Codage et compression de source

N. Lebedev

Table des matières

1 Bases de la théorie de l'information

1.1 Introduction

La théorie de l'information cherche à répondre à deux principales questions :

1. Quelle est la limite ultime de la compression des données issues d'une source ?
   Réponse : Entropie $H(X)$, d'une source X, qui caractérise le contenu informatif intrinsèque **minimal** nécessaire pour décrire la source. $H(X) \leq \overline{R}$ [bits/symbole], ce qui veut dire qu'un nombre moyen de bits par symbole de la source qui sera nécessaire pour encoder ou représenter cette source **sans pertes**, devra être supérieur à l'entropie de cette source.
2. Quel est le débit maximal de transmission des données à travers un canal de communication ?
   Réponse : Capacité du canal C, ainsi le débit réalisable devra être toujours inférieur à la capacité d'un canal donné $R \leq C$ [bits/s], si l'on veut faire tendre la probabilité d'erreur vers une valeur faible.

Toutes les techniques de transmission et de communication peuvent être vues dans ce champ sur la Fig. 1.

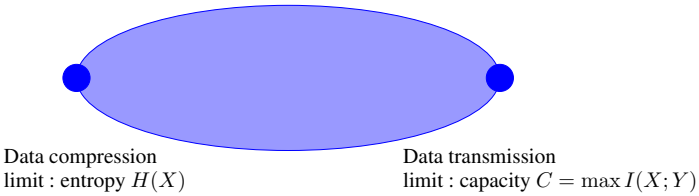


FIGURE 1 – *Limites fondamentales de la théorie de l'information et de communication.*

Afin de pouvoir concevoir et mettre un œuvre un système de communication, il est ainsi nécessaire de :

1. définir les modèles des sources et des canaux ;
2. trouver leurs limites théoriques (entropie, capacité) soit sous forme analytique d'une expression mathématique, soit de manière empirique ;
3. concevoir les algorithmes de compression, codage, modulation,... permettant d'approcher ces limites dans un système réel.

De manière générale, la sortie d'une source quelconque peut être considérée aléatoire, inconnue (text d'un livre, événement climatique, échantillons de la parole numérisée,...) pour l'observateur ou le destinataire de ces données. Dans le cas contraire, si la sortie de la source était connue, prédite, le problème même de la communication deviendrait caduque—peu d'utilité de transmettre l'information déjà connue au destinataire.

1.2 Caractérisation des sources

Il convient de distinguer deux types de sources :

1.2.1 Source discrète

qui produit les symboles $X = x_1, \ldots, x_L$ appartenant à un alphabet ou à un ensemble fini de valeurs. Ex : source binaire $L = 2$, quaternaire $L = 4$, octale $L = 8$,...

Les deux modèles mathématiques utilisés couramment sont :

1. DMS (Discrete Memoryless Source), est une source discrète sans mémoire, qui produit une suite de symboles $X(t_i) = X_i$ aux instants de temps t_i, qui sont statistiquement indépendants, $P(X_i, X_j) = P(X_i)P(X_j)$, pour tous instants t_i, t_j. Ce sont les échantillons d'un processus aléatoire quelconque prélevés aux instants de temps t_i.
2. Sources dont les blocs de symboles successifs sont dépendants statistiquement (échantillons de la parole d'un langage naturel, lettres dans les mots d'un text écrit, bits des données encodés par un code ou un chiffrement,...). De telles sources sont souvent décrites par une propriété de stationnarité, à savoir, le fait de préserver constante, invariable, la distribution de probabilité conjointe de (n) symboles successifs observés à un certain instant (t_i) quelconque, c'est à dire, peu importe le décalage, retard ou avance, dans le temps τ. Cela se formalise comme ceci :

$$p(X(t_i), \ldots, X(t_{i+(n-1)})) = p(X(t_i + \tau), \ldots, X(t_{i+(n-1)} + \tau)).$$

1.2.2 Source analogique ou continue

fournit un signal qui très souvent décrit une mesure d'une grandeur physique faite par un capteur (température, pression, flux optique,...) représenté par une forme d'onde $x(t)$. Ce signal, est typiquement modélisé comme réalisation d'un processus aléatoire $X(t)$. Souvent, l'hypothèse de stationnarité est vérifiée sur l'échelle de temps nécessaire pour le traitement de ce signal. Dans ce cas, ce processus est souvent décrit par sa fonction d'autocorrélation $R_x(\tau)$ et sa DSP (Densité Spectrale de Puissance) $G_x(f)$, car ce signal est vu comme signal en puissance finie (et énergie infini $E_x = \infty$).

Si le processus aléatoire $X(t)$ a le contenu spectral à bande limitée [1] (ex : signal de la parole filtré, signal d'une porteuse Wifi, ou DSL, ou 4G, démodulé en bande de base), à savoir, sa DSP $G_x(f) = 0$ for $|f| \geq f_m$, le théorème d'échantillonnage (Shannon-Nyquist) permettra de reconstruire le signal original à partir d'un nombre d'échantillons qui tend vers l'infini.

$$X(t) = \sum_i X\left(\frac{i}{2f_m}\right) \times \text{sinc}\left(2f_m(t - \frac{i}{2f_m})\right),$$

où $X(t_i = \frac{i}{2f_m})$ sont les échantillons du signal qui ont été prélevés au moins à la fréquence de Shannon-Nyquist, c'est à dire $f_s \geq 2f_m$, voir Fig. 2 (démo).

La source continue numérisée avec le respect du théorème d'échantillonnage de Shannon, peut ainsi être décrite de manière équivalente par une source discrète, par la densité de probabilité conjointe $p(X_i, ..., X_{i+n-1})$ de (n) symboles successifs inter-dépendants. Notons, que ce signal échantillonné tout en étant à temps discret, reste un signal analogique—les amplitudes peuvent être les valeurs quelconques réelles limitées à l'intervalle de la dynamique du capteur $X(\frac{i}{2f_m}) \in \mathbb{R}_{[-V_{min}, V_{max}]}$.

En général, l'étape suivante de traitement est justement, la **quantification** qui consiste à représenter, à approximer, la valeur d'amplitude réelle $X(\frac{i}{2f_m})$ de chaque échantillon, par la valeur d'un niveau d'amplitude discrète le plus proche sur une grille de quantification de taille finie de L niveaux.

Ainsi, par les procédés d'échantillonnage et de quantification, la source analogique est transformée en source discrète équivalente. Rappelons, que cette procédure dite de *formatage de source*

1. Si $X(t)$ n'est pas à bande limitée, la transmission et la récupération parfaite sans perte de ce signal ne sera pas possible, car aussi grande que puisse être la fréquence d'échantillonnage dans un système électronique réalisable, il y aura toujours un recouvrement spectral *(aliasing)* dû à l'échantillonage en temps du signal, et l'application d'un filtrage nécessaire à la limitation de la bande d'un système réaliste fera perdre une partie des fréquences utiles, tout en subissant de l'interférence due aux autres, à cause du recouvrement spectral.

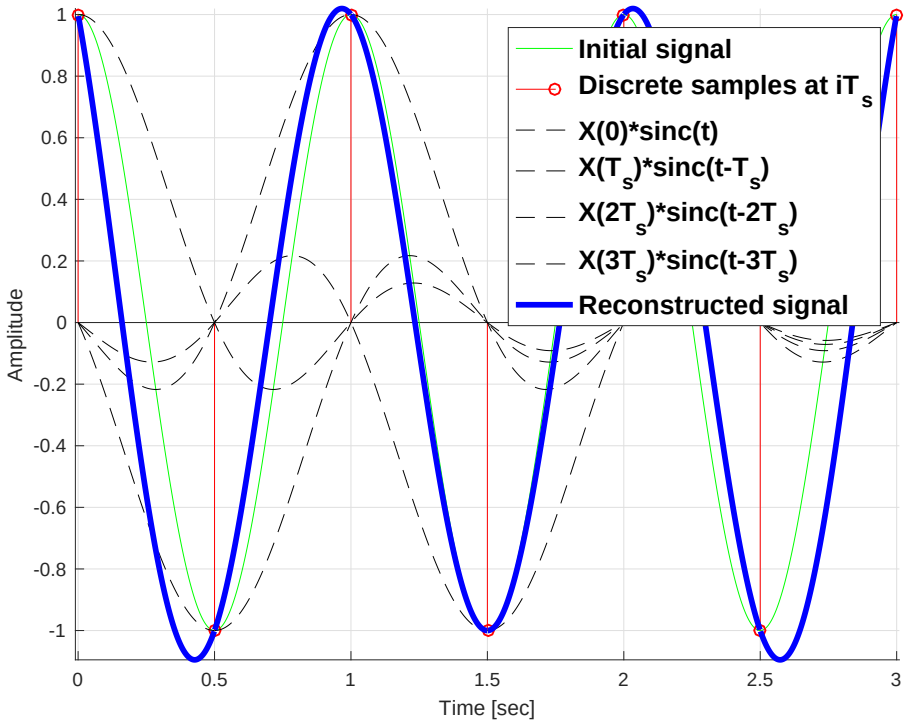


FIGURE 2 – *Reconstruction d'un signal à partir de ses échantillons à la fréquence de Nyquist* $f_s = 2f_m$. *Axe X est normalisé en* $T_s = 1/f_s$

est à pertes, à cause de le génération de l'erreur (ou du bruit) de quantification de variance $\sigma_q^2 = q^2/12$. Le pas de quantification $q = (V_{max} - (-V_{min}))/L$ et ainsi le nombre de niveaux L fini, rendent ce bruit inévitable.

Cette source discrète équivalente produit les symboles d'un alphabet fini de taille L. Si ces symboles sont indépendants entre eux, chaque symbole est caractérisé par sa probabilité de réalisation :

$$p_i = P(X = x_i),\ 1 \leq i \leq L; \quad \sum_{i=1}^{L} p_i = 1. \tag{1}$$

Afin de transmettre par un système de communication numérique un flux de symboles produit par une telle source, il faut **encoder les symboles en format binaire**.

Pour cela, il existent de nombreuses techniques, les plus efficaces (?) sont les techniques de compression de source, dont les objectifs sont :

— caractériser la source par son contenu d'information intrinsèque minimal.

— extraire cette information, ou, en d'autres termes, réduire la redondance propre aux messages de la source, et de ne transmettre que le stricte minimum d'information la décrivant.

Cette approche s'avérera être plus efficace et économe, en comparaison, par exemple, aux méthodes de formatage seul, de type modulation MIC (Modulation par Impulsion Codées), qui effectuent la conversion analogique-numérique et le mapping (étiquetage) des amplitudes discrètes des échantillons sur les mots-codes *codewords* binaires, mais pas la compression.

Il se pose naturellement la question de la mesure de la quantité d'information de la source.

1.3 Quantité d'information

Une observation suivante peut être faite : **l'information est une mesure de l'incertitude**, de manque de connaissance que l'on peut avoir au sujet de la source, décrite par ses possibles réalisations (symboles). Voici une interprétation d'une telle affirmation.

— Un évènement très probable, fréquent, fournit peu de connaissance et donc d'information. Ex : un évènement certain ($Prob = 1$) contient aucune, zéro information.
— Un évènement peu probable, rare, fournit beaucoup d'information.

Plus d'incertitude ⇔ plus d'information

1.3.1 Réalizations individuelles d'une seule VA (variable aléatoire)

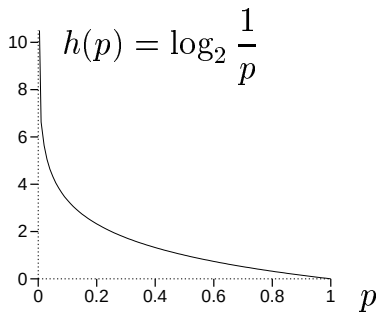
Considérons une variable aléatoire discrète X décrite par sa function de masse de probabilité, $p(x)$, et un nombre fini de réalisations—symboles de la source discrète sans mémoire (DMS) : $Prob(X = x_i) = p_i$, $i = 1, \ldots, L$, où L est la taille de l'alphabet de la source.

Une question se pose : comment évaluer la quantité d'information $I(\cdot)$ contenue dans (ou fournie par) chacune de ces réalisations individuelles $X = x_i$? En concordance avec l'affirmation du début de la section, cette information devrait être une fonction qui a pour argument la probabilité, et qui décroit avec la probabilité qui augmente. La définition devenue aujourd'hui classique suit :

Déf 1.1. $I(x_i)$ *est l'***auto-information** *d'un événement* $X = x_i$ *:*

$$I(x_i) = \log_2 \frac{1}{p_i} = -\log_2 p_i, \ \textit{[bits]} \tag{2}$$

Cette définition est appropriée, car pour $p(x_i) \to 0$, l'auto-information $I(x_i) \to \infty$, et pour $p(x_i) \to 1$, $I(x_i) \to 0$. Notons que $I(x_i) \geq 0$. Voir Fig. 3-1.



p	$h(p)$	$H_2(p)$
0.001	10.0	0.011
0.01	6.6	0.081
0.1	3.3	0.47
0.2	2.3	0.72
0.5	1.0	1.0

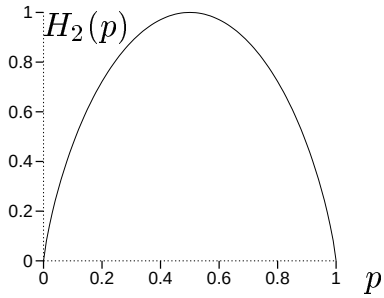


FIGURE 3 – *Auto-information* $I(p)$ *et entropie* $H_2(p)$ *D'une variable aléatoire binaire [MacKay].*

L'unité de l'information dépend de la base de logarithme utilisé : **nats (natural units)** si $\ln_e(\cdot)$ or **bits (binary digits)** if $\log_2(\cdot)$. Sachant que $\ln(a) = \ln 2 \times \log_2 a$, la conversion est facile :

$$I_{nats}(x_i) = \ln(2) I_{bits}(x_i) \approx 0,7 \cdot I_{bits}(x_i)$$

1.3.2 Exemples.

Soit un jeu de 32 cartes. Considérons quelques événements.

Ex 1 : Prob(tirer une figure rouge {♢, ♡} = 1/2. $I = 1$ bit.

Ex 2 : Prob(Dame de pique Q♠) = 1/32. $I = 5$ bits.

Ex 3 : Une source binaire qui produit un bit $x_i \in \{0, 1\}$ toutes les T_b seconds, et les bits sont équiprobables avec $p(0) = p(1) = 1/2$. La quantité d'information dans chaque bit est $I(x_0) = I(x_1) = -\log_2(\frac{1}{2}) = 1$ bit (il n'y pas de surprise !). Ainsi, un bit suffirait pour coder cette source.

Ex 4 : Soit une source DMS, qui fournit un flux binaire des bits indépendants. Prenons un bloc de k bits pour former ou étiqueter un symbole M-aire $X^m, m = 1, \ldots, M = 2^k$, de durée $T_s = kT_b$. L'usage type consiste à mapper chacun des blocs de k bits sur un symbole de modulation d'ordre M, pour ensuite créer un signal temporel à être transmis dans le canal.

Dans cet exemple, les symboles sont distribués selon la loi discrète uniforme, et sont ainsi équiprobables $p(x^m) = \frac{1}{M} = (\frac{1}{2})^k = p^k$.

Question : quelle quantité d'information est dans chaque bloc $X^m = x^m$?

Réponse : $I(x^m) = -\log_2 p(x^m) = -\log_2 \frac{1}{M} = \log_2 2^k = k$ bits !

A noter que $I(\cdot)$ est additive pour les bits indépendants (propriété de $\log$).

A ce stade, plusieurs questions viennent à l'esprit, concernant la source dont les symboles ne sont pas équiprobables.

1. Combien de bits sont nécessaires pour encoder chaque symbole ?
2. Faut-il utiliser le même nombre de bits pour un mot-code correspondant à chaque symbole ou bien les codes de longueur différentes ? (codes à longueur fixe ou variable)
3. Enfin, la question la plus importante : quelle est la quantité d'information minimale dans la source, et ainsi, quelle sera la taille moyenne minimale d'un descripteur ou code, pour encoder ou décrire cette source ?

Intuition : codes plus courts pour les symboles fréquents.

Pour répondre à ces questions, il faut pouvoir caractériser la source.

1.4 Entropie $H(X)$ d'une source discrète

Déf 1.2. *L'entropie $H(X)$ d'une source discrète décrite par les réalisations x_i d'une variable aléatoire (VA) X est définie comme ceci :*

= 1) (math) Auto-information moyenne [Shannon/symbol] = [bits] d'une VA X via ses réalisations :

$$H(X) = \sum_i p_i I(x_i) = -\sum_i p_i \log_2 p_i, \text{ [bits]} \tag{3}$$

= 2) (interprétation) incertitude moyenne dans la VA X ;

= 3) (descripteur) nombre moyen minimal de bits nécessaire pour décrire cette VA.

Ex 5 : Source binaire (générique, non-équiprobable).

$$X = \begin{cases} 1, \text{ avec proba } p \\ 0, \text{ avec proba } 1-p, \end{cases} \tag{4}$$

L'entropie de cette source se calcule comme (voir Fig. 3) :

$$H_2(p) = -p \log_2 p - (1-p) \log_2 (1-p) \text{ [bits]}. \tag{5}$$

Ex 4 (suite) : pour une variable aléatoire de loi discrète uniforme de $(M = 2^k)$ réalisations équiprobables (symboles de la source)

$$H(X) = -\sum_{i=1}^{M} \frac{1}{M} \log_2 \frac{1}{M} = \log_2 M = k \text{ bits}. \tag{6}$$

1.4.1 Propriétés de l'entropie

— Propriété : l'entropie est positive : $0 \leq H(X) \leq \log_2 L$.

Exercice 1 : Prouvez-le !

— En utilisant [Cover, Th 2.6.4] considérez $p(X)$ et $u(X) = 1/L$. En utilisant une mesure qui s'appelle divergence entre les deux lois de probabilité, $D(p||u) \geq 0$, qui est positive, on a $D(p||u) = \sum p \log_2 \frac{p}{u} = \log_2 L - H(X) \geq 0$.

— Maximisation sous contrainte d'un Lagrangien [Cours S. Chen] : $\mathcal{L} = \left(\sum p_i \log_2 \frac{1}{p_i}\right) + \lambda(1 - \sum p_i)$.

— Propriété : l'entropie binaire $H_2(p)$ est concave en p, avec l'incertitude maximale pour les événements équiprobables, ce qui est assez intuitif. « En êtes-vous sûr.e ? — Hmm, fifty-fifty » .

Théorème 1.1. *[Shannon] : il est toujours possible de construire une représentation d'une variable aléatoire (source discrète finie) avec un nombre moyen de bits par symbole, qui est au plus à un (1) bit de l'entropie [Cover] :*

$$H(X) \leq \overline{R} \leq H(X) + 1, \tag{7}$$

où $\overline{R} = \sum_i p_i \cdot \ell_i$, avec $\overline{R}$ [bits], la longueur moyenne des mots-codes, et ℓ_i, la longueur en bits de chaque mot-code pour le i-ième symbole x_i.

Théorème 1.2. *Codage-compression de source [Shannon] : une source discrète X produisant les blocs de n symboles peut approximativement être décrite par $nH(X)$ bits.*

1.5 Information mutuelle

1.5.1 Réalisation individuelle de 2 VAs dépendantes

L'information produite par la source est typiquement transformée, traitée dans la chaine de communication. Ce traitement peut consister en codage, transmission dans un canal bruité, et schématisé comme $X \rightarrow$ traitement $\rightarrow Y$. Par exemple, soit X représente les symboles en entrée du canal, $x_i, i = 1, \ldots, n$. Soit $Y,\ y_j, j = 1, \ldots, m$ représente les sorties observées du **canal bruité**, les symboles bruités. Cette transmission dans un canal bruité peut être décrite via une distribution, ou une fonction de masse conjointe $p(X, Y)$. Il est important de remarquer que si le canal n'avait pas de bruit, le problème de transmission d'information n'existerait pas !

Déf 1.3. *L'***information mutuelle** *$I(x_i;y_j)$ est la quantité d'information que la réalisation y_j fournit sur x_i. En d'autres mots, c'est l'incertitude résiduelle que l'on peut avoir sur x_i, une fois y_j, qui y est statistiquement liée, est observée. C'est donc la différence entre l'auto-information (incertitude) dans la réalisation de x_i, et l'information conditionnelle dans x_i après (a posteriori) avoir observé y_j :*

$$I(x_i;y_j) = I(x_i) - I(x_i/y_j) = \log_2 \frac{p(x_i/y_j)}{p(x_i)} .$$

Propriétes de l'information mutuelle :

Proriété : I est symétrique, $I(x_i; y_j) = I(y_j; x_i)$.

Proriété : $I(x_i; y_j) \in \mathbb{R}$ est un nombre réel, le $\log()$ du rapport des probabilités, peut donc être négative ou positive. Cas particuliers :

— $I > 0 \Leftrightarrow p(x_i/y_j) > p(x_i)$ signifie que la réalisation de y_j réduit l'incertitude sur x_i, la réalisation conditionnelle de (x/y) et ainsi moins informative que (x) seule. C'était attendu—l'observation d'une VA dépendante y apporte moins d'information que ne pourrait l'observation directe de x, inaccessible.

— $I < 0 \Leftrightarrow p(x_i/y_j) < p(x_i)$ induit une confusion : la réalisation de y_j qui ne fait que dépendre de x_i, apporte plus d'information ($p(x_i/y_j) < p(x_i) \iff I(x_i/y_j) > I(x_i)$) sur x_i que la réalisation de x_i seule. [2].

— $I = 0$ lorsque X, Y sont indépendants, c'est à dire, $p(x_i/y_j) = p(x_i)$, c'est un canal « inutile ».

— $I(x_i; y_j) = I(x_i)$, lorsque X, Y sont complétement dépendants. C'est facile à interpréter car correspond au cas du canal sans bruit, où la réalisation y_j décrit complètement x_i, et $p(x_i/y_j) = 1$.

NB : jusqu'à maintenant, ont été considérées x_i, y_j , réalisations individuelles (symboles) des VAs X et Y, et non leurs ensembles complets de réalisations.

1.5.2 (Average) mutual information

En moyennant justement $I(x_i; y_j)$ sur toutes les valeurs que peuvent prendre X et Y par rapport à leur distribution conjointe $p(x, y)$, est obtenue .

Déf 1.4. $I(X; Y)$, **l'information mutuelle moyenne** *entre X et Y :*

$$
\begin{aligned}
I(X;Y) &= \mathbb{E}_{p(x,y)} I(x_i; y_j) = \sum_i \sum_j p(x_i, y_j) I(x_i; y_j) && (8)\\
&= \sum_x \sum_y p(x, y) \log_2 \frac{p(x/y)}{p(x)} = H(X) - H(X/Y) && (9)\\
&= \sum_x \sum_y p(x, y) \log_2 \frac{p(x/y) \cdot p(y)}{p(x) \cdot p(y)} && (10)\\
&= \sum_x \sum_y p(x, y) \log_2 \frac{p(x, y)}{p(x) \cdot p(y)} && (11)\\
&= H(X) + H(Y) - H(X, Y) \, . && (12)
\end{aligned}
$$

Inteprétation 1. $I(X, Y)$ est la réduction de l'incertitude sur X grâce à l'observation de Y, éq (9) ou, en d'autres mots, la connaissance additionnelle sur X apportée par Y. (Les indices i, j ont été abandonnés pour simplifier la notation).

1.5.3 Propriétés de l'information mutuelle moyenne

1. Propriété : $I(X; Y) \geq 0$
   — $H(X)$ est l'incertitude moyenne intrinsèque caractérisant la VA X, avant toute réalisation.
   — $H(X/Y)$ est l'incertitude résiduelle sur X une fois Y a été observé.

   Le conditionnement (connaissance additionnelle) peut uniquement réduire l'entropie, et donc
   $$H(X) \geq H(X/Y).$$

2. Ex : Canal Binaire non-Symétrique. Par commutativité et en utilisant le fait que $P(y/x)$ peuvent être connues a priori par une loi de perturbation, ou la distribution statistique du bruit aléatoire (ex : proba d'erreur $p_0 = p_1 = 1/4$), et $P(y)$ peut être calculée comme distribution marginale $p(y_j) = \sum_i p(y_j/x_i) \cdot p(x_i)$. [Proakis p.83-84].

Ex 6 : Preuves produites devant la justice. Supposons qu'un jugement sur la culpabilité ou non d'un prévenu doit être prononcé, en tenant compte des preuves apportées, des deux types : à décharge, un alibi, par exemple, ou preuve à charge qui est preuve de culpabilité.

Le jugement résultant est donc une source binaire, seules deux réalisations sont possibles : $X = \{COUPABLE, NON - COUPABLE\} = \{C, NC\}$.

La culpabilité X dépend des preuves à décharge ou alibi, notées $Alibi_i = A_i$ ou des preuves de culpabilité $Preuve_k = P_k$ apportées :

$$Y = \{A_1, ..., A_i, ..., A_m, P_1, ..., P_k, ..., P_n\}$$

Supposons qu'une nouvelle preuve à décharge est produite devant le juge (alibi A_i). Celui-ci va donner un terme $I(X = C; A_i) = \log_2 \frac{p(C/A_i)}{p(C)} < 0$, car la probabilité de culpabilité étant donné un alibi $p(C/A_i)$ et plus petite qu'une probabilité de culpabilité seule $p(C)$. Pondéré par $p(C, A_i) > 0$, cela apportera un terme négatif (réduction de culpabilité) à l'information mutuelle moyenne. Mais ! d'autre preuves de culpabilité P_k peuvent peser plus lourd dans la décision de justice. Chaque preuve de culpabilité donnera un terme positif $I(X = C; P_k) = \log_2 \frac{p(C/P_k)}{p(C)} > 0$ dans l'information mutuelle moyenne. Ainsi, le prévenu peut encore être reconnu coupable à l'issue du procès, si $I(X = C; Y) > 0$, même s'il y a eu des preuves à décharge (alibis).

2. Symétrie, facile à voir à partir de l'éq. (11)

$$I(X;Y) = H(X) - H(X/Y) = H(Y) - H(Y/X) = I(Y;X) \tag{13}$$

3. $I(X;Y) = 0$, ssi (si et seulement si) X et Y sont indépendants. Dans ce cas $H(X/Y) = H(X)$.
4. $I(X;X) = H(X)$ auto-information moyenne n'est autre qu'entropie de la source. En effet, dans ce cas X, Y sont complètement dépendantes, et $p(x/y) = 1 \Rightarrow H(X/Y) = 0$. Cela revient à observer directement la source, sans canal, ce qui n'est pas un problème de communication.
5. $I(X;Y) \leq \min\{H(X); H(Y)\} \leq \min\{\log_2 Card\{\mathcal{X}\}; \log_2 Card\{\mathcal{Y}\}\}$.
6. (opt) Attention ! $I(X;Y;Z)$ ne peut pas être défini, n'a pas de sens ! [MacKay, chap. 8].

Ces relations sont représentées graphiquement sur les Fig. 4 et Fig. 5. Il faut toutefois faire attention avec la seconde, les diagrammes de Venn, car la zone d'intersection de trois cercles pouvant exister, cela laisserait à penser que cela représente l'information mutuelle moyenne de trois variables aléatoires, or, c'est faux !

1.6 Entropy conditionnelle et conjointe

Ces deux quantités, notées $H(X/Y)$ et $H(X,Y)$ apparaissent dans la définition de $I(X;Y)$. L'entropie conjointe est définie comme ceci :

$$\begin{aligned} H(X,Y) &= -\sum_x \sum_y p(x,y) \log_2 p(x,y) \\ &= H(X) + H(Y/X) = H(Y) + H(X/Y) \end{aligned} \tag{14}$$

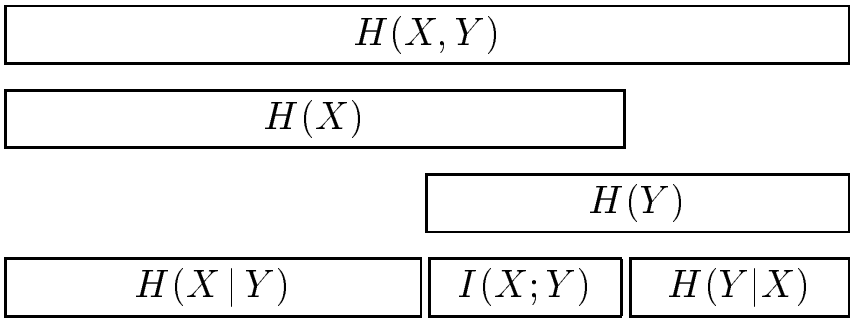


FIGURE 4 – *Relations entre les quantités définies par la théorie de l'information 1 [MacKay].*

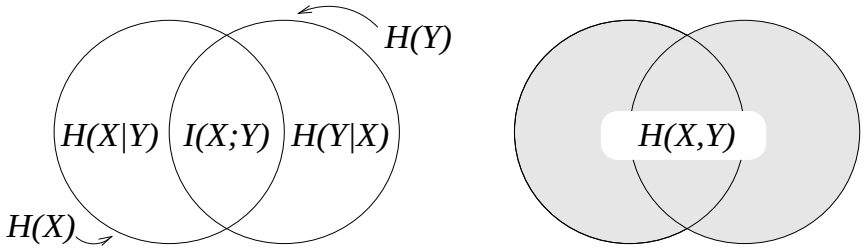


FIGURE 5 – *Relations entre les quantités définies par la théorie de l'information 2 [MacKay].*

1.6.1 Propriétés

1. $H(X/Y) \neq H(Y/X)$ asymétrique.
2. $H(X,Y) = H(X) + H(Y/X) \leq H(X) + H(Y)$, cela découle directement des propriétés de $I(X;Y)$. Il y aura égalité si et seulement si X, Y sont indépendantes, car dans ce cas $H(Y/X) = H(Y)$.
3. Généralisation de l'entropie conjointe de pour n variables aléatoires, $X_1, \ldots, X_n$, pouvant, par exemple, représenter un vecteur aléatoire.
   Règle de chaine d'entropies *(chain rule)*.

$$H(X_1, \ldots, X_n) = \sum_{i=1}^{n} H(X_i/X_{i-1}, \ldots, X_n) \leq \sum_i H(X_i) \tag{15}$$

   Cette égalité est facile à observer à partir de la règle de chaine sous-jacente pour les probabilités sous le log. La dernière inégalité vient directement de la propriété précédente.
4. Si $X_1, \ldots, X_n$ sont le VAs iid (indépendantes et identiquement distribuées, chacune ayant pour entropie $H(X)$), alors $H(X_1, \ldots, X_n) = nH(X)$. Cela renvoie à un des théorèmes de Shannon, évoqués précédemment, Théorème 1.2.

1.7 Etude détaillée : Canal Binaire Symétrique (CBS)

La source est décrite par une VA binaire X, pas nécessairement équiprobable, $P(X = 0) = q$ ou $P(X = 1) = 1 - q$. La probabilité d'erreur dans le canal est symétrique (d'où son nom) $P(Y = 1/X = 0) = P(Y = 0/X = 1) = p$, et si l'absence d'erreur avec : $(1 - p)$. Ce canal est schématisé sur la Fig. 6. Le fonctionnement est le suivant : chaque bit transmis par la source, sur temps-bit, peut subir (avec (p)) ou non (avec $(1 - p)$) l'erreur dans le canal, être inversé.

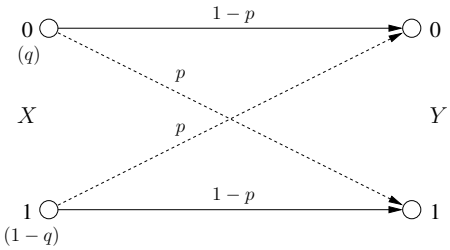


FIGURE 6 – *CBS—Canal Binaire Symétrique.*

$I(X;Y)$ du CBS peut être vu comme une fonction de p, q. C'est tracé sur Fig. 7 sous forme de famille de courbes $I = fct(q)$, paramétrées par p.

1.7.1 Capacité du CBS

En règle générale, la capacité d'un canal est donnée par la maximisation de l'information mutuelle sur la distribution de la source $q(X)$.

$$C = \max_{q(x)} I(X;Y) \ . \tag{16}$$

Pour un canal simple de type CBS, on observe empiriquement à partir de la Fig. 7 le fait que $I(X;Y)$ atteint son maximum lorsque $q = 1/2$, et ce pour toutes les valeurs de p, probabilité d'erreur dans le canal.

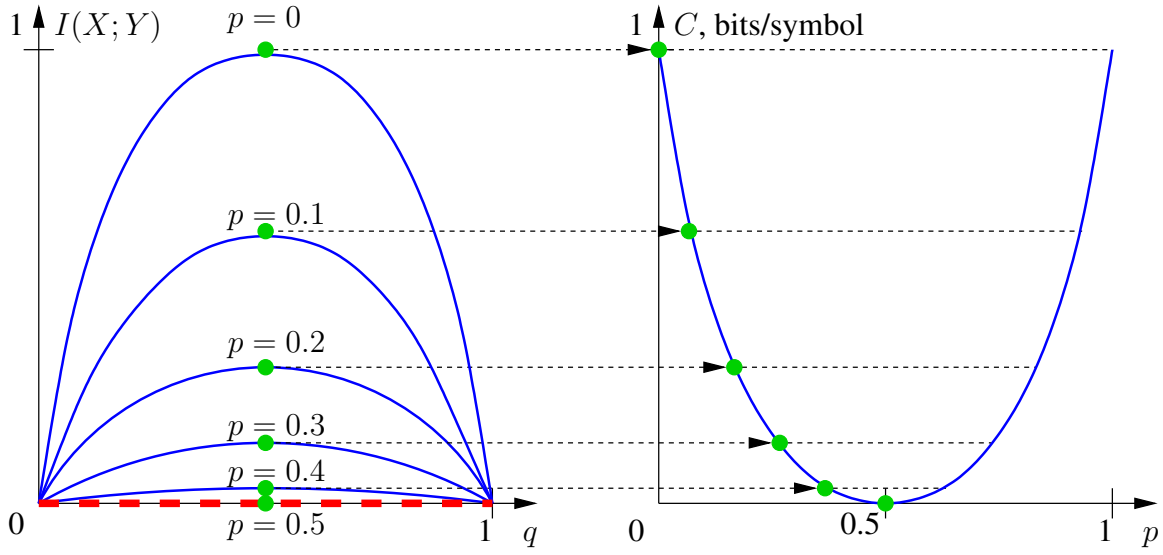


FIGURE 7 – *Information mutuelle et capacité du CBS.*

Pour ce canal, une expression analytique de la capacité peut être obtenue (donnée sans démonstration). Cette capacité C_{BSC} est présentée sur la Fig. 7 :

$$C_{BSC} = 1 - H_2(p) \ . \tag{17}$$

Exercice 3(*) : démontrez-le ! Astuce : dériver $I(X;Y)$ par rapport à p, q. Puis, vérifier que son maximum sur $p(X)$ est obtenu lorsque la source est équiprobable, $q = 1/2$, $\forall p$, c'est à dire :

$$C_{BSC} = \max_{q(x)} I(X;Y) \tag{18}$$

$$= \underbrace{1}_{H(X)=H_2(q=\frac{1}{2})=1} \underbrace{-(p\log\frac{1}{p} + (1-p)\log\frac{1}{1-p})}_{H(X/Y)=H_2(p)} . \tag{19}$$

Notons, que $H(X/Y) = H_2(p)$ est l'entropie conditionnelle qui est égale dans ce cas à l'entropie binaire, mais qui n'est pas celle de la source.

1.7.2 Propriétés du CBS

A partir de l'équation (19), il suit :

1. Pour $p = 1/2$, le canal est inutile, car $I(X;Y) \equiv 0, \forall q$. C'est le pire cas qui peut se produire, le $50/50$. Sa capacité est alors $C_{CBS}(p = 1/2) = 0$.
2. Pour $p = 0$, or $p = 1$, il n'y a aucune erreur, car la sortie du canal Y dépend complètement de son entrée X, et ainsi $I(X;Y) = H(X)$ (la courbe supérieure). Son maximum est atteint lorsque la source binaire est équiprobable, $q = 1/2$. Ainsi, la capacité du CBS sans erreur est $C_{BSC}(p = 0, q = 1/2) = 1$ bit par utilisation du canal.
3. Notons, que pour $1/2 < p \leq 1$, on peut inverser 0 et 1 en sortie du canal, et C_{BSC} devient symétrique autour de $p = 1/2$.

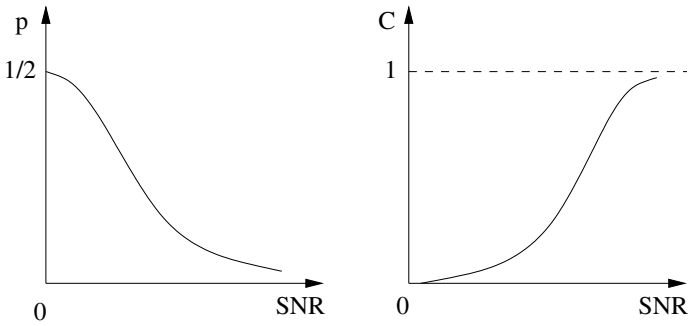


FIGURE 8 – *CBS : $Prob(error)$ et C en fonction du RSB.*

Typiquement, pour une modulation à deux états, $p = Prob(error)$ dans un canal est une fonction monotone décroissante du RSB (Rapport Signal-sur-Bruit). Par conséquent, sa capacité C augmente de manière monotone avec RSB qui augmente, comme esquissé sur Fig. 8 [Proakis].

1.8 (Option) Information mutuelle vue comme distance de Kullback-Leibler (ou divergence).

$$D(q||u) = \sum q(x)\log\frac{q(x)}{u(x)} = \log_2 L - H_q(X) ,$$

Cette quantité est une mesure de différence entre deux distributions de probabilité. Interprétation : si, faute de connaissance sur la vraie distribution de la source $q(x)$, l'hypothèse d'une équiprobabilité

de la source $u(x) = 1/L$ est faite [3], il sera nécessaire d'utiliser D bits en plus pour représenter, encoder cette source ayant pour entropie $H(X)$, par rapport au nombre de bits qui serait nécessaire, si la vraie distribution était connue. C'est donc une perte.

2 Codage et compression de source. Codage des symboles

L'objectif de la compression de la source, est de représenter les symboles de la source avec le plus petit nombre de bits pour chacun, avant de les traiter (encoder, moduler, transmettre,...) Puis, être capable de reconstruire le message initial de la source.

La limite minimale de la compression de la source est son entropie $H(X)$

Les méthodes peuvent être classées en deux catégories :

— Compression sans pertes ou entropique *(lossless)* : utiliser un nombre minimal de bits, permettant la restauration parfaite du message de la source. Usage type : data. **C'est cette approche qui sera étudiée dans ce cours**.

— Compression avec pertes : utiliser un nombre minimal de bits, tout en acceptant une quantité contrôlable de distorsion D_{max} dans le message source reconstruit, ce qui ne nuira pas à sa perception. Usage : parole, musique.

2.1 Codage source de longueur fixe

Soit un canal discret sans mémoire (DMS) produisant les symboles $x_1, \ldots, x_L$ à partir d'un alphabet de taille finie L, par example, alphabet d'un langage naturel (on accepte ici cette approximation erronée,—la sémantique de la langue, l'orthographe, rendent les lettres successives dépendantes entre elles), ou un nombre fini de niveaux de quantification du signal pour un CAN.

Notons $R \in \mathbb{N}^+$ [bits/symbole source], la longueur du code binaire, la même pour chaque symbole de la source, il y a donc 2^R mots-codes possibles.

$$R = \begin{cases} \log_2 L, & \text{if } L = 2^R, \text{si } L \text{ est la puissance de 2 ,} \\ \lfloor \log_2 L \rfloor + 1, & \text{sinon.} \end{cases} \tag{20}$$

où $\lfloor \log_2 L \rfloor$ représente l'arrondi au plus petit entier inférieur, l'operation 'floor()'. Cela peut être réécrit de la manière suivante :

$$\log_2 L \leq R < \log_2 L + 1 \ . \tag{21}$$

Ex 8 : source binaire $L = 2$ avec les réalisations $\{0, 1\}$. Evidemment, $R = 1$ [bit/symbole].

Ex 9 : dé : Dé de $L = 6$ faces $\{1, 2, 3, 4, 5, 6\}$. Ici $R = \lfloor \log_2 6 \rfloor + 1 = 3$ [bits/symbole].

L'utilisation des propriétés de l'entropie permet la généralisation suivante :

$$H(X) \leq \log L \leq R < \log L + 1 \tag{22}$$

Ainsi, si les symboles sont équiprobables et indépendants, la partie droite de l'équation devient $H(X) = \log L$ et ainsi (comparez au Théorème 1.1) :

$$H(X) \leq R < H(X) + 1 \tag{23}$$

3. ...qui peut être complètement erronée, mais c'est le mieux que l'on puisse faire sans connaissance aucune sur la source.

Déf 2.1. *L'efficacité du code est définie comme ceci :*

$$\textit{Eff} = \frac{H(X)}{R} \leq 1. \tag{24}$$

Pour être considéré comme efficace, un code doit avoir $Eff \approx 1 = 100\%$.

Ex 10 : Opérations arithmétiques. Il s'agit de les encoder en code binaire. L'alphabet de taille $L = 4$ de cette source représente les opérations $\mathcal{A} = \{+, -, \times, \div\}$, et $R = \log_2 L = 2$ [bits/-symbole]. Equi-probables, donc $H(X) = \log_2 L = 2 = R$. Pour ce cas particulier de source équi-probable et dont le nombre de symboles est la puissance de 2, le code-index (énumération binaire) est efficace, Eff $= H(X)/R = 1$.

Si la taille de l'alphabet L n'est pas la puissance de 2, la partie à droite de l'équation (23) sera $R < H(X) + 1$. Cela reste à un bit de l'entropie, mais un code de longueur fixe peut être peu efficace, Eff ≈ 0.7 ou même moins. Ce sera observé pour les alphabets de petite taille L, et une fonction de masse $p(x)$ disparate, lorsqu'un des événements est très probable, avec $p(x_i) \approx 1$.

Ex 11 : Dans la ville de Tonnerre (89-Yonne), il est très fréquent d'avoir un épisode orageux en pleine journée ensoleillée de juillet, cet événement est décrit par une VA X. La probabilité d'orage est proche de $p = 0,99$, sinon, la journée reste claire avec la probabilité $1 - p = 0,01$. L'entropie de cette source des données météorologique est $H_2(p) = 0,08$ [bits/symbole].

Pour sauvegarder et transmettre les données collectée sur une longue période, il faut choisir une méthode de codage. Un choix naïf serait d'encoder cette source binaire sur $R = 1$ bit. Un tel code serait très peu efficace : $Eff = H(X)/R = 0,08 = 8\ \%$, très faible.

2.1.1 Codage par blocs-symboles à longueur fixe

Problème : comment augmenter l'efficacité du codage pour l'exemple précédent ?
Solution : agrandir artificiellement la taille de l'alphabet L, seul paramètre qu'il est possible de contrôler, les probabilités ne sont pas modifiables.
Idée : regrouper plusieurs symboles successifs en blocs, en créant un alphabet de plus grande taille de tels symboles composites.

Soit les blocs de J symboles, donnant le nombre total de L^J (L puissance J) blocs possibles, la taille de nouvel alphabet L_J, (J est juste un indice pour démarquer de L).

Ex 11 : $L = 4$ **symboles** (x_1, x_2, x_3, x_4) **groupés par blocs de** $J = 2$ **symbols.** Il y a en tout $L^J = 4^2 = 16$ blocs :

$$\begin{aligned} \{v_{11} &= (x_1, x_1), v_{12} = (x_1, x_2), v_{13} = (x_1, x_3), v_{14} = (x_1, x_4), & (25)\\ v_{21} &= (x_2, x_1), v_{22} = (x_2, x_2), \ldots, v_{44} = (x_4, x_4)\} & (26) \end{aligned}$$

Le codage de ces blocs en utilisant les mots-codes de même longueur fixe $R_J \in \mathbb{N}^+$ [bits/-symbole], est décrit par les équations similaires à celles utilisées pour les symboles individuels précédemment :

$$\begin{aligned} \log_2 L^J &\leq R_J < \log_2 L^J + 1 & (27)\\ \Leftrightarrow\ J \log_2 L &\leq R_J < J \log_2 L + 1 & (28)\\ \Leftrightarrow\ \log_2 L &\leq \frac{R_J}{J} < \log_2 L + \frac{1}{J} & (29) \end{aligned}$$

$\tilde{R} = R_J/J \in \mathbb{Q}^+$ [bits / symbol] n'est en général plus un nombre entier (de bits) et ne peut donc pas être la longueur d'un mot-code. C'est plutôt une métrique pour mesurer l'efficacité du code, ramenée à un symbole de l'alphabet initial. En choisissant les codes avec J, nombre de symboles par bloc très grand, $\tilde{R}$ va diminuer et éventuellement tendre vers $\log_2 L$. Remarquons, que J ne doit pas nécessairement être très grand,—en général, une valeur assez raisonnable de J entre 3 et 8 [symboles/bloc] sera suffisante pour qu'un code à longueur fixe atteigne $Eff \approx 98\%$.

Ex 12 : Soit une source DMS avec $L = 10$ symboles équiprobables et entropy $H(X) = \log_2 10 = 3.32$ [bits/symbole]. Ainsi, 4 bits devraient être utilisés pour un code de longueur fixe. Considérons alors les blocs de J symbols $\Leftrightarrow$ mots-codes de longueur $R_J = \lfloor J \log_2 L \rfloor + 1$. La table Tab. 1 présente les paramètres d'un tel code pour différentes valeures de J, nombre de symboles de source groupés pour former un symbole composite, qui sera codé en binaire.

J	$J \log_2 L$	R_J [bits]	Taille nouvel alphabet, L^J	Nb de codes 2^{R_J}	Métrique $\tilde{R} = \frac{R_J}{J}$ [bits/symb]	$Eff = \frac{H(X)}{\tilde{R}}$
1	$3.32 <$	4	$10 <$	16	4	83%
2	$6.64 <$	7	$100 <$	128	3.5	95%
3	$9.96 <$	10	$1000 <$	1024	3.33	99.7%
4	$13.3 <$	14	$10000 <$	16384	3.5	95%
5	$16.6 <$	17	$100000 <$	131072	3.4	97.7%
6	$19.9 <$	20	$1000000 <$	1048576	3.33	99.7%

TABLE 1 – Codes de longueur fixe pour blocs de symboles : métrique et efficacité en fonction de J.

Questions :
— Pourquoi l'efficacité ne croit-elle pas de manière monotone avec J ?
— Quelle propriété essentielle de la source n'a pas été prise en compte dans cette méthode de codage ?

2.2 Compression de source sans pertes

2.2.1 Mots-codes de longueurs différentes

—utilisés lorsque les symboles de la source ne sont pas équiprobables.
Idée : (symboles plus probables, plus fréquents) $\Rightarrow$ (codes plus courts).
Exemple : code de Morse [4]. Pour les lettres fréquentes $\Rightarrow$ utiliser les mots-codes plus courts, ce qui requiert moins de gestes manuels de la part du télégraphiste, rendant la signalisation plus rapide.

Le propriétés souhaitées de tels code sont :
— Unicité du décodage (non-ambiguïté)—les mots-codes de longueur différentes, doivent être décodés de manière non-ambiguë par le récepteur.
— Instantanéité du décodage—faible délai algorithmique. Cette propriété stipule qu'un mot-code doit pouvoir être décodé immédiatement après que son dernier bit ait été reçu, sans attendre la réception des mots-codes suivants.

4. Il est intéressant de noter que le code de Morse a été mis en œuvre à la fin du 19^e siècle, 50 ans avant le développement de la théorie de l'information par Shannon, et des mesures entropiques des codes.

Ces deux règles se résume en « condition du préfixe », qui s'énonce ainsi : aucun mot-code n'est un début d'un autre. On appellera ce type de code les **codes préfixes**.

Formellement, cela signifie qu'un mot-code c_k de longueur ℓ_k ayant pour ses éléments les bits $(b_1, \ldots, b_k)$, ne contient en son début aucun autre mot de code c_m de longueur $\ell_m,\ 1 \leq \ell_m \leq \ell_{k-1}$ contenant les mêmes bits $(b_1, \ldots, b_{\ell_m})$.

Considérons une source DMS aux symboles x_1, x_2, x_3, x_4, et leurs probabilités correspondantes p_i, comme présenté dans la Table 2.2.1. L'analyse de 3 codes est proposée.

Symboles	$p(x_i)$	Code I	Code II	Code III
x_1	1/2	1	0	0
x_2	1/4	00	01	10
x_3	1/8	01	011	110
x_4	1/8	10	111	111
	$H(X) = 1,75$	$\overline{R} = 1,5$*	$\overline{R} = 1,75$	$\overline{R} = 1,75$

TABLE 2 – Comparaison des 3 codes.

— Code I est erroné par sa nature. Pour le voir, il faut remarquer que la longueur moyenne de ses mots-codes est inférieure à l'entropie, $\overline{R} < H(X)$, ce qui contredit la définition même de celle-ci comme longueur moyenne minimale d'un code nécessaire pour représenter cette source. Ainsi, ce code est indécodable, voire pire—menant à la propagation catastrophique des erreurs. Soit, par exemple, une séquence reçue $[1001]$. Cela peut être décodé soit comme $[10|01] = [x_4, x_3]$, soit comme $[1|00|1] = [x_1, x_2, x_1]$.

— Code II permet l'unicité de décodage. Toutefois, il n'est pas à décodage instantané, car la condition du préfixe n'est pas satisfaite. Soit, une séquence reçue $[01001]$. Le premier $[0]$ peut être vu soit comme x_1 ; soit comme un début de x_2 $[01]$; soit du x_3 $[011]$. Il faudra donc cumuler plus de bits pour pouvoir décoder ce code, cela nécessite la gestion d'indexation de mémoire.

— Code III est cette fois le bon ! Il satisfait les deux conditions annoncées.

NB : Souvent, il peut être utile de représenter les mots-codes comme nœuds dans un arbre binaire, voir Fig. 9 pour le code III.

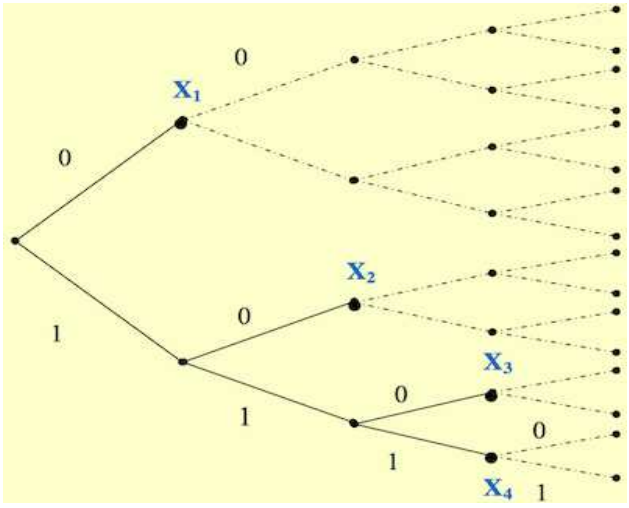


FIGURE 9 – *L'arbre binaire et les mots-codes pour le Code III.*

A ce stade de présentation, l'objectif se précise : comment concevoir un code **efficace, uniquement décodable, si possible instantané**, de sorte à avoir la longueur moyenne $\overline{R}$ des mots-codes la plus petite, donc la plus proche de l'entropie.

$$H(X) \lessapprox \overline{R} = \sum_{i=1}^{L} p(x_i)\ell_i.$$

2.2.2 Inégalité de Kraft-McMillan

est la condition nécessaire et suffisante pour l'existence d'un code binaire qui i) remplit la condition de préfixe ii) instantané et iii) uniquement décodable.

Théorème 2.1. *Soient L mots-codes d'un code binaire instantané et uniquement décodable, avec les longueurs ℓ_i ,*

$$1 \leq \ell_1 \leq \ldots \leq \ell_i \leq \ldots \leq \ell_L \,, \tag{30}$$

alors, l'inégalité suivante est vérifiée :

$$\sum_{i=1}^{L} 2^{-\ell_i} \leq 1. \tag{31}$$

NB : dans l'équation (30), l'égalité dans $\ell_i \leq \ell_{i+1}$ signifie qu'il peuvent exister les mots-codes de même longueur.

Preuve :

1. Condition suffisante. Si les longueurs des mots-codes $1 \leq \ell_1 \leq \ldots \leq \ell_i \leq \ldots \leq \ell_L$ satisfont la condition de préfixe, alors l'inégalité de Kraft est vérifiée.

   Considérons un arbre binaire de profondeur maximale $\ell = \ell_L$: chaque nœud de niveau précédent $(d-1)$ donne naissance à deux branches menant à deux nœuds-fils de niveau suivant (d), où $1 \leq d \leq \ell_L$. Ainsi, **chaque niveau correspond à une longueur des mots-codes**. Le niveau (0) est la racine de l'arbre (il n'existe pas de mot-code de longueur nulle, 0), le niveau $(d = 1)$ peut représenter un mot-code de longueur 1 bit, si un tel mot-code est utilisé, et ainsi de suite. Dans un tel arbre, il y a 2^d nœuds-branches de niveau d et ainsi 2^{ℓ_L} feuilles au dernier niveau ℓ_L. L'étiquette de chaque branche correspond à un bit dans un mot-code, et ainsi, les étiquettes des branches menant depuis la racine à un nœud au niveau d donné peuvent composer un mot-code, on dira par abus du langage « le mot-code d'un nœud ».

   Ce nœud au niveau d possède $2^{\ell_L - d}$ nœuds feuilles au niveau maximal ℓ_L. Ainsi, le mot-code de ce nœud serait l'ancêtre (le préfixe) pour tous les autres qui en émanent. Il ne doivent donc pas être choisis pour les mots-codes, afin d'assurer l'unicité du décodage.

   Ainsi, la **condition de préfixe** doit être remplie : **aucun mots-code n'est ancêtre (préfixe) d'un autre**.

   Vient ensuite la construction du code, respectant cette condition.

   — Choisir un nœud d'un niveau ℓ_1 quelconque, son mot-code sera le premier mot-code c_1 du code en construction. Sans perdre de généralité, prenons comme premier mot-code pour un symbole source donné [5] un code de longueur $\ell_{(i=1)} = \ell_1 = 1$ bit. La condition de préfixe interdit l'utilisation de toute la partie de l'arbre émanant de ce nœud, cela élimine du choix des codes possibles $2^{\ell_L - \ell_1}$ feuilles (fraction $1/2^{\ell_1} = 1/2$ de toutes les feuilles).

5. Lequel et pourquoi est discuté dans la section suivante.

— Procéder de la même manière pour choisir un mot-code suivant c_2 en commençant par un quelconque nœud disponible de niveau $\ell_2 \geq \ell_1$, selon (30), ici $\ell_2 = 2$. Cela impose d'éliminer encore $2^{\ell-\ell_2}$ feuilles (fraction $1/2^{\ell_2} = 1/4$ de toutes).
— Continuer ainsi, jusqu'à ce que le dernier mot-code de la feuille au niveau ℓ_L soit attribué (il est possible d'en avoir deux à la même longueur ℓ_L, si ℓ_L est impaire).

Observons, qu'à un certain niveau interne, au milieu de l'arbre, $\ell_j, j < L$, la fraction de toutes les feuilles éliminées par ce procédé est [6] :

$$\sum_{i=1}^{j,j<L} 2^{-\ell_i} < \sum_{i=1}^{L} 2^{-\ell_i} \leq 1 \; . \tag{32}$$

Par conséquent, il y a toujours un nœud disponible de niveau $i > j$ pour que son code soit attribué à un autre symbole.

2. Condition nécessaire. Soit les entiers positifs ℓ_i qui vérifient l'inégalité de Kraft-McMillan. Alors, il existe un code préfixe, avec les longueurs des mot-codes ℓ_i (mais le code lui-même reste à construire !) Le nombre total des feuilles éliminées est :

$$\sum_{i=1}^{L} 2^{\ell_L - \ell_i} = 2^{\ell_L} \sum_{i=1}^{L} 2^{-\ell_i} \leq 2^{\ell_L} \; .$$

En divisant les parties gauche et droite par 2^{ℓ} donne un résultat escompté :

$$\sum_{i=1}^{L} 2^{-\ell_i} \leq 1.$$

Quelques remarques intéressantes :
— Par opposition, si l'inégalité de Kraft n'est pas satisfaite pour un code aux longueurs ℓ_i choisies, le code n'est pas uniquement décodable, il est ambiguë.
— Si l'égalité stricte est satisfaite, le code est dit complet.
— La condition de préfixe est plus forte que juste l'unicité du décodage, car un code peut être décodable mais non-instantané (Code II).

2.3 Codage de Huffman

L'algorithme de Huffman permet de construire un code qui est **efficace (optimal), instantané and uniquement décodable**.
Optimalité : la longueur moyenne $\overline{R}$ des mots-codes générés par l'algorithme de Huffman et la plus petite de tous les autres codes.

Exemple. 4 symboles $\{x_1, \ldots, x_{L=4}\}$ avec $p_1 = 0,65;\ p_2 = 0,2;\ p_3 = 0,15; p_4 = 0,1$. Calculer l'entropie de la source. Construire un code Huffman. Arbre.

Solution :
— L'entropie de cette source est de $H(X) \approx 1.61$.

6. L'inégalité "< 1" dans la partie droite vient de la somme des n premiers termes d'une série géométrique (pour $r \neq 1$) : $s_n = a + ar + ar^2 + ar^3 + \cdots + ar^{n-1} = \sum_{k=0}^{n-1} ar^k = a\left(\frac{1-r^n}{1-r}\right)$, où a est le premier terme dans la série, et r son ratio. Dans notre cas, $a = 1/2$, $r = 1/2$, et ainsi $s = 1 - (\frac{1}{2})^n < 1$.

L'égalité "$= 1$" est obtenue dans le cas où il existent deux mot-codes de même longueur maximale n, et ainsi, le dernier terme $(\frac{1}{2})^n$ sera compté deux fois dans la somme, donnant 1.

— Les codes construits selon Huffman sont $c_1 = [0], c_2 = [11], c_3 = [100], c_4 = [101]$.
— La longueur moyenne des mots-codes produits par l'algorithme de Huffman est $\overline{R} = 1,8$ [bits/symb]. Ainsi, pour le code Huffman $Eff_H = \frac{H(X)}{\overline{R}} \approx 0,89$.
— C'est mieux en comparaison avec un code à longueur fixe qui a besoin $R = 2 = \log_2 4$, and $Eff = \frac{1,6}{2} = 0,8$, même pour cet alphabet de faible taille.

2.3.1 Algorithme de Huffman

1. Ordonner, ou trier tous les symboles (ou ensembles) dans l'ordre décroissant de leur probabilités : $P(x_i) \ldots \geq P(x_j) \ldots \geq P(x_k) \geq P(x_\ell)$.
2. Combiner les 2 symboles (ensembles) les moins probables (x_k, x_ℓ) dans un ensemble (x'_k). Additionner leurs probabilités, pour obtenir la probabilité de ce nouvel ensemble $P(x'_k) = P(x_k) + P(x_\ell)$.
3. Marquer :
   — Pour les symboles (ou ensembles) les plus probables des deux ayant composé le nouvel ensemble auxe étapes 1),2) , marquer le bit de poids faible suivant à gauche par $\ll 0 \gg$(qui marque alors tous les symboles contenu dans cet ensemble)
   — Pour l'autre symbole (ensemble), le moins probable des deux, marquer par $\ll 1 \gg$.
   Répéter les étapes $(1) - (2) - (3)$, jusqu'à n'avoir plus qu'un seul ensemble, comprenant alors tous les symboles et dont la probabilité cumulée est $Prob = 1$.

Récupérer les codes c_i de chaque symbole x_i.
Notons, que cela peut être fait de deux manières : soit lors des marquages, en stockant avec chaque symbole son descripteur (code) mis à jour, soit en attendant la fin du procédé, et en remontant cet arbre réalisé sous forme d'une liste chainée, jusqu'à chaque symbole, en marquant le plus/moins probable par $\{0, 1\}$

Example [MCH]. Source de 8 symboles $x_i, i = 1, \ldots, L = 8$, avec $p(x_i)$,, son entropie est $H(X) = 2,62$. L'arbre du code Huffman est présenté sur la Fig. 10. La longueur moyenne des mots-codes résultant est $\overline{R} = 2.7$. L'efficacité de ce code est alors $Eff_H = \frac{H(X)}{\overline{R}} = 97\%$. Celle-ci est bien supérieure à l'efficacité du code de longueur fixe avec $R = 3 = \log_2 L$, and $Eff = \frac{2.62}{3} = 87\%$.

Exercice. Question : Peut-on faire encore mieux que Huffman, surtout pour les alphabet de petite taille ? Exemple : x_1, x_2, x_3 with $p(x_1) = 0,55,\ p(x_2) = 0,3,\ p(x_3) = 0,15$. The Huffman code is simple [0], [10], [11].
Astuce : code Huffman sur les blocs de J symbols. Essayez, par exemple, $J = 3$. Quelle est l'efficacité Eff d'un tel code ?

2.3.2 (Option) Huffman sur les blocs de symboles

Dans les exemples précédents, il a été envisagé d'appliquer le code Huffman sur les symboles ou blocs de symboles issus de la source sans mémoire, DMS. Cette approche de codage Huffman sur les blocs est également utilisable pour les sources stationnaires, dont plusieurs symboles sont inter-dépendants.

Soit une source produisant les tranches de J symboles dépendants. L'entropie d'une telle source peut être notée $H(X_1, \ldots, X_J)$, et peut être ramenée à une sorte de métrique comparative, $H_J(X) = H()/J$ entropie moyenne par symbole. L'application du code Huffman à ces blocs-symboles donnera le code à longueur moyenne des descripteurs (mots-code) notée $\overline{R}_J$.

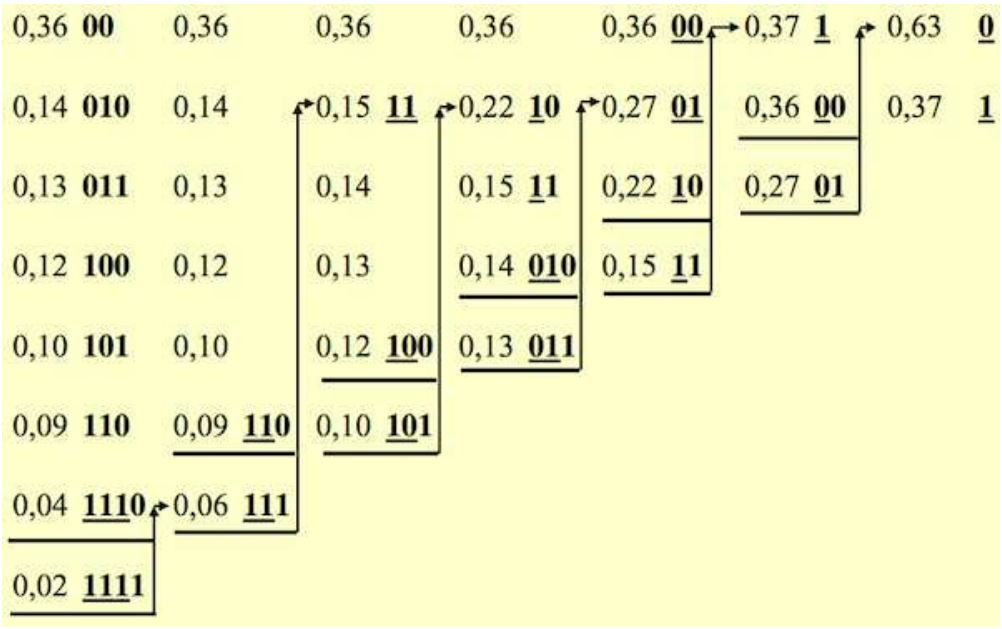


FIGURE 10 – *Huffman code tree.*

$$H(X_1, \ldots, X_J) \leq \overline{R}_J < H(X_1, \ldots, X_J) + 1.$$

En divisant par J, on obtient la métrique : $\overline{R} = \overline{R}_J / J$ [bits/symbole] :

$$H_J(X) \leq \overline{R} < H_J(X) + \frac{1}{J}.$$

Cela permet d'observer, qu'en prenant les blocs avec le nombre de symboles J grand, la longueur moyenne des mots-codes $\overline{R}$ peut être faite très proche de l'entropie $H_J(X)$. Dans la limite $J \to \infty$, $\overline{R}$ est comprise dans (avec $\varepsilon \to_{J \to \infty} 0$) :

$$H_\infty(X) \leq \overline{R} < H_\infty(X) + \varepsilon.$$

En conclusion, notons que la construction d'un code Huffman par symbole ou par blocs requiert la connaissance de la fonction de masse de probabilité des symboles $p(x_i)$ ou conjointe d'un bloc de J symboles, souvent difficile, et parfois impossible à estimer ou à évaluer...

2.4 Codage en flux. Algorithme Lempel-Ziv

Le code Lempel-Ziv-Welsh est justement un algorithme universel de compression des données, qui ne dépend pas de et ne nécessite pas la connaissance des probabilités de la source. C'est une des méthodes les plus couramment utilisées dans les systèmes, par exemple dans le format de compression **.zip**.

2.4.1 Principe de LZW

— Identifier et stocker à partir du flux des bits en sortie de la source, chaque nouvelle « phrase »— séquence de bits qui n'a pas encore observée.
— Construire le **dictionnaire du code comportant toutes les phrases différentes** observées. **La phrase est nouvelle lorsqu'elle diffère de celles précédemment stockées d'un seul dernier bit.**

— Encoder les phrases.
Un mot-code pour chaque nouvelle phrase est obtenu à partir du numéro (ou la position) de la phrase précédente dans l'ordre de l'observation, déjà stockée, en le concaténant avec justement ce dernier bit dans le nouvelle phrase, qui les distingue. Une nouvelle phrase ainsi obtenue est stockée à son tour, dans sa position ordonnée dans la table de codage. Le numéro ou la position est entendu en binaire.
$[N\_pos\_prev|last\_bit]$.

Example [MCH]. Codage LZW pour une séquence $[10101101001001110101000...]$. A chaque étape, les phrases existantes sont soulignées.

1. Identifier une nouvelle phrase, par ex : $i = [10|0] = [prev\_phrase = 10|last\_bit = 0]$, en comparant avec la.es phrase.s déjà stockées dans le dictionnaire $[10]$.
2. Prendre sa position $[0011]$,—celle-ci sera la première partie d'un mot-code pour la nouvelle phrase de l'étape 1), $[100]$.
3. Construire un mot-code : $c = [N\_pos\_prev|last\_bit] = [0011|0]$. Stocker dans la liste à sa position dans l'ordre d'observation $[0111]_{d=7}$.

	Position	Dictionnaire	Mots code
1	0001	1	00001
2	0010	0	00000
3	0011	10	00010
4	0100	11	00011
5	0101	01	00101
6	0110	00	00100
7	0111	100	00110
8	1000	111	01001
9	1001	010	01010
10	1010	1000	01110

FIGURE 11 – *LZW encoding procedure.*

La compression forte n'est pas évidente à constater dans cet exemple (29 bits de la source $\rightarrow$ 50 bits codés), mais ce code devient très efficace pour de longues séquences de bits (ex : fichiers stockés sur disque).

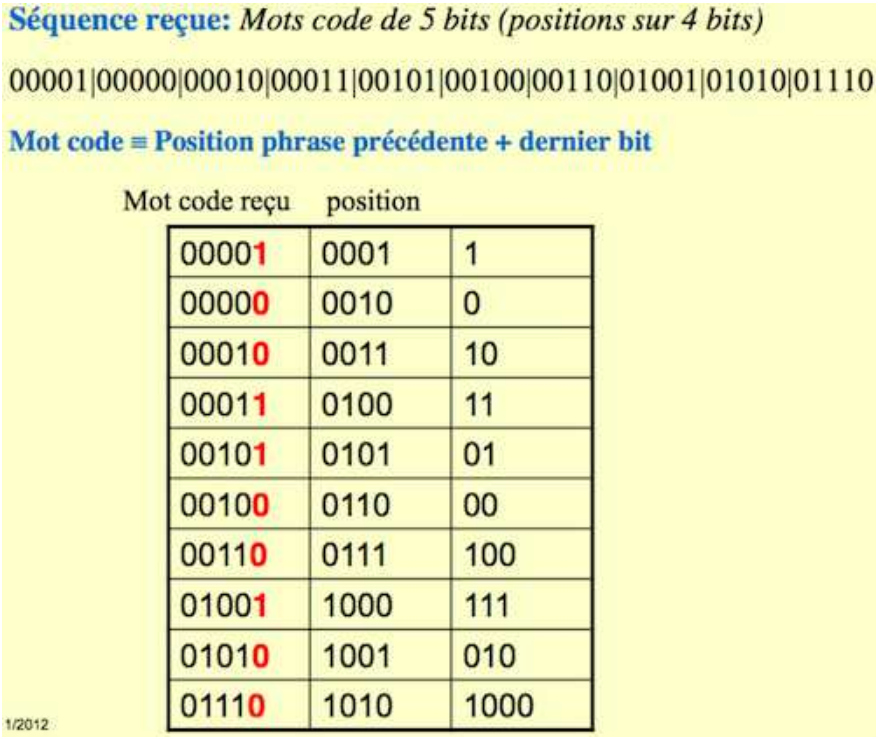
2.4.2 Décodage LZW

Le décodeur effectue une procédure quasi-identique—il construit la copy du dictionnaire et décode en même temps.

Procédure :

— La séquence codée est découpées en mots-codes, ici de longueur 5 bits.
— Dans chaque mot-code, le préfixe donne le numéro de la phrase déjà décodée avant, qui diffère de celle encodée par ce mot-code en son dernier bit. La phrase reconstruite et donc $[phrase\_prec|last\_bit\ CW]$.

Ce codage par flux sous-entend les tables de tailles infinie. En pratique, cela se réalise à l'aide d'une mémoire tampon glissante, en vidant les anciennes entrées de la table.



Séquence reçue: *Mots code de 5 bits (positions sur 4 bits)*

00001|00000|00010|00011|00101|00100|00110|01001|01010|01110

Mot code ≡ Position phrase précédente + dernier bit

Mot code reçu	position	
00001	0001	1
00000	0010	0
00010	0011	10
00011	0100	11
00101	0101	01
00100	0110	00
00110	0111	100
01001	1000	111
01010	1001	010
01110	1010	1000

1/2012

FIGURE 12 – *LZW decoding procedure.*

Références

— Codage arithmétique [D. MacKay]. Application : DASHER.
— Codage par ondelettes.
— Codage par ADN. Journal *Nature* tous les 154 sonnets de Shakespeare (ASCII) encodés via ADN. Emily M. LeProust (anc. CPE Lyon) en co-auteure !