de la langue française. En déduire la clé *k* du chiffrement : le *décalage* des 2 courbes.

2. Et maintenant, soit à déchiffrer ce message par la méthode d'*analyse des fréquences* :

UHGCHNK. GHNL OHNL IKHIHLHGL MKHBL PTZHGL IHNK ITKMBK TN STGSBUTK. OHMKX TOBHG OT ITKMBK ET HN OHNL OHNEXS. NG OKTB OTNMHNK OXNM MHNCHNKL OHEXK ATNM IHNK OHBK LT IKHBX. NG SHFUB FTKVATBM XG SBZSTZTGM LNK ET KHNMX.

Partie II. Entropie et arbres de décision

Exercice 3. Le jeu de la fausse pièce. On considère un jeu de n pièces, toutes d'apparence identique. On sait qu'*une seule pièce est fausse*. Elle a un poids différent des autres mais on ne sait pas si elle est plus légère ou plus lourde.
On dispose d'une balance à deux plateaux. A chaque pesée, la balance peut se trouver dans l'une des trois configurations :

G Elle penche vers la gauche.
D Elle penche vers la droite.
E Elle reste à l'équilibre.

L'objectif est de déterminer avec *le moins de pesées possible*s la fausse pièce et si elle plus légère ou plus lourde.

1. Soit $n = 8$.

(*a*) Combien de réponses possibles il y a dans ce problème ?
(*b*) Combien de possibilités différentes peut-on obtenir avec 2 pesées ? Avec 3 pesées ?
Conclusion sur la faisabilité de la détermination de la fausse pièce en 2 ou en 3 pesées.
(*c*) Est-il intéressant de peser deux lots de 4 pièces chacun ? Pourquoi ? Quelle quantité d'information nous apporterait une telle pesée ?
(*d*) A l'aide du programme java à l'adresse : http://nlvm.usu.edu/fr/nav/frames_asid_139_g_4_t_2.html
élaborer une stratégie de pesée permettant de déterminer la fausse pièce en 3 pesées dans tous les cas. Voici quelques conseils qui peuvent être utiles :

i. Essayer, pour commencer, de diviser l'ensemble des pièces en trois lots, de taille à peu près égale. Pour 8 pièces, on peut tester les décompositions 8 = 3 + 3 + 2 ou 8 = 2 + 2 + 4.

ii. Lorsque l'on compare le même lot de pièces L_1 à deux lots différents L_2 et L_3, on peut interpréter les résultats de la façon suivante : si les deux résultats de comparaison sont identiques, la fausse pièce est dans le lot L_1 et nous avons le sens de la différence de poids (+ ou -); il n'est pas possible que l'un des résultats soit **G** et l'autre **D**.

(*e*) Représenter la stratégie sous forme d'arbre de décision. Les feuilles de l'arbre doivent représenter toutes les réponses possibles. La profondeur de l'arbre indique alors le nombre maximal de pesées.

2. *Facultatif.* A l'aide du même programme java, élaborer une stratégie pour $n = 9,10,12$.

3. Codage de source (Codage sans bruit)

1. Codage

D'une manière générale le codage peut être vu comme une transformation de symboles d'un alphabet donné $\Omega_1 = \{s_1, \cdots, s_n\}$ en suites de symboles d'un autre alphabet $\Omega_2 = \{c_1, \cdots, c_d\}$.

Dans un schéma de communication, la source et le canal de transmission n'utilisent pas forcément le même alphabet. Par exemple, la source peut être un texte anglais, utilisant les 26 lettres de l'alphabet latin, et le canal peut être tout support numérique, utilisant l'alphabet binaire. Il se pose donc le problème de *codage* comme "traduction" entre l'alphabet de la source et celui du canal. Si le récepteur utilise le même alphabet que la source, il est également nécessaire de *décoder* le message avant de pouvoir le lire, c'est-à-dire, appliquer la transformation inverse. Toute transformation candidate doit vérifier un certain nombre de critères que nous allons détailler plus loin.

Etant donné qu'il peut exister une grande quantité de codages possibles, il se posera la question de savoir lequel est meilleur que tous les autres. Cela dépendra d'un critère d'optimalité que l'on fixera. Nous allons dans cette perspective considérer deux types d'applications de codage :

1. Codage de source ou encore *codage sans bruit*. Nous allons supposer que la communication se fait via un canal sans bruit. Cela signifie que tout message est transmis de façon exacte. Sous cette hypothèse le meilleur code sera celui qui permettra la transmission la plus rapide possible. Le premier théorème de Shannon donne la solution à ce problème.

2. Codage de canal ou encore *codage en présence de bruit*. En supposant qu'il existe des perturbations pouvant engendrer des erreurs à la réception, on cherchera une méthode de codage permettant une transmission aussi rapide que possible tout en mini- misant la probabilité des erreurs. *Le second théorème de Shannon* donne la solution à ce problème.

2. Premier théorème fondamental

2.1. Codage de source

Nous allons dans un premier temps considérer un modèle de communication avec un canal discret sans bruit. Soient $\Omega_S = \{s_1, \cdots, s_n\}$ l'alphabet de la source et $\Omega_C = \{c_1, \cdots, c_d\}$ l'alphabet du canal.
Supposons que la distribution de probabilité de l'alphabet de la source est connue :

$$P_S : \Omega_S \to [0,1], \quad s_i \mapsto p_S(s_i) \stackrel{\Delta}{=} p_i$$

On peut alors en déduire l'entropie de la source

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Nous utiliserons dans la suite la terminologie suivante :

Lettre, **symbole** ou **caractère**	Tout élément d'un alphabet donné.
Message ou **mot**	Une séquence finie m de caractères d'un alphabet donné.
Longueur de mot	Le nombre $l(m)$ de caractères d'un mot m.

Le codage consiste à faire correspondre à chaque symbole s_i de la source, une séquence $m(s_i)$ de symboles de l'alphabet du canal, appelée **mot-code**. Une telle association peut donc être représentée par un ensemble $\{m_1, m_2, \cdots, m_n\}$ de n mots-codes correspondant chacun à un symbole de l'alphabet de la source :

$$\forall i = 1, \cdots, n \qquad m_i = m(s_i)$$

Si l'on note l_i les longueurs des mots m_i du code, et soit L la variable aléatoire dont la valeur est la longueur du mot-code associé à un symbole émis par la source S. Alors cette variable aléatoire prend les valeurs l_i avec les probabilités

$$p_i = p_S(s_i) \quad i = 1, \cdots, n$$

On définit alors la longueur moyenne du code par :

$$\overline{L} = E[L] = -\sum_{i=1}^{n} p_i l_i \qquad \text{notée encore} \qquad < L >$$

Cette quantité est importante pour l'analyse des caractéristiques d'un code donné. Elle représente le nombre moyen de caractères de l'alphabet Ω_C pour coder un symbole émis par la source S.
Supposons que le temps de transmission est le même pour tous les caractères de l'alphabet du canal. Alors, le temps moyen de transmission d'un symbole de l'alphabet de la source est proportionnel à la longueur moyenne $\overline{L}$ des mots du code. C'est pour cette raison que nous allons dans la suite étudier en détail ce paramètre. Un code doit vérifier un certain nombre de propriétés garantissant la possibilité de reconstituer tout message codé à la réception. Ces propriétés sont les suivantes :

Régularité. Un code $\{m_1, m_2, \cdots, m_n\}$ est dit *régulier* si tous les mots qui le composent sont distincts :

$$m_i \neq m_k \quad \forall i \neq k$$

Cette condition garantit au moins que tout message d'un seul caractère de l'alphabet de la source peut être *décodé*. Un code qui n'est pas *régulier* est dit **singulier** ou **irréversible**.

Déchiffrabilité. Un code *régulier* est dit *déchiffrable* (ou encore à *décodage unique*) si pour toute suite de mots de code $m^1, m^2, \cdots, m^k$, il est possible de distinguer *sans ambiguïté tous les mots* et donc *identifier les symboles* $s^j \quad j = 1, \cdots, k$ composant le message.

Voici un exemple de plusieurs codes possibles pour un même alphabet.

Exemple **3.1.** Soit l'alphabet de source $\Omega_S = \{a, b, c, d\}$ de distribution de probabilité $P_S = \{0.4, 0.3, 0.2, 0.1\}$. L'entropie de cette source est $H(S) \approx 1.85$ bits/symbole. Supposons que le canal est binaire. Le tableau ci-dessous propose quelques codes et leurs longueurs moyennes de mots :

S	*Probabilité*	**Code 1**	**Code 2**	**Code 3**	**Code 4**	**Code 5**	**Code 6**
a	0.4	1	00	0	0	0	0
b	0.3	0	01	10	01	10	11
c	0.2	1	10	01	011	110	100
d	0.1	0	11	010	0111	1110	101
	Longueur moyenne	**1**	**2**	**1.7**	**2**	**2**	**1.9**

Le **code 1** n'est pas *régulier*. En effet, le caractère 0 correspond à 2 caractères différents dans l'alphabet initial. Tous les autres codes du tableau sont *réguliers*. Le **code 2** est un code de *longueur fixe* : tous les mots du code sont de même longueur. Les **codes 3**, **4**, **5** et **6** sont de *longueur variable*.

Le **code 3** est *régulier* mais *pas déchiffrable*. En effet, la *déchiffrabilité* signifie que toute séquence de mots du code correspond à au plus 1 message. Dans le cas du **code 3**, la séquence 010 correspond à la fois à 3 messages différents : "d", "ca" et "ab". Elle ne peut donc pas être décodée correctement. Nous allons maintenant analyser le problème de la *déchiffrabilité* en détail.

2.1.1. Le problème de décodage unique

Il existe plusieurs façons de garantir qu'un code soit déchiffrable :

Codes de longueur fixe. Un code régulier de *longueur fixe* peut toujours être décodé sans ambiguïté car il suffit pour cela de découper la séquence en mots de longueur connue. Cette solution présente néanmoins un désavantage. On peut l'observer dans le tableau ci-dessus. Le **code 2**, de longueur fixe, a une longueur moyenne de code égale à 2 tandis qu'il existe dans la même table des codes avec une longueur moyenne inférieure.

Utilisation d'un séparateur. Il est possible de consacrer un symbole de l'alphabet du canal comme *séparateur* de mots du code. Par exemple, pour un canal binaire, on peut coder le i-ème symbole de la source s_i à l'aide de i caractères "1" et utiliser "0" comme séparateur. Dans le cas de l'*exemple* **3.1.** cela donnerait le code suivant :

S	*Probabilité*	**Code 7**
a	0.4	10
b	0.3	110
c	0.2	1110
d	0.1	11110
	Longueur moyenne	**3**

et la séquence "abc" donnerait "101101110".
On constate que la longueur moyenne qui tient compte du séparateur est plus élevée que pour tous les autres codes.

Codes sans préfixe (≡ instantanés). On dit qu'un mot W est un *préfixe* d'un autre mot V s'il existe un mot U tel que $V = WU$. Autrement dit, le mot V commence par le mot W . Dans ce cas, le mot U s'appelle *suffixe* de V .
On dit qu'un code donné est *sans préfixe* ou *instantané* si aucun mot du code n'est le préfixe d'un autre mot du code.

Dans l'*exemple* **3.1.** le **code 6** est sans préfixe. Un tel code est toujours déchiffrable. En effet, pour décoder une séquence quelconque de mots du code il suffit lire la séquence caractère par caractère de gauche à droite. Dès qu'un mot du code m est formé, on sait qu'il n'est pas le début d'un autre mot. On peut donc séparer (isoler) le mot et continuer ainsi la lecture. Cela donne une procédure de décodage "*pas à pas*".
Prenons une séquence W = 011010011101 du **code 6**. Le décodage de la séquence se passe de la façon suivante :

Pas 1 m^1 = "0" $\Rightarrow$ $s^1 = a$ et $W_1 =$ 11010011101.
Pas 2 m^1 = "11" $\Rightarrow$ $s^2 = b$ et $W_2 =$ 010011101.
Pas 3 m^3 = "0" $\Rightarrow$ $s^3 = a$ et $W_3 =$ 10011101.
Pas 4 m^4 = "100" $\Rightarrow$ $s^4 = c$ et $W_4 =$ 11101.
Pas 5 m^5 = "11" $\Rightarrow$ $s^5 = b$ et $W_5 =$ 101.
Pas 6 m^6 = "101" $\Rightarrow$ $s^6 = d$ et $W_6 = \varnothing$.

On obtient en symboles de l'alphabet de la source : $abacbd$.

On remarque que dans la famille de codes à longueur variable et sans séparateur, les codes sans préfixe ou instantanés représentent un intérêt. Il est évident que tout *code instantané* est *déchiffrable.*

La réciproque n'est pas vraie. Soit en effet une source binaire d'alphabet $\Omega_S = \{a, b\}$ et un canal binaire d'alphabet $\Omega_C = \{0,1\}$. Le code suivant : $m_1 = m(a) = 0$ $m_2 = m(b) = 01$ n'est pas *instantané* car m_1 est un *préfixe* de m_2. Il est tout de même *déchiffrable*.

Pour décoder une séquence de mots de ce code, il suffit de repérer d'abord les positions de "1". Chaque "1" est obligatoirement précédé de "0" et donne le caractère "b". Les autres "0" correspondent au caractère "a".
C'est comme cela qu'on trouve que la séquence "000101000100" correspond au message "$aabbaabaa$".

On voit alors la signification du terme "*instantané*" et l'intérêt principal de ces codes. Un code sans préfixe peut être décodé pas à pas, par séparation successive des mots. Cela peut être fait en temps réel, au fur et à mesure de la réception du message. Tandis qu'un code déchiffrable mais non instantané nécessite un traitement plus long et spécifique au code pour être déchiffré.

Nous allons nous intéresser maintenant au problème de l'existence des codes instantanés. Ce problème se formule de la façon suivante :

Soient l'alphabet de la source $\Omega_S = \{s_1, \cdots, s_n\}$ *de taille* n *et l'alphabet du canal* $\Omega_C = \{c_1, \cdots, c_d\}$ *de taille* d*. Etant donnés* n *nombres entiers positifs non nuls* $(l_1, l_2, \ldots, l_n)$*, existe-t-il un code régulier instantané de* n *mots* $\{m_1, \cdots, m_n\}$ *tel que chaque nombre* l_i *soit la longueur du mot de code* m_i *?*

Le théorème suivant donne la condition nécessaire et suffisante d'existence de tels codes.

Théorème 3.1. Inégalité de Kraft. Un **code instantané** (≡ **sans préfixe**) de longueurs de mots-données $(l_1, l_2, \ldots, l_n)$ existe si et seulement si

$$\sum_{i=1}^{n} d^{-l_i} \leq 1$$

où d est la taille de l'alphabet du *canal*, et n, la taille de l'alphabet de la *source* (d = 2 pour un canal *binaire*).

Historiquement, l'inégalité de Kraft a d'abord été démontrée par Mc Millan comme condition nécessaire et suffisante d'existence de codes déchiffrables de longueurs de mots données. Voici le théorème qui est une extension du théorème précédent sur la classe entière de codes déchiffrables :

Théorème 3.2. Condition de Mc Millan. Un *code déchiffrable* de longueurs de mots données $(l_1, l_2, \ldots, l_n)$ existe si et seulement si

$$\sum_{i=1}^{n} d^{-l_i} \leq 1$$

où d est la taille de l'alphabet du canal, et n, la taille de l'alphabet de la *source* (d = 2 pour un canal *binaire*).

Remarque 3.1. Ces deux théorèmes montrent qu'il existe un code *déchiffrable* de longueurs de mots données $(l_1, l_2, \ldots, l_n)$ si et seulement si il existe un code *instantané* de *mêmes longueurs de mots*.

Nous allons maintenant pouvoir poser le problème de *codage de source* ou encore *codage sans bruit* :

Soit une source S *d'alphabet* $\Omega_S = \{s_1, \cdots, s_n\}$ *de taille* n *et de distribution de probabilités* $P_S = \{p_1, \cdots, p_n\}$*. Soit un canal d'alphabet* $\Omega_C = \{c_1, \cdots, c_d\}$ *de taille* d*, sans bruit, stationnaire et sans mémoire. Existe-t-il un code qui minimise la longueur moyenne de mots* $\overline{L}$ *?*

2.2. Le premier théorème de Shannon

Nous allons approcher la solution du problème de codage de source en trois étapes. D'abord, nous allons énoncer la *borne inférieure pour la longueur moyenne de mots de code*. Ensuite, nous allons proposer une *borne supérieure*. Enfin, le *premier théorème fondamental de la théorie de l'information* montrera qu'il est possible d'*approcher la borne inférieure avec autant de précision que l'on souhaite*.

2.2.1. Borne inférieure de longueur moyenne de code

Théorème 3.3. Borne inférieure. Soit une source S d'alphabet $\Omega_S = \{s_1, \cdots, s_n\}$ de taille n et de distribution de probabilités $P_S = \{p_1, \cdots, p_n\}$. Soit un canal d'alphabet $\Omega_C = \{c_1, \cdots, c_d\}$ de taille d, sans bruit, stationnaire et sans mémoire. Soit un code déchiffrable $\{m_1, \cdots, m_n\}$ de longueurs de mots $\{l_1, \cdots, l_n\}$.
Alors la longueur moyenne de mots de code vérifie :

$$\left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) \geq \frac{H(S)}{\log_2(d)}$$

L'égalité n'est possible que si $\forall i = 1, \cdots, n \quad p_i = d^{-l_i}$.

Preuve du Théorème 3.3.

Puisque le code que nous avons est déchiffrable, il vérifie l'inégalité de Kraft :

$$0 < Q = \sum_{i=1}^{n} d^{-l_i} \leq 1$$

On peut alors définir les nombres :

$$q_i = \frac{d^{-l_i}}{Q} \qquad i = 1, \cdots, n$$

On a alors : $\sum_{i=1}^{n} q_i = 1$ et donc on peut appliquer l'inégalité de Gibbs (voir *lemme* 1.1. *Inégalité de Gibbs*) :

$$\sum_{i=1}^{n} p_i \log_2\left(\frac{p_i}{q_i}\right) \geq 0$$

et l'égalité a lieu *si et seulement si* $\forall i = 1, \cdots, n \quad p_i = q_i$.

On en déduit alors :

$$\sum_{i=1}^{n} p_i \log_2 p_i \geq \sum_{i=1}^{n} p_i \log_2 q_i \tag{3.0}$$

→ $$-\sum_{i=1}^{n} p_i \log_2 p_i \leq -\sum_{i=1}^{n} p_i \log_2 q_i \tag{3.1}$$

→ $$-\sum_{i=1}^{n} p_i \log_2 p_i \leq -\sum_{i=1}^{n} p_i \log_2 \frac{d^{-l_i}}{Q} \tag{3.2}$$

→ $$-\sum_{i=1}^{n} p_i \log_2 p_i \leq -\sum_{i=1}^{n} p_i \log_2 d^{-l_i} + \sum_{i=1}^{n} p_i \log_2 Q \tag{3.3}$$

→ $$-\sum_{i=1}^{n} p_i \log_2 p_i \leq \sum_{i=1}^{n} p_i l_i \log_2 d + \log_2 Q \sum_{i=1}^{n} p_i \tag{3.4}$$

En remarquant que $\sum_{i=1}^{n} p_i l_i = \overline{L}$, que $\sum_{i=1}^{n} p_i = 1$ et que par définition $H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i$, on a :

$$H(S) \leq \overline{L} \cdot \log_2(d) + \log_2 Q$$

Enfin, d'après l'inégalité de Kraft : $Q \leq 1$. Donc $\log_2 Q \leq 0$ d'où l'inégalité à démontrer : $\overline{L} \geq \frac{H(S)}{\log_2(d)}$.

Remarque 3.2. On peut noter que la quantité $\frac{H(S)}{\log_2(d)}$ représente l'entropie de la source calculée par rapport à la base d :

$$\frac{H(S)}{\log_2(d)} = -\sum_{i=1}^{n} p_i \frac{\log_2(p_i)}{\log_2(d)} = -\sum_{i=1}^{n} p_i \log_d(p_i) = H_d(S)$$

Un code dont la longueur moyenne de mots atteint la borne inférieure s'appelle un code **absolument optimal**. Un exemple de code *absolument optimal* est donné par le tableau suivant :

S	*Probabilité*	**Code absolument optimal**
a	0.5	0
b	0.25	10
c	0.125	110
d	0.125	111

et on a : $H(S) = \overline{L} = (0.5 \cdot 1) + (0.25 \cdot 2) + (0.125 \cdot 3 \cdot 2) = \frac{7}{4}$

2.2.2. Borne supérieure de longueur moyenne de code

Cependant, un code absolument optimal n'est pas toujours *réalisable*. En effet, pour atteindre la borne inférieure, les longueurs de mots de code doivent vérifier $p_i = d^{-l_i}$ et donc $l_i = -\frac{\log_2 p_i}{\log_2 d}$.

Or ces nombres ne sont pas forcément entiers. Dans ce cas, la meilleure solution consiste à choisir les longueurs de mots de code de sorte que :

$$\forall i = 1, \cdots, n \qquad -\frac{\log_2 p_i}{\log_2 d} \le l_i < -\frac{\log_2 p_i}{\log_2 d} + 1$$

Théorème 3.4. Borne supérieure. Soit une source S d'alphabet $\Omega_S = \{s_1, \cdots, s_n\}$ de taille n et de distribution de probabilités $P_S = \{p_1, \cdots, p_n\}$. Soit un canal d'alphabet $\Omega_C = \{c_1, \cdots, c_d\}$ de taille d, sans bruit, stationnaire et sans mémoire. Alors il existe un code déchiffrable dont la longueur moyenne de mots de code vérifie :

$$\frac{H(S)}{\log_2(d)} \le \overline{L} \left(= \sum_{i=1}^{n} p_i l_i \right) < \frac{H(S)}{\log_2(d)} + 1$$

Preuve du Théorème 3.4.

Choisissons les longueurs de mots comme précisé ci-dessus : $\forall i = 1, \cdots, n \qquad -\frac{\log_2 p_i}{\log_2 d} \le l_i < -\frac{\log_2 p_i}{\log_2 d} + 1$

Tout d'abord, montrons que l'inégalité de Kraft est vérifiée. En effet, de l'inégalité ci-dessus on déduit que :

$$\forall i = 1, \cdots, n \qquad \log_2 p_i \ge -l_i \log_2 d \quad \Rightarrow \quad p_i \ge d^{-l_i}$$

Alors pour la somme on a :

$$\sum_{i=1}^{n} d^{-l_i} \le \left(\sum_{i=1}^{n} p_i = 1 \right)$$

Alors il existe un code instantané de longueurs de mots l_i que nous avons choisies. Il reste à étudier sa longueur moyenne. On a, du fait que $\sum_{i=1}^{n} p_i = 1$, et de par la relation $\overline{L} = \sum_{i=1}^{n} p_i l_i$:

$$\left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) < \sum_{i=1}^{n} p_i \left(-\frac{\log_2 p_i}{\log_2 d} + 1\right) \qquad \text{car} \qquad l_i < -\frac{\log_2 p_i}{\log_2 d} + 1$$

→ $$\left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) < -\sum_{i=1}^{n} p_i \left(\frac{\log_2 p_i}{\log_2 d}\right) + \sum_{i=1}^{n} p_i$$

→ $$\left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) < -\sum_{i=1}^{n} p_i \left(\frac{\log_2 p_i}{\log_2 d}\right) + 1$$

→ $$\left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) < -\frac{1}{\log_2 d} \sum_{i=1}^{n} p_i \log_2 p_i + 1$$

→ $$\left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) < \frac{H(S)}{\log_2 d} + 1$$

et, en utilisant l'inégalité $\forall i = 1, \cdots, n \qquad l_i \geq -\frac{\log_2 p_i}{\log_2 d}$, soit : $\left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) \geq \left(-\sum_{i=1}^{n} p_i \frac{\log_2 p_i}{\log_2 d} = \frac{H(S)}{\log_2 d}\right)$

→ finalement : $$\frac{H(S)}{\log_2 d} \leq \left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) < \frac{H(S)}{\log_2 d} + 1$$

2.2.3. Extension de source et Premier théorème de Shannon

Nous avons déjà vu que, pour un alphabet et une distribution de probabilité donnés, il est possible de trouver un code déchiffrable et de longueur moyenne des mots de code proche de la borne inférieure $\frac{H(S)}{\log_2 d}$ à un bit de base d près. La dernière question qu'il reste à éclaircir est de savoir s'il est possible d'approcher cette borne avec une précision arbitraire. Pour répondre à cette question nous allons élargir notre point de vue sur le problème d'encodage de messages d'une source. Nous allons introduire l'idée de codage par blocs. Commençons par un exemple.

Exemple **3.2.** Soit une source X d'alphabet $\omega_X = \{a, b\}$ et de distribution de probabilité $P_X = \left\{\frac{3}{4}, \frac{1}{4}\right\}$ et un canal binaire ($d = 2$). L'entropie de cette source est :

$$H(X) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) = -\frac{3}{4}\log_2(3) + \frac{3}{4}\log_2(4) + \frac{1}{4}\log_2(4) = -\frac{3}{4}\log_2(3) + 2 \approx 0.811 \text{ bit/symbole.}$$

En codant cette source caractère par caractère, on peut envisager le code suivant :

$$m_1 = m(a) = 0 \qquad m_2 = m(b) = 1$$

de longueur moyenne de mots $\overline{L}_1 = 1$ bit par caractère. Un message T de longueur initiale $l(T)$ sera alors codé par une suite binaire de $l(T)$ bits et l'application du théorème **3.4.** donne : $H(X) \leq \overline{L}_1 < H(X) + 1$ car $\log_2(d) = 1$.

Et si on voulait coder les messages de cette source en associant *un code aux couples de caractères ?*

Tout d'abord, formalisons cette idée. Avec un alphabet de taille 2 il est possible de former 2^2 = 4 couples de caractères différents. On peut considérer cela comme un nouvel alphabet, d'une nouvelle source, Y. La variable aléatoire Y associée à cette nouvelle source correspond à l'observation de 2 caractères émis indépendamment par la source initiale X. Ainsi, $Y = (X_1, X_2)$ où X_1 et X_2 sont 2 variables aléatoires indépendantes ayant la même distribution que X. Compte tenu de l'indépendance de X_1 et X_2, on peut construire la distribution de probabilité de Y :

Y	*aa*	*ab*	*ba*	*bb*
P_Y	$p(a)^2 = 9/16$	$p(a)p(b) = 3/16$	$p(a)p(b) = 3/16$	$p(b)^2 = 1/16$

Enfin, l'entropie de Y peut aussi être déduite de l'indépendance de X_1 et X_2 :

$$H(Y) = H(X_1, X_2) = H(X_1 \cap X_2) = H(X_1) + H(X_2) = 2H(X)$$

Maintenant, si l'on considère Y comme une source à part entière, on peut appliquer le théorème **3.4.** Il est possible de trouver un code déchiffrable dont la longueur moyenne de mots de code $\overline{L}_2$ vérifie (ici d = 2 et donc $\log_2 d = 1$) :

$$H(Y) \leq \overline{L}_2 < H(Y) + 1 \quad \Leftrightarrow \quad 2H(X) \leq \overline{L}_2 < 2H(X) + 1$$

On a alors la relation :

$$H(X) \leq \frac{\overline{L}_2}{2} < H(X) + \frac{1}{2}$$

Il reste à remarquer que $\overline{L}_2$ est mesurée en "bits par symbole" de l'alphabet de Y. Or, tout symbole de l'alphabet de Y est un couple de caractères de l'alphabet de X. Ainsi, la quantité $\frac{\overline{L}_2}{2}$ représente la longueur moyenne de mot de code par symbole de X. En codant les symboles par couples et non de façon unitaire, nous avons alors un code dont la longueur moyenne de mot de code se trouve dans un intervalle plus petit (de longueur 1/2 au lieu de 1) autour de la borne inférieure : $H(X) \leq \frac{\overline{L}_2}{2} < H(X) + \frac{1}{2}$ et $H(X) \leq \overline{L}_1 < H(X) + \frac{1}{2}$

Voici l'exemple d'un tel code.

$Y = (X_1, X_2)$	*Probabilité*	**Code**
aa	9/16	0
ab	3/16	10
ba	3/16	110
bb	1/16	111

On a $H(Y) = 2H(X) \approx 1.62$ bits/symbole et $\overline{L}_2 = \frac{27}{16} \approx 1.69$ bits/valeur de Y, à comparer à $\overline{L}_1 = 1$ bit/caractère de X.

Enfin, il est très important de remarquer que la longueur moyenne $\overline{L}_2$ représente le nombre moyen de bits pour coder un "symbole" de l'alphabet de la source Y. Donc il s'agit de $\frac{27}{16}$ bits par symbole de Y, or chaque symbole de l'alphabet de Y est un couple de symboles de l'alphabet de X. Ainsi, en unités de mesure de la source X, on a $\overline{L}_2 = \frac{27}{2 \times 16} \approx 0.84$ bit par symbole de X.

Cela signifie qu'un message T de longueur $l(T)$ sera codé, avec le second code (codage par couples), en moyenne par $0.84 l(T)$ bits, au lieu de $l(T)$ bits avec le premier code (codage unitaire).

Ainsi, en introduisant le codage par blocs nous avons trouvé la possibilité de s'approcher de la borne inférieure. En généralisant l'idée de l'exemple précédent on peut formuler et démontrer le premier théorème fondamental de la théorie de l'information.

Définition 3.1. Extension de source. Soit une source X d'alphabet $\Omega_X = \{x_1, \cdots, x_n\}$. On appelle extension d'ordre s de la source X la source $Y = (X_1, \cdots, X_s)$ où $X_i \quad i = 1, \cdots, s$ sont les variables aléatoires indépendantes et identiquement distribuées selon la distribution de X.

Théorème 3.5. Premier théorème de Shannon. Soit une source X d'alphabet $\Omega_X = \{x_1, \cdots, x_n\}$ de taille n et de distribution de probabilités $P_X = \{p_1, \cdots, p_n\}$. Soit un canal d'alphabet $\Omega_C = \{c_1, \cdots, c_d\}$ de taille d, sans bruit, stationnaire et sans mémoire. Alors il existe un procédé de codage déchiffrable dont la longueur moyenne de mots de code est aussi voisine que l'on souhaite de la borne inférieure $\frac{H(S)}{\log_2 d}$.

Preuve du Théorème 3.5.

Soit la suite $(Y_s)_{s=1}^{\infty}$ d'extensions d'ordre s de la source Y. On alors $\forall s \geq 1 \quad H(Y_s) = sH(X)$ et il existe un code déchiffrable dont la longueur moyenne $\overline{L}_s$ de mots de code vérifie :

$$\frac{H(Y)}{\log_2 d} \leq \overline{L}_s < \frac{H(Y)}{\log_2 d} + 1 \quad \Leftrightarrow \quad s\frac{H(X)}{\log_2 d} \leq \overline{L}_s < s\frac{H(X)}{\log_2 d} + 1$$

On a alors la relation :

$$\frac{H(X)}{\log_2 d} \leq \frac{\overline{L}_s}{s} < \frac{H(X)}{\log_2 d} + \frac{1}{s}$$

Il reste à remarquer que $\overline{L}_s$ est mesurée en "bits par symbole" de l'alphabet de Y_s. Or, tout symbole de l'alphabet de Y_s est un s-uplet de caractères de l'alphabet de X. Ainsi, la quantité $\frac{\overline{L}_s}{s}$ représente la longueur moyenne de mot de code par symbole de X. Alors en passant à la limite quand s tend vers ∞, on a :

$$\lim_{s \to \infty} \frac{\overline{L}_s}{s} = \frac{H(X)}{\log_2 d}$$

3. Construction de codes optimaux

Le premier théorème de Shannon établit *l'existence* de codes dont la longueur moyenne de mots est aussi proche que l'on veut de la borne inférieure. Il ne propose cependant pas de méthode pour construire un tel code.
Rappelons qu'un code est dit **absolument optimal** pour une source donnée (représentée par son alphabet et sa distribution de probabilité) si sa longueur moyenne de mots atteint la borne inférieure. Nous avons mentionné que de tels codes n'existent pas toujours pour une source donnée.

Définition 3.2. Code optimal. Soit une source X d'alphabet $\Omega_X = \{x_1, \cdots, x_n\}$ et de distribution de probabilités $P_X = \{p_1, \cdots, p_n\}$ donnés. On dit qu'un code est **optimal** dans une classe de codes pour cette source si sa longueur de mots de code est *minimale* dans la classe considérée.

Dans ce chapitre nous nous intéressons au problème de construction effective de **codes optimaux**. Pour faciliter la compréhension, nous allons nous restreindre à l'étude de **codes binaires sans préfixe** *(instantanés)*. La généralisation des idées que nous allons exposer ici au cas d'alphabets de codes q-aires (de taille q) n'est pas difficile. De plus, les codes binaires occupent une grande place parmi les codes actuellement utilisés tout simplement à cause des principes de fonctionnement des ordinateurs.

3.1. Codes binaires instantanés et arbres

3.1.1. Quelques rappels sur les arbres

Nous rappelons ici quelques notions utiles dans la suite sur les arbres. Etant donné que nous avons choisi de présenter dans ce chapitre seulement les codes binaires, nous n'aurons besoin que *d'arbres binaires*.

Un *arbre binaire* est un cas particulier de *graphe* qui peut être facilement défini de façon récursive : un *arbre binaire* est soit *vide* soit composé d'un nœud particulier, appelé *racine*, d'un *sous-arbre gauche* et d'un *sous-arbre droit* qui sont eux mêmes des *arbres binaires disjoints*. Nous allons tirer de cette définition quelques propriétés essentielles et une description plus détaillée de la structure d'un *arbre binaire*.

1. Comme tout *graphe*, un *arbre binaire* est un couple (N, R) où N est un ensemble de *noeuds* et $R \subset N\text{x}N$ est un ensemble *d'arcs* reliant certains sommets. Un *arbre binaire* est un *graphe orienté* dans lequel les *arcs* sont considérés comme des relations "*père-fils*".

2. La *racine de l'arbre* est l'unique nœud qui n'a pas de père.

3. Dans un arbre binaire, *chaque noeud a au plus deux fils* et *chaque noeud* sauf la racine a *exactement un père*.

4. Les nœuds qui n'ont *pas de fils* s'appellent *feuilles de l'arbre*.

5. les nœuds qui ne sont *ni feuilles ni racine* s'appellent *internes*.

6. Un *chemin* entre 2 noeuds est une suite *d'arcs consécutifs* (2 arcs (i, j) et (k, l) sont consécutifs si l'extrémité du premier est l'origine du second : $j = k$). La *longueur* d'un chemin est le *nombre de branches* qui le constituent.

7. Tout *noeud* est relié à la racine par un *unique chemin*. On dit qu'un noeud est de niveau n si le chemin qui le relie à la racine est de longueur n.

8. On appelle *hauteur* ou *profondeur* d'un arbre la longueur du *plus long chemin* partant de la racine.

9. Un *arbre binaire complet de profondeur* n est un arbre dont *tous les noeuds* sauf les feuilles ont *exactement 2 fils*. Autrement dit, toutes les feuilles appartiennent au même niveau et tous *les noeuds ont exactement* 2 *ou* 0 *fils*. Un tel arbre a exactement 2^n *feuilles*, et pour tout $i \leq n$ il y a exactement 2 noeuds de niveau i.

10. On appelle *arbre binaire incomplet*, un arbre binaire obtenu à partir d'un arbre complet en *supprimant tous les successeurs d'un certain nombre de noeuds* ainsi que *toutes les branches qui touchent ces noeuds*. Les nœuds dont on a supprimé les successeurs deviennent les feuilles.
Attention ! Il existe des arbres qui ne sont *ni complets*, ni *incomplets*. Un arbre *incomplet* est un arbre dont *tous les noeuds ont* 2 *ou* 0 *fils*. A la différence d'un arbre *complet*, il n'est pas exigé que *toutes les feuilles soient de même niveau*.

11. Dans un arbre binaire *complet* ou *incomplet*, le *nombre de feuilles* F et le *nombre de branches* B vérifient la relation $B = 2(F - 1)$

Sur la *figure* 3.1. on peut voir un arbre binaire complet de profondeur 3 et plus loin sur la *figure* 3.2. un arbre incomplet obtenu à partir du premier en supprimant (*élagage*) les descendants de 2 noeuds.

3.1.2. Représentation de codes instantanés par les arbres

Soit $\{m_1, \cdots, m_n\}$ un code binaire sans préfixe de longueur maximale l. Il est évident que chaque mot de ce code peut être représenté par un chemin partant de la racine d'un arbre binaire complet de profondeur l.

Il suffit pour cela d'étiqueter les arcs de l'arbre avec 0 et 1 (*choix arbitraire*). Supposons qu'à un mot m_i de longueur $l_i \leq l$, on associe un chemin de l_i arcs en partant de la racine. Le chemin s'arrête alors à un noeud de niveau l_i. Comme aucun autre mot du code ne peut avoir celui-ci comme préfixe, on peut supprimer tous les descendants du noeud final du chemin. Ce dernier devient alors une feuille. Ainsi on peut associer à tout code sans préfixe un arbre binaire incomplet. L'arbre vu précédemment, est ainsi associé au code {1, 01, 001, 000} (voir *figure* 3.3.).

Ainsi dans un arbre correspondant à un code binaire, les feuilles correspondent aux mots du code. Si ce dernier est associé à l'alphabet d'une source, il est également possible d'associer aux feuilles les probabilités des symboles correspondants.

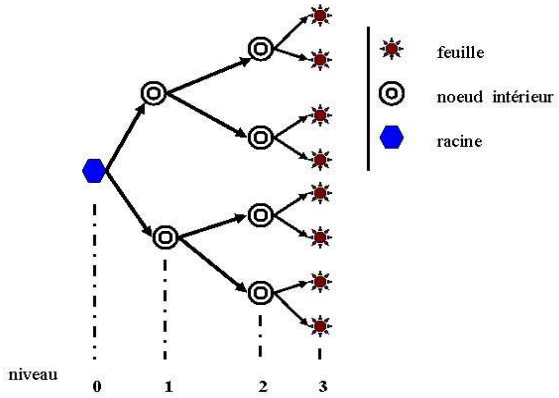


Figure 3.1. *Un arbre binaire complet de profondeur* 3.

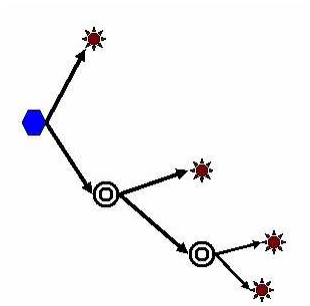


Figure 3.2. *Un arbre binaire incomplet.*

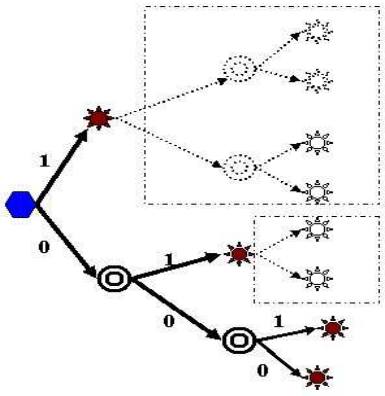


Figure 3.3. *Représentation d'un code par un arbre binaire incomplet (élagage).*

4. Méthode de Huffman de construction de codes optimaux

Nous allons maintenant présenter la méthode de Huffman de construction de codes instantanés optimaux pour une source donnée. Nous commençons par la condition nécessaire d'optimalité que nous présentons ici sans démonstration mais dans le but de faciliter la compréhension de la procédure de Huffman.

Lemme 3.1. Condition nécessaire d'optimalité. Soit une source X d'alphabet $\Omega_X = \{x_1, \cdots, x_n\}$ et de distribution de probabilités $P_X = \{p_1, \cdots, p_n\}$ donnés. Soit $C = \{m_1, \cdots, m_n\}$ un code instantané associé à la source de longueurs de mots $\{l_1, \cdots, l_n\}$. Supposons que les symboles de l'alphabet sont numérotés dans l'ordre décroissant de leurs probabilités :

$$p_1 \geq p_2 \geq \cdots \geq p_n$$

Si plusieurs symboles ont la même probabilité, ils sont numérotés dans l'ordre croissant des longueurs de mots de code correspondants.
Si le code C est optimal pour la source X dans la classe de codes instantanés, alors il vérifie les propriétés suivantes :

1. Les symboles les plus probables ont les longueurs de mots de code les plus petites :

$$p_i \geq p_j \quad \Rightarrow \quad l_i \leq l_j$$

2. Les deux derniers symboles ont des longueurs de mots de code égales :

$$l_n = l_{n-1}$$

3. Parmi les symboles dont la longueur de mots de code est l_n, il y en a au moins deux dont les mots de code ont tous les chiffres identiques sauf le dernier.

La méthode de Huffman (1952) procède de façon récursive, à partir de la source de n symboles, en construisant l'arbre binaire du code à partir des feuilles du niveau le plus élevé. D'après le lemme ci-dessus, ces feuilles doivent correspondre aux symboles les moins probables x_n et x_{n-1}. Le principe général de la construction est de regrouper les deux symboles les moins probables en un seul, $w_{n,n-1}$ en additionnant leurs probabilités et de considérer une nouvelle source de $n-1$ symboles. On parle alors de réduction de source. On applique ainsi le principe récursivement jusqu'à ce qu'il ne reste que deux symboles. On leur attribue alors le code {0, 1}.

Le parcours inverse permet de construire le code de manière également récursive. Soit C_{n-1} le code associé à la source de $n-1$ symboles dont le dernier ($w_{n,n-1}$) a été obtenu en regroupant x_n et x_{n-1}. Alors le code C_n associé à la source de n symboles est obtenu de la façon suivante : les mots du code associés aux symboles x_n et x_{n-1} sont construits en ajoutant au mot-code du symbole $w_{n,n-1}$ respectivement 0 et 1.

Voici un *exemple* :

Etape 1		**Etape 2**		**Etape 3**		**Etape 4**		**Etape 5**	
x_1	0.3	x_1	0.3	x_1	0.3	$x_{3,4,5,6}$	0.45	$x_{1,2}$	0.55
x_2	0.25	x_2	0.25	x_2	0.25	x_1	0.3	$x_{3,4,5,6}$	0.45
x_3	0.2	x_3	0.2	$x_{4,5,6}$	0.25	x_2	0.25		
x_4	0.1	$x_{5,6}$	0.15	x_3	0.2				
x_5	0.1	x_4	0.1						
x_6	0.05								

L'arbre représentant le code résultant est représenté sur la *figure* 3.4. On y retrouve les codes associés à tous les symboles de l'alphabet initial :

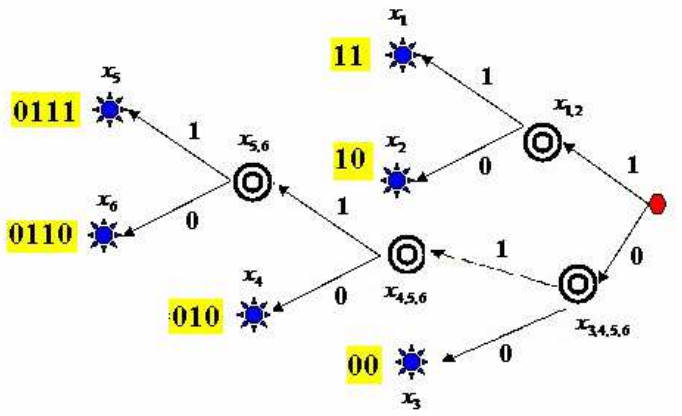


Figure 3.4. *Arbre de code de Huffman*

TD 3. Codage de source

Partie I. Codes sans préfixe.

Nous allons considérer une source d'alphabet $\Omega = \{a, b, c, d, e, f, g, h, i, j\}$ et un canal d'alphabet binaire $\{0, 1\}$.
$\rightarrow$ il y a nécessité de *codage* pour adapter l'alphabet de la source (les symboles à transmettre) à celui du canal.

Rappels

Nous utiliserons dans la suite la terminologie et les définitions suivantes :

- **Lettre, symbole ou caractère.** Tout *élément d'un alphabet* donné.
- **Message** ou **mot.** Une *séquence finie* m *de caractères* d'un alphabet donné.
- **Longueur de mot.** Le *nombre* $l(m)$ *de caractères d'un mot* m.
- **Code.** Un *code binaire* pour l'alphabet donné Ω de taille n est un ensemble de n mots binaires $\{m_1, \cdots, m_n\}$. Un code est *régulier* si les mots correspondant aux différents caractères de l'alphabet sont différents. Dans la suite nous étudions uniquement les *codes réguliers*.

Exemple. Pour notre alphabet un *code régulier* peut être :

s_i	a	b	c	d	e	f	g	h	i	j
m_i	0	1	00	01	11	10	100	101	110	111

- **Code déchiffrable.** Un code binaire est *déchiffrable* si toute séquence de bits peut être décodée de façon *unique*. Le code ci-dessus *n'est pas déchiffrable*. En effet, la séquence de bits "110" peut être interprétée comme "i" ou comme "bf" ou même comme "ea".

- **Code sans préfixe** ou **instantané.** Un code est *sans préfixe* ou *instantané* si *aucun mot code n'est le préfixe d'un autre*. Le code ci-dessus *n'est pas un code sans préfixe*. En effet, le mot code 0 est le début des mots du code 00 et 01. Le code suivant est *sans préfixe* :

s_i	a	b	c	d	e	f	g	h	i	j
m_i	1111	1110	110	1011	1010	100	011	000	010	001

- **Décodage pas à pas d'un code sans préfixe.** Le principe de décodage est simple. Il suffit de lire la séquence codée de gauche à droite jusqu'à ce qu'on trouve un mot du code. On est alors certain que ce mot corresponde sans ambiguïté à un caractère de l'alphabet. On enregistre le caractère et on continue le même processus.

Théorème 3.1. Inégalité de Kraft. Un **code instantané** (≡ **sans préfixe**) de longueurs de mots-données $(l_1, l_2, \ldots, l_n)$ existe si et seulement si

$$\sum_{i=1}^{n} d^{-l_i} \leq 1$$

où d est la taille de l'alphabet du *canal*, et n, la taille de l'alphabet de la *source* (d = 2 pour un canal *binaire*).

Exercice 1. Décodage. Appliquer la procédure de décodage pas à pas à la séquence 10110000101010100 avec le code *sans préfixe* suivant :

s_i	a	b	c	d	e	f	g	h	i	j
m_i	1111	1110	110	1011	1010	100	011	000	010	001

Exercice 2. Code sans préfixe. Vérifier s'il existe un *code binaire sans préfixe* de longueurs de mots données : {4, 4, 4, 3, 2, 4, 3, 4, 3, 4}.

Partie II. Codes sans préfixe et arbres binaires.

Rappels

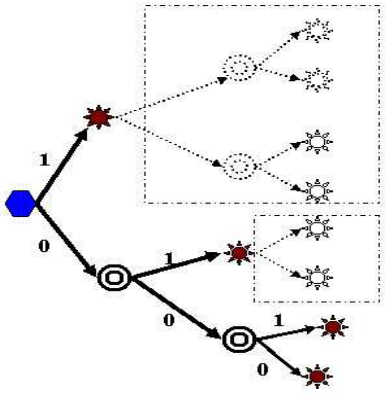


Figure 1. *Exemple*

Soit $\{m_1, \cdots, m_n\}$ un code binaire sans préfixe de longueur maximale l. Il est évident que chaque mot de ce code peut être représenté par un chemin partant de la racine d'un arbre binaire complet de profondeur l. Il suffit pour cela d'étiqueter les arcs de l'arbre avec 0 et 1 arbitrairement.
Supposons qu'à un mot m_i de longueur $l_i \leq l$ on associe un chemin de l_i arcs en partant de la racine. Le chemin s'arrête alors à un noeud de niveau l_i. Comme aucun autre mot du code ne peut avoir celui ci comme préfixe, on peut supprimer tous les descendants du noeud final du chemin (*élagage*). Ce dernier devient alors une feuille. On peut donc associer à tout code sans préfixe, un arbre binaire incomplet.
L'arbre vu précédemment, est ainsi associé au code {1, 01, 001, 000} (*figure* 1).

Dans un arbre associé à un code binaire, les feuilles correspondent aux mots du code. Si ce dernier est lié à l'alphabet d'une source, il est également possible d'associer aux feuilles les probabilités des symboles correspondants.

Exercice 3. Code sans préfixe. Arbre binaire.

1. Construire l'arbre associé au code suivant :

s_i	a	b	c	d	e	f	g	h	i	j
m_i	1111	1110	110	1011	1010	100	011	000	010	001

2. Construire un *code sans préfixe* de longueurs de mots suivantes, à l'aide d'un arbre binaire, en procédant à l'élagage d'un arbre complet :

s_i	a	b	c	d	e	f	g	h	i	j
l_i	4	4	4	3	2	4	3	4	3	4

Partie III. Codage de Huffman

Rappels

Nous allons considérer une source d'alphabet $\Omega = \{a, b, c, d, e, f, g, h, i, j\}$ et un canal d'alphabet binaire $\{0, 1\}$. Voici quelques définitions importantes.

Théorème de la borne inférieure de longueur de code

Théorème 3.3. Borne inférieure. Soit une source S d'alphabet $\Omega_S = \{s_1, \cdots, s_n\}$ de taille n et de distribution de probabilités $P_S = \{p_1, \cdots, p_n\}$. Soit un canal d'alphabet binaire $\Omega_C = \{0, 1\}$, donc de taille $d = 2$, sans bruit, stationnaire et sans mémoire. Soit un code déchiffrable $\{m_1, \cdots, m_n\}$ de longueurs de mots $\{l_1, \cdots, l_n\}$.
Alors la longueur moyenne de mots de code vérifie :

$$\left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) \geq \frac{H(S)}{\log_2(d)} \quad \Rightarrow \quad \left(\overline{L} = \sum_{i=1}^{n} p_i l_i\right) \geq H(S)$$

L'égalité n'est possible que si $\forall i = 1, \cdots, n \qquad p_i = d^{-l_i} = 2^{-l_i}$.

Code absolument optimal. C'est un code dont la *longueur moyenne de mots est égale à la borne inférieure,* $H(S)$.

Code optimal. Un code est dit *optimal* dans une certaine classe de codes si sa *longueur moyenne de mots est minimale dans cette classe*. La classe de codes la plus importante est celle de codes *sans préfixe.*

Exercice 4. Code sans préfixe optimal. Méthode de Huffman. Soit une source d'alphabet Ω et de distribution de probabilité suivante :

s_i	a	b	c	d	e	f	g	h	i	j
p_i	0.2	0.05	0.1	0.05	0.15	0.05	0.1	0.05	0.15	0.1

1. Quelle est la *longueur moyenne minimale* pour un code *binaire* de cette source ?
2. Existe-t-il un *code absolument optimal* ?
3. Construire un *code sans préfixe optimal* selon la *méthode de Huffman*.
4. Représenter ce *code* sous forme *d'arbre*.

———————

4. Compression de données (sans pertes)

1. Codage et compression

Nous allons ici aborder les applications de la théorie de l'information à la compression des données numériques. Nous avons décrit précédemment les méthodes de *codage* qui permettent d'adapter les messages émis par une source à la transmission via un canal tout en assurant la transmission la plus rapide en moyenne et un décodage sans erreurs ni ambiguïtés. Nous avons associé la *rapidité de transmission* à la *longueur des messages codés*. Le paramètre clé est alors la *longueur moyenne des mots de code.*
Les résultats fondamentaux du codage de source ont une autre application : la *compression des données*. En effet, le *premier théorème de Shannon* permet d'établir les *limites théoriques de compression de données sans pertes*, *l'entropie* d'un message (constitué de caractères d'un alphabet binaire) représentant la *longueur moyenne minimale de mots de code* pour ce message. Le codage à longueur de mots fixe est souvent utilisé car il représente l'avantage de simplicité et rapidité d'application. Il est tout de même loin d'être optimal et lorsque les données obtenues sont trop volumineuses, il se pose le problème de la compression.
Il existe deux grandes familles de méthodes de compression :

Compression sans pertes. Il s'agit de méthodes inversibles, garantissant que le message initial peut être restauré intégralement à partir du compressé. Dans ce groupe de méthodes on retrouve en particulier, le format zip utilisé pour l'archivage de fichiers et le format d'images GIF, par exemple. Le taux de compression est limité par l'entropie des données.

Compression avec pertes. Il s'agit de techniques basées sur les approximations des données initiales. Lors de la restauration, le message obtenu n'est pas exactement le message initial. On retrouve dans ce groupe les formats d'images JPEG, audio MP3 et vidéo MPEG par exemple.

Nous allons nous intéresser uniquement aux méthodes de compression sans pertes, basées sur l'utilisation des algorithmes de codage plus ou moins proches de la borne inférieure établie par le premier théorème de Shannon.
On peut regrouper les algorithmes de codage pour compression en deux familles principales :

Algorithmes statistiques. Il s'agit de méthodes basées sur l'utilisation de la table de fréquences d'occurrence des symboles, comme le codage de Huffman.

Algorithmes à dictionnaire. Il s'agit de coder non pas *caractère par caractère* mais *par blocs* en indiquant la position de chaque bloc dans un dictionnaire construit à partir du message.

Tout d'abord, nous introduisons quelques notions clés qui nous seront utiles pour apprécier les qualités de différents codes en tant que méthodes de compression des données. Nous avons vu précédemment qu'il existe une borne inférieure pour la longueur moyenne de mots de tout code déchiffrable dont l'alphabet de canal a d caractères :

$$\frac{H(S)}{d}$$

où $H(S)$ est l'entropie de la source. Dans ce chapitre, nous allons nous restreindre à l'étude de codes binaires, c'est-à-dire, tels que $d = 2$ (*rappel :* $\log_2 d = 1$). Ainsi, dans ce cas particulier, c'est l'entropie de la source $H(S)$ qui représente la valeur minimale possible de longueur moyenne de mots de code.

Soit une source S d'alphabet $\Omega = \{s_1, \cdots, s_n\}$ et d'entropie $H(S)$. Soit un code binaire déchiffrable pour S défini par l'ensemble de mots $\{\omega_1, \cdots, \omega_n\}$ de longueurs respectives $\{l_1, \cdots, l_n\}$. Soit $\overline{L}$ la longueur moyenne de mots de code. On appelle **efficacité** du code, la quantité

$$E = \frac{H(S)}{\overline{L}}$$

et redondance du code la quantité

$$R = 1 - E = 1 - \frac{H(S)}{\overline{L}}$$

Ces quantités permettent de comparer la longueur moyenne de mot de code d'un code donné à la borne inférieure $H(S)$. Rappelons que cette borne n'est pas toujours atteinte, même par les codes optimaux.

Soit un texte T composé de symboles d'un alphabet $\Omega = \{s_1, \cdots, s_n\}$. Soit Λ le nombre de symboles dans T. Si le texte est codé à l'aide d'un code binaire de longueur de mots fixe (ASCII ou UTF-8, par exemple), la mémoire occupée par ces données est proportionnelle au nombre Λ de symboles dans le texte. Par exemple, si on utilise m bits par symbole, la longueur totale de texte codé est

$$l(T) = m\Lambda$$

Le problème de compression des données dans ce cas consiste à rechercher un code binaire pour le texte T tel que la longueur $l(U)$ du message codé noté U vérifie

$$l(U) < l(T)$$

Taux de compression (exprimée en %) : c'est la quantité

$$t = \frac{l(T) - l(U)}{l(T)} = 1 - \frac{l(U)}{l(T)} \text{ encore noté } t = \frac{NC - C}{NC} = 1 - \frac{C}{NC} \text{ avec } \begin{cases} l(T) = NC : \text{taille des données Non Compressées} \\ l(U) = C : \text{taille des données Compressées} \end{cases}$$

Pour appliquer à ce problème la théorie de construction de codes optimaux vue précédemment, nous allons définir la procédure permettant d'associer à un texte donné une distribution de probabilités aux symboles s_i de l'alphabet Ω.

Soit λ_i le nombre d'occurrences du symbole s_i dans le texte T. Notons alors $f_i = \frac{\lambda_i}{\Lambda}$ la fréquence du symbole s_i. On a alors : $\forall i = 1, \cdots, n \qquad 0 \le f_i \le 1 \qquad \sum_{i=1}^{n} f_i = 1$

Ainsi, l'ensemble $\{f_i \quad i = 1, \cdots, n\}$ est une distribution de probabilité sur Ω.

Cela nous permet de considérer le texte à compresser T comme étant un message émis par une source ayant cette distribution de probabilité. Alors les méthodes de construction de codes optimaux et en particulier la méthode de Huffman peuvent être appliquées pour la compression de ce texte.
Il est important de souligner que dans ce cas le code construit est propre au texte T et le taux de compression obtenu dépend également des caractéristiques du texte.

2. Algorithmes de compression statistiques

2.1. Méthode de Shannon-Fano

On suppose la table de fréquences d'apparition des symboles $\{f_i \quad i = 1, \cdots, n\}$ construite. La méthode de Shannon-Fano est une méthode récursive. Elle permet de construire l'arbre du code en partant de sa racine, contrairement à la méthode de Huffman qui procède à partir des feuilles.

Les étapes de la méthode sont les suivantes :

1. On élimine de l'alphabet Ω les symboles dont les fréquences sont nulles et on trie la table dans l'ordre décroissant des fréquences.

2. On divise la table de fréquences en deux parties solidaires et formant une partition de Ω (c'est-à-dire d'intersection nulle et d'union égale à Ω) de telle sorte que les sommes des fréquences de ces parties soient aussi proches que possible (médiane). On attribue (arbitrairement) 1 à la première partie et 0 à la seconde.

3. On applique ensuite ce principe des divisions à chacune des parties récursivement jusqu'à l'obtention de parties à un seul symbole.

2.1.1. Exemple

Soit T = '*cette echelle est belle*'. Il y a Λ = 20 symboles dans ce texte (le caractère « espace » est ignoré). Codé en ASCII (8 bits par symbole) il occupe alors $l(T) = 8 \cdot 20 = 160$ bits. Nous retenons cette valeur comme référence pour évaluer les taux de compression obtenus. Nous allons construire pour ce texte les codes de Shannon-Fano et de Huffman. Tout d'abord, on construit la table des fréquences d'occurrence des symboles du texte :

s_i	e	l	t	c	b	h	s
λ_i	8	4	3	2	1	1	1
f_i	0.4	0.2	0.15	0.1	0.05	0.05	0.05

L'alphabet initial est ainsi $\Omega = \{e, l, t, c, b, h, s\}$. L'entropie associée est $H(T) \approx 2.38$ bits/symbole.

Construction d'un code de Shannon-Fano. Pour commencer on divise l'alphabet en deux sous-ensembles de sommes de probabilités aussi proches que possible. Ici on a $\Omega_1 = \{e, l\}$ avec $P(\Omega_1) = 0.4 + 0.2 = 0.6$ et $\Omega_2 = \{t, c, b, h, s\}$ avec $P(\Omega_2) = 0.15 + 0.1 + 3(0.05) = 0.4$.

On associe 1 comme premier bit des mots de code des symboles de Ω_1 et 0 comme premier bit des mots de code des symboles de Ω_2. Sur chacun des sous-alphabets ainsi obtenus on définit les distributions de probabilités en divisant celles de Ω par $P(\Omega_1)$ et $P(\Omega_2)$ respectivement. On a :

$P_1 =$

s_i	e	l
f_i	4/6	2/6

$P_2 =$

s_i	t	c	b	h	s
f_i	3/8	1/4	1/8	1/8	1/8

On continue ensuite, en appliquant la même procédure à chacun des deux tableaux obtenus. Voici le graphe final :

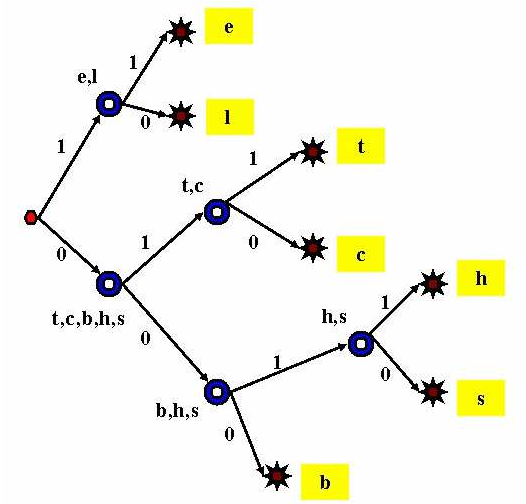


Figure 4.1. *Codage de Shannon-Fano*

Un code possible qui en résulte se lit sur les chemins reliant les feuilles à la racine :

s_i	e	l	t	c	b	h	s
ω_i	11	10	011	010	000	0011	0010
l_i	2	2	3	3	3	4	4

La longueur moyenne de mots de code est

$$\overline{L} = 2(0.4) + 2(0.2) + 3(0.15) + 3(0.1) + 3(0.05) + 4(0.05) + 4(0.05) = 2.5 \text{ bits/symbole.}$$

Le message codé sera (sans les espaces) : U = 0101101101111110100011111010111100100110001110101 1 et on a $l(U) = 50$ bits. Ainsi le taux de compression est : $t = \frac{160 - 50}{160} = 68.75\%$.

2.2. Algorithme de Huffman

Le codage de Huffman représente aussi une méthode de compression car il réduit la longueur moyenne des mots de code. Nous avons décrit cette méthode en détails précédemment. Voici un exemple d'utilisation dans le cas vu dans la section précédente.

Construction d'un code de Huffman. La méthode de Huffman produit le tableau suivant et un arbre de code possible :

e	0.4
l	0.2
t	0.15
c	0.1
b	0.05
h	0.05
s	0.05

e	0.4
l	0.2
t	0.15
hs	0.1
b	0.05

e	0.4
l	0.2
t	0.15
hsb	0.15
c	0.1

e	0.4
hsbc	0.25
l	0.2
t	0.15

e	0.4
lt	0.35
hsbc	0.25

lthsbc	0.6
e	0.4

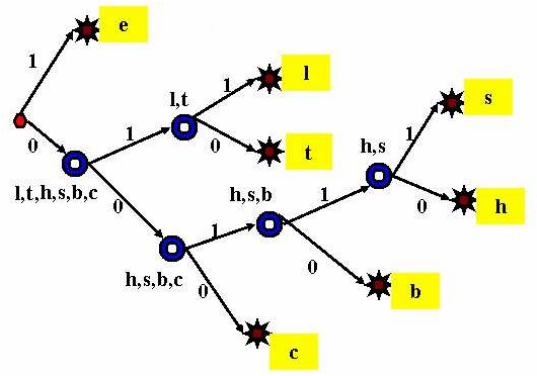


Figure 4.2. *Codage de Huffman*

Un code possible qui en résulte se lit sur les chemins reliant les feuilles à la racine :

s_i	*e*	*l*	*t*	*c*	*b*	*h*	*s*
ω_i	1	011	010	000	0010	00110	00111
l_i	1	3	3	3	4	5	5

La longueur moyenne de mots de code est

$$\overline{L} = 1(0.4) + 3(0.2) + 3(0.15) + 3(0.1) + 4(0.05) + 5(0.05) + 5(0.05) = 2.45 \text{ bits/symbole.}$$

Le message codé sera (sans les espaces) : U = 0001010010110000011010110111100111010001010110111 1 et on a $l(U) = 49$ bits. Ainsi le taux de compression est : $t = 1 - \frac{49}{160} = 69.375\%$.

3. Codes à dictionnaire

Les trois algorithmes les plus connus sont LZ77 (Lempel-Ziv), LZ78 (Lempel-Ziv) et LZW (Lempel-Ziv-Welsh, 1984). Le principe fondateur commun des ces trois méthodes consiste à encoder des séquences de caractères par les références à leurs emplacements dans un dictionnaire. Le dictionnaire est construit à partir du texte lui même et contient toutes les séquences déjà rencontrées. Nous allons dans la suite détailler le fonctionnement de l'algorithme LZ78.

3.1. LZ78

3.1.1. Codage

Le dictionnaire sera construit au fur et à mesure de la lecture du texte à compresser. A tout instant le dictionnaire représente la partie du texte déjà lu, découpée en séquences numérotées. Le numéro 0 est réservé à la chaîne de caractères vide.
Chaque séquence codée sera remplacée par un couple : (i, c). Si la séquence codée a n caractères, i est le numéro d'entrée dans le dictionnaire du préfixe composé des $n-1$ premiers caractères. Si aucun préfixe n'a été trouvé, i = 0, c est le dernier caractère de la séquence codée.
Pour déterminer la nouvelle séquence à encoder on procède de la façon suivante. On parcourt le texte restant à compresser à partir du caractère courant tant que la séquence lue existe dans le dictionnaire. On s'arrête dés qu'une séquence nouvelle, non encore enregistrée dans le dictionnaire, est trouvée. Ce procédé garantit que la nouvelle séquence s est de la forme

$$s = s_d c$$

où c est le dernier caractère lu et s_d est la dernière séquence rencontrée qui appartient au dictionnaire.

Illustrons le propos par un exemple. Supposons que nous cherchons à compresser le texte (caractères « espace » inclus) :
'ELLE EST BELLE CETTE ECHELLE ETERNELLE'.

- On initialise le dictionnaire avec la séquence vide à l'emplacement 0.
- On lit le caractère 'E'. Cette séquence n'est pas présente dans le dictionnaire. On la remplace par le couple (0,E) et on enregistre dans le dictionnaire la séquence 'E' à l'adresse i = 1.
- Le caractère lu : 'L'. Cette séquence n'est pas dans le dictionnaire. On la remplace par le couple (0,L) et on enregistre 'L' dans le dictionnaire à l'emplacement i = 2.
- Caractère lu : 'L'. Séquence présente dans le dictionnaire à l'adresse i = 2. On lit le caractère suivant 'E'. La séquence en attente devient : 'LE'. Elle n'est pas dans le dictionnaire. On la remplace par le couple (2,E) et on l'enregistre dans le dictionnaire à l'adresse i = 3.
- Caractère lu : '␣'. Il n'existe pas dans le dictionnaire. On le remplace par (0,␣) et on l'enregistre à l'adresse i = 4.
- caractère lu : 'E', adresse dans le dictionnaire : i = 1. On lit le caractère suivant. Séquence en attente : 'ES'. On la remplace par (1,S) et on l'enregistre à l'adresse i = 5.
- etc …

Voici le tableau qui résume toute la procédure :

Indice	*Dictionnaire*	*Code*
0	Null	
1	E	(0,E)
2	L	(0,L)
3	LE	(2,E)
4	␣	(0,␣)
5	ES	(1,S)
6	T	(0,T)
7	␣B	(4,B)
8	EL	(1,L)
9	LE␣	(3,␣)
10	C	(0,C)
11	ET	(1,T)
12	TE	(6,E)

Indice	*Dictionnaire*	*Code*
13	␣E	(4,E)
14	CH	(10,H)
15	ELL	(8,L)
16	E␣	(1,␣)
17	ETE	(11,E)
18	R	(0,R)
19	N	(0,N)
20	ELLE	(15,E)

Le code définitif obtenu sera :

(0E, 0L, 2E, 0␣, 1S, 0T, 4B, 1L, 3␣, 0C, 1T, 6E, 4E, 10H, 8L, 1␣, 11E, 0R, 0N, 15E)

Si l'on utilise 1 Octet pour coder une adresse de dictionnaire (nombre entier) et 1 octet pour coder un caractère, nous arrivons ainsi à 40 caractères pour le code final au lieu de 38 pour le texte initial. Il peut sembler que ce codage n'est pas efficace pour compresser les données. Bien au contraire ! Son efficacité s'améliore nettement sur des textes longs. Voici notre exemple, juste un peu prolongé : 'ELLE EST BELLE CETTE ECHELLE ETERNELLE. ELLE EST REELLE'.

Indice	*Dictionnaire*	*Code*
0	Null	
1	E	(0,E)
2	L	(0,L)
3	LE	(2,E)
4	␣	(0,␣)
5	ES	(1,S)
6	T	(0,T)
7	␣B	(4,B)
8	EL	(1,L)
9	LE␣	(3,␣)
10	C	(0,C)
11	ET	(1,T)
12	TE	(6,E)

Indice	*Dictionnaire*	*Code*
13	␣E	(4,E)
14	CH	(10,H)
15	ELL	(8,L)
16	E␣	(1,␣)
17	ETE	(11,E)
18	R	(0,R)
19	N	(0,N)
20	ELLE	(15,E)
21	.	(0,.)
22	␣ES	(13,S)
23	T␣	(6,␣)
24	RE	(18,E)
25	ELLE.	(20,.)

On obtient un code de 50 caractères au lieu de 56 caractères pour le texte initial.

3.1.2. Décodage

L'avantage de la méthode de Lempel-Ziv est que le dictionnaire n'a pas besoin d'être transmis. Seul le code est envoyé. Le dictionnaire, nécessaire au décodage, sera reconstruit par le décodeur. A chaque couple de code lu, le décodeur va en effet effectuer deux actions :

1. Restituer la séquence qu'il représente en concaténant le mot dictionnaire indiqué par l'adresse du couple et le caractère contenu dans le couple.

2. Enregistrer la nouvelle séquence dans le dictionnaire.

Voici le décodage du code de notre premier exemple :
(0E, 0L, 2E, 0␣, 1S, 0T, 4B, 1L, 3␣, 0C, 1T, 6E, 4E, 10H, 8L, 1␣, 11E, 0R, 0N, 15E)

Le dictionnaire est initialisé par la chaîne de caractères vide à l'adresse 0.

- Couple (0,E). Séquence décodée : 'E'. Adresse dans le dictionnaire : i = 1. Texte = 'E'
- Couple (0,L). Séquence décodée : 'L'. Adresse dans le dictionnaire : i = 2. Texte = 'EL'
- Couple (2,E). Séquence décodée : 'LE'. Adresse dans le dictionnaire : i = 3. Texte = 'ELLE'
- Couple (0,␣). Séquence décodée : '␣'. Adresse dans le dictionnaire : i = 4. Texte = 'ELLE '
- Couple (1,S). Séquence décodée : 'ES'. Adresse dans le dictionnaire : i = 5. Texte = 'ELLE ES'
- Couple (0,T). Séquence décodée : 'T'. Adresse dans le dictionnaire : i = 6. Texte = 'ELLE EST'
- Couple (4,B). Séquence décodée : '␣B'. Adresse dans le dictionnaire : i = 7. Texte = 'ELLE EST B'
- Couple (1,L). Séquence décodée : 'EL'. Adresse dans le dictionnaire : i = 8. Texte = 'ELLE EST BEL'
- Couple (3,␣). Séquence décodée : 'LE '. Adresse dans le dictionnaire : i = 9. Texte = 'ELLE EST BELLE '
- Couple (0,C). Séquence décodée : 'C'. Adresse dans le dictionnaire : i = 10. Texte = 'ELLE EST BELLE C'
- Couple (1,T). Séquence décodée : 'ET'. Adresse dans le dictionnaire : i = 11. Texte = 'ELLE EST BELLE CET'
- etc …

3.2. LZW

L'algorithme LZW (Lempel-Ziv-Welsh) est une variante plus récente (1984) de l'algorithme de base de Lempel-Ziv. Il permet de réduire le code en remplaçant le couple (i, c) par le seul indice i du dictionnaire.
Le dictionnaire est initialisé par les codes ASCII de l'alphabet latin étendu : 256 caractères en tout. On lit ensuite le texte à compresser à travers une fenêtre dont la taille grandit, caractère par caractère, jusqu'à rencontrer une séquence qui n'est pas dans le dictionnaire. Comme dans l'algorithme LZ78, cette séquence, si elle est de longueur l, est composée d'un mot du dictionnaire de longueur $l-1$ et du dernier caractère lu. Au lieu de coder cette séquence par le couple (i, c) comme dans l'algorithme LZ78, on code le mot déjà existant dans le dictionnaire, en le remplaçant par son adresse i. On ajoute ensuite au dictionnaire cette nouvelle séquence à une nouvelle adresse. La prochaine fois qu'elle sera lue dans le texte, elle sera codée par son adresse.

Voici ce qu'on obtient en appliquant cette méthode à notre exemple 'ELLE EST BELLE CETTE ECHELLE ETERNELLE'.

- On initialise le dictionnaire avec les 256 caractères ASCII, numérotés de 0 à 255.
- On lit le caractère 'E'. Cette séquence est présente dans le dictionnaire, i = 69. On lit le caractère suivant : 'EL'. Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse i = 256. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'L'.
- On lit le caractère 'L'. Cette séquence est présente dans le dictionnaire, i = 76. On lit le caractère suivant : 'LL'. Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse i = 257. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'L'.
- On lit le caractère 'L'. Cette séquence est présente dans le dictionnaire, i = 76. On lit le caractère suivant : 'LE'. Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse i = 258. On encode 'L' en le remplaçant par 76 et on reprend la lecture au caractère 'E'.
- On lit le caractère 'E'. Cette séquence est présente dans le dictionnaire, i = 69. On lit le caractère suivant :'E␣'.
Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse i = 259. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère '␣'.
- On lit le caractère '␣'. Cette séquence est présente dans le dictionnaire, i = 32. On lit le caractère suivant :'␣E'.
Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse i = 260. On encode '␣' en le remplaçant par 32 et on reprend la lecture au caractère 'E'.
- On lit le caractère 'E'. Cette séquence est présente dans le dictionnaire, i = 69. On lit le caractère suivant : 'ES'. Nouvelle séquence. On l'ajoute au dictionnaire à l'adresse i = 261. On encode 'E' en le remplaçant par 69 et on reprend la lecture au caractère 'S'.
- etc …

Voici le tableau qui résume toute la procédure :

Indice	*Dictionnaire*	*Code*
256	EL	69
257	LL	76
258	LE	76
259	E␣	69
260	␣E	32
261	ES	69
262	ST	83
263	T␣	84
264	␣B	32
265	BE	66
266	LLE	257
267	E␣C	259
268	CE	67
269	ET	69

Indice	*Dictionnaire*	*Code*
270	TT	84
271	TE	84
272	E␣E	259
273	EC	69
274	CH	67
275	HE	72
276	ELL	256
277	LE␣	258
278	␣ET	260
279	TER	271
280	RN	82
281	NE	78
282	ELLE	275

Le code obtenu pour la phrase sera :

69, 76, 76, 69, 32, 69, 83, 84, 32, 66, 257, 259, 67, 69, 84, 84, 259, 69, 67, 72, 256, 258, 26O, 271, 82, 78, 275, 69

Nous avons 28 nombres entre 0 et 511 qui peuvent donc être codés sur 9 bits. On a ainsi 252 bits de code. Le texte initial a 38 caractères ASCII, codés sur 8 bits, donc 304 bits. Bien évidemment, le taux de compression est meilleur pour un texte grand.
Le décodage se fait, comme dans le cas de LZ78 de manière inverse, en restituant le dictionnaire au fur et à mesure de la lecture du code.

3.3. Remarques

La formulation de l'algorithme ne dit rien de particulier sur la taille du dictionnaire. En théorie, l'algorithme fonctionne avec un dictionnaire illimité. Dans la pratique, il est toujours de taille fixe. Lorsqu'il est plein on a plusieurs choix (selon la méthode utilisée) : soit de l'étendre, soit d'en « oublier » une partie, soit de se contenter de l'utiliser tel quel pour encoder simplement le reste du texte.
Le taux de compression obtenu peut être amélioré en utilisant une technique de codage statistique (Huffman, par exemple) sur les adresses. On exploite ainsi les fréquences d'occurrence non seulement des caractères mais aussi des séquences de taille variable.
De nombreux utilitaires de compression des données utilisent les algorithmes à dictionnaire.

La commande **compress sous UNIX** applique l'algorithme LZW avec une taille initiale du dictionnaire de 512 mots (codés sur 9 bits). Si le dictionnaire est rempli, on double sa taille (codage sur 10 bits). Il est possible de préciser la taille maximale du dictionnaire. Lorsque l'algorithme l'atteint, il poursuit le codage de façon statique, sans modifier le dictionnaire.

GZIP, PKZIP, WINZIP : ces logiciels utilisent une variante de l'algorithme LZ77.

Formats d'images GIF, PNG : ils utilisent également une variante du codage par dictionnaire.

4. Exemple. Compression d'images

Les techniques de compression d'images sont aujourd'hui nombreuses, tant dans le groupe de méthodes avec pertes (JPEG) que dans celui des méthodes sans pertes (GIF).
Nous allons aborder ici quelques méthodes simples de compression sans pertes utilisées par les différents formats d'images disponibles.
Pour simplifier les considérations ci-dessous, nous allons étudier les images dites monochromes (en niveaux de gris, par exemple). Une telle image peut être représentée comme une matrice contenant les valeurs de luminance, d'intensité lumineuse (codées par un entier généralement compris entre 0 - Noir - et 255 - Blanc) de chaque point d'image (pixel). La profondeur ou dynamique de codage est le nombre de bits utilisés pour coder la luminance d'un pixel (généralement 8 pour 256 nuances de gris).
Nous allons considérer que les valeurs de luminance sont représentées directement par un entier compris entre 0 et 255. Il existe aussi une approche par palette, une sorte de dictionnaire ou table de couleurs, utilisée surtout pour les images de synthèse. Dans le cas des images réelles, l'utilisation des palettes de couleur est possible mais elle induit une compression avec pertes.

4.1. Codage RLE (Run Length Encoding)

Cette méthode est utilisée par les formats RASTER : BMP (BITMAP) et GIF. Sa popularité est surtout due à la simplicité de sa mise en pratique. Cette méthode de codage n'est pas spécifique aux images : elle est applicable à n'importe quel type de données. Mais son efficacité dépend beaucoup de la nature des données à compresser. Pour simplifier, on la présente d'abord comme une méthode de compression de texte.

On recherche dans le texte à compresser des séquences de caractères répétés et on les remplace par des couples (N, c) où N est le nombre de répétitions et c est le caractère à répéter.

Par exemple, la séquence de 8 caractères identiques *eeeeeeee* sera codée par $(8e)$.
Et le texte de 17 caractères *tttaaattaarrrrbbb* sera codé par $3t3a2t2a4r3b$.

Dans ce dernier exemple, si chaque nombre de répétitions et chaque caractère sont codés sur 8 bits, on obtient ainsi un code de 12×8 = 96 bits, au lieu de 17×8 = 136 bits pour le texte initial en ASCII (8 bits).

Il est juste de remarquer que dans un texte intelligible en français ou en anglais il y a très peu de répétitions de caractères. Or, le codage RLE réduit l'espace mémoire à partir de 3 caractères identiques, le maintient inchangé pour une séquence de 2 caractères répétés et l'augmente pour un seul caractère (isolé). Ainsi la phrase '*ma jolie phrase*' aura un code RLE deux fois plus long que le code ASCII car aucun caractère ne se répète de façon successive.

Pour les images, cette méthode est au contraire très intéressante dans les régions peu contrastées et de faible dynamique de luminance. Car dans les zones, même petites, de couleur constante, plusieurs pixels se succèdent avec les mêmes valeurs de luminance. Par exemple, une image de damier en noir et blanc, aura un taux de compression très élevé. Le codage se fait alors ligne par ligne, en partant du coin en haut à gauche.

4.2. Codage différentiel

Cette technique est souvent utilisée comme prétraitement avant une méthode de compression. On suppose que dans une image les valeurs de pixels sont liées et qu'il est possible de prédire, approximativement, la valeur d'un pixel à partir de ses voisins déjà traités. Dans un parcours d'image ligne par ligne, en commençant par le coin en haut à gauche, pour le pixel courant on connaît les valeurs de ses voisins de la ligne au-dessus (HG - haut-gauche -, HC - haut-centre - et HD - haut-droit). Sur la même ligne on connaît le pixel précédent, P. Alors, en prenant en compte de tous ou d'une partie de ses voisins, on peut estimer, prédire, la valeur du pixel courant à l'aide de la moyenne des voisins. Il est donc inutile de coder cette information, puisque elle peut être restituée. Il suffit juste de coder l'erreur de prédiction (différence entre la valeur prédite et la valeur réelle du pixel).
Généralement, on obtient de très bons taux de compression, en couplant le codage différentiel avec une méthode de compression par dictionnaire. C'est en particulier le cas pour le format d'images PNG, par exemple.

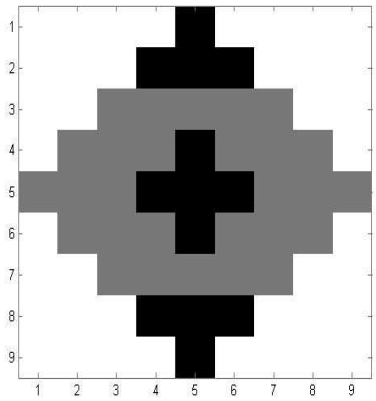
4.3. Méthodes mixtes

De nombreux formats de compression avec pertes utilisent le codage réversible (Huffman ou par dictionnaire) pour optimiser les taux de compression obtenus. Le format JPEG par exemple, utilise une transformée en cosinus discret (DCT) dont les coefficients subissent une procédure de quantification, et auxquels on applique ensuite un codage de Huffman pour réduire encore la place occupée par le code.

TD 4. Compression de données
Codage et compression d'images sans pertes

Exercice 1. Codages RLE (*Run Length Encoding*) et Huffman.

Nous allons étudier quelques applications de techniques de codage destinées à la compression sans pertes des images. Nous allons expérimenter sur la même image plusieurs approches de compression et en choisir la meilleure.
Voici l'image à analyser, de taille 9×9 pixels :



On suppose que chaque pixel de l'image est encodé sur 8 bits (1 Octet) ce qui donne une image en niveaux de gris avec une palette de 256 nuances de gris possibles, représentées par un entier compris entre 0 (Noir) et 255 (Blanc).
Or, cette image ne contient que 3 couleurs : Noir (N = 255), Gris (G = 128), Blanc (B = 0).
Si on représente chaque pixel par le caractère qui représente sa couleur on obtient la matrice

$$\begin{bmatrix} B & B & B & B & N & B & B & B & B \\ B & B & B & N & N & N & B & B & B \\ B & B & G & G & G & G & G & B & B \\ B & G & G & G & N & G & G & G & B \\ G & G & G & N & N & N & G & G & G \\ B & G & G & G & N & G & G & G & B \\ B & B & G & G & G & G & G & B & B \\ B & B & B & N & N & N & B & B & B \\ B & B & B & B & N & B & B & B & B \end{bmatrix}$$

On notera pour référence que, chaque pixel étant codé sur 8 bits, l'image occupe une taille de :
9×9×8 = 648 bits = 81 Octets.

Méthode 1. Codage de Huffman. Dans cet exercice nous allons appliquer à l'image le codage de Huffman en considérant les couleurs comme alphabet de la source. Pour toute lecture, l'image sera parcourue par exemple "ligne par ligne" en commençant par le coin en haut à gauche.

1. Constituer la *table des fréquences des couleurs présentes dans l'image*.
2. Calculer *l'entropie* associée. Que dire de l'efficacité du codage qui associe 8 bits à chaque pixel ? Peut-on trouver un codage plus économique pour cette image ?
3. A partir de la table des fréquences, déterminer un *arbre de Huffman* et le *code* associé.
4. Calculer la *taille du code de Huffman* (en bits) *pour cette image*. Quel est le *taux de compression* ?

Méthode 2. Codage RLE. Nous allons maintenant appliquer une autre technique à la même image : le codage RLE (*Run Length Encoding*). Cette méthode consiste à remplacer chaque séquence de caractères identiques par le couple (nb, c) où nb est le nombre de caractères de la séquence, et c le caractère répété. Par exemple, la séquence de couleurs *NNNBBGGG* sera codée par (3, *N*) (2, *B*) (3, *G*).

1. En parcourant l'image ligne par ligne (continument, sans interruption pour les sauts de ligne), constituer le *code RLE* correspondant. Commencer le codage par le coin en haut à gauche de l'image.
2. Chaque couple obtenu est encodé sur 16 bits : 8 pour le nombre de répétitions, et 8 pour le caractère répété. Quelle est la *longueur du code* (en bits) obtenu ?
3. Calculer le *taux de compression*.

Méthode 3. Codage RLE + Huffman. On reprend le code *RLE* obtenu dans l'exercice précédent. Certains couples (nb, c) se répètent et le nombre de couples réellement présents dans le code n'est pas élevé. On va utiliser le code de Huffman pour réduire la place en mémoire de chaque couple.

1. Constituer la *table des couples (nb, c) présents dans le code RLE de l'image*.
2. Calculer *l'entropie* associée. Que dire de l'efficacité du codage qui associe 16 bits à chaque couple ? Peut-on trouver un codage plus économique ?
3. A partir de la table des fréquences, déterminer un *arbre de Huffman* et le *code* associé.
4. Calculer la *taille du code de Huffman* (en bits) *pour la séquence du code RLE de l'image*. Quel est le *taux de compression* ?
5. Comparer les *performances des 3 méthodes* (*Huffman*, *RLE*, *RLE+Huffman*).

5. Modélisation mathématique d'un canal

1. Quelques notions utiles de probabilités

Soit X une variable aléatoire discrète à support fini définie sur un espace probabilisé fini. Une telle variable est définie par la donnée de son support (ensemble de valeurs possibles) $X \in \{x_i\}_{i=1}^{n}$ et par sa distribution de probabilité $P : \{x_i\}_{i=1}^{n} \to [0,1] \quad x_i \mapsto p(x_i) = P[X = x_i]$ telle que $\forall i = 1 \ldots n \quad 0 \le p(x_i) \le 1$ et $\sum_{i=1}^{n} p(x_i) = 1$.

▪ Distribution conjointe de probabilités

A un couple de variables aléatoires X à valeurs dans $\{x_i\}_{i=1}^{n}$ et Y à valeurs dans $\{y_j\}_{j=1}^{m}$ on associe la **distribution conjointe de probabilités** :

$$P_{XY} : \{(x_i, y_j) \quad (i,j) \in [1,n]\text{x}[1,m]\} \to [0,1] \quad (x_i, y_j) \mapsto p(x_i, y_j) = P(X = x_i \text{ et } Y = y_j)$$

On a les propriétés suivantes : $\forall (i,j) \in [1,n]\text{x}[1,m] \quad 0 \le p(x_i, y_j) \le 1$ et $\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j) = 1$

▪ Distributions marginales de probabilités

Etant donnée la distribution conjointe $p(x_i, y_j)$ d'un couple de variables aléatoires (X, Y), on définit les **distributions marginales de *X*** (P_X ou encore $P(X)$) **et de *Y*** (P_Y ou encore $P(Y)$) respectivement par les relations :

$$\begin{cases} P_X = P(X) : \{x_i\}_{i=1}^{n} \to [0,1] \quad x_i \mapsto p(x_i) = \sum_{j=1}^{m} p(x_i, y_j) \\ P_Y = P(Y) : \{y_j\}_{j=1}^{m} \to [0,1] \quad y_j \mapsto p(y_j) = \sum_{i=1}^{n} p(x_i, y_j) \end{cases}$$

▪ Distributions conditionnelles de probabilités

Soient X et Y deux variables aléatoires discrètes. On associe à l'événement $Y = y_j$ les **distributions conditionnelles** $P(X|y_j)$ (probabilité de *X* sachant y_j, ou encore probabilité de *X* si y_j) et $P(Y|x_i)$ (probabilité de *Y* sachant x_i) définies par

$$\begin{cases} P(X|y_j) : \; x_i \mapsto p(x_i|y_j) = P(X = x_i | Y = y_j) \\ P(Y|x_i) : \; y_j \mapsto p(y_j|x_i) = P(Y = y_j | X = x_i) \end{cases}$$

2. Définitions et propriétés essentielles

Définition 5.1. (Entropie conjointe). Soient $X = \{x_i\}_{i=1}^{n}$ et $Y = \{y_j\}_{j=1}^{m}$ deux variables aléatoires discrètes définies sur un même univers. Soit $p(x_i, y_j) = P(X = x_i \text{ et } Y = y_j)$ et encore noté $p(i, j)$ leur distribution conjointe. Alors l'**entropie conjointe** de X et Y est définie par

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\log_2\left(p(x_i, y_j)\right) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(i, j)\log_2\left(p(i, j)\right)$$

Cette définition peut être généralisée à un nombre quelconque r de variables aléatoires $X_1, X_2, \ldots, X_r$ à travers leur distribution de probabilité conjointe : $p(i_1, i_2, \ldots, i_r) = P[X_{i_1} = x_{i_1} \text{ et } X_{i_2} = x_{i_2} \cdots \text{ et } X_{i_r} = x_{i_r}]$:

$$H(X_1, X_2, \ldots, X_r) = -\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_r=1}^{n_r} p(i_1, i_2, \cdots, i_r)\log_2\left(p(i_1, i_2, \cdots, i_r)\right)$$

Cette notion, que nous introduisons ici dans un contexte volontairement plus général de variables aléatoires, peut être interprétée de différentes façons selon l'application. Voici quelques exemples.

Exemple 5.1. Etude des associations de symboles d'une source. Soit une source S d'alphabet $\Omega = \{a, e, o, c, p, r, t\}$. Supposons que tous les symboles soient équiprobables. L'entropie de cette source est alors

$$H(S) = \log_2(7) \approx 2.8 \text{ bits/symbole}$$

Pour coder chaque symbole d'un message on ne peut pas utiliser en moyenne moins de 2.8 bits par symbole.
On peut se poser la question de distribution de différentes associations possibles de symboles entre eux. Par exemple, on peut étudier les syllabes formées d'une consonne et d'une voyelle.

Soit X la variable aléatoire (v.a.) associée aux voyelles $\{a, e, o\}$ et Y la v.a. associée à l'émission de consonnes $\{c, p, r, t\}$. On s'intéresse aux probabilités d'apparition simultanée d'une de ces voyelles et d'une de ces consonnes dans une même syllabe. Supposons que ces *probabilités* sont données par le tableau suivant :

$Y \setminus X$	a	e	o	P_Y
c	0.05	0.05	0.1	0.2
p	0.1	0.15	0.05	0.3
r	0.05	0.1	0.1	0.25
t	0.15	0.05	0.05	0.25
P_X	0.35	0.35	0.3	

L'entropie conjointe de X et Y est alors : $H(X,Y) = -[6 \cdot 0.05\log_2(0.05) + 4 \cdot 0.1\log_2(0.1) + 2 \cdot 0.15\log_2(0.15)] \approx 3.44$ bits/symbole.

On peut interpréter cette quantité comme l'entropie d'une source $Z = (X, Y)$ dont l'alphabet est formé de couples de symboles *(consonne, voyelle)*. Nous remarquons alors que pour coder chaque couple il suffirait en moyenne 3.44 bits par couple au lieu de 2 × 2.8 = 5.6 bits, si l'on codait chaque symbole du couple séparément.

Exemple 5.2. Etude d'un couple émetteur-récepteur. Dans la suite nous allons étudier en détail les problèmes de transmission de messages émis par une source d'information via un canal de transmission. Nous allons donc associer à la source, d'alphabet $\Omega_X = \{x_1, \cdots, x_n\}$ une variable aléatoire X dont les valeurs sont les symboles émis, et au récepteur, d'alphabet $\Omega_Y = \{y_1, \cdots, y_m\}$, une variable aléatoire Y dont les valeurs sont les symboles reçus. Alors la *distribution conjointe* de X et Y décrit le canal de transmission et son *entropie* est une caractéristique importante du canal.

Définition 5.2. Entropie conditionnelle moyenne. Soient $X=\{x_i\}_{i=1}^n$ et $Y=\{y_j\}_{j=1}^m$ deux variables aléatoires discrètes définies sur un même univers. Soit $p(x_i,y_j)=P(X=x_i \text{ et } Y=y_j)$ et encore noté $p(i,j)$ leur *distribution conjointe*. Posons $p(i|j)=P[X=x_i \mid Y=y_j]=\frac{p(i,j)}{P(Y=y_j)}$. Alors l'**entropie conditionnelle moyenne** de X sachant Y est définie par

$$H(X|Y)=-\sum_{i=1}^{n}\sum_{j=1}^{m}p(i,j)\log_2(p(i|j))=-\sum_{i=1}^{n}\sum_{j=1}^{m}p(i,j)\log_2\left(\frac{p(i,j)}{P(Y=y_j)}\right)$$

Remarque 5.1. Il est utile d'expliciter le terme "*entropie conditionnelle moyenne*". Supposons que $Y=y_j$ et considérons la *distribution conditionnelle* $P(X|y_j)$ de X sachant $Y=y_j$ définie par :

$$P(X=x_i \mid y_j)=P(X=x_i \mid Y=y_j) \qquad i=1,\ldots,n$$

On lui associe *l'entropie conditionnelle* de X sachant $Y=y_j$ comme *l'information conditionnelle moyenne* de X sachant $Y=y_j$:

$$H(X|y_j)=-\sum_{i=1}^{n}p(x_i|y_j)\log_2(p(x_i|y_j))=-\sum_{i=1}^{n}\frac{p(x_i,y_j)}{p(y_j)}\log_2(p(x_i|y_j))$$

Alors *l'entropie moyenne* de X sachant Y est définie par :

$$H(X|Y)=\sum_{j=1}^{m}p(y_j)H(X|y_j)=-\sum_{j=1}^{m}p(y_j)\sum_{i=1}^{n}\frac{p(x_i,y_j)}{p(y_j)}\log_2(p(x_i|y_j))=-\sum_{i=1}^{n}\sum_{j=1}^{m}p(i,j)\log_2(p(i|j))$$

On appelle souvent $H(X)$ *l'entropie a priori* de X et $H(X|Y)$ *l'entropie (moyenne) a posteriori* de X.

Proposition 5.1. Additivité. *L'entropie conjointe* de deux variables aléatoires X et Y est égale à la somme de *l'entropie* de l'une d'elles et de *l'entropie conditionnelle moyenne* de l'autre.

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

Corollaire 5.1.

$$H(X,Y|Z)=H(X|Z)+H(Y|X,Z)$$

Proposition 5.2. Règles de chaînage. Soient $(X_i)_{i=1}^r$ r variables aléatoires discrètes. Alors :

1. La loi de *probabilités conjointes* de $(X_i)_{i=1}^r$ peut être développée comme suit :

$$P(X_1,X_2,\ldots,X_r)=P(X_1)P(X_2|X_1)P(X_3|X_2,X_1)\cdots P(X_r|X_{r-1},\cdots,X_1)=\prod_{k=1}^{r}P(X_k|X_{k-1},\cdots,X_1)$$

2. *L'entropie conjointe* de $(X_i)_{i=1}^r$ vérifie :

$$H(X_1,X_2,\ldots,X_r)=H(X_1)+H(X_2|X_1)+H(X_3|X_2,X_1)+\cdots+H(X_r|X_{r-1},\cdots,X_1)=\sum_{k=1}^{r}H(X_k|X_{k-1},\cdots,X_1)$$

Proposition 5.3. Propriétés.

1. $H(X,Y)\geq 0, \qquad H(X|Y)\geq 0$

L'entropie conjointe $H(X,Y)$ est nulle si et seulement si une seule des combinaisons (x_i, y_j) est possible. *L'entropie conditionnelle moyenne* $H(X|Y)$ est nulle si et seulement si X est une fonction de Y.

2. $H(X,Y)\geq \max(H(X), H(Y))$

3. $H(X,Y)\leq H(X)+H(Y)\leq 2H(X+Y)$

3. Information mutuelle moyenne

Définition 5.3. Information mutuelle moyenne. Soient $(X_i)_{i=1}^{r}$ et $Y=\{y_j\}_{j=1}^{m}$ deux variables aléatoires discrètes définies sur un même univers. Soit $p(i,j)=p(x_i,y_j)=P(X=x_i \text{ et } Y=y_j)$ leur *distribution conjointe*. *L'information mutuelle moyenne* de X et Y est définie par :

$$I(X;Y)=\sum_{i=1}^{n}\sum_{j=1}^{m} p(i,j)\log_2\left(\frac{p(i,j)}{p(x_i)p(y_j)}\right)$$

Remarque 5.2. A partir des définitions données ci-dessus on déduit facilement ces relations :

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

Dans le cas où les variables X et Y représentent respectivement le message émis et le message reçu par le destinataire, cette relation signifie que *l'information mutuelle moyenne* est égale :

- à l'information émise $H(X)$, diminuée de l'incertitude sur le symbole *x* émis quand le symbole *y* reçu est connu, $H(X|Y)$;
- et de façon symétrique, à l'information reçue, diminuée de l'incertitude sur le symbole reçu y quand le symbole émis *x* est connu, $H(Y|X)$.

4. Description mathématique d'une communication

Un canal de transmission peut être vu comme un système qui reçoit en entrée des symboles émis par une source d'alphabet $\Omega_X=\{x_i\}_{i=1}^{n}$ et qui donne en sortie des symboles de l'alphabet du récepteur $\Omega_Y=\{y_j\}_{j=1}^{m}$ (éventuellement différent de Ω_X). Lors de la transmission, des erreurs peuvent se produire de façon aléatoire. Ainsi, le lien entre un symbole émis et un symbole reçu est incertain. On peut alors parler de *canal avec bruit*.
Dans ce qui suit, nous ne nous intéressons pas à la nature de ce bruit ni à ses causes possibles. *La théorie de l'information s'intéresse aux conséquences de l'incertitude introduite par le bruit sur l'exactitude de l'information reçue à la sortie d'un canal de transmission*.

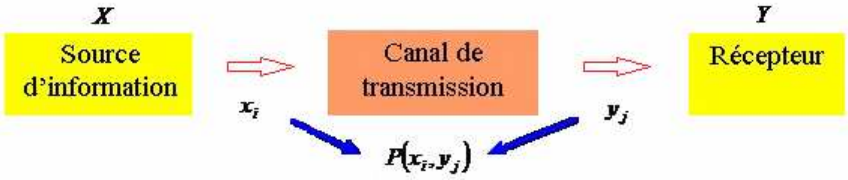


Figure 5.1. *Canal de transmission*

Quelles sont les mesures de quantité d'information qui peuvent être associées à l'ensemble "*source-canal-récepteur*" ?

Commençons par un *exemple simple*.

Soient une source binaire X d'alphabet $\Omega = \{0,1\}$ et de distribution de probabilité uniforme : $p(0) = p(1) = 0.5$. Soit un récepteur Y de même alphabet. Imaginons, qu'à chaque symbole émis, le canal de transmission a la probabilité $\frac{1}{4}$ de faire une erreur. On peut représenter alors le fonctionnement de ce canal par le schéma suivant :

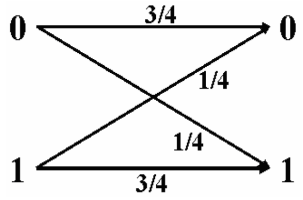


Figure 5.2. *Canal de transmission*

Soient X et Y les variables aléatoires associées respectivement à la source et au récepteur. La probabilité de l'erreur donnée permet d'établir les *probabilités conditionnelles* suivantes :

$$P(Y=0|X=0)=\frac{3}{4} \qquad P(Y=1|X=0)=\frac{1}{4}$$

$$P(Y=0|X=1)=\frac{1}{4} \qquad P(Y=1|X=1)=\frac{3}{4}$$

En connaissant également la *distribution de probabilité de la source* X on peut en déduire la *distribution conjointe*.
En effet :

$$P_{XY}(0,0)=P(X=0)P(Y=0|X=0)=\frac{3}{4} \qquad P_{XY}(0,1)=P(X=0)P(Y=1|X=0)=\frac{1}{8}$$

$$P_{XY}(1,0)=P(X=1)P(Y=0|X=1)=\frac{1}{8} \qquad P_{XY}(1,1)=P(X=1)P(Y=1|X=1)=\frac{3}{8}$$

Enfin, on peut calculer les *distributions marginales* de X et Y :

$$P_X(0)=P_X(1)=\frac{1}{2} \qquad P_Y(0)=P_Y(1)=\frac{3}{8}+\frac{1}{8}=\frac{1}{2}$$

A partir de ces différentes distributions de probabilités, on peut maintenant calculer les *entropies* $H(X)$, $H(Y)$, $H(X,Y)$, $H(X|Y)$ et enfin *l'information mutuelle* $I(X;Y)$:

$$H(X)=H(Y)=1 \text{ bit/symbole.}$$

L'entropie de chacune des variables X et Y décrit la difficulté moyenne de prédire le symbole émis (ou reçu).

$$H(X,Y)=-\left(2\cdot\frac{1}{8}\cdot\log_2\left(\frac{1}{8}\right)+2\cdot\frac{3}{8}\cdot\log_2\left(\frac{3}{8}\right)\right)\approx 1.8 \text{ bits/symbole.}$$

L'entropie mutuelle décrit l'information moyenne apportée par l'observation d'un couple de symboles : *x* (le symbole émis) et *y* (le symbole reçu).

$$H(Y|X)=-\left(2\cdot\frac{1}{8}\cdot\log_2\left(\frac{1}{4}\right)+2\cdot\frac{3}{8}\cdot\log_2\left(\frac{3}{4}\right)\right)\approx 0.8 \text{ bit/symbole.}$$

L'entropie conditionnelle exprime la difficulté moyenne de prévoir le symbole reçu lorsqu'on connaît celui qui a été émis.

$$I(X;Y)=H(X)-H(X|Y)=I(Y;X)=H(Y)-H(Y|X)\approx 1-0.8=0.2 \text{ bit/symbole.}$$

L'information mutuelle moyenne de X et Y.

5. Canal discret stationnaire, sans mémoire. Entropie conjointe - Entropie conditionnelle

Supposons que l'alphabet de la source, en entrée, est composé de n symboles $\Omega_X = \{x_i\}_{i=1}^{n}$ et que l'alphabet du récepteur, en sortie, est composé de m symboles $\Omega_Y = \{y_j\}_{j=1}^{m}$.

Dans la suite, nous allons considérer seulement un cas particulier de canaux de communication, **les canaux discrets sans mémoire et stationnaires**. Un *canal discret sans mémoire et stationnaire* est décrit par la donnée de la *matrice de probabilités conditionnelles*, appelée **matrice de transition**

$$p_{i,j} = P(y_j \mid x_i) \quad i = 1,\ldots,n \quad j = 1,\ldots,m$$

Cette matrice décrit les propriétés du bruit dans le canal.

▪ Le terme "*sans mémoire*" signifie que la réception à tout instant d'un symbole Y ne dépend que du symbole émis X. En particulier, chaque symbole reçu est indépendant des symboles reçus précédemment.

▪ Le terme "*stationnaire*" signifie que les caractéristiques probabilistes du bruit sont indépendantes du temps. Ainsi, à tout instant de transmission, cette matrice est la même.

En pratique, pour décrire les caractéristiques du bruit d'un canal, on dispose souvent de la *matrice de transition* $P_{Y|X} : n \times m$, c'est à dire de la *distribution conditionnelle de la sortie sachant l'entrée*. Dans ce cas, on connaît également la *distribution de probabilité de la source* $P_X = \{p(x_i)\}_{i=1}^{n}$. En connaissant ces deux distributions on peut déduire toutes les autres distributions associées au couple (X,Y) de l'émetteur et du récepteur :

▪ *Distribution de probabilité conjointe :* $p(x_i, y_j) = p(x_i)p(y_j \mid x_i) \quad i = 1,\ldots,n \quad j = 1,\ldots,m$

▪ *Distribution marginale de Y :* $p(y_j) = \sum_{i=1}^{n} p(y_j \mid x_i)p(x_i) \quad j = 1,\ldots,m$

▪ *Distribution conditionnelle $P(X \mid Y)$:* $p(x_i \mid y_j) = \dfrac{p(x_i, y_j)}{p(y_j)} \quad i = 1,\ldots,n \quad j = 1,\ldots,m$

Lorsque toutes ces distributions sont connues, on peut associer à un système de communication "*source - canal - récepteur*" différentes *entropies* :

▪ $H(X)$. **L'entropie de la source.** Elle représente l'information moyenne par symbole de la source ou encore la difficulté moyenne de prédire le symbole émis.

▪ $H(Y)$. **L'entropie du récepteur.** Elle représente l'information moyenne par symbole reçu ou encore la difficulté moyenne de prédire le symbole reçu.

▪ $H(X,Y)$. **L'entropie conjointe "source-récepteur".** Elle représente l'incertitude moyenne du système de communication dans son ensemble, ou encore la quantité de l'information moyenne par paire "symbole émis - symbole reçu".

▪ $H(X \mid Y)$. **L'entropie conditionnelle de la source, sachant le symbole reçu.** Elle représente l'incertitude moyenne sur le symbole émis lorsqu'on connaît le symbole reçu.

▪ $H(Y \mid X)$. **L'entropie conditionnelle du récepteur, sachant le symbole émis.** Elle représente l'incertitude moyenne sur le symbole reçu lorsqu'on connaît le symbole émis.

▪ $I(X;Y)$. **L'information mutuelle moyenne.** Elle représente la quantité moyenne d'information par symbole transmise à travers le canal.

5.1. Exemple complet

Soient une source d'un alphabet de 5 symboles $\Omega_X = \{x_i\}_{i=1}^{n=5} = \{x_1, x_2, x_3, x_4, x_5\}$ et un récepteur d'alphabet ayant quatre symboles $\Omega_Y = \{y_j\}_{j=1}^{m=4} = \{y_1, y_2, y_3, y_4\}$. Supposons que la *matrice de probabilités conjointes* associée à un canal est connue :

$P(X,Y) =$

$X \setminus Y$	y_1	y_2	y_3	y_4
x_1	**0.25**	**0**	**0**	**0**
x_2	**0.1**	**0.3**	**0**	**0**
x_3	**0**	**0.05**	**0.1**	**0**
x_4	**0**	**0**	**0.05**	**0.1**
x_5	**0**	**0**	**0.05**	**0**

Calculons toutes les autres *distributions de probabilités* associées :

▪ **Distribution marginale de la source** $P_X = P(X)$. On utilise la définition avec $n = 5$ et $m = 4$

$$p(x_i) = \sum_{j=1}^{m} p(x_i, y_j) \quad i = 1, \ldots, n$$

Ainsi, en faisant les sommes des éléments de **chaque ligne** de la matrice $P(X,Y)$ on trouve :

$$p(x_1) = 0.25$$
$$p(x_2) = 0.1 + 0.3 = 0.4$$
$$p(x_3) = 0.05 + 0.1 = 0.15$$
$$p(x_4) = 0.05 + 0.1 = 0.15$$
$$p(x_5) = 0.05$$

▪ **Distribution marginale du récepteur** $P_Y = P(Y)$. On utilise la définition avec $n = 5$ et $m = 4$

$$p(y_j) = \sum_{i=1}^{n} p(x_i, y_j) \quad j = 1, \ldots, m$$

Ainsi, en faisant les sommes des éléments de **chaque colonne** de la matrice $P(X,Y)$ on trouve :

$$p(y_1) = 0.25 + 0.1 = 0.35$$
$$p(y_2) = 0.3 + 0.05 = 0.35$$
$$p(y_3) = 0.1 + 0.05 + 0.05 = 0.2$$
$$p(y_4) = 0.1$$

$P(X,Y) =$

$X \setminus Y$	y_1	y_2	y_3	y_4	P_X
x_1	0.25	0	0	0	**0.25**
x_2	0.1	0.3	0	0	**0.4**
x_3	0	0.05	0.1	0	**0.15**
x_4	0	0	0.05	0.1	**0.15**
x_5	0	0	0.05	0	**0.05**
P_Y	**0.35**	**0.35**	**0.2**	**0.1**	

▪ **Distribution conditionnelle** $P(X|Y)$. On utilise la définition avec $n=5$ et $m=4$

$$p(x_i \mid y_j)=\frac{p(x_i,y_j)}{p(y_j)} \quad i=1,\ldots,n \quad j=1,\ldots,m \qquad \rightarrow \qquad P(X|Y)=\begin{pmatrix} 5/7 & 0 & 0 & 0 \\ 2/7 & 6/7 & 0 & 0 \\ 0 & 1/7 & 1/2 & 0 \\ 0 & 0 & 1/4 & 1 \\ 0 & 0 & 1/4 & 0 \end{pmatrix}$$

▪ **Distribution conditionnelle** $P(Y|X)$. On utilise la définition avec $n=5$ et $m=4$

$$p(y_j \mid x_i)=\frac{p(x_i,y_j)}{p(x_i)} \quad i=1,\ldots,n \quad j=1,\ldots,m \qquad \rightarrow \qquad P(Y|X)=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1/4 & 3/4 & 0 & 0 \\ 0 & 1/3 & 2/3 & 0 \\ 0 & 0 & 1/3 & 2/3 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Calculons maintenant les *entropies*.

▪ $H(X)$. **L'entropie de la source.** On utilise la définition avec $n=5$

$$H(X)=-\sum_{i=1}^{n} p(x_i)\log_2[p(x_i)] = -0.25\log_2(0.25)-0.4\log_2(0.4)-2\cdot 0.15\log_2(0.15)-0.05\log_2(0.05)\approx 2.066 \text{ bits/symbole.}$$

▪ $H(Y)$. **L'entropie du récepteur.** On utilise la définition avec $m=4$

$$H(Y)=-\sum_{i=1}^{m} p(y_i)\log_2[p(y_i)] = -2\cdot 0.35\log_2(0.35)-0.2\log_2(0.2)-0.1\log_2(0.1)\approx 1.856 \text{ bits/symbole.}$$

▪ $H(X,Y)$. **L'entropie conjointe "source-récepteur".** On utilise la définition avec $n=5$ et $m=4$

$$H(X,Y)=-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log_2[p(x_i,y_j)] = -0.25\log_2(0.25)-3\cdot 0.1\log_2(0.1)-3\cdot 0.05\log_2(0.05)-0.3\log_2(0.3)\approx 2.665 \text{ bits/symbole.}$$

▪ $H(X|Y)$. **L'entropie conditionnelle de la source, sachant le symbole reçu.** On utilise la définition avec $n=5$ et $m=4$

$$H(X|Y)=-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log_2\left[\frac{p(x_i,y_j)}{p(y_j)}\right]=-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log_2\left[p(x_i|y_j)\right]$$

$$H(X|Y)=-0.25\log_2\left(\frac{5}{7}\right)-0.1\left(\log_2\left(\frac{2}{7}\right)+\log_2\left(\frac{1}{2}\right)\right)-0.05\left(\log_2\left(\frac{1}{7}\right)+2\cdot\log_2\left(\frac{1}{4}\right)\right)-0.3\log_2\left(\frac{6}{7}\right)\approx 0.809 \text{ bit/symbole.}$$

▪ $H(Y|X)$. **L'entropie conditionnelle du récepteur, sachant le symbole émis.** On utilise la définition avec $n=5$ et $m=4$

$$H(Y|X)=-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log_2\left[\frac{p(x_i,y_j)}{p(x_i)}\right]=-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log_2\left[p(y_j|x_i)\right]$$

$$H(Y|X)=-0.1\left(\log_2\left(\frac{1}{4}\right)+2\cdot\log_2\left(\frac{2}{3}\right)\right)-2\cdot 0.05\log_2\left(\frac{1}{3}\right)-0.3\log_2\left(\frac{3}{4}\right)=0.6 \text{ bit/symbole.}$$

5.2. Exemples fréquents de canaux

Il existe quelques classes de canaux plus faciles à analyser. Nous en donnons ici quelques exemples (tirés de [4] et [2]).

5.2.1. Canal avec entrée et sortie indépendantes

Il s'agit d'un canal d'alphabets d'entrée et de sortie respectivement $\Omega_X = \{x_i\}_{i=1}^{n}$ et $\Omega_Y = \{y_j\}_{j=1}^{m}$ tel que, quel que soit le symbole émis, on peut recevoir n'importe quel symbole y_j avec équiprobabilité :

$$p(y_j \mid x_i) = p(y_j) = \frac{1}{m} \qquad i = 1,\ldots,n \quad j = 1,\ldots,m$$

Ainsi sa *matrice de transition* est de la forme :

$$P(Y|X) = \begin{pmatrix} 1/m & 1/m & \cdots & 1/m \\ 1/m & 1/m & \cdots & 1/m \\ \vdots & \vdots & \ddots & \vdots \\ 1/m & 1/m & \cdots & 1/m \end{pmatrix}$$

Etant donnée la distribution de probabilité de la source : $P(x_i) = p_i \qquad i = 1,\ldots,n$ avec $\sum_{i=1}^{n} p_i = 1,$

on en déduit les *probabilités conjointes* :

$$p(x_i, y_j) = p(x_i \cap y_j) = p(x_i)p(y_j|x_i) = p(x_i)p(y_j) = \frac{p_i}{m} \overset{\Delta}{=} q_i \qquad i = 1,\ldots,n \quad j = 1,\ldots,m$$

La matrice de *probabilités conjointes* a alors m colonnes identiques :

$$P(X,Y) = \begin{pmatrix} q_1 & q_1 & \cdots & q_1 \\ q_2 & q_2 & \cdots & q_2 \\ \vdots & \vdots & \ddots & \vdots \\ q_n & q_n & \cdots & q_n \end{pmatrix}$$

La *distribution conditionnelle de la source sachant le récepteur* se calcule comme suit :

$$p(x_i|y_j) = \frac{p(x_i, y_j)}{p(y_j)} = \frac{p(x_i)p(y_j)}{p(y_j)} = p(x_i) = p_i = mq_i \qquad i = 1,\ldots,n \quad j = 1,\ldots,m$$

Pour les *entropies* on a :

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\log_2\left[p(x_i, y_j)\right] = -m\sum_{i=1}^{n} q_i \log_2(q_i)$$

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2\left[p(x_i)\right] = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

$$H(Y) = -\sum_{i=1}^{m} p(y_i)\log_2\left[p(y_i)\right] = \log_2(m)$$

$$H(X|Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\log_2\left[\frac{p(x_i, y_j)}{p(y_j)}\right] = -\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{p_i}{m}\log_2(p_i) = -\sum_{i=1}^{n} p_i \log_2(p_i) = H(X)$$

$$H(Y|X) = H(Y) + H(X|Y) - H(X) = H(Y) = \log_2(m)$$

Et enfin, en ce qui concerne *l'information mutuelle moyenne* :

$$I(X;Y) = H(X) - H(X|Y) = H(X) - H(X) = 0$$

L'interprétation de ces formules est la suivante : un tel canal ne transporte aucune information entre la source et le récepteur ($I(X;Y)=0$). Si un canal sans bruit peut être considéré comme le cas idéal d'un canal sans pertes, alors un canal à entrée et sortie indépendantes est considéré comme ayant la perte maximale d'informations.

5.2.2. Canal sans pertes

Soit, comme ci-dessus, X la variable aléatoire associée à la source et Y la variable aléatoire associée au message reçu. On dit que le *canal* est *sans pertes* si $H(X|Y)=0$. Autrement dit, le symbole envoyé est identifié sans aucune ambiguïté par le symbole reçu. Si l'alphabet de la source est $\Omega_X=\{x_i\}_{i=1}^{n}$ et si celui du récepteur est $\Omega_Y=\{y_j\}_{j=1}^{m}$ avec $m>n$ alors dans un canal sans pertes il est possible de partitionner l'alphabet Ω_Y en n sous-ensembles disjoints $B_i \quad i=1,\cdots,n$ tels que

$$P(x=x_i|y\in B_i)=1$$

Dans ce cas, nous avons :

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

5.2.3. Canal déterministe

On dit que le *canal* est *déterministe* si $H(Y|X)=0$. Autrement dit, le symbole reçu est déterminé de façon unique par le symbole envoyé. Si l'alphabet de la source est $\Omega_X=\{x_i\}_{i=1}^{n}$ et si celui du récepteur est $\Omega_Y=\{y_j\}_{j=1}^{m}$, on a $p(y_j|x_i)$ égal à 0 ou 1. Dans ce cas, nous avons

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

5.2.4. Canal sans bruit

L'absence de bruit dans un canal signifie que la transmission est exacte, sans erreurs. un *canal sans bruit* est donc un canal *sans pertes* et *déterministe*. Le canal peut alors être vu comme une mise en correspondance biunivoque (bijective) des deux alphabets (ayant la même taille $m=n$). Cela se traduit par une *matrice de transition identité* :

$$P(Y|X)=I_n=\begin{pmatrix}1 & 0 & \cdots & 0\\ 0 & 1 & \vdots & 0\\ \vdots & \cdots & \ddots & \vdots\\ 0 & 0 & \cdots & 1\end{pmatrix}$$

La *matrice de probabilités conjointes* est alors *diagonale*

$$P(X,Y)=\begin{pmatrix}p(x_1,y_1) & 0 & \cdots & 0\\ 0 & p(x_2,y_2) & \vdots & 0\\ \vdots & \cdots & \ddots & \vdots\\ 0 & 0 & \cdots & p(x_n,y_n)\end{pmatrix}$$

Il est facile de vérifier les *entropies* :

$$H(X,Y)=H(X)=H(Y)=-\sum_{i=1}^{n}p(x_i,y_i)\log_2[p(x_i,y_i)]$$

$$H(X|Y)=H(Y|X)=0$$

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

On peut facilement interpréter ces relations. En effet, dans un canal sans bruit, à chaque symbole émis correspond un et un seul symbole reçu. Alors lorsque l'on connaît le symbole émis, on connaît *sans aucune incertitude* le symbole reçu et vice versa. Ainsi, les *entropies conditionnelles moyennes* sont *nulles* car elles mesurent précisément l'incertitude moyenne sur le symbole émis (respectivement reçu) sachant le symbole reçu (respectivement émis). C'est aussi pour cette raison que les *incertitudes moyennes par symbole* à la source, $H(X)$, et à la réception, $H(Y)$, sont *les mêmes*.

5.2.5. Canal inutile

Un *canal* est *inutile* si $I(X|Y)=0$. Autrement dit, le canal ne transporte *aucune information*. Cette condition est équivalente à $H(X)=H(X|Y)$: la connaissance du symbole reçu ne modifie pas (n'apporte aucune information supplémentaire) l'incertitude sur le symbole émis.

5.2.6. Canal symétrique

Un *canal symétrique* est caractérisé par une propriété particulière de sa matrice de transition $P(X|Y)$. *Toutes les lignes de cette matrice sont équivalentes entre elles par permutation et toutes les colonnes également*. Cela veut dire *que toutes les lignes sont obtenues par permutation d'une seule d'entre elles, et toutes les colonnes sont obtenues par permutation d'une seule d'entre elles*. Voici un exemple d'une telle matrice :

$P(Y|X)=$

$X \setminus Y$	y_1	y_2	y_3
x_1	1/2	1/6	1/3
x_2	1/6	1/3	1/2
x_3	1/3	1/2	1/6

Proposition 5.4. Dans un *canal symétrique, l'entropie conditionnelle* $H(Y|X)$ *ne dépend pas de la distribution de probabilités de* X.

Preuve de la Proposition 5.4.

Soit $x_i \in \Omega_X \quad i \in [1,\cdots,n]$, un symbole quelconque de l'alphabet de la source. Soit $y_j \in \Omega_Y \quad j \in [1,\cdots,m]$, un symbole quelconque de l'alphabet du message reçu. Calculons

$$H(Y|X=x_i)=-\sum_{j=1}^{m} p(y_j|x_i)\log_2\left[p(y_j|x_i)\right]$$

Or pour un *canal symétrique* toutes les lignes de la matrice de transition sont obtenues par permutation à partir d'une seule d'entre elles. Ainsi, toutes les lignes de la matrice contiennent exactement le même ensemble de valeurs. Notons les $\alpha_1,\alpha_2\cdots,\alpha_m$. Nous avons alors pour tout $x_i \in \Omega_X$

$$H(Y|X=x_i)=-\sum_{j=1}^{m} \alpha_j \log_2(\alpha_j)$$

Ainsi la valeur de $H(Y|x_i)$ ne dépend pas de x_i. Alors on a :

$$H(Y|X)=\sum_{i=1}^{n} p(x_i)H(Y|x_i)=-\sum_{j=1}^{m}\alpha_j\log_2(\alpha_j)\sum_{i=1}^{n}p(x_i)=-\sum_{j=1}^{m}\alpha_j\log_2(\alpha_j) \qquad \left(=-\sum_{j=1}^{m}p(y_j|x_1)\right)$$

Dans cette classe de canaux, le plus connu est le canal symétrique binaire. Les deux alphabets, de source et de récepteur, sont identiques et composés de deux caractères : $\Omega_X=\Omega_Y=\{0,1\}$. La *matrice de transition* est alors égale à :

$$P(Y|X)=\begin{pmatrix}1-\alpha & \alpha\\ \alpha & 1-\alpha\end{pmatrix}$$

où $\alpha \in [0,1]$ est la probabilité d'erreur de transmission.

6. Capacité d'un canal

Nous avons donné plus haut la définition de *l'information mutuelle moyenne* $I(X;Y)$ comme mesure de *l'information transmise en moyenne par symbole à travers le canal*. On peut remarquer que cette quantité dépend du *canal* mais aussi de la *source d'information*. On introduit alors une nouvelle mesure, qui ne décrit que le canal : la *capacité de canal*.

Définition 5.4. Capacité d'un canal. La *capacité d'un canal* est définie par

$$C = \max_{P(X)} I(X;Y) = \max_{P(X)} \left(H(X) - H(X|Y) \right)$$

Le maximum est pris sur toutes les distributions de probabilité possibles de la source.

Exemple 5.3. Capacité d'un canal sans bruit. Considérons un canal sans bruit et une source d'alphabet $\Omega_X = \{x_i\}_{i=1}^n$. Nous avons déjà vu que dans ce cas $I(X;Y) = H(X)$. Alors

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

Or nous avons également vu que l'entropie d'une source est maximale lorsque tous les symboles de l'alphabet sont équiprobables. Sa valeur maximale est alors égale à $\log_2(n)$. Ainsi on en déduit que pour un canal sans bruit d'alphabet de n caractères :

$$C = \log_2(n)$$

6.1. Capacité d'un canal symétrique

Si dans le cas général le calcul de capacité peut s'avérer difficile, il est possible d'obtenir des expressions explicites dans les cas de quelques canaux particuliers que nous avons étudiés précédemment. Ici nous allons développer le cas très utile d'un canal symétrique.

Proposition 5.5. Soit un couple *émetteur-récepteur* (X, Y) avec les alphabets $\Omega_X = \{x_i\}_{i=1}^n$ et $\Omega_Y = \{y_j\}_{j=1}^m$. Soit un *canal symétrique* de *matrice de transition* donnée :

$$P(Y|X) = \begin{pmatrix} p(y_1|x_1) & p(y_2|x_1) & \cdots & p(y_{m-1}|x_1) & p(y_m|x_1) \\ p(y_1|x_2) & p(y_2|x_2) & \cdots & p(y_{m-1}|x_2) & p(y_m|x_2) \\ \vdots & \vdots & & \ddots & \vdots \\ p(y_1|x_n) & p(y_2|x_n) & \cdots & p(y_{m-1}|x_n) & p(y_m|x_n) \end{pmatrix}$$

Alors la *capacité de ce canal* est :

$$C = \log_2(m) + \sum_{j=1}^{m} p(y_j|x_1)$$

Preuve de la Proposition 5.5.

Nous allons utiliser la proposition **5.4.** qui établit une propriété remarquable de l'entropie conditionnelle $H(Y|X)$ dans un canal symétrique : $H(Y|X)$ ne dépend pas de la distribution de X. Alors si l'on cherche à maximiser l'information transmise par le canal

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

nous devons trouver la distribution de X qui maximise $H(Y)$ seulement puisque le deuxième terme de cette expression ne dépend pas de la distribution de X.
Nous savons, d'après les propriétés générales de l'entropie qu'elle atteint sa valeur maximale quand la variable aléatoire correspondante est distribuée uniformément. Montrons que si la source, X, est distribuée uniformément, alors la variable du récepteur, Y, est aussi distribuée uniformément et donc $H(Y)$ est maximale.

Supposons donc que $p(x_i) = \frac{1}{n} \quad \forall x_i \in \Omega_X$. Alors, en utilisant la règle de Bayes, on trouve :

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

Or, dans la matrice de transition d'un canal symétrique toutes les colonnes sont obtenues par permutation de l'une d'entre elles. Donc la somme :

$$\sum_{i=1}^{n} p(y_j | x_i) = \sum_{i=1}^{n} p(y_1 | x_i)$$

ne dépend pas de y_j . Ainsi on a montré que si la distribution de X est uniforme, toutes les probabilités $p(y_j)$ sont égales. Donc dans ce cas $H(Y)$ est maximale et égale à $\log_2(m)$.

La formule de la proposition **5.5.** en découle alors immédiatement. La proposition **5.4.** donne pour $H(Y|X)$:

$$H(Y|X) = -\sum_{j=1}^{m} p(y_j | x_1)$$

_____

TD 5. Modélisation mathématique d'un canal

Partie I. Représentation mathématique d'un canal de transmission

Exercice 1. Exemple. Soient X et Y, 2 variables aléatoires prenant leurs valeurs respectivement dans les alphabets $\Omega_X = \{x_1, x_2, x_3\}$ (de taille n = 3) et $\Omega_Y = \{y_1, y_2\}$ (de taille m = 2) et ayant la *matrice de probabilités conjointes* suivante :

$P(X,Y) =$

$X \setminus Y$	y_1	y_2
x_1	**0.25**	**0**
x_2	**0.1**	**0.3**
x_3	**0.1**	**0.25**

En déduire les *distributions conditionnelles* et *marginales*.

Exercice 1. Exemple. Corrigé.

▪ **Distributions marginales de X, $P_X = P(X)$ et Y, $P_Y = P(Y)$.** On utilise les définitions avec $n = 3$ et $m = 2$:

$$\begin{cases} P_X = P(X) : \{x_i\}_{i=1}^{n} \to [0,1] \quad x_i \mapsto p(x_i) = \sum_{j=1}^{m} p(x_i, y_j) \\ P_Y = P(Y) : \{y_j\}_{j=1}^{m} \to [0,1] \quad y_j \mapsto p(y_j) = \sum_{i=1}^{n} p(x_i, y_j) \end{cases}$$

On peut résumer ces calculs sous forme d'un tableau

$X \setminus Y$	y_1	y_2	P_X
x_1	0.25	0	**0.25**
x_2	0.1	0.3	**0.4**
x_3	0.1	0.25	**0.35**
P_Y	**0.45**	**0.55**	

La *dernière colonne* de ce tableau représente la *distribution marginale de* X et s'obtient en calculant la *somme des éléments de chaque ligne*.

La *dernière ligne* de ce tableau représente la *distribution marginale de* Y et s'obtient en calculant la *somme des éléments de chaque colonne*.

▪ **Distributions conditionnelles** $P(X|Y)$ et $P(Y|X)$. On utilise la définition avec $n = 3$ et $m = 2$:

$$p(x_i \mid y_j) = \frac{p(x_i, y_j)}{p(y_j)} \qquad p(y_j \mid x_i) = \frac{p(x_i, y_j)}{p(x_i)} \qquad i = 1, \ldots, n \quad j = 1, \ldots, m$$

On trouve les matrices :

$$P(X|Y) = \begin{pmatrix} 0.25/0.45 & 0/0.55 \\ 0.1/0.45 & 0.3/0.55 \\ 0.1/0.45 & 0.25/0.55 \end{pmatrix} = \begin{pmatrix} 5/9 & 0 \\ 2/9 & 6/11 \\ 2/9 & 5/11 \end{pmatrix} \qquad P(Y|X) = \begin{pmatrix} 0.25/0.25 & 0/0.25 \\ 0.1/0.4 & 0.3/0.4 \\ 0.1/0.35 & 0.25/0.35 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1/4 & 3/4 \\ 2/7 & 5/7 \end{pmatrix}$$

Exercice 2. Un modèle probabiliste de transmission. Soit une source binaire qui émet des signaux binaires, composés de 0 et de 1. On associé à l'expérience de l'envoi d'un seul symbole, une variable aléatoire X qui prend donc des valeurs dans $\Omega_X = \{0,1\}$ (alphabet de taille n = 2). On suppose que la source émet 0 ou 1 avec équiprobabilité. Autrement dit, la distribution (*marginale*) de X est connue (*uniforme*) : $P_X(0) = P_X(1) = 0.5$. Les symboles émis sont ensuite envoyés via un canal de transmission ayant des perturbations aléatoires. On associe à l'expérience d'observation du symbole reçu la variable aléatoire Y qui peut prendre trois valeurs : $\Omega_Y = \{-1,0,1\}$ (alphabet de taille m = 3). La valeur -1 correspond au cas où le système n'est pas capable d'identifier un 0 ou un 1 à la réception. On a les probabilités de réception suivantes :

- Si le symbole envoyé X est 0, alors on reçoit :
  - Y = 1 avec la probabilité 0.2
  - Y = 0 avec la probabilité 0.7
  - Y = -1 avec la probabilité 0.1
- Si le symbole envoyé X est 1, alors on reçoit :
  - Y = 1 avec la probabilité 0.6
  - Y = 0 avec la probabilité 0.3
  - Y = -1 avec la probabilité 0.1

1. Dire si les probabilités de réception données définissent la *distribution conjointe* ou *conditionnelle* de X et Y ?
Si *conditionnelle*, préciser de quelle variable ?
2. Peut-on donner une représentation sous forme de *graphe des probabilités de réception* ?
3. Former une *matrice* à partir des *probabilités de réception.*
4. Calculer *toutes les distributions* manquantes.
5. Calculer les *entropies* associées au *canal* : $H(X)$, $H(Y)$, $H(X,Y)$, $H(X|Y)$ et $H(Y|X)$.
6. En déduire *l'information mutuelle* $I(X;Y)$ du *canal*.

Exercice 3. Capacité de canal. Soit une source binaire X d'alphabet $\Omega_X = \{0,1\}$ (de taille n = 2). On suppose que la distribution de probabilité (*marginale*) de X est connue :

$$P_X(0) = p \qquad P_X(1) = 1 - p$$

Les symboles émis sont envoyés via un canal de transmission soumis à des perturbations aléatoires. On associe à l'expérience d'observation du symbole reçu la variable aléatoire Y qui peut prendre trois valeurs : $\Omega_Y = \{-1,0,1\}$ (alphabet de taille m = 3). La valeur -1 correspond au cas ou` le système n'est pas capable d'identifier un 0 ou un 1 à la réception. On suppose que la *matrice de transition du canal* est la suivante :

$P(Y|X) =$

$X \setminus Y$	0	-1	1
0	0.8	0.2	0
1	0	0.2	0.8

1. Est-ce un *canal symétrique* ?
2. En utilisant la *matrice de transition* $P(Y|X)$ et la distribution de probabilité (*marginale*) de X, P_X, calculer la *distribution de probabilité conjointe* $P(X,Y)$. En déduire P_Y, la *distribution marginale* du récepteur Y, et la *distribution conditionnelle* $P(X|Y)$.
3. Exprimer *l'entropie* $H_X(p)$ comme fonction du paramètre p.
4. Calculer $H(X|Y)$ en fonction du paramètre p.
5. Calculer $I(X;Y) = H_X(p) - H(X|Y)$. Montrer que $I(X;Y) = 0.8H_X(p)$.
6. En déduire la valeur de la capacité du canal définie par : $C = \max_{P_X} I(X;Y)$ où le maximum est pris selon toutes les distributions de probabilités possibles de la source X.

Partie II. Pour approfondir

Exercice 4. Nombre d'itérations d'un algorithme.

Soit l'algorithme suivant :

Entrée : $X_0 \in \{1,2,3,4\}$ tiré *aléatoirement* avec *équiprobabilité*.

Algorithme

$n \leftarrow 0$

Tant que ($X_n \neq 1$)

$$A_n \leftarrow \begin{cases} 0 \text{ avec probabilité } p_n = \dfrac{1}{X_n} \\ 1 \text{ avec probabilité } q_n = 1 - \dfrac{1}{X_n} \end{cases}$$

Si (X_n est pair) *alors*

$$X_{n+1} \leftarrow \frac{X_n}{2}$$

Sinon

$$X_{n+1} \leftarrow A_n X_n + 1$$

Fin Si

$n \leftarrow n + 1$

Fin Tant que

Sortie : **Nombre d'itérations** n.

Soient X, la variable aléatoire correspondant à l'entrée de l'algorithme, uniformément distribuée sur $\{1,2,3,4\}$ (alphabet de taille n = 4) et N, la variable aléatoire correspondant à la sortie de l'algorithme (alphabet $N = \{0,1,2,3\}$ de taille m = 4). L'objectif est de calculer le *nombre moyen d'itérations* effectuées par l'algorithme. Autrement dit, calculer $E[N]$. Voici les étapes à suivre :

1. Etablir la *matrice de probabilités conditionnelles* $P(N|X)$, en utilisant les schémas sous forme d'arbre pour le déroulement de l'algorithme pour chaque valeur de X (voir l'exemple sur la *figure* **2.**).

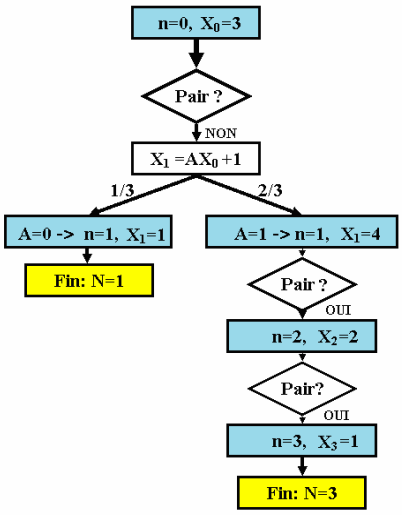


Figure 2. Cas où $X_0 = 3$.

2. Sachant que X est *uniformément distribuée*, déduire la *matrice de probabilités conjointes* de N et X.

3. Calculer la *distribution de probabilités marginale de* N.

4. En déduire le *nombre moyen d'itérations*.

———————

6. Codage de canal (Codage en présence de bruit). Codes correcteurs.

1. Second théorème de Shannon

1.1. Codage de canal

Dans cette partie nous allons poser et étudier le problème de codage dans le contexte de transmission via un canal avec bruit. La différence par rapport au codage sans bruit est dans le fait que lors de la transmission des messages, des erreurs peuvent se produire de façon aléatoire. Ces erreurs sont caractérisées par la donnée de la *matrice de transition du canal*.
Notre problème sera alors d'évaluer la probabilité d'erreur de transmission. Nous allons montrer que cette probabilité dépend non seulement des caractéristiques probabilistes du canal et de la source mais aussi du choix de codage et de décodage.
Alors le problème de choix de meilleur code en présence de bruit sera reformulé pour prendre en compte non seulement la vitesse de transmission moyenne mais aussi la probabilité d'erreur.

1.1.1. Règle de décodage d'un canal avec bruit

En absence de bruit de transmission la fonction de décodage était naturellement définie comme la transformation inverse de la fonction de codage. C'était par définition l'unique façon de retrouver exactement le message émis.
En présence de bruit il existe une incertitude sur le message émis lorsqu'on observe le message reçu. Autrement dit, un message reçu peut correspondre à plusieurs messages en entrée avec une distribution de probabilité qui peut être déduite à partir de la matrice de transition. Pour traiter ce problème et quantifier la probabilité d'erreur de transmission nous allons introduire la notion de règle de décodage de canal.

Définition 6.1. Règle de décodage. Soit un canal avec un alphabet d'entrée $\Omega_X = \{x_i\}_{i=1}^n$ et un alphabet de sortie $\Omega_Y = \{y_j\}_{j=1}^m$. La *règle de décodage de canal* est une fonction déterministe

$$g : \{y_1, \cdots, y_m\} \to \{x_1, \cdots, x_n\}$$

qui à chaque symbole reçu y_j associe un symbole de l'alphabet d'entrée $x_i^* = g(y_j)$.

On peut interpréter cela sous forme de règle de décision : si le symbole y_j est reçu, il est décodé comme x_i^*. Soient les variables aléatoires X, Y et $Z = g(Y)$ représentant respectivement les symboles émis, reçu et décodé. Considérons un exemple. Soit l'alphabet d'entrée de taille 3 et celui de sortie de taille 3. Supposons que l'alphabet d'entrée a la distribution de probabilité suivante

$$p(x_1) = 1/2$$
$$p(x_2) = 1/4$$
$$p(x_3) = 1/4$$

Les probabilités de transition et la règle de décodage sont illustrées par le schéma de la *figure* 6.1.

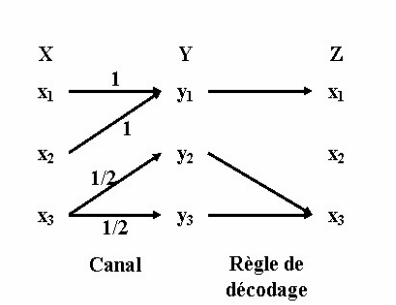


Figure 6.1. *Un exemple de règle de décision*

Considérons la règle de décision g définie par

$$g(y_1) = x_1$$
$$g(y_2) = x_3$$
$$g(y_3) = x_3$$

Le seul cas où une erreur a lieu est l'émission de symbole x_2. Ainsi la probabilité d'erreur est égale à la probabilité d'émission de x_2 donc à 1/4.

Dans le cas général soit E l'événement correspondant à un erreur lors de la transmission d'un seul symbole. Comment peut on calculer $P(E)$? Soient les variables aléatoires X, Y et $Z = g(Y)$ représentant respectivement les symboles émis, reçu et décodé.

Tout d'abord, on peut décomposer cette probabilité selon la formule de Bayes :

$$P(E) = \sum_{i=1}^{n} p(x_i) P(E|x_i)$$

Sachant que le symbole x_i est transmis, l'erreur se produit lorsque la fonction de décodage renvoie un caractère différent de x_i. Ainsi dans ce cas l'événement E équivaut à $g(Y) \neq x_i$. On a donc $P(E|x_i) = P[g(Y) \neq x_i | x_i]$. Cette dernière probabilité se décompose en somme selon les différentes valeurs de Y :

$$P(E|x_i) = P[g(Y) \neq x_i | x_i] = \sum_{j=1}^{m} P(Y = y_j \text{ et } g(y_j) \neq x_i | x_i)$$

Etant donné que la règle de décodage est déterministe on a :

$$P(Y = y_j \text{ et } g(y_j) \neq x_i | x_i) = \begin{Bmatrix} p(y_j|x_i) & \text{si } g(y_j) \neq x_i \\ 0 & \text{si } g(y_j) = x_i \end{Bmatrix} = p(y_j|x_i)\left(1 - \delta_{g(y_j),x_i}\right)$$

où $\delta_{n,k} = \begin{cases} 1 & \text{si } n = k \\ 0 & \text{si } n \neq k \end{cases}$ est le symbole de Kronecker.

Nous obtenons ainsi la probabilité d'erreur conditionnelle pour la transmission d'un seul symbole :

$$p(E|x_i) = \sum_{j=1}^{m} p(y_j|x_i)\left(1 - \delta_{g(y_j),x_i}\right)$$

1.1.2. Notion de code de canal

On peut généraliser les raisonnements de la section précédente au cas où il s'agit de transmettre des messages par blocs de longueur donnée l.

Dans la suite un canal est modélisé par le triplet X, Y, $P(Y|X)$ où X est la variable aléatoire correspondant à l'émission d'un symbole de l'alphabet d'entrée $\Omega_X = \{x_i\}_{i=1}^n$ et Y est la variable correspondant à l'observation d'un symbole reçu dans l'alphabet de sortie $\Omega_Y = \{y_j\}_{j=1}^m$. $P(Y|X)$ est la matrice de transition du canal.

On peut associer à la transmission d'un mot de longueur l une variable aléatoire $X^{(l)}$ à valeurs dans Ω_X^l. Cette variable peut être interprétée comme observation simultanée de l variables aléatoires indépendantes et toutes distribuées comme X :

$$X^{(l)} = (X_1, \cdots, X_l)$$

De même, on peut associer à la réception d'un mot de l caractères la variable aléatoire $Y^{(l)}$ à valeurs dans Ω_Y^l. Cette variable peut être interprétée comme observation simultanée de l variables aléatoires indépendantes et toutes distribuées comme Y :

$$Y^{(l)} = (Y_1, \cdots, Y_l)$$

On peut considérer alors la transmission d'un message de longueur l comme un nouveau canal $X^{(l)}$, $Y^{(l)}$, $P(Y^{(l)}|X^{(l)})$, appelé ***l*-ième extension de canal** initial. La matrice de transition de ce nouveau canal peut être déduite de

$$p[(y_1, \cdots, y_l)|(x_1, \cdots, x_l)] = \prod_{k=1}^{l} p(y_k|x_k)$$

Définition 6.2. Code de canal. Soit un canal X, Y, $P(Y|X)$. Un code (n,l) pour ce canal est un couple (W,g) où

1. $W = \{\omega_1, \cdots, \omega_n\}$ est un ensemble de mots de longueur l dans l'alphabet d'entrée Ω_X, appelés mots du code.

2. g est une règle de décodage

$$g : (\Omega_Y)^l \to W$$

qui associe à toute séquence reçue de longueur l dans l'alphabet de sortie Ω_Y un des mots du code.

Définition 6.3. Probabilité d'erreur conditionnelle. Soient un canal X, Y, $P(Y|X)$ et un code (n,l) pour ce canal (W,g). Pour chaque mot du code ω_i on définit la *probabilité d'erreur conditionnelle*

$$\lambda_i^{(l)} = P(E|\omega_i) = \sum_{(y_1,\cdots,y_l)\in(\Omega_Y)^l} p[(y_1,\cdots,y_l|\omega_l)](1-\delta_{g(y_1,\cdots,y_l),\omega_i})$$

Définition 6.4. Probabilité d'erreur moyenne. Soient un canal X, Y, $P(Y|X)$ et un code (n,l) pour ce canal (W,g). On définit la *probabilité d'erreur moyenne* (algébrique) du code par :

$$\lambda^{(l)} = \frac{1}{n}\sum_{i=1}^{n} \lambda_i^{(l)}$$

Définition 6.5. Probabilité d'erreur maximale. Soient un canal X, Y, $P(Y|X)$ et un code (n,l) pour ce canal (W,g). On définit la *probabilité d'erreur maximale* du code par :

$$\lambda_{\max}^{(l)} = \max_{i=1,\cdots,n} \lambda_i^{(l)}$$

Définition 6.6. Débit de communication d'un code. Soient un canal X, Y, $P(Y|X)$ et un code (n,l) pour ce canal (W,g). On définit le *débit de communication* du code par :

$$R = \frac{\log_2(n)}{l}$$

L'unité de mesure est le *Shannon par symbole transmis*.

1.2. Second théorème de Shannon

Nous allons maintenant pourvoir poser le problème de codage en présence de bruit :

Etant donné un canal X, Y, $P(Y|X)$ de capacité C, est-il possible de transmettre des messages avec un débit aussi proche que possible de C et avec une probabilité d'erreur aussi petite que possible ?

La réponse à ce problème est donnée par le théorème suivant :

Théorème 6.1. Second théorème de Shannon. Soit un canal X, Y, $P(Y|X)$ de capacité $C > 0$. Pour tout $R < C$ il est possible de trouver un code de canal avec un débit R et une probabilité d'erreur aussi petite que possible. Plus précisément, il existe une suite de codes $(M(l),l)$ avec $M(l) = 2^{lR}$ telle que

$$\lim_{l \to \infty} \lambda^{(l)} = 0$$

2. Codes correcteurs d'erreurs

Le second théorème de Shannon établit l'existence de codes de canal permettant de transmettre à un débit aussi proche qu'on le souhaite de la capacité du canal et une probabilité d'erreur arbitrairement petite. Mais, comme dans le cas de codage de source, le théorème ne donne aucune indication sur la façon de construire de tels codes. En pratique, il existe de nombreuses familles de codes permettant d'approcher au mieux les performances optimales, sans pour autant alourdir les temps de codage et décodage.

Dans cette section nous allons présenter une classe particulière de codes de canal qui permet de détecter et corriger assez bien des erreurs de transmission. Il s'agit de *codes binaires linéaires*. Nous introduirons d'abord le groupe G_k formé de k-uplets de bits. Ensuite, on introduira la *distance de Hamming* qui permet d'évaluer efficacement les erreurs de transmission. Enfin, après quelques définitions générales liées à la détection et à la correction d'erreurs nous allons introduire les codes linéaires et leurs propriétés.

2.1. Généralités

Dans la suite de la section nous allons étudier les codes par blocs binaires. On suppose qu'on dispose d'un message à transmettre sous forme d'une suite binaire de longueur p. Pour le codage le message sera découpé en blocs de k bits. On suppose dans la suite que tous les blocs de k bits possibles sont équiprobables. Le codage consiste à remplacer chaque bloc de k bits par un mot binaire de longueur $n \geq k$. Ainsi on définira plus tard un code comme une application qui associe à chaque bloc de longueur k un mot binaire de longueur n. On l'appellera *table de code*.
Après une transmission à travers un canal bruité le mot reçu peut être erroné et ne correspondre à aucun des mots de la table de code. Nous allons donc décrire ci-dessous une *règle de décodage au sens du minimum de distance de Hamming*. Cette règle utilise la notion de distance de Hamming que nous allons d'abord expliciter.
L'objectif principal de conception des codes se pose alors en termes de capacité de détection et de correction des erreurs de transmission. Nous allons insister sur le fait que cette capacité est une propriété intrinsèque d'un code et non le fait d'un choix particulier de règle de décodage.

2.2. Un premier exemple illustratif

Nous allons étudier ici une procédure très simple de codage et décodage permettant de détecter et même corriger des erreurs de transmission : le *code à répétition*.

Codage. Le principe est très simple : chaque bit du message à transmettre sera répété n fois. Ainsi, les blocs ici sont de taille $k = 1$ et les mots de code seront de taille n. Prenons l'exemple de $n = 3$. La table de code est alors :

s	ω
0	000
1	111

Canal de transmission. Supposons que nous utilisons pour la transmission un canal binaire symétrique dont la probabilité d'erreur est $0 < \alpha < 0.5$. La *matrice de transition* est alors

$$P(Y|X) = \begin{pmatrix} 1-\alpha & \alpha \\ \alpha & 1-\alpha \end{pmatrix}$$

Chaque bit transmis a la probabilité α d'être reçu comme son complémentaire.

Règle de décodage. A la réception, le message sera analysé par blocs de 3 bits.

Remarquons ici que dans notre code il n'y a que deux mots : 000 et 111. Cependant, compte tenu des erreurs de transmission l'ensemble de mots possibles à la réception contient tous les 8 mots de 3 bits. Si on reçoit l'un des mots du code, le décodeur le remplacera pas le symbole correspondant.

Mais que doit on faire si on reçoit 010 ?

Il est évident dans ce cas que des erreurs de transmission se sont produites. La règle de décodage doit donc permettre de décider à quel mot de code (à quel symbole) correspond cette séquence. Une règle simple à mettre en oeuvre dans cette situation est la *règle du vote majoritaire* :

- si la majorité des 3 bits reçus sont 0 on considère qu'il s'agit du mot de code 000, et on décode donc par 0.
- si la majorité des bits sont 1 on considère que le mots de code transmis est 111, et on remplace par 1.

Le nombre de bits étant impair, cette règle est non ambigüe. Voici un exemple d'application de ce schéma sur un message de 5 bits.

Message original	1	0	1	0	1
Message codé	**111**	**000**	**111**	**000**	**111**
Message reçu	101	101	110	000	011
Message décodé	1	1	1	0	1

Cet exemple montre qu'il est possible de détecter et corriger les erreurs de transmission presque toujours. En effet, si un mot subit deux erreurs de transmission ou plus, le décodage sera inexact.

Revenons à la signification de la règle du vote majoritaire. Dans notre code il n'y a que 2 mots 000 et 111. Pour un mot reçu de 3 bits, le vote majoritaire revient à lui associer celui des 2 mots de code qui a le plus de bits en commun avec ce mot. Cette règle peut donc être étendue à d'autres types de codes, en prenant en compte les ressemblances, bit par bit entre 2 mots. Ce sera fait plus loin à l'aide de la notion de distance de Hamming.

Donnons maintenant une interprétation probabiliste de la règle du vote majoritaire. Considérons un mot reçu $\mathbf{t} = t_1 t_2 t_3$ de 3 bits. Evaluons les probabilités conditionnelles $P(\mathbf{t}|s=0)$ et $P(\mathbf{t}|s=1)$ pour essayer de savoir laquelle des deux est plus importante. En supposant les symboles transmis de façon indépendante les uns des autres, on a :

$$P(\mathbf{t} = t_1 t_2 t_3 | s=0) = P(\mathbf{t} = t_1 t_2 t_3 | \omega = 000) = p(t_1|0)p(t_2|0)p(t_3|0)$$

D'après la *matrice de transition du canal* :

$$p(t_i|0) = \begin{cases} 1-\alpha & \text{si } t_i = 0 \quad \text{transmission correcte} \\ \alpha & \text{si } t_i = 1 \quad \text{transmission erronnée} \end{cases} \qquad i = 1,2,3$$

Soit l le nombre de bits 0 dans t. Alors il y a l bits corrects et $3-l$ bits 1 donc transmis avec erreur. Alors

$$P(\mathbf{t} = t_1 t_2 t_3 | s=0) = (1-\alpha)^l \alpha^{3-l}$$

Pour la même séquence $\mathbf{t} = t_1 t_2 t_3$ on a alors :

$$P(\mathbf{t} = t_1 t_2 t_3 | s=1) = (\alpha)^l (1-\alpha)^{3-l}$$

car si c'est 1 qui est transmis alors chaque 0 dans t correspond à une erreur, donc à la probabilité α. Pour comparer les deux probabilités, considérons leur rapport :

$$\frac{P(\mathbf{t} = t_1 t_2 t_3 | s=0)}{P(\mathbf{t} = t_1 t_2 t_3 | s=1)} = \left(\frac{1-\alpha}{\alpha}\right)^l \left(\frac{\alpha}{1-\alpha}\right)^{3-l} = \left(\frac{1-\alpha}{\alpha}\right)^{2l-3}$$

Comme $\alpha < 0.5$, la quantité $\frac{1-\alpha}{\alpha}$ est telle que $\frac{1-\alpha}{\alpha} > 1$. Nous remarquons alors que si les 0 sont majoritaires dans $\mathbf{t}$, c'est à dire si $2l > 3$, la probabilité de réaliser une telle séquence $\mathbf{t}$, sachant que le mot envoyé est 000 est plus importante que la probabilité de réaliser la même séquence en envoyant le mot 111. D'après la règle de vote majoritaire dans ce cas on choisit justement le mot 000. Si ce sont les 1 qui sont majoritaires alors $2l < 3$ et on choisit encore des deux hypothèses celle qui rend la séquence de bits reçue la plus probable. C'est ce qu'on appelle la *règle du maximum de vraisemblance*.

2.3. Groupe G_k

Considérons l'ensemble de séquences binaires de taille k. On le note $G_k = \{0,1\}^k$. On a $Card(G_k) = 2^k$. On pourra appeler les éléments de cet ensemble *mots* ou *vecteurs*.
Sur cet ensemble nous définissons l'opération d'*addition modulo* 2, notée $\oplus$, entre les éléments. Cela revient à effectuer l'opération *XOR* (ou exclusif) bit par bit (l'opération $\oplus$ est commutative) :

b_1	b_2	$b_1 \oplus b_2$
1	1	0
1	0	1
0	1	1
0	0	0

Muni de cette loi additive, G_k est un groupe abélien d'élément neutre $(0,\cdots,0)$. Si on le munit également d'une loi externe multiplicative selon la table de ET logique (l'opération $\otimes$ est commutative) :

b_1	b_2	$b_1 \otimes b_2$
1	1	1
1	0	0
0	1	0
0	0	0

G_k est un espace vectoriel.

2.4. Distance de Hamming et décodage du maximum de vraisemblance

2.4.1. Distance de Hamming

On définit sur G_k une distance de la façon suivante.

Définition 6.7. Distance de Hamming. Soient $v=(v_1,\cdots,v_k)$ et $\omega=(\omega_1,\cdots,\omega_k) \in G_k$ (deux vecteurs de G_k). On appelle *distance de Hamming* le nombre de bits différents entre v et ω :

$$d_h(v,\omega)=Card\{i\in[1,\cdots,n] \mid v_i \neq \omega_i\}$$

Il est assez facile de vérifier que la distance de Hamming est une *distance*, c'est à dire qu'elle vérifie les trois axiomes :

1. $\forall v,\omega\in G_k \quad d(v,\omega)\geq 0$ et $d(v,\omega)=0 \Leftrightarrow v=\omega$
2. $\forall v,\omega\in G_k \quad d(v,\omega)=d(\omega,v)$
3. $\forall v,\omega,z\in G_k \quad d(v,\omega)\leq d(v,z)+d(z,\omega)$

A ce titre, la distance de Hamming induit une certaine topologie sur l'ensemble G_k. En particulier, on peut définir la notion de *boule ouverte* de rayon r centrée en $\omega\in G_k$ de la façon suivante :

$$B_h(\omega,r)=\{v\in G_k \quad d_h(\omega,v)<r\}$$

Pour la *boule fermée* :

$$\overline{B_h}(\omega,r)=\{v\in G_k \quad d_h(\omega,v)\leq r\}$$

Une boule ouverte $B_h(\omega,r)$ définit l'ensemble de mots binaires $v\in G_k$ qui différent de ω en moins de r bits. Par exemple, si $k=4$ et $\omega=1010$ alors

$$B_h(\omega,2)=\{1010,0010,1110,1000,1011\}$$

C'est l'ensemble des mots que l'on peut obtenir à partir de ω en changeant 0 ou 1 bit à la fois, bit à bit.

Définition 6.8. Poids d'un vecteur. Soit $v=(v_1,\cdots,v_k) \in G_k$. On appelle *poids de* v le nombre de bits différents de 0 dans v :

$$w(v)=Card\{i\in[1,\cdots,n] \mid v_i=1\}$$

On peut montrer facilement que :

Proposition 6.1. Soient $v, \omega \in G_k$. Alors

$$d_h(v,\omega) = w(v \oplus \omega)$$

Exemple 6.1. Soit $k = 5$. G_5 contient 2^k mots binaires de longueur 5. Soient

$$v = 00110, \quad \omega = 10101$$

On a alors

$$d_h(\omega, v) = 3, \qquad \omega \oplus v = 10011, \qquad w(\omega \oplus v) = 3$$

Puisque la distance de Hamming comptabilise les différences entre deux mots binaires, il est naturel de l'utiliser comme critère de décodage après une transmission erronée.

Définition 6.9. Décodage selon le minimum de distance de Hamming. Soit une suite de p mots $C = \{\omega_1, \cdots, \omega_p\} \subset G_n$ de longueur n qui forment un code. Soit $\mathbf{z} \in G_n$ un mot de longueur n reçu après transmission à travers un canal bruité d'un des mots du code C. La règle de décodage selon le minimum de distance de Hamming associe à $\mathbf{z}$ le mot de code ω le plus proche de $\mathbf{z}$ selon d_h :

$$g(\mathbf{z}) = \omega_i \in C \quad / \quad d_h(\mathbf{z}, \omega_i) = \min_{\omega \in C} d_h(\mathbf{z}, \omega)$$

2.4.2. Décodage au sens du maximum de vraisemblance

Définition 6.10. Décodage au sens du maximum de vraisemblance. Soit une suite de p mots $C = \{\omega_1, \cdots, \omega_p\} \subset G_n$ de longueur n qui forment un code. Soit $\mathbf{z} \in G_n$ un mot de longueur n reçu après transmission à travers un canal bruité d'un des mots du code C. On appelle décodage au sens du maximum de vraisemblance la règle de décodage qui associe à $\mathbf{z}$ le mot $\omega \in C$ qui maximise la probabilité de réalisation de $\mathbf{z}$ sachant ω :

$$g(\mathbf{z}) = \omega_i \in C \quad / \quad P(\mathbf{z}|\omega_i) = \max_{\omega \in C} P(\mathbf{z}|\omega)$$

Nous allons montrer maintenant une proposition importante qui établit le lien entre le *décodage au sens du maximum de vraisemblance* et celui *au sens du minimum de la distance de Hamming*.

Proposition 6.2. Si le *canal de transmission* est *binaire* et *symétrique* alors le *décodage au sens du maximum de vraisemblance est équivalent au décodage au sens du minimum de distance de Hamming*.

Preuve de la Proposition 6.2.

Supposons que nous utilisons un canal binaire symétrique de probabilité d'erreur $\alpha < 0.5$. Soit $\mathbf{z} = z_1, \cdots, z_n$ le mot binaire reçu. Rappelons que la transmission de chaque symbole est indépendante des autres symboles. Soit un mot de code $\omega \in C$. Admettons l'hypothèse que l'on a reçu $\mathbf{z}$ en ayant envoyé ω. Alors chaque bit différent entre $\mathbf{z}$ et ω représente une erreur de transmission, de probabilité $\alpha < 0.5$. Tous les autres bits sont des transmissions correctes de probabilité $1 - \alpha$. Le nombre de bits différents entre $\mathbf{z}$ et ω correspond à la distance de Hamming entre les deux mots : $d_h(\mathbf{z}, \omega)$. Alors on a :

$$P(\mathbf{z}|\omega) = \alpha^{d_h(\mathbf{z},\omega)} (1-\alpha)^{n - d_h(\mathbf{z},\omega)}$$

Soient deux mots de code ω_1 et ω_2. Notons $d_1 = d_h(\mathbf{z}, \omega_1)$ et $d_2 = d_h(\mathbf{z}, \omega_2)$. Comparons les vraisemblances associées aux deux mots :

$$\frac{P(\mathbf{z}|\omega_1)}{P(\mathbf{z}|\omega_2)} = \frac{\alpha^{d_1}(1-\alpha)^{n-d_1}}{\alpha^{d_2}(1-\alpha)^{n-d_2}} = \left(\frac{1-\alpha}{\alpha}\right)^{d_2-d_1}$$

Comme $\alpha < 0.5$, on a $\frac{1-\alpha}{\alpha} > 1$. Alors d'après la dernière relation :

$$P(\mathbf{z}|\omega_1) \leq P(\mathbf{z}|\omega_2) \Leftrightarrow d_1 \geq d_2$$

Ainsi, la *vraisemblance* $P(\mathbf{z}|\omega)$ est *maximale* quand la *distance de Hamming* $d_h(\mathbf{z}, \omega)$ est *minimale*.

Dans la suite nous allons travailler avec la **règle de décodage du minimum de distance de Hamming**.

2.5. Codes correcteurs

Définition 6.11. Code (*k*, *n*). On appelle *code de paramètres* (k, n) toute application injective $\phi : G_k \rightarrow G_n$.

- Le paramètre k s'appelle *dimension du code*.
- Le paramètre n s'appelle *longueur du code*.
- L'ensemble image de ϕ

$$C = \{\phi(m) \quad m \in G_k\}$$

s'appelle *image du code*. Ses éléments s'appellent *mots de code*. Les éléments de G_k s'appellent *mots de source*.

- Le paramètre k représente donc le nombre de bits des mots *avant codage* (*mots-sources*).
- Le paramètre n représente le nombre de bits des mots *après codage* (*mots-codes* issus du codage).

Un code est une association entre les blocs de longueur k et des mots binaires de longueur n. Il faut que l'*application* qui le représente soit *injective* pour garantir que *deux blocs différents* soient codés par *deux mots différents*.

Exemple 6.2. Code de répétition. Le code de répétition vu avant est un code de paramètres (k = 1, n = 3).
Les mots de source sont $\{m_1 = 0, m_2 = 1\}$. L'image de ce code est

$$C = \{000, 111\}$$

Définition 6.12. Distance minimale d'un code. Soit un code de paramètres (k, n) dont l'image est l'ensemble de mots C. On appelle *distance minimale d'un code* la valeur :

$$d = \min_{\omega, v \in C \ \omega \neq v} d_h(\omega, v)$$

C'est la *plus petite distance de Hamming* entre deux mots différents du code. Ce paramètre joue un rôle très important dans la définition des capacités d'un code à détecter et corriger les erreurs de transmission.

2.5.1. Détection et correction d'erreurs

On suppose que la règle de décodage utilisée est celle du minimum de distance de Hamming. Soit un code (k, n) d'image C. Supposons qu'un mot du code $\omega \in C$ est transmis. On reçoit le mot $\mathbf{z} \in G_n$. Si $\mathbf{z}$ coïncide avec un des mots du code, $v \in C$, il sera décodé comme v. Même si ce n'est pas le mot du code qui a été transmis. Cela peut se produire si les erreurs se produisent exactement sur les bits qui sont différents entre le mot transmis, ω, et le mot du code reçu, v.

Donc s'il y a $d_h(\omega, v)$ erreurs et si elles se produisent "*aux bons endroits*", elles sont indétectables. On détecte les erreurs si le mot reçu, $\mathbf{z}$, ne correspond à aucun des mots de code.

Ainsi nous pouvons dire que le nombre maximal d'erreurs qu'il est possible de détecter quels que soient les bits où elles se produisent est égal à $d-1$, où d est la distance minimale du code.

En effet, pour obtenir à partir d'un mot du code $\omega \in C$, un autre mot du code, $v \in C$, il faut changer au moins d bits.

Donc si on change moins de d bits on obtient forcément un mot qui n'est pas dans le code. On détecte alors la présence des erreurs à la réception. Nous venons de justifier la définition suivante :

Définition 6.13. Capacité de détection. On appelle *capacité de détection* d'erreurs d'un code, le *nombre maximal* e_d d'*erreurs* qu'il est **toujours** possible de détecter. On a :

$$e_d = d - 1$$

où d est la distance minimale du code.

Lorsque les erreurs sont détectées on ne sait pas quels sont les bits erronés et combien ils sont. On sait juste que le mot reçu n'est égal à aucun des mots du code. La règle de décodage que l'on a choisie va alors remplacer le mot reçu par celui des mots du code qui est le plus proche selon la distance de Hamming. Si cette action conduit au mot qui a réellement été transmis, elle équivaut à la correction des erreurs. Nous allons donc définir un autre paramètre important pour juger de la qualité d'un code : la *capacité de correction d'erreurs.*

Définition 6.14. Capacité de correction. On appelle *capacité de correction d'erreurs* d'un code, le *nombre maximal* e_c d'*erreurs* qu'il est **toujours** possible de corriger. On a :

$$e_c = E\left[\frac{d-1}{2}\right]$$

où d est la distance minimale du code, $E[\,]$ est la partie entière d'un réel.

Preuve

La formule donnée dans la définition de la capacité de correction nécessite une justification. Remarquons que si $C = \{\omega_1, \cdots, \omega_p\}$ est l'image d'un code (k, n) de distance minimale d, alors toutes les boules fermées $\overline{B_h}(\omega_j, r)$ de rayon $r = E\left[\frac{d-1}{2}\right]$ sont deux à deux disjointes. Soit $\omega \in C$ le mot de code transmis. Tant que le mot reçu reste dans la boule $\overline{B_h}(\omega, r)$, il sera décodé comme ω, et donc les erreurs seront corrigées. Ainsi le plus grand nombre d'erreurs corrigibles et $E\left[\frac{d-1}{2}\right]$.

2.6. Codes linéaires correcteurs d'erreurs

Nous allons enfin nous intéresser ici à une famille particulière de codes (k, n), les codes linéaires. Leur principal intérêt réside dans la facilité de mise en oeuvre et la rapidité des algorithmes de décodage.

Définition 6.15. Code correcteur linéaire (k, n). Un code linéaire (k, n) est un code de classe (k, n) tel que l'application $\phi : G_k \to G_n$ qui le génère est une application linéaire, c'est-à-dire, tel qu'il existe une matrice $G \in \mathrm{M}^{n,k}(0,1)$ telle que :

$$\phi(m) = Gm \qquad \forall m = \begin{pmatrix} m_1 \\ \vdots \\ m_p \end{pmatrix}$$

On appelle G la **matrice génératrice** du code. Elle est de dimension n x k .

Remarque 6.1. Ici les coefficients de la matrice G prennent ses valeurs dans {0, 1}. La multiplication matricielle se fait selon les opérations $\oplus$, $\otimes$ sur {0, 1}, c'est à dire dans l'arithmétique modulo 2. Par exemple, avec $k = 2$ et $n = 3$, le mot-code associé au clair $(1\ 1)^T$ est :

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} (1 \otimes 1) \oplus (1 \otimes 1) \\ (0 \otimes 1) \oplus (1 \otimes 1) \\ (1 \otimes 1) \oplus (0 \otimes 1) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

Remarque 6.2. Si le code est linéaire alors son image, C, l'ensemble de tous les mots de code, est un sous-espace vectoriel de G_n. Il contient donc obligatoirement le mot nul 000 . . . 000 de n symboles.

Définition 6.16. Forme systématique. On dit qu'un code linéaire (k, n) est *systématique* si et seulement si sa matrice génératrice $\tilde{G}$, de dimension n x k, est de la forme (en blocs) :

$$\tilde{G} = \begin{pmatrix} I_k \\ \hline G' \end{pmatrix}$$

où I_k est la matrice identité de dimension k et G' est une matrice de n - k lignes et k colonnes.

Proposition 6.3. Les *codes systématiques* ont la propriété intéressante : pour tout mot de source $m \in G_k$, le mot de code correspondant $\omega = Gm$ a m comme préfixe. C'est-à-dire :

$$\exists \omega' \quad / \quad \omega = m\omega'$$

Exemple 6.3. Un codage de parité paire consiste à ajouter (concaténer) à chaque mot de source un bit de contrôle qui représente la parité du nombre de 1 dans le bloc constitué de l'ensemble mot-source et bit de contrôle. Si le nombre de bits 1 est pair dans cet ensemble, le bit de contrôle ajouté est 0, sinon on ajoute 1. Par exemple, pour les blocs de 3 bits, le code de parité est défini par le tableau suivant :

m	000	100	010	001	110	011	101	111
$\phi(m)$	000**0**	100**1**	010**1**	001**1**	110**0**	011**0**	101**0**	111**1**

C'est un code de paramètres (k, n) = (3, 4). Chaque mot de source m est le préfixe du mot de code correspondant. C'est un code systématique. Il est linéaire car on peut remarquer que le bit de parité est toujours égal à la somme (modulo 2 !) des bits du mot m. Alors la matrice génératrice de ce code est très simple :

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

Définition 6.17. Codes équivalents. Deux codes sont dits *équivalents* s'ils ont la *même image*.

Pour déterminer si un code linéaire de matrice génératrice G donnée possède un équivalent systématique $\tilde{\tilde{G}}$, il suffit d'effectuer une opération d'é*chelonnage de la matrice* en faisant des opérations élémentaires de *combinaisons linéaires sur les colonnes de* G. Les matrices G et $\tilde{\tilde{G}}$ produisent des *codes équivalents* (identiques).

G et $\tilde{\tilde{G}}$ produisent la même image, mais les mots-codes générés ne correspondent pas aux mêmes mots-sources.

Exemple 6.4. Soit le code de *matrice génératrice* G de dimension (n = 5)x(k = 3) :

$$G = \begin{matrix} c_1 & c_2 & c_3 \end{matrix} \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

L'obtention de la forme systématique $\tilde{\tilde{G}}$ implique un échelonnage de G suivant ses colonnes. Notons les c_1, c_2, c_3.

Les premières opérations visent à annuler les éléments $G_{1,2}$ et $G_{1,3}$. Le pivot est $G_{1,1}$.

On réalise d'abord les combinaisons linéaires $\begin{cases} c_1 \leftarrow c_1 \\ c_2 \leftarrow c_1 + c_2 \\ c_3 \leftarrow c_3 \end{cases}$ on obtient : $G'' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$

Dans un 2nd temps, on annulle $G''_{2,3}$ $\begin{cases} c_1 \leftarrow c_1 \\ c_2 \leftarrow c_2 \\ c_3 \leftarrow c_2 + c_3 \end{cases}$ on obtient : $G''' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$

Enfin, on annulle $G''_{3,1}$ et $G''_{3,2}$ $\begin{cases} c_1 \leftarrow c_1 + c_3 \\ c_2 \leftarrow c_2 + c_3 \\ c_3 \leftarrow c_3 \end{cases}$ $\Rightarrow$ $\tilde{\tilde{G}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$

On a un équivalent systématique du code car les 3 1ères lignes de la matrice G forment une matrice identité de rang 3.

$$\tilde{\tilde{G}} = \begin{pmatrix} I_3 \\ \hline G' \end{pmatrix} \quad \text{avec} \quad I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{et} \quad G' = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

La *table* C *des mots-codes* peut ainsi s'obtenir : (G et $\tilde{G}$ produisent la même image de mots-codes, mais pour des mots-sources différents.)

#	*mot source* $m_i^{\mathbf{T}}$	*mot-code* $\omega_i^{\mathbf{T}} = (Gm_i)^{\mathbf{T}} = (\tilde{G}m_i)^{\mathbf{T}} = \left[\left(\frac{I_k}{G'}\right)m_i\right]^{\mathbf{T}} = \left(\frac{m_i}{G'm_i}\right)^{\mathbf{T}} = \left[m_i^{\mathbf{T}} \middle| (G'm_i)^{\mathbf{T}}\right]$	*distance, poids* $d_h(\omega_i,\mathbf{0}) = w(\omega_i)$ (= *nombre de bits* 1 *de* ω_i)
1	000	000 00	0
2	001	001 01	2
3	010	010 11	3
4	011	011 10	3
5	100	100 00	1
6	101	101 01	3
7	110	110 11	4
8	111	111 10	4

$\rightarrow d = 1$

La proposition suivante donne une méthode très simple pour évaluer la *distance minimale* (et donc les capacités de détection et de correction d'erreurs) d'un code linéaire.

Proposition 6.4. Distance minimale d'un code linéaire. La *distance minimale d'un code linéaire* est égale au *plus petit poids non nul des mots de code*.

Preuve de la Proposition 6.4.

Rappelons que la distance minimale d d'un code est :

$$d = \min_{\omega,v\in C \;\; \omega\neq v} d_h(\omega,v)$$

Remarquons également que :

$$d_h(\omega,v) = w(\omega \oplus v)$$

Or, si le code est linéaire, son image C est un sous-espace vectoriel. Donc pour tout couple de mots $\omega, v \in C$, on a : $\omega \oplus v \in C$. Si $\omega \neq v$ alors $\omega \oplus v \neq 0$. On a donc :

$$d = \min_{\omega,v\in C \;\; \omega\neq v} d_h(\omega,v) = \min_{\omega,v\in C \;\; \omega\neq v} w(\omega \oplus v) = \min_{z\in C \;\; z\neq 0} w(z)$$

Proposition 6.5. Borne de Singleton. La *distance minimale d'un code linéaire* de dimension k et de longueur n vérifie

$$d \leq n + 1 - k$$

2.6.1. Matrice de contrôle et décodage d'un code linéaire

Définition 6.18. Matrice de contrôle d'un code linéaire. Soit un code linéaire (k, n) de matrice génératrice G et d'image C. On appelle *matrice de contrôle* de ce code toute matrice $H \in M^{\,n-k,n}$ telle que pour tout mot de code $\omega = \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_p \end{pmatrix}$ on a :

$$H\omega = \mathbf{0}$$

Autrement dit, H est une matrice dont le noyau coïncide avec l'ensemble de mots de code C.

Généralement il n'est pas facile de trouver la *matrice de contrôle* pour un code donné. Du point de vue algébrique, cela revient à trouver la base du sous-espace vectoriel orthogonal à C. Mais dans le cas particulier de *codes systématiques*, la proposition suivante permet de construire facilement la *matrice de contrôle*.

Proposition 6.6. Matrice de contrôle d'un code systématique. Soit un code linéaire (k, n) *systématique* de *matrice génératrice* $\tilde{G} = \begin{pmatrix} I_k \\ \hline G' \end{pmatrix}$. Alors sa *matrice de contrôle* est :

$$H = \left(G' \mid I_{n-k} \right)$$

Définition 6.19. Syndrome et erreur. Soit un code linéaire (k, n) de matrice génératrice G et de matrice de contrôle H. Soit $W \in C$ un mot de code, et $Z \in \{0,1\}^n$, un mot reçu après la transmission de W à travers un canal bruité.

On appelle **mot erreur**, le mot $E \in \{0,1\}^n$ tel que :

$$Z = W \oplus E \quad \Rightarrow \quad E = Z \oplus W$$

On appelle **mot syndrome** de Z, le mot :

$$S = HZ$$

Chaque bit du mot erreur E est nul si les bits correspondants de Z et W coïncident, et est égal à 1 s'ils sont différents. Ainsi

$$w(E) = d_h(W, Z)$$

Si on a trouvé le mot erreur E on peut retrouver W par la relation :

$$W = Z \oplus E$$

Donc, pour corriger les erreurs de transmission, il suffit de trouver le vecteur E.

Si le *syndrome* de Z, soit $S = HZ$, est *nul* ($S = \mathbf{0}$), c'est que Z est un *mot de code* (par définition de la matrice de contrôle). Il sera alors décodé par le bloc (*mot source*) correspondant dans la table du code. Ainsi, **un syndrome nul correspond à l'absence ou à la non détection des erreurs de transmission**.
Un syndrome non nul équivaut à la détection d'erreurs, car cela signifie que le mot reçu Z n'est pas un mot de code.

Proposition 6.7.
Soit un code linéaire (k, n) de matrice génératrice G et de matrice de contrôle H. Soit $W \in C$ un mot-code transmis, et $Z \in \{0,1\}^n$, un mot reçu après la transmission de W à travers un canal bruité. Soient $E = Z \oplus W$ le mot erreur, et $S = HZ$ le syndrome de Z. Alors le *mot erreur* E *a le même syndrome que* Z : $HE = HZ$

Preuve de la Proposition 6.7.

Par définition de la matrice de contrôle et du fait que W est un mot du code, on a :

$$HW = \mathbf{0}$$

Alors

$$HE = H(W \oplus Z) = HW \oplus HZ = HZ = S$$

Définition 6.20. Classe latérale. Soit un code linéaire (k, n) de matrice de contrôle H. Soit Z un mot de $\{0,1\}^n$. On appelle *classe latérale de Z, l'ensemble des mots ayant le même syndrome S que Z.*

$$\mathbf{L}(Z) = \{W \in \{0,1\}^n \quad HW = HZ = S\}$$

Ainsi nous savons où chercher le vecteur erreur E pour la correction du mot reçu, Z. Il se trouve dans la classe latérale de Z. Il reste à savoir comment le choisir. Rappelons que le décodage se fera selon le principe du minimum de distance de Hamming. Le mot reçu Z sera remplacé par le mot de code W le plus proche de Z au sens du minimum de distance de Hamming. Nous avons déjà mentionné que le poids du vecteur erreur $E = Z \oplus W$ entre le mot reçu Z et un mot de code W est égal à la distance de Hamming entre Z et W. Or, d'après la proposition **6.7.**, on a :

$$\forall W \in C \quad E = Z \oplus W \in \mathbf{L}(Z)$$

Et réciproquement,

$$\forall E \in \mathbf{L}(Z) \quad W = Z \oplus E \in C$$

Donc trouver le mot de code le plus proche du mot reçu Z équivaut à trouver l'élément $E \in \mathbf{L}(Z)$ de plus petit poids.

Proposition 6.8. Soit un code linéaire de matrice de contrôle H. Le décodage au sens du minimum de distance de Hamming d'un mot Z revient à trouver dans la classe latérale de Z, le mot de plus petit poids, E, et à remplacer Z par le mot de code $W = Z \oplus E \in C$.

En pratique, on explore $\mathbf{L}(Z)$ dans l'ordre croissant des poids de mots. Pour faciliter les explications nous allons introduire une famille de mots particuliers :

$$E_i = (0 \cdots 010 \cdots 0)$$

où pour chaque mot $E_i \quad i = 1, \cdots, n$, il n'y a qu'un seul bit égal à 1, à la i-éme position, faisant ainsi référence à 1 bit erroné.

Remarquons qu'un mot E de poids w est égal à la somme de w mots parmi les $E_i \quad i = 1, \cdots, n$

$$w(E) = w \quad \Leftrightarrow \quad \exists i_1, \cdots, i_w \quad / \quad E = \sum_{j=1}^{w} E_{i_j}$$

Alors le résultat de produit HE est une combinaison linéaire des colonnes de la matrice H, notées $c_1, \cdots, c_n$

$$HE = \sum_{j=1}^{w} c_{i_j}$$

car les autres bits de E sont nuls.

Revenons maintenant au problème de recherche de l'erreur de poids minimum dans la classe latérale d'un mot reçu Z. Soit $S = HZ$ le syndrome de Z. Soit $0 \le w \le e_c$ (e_c est la capacité de correction du code). Dire qu'il existe dans $\mathbf{L}(Z)$ un mot E de poids w équivaut à :

$$\exists i_1, \cdots, i_w \quad / \quad E = \sum_{j=1}^{w} E_{i_j} \quad \text{et} \quad HE = \sum_{j=1}^{w} c_{i_j} = S$$

Autrement dit, il existe un choix de w colonnes de la matrice de contrôle H dont la somme est égale à S. Alors on commence par tester les poids w allant de 0 à e_c jusqu'à ce qu'on trouve une somme de w colonnes de H égale à S. Le vecteur erreur est alors égal à la somme correspondante de E_{i_j}.

Par exemple, pour $w = 0$. Si on trouve que $HZ = 0$ alors Z est un mot de code. Sinon, on passe à $w = 1$. Pour $w = 1$ si on trouve une colonne c_i de H telle que $c_i = S$, alors le mot de code corrigé sera $W = Z \oplus E_i$. Sinon on passe à $w = 2$. S'il existe 2 colonnes parmi n de la matrice H telles que $c_i + c_j = S$ alors le mot corrigé sera $W = Z \oplus E_i \oplus E_j$. Sinon, on continue avec $w = 3$ et ainsi de suite jusqu'à $w = e_c$. On a donc l'algorithme de décodage suivant :

Algorithme 6.1. Décodage d'un code linéaire.

Etape 1. $w = 0$. On vérifie si $S = HZ = \mathbf{0}$. Si oui, Z est un mot de code. Sinon, on pose $w = 1$ et on passe à l'**étape 2.**
Etape 2. *Tant que* $w \le e_c$ faire

Tester toutes les sommes de w colonnes de H jusqu'à ce que

$$\sum_{j=1}^{w} c_{i_j} = S$$

Si égalité trouvée, renvoyer $W = Z \oplus \sum_{j=1}^{w} E_{i_j}$ et sortir de la boucle d'itérations. *Sinon*, faire $w = w + 1$.

Fin de *Tant que*.
Etape 3. *Si* $w = e_c + 1$ et aucune solution n'est trouvée, *renvoyer* "erreur de décodage".

———————

TD 6. Codage de canal.
Codes linéaires correcteurs d'erreurs

Exercice 1. Code à répétition. Nous allons expérimenter l'utilisation d'un code à répétition d'ordre 3 pour la transmission d'images binaires en Noir & Blanc. On suppose qu'une image binaire carrée est représentée par une matrice $N \times N$ pixels (image *raster* ou encore *bitmap*) avec des valeurs de pixels à 0 (Noir) ou 1 (Blanc).
Le code à répétition que nous allons étudier consiste à répéter chaque bit 3 fois avant de le transmettre. Par exemple, le message 10100 sera transmis sous la forme 111 000 111 000 000.
On suppose qu'on utilise pour la transmission un canal binaire symétrique de probabilité d'erreur α. Chaque bit a donc la probabilité $1-\alpha$ d'être transmis correctement et la probabilité α d'être inversé (complémenté).
Le décodage se fait par vote majoritaire. Chaque bloc de 3 bits reçu est décodé par le bit qui y est majoritaire. La règle de décodage est donc donnée par le tableau suivant :

Bloc reçu	000	001	010	100	101	011	110	111
Caractère décodé	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**

Pour les images, il sera plus pratique de réaliser ce codage de la façon suivante. On transmet la même image 3 fois de suite à travers le canal. Ensuite on réalise le vote majoritaire, bit par bit, entre les 3 images reçues, pour obtenir l'image d'origine.

I. Expérience.

Dans cette partie, nous allons d'abord réaliser quelques expériences à l'aide d'un programme *Scilab* fourni, **image.sci**. Le programme applique la méthode à une petite image de test, de taille 13×17. L'image est définie par une matrice A dont les valeurs sont binaires. La fonction principale, *CTD*(A, *tauxErr*), réalise les actions suivantes :

- Affichage de l'image d'origine, A.
- Génération aléatoire de 3 matrices binaires représentant les bits erronés lors de la transmission;
tauxErr est le paramètre qui définit la probabilité d'erreur.
- Calcul de 3 matrices transmises en inversant les bits de la matrice originale indiqués par les matrices d'erreur;
cette opération correspond à la transmission 3 fois de la même matrice à travers le canal.
- Affichage des 3 images reçues.
- Calcul par vote majoritaire bit par bit de la matrice définitive reçue; cette opération correspond au décodage.
- Affichage de la matrice résultant du décodage.

La fonction *CTD*(A, *tauxErr*) renvoie la *matrice de cumul d'erreurs*. Chaque élément de cette matrice représente le *nombre d'erreurs* (0, 1, 2, ou 3) pour chaque pixel sur les 3 transmissions.

Manipulation (facultative)

1. Exécuter le fichier *image.sci* dans *Scilab*. Le taux d'erreur *tauxErr* est fixé dans le fichier à 0.1. Est ce que le *décodage* est parfait ?
2. Essayer la procédure avec un *taux d'erreur* plus petit (0.05) et plus grand (0.3). Qu'observe-t-on ?

Interpétation

3. Quel est le lien entre la *matrice de cumul d'erreurs* et la possibilité de *correction des erreurs* ?

II. Analyse.

1. Ce *code* contient deux mots $\omega_1 = 000$ et $\omega_2 = 111$. Calculer la *distance minimale* de ce code et les *capacités de détection et de correction*.
2. Ce code est-il *linéaire* ? Si oui, quelle est sa *matrice génératrice* G ?

Exercice 2. Code linéaire. Soit un code défini par sa *matrice génératrice* de dimension 8x4 :

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

1. Donner les paramètres (k, n) du code.

2. Ecrire la matrice G sous forme *systématique*.

3. Trouver l'ensemble de *mots de code* (*image* C du code).

4. Déterminer la *distance minimale* du code. En déduire la *capacité de correction*.

5. A partir de la *matrice génératrice*, déterminer la *matrice de contrôle* du code.

6. On reçoit les mots suivants :

- $m_1 = (0001\,0101)^{\mathbf{T}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$
- $m_2 = (0100\,0111)^{\mathbf{T}} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$
- $m_3 = (0011\,1010)^{\mathbf{T}} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$

Sans utiliser la table de mots de code, comment savoir si ces mots sont des *mots-codes* ?

7. En supposant qu'il y a eu *au plus* 1 *erreur de transmission par mot*, est-il possible de *retrouver* les mots de code transmis ?

———————