Théorie de l'information
&
Codes correcteurs d'erreurs

Eric Fabre
Chargé de Recherches INRIA

1er mars 2000

Contents

Chapter 1

Les concepts de la théorie de l'information

Ce chapitre décrit les objectifs de la théorie de l'information et ses fondements historiques. Après un aperçu de l'organisation du cours, une suite d'exemples intuitifs permet de cerner progressivement la notion d'information. On définit ensuite les notions de source d'information et d'entropie d'une source. On finit par des rappels de théorie des probabilités.

1.1 Introduction

La théorie de l'information (TI) ne s'est érigée en discipline autonome que très progressivement, poussée par la force des concepts qu'elle a dégagés. À l'origine, les préoccupations de son fondateur, Claude Shannon[1], étaient essentiellement pratiques. L'idée de communications numériques, à partir de l'échantillonnage suivi de la quantification d'un signal analogique, était dans l'air depuis quelques temps. En partie pour lutter contre le bruit des transmissions analogiques (et peut-être pour faciliter le cryptage des messages à transmettre...) Il s'agissait alors de savoir combien de bits du signal numérisé étaient réellement utiles à la compréhension du message par le récepteur. La réponse à cette question était d'autant plus délicate que les canaux de transmission (hertziens par exemple) étant bruités, une partie de cette information serait altérée avant d'atteindre son récepteur...

[1] Claude Shannon, encore vivant aujourd'hui(1999), est mondialement reconnu comme le "père" de la théorie de l'information. Un concensus très large fixe l'avènement de cette discipline à 1948, date de publication de l'article "A mathematical theory of communication". "Communication in the presence of noise", publié l'année d'après, a rendu son auteur mondialement célèbre.

1.1.1 Objectifs de la théorie de l'information

Schématiquement, vu d'aujourd'hui, on pourrait classer ces problèmes en trois catégories. À chacun d'entre eux Claude Shannon aura apporté une contribution fondamentale.

Mesurer l'information

Cela pose en fait deux questions : 1/ définir d'abord la notion d'information, et 2/ concevoir un moyen de la quantifier. La première question est délicate car elle semble faire appel à beaucoup de suggestivité. Le plus simple est de définir une *source d'information*, qui ne sera rien d'autre qu'une variable aléatoire. Pour mesurer l'information apportée par une source, on utilisera la notion d'*entropie* d'une source, concept central en TI. Cette quantité donne lieu à beaucoup d'interprétations. On verra que l'entropie a une unité : le *bit*.

Comprimer l'information

Une signal digital, une image numérique, un fichier informatique de texte, etc. peuvent être extrêmements redondants. Chacun a utilisé un jour des logiciels de "compression" qui permettent de récupérer de la place sur un disque dur. Plus généralement, toutes les applications qui ont des *besoins de stockage* (bases de données, CD audio, archivage INA des émissions radio et télé, boîtes noires d'avions, etc.) ou des *débits de transmission limités* (télévision satellite ou par câble, radio numérique, téléphonie cellulaire ou filaire, etc.) se posent la question de supprimer au mieux la redondance.

Bien entendu, le procédé n'a d'intérêt que s'il est réversible... Du moins faut-il savoir à partir de quand la compression se fait avec pertes. Noter que la compression avec pertes est utile. Tout un champ de recherche, notamment pour le son et l'image fixe ou animée, se préoccupe de méthodes de compression dont les pertes restent peu sensibles à l'auditeur / au spectateur. Pour le son, on peut citer le téléphone, qui n'a pas la qualité du CD mais reste compréhensible, ou encore le minidisc, qui utilise l'effet de masquage acoustique d'un son par un autre. En vidéo, citons les normes MPEG, le codage visiophone, qui rafraîchit plus souvent l'image des lèvres que celle du reste du visage, etc.

On verra que la limite de compression du signal issu d'une source d'information est justement... l'entropie de cette source. On verra aussi les algorithmes de compression standard. Il s'agit bien sûr d'algorithmes travaillant "en aveugle", i.e. sans connaissance sur le contenu sémantique du train de bit à comprimer.

Transmettre l'information

Comme on l'a dit plus haut, c'est l'un des problèmes fondateurs de la TI.Il peut se formuler ainsi : on dispose d'un message formé d'une suite de symboles puisés

dans un alphabet $\mathcal{A}$. Par exemple $\mathcal{A} = \{0, 1\}$ ou $\mathcal{A} = \{a, b, c, \ldots\}$. Ce message doit être transmis à travers un canal *bruité*, qui peut altérer aléatoirement les symboles (fig. 1.1). Par exemple, `zolicatu` est émis et `tolirotu` est reçu. Les propriétés statistiques du bruit sont connues. Pour lutter contre ces altérations, l'idée est de *rajouter de la redondance* dans le signal : c'est ce que l'on appelle le *codage* du message. C'est en quelque sorte l'inverse de la compression... Par exemple, on peut décider de répéter 3 fois le signal à l'émission. Ainsi, si l'on reçoit `tolirotu, zalicaru, zoticaru`, par une règle majoritaire sur chaque lettre, on retrouve bien `zolicatu`. Bien entendu, une décision majoritaire est plus facile à mettre en oeuvre sur des bits que sur des lettres...

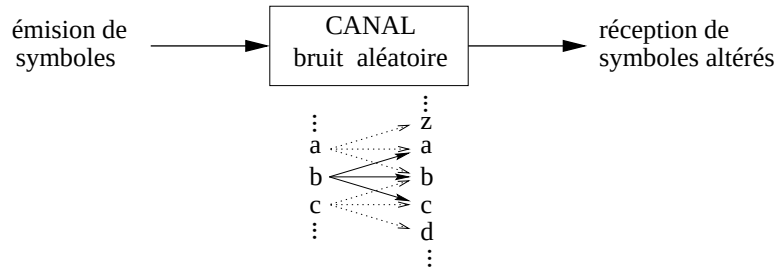

Figure 1.1: Canal de transmission. Chaque lettre émise peut être transformée en une autre lettre par le bruit du canal. La loi de probabilité du bruit est connue. Par exemple, un canal "à bruit additif" dit que la lettre reçue est voisine de la lettre émise (voisinage dans l'alphabet). La probabilité de l'altération peut décroître avec l'importance du décalage.

En répétant le signal 3 fois, on a construit un codage de l'information dont le *rendement* est 1/3. En effet, il faut émettre 3 symboles pour chaque *symbole utile* du message. Cette perte de rendement est le prix à payer pour lutter contre le bruit. Avant 1948 et les travaux de Shannon, le gens pensaient que l'on ne pouvait transmettre sans erreur sur un canal bruité qu'en augmentant infiniment la redondance du signal. C'est à dire en faisant tendre le rendement vers zéro (fig 1.2)... Shannon a montré que cette intuition était fausse. En fait, on peut transmettre sans erreur pour des rendements non nuls (fig 1.2)! Il existe une valeur limite du rendement, notée C, que l'on appelle la *capacité du canal*. Si l'on veut émettre avec un rendement R supérieur à C, on est sûr de faire des erreurs : l'effet du bruit ne pourra pas être corrigé. En revanche, pour $R < C$, il est possible de construire un codage de rendement R et corrigeant toutes les erreurs. Bien entendu, après une courte période d'incrédulité, ce résultat surprenant a rendu célèbre son auteur.

Le résultat de Shannon est un peu plus fin en réalité. Il s'agit d'un résultat asymptotique : Shannon propose une suite de codes de rendement $R < C$. Ces codes sont de plus en plus longs, c'est à dire prennent des mots de k symboles en entrée et construisent des mots de $n = k/R$ symboles à émettre. Et lorsque k tend

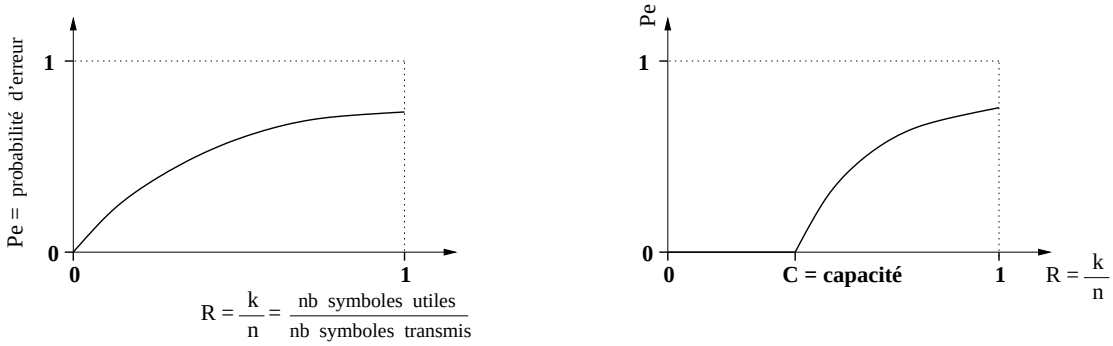

Figure 1.2: Avant Shannon (à gauche) on croyait que la probabilité d'erreur d'une transmission ne tendait vers 0 que si le rendement tendait vers 0, c'est à dire qu'il fallait répéter de nombreuses fois un symbole pour être sûr de sa transmission correcte. Shannon a montré (à droite) que cette répétition pouvait être limitée : on peut transmettre sans erreur pour des rendements non nuls, pour autant que ceux-ci sont inférieurs à la *capacité du canal*.

vers l'infini, alors la probabilité d'erreur tend vers zéro. Comme on le verra, ces codes sont inutilisables en pratique car ils n'ont pas de structure interne : le codage d'une suite de k symboles en un mot de n symboles à transmettre se fait à l'aide d'une table, et de même pour le décodage. Lorsque k augmente, la taille du dictionnaire explose, ce qui les rend inutilisables... Il faut recourir à d'autres techniques pour émettre avec peu d'erreurs, même pour $R < C$. C'est tout le travail de conception des *codes correcteurs d'erreurs*, qui doivent offir suffisamment de structures pour permettre codage et décodage à l'aide d'algorithmes, non plus de tables.

1.1.2 Organisation du cours

Le cours se divise en trois grandes parties. Les deux premières traitent des concepts fondamentaux de la théorie de l'information : entropie et capacité de canal, et reposent essentiellement sur la théorie des probabilités. La troisième s'intéresse aux techniques de construction de codes correcteurs d'erreurs. Elle repose sur des notions usuelles d'algèbre (espaces vectoriels, division de polynômes, primalité...) particularisées au cas des corps finis.

Introduction

Elle correspond au premier chapitre et cherche à donner une compréhension intuitive de la notion d'information, et à présenter les objectifs de la TI. On y rappelle quelques bases de probabilités.

Entropie et compression

Ces chapitres s'attachent à mesurer la quantité d'information apportée par une *source d'information*, quantité que l'on appellera l'*entropie* de cette source. Diverses interprétations de cette quantité sont données. On introduit aussi la notion d'*information mutuelle* : l'information apportée par une source sur une autre. Les propriétés de ces quantités sont largement explorées, et on développe des règles de "calcul entropique". On s'intéresse ensuite au *codage de source*, c'est à dire à la transmission[2] la plus économique (en nombre de bits utilisés) de l'information produite par une source. Cela peut se voir comme un problème de *compression* de données. Le résultat important de cette section est le *premier théorème de Shannon*, établissant une limite à la compression sans perte. On verra en particulier un algorithme classique de compression : l'algorithme de Lempel-Ziv.

Capacité de canal et transmission

Cette partie se préoccupe de transmission d'information sur un canal bruité. Plusieurs types de canaux de transmission y sont décrits, et l'on s'intéresse en particulier à la *capacité* d'un canal. Comme pour l'entropie, les propriétés et règles de calcul sont explorées, par exemple lorsque l'on combine plusieurs canaux (en série, en parallèle). Le résultat important est ici le *second théorème de Shannon*, qui démontre que l'on peut transmette sans erreur sur un canal bruité, pourvu que le rendement du code utilisé ne dépasse pas la capacité du canal.

Codes correcteurs et algorithmes de décodage

Ces derniers chapitres complètent le second théorème de Shannon, qui ne donne pas de technique de codage utilisable pour des transmissions fiables. On introduit les concepts de *code détecteur et correcteur d'erreur*. En particulier, on s'intéresse à des codes dits "algébriques", c'est à dire ayant des propriétés structurelles fortes qui facilitent le codage et le décodage. Ces codes sont au minimum des *codes linéaires*, pour lesquels on verra les notions de matrice génératrice, de matrice de contrôle, de syndrôme... (Quelques rappels d'algèbre sur les corps finis seront faits.) Parmi ces codes linéaires, on explorera quelques sous-classes très largement utilisées : les *codes cycliques* et les *codes convolutifs*. Les algorithmes de décodage associés seront détaillés ; on s'intéressera en particulier aux notions de *décodage soft*, de *décodage hard*, de *décodage au maximum de vraisemblance*. Quelques applications seront enfin détaillées (le compact disc), avec de nouvelles perspectives sur les techniques de codage actuelles, en particulier les *turbo-codes*, codes les plus performants à ce jour.

[2] on fait ici l'hypothèse d'un canal parfait

1.2 La notion d'information

1.2.1 Laissons parler l'intuition...

Les personnes qui parlent le plus d'information sont... les journalistes! Et defait, en observant la presse, quelques propriétés de l'information peuvent être dégagées. Imaginons par exemple que deux pays entrent soudainement en guerre. Les journaux en font la une le jour même et augmentent leur tirage. Une semaine plus tard, on en parle moins, et un mois après, il n'en est plus fait mention (même si la guerre continue). Si d'improviste un armistice est signé, la presse s'y intéresse à nouveau, avant de s'orienter vers d'autres événements. Cela nous apprend tout d'abord que *l'information est attachée à un événement.* Si l'on se donne comme mesure de la quantité d'information le nombre de colonnes à la une, on voit aussi qu'*un événement contient d'autant plus d'information qu'il est imprévisible.* Enfin, troisième leçon, *la quantité d'information dépend du degré d'imprévisibilité de l'événement, pas de sa nature.*

On peut corroborer ces observations en considéranr une information toujours donnée dans les journaux, dans une quantité constante : la météo. C'est qu'en effet le temps de demain est toujours aussi imprésisible... À y regarder de plus près, on notera toutefois des variations à la place accordée, par exemple en plein été (temps stable) ou lors d'événements exceptionnels (sécheresses, tempêtes,...).

Troisième exemple tiré de la presse : le tiercé. Imaginons qu'un turfiste veuille stocker sur son ordinateur les résultats du tiercé, sur les dix dernières années, pour des courses de 16 chevaux. Combien de bits seraient nécessaires à ce stockage ? Une solution simple consiste à coder sur 4 bits le numéro de chaque cheval. Un résultat de tiercé occupe alors $3\times4=12$ bits. Si l'on apprend maintenant que neuf fois sur dix le tiercé gagnant est "1,2,3", on peut imaginer un autre "codage" plus économique. On enregistre le bit "1" pour coder ce résultat fréquent, et pour les jours où le tiercé est différent, on enregistre "0" suivi des 12 bits donnant les numéros des chevaux gagnants. Ce petit exemple renforce les observations précédentes : la quantité d'information (en nombre de bits) portée par un événement (= un résultat de tiercé) dépend du degré d'incertitude de cet événement. Plus l'événement est improbable, plus il porte d'information.

1.2.2 Sources d'information, information moyenne

Pour formaliser un peu les choses, on voit qu'il y a de l'information là où il y a de l'aléatoire. Cela nous conduit à poser la

Définition 1 *Une* source d'information *est une variable aléatoire*[3]*, de loi connue.*

[3]La section 1.3 rappelle en détail les définitions de base des probabilités.

Notons X cette variable, définie sur l'espace probabilisé Ω, et à valeurs dans $\mathcal{X}$

$$X \ : \ \Omega \longrightarrow \mathcal{X}$$

Chaque valeur $x \in \mathcal{X}$ de X a pour probabilité (ou vraisemblance) $I\!P_X(x)$, et représente un événement tel qu'on l'a vu plus haut : guerre, paix, temps de demain, résultat du tiercé... La source d'information produit ainsi un événement (donne une valeur) chaque fois qu'on l'interroge.

Appelons ϕ la fonction mesurant l'information portée par chaque événément x. On a vu que cette mesure ne dépend que de la vraisemblance de l'événement, pas de sa nature. On a donc $\phi(x) = \Phi(I\!P_X(x))$. On sait en outre que plus l'événement est imprévisible, plus il porte d'information. Donc la fonction Φ est décroissante. Sans connaître Φ, on peut définir la quantité d'information moyenne apportée par une source comme

$$H(X) = \sum_{x \in \mathcal{X}} \Phi(I\!P_X(x)) \cdot I\!P_X(x)$$

que l'on appellera *entropie de la source* X. Reste à trouver Φ...

1.2.3 Mesurer l'information

Le "jeu des 20 questions" va nous permettre de deviner la fonction Φ. Prenons comme espace probabilisé Ω l'ensemble des huit cartes suivantes : $1\heartsuit, 2\heartsuit, 3\heartsuit, 4\heartsuit$, $3\spadesuit, 4\spadesuit, 4\diamondsuit, 5\clubsuit$. Cet ensemble est muni de la loi uniforme (vraisemblance 1/8 pour chaque carte). La règle du jeu est alors la suivante : une carte est tirée au hasard dans Ω - c'est une source d'information - et il s'agit de deviner la *couleur* de cette carte, i.e. $\heartsuit, \spadesuit, \diamondsuit$ ou $\clubsuit$. Pour cela on pose des questions à réponse *oui/non*. Il s'agit bien entendu de trouver la bonne couleur avec un nombre minimal de questions.

Stratégie 1. On passe toutes les cartes en revue : "Est-ce l'as de coeur ?", si oui, la couleur est coeur, sinon "Est-ce le 2 de coeur?", etc. Il est facile de vérifier que cette stratégie permet de conclure avec une moyenne de $4,375 = \frac{35}{8}$ questions (**exercice**).

Stratégie 2. On interroge sur la couleur uniquement. Ici, l'ordre des questions est important : on a intérêt à demander tout de suite si c'est du coeur, puisque c'est la couleur la plus fréquente. Ainsi, 4 fois sur 8 il suffit d'une question pour conclure. Si ce n'est pas coeur, demander si c'est du pique : on conclut alors en 2 questions, 2 fois sur 8. Pour les cas restants, une troisième question est nécessaire. Le nombre de questions moyen est donc de 1,75 (**exercice**). On peut démontrer que c'est l'optimum.

Formalisation. La source d'information est ici la variable aléatoire X, définie sur Ω, à valeurs dans $\{\heartsuit, \spadesuit, \diamondsuit, \clubsuit\}$ avec la loi $P_X = (\frac{4}{8}, \frac{2}{8}, \frac{1}{8}, \frac{1}{8})$. Pour *transmettre* la valeur de X, on peut imaginer un code fondé sur la stratégie 2 : on envoie 1 si la

réponse à la question est *oui* et 0 sinon. Cela donne le code

x	code	$\phi(x)$	$\mathbb{P}_X(x)$	$-\log_2(\mathbb{P}_X(x))$
♡	1	1	$4/8 = 2^{-1}$	1
♠	01	2	$2/8 = 2^{-2}$	2
♢	001	3	$1/8 = 2^{-3}$	3
♣	000	3	$1/8 = 2^{-3}$	3

Définissons l'information portée par une valeur x comme le nombre de bits nécessaires pour la transmettre, ou de façon équivalente comme le nombre de questions à poser à la source pour deviner la valeur tirée. On retrouve bien qu'il faut d'autant plus de questions que la vraisemblance de x est faible. Plus précisément, $\phi(x) = \Phi(\mathbb{P}_X(x)) = -\log_2(\mathbb{P}_X(x))$ où $\log_2$ représente le log en base 2 ($\log_2(y) = \ln(y)/\ln(2)$). Il vient

Définition 2 *L'*entropie *de la variable aléatoire X de loi $\mathbb{P}_X$ est définie par*

$$H(X) = -\sum_{x\in\mathcal{X}} \mathbb{P}_X(x)\cdot\log_2(\mathbb{P}_X(x))$$

C'est le nombre moyen de bits nécessaires à la transmission de X, ou encore le nombre moyen (minimal) de questions nécessaires pour deviner X.

1.3 Rappels de probabilités

1.3.1 Variable aléatoire

Espace probabilisé

Un *espace probabilisé* est composé d'un ensemble Ω muni d'une loi de probabilité $\mathbb{P}$. Nous ne considérerons que des lois de probabilité sur des ensembles discrets dans ce cours. Ω peut donc se voir comme $\{\omega_1, \ldots, \omega_m\}$, où ω_i est un *événement atomique*. Un *événement* est un sous-ensemble de Ω. Il est souvent décrit par un prédicat ; par exemple, si une fonction *couleur* est définie sur Ω,

$$couleur \;:\; \Omega \longrightarrow \{blanc, bleu, vert, \ldots\}$$

l'événement *blanc* représentera le sous-ensemble des ω_i pour lesquels *couleur* vaut *blanc*, ce qui se note encore $couleur^{-1}(blanc)$. La loi $\mathbb{P}$ est une fonction de Ω vers $[0,1]$, telle que

$$\sum_{i=1}^{n} \mathbb{P}(\omega_i) = 1$$

On écrira souvent p_i au lieu de $\mathbb{P}(\omega_i)$. $\mathbb{P}$ est étendue par additivité à $\mathcal{P}(\Omega)$, l'ensemble des parties de Ω, c'est à dire l'ensemble des événements. La probabilité d'un événement $A \subset \Omega$ est donc

$$\mathbb{P}(A) = \sum_{\omega_i \in A} \mathbb{P}(\omega_i)$$

Il en découle $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B)$ lorsque $A \cap B = \emptyset$, et $\mathbb{P}(\emptyset) = 0$.

Variable aléatoire

Une *variable aléatoire* X est une fonction d'un espace probabilisé vers un autre espace :

$$X : \Omega \to \mathcal{X}$$

On appelle une valeur $x \in \mathcal{X}$ de la variable X une *réalisation* de la variable X. La loi de probabilité de Ω est transportée sur $\mathcal{X}$ par la fonction X : on parle de *loi image*

$$\forall x \in \mathcal{X}, \quad \mathbb{P}_X(x) \triangleq \mathbb{P}\left(X^{-1}(x)\right)$$

Il est facile de vérifier que c'est bien une loi de probabilité :

$$\sum_{x \in \mathcal{X}} \mathbb{P}_X(x) = 1$$

Un cas particulier consiste à prendre la fonction sous forme d'un couple de fonctions :

$$(X, Y) : \Omega \to \mathcal{X} \times \mathcal{Y}$$

La probabilité image se construit de la même façon :

$$\begin{array}{rcl} \forall (x,y) \in \mathcal{X} \times \mathcal{Y}, \qquad \mathbb{P}_{X,Y}(x,y) & \triangleq & \mathbb{P}\left((X,Y)^{-1}(x,y)\right) \\ & = & \mathbb{P}\{\omega \in \Omega : X(\omega) = x, Y(\omega) = y\} \end{array}$$

$\mathbb{P}_{X,Y}$ est appelée *loi jointe des variables X et Y*. Il en découle la notion de *loi marginale* : "marginaliser" signifie extraire d'une loi jointe la loi d'une variable particulière. Par exemple pour retrouver $\mathbb{P}_X$ à partir de $\mathbb{P}_{X,Y}$:

$$\begin{array}{rcl} \mathbb{P}_X(x) & = & \mathbb{P}\{\omega : X(\omega) = x\} \\ & = & \mathbb{P}\{\omega : X(\omega) = x, Y(\omega) \text{ quelconque}\} \\ & = & \mathbb{P}\left(\bigcup_y \{\omega : X(\omega) = x, Y(\omega) = y\}\right) \\ & = & \sum_y \mathbb{P}\{\omega : X(\omega) = x, Y(\omega) = y\} \\ & = & \sum_y \mathbb{P}_{X,Y}(x,y) \end{array}$$

La loi de la variable X dans le couple (X, Y) s'obtient donc en "intégrant $\mathbb{P}_{X,Y}$ en Y" : on somme sur la variable que l'on veut éliminer. Si l'on représente $\mathbb{P}_{X,Y}$

dans un tableau à deux entrées (fig. 1.3), calculer la loi de X (indice de colonne) revient à sommer $\mathbb{P}$ dans chaque colonne. Cela donne une loi de probabilité sous forme de ligne, que l'on écrit dans la marge de la matrice, d'où le nom de marginale. Bien entendu, loi jointe et technique de marginalisation s'étendent à des n-uplets de variables aléatoires.

Espérance

Lorsque la variable X est à valeurs dans $\mathbb{R}$, on peut calculer sa *moyenne*, ou son *espérance* :

$$\mathbb{E}(X) = m_X = \sum_x x \cdot \mathbb{P}_X(x)$$

De même pour toute variable aléatoire définie à partir de X, par exemple, si $Y = \phi(X)$, on a

$$\mathbb{E}(Y) = \mathbb{E}(\phi(X)) = \sum_x \phi(x)\mathbb{P}_X(x) = \sum_y y\mathbb{P}_Y(y)$$

Le second moment de X, c'est à dire la moyenne de X^2, permet de définir la *variance* de X, qui caractérise l'excursion de X autour de sa moyenne

$$V(X) = \sigma_X^2 = \mathbb{E}(X^2) - \mathbb{E}(X)^2$$

σ_X, appelé *écart-type* de X, est la racine carrée de la variance de X. On observera qu'une variable aléatoire de variance nulle est constante et égale à sa moyenne.

Indépendance

Il s'agit là d'une notion très importante pour la théorie de l'information. Intuitivement, l'*indépendance* de deux variables aléatoires signifie que leurs comportements sont autonomes, c'est à dire qu'il n'existe pas de lien statistique entre elles, ou encore que *la connaissance de l'une de ces variables n'apporte aucune information sur l'autre.* On dit que les variables X et Y sont *indépendantes* ssi leur loi jointe vérifie :

$$\forall (x,y) \in \mathcal{X} \times \mathcal{Y}, \quad \mathbb{P}_{X,Y}(x,y) = \mathbb{P}_X(x)\mathbb{P}_Y(y) \tag{1.1}$$

L'indépendance se généralise au cas de n variables[4].

Exemple. (fig. 1.3) Considérons un jeu de 54 cartes, sans les jokers ; l'espace Ω est donc l'ensemble des 52 cartes restantes, et est muni de la loi uniforme (chaque carte est équiprobable de probabilité 1/52). On construit les deux variables aléatoires X et Y qui à chaque carte associent respectivement sa valeur (1,2,...,dame,roi) et

[4]Attention, $X_1 \ldots, X_n$ sont indépendantes implique bien entendu que les X_i sont indépendantes deux à deux, mais la réciproque est fausse. Construire un contre-exemple avec trois variables (**exercice**).

sa couleur (pique, coeur, carreau, trèfle). On vérifie que X et Y sont distribuées uniformément : $I\!P_X(x) = 1/13$ et $I\!P_Y(y) = 1/4$ pour tout x et tout y. Comme pour toute paire (x, y) on a $I\!P_{X,Y}(x, y) = 1/52 = I\!P_X(x)I\!P_Y(y)$, les deux variables X et Y sont bien indépendantes. Et en effet, connaître la couleur d'une carte ne nous apporte aucune connaissance sur sa valeur.

Contre-exemple. Dans un jeu de tarot (78 cartes), on construit les variables X = valeur, Y = couleur (l'atout étant la cinquième couleur). Vérifiez que X et Y ne sont plus indépendantes en exhibant un couple (x, y) pour lequel l'égalité ne tient pas.

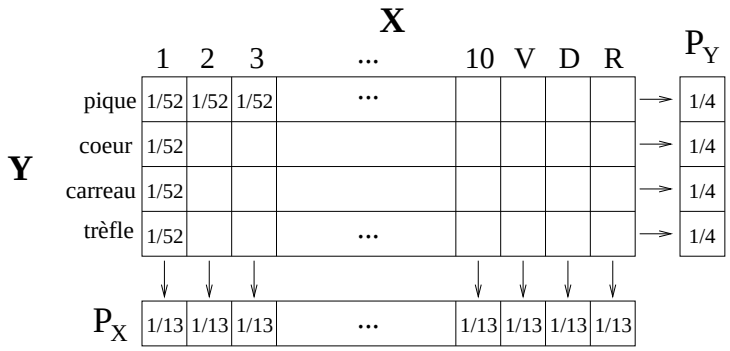

Figure 1.3: Jeu de 52 cartes. Représentation de la marginalisation d'une loi jointe. On peut ainsi vérifier que X et Y sont indépendantes.

L'indépendance entraîne la propriété suivante, facile à vérifier (**exercice**):

$$I\!E\left[\phi(X)\psi(Y)\right] \;=\; I\!E\left[\phi(X)\right]I\!E\left[\psi(Y)\right] \tag{1.2}$$

On peut aussi démontrer facilement que si elle est vraie pour tout choix des fonctions ϕ et ψ, alors X et Y sont indépendantes (**exercice**). Noter en revanche que la relation suivante, elle, est toujours vraie, avec ou sans independance :

$$I\!E\left[\phi(X) + \psi(Y)\right] \;=\; I\!E\left[\phi(X)\right] + I\!E\left[\psi(Y)\right] \tag{1.3}$$

L'indépendance peut se réinterpréter à l'aide de la notion de loi conditionnelle.

1.3.2 Loi conditionnelle

Loi conditionnelle

La *loi conditionnelle de X sachant Z* est définie par

$$\forall (x, z) \in \mathcal{X} \times \mathcal{Z}, \quad I\!P_{X|Z}(x|z) \;\triangleq\; \frac{I\!P_{X,Z}(x, z)}{I\!P_Z(z)} \tag{1.4}$$

Il s'agit d'une loi de probabilité sur $\mathcal{X}$, et cette loi est *paramétrée* par Z : on a une loi sur $\mathcal{X}$ pour toute valeur z. Cette loi conditionnelle n'est définie que pour les z de probabilité non nulle. Intuitivement, la loi conditionnelle de X sachant $Z = z$ représente "l'aléatoire restant sur X lorsque l'on a observé que la variable Z portait la valeur z".

Conséquence immédiate des définitions (1.1) et (1.4), X et Y sont indépendantes ssi $\mathbb{P}_{X|Y=y} \equiv \mathbb{P}_X$, $\forall y$, ou de façon symétrique $\mathbb{P}_{Y|X=x} \equiv \mathbb{P}_Y$, $\forall x$. Il ne sert donc à rien d'observer l'une des variables pour tenter de prédire la seconde (voir le paragraphe sur l'estimation).

Exemple. Reprenons le jeu de 52 cartes, auquel on enlève valet, dame et roi de pique. Quelle est la loi conditionnelle de X sachant $Y = \mathit{pique}$? On a $\mathbb{P}_{X,Y}(x, \mathit{pique}) = 1/49$ pour $x \in \{1, \ldots, 10\}$ et 0 sinon. On a encore $\mathbb{P}_Y(\mathit{pique}) = 10/49$. Il vient $\mathbb{P}_{X|Y}(x|\mathit{pique}) = 1/10$ si $x \in \{1, \ldots, 10\}$ et 0 sinon. C'est bien une loi de probabilité sur $\mathcal{X}$.

Interprétation : pour construire $\mathbb{P}_{X|Y=\mathit{pique}}$, on restreint Ω au sous-ensemble des ω vérifiant $Y = \mathit{pique}$, et on renormalise la loi $\mathbb{P}$ de façon à ce que sa somme fasse 1. On a restreint la loi uniforme à un ensemble de 10 cartes, d'où la valeur 1/10. On peut vérifier que $\mathbb{P}_{X|Y}(x|\mathit{coeur}) = 1/13$ pour toute valeur de x.

Exercice. En général, la loi conditionnelle de X sachant $Z = z$ change avec la valeur z prise par Z. Montrer que si cette loi ne change pas, c'est qu'il y a indépendance de X et de Z.

Espérance conditionnelle

On peut calculer la moyenne de X sous la loi conditionnelle $\mathbb{P}_{X|Z=z}$. Cette moyenne dépend donc de la valeur z prise par Z. Elle est définie par :

$$\mathbb{E}\,(X|Z = z) = \sum_x x \cdot \mathbb{P}_{X|Z}(x|z)$$

Bien entendu, si X et Z sont indépendantes, alors $\mathbb{E}\,(X|Z = z)$ ne dépend pas de z et vaut $\mathbb{E}\,(X)$.

Généralement, $\mathbb{E}\,(X|Z = z)$ dépend de la valeur observée pour Z. On peut donc voir cette espérance conditionnelle comme une variable aléatoire[5] $\phi(Z)$, une fonction de Z. Rien n'empêche donc de calculer sa moyenne. Il vient immédiatement

$$\begin{array}{rcl} \mathbb{E}\,[\mathbb{E}\,(X|Z)] & = & \sum_z \mathbb{E}\,(X|Z = z)\mathbb{P}_Z(z) \\ & = & \sum_z \sum_x x\mathbb{P}_{X|Z}(x|z)\mathbb{P}_Z(z) \\ & = & \sum_{x,z} x\mathbb{P}_{X,Z}(x, z) \\ & = & \sum_x x\mathbb{P}_X(x) \\ & = & \mathbb{E}\,(X) \end{array}$$

[5]Attention, parler de la variable aléatoire "$X|Z$" n'a pas de sens. Un tel objet n'existe pas. Le conditionnement "sachant Z" s'applique à l'*espérance* de X, ou à la *loi* de X.

Exemple. Dans le contexte de l'exemple ci-dessus, $\mathbb{E}(X|Y = pique) = 5.5$ et $\mathbb{E}(X|Y = coeur) = 7$ avec la convention $V = 11, D = 12, R = 13$. Vérifier que $\mathbb{E}[\mathbb{E}(X|Y)] = \mathbb{E}(X)$.

Indépendance conditionnelle

Les variables X et Y sont *conditionnellement indépendantes sachant* Z ssi la loi conditionnelle du couple (X, Y) s'écrit sous forme produit, comme dans le cas de l'indépendance :

$$\forall (x, y, z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}, \quad \mathbb{P}_{X,Y|Z}(x, y|z) = \mathbb{P}_{X|Z}(x|z)\mathbb{P}_{Y|Z}(y|z) \tag{1.5}$$

On retrouve ainsi toutes les interprétations et propriétés de l'indépendance, mais celle-ci n'est valable que lorsque l'on a observé une valeur de Z, c'est à dire dans chaque sous-ensemble $Z^{-1}(z)$ de Ω. On a par exemple, en reprenant (**??**)

$$\mathbb{E}(XY|Z = z) = \mathbb{E}(X|Z = z)\mathbb{E}(Y|Z = z)$$

pour toute valeur z, etc.

Exemple. Reprenons le cas du jeu de tarot, avec X=valeur de la carte, Y=couleur, $Z \in \{atout, pas - atout\}$. On a mentionné que X et Y ne sont pas indépendantes. Toutefois, elles sont indépendantes conditionnellement à Z. En effet, sachant $Z = pas - atout$, on est ramené à un jeu de 56 cartes (voir plus haut), pour lequel valeur et couleur sont indépendantes, et sachant $Z = atout$, Y est figé à la couleur *atout*. On retrouve dans ce cas aussi la forme produit (**exercice**).

Estimation au maximum de vraisemblance

Lorsque l'on ne peut observer directement la valeur prise par une variable aléatoire X, on peut chercher à *estimer* cette valeur. Un estimateur immédiat consiste à prendre la valeur la plus vraisemblable : c'est l'*estimateur au maximum de vraisemblance*, défini par

$$\hat{X} = \arg\max_{x \in \mathcal{X}} \mathbb{P}_X(x)$$

Cet estimateur a la propriété de minimiser la probabilité d'erreur, c'est à dire la probabilité de l'événement $X \neq \hat{X}$ (**exercice**). Dans de nombreuses situations, X n'est pas observée directement, mais on a accès à la valeur d'une autre variable, Y, liée à X. Cela donne un estimateur[6] de X "sachant Y" :

$$\hat{X}_{|Y=y} = \arg\max_{x \in \mathcal{X}} \mathbb{P}_{X|Y}(x|Y = y)$$

[6] appelé souvent estimateur au maximum de vraisemblance *a posteriori* de X sachant Y

qui varie avec la valeur y prise par Y ($\hat{X}_{|Y}$ est donc une variable aléatoire de la forme $\phi(Y)$).

Exercice. Montrer que si X et Y sont indépendantes, alors les deux estimateurs ci-dessus coïncident, c'est à dire que Y n'aide pas à estimer X.

Exercice. Montrer au contraire que si elles sont liées, la probabilité d'erreur est plus faible pour le second estimateur, i.e. $\mathbb{P}(X \neq \hat{X}) \geq \mathbb{P}(X \neq \hat{X}_{|Y})$. Indication : observer que dans le terme de droite l'événement "erreur" est une fonction du couple (X, Y).

1.3.3 Loi des grands nombres

On considère maintenant une suite $(Y_n)_{n\in\mathbb{N}}$ de variables aléatoires à valeurs dans $\mathbb{R}$, indépendantes et identiquement distribuées (i.i.d.) de loi $\mathbb{P}_Y$. Chaque Y_n ayant la même loi P_Y a donc la même moyenne m_Y. L'indépendance permet d'écrire la probabilité de tout n-uplet :

$$\mathbb{P}(y_1, \ldots, y_n) = \mathbb{P}_Y(y_1) \cdots \mathbb{P}_Y(y_n)$$

On construit la suite $(Z_n)_{n\in\mathbb{N}}$ par

$$Z_n = \frac{1}{n} \sum_{i=1}^{n} Y_i$$

Z_n mesure donc la *moyenne empirique* des Y_i sur les n premiers éléments. La *loi faible des grands nombres* affirme que Z_n converge *en probabilité* vers la *moyenne statistique* m_Y de la suite, c'est à dire

$$\forall \epsilon > 0, \qquad \mathbb{P}\left(|Z_n - m_Y| > \epsilon\right) \to_n 0$$

La *loi forte des grands nombres* affirme que cette convergence est même *presque sûre*. En clair, soit

$$\mathcal{E} = \{(y_n) : z_n \not\to_n 0\}$$

l'ensemble des réalisations de la suite (Y_n) pour lesquelles la suite (Z_n) ne converge pas vers m_Y, alors la probabilité de cet ensemble est nulle

$$\mathbb{P}(\mathcal{E}) = 0$$

On n'a donc aucune chance de tomber sur une suite (y_n) pour laquelle la moyenne empirique ne converge pas vers la moyenne statistique.

On utilisera ce résultat sous une forme un peu plus spécialisée. Soit une suite i.i.d. (X_n) à valeurs dans $\mathcal{X}$. On construit $Y_i = \mathbb{1}(X_i \in A)$, où $A \subset \mathcal{X}$ et $\mathbb{1}$ est la

fonction indicatrice. Y_i vaut donc 1 si $X_i \in A$ et 0 sinon. (Y_n) est une suite i.i.d. à valeurs numériques, et

$$m_Y = \mathbb{E}(Y_i) = \mathbb{P}(X_i \in A) = \mathbb{P}_X(A)$$

Z_n compte donc la fréquence empirique de passage dans l'ensemble A pour la suite (X_n). La loi des grands nombres (faible ou forte) nous indique donc que Z_n converge vers la probabilité de l'ensemble A pour la loi $\mathbb{P}_X$. En d'autres termes, l'*histogramme*[7] converge vers la vraie loi de X.

[7] L'histogramme est la courbe obtenue en comptant le nombre moyen de fois où la suite (X_n) passe par chaque valeur de $\mathcal{X}$.

Chapter 2

Entropie - Information mutuelle

2.1 Entropie d'une variable aléatoire

2.1.1 Définition

Soit X une variable aléatoire discrète, définie sur $(\Omega, I\!P)$, à valeurs dans $\mathcal{X}$ et donc de loi $I\!P_X$, loi image de $I\!P$ par X. Dans la suite, s'il n'y a pas ambiguïté, on ne mentionnera pas la variable concernée dans $I\!P$. On écrira aussi $p(x_i)$ ou simplement p_i au lieu $I\!P_X(x_i)$, lorsque X peut prendre les valeurs $x_1, \ldots, x_n$.

Définition 3 *L'entropie de X est définie par*

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \cdot \log_2 p(x) = I\!E\left[-\log_2 p(X)\right]$$

On prend comme convention $0 \log_2 0 = 0$, de sorte que les événements de probabilité nulle ne sont pas comptabilisés dans la somme. $H(X)$ se mesure en *bits* lorsque le logarithme est à base 2. S'il est à base e, l'unité est le *nat* (pour logarithme *nat*urel), donc 1 bit = ln(2) nat.

2.1.2 Propriétés

1. $H(X)$ ne dépend que de la loi de X, pas des valeurs prises par X. Ainsi, si f est une bijection sur $\mathcal{X}$, $H(f(X)) = H(X)$.

2. $H(X)$ est une quantité positive, car $0 \leq p(x) \leq 1$ et donc $-\log_2 p(x) \geq 0$.

3. Pour la loi uniforme, $H(X) = \log_2 |\mathcal{X}|$, où $|\mathcal{X}|$ est le cardinal de $\mathcal{X}$.

4. $H(X) = 0$ ssi X est une variable déterministe, i.e. qui ne peut prendre qu'une seule valeur. Autrement dit ssi $I\!P_X$ est de la forme $(0, \ldots, 0, 1, 0, \ldots, 0)$ (**exercice**).

5. **Lemme 1** *Soit $g : \mathcal{X} \to \mathcal{Y}$, $H(g(X)) \leq H(X)$ avec égalité ssi g est injective (on suppose $p(x) > 0, \forall x$).*

   Ce résultat sera démontré plus loin.

   Exercice. Vérifer l'inégalité avec X = carte tirée au hasard dans un jeu de 32, et g la fonction *couleur*.

6. $H(X)$ mesure l'incertiture que l'on a sur X. Pour s'en convaincre, on peut oberver le cas d'une variable de Bernouilli : X peut prendre deux valeurs avec les probabilités p et $1-p$. La courbe $H(X)$ en fonction de p est donnée figure 2.1 L'entropie est maximale lorsqu'aucune des valeurs de X n'est plus

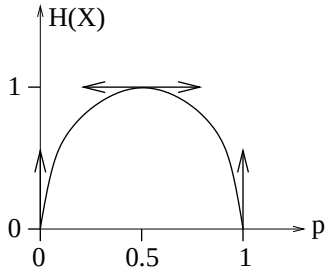

Figure 2.1: Entropie d'une variable de Bernouilli, en fonction du paramètre p.

   plausible que l'autre, $H(X)$ vaut alors $\log_2(2) = 1$ bit. Plus on peut "deviner" la valeur qui sortira, plus $H(X)$ décroît, pour finir à 0 lorsque l'une des deux valeurs est certaine.

7. **Lemme 2** *$H(X)$ est maximale lorsque X est muni de la loi uniforme.*

   En effet, notons $p_i = I\!P_X(x_i)$, et calculons

$$\max_{(p_1,p_2,\ldots)} \sum_i p_i \log_2 p_i \quad \text{sous la contrainte} \quad \sum_i p_i = 1$$

   Par la méthode du multiplicateur de Lagrange, on pose

$$J_\lambda(p_1, p_2, \ldots) = \sum_i p_i \log_2 p_i - \lambda\Big(\sum_i p_i - 1\Big)$$

   En maximisant par rapport à chaque p_i, il vient

$$\frac{\partial J_\lambda}{\partial p_i} = \frac{1}{\ln 2} + \log_2 p_i - \lambda = 0 \qquad \Longrightarrow \qquad p_i = 2^{\lambda - \frac{1}{\ln 2}}$$

   En ajustant λ pour satisfaire la contrainte $\sum_i p_i = 1$, on trouve bien la loi uniforme.

2.1.3 Interprétation

Au chapitre précédent, on a cherché à définir une mesure de l'information moyenne apportée par une source comme :

- le nombre de questions qu'il faut poser - en moyenne - pour deviner la valeur produite par la source, ou, de façon équivalente, comme
- le nombre moyen de bits qu'il faut émettre pour communiquer cette valeur.

Cela nous a permis de deviner la forme de la fonction entropie. On revient ici sur cette interprétation.

Considérons une variable X pouvant prendre 16 valeurs. Si on ne connaît pas la loi de X, il faut utiliser 4 bits ($2^4 = 16$) pour transmettre chaque valeur de X, et l'on ne peut pas faire mieux. Si l'on connaît la loi de X, on peut essayer d'améliorer le nombre moyen de bits transmis en attribuant moins de bits aux valeurs fréquentes. Supposons d'abord que X a la loi uniforme, par raison de symétrie toutes les valeurs seront décrites par le même nombre de bits, soit 4 (fig. 2.2). On retrouve bien $H(X) = \log_2 |\mathcal{X}| = \log_2 16 = 4$. Considérons maintenant Y, une variable aléatoire

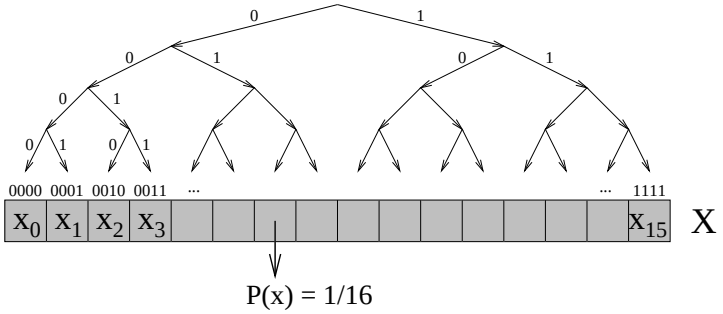

Figure 2.2: Construction d'un code de transmission, pour la variable aléatoire X munie de la loi uniforme. On peut utiliser un arbre dyadique dont chaque feuille pointe vers l'une des valeurs x. Chaque pavé gris correspond à une probabilité de 1/16. L'aire grise totale vaut donc 1.

construite par $Y = f(X)$ où f n'est pas injective. Elle rend donc indistingables certaines valeurs de X. La loi de probabilité de Y, loi image de $I\!P_X$ par f, est obtenue en sommant les vraisemblances des éléments x devenus indistingables après f (fig. 2.3). Toute loi de probabilité discrète peut se construire / se simuler sur ce mode, à partir d'une loi uniforme sur un ensemble assez grand de valeurs[1]. Tirer une valeur de Y équivaut à tirer une valeur de X et à prendre son image par f. Pour transmettre Y, on n'a plus besoin d'autant de bits que pour X puisque les x donnant le même y n'ont plus besoin d'être distingués, comme illustré figure 2.3. On constate alors que plus on regroupe de cases (= de valeurs x) pour décrire

[1] c'est d'ailleurs pour cela que les ordinateurs ne disposent que d'un simulateur de loi uniforme

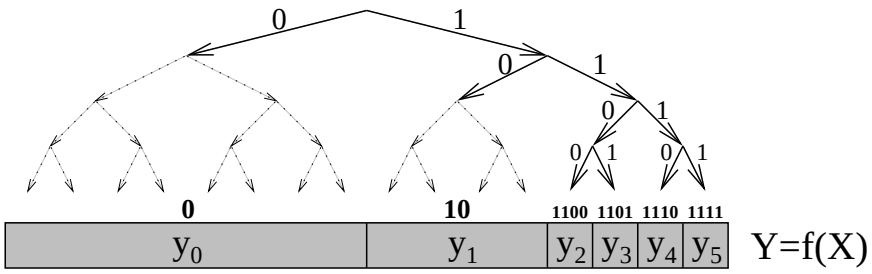

Figure 2.3: Construction d'un code de transmission, pour la variable aléatoire $Y = f(X)$. On peut utiliser l'arbre dyadique de X *tronqué* car certains bits décrivant X deviennent inutiles pour reconnaître Y.

la probabilité d'un y, plus on réduit le nombre de bits nécessaires à identifier le rectangle obtenu. Plus spécifiquement, sur l'exemple de la figure, qui représente une loi dyadique, on a clairement

$$\text{nombre de bits décrivant la valeur } y = -\log_2 p(y)$$

La quantité $-\log_2 p(y)$ est parfois appelée *quantité d'information portée par l'événement* y.

On reviendra plus loin sur ces schémas (à bien comprendre) pour illustrer l'inégalité de Kraft et la construction du code de Shannon (chapitre 3 sur le codage de source et la compression de données).

2.2 Entropie jointe et entropie conditionnelle

2.2.1 Entropie jointe

Considérons le couple (X, Y) comme une seule variable aléatoire sur Ω, à valeurs dans $\mathcal{X} \times \mathcal{Y}$. Son entropie est donnée par

$$\begin{aligned} H(X,Y) &= -\sum_{x\in\mathcal{X}, y\in\mathcal{Y}} p(x,y)\log_2 p(x,y) \\ &= \mathbb{E}\left[-\log_2 p(X,Y)\right] \end{aligned} \tag{2.1}$$

Cette formule se généralise au cas de n variables aléatoires $X_1, \ldots, X_n$. Noter que l'entropie jointe ne dépend pas de l'ordre des variables :

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

2.2.2 Entropie conditionnelle

Après avoir calculé l'entropie d'une loi jointe, on peut calculer l'entropie d'une loi conditionnelle :

Définition 4 *L'entropie conditionnelle de X sachant la valeur y de Y est l'entropie de la loi conditionnelle $p_{X|Y=y}$*

$$\begin{aligned} H(X|Y=y) &= -\sum_x p(x|y)\log_2 p(x|y) \qquad (2.2)\\ &= \mathbb{E}\left[-\log_2 p(X|Y)\,|\,Y=y\right] \end{aligned}$$

L'entropie conditionnelle de X sachant Y est la moyenne (en Y) de la quantité précédente :

$$\begin{aligned} H(X|Y) &= \sum_y H(X|Y=y)\cdot p(y) \qquad (2.3)\\ &= -\sum_{x,y} p(x,y)\log_2 p(x|y) \qquad (2.4)\\ &= \mathbb{E}\left[-\log_2 p(X|Y)\right] \end{aligned}$$

Les définitions ci-dessus demandent quelques commentaires. (2.2) mesure l'incertitude qui reste sur X lorsque l'on a observé *une* valeur particulière y de Y. C'est donc une variable aléatoire fonction de Y. Si on en fait la moyenne, comme dans (2.3), on obtient alors une mesure de l'incertitude moyenne qui reste sur X lorsque Y est accessible. Noter que (2.4) repose sur une somme double, comme (2.1) ; la différence est dans le terme moyenné : $p(X|Y)$ pour la première, $p(X,Y)$ pour la seconde. On a utilisé la règle de Bayes $p(x,y)=p(x|y)p(y)$ pour obtenir (2.4).

Exemple. Soit $\Omega=\{7\heartsuit,8\heartsuit,9\heartsuit,10\heartsuit,V\heartsuit,D\heartsuit,R\heartsuit,As\heartsuit,V\spadesuit,D\spadesuit,R\spadesuit,As\spadesuit\}$ muni de la loi uniforme, et X et Y les variables aléatoires donnant la couleur et la valeur de la carte. Le couple (X,Y) permet ainsi d'identifier la carte. On a

$$\begin{aligned} H(X,Y) &= \log_2|\Omega| = \log_2 12 = 2+\log_2 3\\ H(X) &= -\frac{2}{3}\log_2\frac{2}{3}-\frac{1}{3}\log_2\frac{1}{3} = -\frac{2}{3}+\log_2 3\\ H(Y) &= 4(-\frac{1}{12}\log_2\frac{1}{12})+4(-\frac{2}{12}\log_2\frac{2}{12}) = \frac{4}{3}+\log_2 3 \end{aligned}$$

Par ailleurs,

$$\begin{aligned} H(Y|X=\heartsuit) &= \log_2 8 = 3\\ H(Y|X=\spadesuit) &= \log_2 4 = 2\\ H(Y|X) &= 3\frac{2}{3}+2\frac{1}{3} = \frac{8}{3} \end{aligned}$$

et de même

$$\begin{aligned} H(X|Y=7) &= 0 \qquad\qquad \text{idem pour } 8,9,10\\ H(X|Y=As) &= \log_2 2 = 1 \qquad \text{idem pour } V,D,R\\ H(X|Y) &= (0\frac{1}{12})4+(1\frac{2}{12})4 = \frac{2}{3} \end{aligned}$$

2.2.3 Propriétés

Lemme 3 $H(X,Y) = H(X) + H(Y|X) = H(Y) + H(Y|X)$

Il suffit de montrer la première égalité, l'autre vient par symétrie.

$$\begin{aligned} H(X,Y) &= -\sum_{x,y} p(x,y) \log_2 p(x,y) \\ &= -\sum_{x,y} p(x,y) \log_2 p(x)p(y|x) \\ &= -\sum_{x,y} p(x,y) \log_2 p(x) - \sum_{x,y} p(x,y) \log_2 p(y|x) \\ &= -\sum_{x} p(x) \log_2 p(x) - \sum_{x,y} p(x,y) \log_2 p(y|x) \\ &= H(X) + H(Y|X) \end{aligned}$$

Conséquence : $H(Y|X) \neq H(X|Y)$ en général. On peut vérifier ces propriétés dans l'exemple ci-dessus.

Lemme 4 $H(X|Y) \geq 0$ *et s'annule ssi* $X = f(Y)$

En effet, $H(X|Y) = 0$ signifie que $H(X|Y = y) = 0$ pour toute valeur y telle que $p(y) \neq 0$. Cela signifie que la loi $p(X|Y = y)$ est singulière, c'est à dire porte sur une seule valeur x. Cette valeur x peut changer avec le y considéré. On définit ainsi la fonction $f : y \mapsto x$. Remarque : en toute rigueur f n'est pas définie pour les y tels que $p(y) = 0$, on a donc $X = f(Y)$ "presque sûrement" : l'ensemble des ω pour lesquels c'est faux est de probabilité nulle (**exercice**).

Ce lemme nous permet de démontrer le lemme 1 qui était resté en suspens. En effet

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

Or $H(g(X)|X) = 0$ et $H(X|g(X)) \geq 0$, donc on a bien $H(g(X)) \leq H(X)$. Il y a égalité ssi $H(X|g(X)) = 0$, c'est à dire (lemme 4) ssi $X = f(g(X))$, ce qui signifie que g est injective.

Lemme 5 $H(X|X) = 0$ *et donc* $H(X,X) = H(X)$

Découle des lemmes 3 et 4. En d'autres termes, répéter n'augmente pas la quantité d'information... Même si l'on répète d'une autre façon puisque $H(X, f(X)) = H(X)$.

Lemme 6 $H(X) \geq H(X|Y)$ *avec égalité ssi* X *et* Y *sont indépendantes.*

En d'autres termes, *le conditionnement réduit l'entropie*. Cela est logique car observer une variable Y liée à X nous apporte en général de l'information sur X, donc diminue notre incertitude sur X. Au pire, X et Y sont indépendantes et donc Y ne nous apprend rien sur X. La preuve sera donnée plus loin : on montrera en fait que $H(X)-H(X|Y) = H(Y)-H(Y|X)$ est une quantité positive, et nulle seulement en cas d'indépendance. Noter que l'implication "indépendance $\Rightarrow H(X) = H(X|Y)$" est facile à montrer (**exercice**), c'est la réciproque qui est difficile.

Lemme 7 *$H(X,Y) \leq H(X) + H(Y)$, avec égalité ssi X et Y indépendantes.*

Découle du lemme 6 (utiliser le lemme 3).

2.2.4 Généralisations - Formule des conditionnements successifs

Il est important de garder à l'esprit que, dans les lemmes ci-dessus, X et Y peuvent chacun représenter plusieurs variables. Ainsi le lemme 3 se généralise en

Lemme 8 (Formule des conditionnements successifs)

$$H(X_1,\ldots,X_n) = H(X_1) + H(X_2|X_1) + \ldots + H(X_n|X_1,\ldots,X_{n-1})$$

La preuve vient par récurrence. On prend $X = (X_1,\ldots,X_{n-1})$ et $Y = X_n$, et l'on applique $H(X,Y) = H(X) + H(Y|X)$. On recommence ensuite sur $H(X) = H(X_1,\ldots,X_{n-1})$. Noter que cette formule se lit de la façon suivante[2]

$$H(X_1,\ldots,X_n) = \overbrace{H(X_1) + \ldots + H(X_k|X_1^{k-1})}^{H(X_1,\ldots,X_k)} + \overbrace{H(X_{k+1}|X_1^k) + \ldots + H(X_n|X_1^{n-1})}^{H(X_{k+1},\ldots,X_n|X_1,\ldots,X_k)}$$

Exercice. Soit un jeu de 32 cartes, sur lequel on définit les trois variables $X =$ couleur de la carte ($\spadesuit, \heartsuit, \diamondsuit, \clubsuit$), $Y =$"figure" ou "carte basse", et $Z =$ rang dans sa catégorie (α, β, γ et δ pour 7 ou V, 8 ou D, 9 ou R, 10 ou As). Calculer $H(X,Y,Z)$ de deux façons.

La généralisation du lemme 6 est aussi très importante.

Lemme 9 *$H(X|Z) \geq H(X|Y,Z)$ avec égalité ssi X et Y sont indépendantes sachant Z.*

En d'autres termes, *plus on conditionne, plus on réduit l'entropie*. Là encore, la preuve en est remise à plus tard. On procédera comme pour le lemme 6, en montrant que $H(X|Z)-H(X|Y,Z) = H(Y|Z)-H(Y|X,Z)$ est une quantité positive, et nulle ssi X et Y sont indépendantes sachant Z. Noter que l'égalité ci-dessus est immédiate (**exercice**).

De même que le lemme 7 reformulait le lemme 6, le lemme 9 se reformule en

[2]On adopte ici la notation X_1^n pour $X_1,\ldots,X_n$, par souci de compacité.

Lemme 10 *$H(X,Y|Z) \leq H(X|Z)+H(Y|Z)$ avec égalité ssi X et Y sont indépendantes sachant Z.*

Pour le montrer (**exercice**), considérer le développement de $H(X,Y,Z)$ par la formule des conditionnements successifs.

2.3 Méthode graphique pour le calcul entropique

Les propriétés de l'entropie, avec toutes leurs variantes (conditionnelle, multivariables, liens avec l'indépendance...) peuvent sembler compliquées à retenir. Fort heureusement, il existe une méthode "graphique" intuitive qui permet de retrouver les règles de décomposition d'une entropie de plusieurs variables.

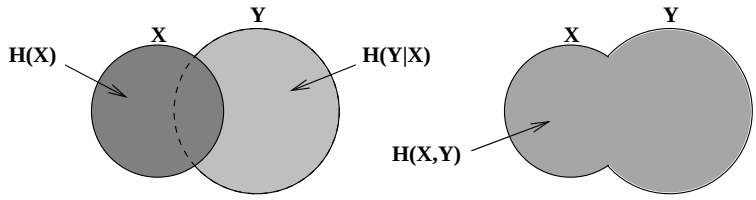

Figure 2.4: Représentation graphique des règles de décomposition de l'entropie.

L'idée consiste à représenter chaque variable par un ensemble, dont "l'aire" figure la quantité d'information portée par cette variable. La figure 2.4 donne la règle de base de cette représentation : l'aire de l'union des ensembles X et Y représente $H(X,Y)$, et le complémentaire de l'ensemble X dans "$X \cup Y$" représente donc $H(Y|X)$ puisque $H(X,Y) = H(X)+H(Y|X)$. On retrouve en particulier $H(Y|X) \leq H(Y)$. En généralisant (lemme 8)

$$\begin{array}{rcl} H(X_1,\ldots,X_n) & \longleftrightarrow & \text{aire de } (X_1 \cup \ldots \cup X_n) \\ H(X_1,\ldots,X_k|X_{k+1},\ldots,X_n) & \longleftrightarrow & \text{aire de } [(X_1 \cup \ldots \cup X_k) \setminus (X_{k+1} \cup \ldots \cup X_n)] \end{array}$$

Par exemple, avec trois ensembles (figure 2.5) on retrouve $H(X,Y,Z) = H(X) + H(Y|X) + H(Z|X,Y)$.

L'intersection des ensembles X et Y représente en quelque sorte une "information commune" aux deux variables, qui a pour valeur $H(X)-H(X|Y) = H(Y)-H(Y|X)$ et que l'on étudiera plus loin (section 2.5). En vertu des lemmes 6 et 7, cette quantité s'annule ssi on a indépendance de X et Y, ce qui donne la figure 2.6.

Avec trois ensembles, on retrouve les versions généralisées de ce résultat : $H(X|Y)-H(X|Y,Z) = H(Y|Z) - H(Y|X,Z)$ représente une information partagée entre X et Y lorsque Z est connue, qui ne s'annule qu'en cas d'indépendance conditionnelle (figure 2.7).

Un dernier cas mérite d'être mentionné (fig. 2.8), correspondant à la situation $X = f(Y)$. Ce diagramme illustre les propriétés énoncées aux lemmes 1 et 4.

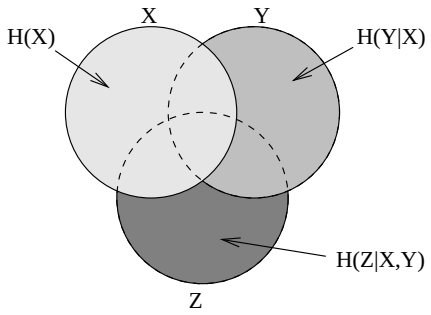

Figure 2.5: Décomposition de $H(X, Y, Z)$.

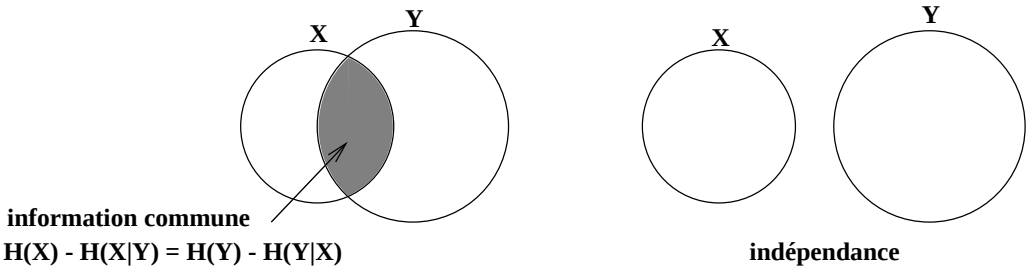

Figure 2.6: Représentation de l'indépendance de X et Y, on a alors $H(X, Y) = H(X) + H(Y)$.

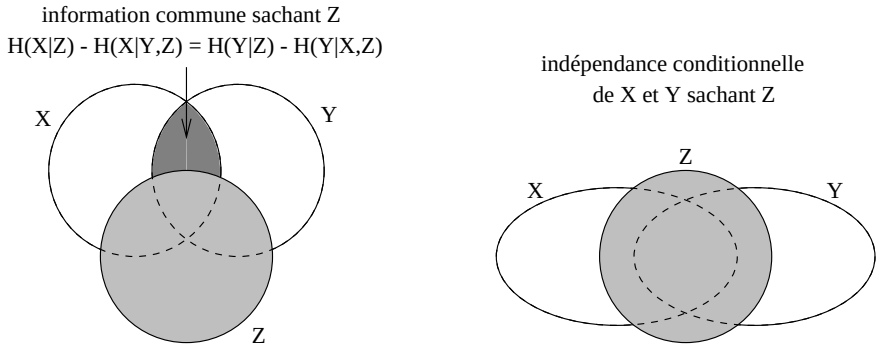

Figure 2.7: Indépendance conditionnelle de X et Y sachant Z.

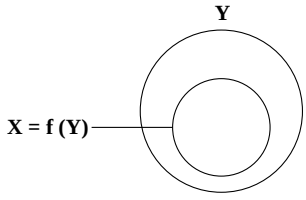

Figure 2.8: Cas où $X = f(Y)$.

2.4 Entropie relative entre deux lois (distance de Kullback-Leibler)

On a donné comme interprétation de l'entropie : "c'est le nombre moyen de bits nécessaires pour transmettre les valeurs produites par une source d'information". Cette affirmation n'a pas encore été démontrée (ce sera fait au chapitre 3), on en a seulement donné une illustration intuitive (section 2.1.3). Dans cette interprétation, on affectait[3] $-\log_2 p(x)$ bits à une valeur x de vraisemblance $p(x)$.

Imaginons maintenant que, par erreur, on a construit une technique de codage pour X en supposant la loi q, alors que X est de loi p. On a donc affecté $\log_2 q(x)$ bits à x au lieu de $\log_2 p(x)$. La transmission va alors demander

$$N_q(X) = -\sum_x p(x) \log_2 q(x)$$

bits en moyenne. Est-ce plus ou moins que $H(X)$? On étudie alors la différence $N_q(X) - H(X)$

Définition 5 *L'*entropie relative - *ou* distance de Kullback - *entre les lois p et q est donnée par*

$$D(p\|q) \quad = \quad \sum_x p(x) \log_2 \frac{p(x)}{q(x)} \tag{2.5}$$

La somme se fait sur les x tels que $p(x) \neq 0$ (grâce à la convention $0 \log_2 0 = 0$). Mais si q a un support plus petit que celui de p, i.e. $\exists x : p(x) \neq 0, q(x) = 0$, alors $D(p\|q) = +\infty$. Noter que $D(p\|q) \neq D(q\|p)$ en général.

Lemme 11 *$D(p\|q) \geq 0$, et s'annule ssi $p = q$.*

Donc, en se trompant de loi pour concevoir un code de transmission, on émet en moyenne $D(p\|q)$ bits de trop. La preuve de ce lemme est une application directe de l'inégalité de Jensen, pour les fonctions convexes.

Rappels sur les fonctions convexes

Une fonction f est dite convexe sur un intervalle I ssi

$$\forall s, t \in I, \quad \forall 0 \leq \lambda \leq 1, \quad f(\lambda s + (1-\lambda)t) \leq \lambda f(s) + (1-\lambda) f(t)$$

f est dite strictement convexe sur I ssi l'inégalité est stricte pour $0 < \lambda < 1$ et $s \neq t$. Exemple : $f(s) = |s|$ et $g(s) = s^2$ sont convexes, mais seule g est strictement

[3] C'est la "règle de Shannon", qui sera vue plus loin.

convexe. Cette caractérisation s'étend trivialement par récurrence (**exercice**) en

$$\forall s_1, \ldots, s_n \in I, \quad \forall \lambda_1, \ldots, \lambda_n \geq 0, \quad \sum_{i=1}^{n} \lambda_i = 1, \quad f(\sum_i \lambda_i s_i) \leq \sum_i \lambda_i f(s_i)$$

La stricte convexité se caractérise encore par une inégalité stricte lorsque les s_i ne sont pas tous identiques, et les $\lambda_i > 0$. Rien n'empêche maintenant de voir les λ_i

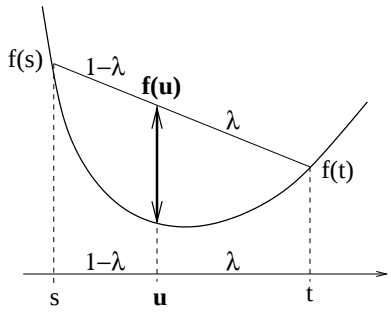

Figure 2.9: Une fonction strictement convexe. Ici $u = \lambda s + (1 - \lambda)t$.

comme des probabilités. Il vient immédiatement

Lemme 12 *Soient Y une variable aléatoire discrète à valeurs numériques, et f une fonction convexe (sur un intervalle de $\mathbb{R}$ contenant $\mathcal{Y}$), alors*

$$\boldsymbol{E}\left[f(Y)\right] \geq f[\boldsymbol{E}\left(Y\right)]$$

Si f est strictement convexe, l'inégalité est stricte, sauf lorsque Y est une variable déterministe (c'est à dire a une loi singulière portant sur une seule valeur) auquel cas on a égalité.

Pour montrer le lemme 11, on pose alors $Y = \frac{q(X)}{p(X)}$. Observer que c'est bien une variable aléatoire, à valeurs numériques, et fonction de X. Comme $f = -\log_2$ qui est strictement convexe, il vient

$$\begin{aligned} D(p\|q) &= \boldsymbol{E}\left(-\log_2 Y\right) \\ &\geq -\log_2 \boldsymbol{E}\left(Y\right) \\ &= -\log_2 \Big(\sum_{x:\, p(x)>0} p(x) \frac{q(x)}{p(x)} \Big) \\ &\geq -\log_2 \Big(\sum_x q(x) \Big) \\ &= -\log_2 1 \\ &= 0 \end{aligned}$$

D'après le lemme précédent, l'égalité est obtenue ssi Y est déterministe, c'est à dire ssi $\frac{q(x)}{p(x)} = cste = 1$, soit $p = q$.

2.5 Information mutuelle entre deux variables aléatoires

Nous allons maintenant définir et étudier la quantité "information commune" qui a été mise en évidence sur la figure 2.6.

2.5.1 Définition

Définition 6 *L'*information mutuelle *entre les variables X et Y est*

$$\begin{aligned} I(X;Y) &= H(X) + H(Y) - H(X,Y) \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \end{aligned}$$

Il est facile de vérifier l'équivalence de ces trois formules à l'aide du lemme 3. C'est évidemment une quantité symétrique : $I(X;Y) = I(Y;X)$. Noter le " ; " dans la notation $I(X;Y)$.

Quelle est la signification de cette quantité ? Si l'on veut transmettre une valeur observée pour le couple (X,Y), il nous faudra $H(X,Y)$ bits en moyenne. On peut opérer cette transmission de la façon suivante : on envoie d'abord la valeur de X, ce qui demande $H(X)$ bits en moyenne. Puis on communique la valeur de Y. Comme le récepteur a déjà reçu X, il suffit de lui envoyer l'information suffisante pour déterminer Y lorsque X est connue, soit $H(Y|X)$ bits en moyenne. On peut procéder d'une autre façon. On envoie d'abord X, puis Y, sans tenir compte du X envoyé. Cela demande en moyenne $H(X) + H(Y)$ bits. Cette technique est moins efficace car elle ne tient pas compte du lien statistique entre X et Y. Et de fait elle utilise $I(X;Y) = H(X) + H(Y) - H(X,Y)$ bits supplémentaires en moyenne. $I(X;Y)$ mesure donc la quantité d'information commune à X et Y, ce qui donne une idée du "degré de dépendance" de X et Y.

Noter que X et Y peuvent représenter chacun plusieurs variables, par exemple $X = (X_1, \ldots, X_n)$ et $Y = (Y_1, \ldots, Y_m)$. On écrit alors

$$I(X;Y) = I(X_1, \ldots, X_n; Y_1, \ldots, Y_m)$$

d'où l'importance du " ; ".

2.5.2 Propriétés

Lemme 13

$$\begin{aligned} I(X;Y) &= \sum_{x,y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} \\ &= D(I\!P_{X,Y} \| I\!P_X I\!P_Y) \\ &= I\!E\left[\log_2 \frac{p(X,Y)}{p(X)p(Y)}\right] \end{aligned}$$

Ce lemme peut être vu comme une définition équivalente de $I(X;Y)$. On l'obtient trivialement à partir de la première expression dans la définition 6 (**exercice**).

Lemme 14 *$I(X;Y) \geq 0$ et s'annule ssi X et Y sont indépendantes.*

En effet, $I(X;Y)$ est la distance de Kullback entre la loi jointe de $I\!P_{X,Y}$ et le produit des marginales $I\!P_X I\!P_Y$. C'est donc une quantité positive (lemme 11) qui ne s'annule que si les deux lois sont identiques, i.e. $I\!P_{X,Y} = I\!P_X I\!P_Y$, ce qui est la définition de l'indépendance.

Conséquence immédiate, $H(X) \geq H(X|Y)$, avec égalité ssi X et Y sont indépendantes. Cela prouve ainsi le lemme 6 - et par suite le lemme 7 - qui était resté en suspens.

Lemme 15 *$I(X;Y) \leq \min(H(X), H(Y))$ avec égalité ssi l'une des variables est fonction de l'autre.*

Preuve directe par le lemme 4. On notera en particulier que $I(X;X) = H(X)$.

2.6 Information mutuelle conditionnelle

Comme pour l'entropie, on étend ici la notion d'information mutuelle à des lois conditionnelles. On va ainsi définir "l'information commune à X et Y sachant Z" de la figure 2.7.

2.6.1 Définition

Définition 7 *L'*information mutuelle *entre les variables X et Y sachant Z est*

$$\begin{aligned} I(X;Y|Z) &= H(X|Z) + H(Y|Z) - H(X,Y|Z) \\ &= H(X|Z) - H(X|Y,Z) \\ &= H(Y|Z) - H(Y|X,Z) \end{aligned}$$

Ces formules ne diffèrent de la définition 6 que par le conditionnement en Z. On laisse le soin au lecteur d'en vérifier l'équivalence (**exercice**).

La signification de $I(X;Y|Z)$ va de soi : c'est l'information commune à X et Y lorsque Z est connue. Dans un point de vue transmission, c'est "le nombre de bits communs à X et Y" lorsque émetteur et récepteur connaissent Z.

2.6.2 Propriétés

Lemme 16 *On peut définir l'information mutuelle entre X et Y sachant la valeur z pour Z à partir de la loi conditionnelle $\mathbb{P}_{X,Y|Z=z}$*

$$\begin{aligned} I(X;Y|Z=z) &= D\left(\mathbb{P}_{X,Y|Z=z} \| \mathbb{P}_{X|Z=z}\mathbb{P}_{Y|Z=z}\right) \\ &= \sum_{x,y} p(x,y|z) \log_2 \frac{p(x,y|z)}{p(x|z)p(y|z)} \\ &= \mathbb{E}\left[\log_2 \frac{p(X,Y|Z)}{p(X|Z)p(Y|Z)} \,\middle|\, Z=z\right] \end{aligned}$$

L'information mutuelle conditionnelle de X et Y sachant Z se reformule alors comme

$$\begin{aligned} I(X;Y|Z) &= \sum_z I(X;Y|Z=z) \cdot p(z) \\ &= \sum_{x,y,z} p(x,y,z) \log_2 \frac{p(x,y|z)}{p(x|z)p(y|z)} \\ &= \mathbb{E}\left[\log_2 \frac{p(X,Y|Z)}{p(X|Z)p(Y|Z)}\right] \end{aligned}$$

Ce lemme donne une définition équivalente de $I(X;Y|Z)$. On l'obtient trivialement à partir de la première expression dans la définition 6 (**exercice**). On remarquera la similitude avec la définition de l'entropie conditionnelle : on commence par définir $I(X;Y|Z=z)$, que l'on moyenne en z pour avoir $I(X;Y|Z)$.

Lemme 17 *$I(X;Y|Z) \geq 0$ et s'annule ssi X et Y sont indépendantes sachant Z.*

En effet, $I(X;Y|Z=z) \geq 0$ et s'annule ssi $\mathbb{P}_{X;Y|Z=z} = \mathbb{P}_{X|Z=z}\mathbb{P}_{Y|Z=z}$ pour toute valeur de z telle que $p(z) > 0$. C'est la définition de l'indépendance conditionnelle.

Lemme 18 *$I(X;Y|Z) \leq I(X;Y)$ est* faux *en général.*

En d'autres termes, contrairement à l'entropie, le conditionnement ne réduit pas l'information mutuelle. Au contraire, il peut la faire augmenter. On verra section 2.8 une condition suffisante pour que le conditionnement réduise l'information mutuelle.

Exercice. Soient X et Y deux variables indépendantes à valeurs numériques. On construit $Z = X + Y$. Vérifier sur cet exemple que $I(X;Y|Z) > I(X;Y)$.

2.6.3 Formule des conditionnements successifs

Comme pour l'entropie, cette formule permet de calculer une information mutuelle "par morceaux". On va la retrouver sur les diagrammes de Venn.

Lemme 19 (Formule des conditionnements successifs)

$$I(X_1,\ldots,X_n;Y) = I(X_1;Y) + I(X_2;Y|X_1) + \ldots + I(X_n;Y|X_1,\ldots,X_{n-1})$$

La preuve repose sur la formule des conditionnements successifs établie pour l'entropie.

$$\begin{aligned} I(X_1,\ldots,X_n;Y) &= H(X_1,\ldots,X_n) - H(X_1,\ldots,X_n|Y) \\ &= \sum_i H(X_i|X_1,\ldots,X_{i-1}) - \sum_i H(X_i|X_1,\ldots,X_{i-1},Y) \\ &= \sum_i [H(X_i|X_1,\ldots,X_{i-1}) - H(X_i|X_1,\ldots,X_{i-1},Y)] \\ &= \sum_i I(X_i;Y|X_1,\ldots,X_{i-1}) \end{aligned}$$

Comme pour l'entropie, cette formule se lit

$$\begin{aligned} &I(X_1,\ldots,X_n;Y) = \\ &\underbrace{I(X_1;Y) + \ldots + I(X_k;Y|X_1^{k-1})}_{I(X_1,\ldots,X_k;Y)} + \underbrace{I(X_{k+1};Y|X_1^k) + \ldots + I(X_n;Y|X_1^{n-1})}_{I(X_{k+1},\ldots,X_n;Y|X_1,\ldots,X_k)} \end{aligned}$$

2.7 Méthode graphique pour le calcul entropique (suite)

Comme on l'avait indiqué sur les figures 2.6 et 2.7, l'information mutuelle se représente comme l'intersection de deux ensembles, et l'information mutuelle conditionnelle comme cette intersection moins l'ensemble conditionnant, ce qui donne les nouvelles règles

$$\begin{aligned} I(X_1,\ldots,X_n;Y_1,\ldots,Y_m) &\longleftrightarrow \text{aire de } (X_1\cup\ldots\cup X_n)\cap(Y_1\cup\ldots\cup Y_m) \\ I(X_1,\ldots,X_n;Y_1,\ldots,Y_m|Z_1,\ldots,Z_l) &\longleftrightarrow \text{aire de } [(X_1\cup\ldots\cup X_n)\cap(Y_1\cup\ldots\cup Y_m) \\ &\qquad \setminus(Z_1\cup\ldots\cup Z_l)] \end{aligned}$$

On trouvera en illustration la décomposition de $I(X;Y,Z)$ sur la figure 2.10.

Les diagrammes de Venn constituent un excellent outil à la fois mnémotechnique et de calcul. On peut notamment les utiliser pour retrouver les formules "en log" de toutes les quantités définies dans ce chapitre, à partir de la seule définition de l'entropie. Lorsqu'il y a plus de 4 variables à représenter, il faut toutefois prendre garde de faire figurer dans le diagramme toutes les intersections possibles (2^n zones s'il y a n variables), ce qui peut demander de représenter une variable par plusieurs ensembles.

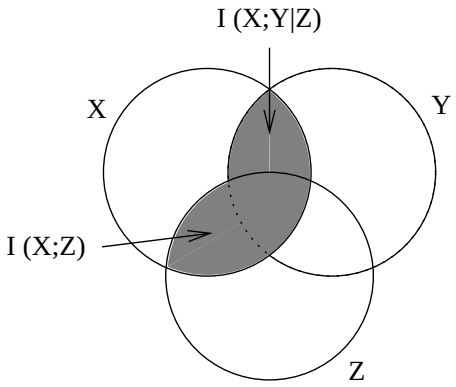

Figure 2.10: Décomposition de $I(X;Y,Z)$ à l'aide des diagrammes de Venn.

2.8 Théorème sur le traitement de l'information

Définition 8 *Les variables $X_1, \ldots, X_n$ forment une chaîne de Markov ssi*

$$\forall k,\ \mathbb{P}_{X_1,\ldots,X_{k-1},X_{k+1},\ldots,X_n|X_k} = \mathbb{P}_{X_1,\ldots,X_{k-1}|X_k}\mathbb{P}_{X_{k+1},\ldots,X_n|X_k}$$

c'est à dire ssi le passé $(X_1, \ldots, X_{k-1})$ et le futur $(X_{k+1}, \ldots, X_n)$ sont indépendants sachant le présent X_k, ou de façon équivalente ssi

$$\forall k,\ \mathbb{P}_{X_{k+1}|X_1,\ldots,X_k} = \mathbb{P}_{X_{k+1}|X_k}$$

c'est à dire si le futur ne dépend du passé que par le présent.

Ces deux formulations sont équivalentes (**exercice**), mais on utilise plus souvent la seconde qui permet de décomposer la loi de $(X_1, \ldots, X_n)$ en

$$p(x_1, \ldots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_2)\cdots p(x_n|x_{n-1})$$

Une chaîne de Markov se caractérise ainsi par une loi initiale $\mathbb{P}_{X_1}$ et une suite de *probabilités de transition*[4] $\mathbb{P}_{X_{k+1}|X_k}$. Si ces dernières ne dépendent pas de k, la chaîne est dite homogène. Cela justifie la notation que nous adopterons dans la suite : $X_1 \to X_2 \to \ldots \to X_n$. Remarque : la symétrie de la première définition nous indique que l'on a aussi $X_n \to X_{n-1} \to \ldots \to X_1$.

Exemple. La suite des scores lors d'une partie de tennis constitue une chaîne de Markov.

En termes de diagramme de Venn, on peut montrer (par récurrence) qu'une chaîne de Markov se représente comme sur la figure 2.11 (**exercice**).

[4]Une façon habituelle, pour des variables à valeurs numériques, de définir $P_{X_{k+1}|X_k}$ consiste à poser $X_{k+1} = X_k + W_{k+1}$ où W_{k+1} est indépendante de $(X_1, \ldots, X_k)$.

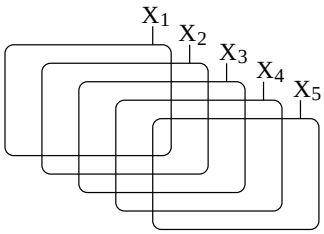

Figure 2.11: Représentation de la chaîne de Markov $X_1 \to X_2 \to X_3 \to X_4 \to X_5$ en diagramme de Venn.

Théorème 1 (data processing theorem)

$$X \to Y \to Z \qquad \Rightarrow \qquad I(X;Y) \geq I(X;Z)$$

Bien entendu, par raison de symétrie, on a aussi $I(Y;Z) \geq I(X;Z)$. Pour montrer ce résultat, on calcule $I(X;Y,Z)$ de deux façons (via le lemme 19)

$$\begin{aligned} I(X;Y,Z) &= I(X;Y) + I(X;Z|Y) \\ &= I(X;Z) + I(X;Y|Z) \end{aligned}$$

Or $I(X;Z|Y) = 0$ car X et Z sont indépendantes sachant Y, et $I(X;Y|Z) \geq 0$, cqfd.

Corollaire 1 *Si $X \to Y \to Z$, alors $I(X;Y|Z) \leq I(X;Y)$.*

C'est un sous-produit de la preuve, qui donne une condition suffisante pour que le conditionnement réduise l'information mutuelle entre deux variables. Rappelons que l'inégalité est fausse en général.

Exercice. Visualiser ces deux résultats sur un diagramme de Venn.

Application

On cherche à transmettre $X \in \{0, 1, \ldots, N-1\}$, uniformément distribuée, à travers un canal bruité. On récupère $Y = X + W$ modulo N à la sortie, où le bruit W est indépendant de X. Montrer que $H(X|Y) > 0$ ssi W n'est pas déterministe. Un ingénieur prétend qu'en mettant en oeuvre des traitements sophistiqués de sa fabrication, il est capable de reconstruire X à partir de Y. Est-ce possible ?

Soit f l'algorithme de restauration mis en oeuvre, de sorte que $Z = f(Y)$ est la version "débruitée" du message reçu. On a clairement $X \to Y \to Z = f(Y)$, et donc $I(X;Y) \geq I(X;Z)$. Si l'on peut reconstruire X par f, alors $I(X;Z) = I(X;X) = H(X)$. Donc $I(X;Y) \geq H(X)$, ce qui n'est posible que si X est une fonction de Y, c'est à dire si $H(X|Y) = 0$. Donc la fonction f ne débruite que les situations sans bruit...

Plus généralement, le théorème énonce donc qu'en traitant l'information Y, on ne peut pas retrouver de l'information perdue sur X : $I(X;Y) \geq I(X;f(Y))$. Attention, cela ne signifie pas que f est inutile ; elle peut servir à mettre en forme Y, de façon à séparer ce qui reste d'information utile de ce qui perturbe le message (cf tous les algorithmes de traitement de signal).

Inégalité de Fano

Cette inégalité donne une borne inférieure à la probabilité d'erreur dans une chaîne de transmission.

Considérons une chaîne de transmission $X \to Y$, où $X \in \mathcal{X}$ est la valeur émise, et la valeur reçue $Y \in \mathcal{Y}$ est une version bruitée de X. On construit un estimateur $\hat{X}$ de X par $\hat{X} = f(Y)$, de sorte que $X \to Y \to \hat{X}$. On définit enfin une variable d'erreur E par $E = \mathbb{1}(X \neq \hat{X})$, qui est donc une fonction de (X,Y). La probabilité d'erreur P_e est donc $\mathbb{E}(E) = \mathbb{P}(X \neq \hat{X})$. On note $H(E) = H(P_e)$.

Lemme 20 (inégalité de Fano)

$$H(P_e) + P_e \log_2(|\mathcal{X}| - 1) \geq H(X|Y)$$

Cette inégalité peut être affaiblie en

$$1 + P_e \log_2 |\mathcal{X}| \geq H(X|Y) \qquad ou \qquad P_e \geq \frac{H(X|Y) - 1}{\log_2 |\mathcal{X}|}$$

On remarque au passage que $P_e = 0$ entraine $H(X|Y) = 0$, ce qui est assez intuitif. La preuve vient en calculant $H(E,X|Y)$ de deux façons

$$\begin{aligned} H(E,X|Y) &= H(X|Y) + \underbrace{H(E|X,Y)}_{=0} \\ &= \underbrace{H(E|Y)}_{\leq H(P_e)} + \underbrace{H(X|E,Y)}_{\leq P_e \log_2(|\mathcal{X}|-1)} \end{aligned}$$

En effet, E est fonction de X,Y, donc $H(E|X,Y) = 0$, et le conditionnement réduit l'entropie donc $H(E|Y) \leq H(E) = H(P_e)$. Enfin $H(E|X,Y)$ se borne par

$$\begin{aligned} H(X|E,Y) &= \mathbb{P}(E=0)\, H(X|E=0,Y) + \mathbb{P}(E=1)\, H(X|E=1,Y) \\ &\leq (1-P_e)\cdot 0 + P_e \cdot H(X|X \neq f(Y)) \\ &\leq P_e \cdot \log_2(|\mathcal{X}| - 1) \end{aligned}$$

puisque si $E = 0$, $X = f(Y)$, et si $E = 1$, X ne peut prendre que les valeurs de $\mathcal{X} \setminus \{f(Y)\}$.

Chapter 3

Codage de source, compression de données

3.1 Introduction

Une source d'information a été définie comme une variable aléatoire X. Nous nous intéressons maintenant à la suite des valeurs que peut émettre cette source, que nous allons modéliser comme une suite $X_1, X_2, \ldots, X_n$ de variables aléatoires. Pour simplifier, nous supposerons que les X_i sont indépendantes - on dit alors que la source est sans mémoire - et identiquement distribuées (i.i.d.) de loi $\mathbb{P}_X$. Les résultats de ce chapitre s'étendent néanmoins au cas de sources avec mémoire, i.e. telles que les X_i sont liées.

L'objectif de ce chapitre est de coder efficacement un *message* produit par X, c'est à dire une suite de valeurs $x_1, \ldots, x_n$, afin de le transmettre sur un *canal sans bruit*. Noter qu'un espace de stockage (disque dur, CDROM, etc.) est un cas particulier de canal sans bruit. Pour le codage, on dispose de D symboles que l'on appellera des *lettres*[1]. Le plus souvent, on supposera $D = 2$, ce qui signifie que les suites de valeurs de X sont codées sous forme de trains de bits ; la généralisation à $D > 2$ est immédiate. Par "codage efficace", on entend que le train de bits résultant du codage doit être le plus court possible, en moyenne. Si le codage est destiné à de la transmission sur un canal parfait, cela garantit une transmission rapide. S'il est destiné à du stockage d'information, cela garantit un encombrement mémoire minimal. On parle alors de compression de données.

Nous avons déjà vu au chapitre 2 quelques arguments allant dans le sens d'un codage efficace. On a mentionné par exemple qu'il doit accorder peu de bits (ou de lettres) aux valeurs $x \in \mathcal{X}$ fréquentes, et beaucoup aux valeurs rares. C'est par exemple une idée qui a été utilisée dans la construction de l'alphabet Morse, lequel repose sur $D = 4$ lettres : *trait* $(-)$, *point* $(\cdot)$, *silence court, silence long*. Ainsi *trait*

[1] Ce vocable peut changer selon les auteurs.

et *point* correspondent à des lettres (de l'alphabet) très fréquentes $t = -$ et $e = \cdot$. De même $i = \cdot\cdot$, etc. En revanche $x = -\cdot\cdot-$, $y = -\cdot--$ et $z = --\cdot\cdot$. Si on y regarde de plus près, on constate que ce codage est adapté à la distribution statistique des lettres de l'alphabet pour la langue anglaise.

Le chapitre est organisé de la façon suivante. On s'intéresse tout d'abord à la distribution statistique des suites $x_1, \ldots, x_n$, qui révèle une propriété très particulière : les séquences se partagent en séquences "normales" ou "typiques", et séquences "anormales" ou "atypiques". Les premières concentrent l'essentiel de la probabilité et sont toutes équiprobables, les autres sont rares. Cela nous permettra d'établir une première technique de codage, qui traite *en bloc* une suite de lettres. On s'attachera ensuite à construire des codes de type *lettre à lettre*, comme pour le Morse, les plus efficaces possibles. On verra que cela demande de connaître la distribution $\mathbb{P}_X$. Lorsque cette distribution n'est pas connue exactement, on appliquera un algorithme dit "universel" de codage, très fréquemment utilisé en compression de données : l'algorithme de Lempel-Ziv, qui se cache derrière des utilitaires très courants comme *zip, gzip*, etc.

3.2 Propriété asymptotique d'équirépartition (AEP)

Nous nous appuyons ici sur la loi des grands nombres établie au chapitre 1 section 1.3.3 pour étudier la vraisemblance de longues suites $x_1, \ldots, x_n$.

3.2.1 Définition de l'AEP

Théorème 2 *Si les variables $X_1, \ldots, X_n, \ldots$ sont i.i.d. de loi $\mathbb{P}_X$, alors*

$$-\frac{1}{n}\log_2 \mathbb{P}(X_1, \ldots, X_n) \xrightarrow[n]{\mathbb{P}} H(X)$$

où $H(X)$ est l'entropie de la loi $\mathbb{P}_X$, et la convergence se fait en probabilité (c'est le sens de $\mathbb{P}$ sur la flèche).

En effet, considérons la variable aléatoire du membre de gauche, qui est une fonction du n-uplet $(X_1, \ldots, X_n)$. Elle s'écrit

$$Z_n \triangleq -\frac{1}{n}\log_2 \mathbb{P}(X_1, \ldots, X_n) = \frac{1}{n}\sum_{i=1}^{n} -\log_2 \mathbb{P}(X_i)$$

On a vu que la variable aléatoire Z_n converge en probabilité vers une variable *déterministe* qui n'est autre que (voir section 1.3.3, en posant $Y_i = -\log_2 \mathbb{P}(X_i)$)

$$\mathbb{E}\left[-\log_2 \mathbb{P}(X_i)\right] = H(X)$$

Si on exprime cette convergence en probabilité plus clairement, on a

$$I\!P\left[\left|\frac{1}{n}\sum_{i=1}^{n} - \log_2 I\!P(X_i) - H(X)\right| > \varepsilon\right] \underset{n}{\longrightarrow} 0$$

Remarque. On a vu que cette convergence est aussi presque sûre (loi forte des grands nombres).

On peut exprimer sous une autre forme le théorème 2. Il nous dit en effet

$$I\!P(X_1, \ldots, X_n) \underset{n}{\xrightarrow{I\!P}} 2^{-nH(X)} \tag{3.1}$$

convergence en probabilité mais aussi presque sûre. Cela signifie que la vraisemblance $I\!P(x_1, \ldots, x_n)$ d'une suite de valeurs converge vers $2^{-nH(X)}$. Nous allons exploiter cette propriété pour définir une partition de l'ensemble des séquences de taille n.

Définition 9 *Soit $\varepsilon > 0$, on définit l'ensemble des* séquences typiques *de taille n par*

$$\begin{aligned} A_n^\varepsilon &= \left\{(x_1, \ldots, x_n) : \quad 2^{-n(H(X)+\varepsilon)} \leq p(x_1, \ldots, x_n) \leq 2^{-n(H(X)-\varepsilon)} \right\} \\ &= \left\{(x_1, \ldots, x_n) : \quad H(X) - \varepsilon \leq -\frac{1}{n}\log_2 p(x_1, \ldots, x_n) \leq H(X) + \varepsilon \right\} \end{aligned}$$

Une séquence $x_1, \ldots, x_n$ est donc typique si sa vraisemblance est proche de $2^{-nH(X)}$. Sinon elle est dite *atypique*[2].

Théorème 3 *Pour n suffisamment grand, on a*

$$\begin{aligned} I\!P(A_n^\varepsilon) &> 1 - \varepsilon \\ |A_n^\varepsilon| &\leq 2^{n(H(X)+\varepsilon)} \\ |A_n^\varepsilon| &\geq (1-\varepsilon)\, 2^{n(H(X)-\varepsilon)} \end{aligned}$$

En effet

1. La convergence en probabilité 3.1 se traduit par $I\!P(A_n^\varepsilon) \to 1$. Pour n assez grand, on a donc $I\!P(A_n^\varepsilon) > 1 - \varepsilon$.

[2]Il ne faut pas s'étonner de ces définitions, qui énoncent une propriété très intuitive. Considérons une variable binaire X, valant 1 avec la probabilité p. Si l'on observe une longue séquence de tirages de X, on observera une proportion p de "1", et une proportion $1-p$ de "0". Une telle séquence est dite typique de la loi de X, sinon elle est atypique. La séquence $0000\ldots0$ est par exemple d'autant plus atypique que n est grand.

2. La vraisemblance de chaque suite typique vérifie par définition

$$p(x_1, \ldots, x_n) \ \geq \ 2^{-n(H(X)+\varepsilon)}$$

On a donc

$$\begin{aligned} 1 \ \geq \ I\!P(A_n^\varepsilon) \ &= \sum_{(x_1,\ldots,x_n)\in A_n^\varepsilon} p(x_1, \ldots, x_n) \\ &\geq \sum_{(x_1,\ldots,x_n)\in A_n^\varepsilon} 2^{-n(H(X)+\varepsilon)} \\ &= \ |A_n^\varepsilon| \cdot 2^{-n(H(X)+\varepsilon)} \end{aligned}$$

d'où la seconde inégalité.

3. Pour la troisième, on procède de la même manière, en choisissant n de façon à garantir $I\!P(A_n^\varepsilon) > 1 - \varepsilon$.

Signification de ce théorème

Le théorème 3 peut se résumer schématiquement de la façon suivante. Lorsque n est grand, la probabilité d'avoir une séquence $(x_1, \ldots, x_n)$ typique est quasiment de 1. Il y a environ $2^{nH(X)}$ séquences typiques de taille n, et chacune est (environ) de probabilité $2^{-nH(X)}$. Tout se passe donc, pour n grand, comme si l'on avait une loi uniforme sur un ensemble de taille $2^{nH(X)}$.

3.2.2 Application au codage

On cherche maintenant à transmettre les valeurs produites par la source X en considérant des blocs de n valeurs consécutives, qui seront transmises simultanément. n est fixé dans toute cette section. Pour cela, on construit un code de transmission en distinguant les séquences typiques des atypiques.

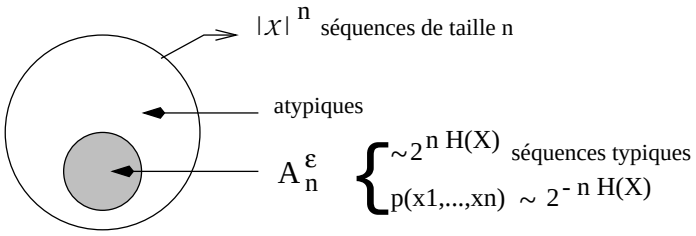

Figure 3.1: Partition de l'ensemble des séquences possibles en typiques et atypiques. La probabilité de l'ensemble gris est proche de 1.

Codage :

- On ordonne les séquences de A_n^ε (par exemple avec un ordre lexicographique) et on les numérote, de 1 à $2^{n\,(H(X)+\varepsilon)}$. Cela demande $\lceil n\,(H(X)+\varepsilon)\rceil$ bits[3].
- On fait de même pour les séquences atypiques. Cela demande $\lceil n\,\log_2|\mathcal{X}|\rceil$ bits (en comptant large).
- Enfin, on ajoute le préfixe "1" pour les séquences typiques, et "0" pour les autres.

Remarques. Il faut moins de bits pour coder une séquence typique. Pour les séquences atypiques, on a choisi un codage assez brutal.

Quelle est la longueur moyenne de ce code ? Notons x_1^n une séquence, et $L(x_1^n)$ sa longueur. On a

$$\begin{aligned}
I\!E\,[L(X_1^n)] &= \sum_{x_1^n} L(x_1^n)p(x_1^n) \\
&= \sum_{x_1^n\in A_n^\varepsilon} L(x_1^n)p(x_1^n) + \sum_{x_1^n\notin A_n^\varepsilon} L(x_1^n)p(x_1^n) \\
&= 1+\lceil n\,(H(X)+\varepsilon)\rceil\cdot I\!P(A_n^\varepsilon) + \lceil n\,\log_2|\mathcal{X}|\rceil\cdot[1-I\!P(A_n^\varepsilon)] \\
&\leq 1+[1+n\,(H(X)+\varepsilon)]\cdot 1 + (1+n\log_2|\mathcal{X}|)\cdot\varepsilon \\
&= n(H(X)+\varepsilon')
\end{aligned}$$

où $\varepsilon' = \varepsilon\log_2|\mathcal{X}|+2/n+\varepsilon/n+\varepsilon$ peut être rendu aussi petit que l'on veut, en jouant sur ε et n.

Théorème 4 *Soient $\alpha>0$ et $X_1,\ldots,X_n$ une séquence iid de valeurs émises par la source X, de loi $I\!P_X$. Si n est suffisamment grand, il existe un codage (binaire) des séquences $x_1,\ldots,x_n$ dont la longueur moyenne vérifie*

$$I\!E\left[\frac{1}{n}L(X_1^n)\right] \leq H(X)+\alpha$$

On peut donc transmettre X en utilisant $H(X)$ bits par variable, en moyenne, pourvu que l'on transmette des séquences assez longues.

Remarque. Si l'on avait codé les séquences sur $D>2$ symboles au lieu de $D=2$ bits, on aurait les mêmes résultats en remplaçant $\log_2$ par $\log_D$ dans toutes les expressions. En particulier $H_2(X)$ devient $H_D(X)$, $2^{nH_2(X)}$ devient $D^{nH_D(X)}$, etc.

[3] $\lceil x\rceil$ représente $n+1$ pour $n\leq x<n+1$

3.3 Classes de codes de source

Le problème de l'AEP est qu'elle impose de coder de longues séquences. Elle est donc d'un intérêt pratique limité (codes gigantesques, retard au décodage, etc.). On va donc s'attacher maintenant à construire des codes les plus efficaces possibles pour les valeurs elles-mêmes de X, non plus pour des séquences de valeurs.

3.3.1 Classes de codes

Définition 10 *Un* code de source *pour X est une fonction $C : \mathcal{X} \to \mathcal{A}^+$, où $\mathcal{A}$ est un ensemble de D symboles, et $\mathcal{A}^+$ l'ensemble des séquences de symboles de longueur finie non nulle. La suite de symboles $C(x)$ est le* mot de code *associé à la valeur x.*

Exemple. $\mathcal{X} = \{a, b, c, \ldots, p\}$ et $\mathcal{A} = \{0, 1\}$, donc $D = |\mathcal{A}| = 2$. On peut coder les valeurs de X par la transcription binaire du rang de la lettre dans l'alphabet, ce qui donne $C(a) = 0000$, $C(b) = 0001$, etc.

Exemple. On peut aussi choisir de ne pas garder les 0 non significatifs à gauche, ce qui donne $C(a) = 0$, $C(b) = 1$, $C(c) = 10$, $C(d) = 11$, etc.

On notera $L : \mathcal{A}^+ \to \mathbb{N}$ la fonction donnant la longueur d'une suite de symboles. $L_C(x)$, ou simplement $L(x)$ voire l_x, représentera la longueur du mot de code associé à x, soit $L(C(x))$. La longueur moyenne d'un code est ainsi

$$\mathbb{E}\left[L(X)\right] = \sum_{x \in \mathcal{X}} L[C(x)]\mathbb{P}_X(x) = \sum_x l_x p_x$$

Définition 11 *Un code (de source) C est dit* non-singulier *ssi C est injective.*

C'est la première propriété à lui demander : on doit pouvoir retrouver x à partir de $C(x)$.

Définition 12 *L'*extension *du code C à des séquences de valeurs de X est définie par la concaténation des mots de codes:*

$$\begin{array}{rccc} C & : & \mathcal{X}^n & \to & \mathcal{A}^+ \\ & & (x_1, x_2, \ldots, x_n) & \mapsto & C(x_1)C(x_2)\cdots C(x_n) \end{array}$$

Définition 13 *Un code est dit* à décodage unique *ssi son extension est non-singulière, pour toute valeur de n.*

Cela signifie qu'à partir de la suite de symboles résultant du codage d'une séquence $x_1, \ldots, x_n$, on peut retrouver sans ambiguité cette séquence. Noter que l'on peut avoir besoin de tous les symboles pour déterminer l'un des x_i.

Exercice. Montrer que le premier code de l'exemple ci-dessus est à décodage unique, mais pas le second.

Définition 14 *On dit que C est un code* préfixe *(ou* instantané*, ou* autoponctué*) ssi aucun mot de code n'est préfixe d'un autre.*

Cette propriété est très importante en pratique car elle permet de décoder une séquence transmise "à la volée". En effet, chaque fois qu'un mot de code est reconnu dans la suite de lettres, on peut placer une césure et afficher la valeur x de $\mathcal{X}$ à laquelle il correspond. Cette propriété évite de conserver une lettre de $\mathcal{A}$ pour marquer la séparation des mots. On notera qu'un code préfixe est nécessairement à décodage unique, mais l'inverse n'est pas vrai.

Exercice. On prend $\mathcal{A} = \{0, 1\}$ et $\mathcal{X} = \{\alpha, \beta, \gamma\}$. Ces valeurs sont codées respectivement par 00, 01 et 010. C est clairement un code non-singulier, mais il n'est pas instantané car $C(\beta)$ est un préfixe de $C(\gamma)$. Montrer comment décoder la séquence 0100000101000. Démontrer ensuite que ce code est à décodage unique (indication : raisonner sur le nombre de 0 dans les intervalles définis par les 1).

3.3.2 Inégalité de Kraft

On se pose maintenant la question suivante : la source X peut émettre N valeurs différentes, qu'il faut coder en utilisant un alphabet $\mathcal{A} = \{a_1, \ldots, a_D\}$ de D symboles. Le code obtenu doit être préfixe, et de longueur moyenne minimale[4].

Le premier résultat énoncé montre que la longueur des mots du code doit satisfaire quelques contraintes.

Théorème 5 (inégalité de Kraft) *Soit C un code préfixe pour la source X, avec $|\mathcal{X}| = N$, formé sur un alphabet $\mathcal{A} = \{a_1, \ldots, a_D\}$ de D symboles. Soient $l_1, \ldots, l_N$ les longueurs des mots de codes associés aux valeurs de X, on a*

$$\sum_{i=1}^{N} D^{-l_i} \quad \leq \quad 1 \tag{3.2}$$

Réciproquement, soient N longueurs $l_1, \ldots, l_N$ vérifiant (3.2), alors il existe un code préfixe de N mots, construits sur D lettres, satisfaisant ces longueurs.

Ce résultat a l'air compliqué, mais il n'y a pas de mystère. Il dit seulement que pour coder beaucoup de valeurs de X, il faut suffisamment de symboles (ou de bits)... Pour le démonter, on peut supposer $D = 2$ sans perte de généralité, et l'on va revenir à un schéma déjà rencontré[5].

[4] Ce critère de longueur moyenne a des implications très concrètes : chaque symbole à émettre demande de l'énergie au système de transmission, et donc représente un coût au final.

[5] Il nous avait déjà permis de deviner que l'entropie mesure le nombre moyen de bits nécessaires pour transmettre les valeurs produites par une source, chapitre 2, figure 2.3.

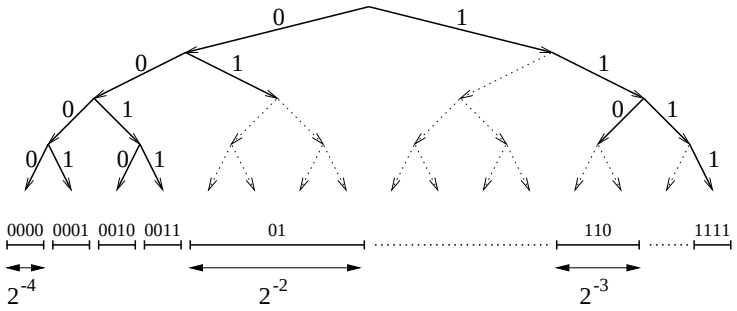

Figure 3.2: Un code (binaire) préfixe placé sur un arbre dyadique révèle en fait une coupure de cet arbre.

CN. On commence par tracer un arbre dyadique (infini), qui représente toutes les séquences binaires que l'on peut construire (fig. 3.2). Un chemin fini descendant représente ainsi un mot de code possible. Dans un code préfixe, aucun mot n'est préfixe d'un autre. Il s'ensuit que deux mots qui commencent par la même branche doivent nécessairement se séparer. Si on représente tous les mots du code par leur point d'arrivée dans l'arbre, on obtient ainsi une *coupure* de l'arbre dyadique. À toute feuille de profondeur i de cet arbre coupé on associe un intervalle de longueur 2^{-i}. La somme des longueurs de ces intervalles ne peut alors dépasser $2^0 = 1$, qui revient à couper l'arbre à sa racine. Cela donne l'inégalité de Kraft.

CS. Si on se donne maintenant les longueurs vérifiant l'inégalité, il faut montrer que l'on peut remonter à une coupure de l'arbre dyadique. Il est assez facile de construire une méthode pour cela (**exercice**). (Indication : classer les longueurs par ordre décroissant.)

Corollaire 2 *Le théorème reste valable pour $N = \infty$, c'est à dire pour un ensemble dénombrable de valeurs à coder.*

L'inégalité de Kraft simplifie beaucoup la recherche de codes de longueur moyenne optimale. En effet, il suffit maintenant de rechercher le meilleur jeu de longueurs l_x associées aux valeurs x, sous la contrainte (3.2), et l'on saura ensuite obtenir un code correspondant. Cette question fait l'objet de la section suivante.

Théorème 6 *Le théorème précédent reste valable pour un code à décodage unique (non préfixe).*

Pour le montrer, nous allons majorer $(\sum_i D^{-l_i})^k$. Il vient

$$\begin{aligned}\Big(\sum_i D^{-l_i}\Big)^k &= \sum_{i_1}\cdots\sum_{i_k} D^{-l_{i_1}}\cdots D^{-l_{i_k}} \\ &= \sum_{(x_1,\ldots,x_k)\in\mathcal{X}^k} D^{-L(x_1)-\ldots-L(x_k)} \\ &= \sum_{(x_1,\ldots,x_k)\in\mathcal{X}^k} D^{-L(x_1,\ldots,x_k)}\end{aligned}$$

où $L(x_1,\ldots,x_k) = L(x_1)+\ldots+L(x_k)$ est la longueur du code associé à la séquence $x_1,\ldots,x_k$ dans la k^{eme} extension du code C. Dans ce code étendu C^k, on peut regrouper les mots par longueurs. Soit $l_{\max}$ la taille du plus grand mot de code dans C, on a

$$\Big(\sum_i D^{-l_i}\Big)^k = \sum_{l=1}^{k\cdot l_{\max}} a(l) D^{-l}$$

où $a(l)$ est le nombre de mots de taille l dans C^k. Comme par hypothèse C^k est à décodage unique, il contient au plus D^l séquences de taille l, d'où

$$\Big(\sum_i D^{-l_i}\Big)^k \leq \sum_{l=1}^{k\cdot l_{\max}} D^l D^{-l} = k\cdot l_{\max}$$

Cela donne $\sum_i D^{-l_i} \leq (k\cdot l_{\max})^{1/k}$, vrai pour toute valeur de k. Par $k\to\infty$, il vient $\sum_i D^{-l_i} \leq 1$, c'est à dire l'inégalité de Kraft.

Puisque un code uniquement décodable mais non-préfixe vérifie aussi l'inégalité de Kraft, en reprenant ses longueurs l_i, on peut construire un code préfixe ayant exactement la même longueur moyenne. Il s'ensuit que les codes optimaux doivent être recherchés parmi les codes préfixes : les codes uniquement décodables ne sont pas meilleurs, tout en étant moins commodes...

3.4 Code de Shannon

On suppose que X peut prendre N valeurs, $\mathcal{X} = \{\alpha_1,\ldots,\alpha_N\}$, de probabilités respectives $p_1,\ldots,p_N$. Ces valeurs sont codées sur un alphabet $\mathcal{A}$ de D lettres, et donnent des mots de code de longueurs respectives $l_1,\ldots,l_N$, i.e. $l_i = L[C(\alpha_i)]$. La longueur moyenne du code est ainsi donnée par

$$L_X \triangleq I\!E\left[L\circ C(X)\right] = \sum_{i=1}^{N} l_i\cdot p_i \tag{3.3}$$

On cherche les codes donnant un L_X minimal. Comme on l'a vu plus haut, il suffit de se préoccuper des longueurs l_i ; le code (préfixe) peut s'en déduire simplement. Il s'agit donc de minimiser (3.3) sous la contrainte que les l_i sont entières et vérifient l'inégalité de Kraft.

3.4.1 Minoration de la longueur moyenne optimale

On va commencer par abandonner la contrainte $l_i \in \mathbb{N}$ et supposer $l_i \in \mathbb{R}$. Nous reste une contrainte d'inégalité : $\sum_i D^{-l_i} \leq 1$. Il est facile de voir que L_X est maximum lorsque cette contrainte est active, on va donc supposer l'égalité, ce qui nous ramène à une structure classique de problème d'optimisation, qui peut se résoudre par une technique simple de Lagrangien :

$$\begin{aligned} J_\lambda(l_1,\ldots,l_N) &= \sum_i l_i\, p_i + \lambda\big(\sum_i D^{-l_i} - 1\big) \\ \frac{\partial J_\lambda}{\partial l_i} &= p_i - \lambda\, D^{-l_i}\ln D = 0 \quad \Rightarrow \quad D^{-l_i^\star(\lambda)} = \frac{p_i}{\lambda \ln D} \end{aligned}$$

Pour satisfaire la contrainte, on voit alors qu'il faut prendre $\lambda = \frac{1}{\ln D}$, d'où

$$l_i^\star = -\log_D p_i$$

La longueur moyenne vérifie alors

$$L_X^\star = \sum_i l_i^\star p_i = -\sum_i p_i \log_D p_i = H_D(X)$$

Théorème 7 *Si X est codé par des mots formés sur un alphabet $\mathcal{A}$ de D lettres, avec un code préfixe (ou même seulement uniquement décodable), alors la longueur moyenne de ce code vérifie*

$$L_X \geq H_D(X)$$

La borne est atteinte ssi $p_i = D^{-l_i}$, $l_i \in \mathbb{N}$, c'est à dire si $\mathbb{P}_X$ est une loi D-adique.

Ce résultat entraine plusieurs commentaires. On avait déjà vu section 3.2 qu'il était possible de transmettre X avec $H(X)$ bits, ou $H_D(X)$ lettres, en moyenne par valeur x, ceci en considérant de longues séquences. On n'avait pas montré toutefois que c'était l'optimum. Ici, on vient de montrer qu'il n'est pas possible de faire mieux si l'on transmet valeur par valeur. Mais même en considérant des séquences la borne demeure. En effet, considérons la séquence $X_1,\ldots,X_n$ comme une seule variable. On ne peut la transmettre avec moins de $H_D(X_1,\ldots,X_n)$ lettres en moyenne. Or comme les X_i sont indépendantes, il vient $H_D(X_1,\ldots,X_n) = nH_D(X)$; il faut donc bien au minimum $H_D(X)$ lettres par valeur x (en moyenne). Ce théorème donne donc une borne absolue à la compression sans perte d'une séquence $X_1,\ldots,X_n$.

Remarque : on vient en fait de démontrer la signification intuitive de l'entropie que nous avions donnée au chapitre 2, section 2.1.3. On avait en particulier illustré cette interprétation par des lois dyadiques (fig. 2.2 et 2.3).

3.4.2 Majoration de la longueur moyenne optimale

Que se passe-t-il si l'on réintroduit les contraintes $l_i \in N$? Bien entendu, on fait moins bien, mais de combien ? Une idée naturelle consiste à prendre[6]

$$l_i \;=\; \lceil l_i^\star \rceil \;=\; \lceil -\log_D p_i \rceil$$

L'inégalité de Kraft reste satisfaite puisque l'on a augmenté les longueurs. Il vient alors

$$\begin{array}{ccccc} \displaystyle\sum_i -p_i \log_D p_i & \leq & \sum_i p_i\, l_i & < & 1 + \displaystyle\sum_i -p_i \log_D p_i \\ H_D(X) & \leq & L_X & < & 1 + H_D(X) \end{array}$$

Théorème 8 (1^{er} théorème de Shannon) *Le code optimal pour la source X, sur un alphabet de D lettres, est de longueur moyenne L_X vérifiant*

$$H_D(X) \;\leq\; L_X \;<\; 1 + H_D(X)$$

En d'autres termes, l'obligation d'utiliser des longueurs entières nous fait perdre au plus une lettre en moyenne, par rapport à la borne absolue. On est parfois contraint à cette perte. Par exemple, pour transmettre une variable X de Bernouilli, il faut un bit par valeur, quelle que soit l'entropie de X.

Peut-on faire mieux ? Oui, en regroupant les valeurs à émettre. Supposons que l'on transmet $X_1, \ldots, X_n, \ldots$ à partir d'un code pour k valeurs consécutives, il vient :

$$H_D(X_1, \ldots, X_k) \;\leq\; L_{X_1^k} \;<\; 1 + H_D(X_1, \ldots, X_k)$$

Comme $H_D(X_1, \ldots, X_k) = kH_D(X)$, il vient

$$H_D(X) \;\leq\; \frac{1}{k} L_{X_1^k} \;<\; \frac{1}{k} + H_D(X)$$

Avec ce procédé, on a réparti la lettre perdue sur k valeurs consécutives. On ne perd donc plus que $\frac{1}{k}$ lettre par valeur de X émise. On notera que c'est un progrès par rapport à l'AEP puisque l'on avait alors

$$\frac{1}{k} L_{X_1^k} \leq H_D(X) + \alpha$$

uniquement pour $k \to \infty$. Ici on contrôle la longueur de la séquence à utiliser pour une perte α donnée par rapport à $H_D(X)$.

[6]Rappel : $\lceil u \rceil$ représente la partie entière supérieure de u, c'est à dire $n+1$ lorsque $n < u \leq n+1$.

Commentaires

- La technique $l_i = \lceil -\log_D p_i \rceil$ pour construire un code préfixe est due à Shannon. Un code ainsi construit est donc appelé *code de Shannon*.
- Le premier théorème de Shannon contient en fait deux résultats : $L_X \geq H_D(X)$ (limite de la compression sans perte) et $\frac{1}{k} L_{X_1^k} \leq H_D(X) + \frac{1}{k}$ (on peut atteindre cette limite).

3.5 Code de Huffman

Shannon a cherché les longueurs optimales des mots de code dans $\mathbb{R}$, puis les a arrondies pour construire un code préfixe, dit "code de Shannon". Huffman, lui, a recherché directement un code préfixe optimal, en prenant en compte dès le début les contraintes $l_i \in \mathbb{N}$. Sa construction repose sur 4 idées force, que nous illustrons dans le cas binaire (l'extension est immédiate).

3.5.1 Quatre idées force

Idée 1

Les valeurs de x les moins probables doivent être associées aux mots les plus longs. Supposons que $\mathcal{X} = \{\alpha_1, \ldots, \alpha_N\}$ est ordonné de sorte que

$$p_1 \geq p_2 \geq \ldots \geq p_N$$

les longueurs de mots associées doivent alors vérifier

$$l_1 \leq l_2 \leq \ldots \leq l_N$$

car on veut minimiser $L_X = \sum_i l_i p_i$. Si les longueurs ne sont pas dans cet ordre, il suffira simplement de réaffecter les mots de code.

Idée 2

Les deux mots (D mots en général) les moins probables sont de même longueur : $l_{N-1} = l_N$. En effet, aucun mot du code n'est préfixe de $C(\alpha_N)$. Soit $l = \max_{i=1}^{N-1} l_i = l_{N-1}$; il vient que les lettres de rang $l+1$ à l_N dans $C(\alpha_N)$ sont inutiles pour distinguer $C(\alpha_N)$ des autres mots du code. On peut donc les retirer, ce qui donne un meilleur code préfixe, avec $l_N = l_{N-1}$.

Idée 3

On peut s'arranger pour que les deux (D) derniers mots $C(\alpha_{N-1})$ et $C(\alpha_N)$ ne diffèrent que par le dernier bit (la dernière lettre). Cela découle de la CS de l'inégalité de Kraft : on peut s'arranger pour que les deux (D) dernières longueurs soient associées à des chemins parallèles dans l'arbre dyadique (D-adique), divergeant seulement au dernier bit (à la dernière lettre). Cela donne, dans le cas binaire

$$\begin{aligned} C(\alpha_{N-1}) &= \varepsilon_1\,\varepsilon_2\,\cdots\,\varepsilon_{l_N-1}\,0 \\ C(\alpha_N) &= \underbrace{\varepsilon_1\,\varepsilon_2\,\cdots\,\varepsilon_{l_N-1}}_{l_{N-1}=l_N}\,1 \end{aligned}$$

Idée 4

On peut réduire les deux (D) derniers mots du code à leur préfixe commun ($\varepsilon_1\,\varepsilon_2\,\cdots\,\varepsilon_{l_N-1}$ dans l'exemple ci-dessus), ce qui représente la paire (le D-uplet) (α_{N-1}, α_N), de probabilité $p_{N-1} + p_N$. On obtient ainsi un nouveau code préfixe sur les valeurs $\{\alpha_1, \ldots, \alpha_{N-2}, (\alpha_{N-1}, \alpha_N)\}$. Construire le meilleur code préfixe sur N valeurs se ramène de cette façon à trouver le meilleur code sur $N-1$ valeurs. Cela permet d'initier une technique de construction récursive.

3.5.2 Mise en oeuvre (exemples)

Exemple 1

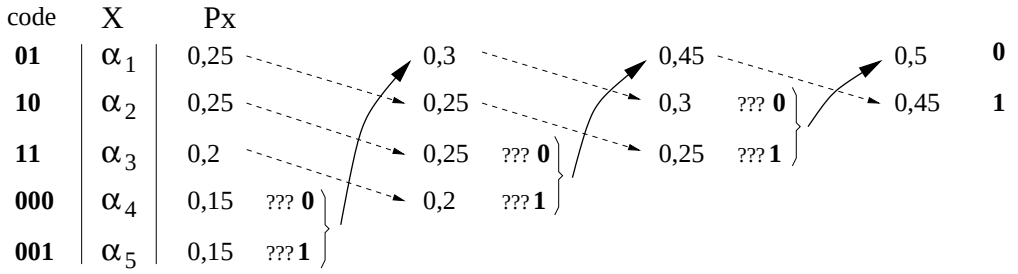

Figure 3.3: Construction d'un code de Huffman binaire.

La construction d'un code de Huffman reprend exactement les idées ci-dessus. On considère les 2 valeurs les moins probables, et l'on écrit qu'elles diffèrent par le dernier bit. Puis on regroupe ces deux valeurs, on reclasse les $N-1$ éléments restants par probabilités décroissantes, et on recommence. À la fin, il suffit de suivre le chemin à l'envers (i.e. de reséparer les symboles groupés) pour retrouver les bits utilisés pour les distinguer. La construction la plus commode utilise un tableau, comme figure 3.3. Bien entendu, lors de chaque regroupement de symboles, on a le

choix dans l'attribution de la lettre qui distinguera ces symboles dans le groupe. On peut donc construire plusieurs codes de Huffman.

Exemple 2

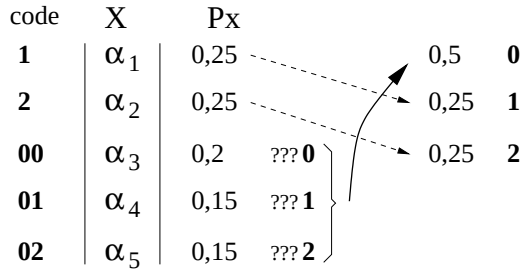

Figure 3.4: Construction d'un code de Huffman ternaire.

La figure 3.4 illustre la construction d'un code de Huffman avec $D = 3$. On remarque qu'à chaque étape le nombre de valeurs diminue de $D - 1$. On regroupe en effet D valeurs en une seule. Ainsi la procédure fonctionne correctement lorsqu'il y a $D + k(D - 1)$ symboles à coder, où k est le nombre d'étapes. Si ce n'est pas le cas, on complète $\mathcal{X}$ avec des symboles fictifs, de probabilité nulle, avant d'appliquer la méthode de construction. La technique est illutrée sur l'exemple 3.

Exemple 3

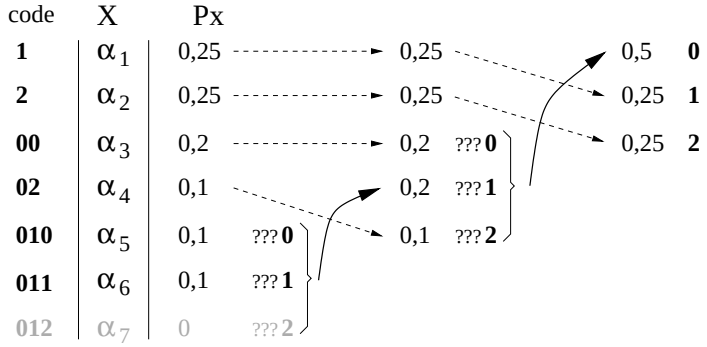

Figure 3.5: Construction d'un code de Huffman ternaire, avec ajout de symboles fictifs.

3.5.3 Résultat fondamental

Théorème 9 *Le (un) code de Huffman réalise la meilleure longueur moyenne parmi tous les codes préfixes (ou à décodage unique).*

En effet, soient $l_1, \ldots, l_N$ les longueurs de mots associées à un code préfixe C quelconque. Si ces longueurs ne satisfont pas les idées 1 et 2, ce code est moins bon qu'un code de Huffman C_H. Quitte à modifier les lettres dans C, on peut s'arranger pour obtenir un nouveau code préfixe C' de même longueur moyenne, et satisfaisant l'idée 3. (Cela revient à modifier la coupure de l'arbre dyadique, à longueurs de mots fixées.) En retirant la dernière lettre des deux mots les plus longs, on obtient un code C'' sur un ensemble restreint de $N - (D-1)$ symboles qui vérifie

$$L_C = L_{C'} = L_{C''} + p_{N-D+1} + \ldots + p_N$$

On peut procéder de même avec le code de Huffman, et il vient $L_C - L_{C_H} = L_{C''} - L_{C''_H}$. Si donc C est meilleur que C_H, alors C'' est meilleur que C''_H. En poursuivant pas récurrence, on montre que les longueurs l_i d'un code optimal sont celles d'un code de Huffman.

Remarques :

1. Ce résultat ne porte en fait que sur les longueurs des mots. Il ne dit pas que tous les codes optimaux sont obtenus pas la construction de la section précédente. En effet, l'étape de réaffectation des chemins de l'arbre D-adique n'y est pas prise en compte.

2. Cas particulier de ce théorème : un code de Huffman est meilleur qu'un code de Shannon. Considérons une variable X de Bernouilli, avec $p_1 = 0,9999$ et $p_2 = 0,0001$. La technique de Shannon dit qu'il faut affecter $\lceil -\log_2 p_i \rceil$ bits à α_i. Cela donne 1 bit pour α_1 et 14 bits pour α_2 ! Un code de Huffman se contentera d'un seul bit pour α_2...

3. Bien entendu, même avec un code de Huffman, on n'atteint pas l'entropie comme longueur moyenne minimale. Pour y arriver, il faut grouper les symboles à transmettre par paquets, comme on l'a vu pour le code de Shannon.

3.6 Code de Lempel-Ziv

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

bits par valeur à transmettre[7]. Ces codes procèdent par encodage de suites i.i.d. de symboles, comme on l'a vu dans la section consacrée à l'AEP, en définissant des suites typiques et atypiques. Les résultats mis en oeuvre sont un peu plus fins que l'AEP, néanmoins, puisqu'on ne connaît pas $\mathbb{P}_X$.

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

- les algorithmes de codage et de décodage sont très simples,
- elle ne demande aucune connaissance sur la source,
- elle est asymptotiquement optimale,
- elle fonctionne même avec des sources à mémoire.

Bien entendu, elle procède encore une fois par encodage de suites de valeurs produites par la source. Avant de la décrire, nous allons présenter quelques notions concernant les sources à mémoire.

3.6.1 Taux d'entropie d'une source

Pour modéliser une source X avec mémoire, on abandonne l'hypothèse que les variables $X_1, \ldots, X_n, \ldots$ sont i.i.d. On a donc affaire à un processus stochastique $\mathbf{X}$ qui n'est plus un bruit blanc, ni stationnaire *a priori*.

Définition 15 *Le* taux d'entropie *du processus* $\mathbf{X}$ *est défini par*

$$H(\mathbf{X}) = \lim_{n\to\infty} \frac{1}{n} H(X_1, \ldots, X_n)$$

*lorsque cette limite existe. L'*accroissement asymptotique d'entropie *est défini par*

$$H'(\mathbf{X}) = \lim_{n\to\infty} H(X_n | X_1, \ldots, X_{n-1})$$

lorsque cette limite existe.

Intuitivement, ces deux quantités mesurent le degré d'incertitude apporté au processus par une variable supplémentaire. Elles sont différentes en général.

Exemple. Si le processus est iid, on retrouve $H(\mathbf{X}) = H'(\mathbf{X}) = H(X)$. Toutefois, si les X_n sont seulement indépendantes, ces deux quantités ne sont pas forcément identiques, et peuvent même ne pas exister.

[7] Si on code sur D lettres, les codes demandent R lettres en moyenne pour les sources telles que $H_D(X) < R$.

Lemme 21 *Si* $\mathbf{X}$ *est un processus stationnaire (au sens strict), alors* $H(\mathbf{X})$ *et* $H'(\mathbf{X})$ *existent et sont identiques.*

La stationnarité au sens strict signifie que, pour tout k, la loi du k-uplet $(X_{n+1}, \ldots, X_{n+k})$ ne dépend pas de sa position n dans le processus $\mathbf{X}$. Il vient que la suite $u_n = H(X_n|X_1, \ldots, X_{n-1})$ converge. En effet, c'est une suite positive décroissante puisque

$$\begin{aligned} u_{n+1} &= H(X_{n+1}|X_1, \ldots, X_n) \\ &\leq H(X_{n+1}|X_2, \ldots, X_n) \\ &= H(X_n|X_1, \ldots, X_{n-1}) \\ &= u_n \end{aligned}$$

Donc $H'(\mathbf{X})$ est bien définie. Par ailleurs, en utilisant la formule des conditionnements successifs,

$$v_n \;=\; \frac{1}{n} H(X_1, \ldots, X_n) \;=\; \frac{1}{n} \sum_{k=1}^{n} H(X_k|X_1, \ldots, X_{k-1}) \;=\; \frac{1}{n} \sum_{k=1}^{n} u_k$$

v_n est donc la moyenne de Cesaro d'une suite convergente ; elle converge donc vers la même limite, ce qui prouve que $H(\mathbf{X})$ existe et vaut $H'(\mathbf{X})$.

Exemple. Dans le cas d'une chaîne de Markov stationnaire, $H(X_n|X_1, \ldots, X_{n-1}) = H(X_n|X_{n-1}) = H(X_2|X_1)$. Le taux d'entropie de la chaîne vaut donc $H(X_2|X_1)$: l'incertitude supplémentaire apportée par chaque maillon de la chaîne est constante.

3.6.2 Algorithme de Lempel et Ziv

Par souci de simplicité, nous allons décrire l'algorithme en supposant que X est une source binaire, éventuellement avec mémoire, à coder sur un alphabet binaire. La méthode s'étend directement à une source prenant N valeurs distinctes, à coder avec D lettres.

L'algorithme procède en deux passes, sur une suite finie de bits $x_1, \ldots, x_n$. La première passe décompose le train de bits en segments de sorte que chaque nouveau segment n'ait pas été rencontré précédemment. Sur la suite

$$1011010100010\ldots$$

cela donne la segmentation

$$1|0|11|01|010|00|10|\ldots$$

Il vient que chaque segment est composé d'un préfixe, correspondant à un segment déjà rencontré, plus un bit. À la fin de la première passe, on compte alors le nombre de segments, noté $c(n)$.

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

$$(000, 1)(000, 0)(001, 1)(010, 1)(100, 0)(010, 0)(001, 0)\ldots$$

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

Théorème 10 *Soit* $\mathbf{X}$ *un processus stationnaire et ergodique*[8]*, de taux d'entropie* $H(\mathbf{X})$*. Soit* $l(X_1, \ldots, X_n)$ *la longueur du mot de code de Lempel-Ziv associé à la suite* $X_1, \ldots, X_n$ *(c'est une variable aléatoire). Alors presque sûrement*

$$\limsup_{n\to\infty} \frac{1}{n} l(X_1, \ldots, X_n) \quad \leq \quad H(\mathbf{X})$$

Ce résultat signifie que le codage de Lempel-Ziv est asymptotiquement optimal : le nombre de bits par symbole de X à transmettre n'excède pas l'incertitude moyenne introduite par chaque symbole. Cet algorithme est standard pour la compression de fichiers (*compress* sous Unix, *arc* sur PC). Il réduit typiquement les fichiers ACSII d'un facteur 2. Il est aussi utilisé dans les protocoles d'échanges de fichiers.

[8] L'ergodicité d'un processus stationnaire entraîne que les moyennes empiriques $\frac{1}{n}\sum_{i=1}^{n} f(X_i)$ convergent vers les moyennes statistiques $\boldsymbol{E}[f(X)]$.

Chapter 4

Transmission de données sur un canal bruité

4.1 Introduction

Au chapitre précédent, nous nous sommes intéressés à la représentation compacte des valeurs émises par une source X à l'aide d'un ensemble de lettres $\mathcal{A}$. Le codage $C : \mathcal{X} \to \mathcal{A}^*$ consistait à représenter chaque valeur x par un mot de code $C(x)$, de sorte que la longueur moyenne du code soit minimale. On a vu qu'il fallait au-moins $H_D(X)$ lettres par valeur de X (en moyenne), et que cette borne pouvait être atteinte. Ce résultat permet de "comprimer" la source X, c'est à dire de la représenter de la manière la plus compacte, à des fins de stockage ou de transmission sur un canal sans bruit.

En comprimant ainsi la source, on a enlevé toute la redondance qu'elle contenait. On pourrait ainsi montrer que les lettres d'une séquence codée sont (ou plutôt tendent à être) indépendantes et uniformément distribuées. Il s'ensuit que chacune est indispensable à un décodage correct ; on est donc sûr d'obtenir un message erroné après décodage si l'une de ces lettres est altérée ou perdue.

Nous allons maintenant nous intéresser à la transmission fiable d'un message sur un canal *bruité*, c'est à dire susceptible d'altérer les lettres émises. Clairement, transmettre une source sans redondance conduira à des erreurs inévitables. Lutter contre les erreurs introduites par le canal va ainsi demander de *rajouter de la redondance* dans la suite de symboles émis, afin de repérer et corriger les éventuelles erreurs. Exemple de redondance : *suppasons qui vouh recevaz la mesrage suilant*, vous n'aurez aucun mal à détecter et corriger les lettres erronées en utilisant la redondance interne des mots de la langue française. D'un point de vue technique, rajouter de la redondance revient à rajouter au message des lettres supplémentaires qui sont fonction des lettres "utiles". C'est, en un sens, l'opération inverse de la compression. On verra au chapitre suivant comment construire cette information

redondante. Nous allons tout d'abord déterminer la quantité d'information redondante nécessaire à une transmission fiable, quantité qui découle des propriétés du canal de transmission. Il s'agit là de l'un des résultats fondamentaux de la théorie de l'information : le second théorème de Shannon.

4.2 Capacité de canal

4.2.1 Notations, définition d'un canal

Un système de transmission avec *codage de canal* peut se lire comme

$$\mathcal{X} \to [\text{codage}] \to \mathcal{A}^* \to [\text{canal}] \to \mathcal{B}^* \to [\text{décodage}] \to \mathcal{Y}$$

où

- $\mathcal{X}$ contient les M valeurs possibles pour la source X,
- $\mathcal{A}$ est l'alphabet d'entrée du canal, de sorte que $\mathcal{A}^*$ contient les mots du *code de canal*,
- $\mathcal{B}$ est l'alphabet de sortie du canal, qui peut être différent de $\mathcal{A}$, et donc $\mathcal{B}^*$ représente les mots de code après bruitage par le canal,
- $\mathcal{Y}$ contient les valeurs décodées de X. On a généralement $\mathcal{Y} = \mathcal{X}$, et l'on note $Y = \hat{X}$ pour indiquer que la sortie Y du système donne une estimation de l'entrée X.

Dans la suite, par souci de cohérence avec les chapitres précédents, nous allons noter A_1^n ou simplement A^n une suite de n lettres en entrée du canal, et B_1^n ou B^n les lettres de sortie. Certaines références (notamment "Information Theory" de Cover et Thomas) utilisent toutefois la notation $\mathcal{X}$ et $\mathcal{Y}$ comme alphabets d'entrée et de sortie du canal.

Définition 16 *Un canal discret est défini par le triplet $(\mathcal{A}, \mathbb{P}_{B|A}, \mathcal{B})$ où $\mathcal{A}$ est l'alphabet d'entrée, $\mathcal{B}$ l'alphabet de sortie, et $\mathbb{P}_{B|A}$ la probabilité de transition du canal. $\mathbb{P}_{B|A}$ est souvent noté sous forme d'une matrice (stochastique) de taille $|\mathcal{A}| \times |\mathcal{B}|$.*

Le canal est donc caractérisé par la distribution des lettres de sortie pour chaque lettre émise a. Un canal parfait correspond ainsi à $\mathbb{P}_{B|A} = \mathbb{1}_{B=A}$. On considérera dans la suite des canaux discrets *sans mémoire*, c'est à dire perturbant les lettres de façon indépendante[1] :

$$p(b_1^n|a_1^n) = \prod_{k=1}^{n} p(b_k|a_k)$$

[1] En supposant en outre qu'ils sont stationnaires, c'est à dire que la probabilité de transition par lettre qui les caractérise ne varie pas d'une lettre à l'autre.

Exemple. La figure 4.1 représente un canal binaire : $\mathcal{A} = \mathcal{B} = \{0, 1\}$. p désigne la probabilité de recevoir 1 lorsque 0 a été émis, et $1 - p$ est donc la probabilité de transmission correcte du 0. Symétriquement pour q lorsque 1 est émis. Un tel canal binaire est dit symétrique si $p = q$, et p est alors appelé probabilité d'erreur du canal.

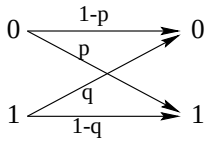

Figure 4.1: Canal binaire.

4.2.2 Capacité d'un canal

Définition 17 *La* capacité C *d'un canal discret sans mémoire est donnée par*

$$C = \max_{\mathbb{P}_A} I(A; B)$$

Le max est donc pris sur toutes les distributions possibles $\mathbb{P}_A$ de l'entrée A. Comme le canal est défini par $\mathbb{P}_{B|A}$, on dispose bien de la loi jointe $\mathbb{P}_{A,B}$ qui permet de calculer l'information mutuelle $I(A; B)$. Nous allons voir dans ce chapitre que *la capacité C représente le nombre moyen de bits que l'on peut transmettre sans erreur par utilisation du canal* (une utilisation est l'envoi d'une lettre).

Propriétés

1. $C \geq 0$ car $I(A; B) \geq 0$
2. $C \leq \log_2 |\mathcal{A}|$ car $C \leq \max H(A)$, obtenu pour la loi uniforme,
3. $C \leq \log_2 |\mathcal{B}|$, même argument
4. C existe toujours et est unique, car $I(A; B)$ est une fonction continue concave de $\mathbb{P}_A$; cela ne signifie pas toutefois que C est facile à calculer analytiquement, mais elle est facilement accessible par optimisation numérique.

4.2.3 Exemples

Canal parfait

Le canal parfait (fig. 4.2) sur $\mathcal{A} = \mathcal{B} = \{0, 1, \ldots, D - 1\}$ a pour probabilité de transition $\mathbb{P}_{B|A} = \mathbb{1}_{B=A}$. Il vient $I(A; B) = H(B) - H(B|A) = H(B) = H(A)$ qui est donc maximale pour la loi uniforme, d'où $C = \log_2 D$. Le canal parfait binaire transmet ainsi sans erreur 1 bit par utilisation. Normal.

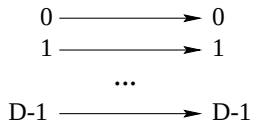

Figure 4.2: Canal parfait D-aire.

Canal sans interférence

C'est un cas (fig. 4.3) où l'alphabet de sortie diffère de l'alphabet d'entrée. On a $I(A;B) = H(A) - H(A|B)$, or $H(A|B) = 0$ puisque la sortie permet de retrouver exactement l'entrée, d'où $C = \max H(A) = 1$.

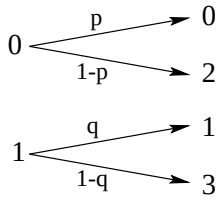

Figure 4.3: Canal sans interférence.

Machine à écrire bruitée

Ce canal (fig. 4.4) travaille sur $\mathcal{A} = \mathcal{B} = \{a, b, c, d, \ldots, z\}$. La lettre émise est soit reproduite fidèlement, soit changée en la lettre suivante de l'alphabet (modulo 26), avec une probabilité 1/2. Il vient donc $H(B|A) = 1$. D'où $I(A;B) = H(B) - H(B|A) \leq \log_2(26) - 1 = \log_2(13)$. Donc $C \leq \log_2(13)$. Cette borne peut en fait être atteinte en n'utilisant qu'une lettre sur deux à l'émission, i.e. $\{a, c, e, \ldots, y\}$ par exemple, ce qui nous ramène dans le cas du canal sans interférence.

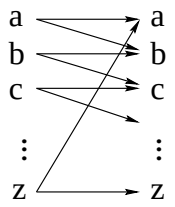

Figure 4.4: La machine à écrire bruitée.

Canal binaire symétrique

Voir figure 4.1 avec $p = q$. On a

$$I(A;B) = H(B) - H(B|A) = H(B) - \sum_a p(a) H(B|A=a)$$

Or $H(B|A=a)$ ne dépend pas de a et vaut $H(p)$, entropie d'une variable de Bernouilli de paramètre p. Donc $I(A;B) = H(B) - H(p) \leq 1 - H(p)$. Cette borne est atteinte lorsque A a la loi uniforme : B est alors aussi de loi uniforme.

Canal binaire à effacement

Ce canal (fig. 4.5) suppose $\mathcal{A} = \{0,1\}$ et $\mathcal{B} = \{0,1,\varepsilon\}$; il efface la lettre émise avec une probabilité p.

$I(A;B) = H(B) - H(B|A)$, et l'on a clairement $H(B|A) = H(p)$ (voir cas précédent). Soit E l'indicateur d'effacement, qui vaut 1 lorsque l'on a reçu ε et 0 sinon. Comme E se déduit de B, il vient

$$\begin{aligned} H(B) &= H(B,E) \\ &= H(E) + H(B|E) \\ &= H(p) + p \cdot \underbrace{H(B|E=1)}_{0} + (1-p) \cdot \underbrace{H(B|E=0)}_{H(A)} \\ &= H(p) + (1-p)H(A) \end{aligned}$$

Il vient donc $I(A;B) = (1-p)H(A) \leq 1-p$. La borne est atteinte avec la loi uniforme sur l'entrée A, d'où $C = 1-p$.

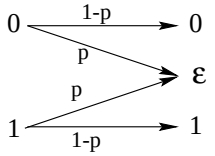

Figure 4.5: Canal binaire à effacement.

Ce résultat est assez intuitif : pour effectivement transmettre un message sans erreur sur ce canal, il suffit d'adopter un protocole simple de feedback : lorsqu'une lettre a été effacée, on demande sa retransmission. On transmet ainsi correctement une proportion $1-p$ des bits, soit $1-p$ bit par utilisation du canal. La capacité du canal est donc bien le nombre de bits utiles que l'on peut transmettre par utilisation du canal. On verra plus loin que l'on peut atteindre les mêmes performances sans utiliser de feedback (la capacité d'un canal n'est pas augmentée si l'on ajoute une voie de feedback).

On observe aussi sur cet exemple que $C = 1 - p$ est le meilleur rendement d'un code fiable (c'est à dire corrigeant toutes les erreurs) pour ce canal. En effet, pour k bits utiles à transmettre, il faudra utiliser $n = k/(1-p)$ fois le canal en moyenne, d'où $R = 1 - p$. Le théorème de Shannon généralisera ce résultat.

Canal symétrique

C'est une généralisation du canal binaire symétrique.

Définition 18 *Un canal de transmission est* symétrique *ssi les colonnes (comme les lignes) de sa matrice de transition $\mathbb{P}_{B|A}$ s'obtiennent par permutation circulaire.*

Exemple. Sur des alphabets de 3 lettres

$$\mathbb{P}_{B|A} = \begin{pmatrix} 0,3 & 0,2 & 0,5 \\ 0,5 & 0,3 & 0,2 \\ 0,2 & 0,5 & 0,3 \end{pmatrix}$$

Exemple. Un cas typique de canal symétrique est la situation $\mathcal{A} = \mathcal{B} = \{0, 1, \ldots, D-1\}$, où $\mathbb{P}_{B|A}$ est définie par $B = A + Z$ modulo D. La loi de Z détermine alors la première ligne de la matrice $\mathbb{P}_{B|A}$. Ci-dessus, on a $Z = 0, 1$ ou 2 avec les vraisemblances $(0,3 \;\; 0,2 \;\; 0,5)$.

La capacité d'un tel canal est donnée par $C = \log_2 |\mathcal{B}| - H(Z)$ (**exercice**).

4.2.4 Classification générale des canaux

Généralement, un *canal de transmission* est défini globalement pour des mots de taille n, par une probabilité de transition $\mathbb{P}_{B^n|A^n}$. Il est donc caractérisé par la distribution sur les mots de sortie pour chaque mot d'entrée a_1^n. Un canal parfait correspond ainsi à $\mathbb{P}_{B^n|A^n} = \mathbb{1}_{B^n=A^n}$. Par la règle de Bayes, la loi de transformation peut s'écrire[2] :

$$p(b_1^n|a_1^n) = \prod_{k=1}^{n} p(b_k|a_1^n, b_1^{k-1})$$

On dit que le canal est *causal* si chaque lettre d'entrée a_k provoque l'émission d'une lettre b_k en sortie. b_k ne dépend donc pas de a_{k+1}^n. D'où

$$p(b_k|a_1^n, b_1^{k-1}) = p(b_k|a_1^k, b_1^{k-1})$$

Le canal est dit *sans feedback* si la lettre de sortie b_k ne dépend que des entrées, et pas des sorties précédentes, soit

$$p(b_k|a_1^n, b_1^{k-1}) = p(b_k|a_1^n)$$

[2]Dans les écritures qui suivent, les minuscules $(a, b, \ldots)$ représentent des valeurs (de $A, B, \ldots$), et l'on suppose que les équations valent pour toutes les valeurs possibles.

Ainsi, un canal causal sans feedback vérifie

$$p(b_k|a_1^n, b_1^{k-1}) = p(b_k|a_1^k)$$

Le canal sans mémoire est ainsi un cas particulier de canal causal sans feedback.

4.3 Théorème de Shannon

4.3.1 Définitions préliminaires

Définition 19 *Un (M, n)-code pour le canal $(\mathcal{A}, \mathbb{P}_{B|A}, \mathcal{B})$ est constitué de*

1. *un ensemble d'indices $\mathcal{X} = \{1, 2, \ldots, M\}$ représentant les M valeurs de la source X à transmettre,*
2. *une fonction de codage notée $f : \{1, 2, \ldots, M\} \to \mathcal{A}^n$; on appelle alors $f(1), f(2), \ldots, f(M)$ les mots du code, ou le codebook,*
3. *une fonction de décodage $g : \mathcal{B}^n \to \{1, 2, \ldots, M\}$.*

M est donc le nombre de symboles à transmettre et n la longueur des mots (constante ici, contrairement au cas du codage de source). Noter que si la fonction de codage f doit être injective, ce n'est pas le cas de la fonction de décodage g.

Définition 20 *Le* taux *ou* rendement *R d'un (M, n)-code est le rapport*

$$R \triangleq \frac{\log_2 M}{n}$$

Il représente le nombre de bits transmis par utilisation du canal, ou par lettre envoyée sur le canal[3]*.*

Exemple. Pour transmettre les 26 lettres de l'alphabet plus l'espace, soit $M = 27$ symboles, si on utilise un codage sur 5 bits, le rendement est $R = \log_2(27)/5 = \log_2 \frac{27}{32}$. Si on code sur 8 bits, $R = \log_2(27)/8$ qui est moins efficace. Si on code avec l'alphabet $\mathcal{A} = \{\alpha, \beta, \gamma\}$, il suffit de $n = 3$ lettres de $\mathcal{A}$ pour représenter les $M = 27$ symboles. D'où $R = \log_2(27)/3 = \log_2(3) > 1$ bits par lettre de $\mathcal{A}$ émise sur le canal. (Mesuré en symboles ternaires, on aurait dans ce dernier cas $R = 1$ lettre utile par lettre émise.)

[3] Avec cette définition, on peut avoir $R > 1$. Si on prenait comme définition $R = \frac{\log_D M}{n}$, avec $D = |\mathcal{A}|$, alors R serait compris entre 0 et 1 et se mesurerait en "lettre utile transmise par lettre émise".

Définition 21 *Pour un (M,n)-code (f,g) sur $(\mathcal{A}, I\!P_{B|A}, \mathcal{B})$, la probabilité conditionnelle d'erreur pour le symbole i est définie par*

$$\lambda_i \;=\; I\!P[g(B^n) \neq i | A^n = f(i)] \;=\; \sum_{b^n} I\!P[b^n | A^n = f(i)] \cdot I\!\!I_{g(b^n)\neq i}$$

La probabilité maximale d'erreur du code est définie par

$$\lambda^{(n)}_{\max} \;=\; \max_{1\leq i\leq M} \;\lambda_i$$

Définition 22 *Pour un canal donné $(\mathcal{A}, I\!P_{B|A}, \mathcal{B})$, le taux R est* réalisable *ssi il existe une suite de $\big(\lceil 2^{nR}\rceil, n\big)$-codes telle que la probabilité d'erreur $\lambda^{(n)}_{\max} \xrightarrow[n\to\infty]{} 0$.*

Cela signifie que, quitte à envoyer de longues séquences de (n) lettres, alors on peut transmettre R bits par lettre avec une erreur aussi petite que l'on veut. On retrouve des considérations assez proches de l'AEP, et c'est en fait l'outil principal des résultats suivants.

4.3.2 Deuxième théorème de Shannon

Théorème 11 (2^{eme} théorème de Shannon) *Soit un canal discret sans mémoire de capacité C, alors*

- **CS :** *tout taux de code R tel que $R < C$ est réalisable,*
- **CN :** *inversement, si une suite de $\big(\lceil 2^{nR}\rceil, n\big)$-codes est asymptotiquement sans erreur (i.e. $\lambda^{(n)}_{\max} \xrightarrow[n\to\infty]{} 0$), alors $R \leq C$.*

La capacité du canal est donc bien nommée : on ne pourra émettre sans erreur que si on choisit un rendement inférieur à la capacité. Cela caractérise donc la redondance minimale que doit contenir le code pour corriger les erreurs introduites par le canal. Si $\mathcal{A} = \{0,1\}$, pour transmettre k bits utiles ($M = 2^k$) sans erreur, il faut "étaler" l'information sur au-moins $n = k/C$ bits, avec $C < 1$ ici. Ce résultat est vrai asymptotiquement bien entendu, i.e. pour $k, n \to \infty$. La construction des $n-k$ bits redondants est une question délicate, c'est le problème central des codes correcteurs d'erreurs. On verra dans les chapitres suivants que la solution simple consistant à répéter des bits utiles est en fait assez mauvaise... On verra section 4.3.5 à quoi doivent ressembler les bons codes correcteurs.

La condition nécessaire n'est pas très difficile à montrer ; elle fait l'objet de la section suivante. En revanche, la condition suffisante est plus compliquée car il s'agit de construire une suite de codes dont la probabilité d'erreur tende vers 0. Cette construction repose sur la notion de séquence typique ; nous n'en donnerons que les grandes lignes.

4.3.3 Preuve de la CN

On dispose d'une suite de $\big(\lceil 2^{nR}\rceil, n\big)$-codes telle que $\lambda_{\max}^{(n)} \xrightarrow[n\to\infty]{} 0$ pour un canal discret sans mémoire $(\mathcal{A}, I\!P_{B|A}, \mathcal{B})$; il s'agit de montrer $R \leq C$.

Pour n fixé, considérons le canal global transmettant un paquet de n lettres. Il a en entrée une source X prenant $M = 2^{nR}$ valeurs, $\mathcal{X} = \{v_1, v_2, \ldots, v_M\}$, et la sortie Y après décodage donne une estimée $\hat{X}$ de l'entrée.

$$X \xrightarrow{f^{(n)}} A^n \xrightarrow{canal} B^n \xrightarrow{g^{(n)}} \hat{X}$$

$f^{(n)}$ et $g^{(n)}$ représentent respectivement les fonctions de codage et de décodage du n^{eme} code. En donnant une loi $I\!P_X$ à la source, on peut calculer la probabilité d'erreur $P_e^{(n)} \triangleq I\!P(\hat{X} \neq X)$ de ce système :

$$\begin{aligned} \lambda_i^{(n)} &= I\!P(\hat{X} \neq v_i | X = v_i) \\ P_e^{(n)} &= \sum_{i=1}^{2^{nR}} I\!P_X(v_i) \cdot \lambda_i^{(n)} \;\leq\; \lambda_{\max}^{(n)} \end{aligned}$$

Il s'ensuit que la probabilité d'erreur $P_e^{(n)}$ tend vers zéro.

Lemme 22 *Soit $(\mathcal{A}, I\!P_{B|A}, \mathcal{B})$ un canal discret sans mémoire de capacité C. Le canal $(\mathcal{A}^n, I\!P_{B^n|A^n}, \mathcal{B}^n)$ obtenu par extension à des paquets de n lettres est de capacité nC.*

Il s'agit de montrer

$$\max_{I\!P_{A^n}} I(A^n; B^n) \;=\; n \cdot \max_{I\!P_A} I(A; B)$$

Le canal étant sans mémoire, il vient

$$I\!P_{B^n|A^n} \;=\; \prod_{i=1}^{n} I\!P_{B_i|A_i} \qquad \text{d'où} \qquad H(B^n|A^n) \;=\; \sum_{i=1}^{n} H(B_i|A_i)$$

Il s'ensuit

$$\begin{aligned} I(A^n; B^n) &= H(B^n) - H(B^n|A^n) \\ &= H(B^n) - \sum_{i=1}^{n} H(B_i|A_i) \\ &\leq \sum_{i=1}^{n} H(B_i) - \sum_{i=1}^{n} H(B_i|A_i) \\ &= \sum_{i=1}^{n} I(A_i\,; B_i) \\ &\leq nC \end{aligned}$$

Pour montrer que la borne peut être atteinte, il suffit de supposer les A_i indépendants, ce qui remplace la première inégalité par une égalité. Si en plus on les prend iid de loi $I\!P_A^*$ avec $I\!P_A^* = \arg\max_{I\!P_A} I(A;B)$, la seconde inégalité devient elle aussi égalité.

Remarque. Ce résultat montre en particulier qu'en utilisant n canaux identiques en parallèle on multiplie par n la capacité. Plus généralement, les capacités s'ajoutent sur des canaux parallèles (même preuve). On n'utilisera ci-dessous que la propriété $I(A^n;B^n) \leq nC$.

Preuve de la CN

Soit $E = I\!1_{\hat{X}\neq X}$ la variable d'erreur du code, qui vaut 1 lorsque le décodage est incorrect et 0 sinon. Il vient $I\!P(E=1) = P_e^{(n)}$. On observe aussi que l'on a une chaîne de Markov $X \to A^n \to B^n \to \hat{X}$. Supposons que X est muni de la loi uniforme sur ses $M = 2^{nR}$ valeurs. Il vient

$$\underbrace{H(X)}_{=nR} = H(X|B^n) + I(X;B^n) \tag{4.1}$$

Nous allons majorer chacun des termes de droite dans (4.1). Pour le second, en utilisant le théorème sur le traitement de l'information dans la chaîne $X \to A^n \to B^n$ et le lemme précédent, on a

$$I(X;B^n) \leq I(A^n;B^n) \leq nC \tag{4.2}$$

Pour le premier terme, on a

$$\begin{aligned} H(X,E|B^n) &= \overbrace{H(E|B^n,X)}^{0} + H(X|B^n) \\ &= H(X|E,B^n) + \underbrace{H(E|B^n)}_{\leq H(E)\leq 1} \end{aligned}$$

on en déduit $H(X|B^n) \leq H(X|E,B^n) + 1$. Reste à majorer $H(X|E,B^n)$, or

$$H(X|E,B^n) = I\!P(E=0)\cdot \underbrace{H(X|E=0,B^n)}_{0} + I\!P(E=1)\cdot \underbrace{H(X|E=0,B^n)}_{\leq H(X)=nR}$$

le premier terme est nul car lorsqu'il n'y a pas d'erreur X est une fonction de B^n puisque $X = \hat{X} = g^{(n)}(B^n)$; on en tire

$$H(X|B^n) \leq 1 + H(X|E,B^n) \leq 1 + P_e^{(n)} nR \tag{4.3}$$

En reportant les majorations (4.2) et (4.3) dans (4.1), il vient

$$\begin{aligned} nR &\leq 1 + P_e^{(n)} nR + nC \qquad (4.4) \\ \text{soit} \qquad R &\leq C + R\,P_e^{(n)} + \frac{1}{n} \end{aligned}$$

et il suffit de faire tendre n vers ∞ pour en déduire $R \leq C$.

4.3.4 Autre conséquence de la CN

En transformant (4.4) on obtient une minoration de la probabilité d'erreur

$$P_e^{(n)} \;\geq\; 1 - \frac{C}{R} - \frac{1}{nR}$$

Lorsque $R > C$, $1 - C/R > 0$ et il existe un n_0 au-delà duquel $P_e^{(n)} \geq 1 - \frac{C}{R} - \varepsilon > 0$. Donc, les longs codes font forcément des erreurs, i.e. on ne peut atteindre une probabilité d'erreur arbitrairement petite. C'est en fait vrais pour *tous* les codes car s'il existait des codes courts atteignant une probabilité d'erreur arbitraitement petite, il suffirait de les concaténer pour former des codes longs fiables.

Au bilan, la probabilité d'erreur se comporte donc comme sur la figure 4.6.

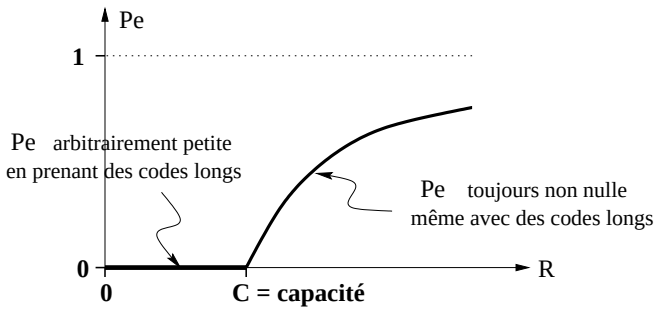

Figure 4.6: Meilleure probabilité d'erreur atteignable en fonction du rendement du code de canal.

4.3.5 Transmission sans erreur

On cherche à savoir à quoi doivent ressembler les codes "parfaits" permettant une transmission sans erreur avec $R = C$. Bien entendu il est hors de question de les caractériser complètement ici, mais on peut néanmoins exhiber quelques propriétés. Pour cela nous revenons sur la preuve du lemme 22.

Avec un code est parfait, le décodage donne $\hat{X} = X$; on a donc la chaîne de Markov

$$X \xrightarrow{f^{(n)}} A^n \xrightarrow{canal} B^n \xrightarrow{g^{(n)}} \hat{X} = X \xrightarrow{f^{(n)}} A^n \xrightarrow{canal} \ldots$$

de laquelle on tire

$$H(X) \;=\; I(X;X) \;=\; I(X;\hat{X}) \;=\; I(A^n;B^n)$$

par applications successives du théorème sur le traitement de l'information (**exercice**). Supposons que X est muni de la loi uniforme, soit $H(X) = nR$. On a montré lors de la preuve du lemme 22 que $I(A^n;B^n) \leq nC$, donc

$$nR \;=\; H(X) \;=\; I(A^n;B^n) \leq nC$$

Mais pour un code parfait $R = C$, il s'ensuit que $I(A^n; B^n) = nC$, c'est à dire que l'on atteint la capacité du canal "par blocs" du lemme 22. Comme on l'a vu, ceci n'est possible que si les A_i sont iid de loi $\mathbb{P}_A^* = \arg\max_{\mathbb{P}_A} I(A; B)$.

Une bonne fonction de codage $f^{(n)}$ doit ainsi s'arranger pour que les lettres d'entrée du canal A_i soient indépendantes – ce qui garantit que les lettres de sortie B_i le sont aussi. Cela permet trois choses

- l'information de X est "étalée" de façon très uniforme puisque chaque lettre est d'entropie R ; dans le cas où $\mathcal{A} = \{0, 1\}$, cela signifie que chaque bit transmis transporte $R < 1$ bit utile,
- comme chaque A_i transmis obéit à $\mathbb{P}_A^*$, il utilise au mieux le canal, c'est à dire exploite les lettres de $\mathcal{A}$ les moins bruitées par le canal,
- enfin comme les lettres d'entrée sont indépendantes, le bruit introduit sur l'une d'entre elles par le canal ne perturbe pas les autres, i.e. il suffit de B_i pour estimer A_i.

4.3.6 Une idée de la preuve de la CS

À compléter...

4.4 Connexion d'une source à un canal

4.4.1 Le problème

On dispose d'une source X qui peut prendre M valeurs, $\mathcal{X} = \{v_1, v_2, \ldots, v_M\}$, de loi $\mathbb{P}_X$ et d'entropie $H(X)$. On dispose aussi d'un canal de transmission $(\mathcal{A}, \mathbb{P}_{B|A}, \mathcal{B})$ de capacité C. On souhaite transmettre X sans erreur sur ce canal, quitte à transmettre de longues séquences $X_1, X_2, \ldots, X_n$ de valeurs : i.e. il s'agit d'avoir $P_e^{(n)} \to 0$. On suppose que l'on utilise le canal au plus une fois par valeur de X à transmettre[4], en moyenne.

Question : quelle est la capacité de canal minimale nécessaire pour que cette transmission fiable soit possible ?

On dispose de deux solution pour cette transmission.

1. On peut tenter de construire directement une suite de (M^n, n)-codes dont la probabilité d'erreur tende vers 0.

$$X^n \xrightarrow{f^{(n)}} A^n \xrightarrow[C]{canal} B^n \xrightarrow{g^{(n)}} \hat{X}^n$$

[4]Ce n'est pas vraiment une contrainte car on peut considérer que le canal émet des paquets de k lettres, comme vu au lemme 22, lorsque $|\mathcal{A}| \leq M$.

2. On peut aussi procéder en deux étapes. D'abord *comprimer* la source X, sur un alphabet intermédiaire quelconque, puis coder et transmettre le résultat de la compression.

$$X^n \xrightarrow{compression} Z^k \xrightarrow{f'^{(n)}} A^n \xrightarrow[C]{canal} B^n \xrightarrow{g'^{(n)}} \hat{Z}^k \xrightarrow{decompression} \hat{X}^n$$

Nous allons montrer que la seconde technique, qui est un cas particulier de la première, permet d'atteindre les mêmes performances. En particulier, on va voir que l'on peut émettre X sans erreur sur le canal – toujours à raison d'au-plus une utilisation du canal par valeur transmise – ssi $H(X) < C$. Cela est assez intuitif puisque la source produit $H(X)$ bits par symbole, et le canal peut transporter C bits sans erreur par utilisation.

Ce résultat, connu sous le nom de *théorème de séparation*, est très important en pratique : il signifie que l'on peut concevoir un système de transmission pour X en deux étapes, en se préoccupant séparément du *codage de source* (retrait de l'information redondante) et du *codage de canal* (ajout d'une redondance adaptée au canal, pour lutter contre les erreurs).

4.4.2 Théorème de séparation

Théorème 12 (dû à Shannon) *Soient X une source d'information d'entropie $H(X)$, à valeurs dans $\mathcal{X}$, $|\mathcal{X}| = M$, et $(\mathcal{A}, I\!P_{B|A}, \mathcal{B})$ un canal de capacité C.*

- **CN.** *S'il existe une suite de (M^n, n)-codes pour X telle que $P_e^{(n)} \to 0$, alors $H(X) \leq C$.*
- **CS.** *Si $H(X) < C$, alors on peut transmettre X (asymptotiquement) sans erreur sur le canal, à raison d'une émission par valeur de X. Cette transmission peut se faire par codage de source + codage de canal.*

CN. On va supposer que l'on dispose d'une suite de (M^n, n)-codes asymptotiquement sans erreur (de la forme 1 ou 2), et montrer que ce n'est possible que si $H(X) \leq C$.

On procède exactement comme pour la CN du deuxième théorème de Shannon. Soit la variable d'erreur $E = I\!1_{\hat{X}^n \neq X^n}$, alors

$$\begin{aligned} H(X^n|\hat{X}^n) &= H(E, X^n|\hat{X}^n) \\ &= H(E|\hat{X}^n) + I\!P(E=0)H(X^n|\hat{X}^n, E=0) \\ &\quad + I\!P(E=1)H(X^n|\hat{X}^n, E=1) \\ &\leq H(E) + I\!P(E=1)H(X^n) \\ &\leq 1 + P_e^{(n)} n \log_2 |\mathcal{X}| \end{aligned} \tag{4.5}$$

qui est en fait l'*inégalité de Fano* que nous avons déjà vue. On a ensuite, par le théorème du traitement de l'information,

$$I(X^n; \hat{X}^n) \;\leq\; I(A^n; B^n) \;\leq\; nC \tag{4.6}$$

On injecte maintenant (4.5) et (4.6) dans l'expression de $H(X^n)$ pour obtenir

$$\begin{aligned} H(X^n) &= H(X^n|\hat{X}^n) + I(X^n; \hat{X}^n) \\ nH(X) &\leq 1 + P_e^{(n)} n \log_2 |\mathcal{X}| + nC \\ H(X) &\leq C + P_e^{(n)} \log_2 |\mathcal{X}| + \frac{1}{n} \end{aligned}$$

et il suffit de passer à la limite pour conclure.

CS. En supposant $H(X) < C$, on va construire une fonction de codage de X^n vers A^n et montrer qu'elle est asymptotiquement sans erreur. On procède par la seconde technique : codage de source + codage de canal.

Codage de source. On utilise l'AEP (pas Shannon, ni Huffman). Pour n grand, on sépare les séquences $x_1, \ldots, x_n$ en typiques et atypiques. On peut s'arranger pour avoir $\mathbb{P}[A_\varepsilon^{(n)}] > 1 - \varepsilon'$. Seules les séquences typiques seront transmises, les atypiques donneront lieu à une erreur de transmission :

$$P_{e_1}^{(n)} \;<\; \varepsilon'$$

Il y a $2^{n(H(X)+\varepsilon)}$ séquences (équiprobables) dans $A_\varepsilon^{(n)}$, il suffit donc de nR bits pour les coder, avec $R = H(X) + \varepsilon$. Tout se passe donc comme si on avait une nouvelle source émettant 2^{nR} symboles équiprobables.

Codage de canal. Quitte à prendre n plus grand encore, on sait que l'on peut émettre 2^{nR} symboles en n utilisations du canal dès lors que $R < C$, avec une probabilité d'erreur

$$P_{e_2}^{(n)} \;<\; \varepsilon''$$

Bilan. Les deux codages mis bout à bout donnent une probabilité d'erreur totale $P_e^{(n)} < \varepsilon' + \varepsilon''$ que l'on peut rendre arbitrairement petite. Cela montre qu'une technique de codage en deux étapes permet d'atteindre la performance optimale d'un système de codage.

Remarques.

- Le théorème précédent est en fait valable dès que l'AEP marche. En particulier, il fonctionne pour des sources à mémoire stationnaires et ergodiques, en remplaçant l'entropie $H(X)$ par un taux d'entropie $H(\mathbf{X})$.

- On n'a pas toujours intérêt à suivre les deux étapes codage de source + codage de canal, puisqu'en pratique cela revient à retirer de la redondance pour en remettre... Il arrive que la source contienne naturellement une redondance qui la rend résistante au bruit introduit par le canal. Penser par exemple à de la parole, qui reste compréhensible même en présence d'un très fort bruit blanc (le décodeur est bien fait...). Ainsi, même si théoriquement on n'y perd pas avec deux codages, il peut être plus simple de tout faire directement. On parle alors de *codage conjoint source-canal*, qui est un sujet de recherche intense aujourd'hui.

4.5 Capacité d'un canal avec feedback

À compléter...

Chapter 5

Introduction aux codes correcteurs

5.1 Introduction

Le chapitre précédent a montré que la transmission sans erreur sur un canal bruité était possible, pour autant que certaines contraintes de "débit" d'information utile en entrée du canal étaient respectées. Ainsi a-t-on vu que le *rendement* R d'un code de canal ne doit pas dépasser la capacité C du canal. De même, peut-on transmettre sans erreur une source X sur un canal – à raison d'une utilisation du canal par valeur à transmettre – seulement si $H(X) < C$, c'est à dire si le débit d'information de la source ne dépasse pas celui que l'on peut transmettre sans erreur sur le canal.

Ces résultats nous renseignent sur la *quantité* minimale de redondance que doit contenir un code de canal, mais ne disent pas précisément comment construire un tel code. Les constructions vues jusque là, fondées sur l'AEP, donnent en fait des codes *très peu structurés*, codables et décodables seulement par comparaison avec un codebook. Leur complexité de mise en oeuvre les rend inapplicables en pratique : en effet, ces codes sont *asymptotiquement* performants, ils supposent donc des codebooks construits pour de très longues séquences d'entrée. Le stockage de tables de codage/décodage aussi grandes est impossible (sans parler de la difficulté de retrouver un mot dans la table pour le décodage). Il s'agit donc de trouver des codes tout aussi efficaces mais offrant une *structure interne riche*, permettant des opérations de codage/décodage par *calcul* plutôt que par comparaison avec une table. C'est la question que nous examinons dans ce chapitre.

Un code de canal était vu comme une paire (f, g) de fonctions de codage/décodage entre un ensemble $\mathcal{X}$ de M symboles et un ensemble $\mathcal{A}^n$ de mots de n lettres sur l'alphabet $\mathcal{A}$. Sans perte de généralité, on peut voir $\mathcal{X}$ comme $\mathcal{A}^k$, $k < n$. Cette séquence de taille k peut en particulier résulter d'une compression (voir le théorème

de séparation), et k est alors égal[1] à $H_D(X)$, où $D = |\mathcal{A}|$. Les k lettres d'entrée sont dites *lettres utiles* car elles portent l'information à transmettre. Elles se retrouvent "diluées" sur n lettres par la fonction de codage. Le rendement est ainsi $R = \frac{k}{n} \log_2 D$, en bits par lettre émise. L'image de f est appelée le *code*, ou *codebook* ou *ensemble des mots du code*. Dans la suite, on ne distinguera pas les fonctions de codage/décodage identiques à une bijection près : $(f, g) \equiv (f \circ \phi, \phi^{-1} \circ g)$ où ϕ est bijective sur $\mathcal{A}^k$. On s'intéressera donc principalement aux propriétés du *code* lui-même, dont on verra que la structure interne guide en fait les méthodes de codage et surtout de décodage.

Dans ce chapitre, $\mathcal{A}$ ne sera pas un alphabet quelconque mais un *corps fini* $\mathbb{F}_q$. Le plus souvent, on prendra $\mathcal{A} = \mathbb{F}_2 = \{0, 1\}$ ou les extensions de $\mathbb{F}_2$: $\mathcal{A} = (\mathbb{F}_2)^m$. On supposera aussi $\mathcal{B} = \mathcal{A}$, pour simplifier.

5.2 Codage et décodage : généralités

5.2.1 Code bloc

On va s'intéresser tout d'abord à des *codes blocs*, transformant des mots de k lettres en mots de $n \geq k$ lettres. On les utilise en tronçonnant le train de lettres entrant dans le codeur en paquets de k lettres, chacun étant codé séparément.

Définition 23 *Un* code *est un sous-ensemble de $\mathcal{A}^n$, noté $\mathcal{C}$ génériquement, où $\mathcal{A}$ est l'*alphabet *du code, et n la* longueur *du code (i.e. le nb de lettres de chaque mot). $|\mathcal{C}|$ est la* taille *du code. Si $|\mathcal{C}| = D^k$ avec $D = |\mathcal{A}|$, on dit que $\mathcal{C}$ est de* paramètres[2] *(n, k). On a toujours $k \leq n$. Le* rendement *du code est défini par $R = \frac{k}{n} \log_2 D$ bits par lettre émise.*

Remarque. Le rendement mesure la proportion de lettres utiles transmises dans un mot de code. Si on le mesure donc en "lettre utile par lettre émise", il vaut k/n. Comme chaque lettre correspond à $\log_2 D$ bits, mesuré en bits utiles par lettre émise, on retrouve la formule de la définition.

Définition 24 *Un code (n, k) est dit* systématique *ssi il laisse intactes les k lettres d'entrée et se contente de rajouter $n - k$ lettres supplémentaires, appelées* lettres redondantes *ou* parity checks.

Les lettres supplémentaires sont fonction des lettres utiles et permettent de vérifier la cohérence du message.

[1] Ce raisonnement schématique est un peu abusif et ne sert qu'à fixer les idées ; il devient plus rigoureux si l'on considère de longues séquences de valeurs de X.

[2] Attention, ne pas confondre un (M, n)-*code*, comme défini dans le chapitre sur la capacité de canal, et un code de *paramètres* (n, k). Un (M, n)-code a pour paramètres $(n, \log_D M)$. Inversement, un code de paramètres (n, k) constitue un (D^k, n)-code.

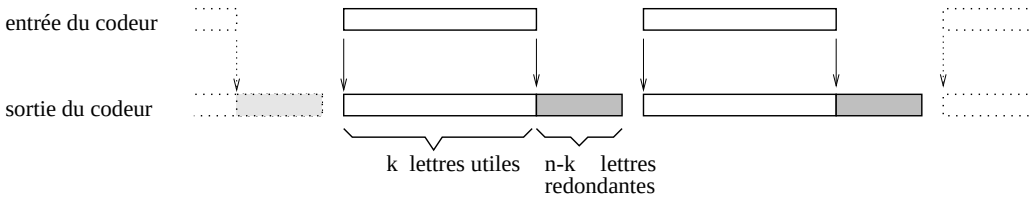

Figure 5.1: Structure d'un code systématique.

Exemple. Le traditionnel bit de parité : sur un octet, le huitième bit est la somme des 7 précédents. Cela donne un code $(8, 7)$ (binaire), qui permet de *détecter* une erreur, c'est à dire un bit changé, mais pas de la *corriger*.

Exemple. Le code à répétition, envoie n fois le même bit. C'est donc un code $(n, 1)$, qui permet de détecter jusqu'à $n-1$ erreurs, et d'en *corriger* $\frac{n-1}{2}$ si n est impair, par décision à la majorité. Le code à répétition offre un bon pouvoir de correction, mais c'est au prix d'un rendement très faible. On verra de bien meilleurs codes plus loin.

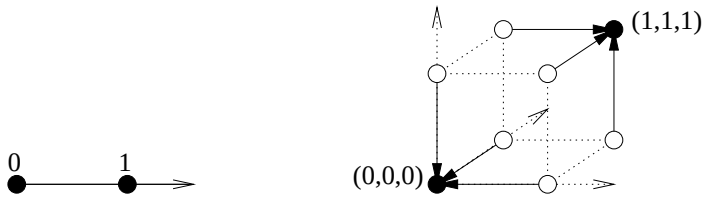

Figure 5.2: Code à répétition $(3, 1)$. Ce code permet de corriger 1 bit erroné : les flèches indiquent quel mot doit être décodé en fonction de l'élément reçu. Si deux bits sont erronés, une erreur de décodage se produit.

5.2.2 Distance de Hamming

En plongeant les mots de $\mathcal{A}^k$ dans un espace plus grand, $\mathcal{A}^n$, on peut s'arranger pour "espacer" davantage les éléments du code. On le voit bien sur le code à répétition $(3, 1)$, qui représente les bits à transmettre par les extrémités de la diagonale du cube unité. C'est en fait toute l'idée des codes correcteurs d'erreurs : plus les mots sont éloignés dans $\mathcal{A}^n$, moins le bruit du canal conduit à les confondre. Le décodage *au plus proche voisin* donne alors de bons résultats. Avant de détailler, commençons par introduire une notion de distance.

Définition 25 *Le* poids de Hamming *d'un mot de code m, noté $w(m)$, est le nombre de lettres non nulles dans m. La* distance de Hamming *entre les mots de code m_1 et m_2, notée $d(m_1, m_2)$, est le nombre de lettres par lequel ils diffèrent, c'est à dire $w(m_1 - m_2)$.*

On utilise ici pour la première fois le fait que $\mathcal{A}$ est un corps. La différence $m_1 - m_2$ est calculée lettre à lettre.

Exemple. $w(010011) = 3$, et $d(01101, 01000) = w(00101) = 2$.

Définition 26 *La* distance minimale *d'un code, notée d, est*

$$d \;=\; \min_{m_1, m_2 \in \mathcal{C}} d(m_1, m_2)$$

On dit ainsi que $\mathcal{C}$ est un code de paramètres (n, k, d).

Exemple. Ajouter un bit de parité donne un code $(n, n-1, 2)$. Le code à répétition est un code $(n, 1, n)$.

5.2.3 Borne de Gilbert-Varshamov

Comme on va le voir, la distance minimale est une propriété essentielle d'un code correcteur : plus elle est grande, plus on est capable de résister aux erreurs introduites par le canal. Il est alors naturel, fixant le nombre de lettres n, de chercher le nombre maximal de mots que peut contenir un code de distance minimale d. C'est un problème d'empilement de sphères (*sphere packing problem*).

Lemme 23 *Soient $\mathcal{A} = \{0, 1\}$ et $d \leq n$ dans $\mathbb{N}$. Soit $\mathcal{C} \subset \mathcal{A}^n$ un code de distance minimale $\geq d$, et soit $V(n, d)$ le nombre maximum de mots que $\mathcal{C}$ peut contenir. Si d est impair, avec $d = 2e + 1$, alors*

$$V(n, d) \cdot \sum_{k=0}^{e} C_n^k \;\leq\; 2^n$$

D'autre part,

$$V(n, d) \cdot \sum_{k=0}^{d-1} C_n^k \;\geq\; 2^n$$

En effet, soit $S(m, i)$ les sphères de centre $m \in \mathcal{A}^n$ et de rayon i dans $\mathcal{A}^n$, i.e. l'ensemble des mots m' de $\mathcal{A}^n$ tels que $d(m, m') \leq i$. Les sphères $S(m, e)$, pour $m \in \mathcal{C}$, sont disjointes puisque $e < \frac{d}{2}$. Chacune contient $\sum_{k=0}^{e} C_n^k$ éléments. D'où la première inégalité. Un code pour lequel cette borne supérieure est atteinte est dit *parfait* (voir plus loin).

Par ailleurs, soit $\mathcal{C}$ un code maximal de distance minimale d. Alors il n'existe aucun mot $m \in \mathcal{A}^n$ qui soit à distance $\geq d$ de tous les autres mots de $\mathcal{C}$, sinon on peut rajouter m à $\mathcal{C}$ sans changer la distance minimale et avoir un code plus grand. Donc les sphères $S(m, d-1)$, $m \in \mathcal{C}$, recouvrent entièrement $\mathcal{A}^n$. Chacune contient $\sum_{k=0}^{d-1} C_n^k$ éléments, d'où la seconde inégalité.

rajouter un mot sur la borne de Hamming : lorsqu'on sait que le code est t correcteur

5.2.4 Décodage à distance minimale

Supposons qu'un mot $m = l_1 l_2 \cdots l_n$ est émis sur un canal discret sans mémoire, et que l'on reçoit $m' = l'_1 l'_2 \cdots l'_n$. On a donc

$$\mathbb{P}(m'|m) \quad = \quad \prod_{i=1}^{n} \mathbb{P}(l'_i|l_i)$$

Le décodage *au maximum de vraisemblance a priori*[3] choisit comme mot d'entrée estimé

$$\hat{m} \quad = \quad \arg\max_{m \in \mathcal{C}} \mathbb{P}(m'|m)$$

Avec un canal symétrique, $\mathbb{P}(l'_i|l_i) = \phi(l'_i - l_i)$ où ϕ est la loi du bruit ajouté par le canal. On peut alors établir une "distance[4] statistique" entre deux mots par

$$\begin{aligned} \delta(m', m) &= -\ln \mathbb{P}(m'|m) + \ln \mathbb{P}(m'|m') \\ &= -\sum_{i=1}^{n} \ln \phi(l'_i - l_i) + n \ln \phi(0) \end{aligned}$$

Observer que maximiser $-\ln \mathbb{P}(m'|m)$ en m donne bien l'estimateur $\hat{m}$. Le terme constant $\ln \mathbb{P}(m'|m')$ permet d'assurer $\delta(m', m) = 0$ pour $m = m'$. Le décodage revient alors à trouver le mot du code le plus proche de m' pour cette "distance".

Revenons au cas binaire, avec $p < \frac{1}{2}$ la probabilité d'erreur du canal binaire symétrique. Il vient alors $\phi(e) = p \cdot \mathbb{1}_{e=1} + (1-p) \cdot \mathbb{1}_{e=0}$, ce qui donne

$$\begin{aligned} \delta(m', m) &= -w(m' - m) \cdot \ln p - [n - w(m' - m)] \cdot \ln(1-p) + n \cdot \ln(1-p) \\ &= w(m' - m) \cdot \ln \frac{1-p}{p} \end{aligned}$$

Comme $p < \frac{1}{2}$, minimiser $\delta(m', m)$ revient à minimiser la distance de Hamming. En d'autres termes, d'un point de vue vraisemblance, on paie un coût fixe pour chaque erreur. Il suffit donc de minimiser le nombre d'erreurs.

Construire un bon code correcteur revient donc à espacer le plus possible les mots du code dans $\mathcal{A}^n$, pour la distance δ en général, et d dans le cas binaire. Chaque mot m est ainsi au centre d'une *cellule de décodage*, constituée de l'ensemble des mots de $\mathcal{A}^n$ qui sont plus près de m que de tout autre élément de $\mathcal{C}$. On appelle cette partition un *diagramme de Voronoï*.

[3] Le maximum de vraisemblance *a posteriori* consisterait à prendre en compte la loi *a priori* $\boldsymbol{P}(\mu)$ sur les mots d'entrée : $\hat{m} = \arg\max_{\mu \in \mathcal{C}} \boldsymbol{P}(m'|\mu)\boldsymbol{P}(\mu) = \arg\max_{\mu \in \mathcal{C}} \boldsymbol{P}(\mu|m')$.

[4] Attention : ce n'est pas à proprement parler une distance. L'inégalité triangulaire n'est pas vraie. La positivité n'est garantie que si $\boldsymbol{P}(m'|m)$ est maximum pour $m' = m$. Enfin δ ne s'annule en $m' = m$ que si ce maximum est unique.

Lemme 24 *Un code binaire de distance minimale d est susceptible de corriger t erreurs et e effacements dès lors que $2t + e < d$. S'il n'y a pas d'effacement, il peut ainsi corriger $t = \lfloor \frac{d-1}{2} \rfloor$ erreurs.*

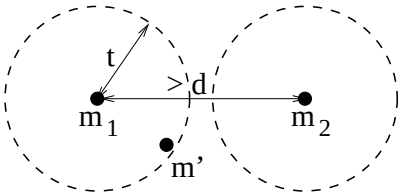

Figure 5.3: Cellules de décodage.

En effet, s'il n'y a pas d'effacement, chaque mot est au centre d'une cellule de diamètre au-moins égal à t tel que $2t < d$. Un erreur altérant moins de t bits sera ainsi corrigée par le décodage à distance minimale. S'il y a e effacements, le même raisonnement s'applique avec $d - e$ au lieu de d (**exercice**).

Définition 27 *Le code $\mathcal{C}$, de paramètres (n, k, d), est dit* parfait *si les boules de diamètre $\frac{d-1}{2}$ et de centre les mots de $\mathcal{C}$ forment un pavage de $\mathcal{A}^n$.*

"Pavage" signifie que l'on a la partition $\mathcal{A}^n = \cup_{m \in \mathcal{C}} S(m, \frac{d-1}{2})$. On atteint donc la borne dans la première inégalité de Gilbert-Varshamov, d'où

$$|\mathcal{C}| \cdot \sum_{k=0}^{\frac{d-1}{2}} C_n^k \quad = \quad 2^n$$

Remarque : un code parfait a nécessairement une distance minimale impaire.

5.2.5 Exemple

Nous allons étudier les propriétés du code de Hamming $(7, 4)$. C'est un code systématique. Les 4 bits utiles étant notés u_1, u_2, u_3, u_4, les bits de redondance r_1, r_2, r_3 sont construits par $r = f(u)$ définie par

$$\begin{aligned} r_1 &= u_2 + u_3 + u_4 \\ r_2 &= u_1 + u_3 + u_4 \\ r_3 &= u_1 + u_2 + u_4 \end{aligned}$$

On peut ainsi les lire comme des bits de parité sur des sous-ensembles de bits utiles, représentés figure 5.4.

On reçoit le mot $m' = u'r' = u'_1 u'_2 u'_3 u'_4 r'_1 r'_2 r'_3$. Pour le décodage, on regarde si le mot est dans le code, par la CNS $r' = f(u')$. Si ce n'est pas le cas, on forme un

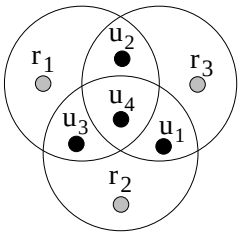

Figure 5.4: Le code de hamming, vu comme mettant des bits de parité sur des sous-ensembles de bits utiles. La somme des bits à l'intérieur de chaque cercle doit être nulle.

syndrome s par $s = r' - f(u')$. Avoir un syndrome non nul atteste non seulement de la présence d'une erreur, mais permet aussi de caractériser cette erreur (voir plus loin). Sur cet exemple, supposons qu'il y a une seule erreur, on peut alors construire le tableau suivant (utiliser la figure)

erreur sur	syndrome
u_1	011
u_2	101
u_3	110
u_4	111
r_1	100
r_2	010
r_3	001

qui permet de détecter, localiser et donc corriger sans ambiguïté l'erreur. Il s'ensuit que $d \geq 3$.

Exercice. Vérifier que $d = 3$, par exemple en construisant tous les mots du code. Montrer qu'il existe des cas d'erreur double que l'on ne peut pas corriger parfaitement. Le code de Hamming $(7, 4)$ est-il parfait ?

5.3 Codes linéaires

Nous allons commencer à utiliser explicitement le fait que $\mathcal{A}$ est un corps fini. Les codes linéaires couvrent la quasi totalité des codes correcteurs utilisés en pratique. Ils contiennent en particulier les codes cycliques et convolutifs étudiés plus loin.

5.3.1 Définition

Définition 28 *Le code $\mathcal{C}$, de paramètres (n, k), est* linéaire *ssi ses mots forment un sous-espace vectoriel de dimension k dans $\mathcal{A}^n$.*

Plus spécifiquement, $\mathcal{C}$ est un code linéaire ssi

$$\forall m_1, m_2 \in \mathcal{C}, \quad \forall a_1, a_2 \in \mathcal{A}, \quad a_1 \cdot m_1 + a_2 \cdot m_2 \in \mathcal{C}$$

soit, dans le cas binaire, ssi la somme de deux mots du code est encore un mot du code (la somme est ici à entendre bit à bit).

Exercice. Vérifier que le code de Hamming $(7, 4)$ est linéaire.

Lemme 25 *Pour un code linéaire,*

$$d \;=\; \min_{m \in \mathcal{C}^*} w(m) \qquad \textit{et} \qquad d \leq n + 1 - k$$

où $\mathcal{C}^*$ est le code $\mathcal{C}$ privé du mot nul (de poids 0).

En effet, $d(m_1, m_2) = w(m_1 - m_2)$ et $m_1 - m_2 \in \mathcal{C}$. Par ailleurs, soir $\mathcal{C}'$ le sous-espace vectoriel de $\mathcal{A}^n$ formé des mots dont les $k-1$ dernières lettres sont nulles. Comme $\dim(\mathcal{C}') = n - (k-1)$ et $\dim(\mathcal{C}) = k$, il vient $\dim(\mathcal{C} \cap \mathcal{C}') \geq 1$. Donc $\mathcal{C}$ contient au-moins un mot de $\mathcal{C}'$, qui est donc de poids $\leq n - (k-1)$.

Exercice. Vérifier que $d = 3$ pour le code de Hamming $(7, 4)$.

Calculer d est difficile en général, sauf à parcourir tous les mots du code. Il n'existe des formules que pour des codes très structurés (cf. plus loin). En général, toutefois, on parvient sans trop de difficulté à donner des bornes sur d.

d sert bien entendu à apprécier les capacités correctrices d'un code. Ce paramètre n'est pas pertinent à lui seul toutefois, car il se peut que toutes les paires de mots sauf une soient à une distance bien plus grande que d. On utilise plutôt le *spectre des distances de Hamming*, polynôme de la forme

$$\sum_{i \geq d} c_i \, x^i$$

où c_i est le nombre de mots du code de poids i. Des majorations simples de la probabilité d'erreur après décodage peuvent être obtenues à partir de ce polynôme (ou d'approximations de ce polynôme).

5.3.2 Matrice génératrice

Soit $(m_1, m_2, \ldots, m_k)$ une *base* de $\mathcal{C}$, sous-espace vectoriel de $\mathcal{A}^n$. Chaque m_i est ainsi composé de n lettres ; on les note sous forme de vecteurs ligne. Tous les mots de $\mathcal{C}$ peuvent ainsi s'écrire comme combinaison linéaire des m_i.

Définition 29 *Soit* $(m_1, m_2, \ldots, m_k)$ *une* base *de* $\mathcal{C}$ *dans* $\mathcal{A}^n$*, la matrice*

$$G \;=\; \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_k \end{pmatrix}$$

de $\mathcal{A}^{k\times n}$ *est une* matrice génératrice *de* $\mathcal{C}$*. Ainsi,*

$$m \in \mathcal{C} \quad \Leftrightarrow \quad \exists\, u \in \mathcal{A}^k,\;\; m = u \cdot G$$

(Dans l'équation précédente, u est aussi un vecteur ligne.) Si l'on prémultiplie G par une matrice de $\mathcal{A}^{k\times k}$ inversible, on obtient clairement une autre matrice génératrice du code – c'est la similitude de deux codes à une bijection près sur l'entrée dont nous avons parlé.

Exemple. Deux matrices génératrices du code de Hamming $(7, 4)$, la seconde étant sous forme systématique :

$$G_1 \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad G_2 \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Dans chacune, la sous-matrice 4×4 de gauche est de rang plein, ce qui montre que les vecteurs ligne sont linéairement indépendants. Toute ligne de G_2 est combinaison linéaire de lignes de G_1 : par exemple $m_{2,3}$, 3^{eme} ligne de G_2, est la somme des 2 lignes centrales de G_1, soit $m_{2,3} = m_{1,2} + m_{1,3} = (0, 1, 1, 0)G_1$. Il s'ensuit que les deux matrices engendrent bien le même sous-espace de $(\mathbb{F}_2)^7$, c'est à dire le même code $\mathcal{C}$. En considérant $(u_1, u_2, u_3, u_4)G_2$, on remarque que le bit de redondance r_1 vérifie $r_1 = u_2 + u_3 + u_4$, ce qui est bien la formule donnée section 5.2.5. Il en va de même pour r_2 et r_3.

Définition 30 *Les codes* $\mathcal{C}$ *et* $\mathcal{C}'$ *sont* équivalents *ssi ils possèdent les mêmes mots, à une permutation près des lettres, i.e. ssi pour toute matrice génératrice* G *de* $\mathcal{C}$ *il existe une permutation* π *de taille* n *telle que* $G' = G \cdot \pi$ *est une matrice génératrice de* $\mathcal{C}'$*.*

Théorème 13 *Tout code linéaire est équivalent à un code linéaire systématique.*

La matrice génératrice d'un code systématique est de la forme

$$G \;=\; [I_k \;\; P]$$

où I_k est la matrice identité de taille k. Soit G une matrice génératrice de $\mathcal{C}$. Comme $\mathcal{C}$ est de dimension k, G est de rang k. Il existe donc un sous-ensemble de k colonnes de G (d'indices $i_1 < i_2 < \ldots < i_k$) formant une matrice M inversible de $\mathcal{A}^{k\times k}$. Soit π la permutation qui place ces colonnes en tête, et soit

$$G' \;=\; G \cdot \pi \;=\; [M \;\; Q]$$

G' est la matrice génératrice d'un code $\mathcal{C}'$ équivalent à $\mathcal{C}$. En la prémultipliant par

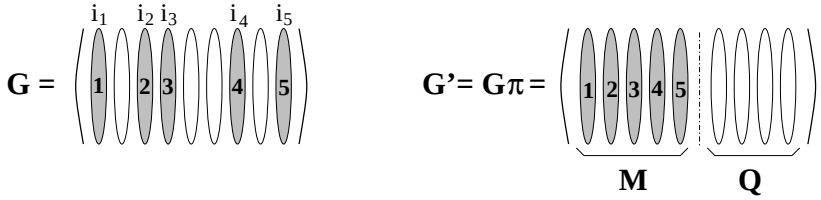

Figure 5.5: Ensemble d'information.

M^{-1} on obtient alors une autre matrice génératrice de $\mathcal{C}'$

$$G'' \;=\; M^{-1} \cdot G' \;=\; [I \;\; P]$$

qui en fait un code systématique. En faisant agir π^{-1} sur G'', on retrouve une matrice génératrice de $\mathcal{C}$ qui nous montre que $\mathcal{C}$ restreint aux lettres de positions $i_1, i_2, \ldots, i_k$ remplit tout $\mathcal{A}^k$. On appelle ainsi ces indices un *ensemble d'information* pour le code $\mathcal{C}$.

5.3.3 Matrice de contrôle

Soit $\mathcal{C}$ un code linéaire de matrice génératrice $G = [I_k \;\; P]$ (forme systématique). $m \in \mathcal{A}^n$ est un mot du code $\mathcal{C}$ ssi il s'écrit $m = u \cdot G$ où u est un élément de $\mathcal{A}^k$ et représente les k lettres utiles à transmettre. Détaillons les lettres de m et u : $m = [l_1, \ldots, l_n]$, $u = [u_1, \ldots, u_k]$. Il vient

$$\begin{aligned} [l_1, \ldots, l_k] &= [u_1, \ldots, u_k] \\ [l_{k+1}, \ldots, l_n] &= [u_1, \ldots, u_k] \cdot P \end{aligned}$$

En d'autres termes, m est un mot de $\mathcal{C}$ ssi $[l_1, \ldots, l_k] \cdot P = [l_{k+1}, \ldots, l_n]$. Posant

$$H \;\triangleq\; [\, -P^T \; I_{n-k} \,]$$

il vient que $m \in \mathcal{C}$ ssi $m \cdot H^T = 0$.

Définition 31 *H est une* matrice de contrôle *pour le code $\mathcal{C}$ ssi elle vérifie*

$$\forall m \in \mathcal{A}^n, \;\; [\; m \in \mathcal{C} \;\Leftrightarrow\; m \cdot H^T = 0 \;]$$

Lemme 26 *Soit $G = [I_k \quad P]$ une matrice génératrice (systématique) du code $\mathcal{C}$. Une* matrice de contrôle *de $\mathcal{C}$ est obtenue par $H = [-P^T \; I_{n-k}]$.*

On montre en fait à la section suivante qu'une matrice de contrôle est au-moins de dimension $(n-k) \times n$. Elle représente toutes les contraintes que doivent satisfaire les lettres d'un mot de code. Si $\mathcal{C}$ est de dimension k dans $\mathcal{A}^n$, il peut en effet se construire comme intersection de $n-k$ hyper-plans. Chaque ligne de H donne l'équation de l'un de ces hyperplans.

Définition 32 *Soient $m \in \mathcal{A}^n$ et H une matrice de contrôle du code $\mathcal{C}$, le syndrôme de m est défini par*

$$s \quad = \quad m \cdot H^T \qquad s \in \mathcal{A}^{n-k}$$

Lemme 27 *Soit $m \in \mathcal{A}^n$, m est un mot du code $\mathcal{C}$ ssi son syndrome est nul.*

Remarque. En construisant H à partir d'une matrice génératrice systématique de $\mathcal{C}$, il est facile de vérifier que tous les éléments de $\mathcal{A}^{n-k}$ sont des syndromes possibles.

Exemple. Considérons la matrice génératrice systématique G_2 du code de Hamming $(7, 4)$, elle induit la matrice de contrôle H_2 :

$$G_2 \; = \; \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad H_2 \; = \; \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Exercice. Construire un mot m du code. Vérifier $m \cdot H_2^T = 0$. Calculer $G_2 \cdot H_2^T$. Vérifier que les syndromes calculés plus haut (section 5.2.5) pour le code de Hamming $(7, 4)$ correspondent bien à la formule eH_2^T.

Lemme 28 *Soit H une matrice de contrôle du code $\mathcal{C}$. $\mathcal{C}$ est de distance minimale d ssi tout sous-ensemble de $d-1$ colonnes de H est libre (i.e. linéairement indépendant) et il existe d colonnes de H liées.*

La preuve découle directement de la définition de H. Tout élément $m \in \mathcal{A}^n$ de poids $< d$ ne peut être dans $\mathcal{C}$, et donc ne peut vérifier $m \cdot H^T = 0$. Donc on ne peut pas trouver $d-1$ colonnes de H liées. En revanche, il existe un mot de poids d dans $\mathcal{C}$ ses lettres non nulles construisent une combinaison linéaire nulle de colonnes de H. Ces d colonnes sont donc liées.

5.3.4 Code dual

On peut munir $\mathcal{A}^n$, vu comme un $\mathcal{A}$-espace vectoriel, d'un produit scalaire :

$$\langle m, m' \rangle \;=\; m \cdot (m')^T \;=\; \sum_{i=1}^{n} l_i \, l'_i$$

où $m = l_1 \cdots l_n$ et $m' = l'_1 \cdots l'_n$. Deux vecteurs de $\mathcal{A}^n$ sont dits *orthogonaux* ssi leur produit scalaire est nul. *Attention :* ce produit scalaire ne définit pas une norme, car il existe des vecteurs dits *isotropes* dont le produit scalaire avec eux-mêmes est nul.

Exemple. $(1, 1)$ est isotrope dans $(\mathbb{F}_2)^2$ car $1^2 + 1^2 = 0$.

Définition 33 *Soit $\mathcal{C}$ un code de $\mathcal{A}^n$, son* code dual *est défini par*

$$\mathcal{C}^\perp \;=\; \{m' \in \mathcal{A}^n \;:\; \forall m \in \mathcal{C}, \langle m, m' \rangle = 0\}$$

Lemme 29 *Soit $\mathcal{C}$ un code sur $\mathcal{A}^n$, si $\dim \mathcal{C} = k$ alors $\dim \mathcal{C}^\perp = n - k$. Il s'ensuit que $(\mathcal{C}^\perp)^\perp = \mathcal{C}$.*

Il faut prendre garde que $\mathcal{C} \cap \mathcal{C}^\perp$ n'est pas toujours égal à $\{0\}$, contrairement aux espaces vectoriels sur $\mathbb{R}$ ou $\mathbb{C}$, à cause des vecteurs isotropes. Pourtant la règle des dimensions reste vraie. En effet, soit $G = [I_k \;\; P]$ génératrice de $\mathcal{C}$, et soit $H = [-P^T \;\; I_{n-k}]$ matrice de contrôle, alors H génère $\mathcal{C}'$ qui est égal à $\mathcal{C}^\perp$. En effet,

$$\begin{aligned} \langle m, m' \rangle &= \langle uG, vH \rangle \\ &= (uG) \cdot (vH)^T \\ &= uGH^T v^T \\ &= 0 \end{aligned}$$

car par construction $GH^T = 0$. Donc $\mathcal{C}' \subset \mathcal{C}^\perp$, d'où $\dim \mathcal{C}^\perp \geq n - k$. Si $\dim \mathcal{C}^\perp > n-k$, alors $\exists\, m' \in \mathcal{C}^\perp \backslash \mathcal{C}'$, et on peut le choisir tel que ses $n-k$ dernières composantes sont nulles (en retirant sa projection sur $\mathcal{C}'$). Ce vecteur vérifie

$$\begin{aligned} [I_k \;\; P](m')^T &= 0 \qquad \text{car } m' \perp \mathcal{C} \\ [0 \;\; I_{n-k}](m')^T &= 0 \qquad \text{à cause de la structure de } m' \end{aligned}$$

Il vient que m' est dans le noyau à droite de la matrice

$$\begin{pmatrix} I_k & P \\ 0 & I_{n-k} \end{pmatrix}$$

or cette matrice est de rang plein, donc $m' = 0$, ce qui contredit l'hypothèse. Et finalement $\dim \mathcal{C}^\perp = n - k$, soit $\mathcal{C}^\perp = \mathcal{C}'$.

Pour montrer que $(\mathcal{C}^\perp)^\perp = \mathcal{C}$, il suffit de remarquer $\mathcal{C} \subset (\mathcal{C}^\perp)^\perp$; on conclut par l'égalité des dimensions.

Lemme 30 *Soit $\mathcal{C}$ un code de matrice génératrice G et de matrice de contrôle H. Son code dual $\mathcal{C}^\perp$ est engendré par H et admet G comme matrice de contrôle.*

Cela découle immédiatement de la preuve précédente.

5.3.5 Décodage par syndrome

Partant d'un mot reçu $m' \in \mathcal{A}^n$, le décodage consiste à trouver le mot $\hat{m}$ de $\mathcal{C}$ le plus proche pour la distance de Hamming[5]. On peut écrire

$$m' \;=\; m + e$$

où e est l'erreur qui s'est rajoutée au mot de code émis m. Le décodage à distance minimale recherche le $\hat{e}$ de poids minimal tel que $m' - \hat{e} \in \mathcal{C}$. Si on a de la chance, $\hat{e} = e$ et on a corrigé l'erreur de transmission, sinon on fait une erreur de décodage...

On peut caractériser l'erreur e en calculant le syndrome de m' :

$$m' \cdot H^T \;=\; m \cdot H^T + e \cdot H^T \;=\; e \cdot H^T$$

Si le syndrome est nul, m' est un mot du code, et l'on décide $\hat{m} = m'$. Pour chaque autre valeur possible du syndrome, il faut construire une estimée $\hat{e}$ de l'erreur de transmission, puis on prend $\hat{m} = m' - \hat{e}$.

On commence par définir une relation d'équivalence sur $\mathcal{A}^n$: deux éléments μ et μ' de $\mathcal{A}^n$ sont équivalents ssi ils ont le même syndrome :

$$\mu \equiv \mu' \;\Leftrightarrow\; \mu - \mu' \in \mathcal{C} \;\Leftrightarrow\; \mu H^T = \mu' H^T$$

La classe d'équivalence d'un élément μ est parfois appelée *classe latérale* de μ. Le quotient de $\mathcal{A}^n$ par la relation $\equiv$, noté $\mathcal{A}^n/_{\equiv}$ ou $\mathcal{A}^n/_{\ker H^T}$, contient D^{n-k} classes de D^k éléments, une classe par syndrome. Le *chef de classe* est l'élément de poids minimal dans la classe. On le note e_i pour la classe E_i, $i = 0 \dots D^{n-k} - 1$. Par construction, on a donc $E_i = e_i + \mathcal{C}$.

> *Dans cette construction, e_i représente l'erreur de poids minimal qui provoque l'apparition du symptôme de la classe E_i.*

Ainsi, partant d'un symptôme, on obtient l'erreur à retrancher au mot m' reçu pour retrouver le mot du code le plus proche.

Exemple. Soit $\mathcal{C}$, code $(5, 2, 3)$ sur $\mathcal{A} = \mathbb{F}_2$ engendré par G,

$$G \;=\; \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{array}\right) \qquad H \;=\; \left(\begin{array}{cc|ccc} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{array}\right)$$

(Noter que + et - coïncident sur $\mathbb{F}_2$.) Le *tableau standard*, représentant les classes de $\mathcal{A}^n = \mathbb{F}_2^5$ sur chaque horizontale, est donné par

[5] Cela se généralise à la "distance statistique" définie section 5.2.4.

classe	syndrome	chef			
E_0	000	00000	10011	**01110**	11101
E_1	001	00001	10010	01111	11100
E_2	010	00010	10001	01100	11111
E_4	100	00100	10111	01010	11001
E_6	110	01000	11011	00110	10101
E_3	011	**10000**	00011	**11110**	01101
E_5	101	11000	01011	10110	00101
E_7	111	10100	00111	11010	01001

On le construit en mettant sur la première ligne les mots du code. Puis on cherche une "erreur", c'est à dire un mot de $\mathcal{A}^n$ hors du code et de poids le plus faible possible, dont on construit la classe en lui ajoutant tous les mots du code (soit la ligne 1), puis on recherche une nouvelle erreur, etc. jusqu'à avoir placé tous les éléments de $\mathcal{A}^n$.

Exemple de décodage : on a reçu $m' = 11110$, de syndrome 011, donc appartenant à la classe E_3. L'erreur estimée est $e_3 = 10000$, le premier mot de la ligne, si on la retranche on obtient $\hat{m} = 01110$, qui est en fait le premier mot de la colonne.

Il peut y avoir ambiguité : c'est le cas lorsqu'il y a plusieurs mots de poids minimal dans une classe (par ex. E_5 et E_7). Dans ce cas on choisit l'un de ces mots comme "erreur" : on est en effet incapable de les distinguer, ils correspondent à deux $\hat{m}$ de même vraisemblance. Il est normal que l'on ne puisse pas corriger toutes les erreurs puisque $d \leq n-k+1$; on peut espérer corriger au plus $\frac{n-k}{2}$ erreurs. Avec $d = 3$, on ne corrige qu'une erreur ; les erreurs doubles donnent lieu à ambiguité.

Noter que le tableau standard n'est jamais utilisé en pratique ; ce n'est qu'une représentation à garder en tête, pour comprendre les phénomènes. On a recours en général à des méthodes *calculatoires*, utilisant la structure du code, pour localiser directement les bits erronés.

5.4 Exemples de codes linéaires classiques

5.4.1 Codes de Hamming

On construit un code de Hamming $\mathcal{C}_H$ à partir de sa matrice de contrôle H. H est une matrice à l lignes et à $2^l - 1$ colonnes, constituées de tous les vecteurs (colonnes) non nuls de $\mathcal{A}^l$ avec $\mathcal{A} = \mathbb{F}_2 = \{0, 1\}$. Il vient $n = 2^l - 1$ et $n - k = l$ soit $k = 2^l - 1 - l$. Pour $l = 3$, on peut vérifier cette structure sur la matrice H_2 de la section 5.3.3.

Proposition 1 *Un code de Hamming est parfait et 1-correcteur.*

Il suffit de remarquer qu'il est de distance minimale $d = 3$, ce qui est une conséquence directe du lemme 28 (**exercice**). Pour montrer qu'il est parfait, on vérifie qu'il atteint la borne de Gilbert-Varshamov (**exercice**), c'est à dire que les sphères $S(m, 1)$ recouvrent $\mathcal{A}^n$ et sont d'intersection vide.

Remarque. Chaque élément m' de $\mathcal{A}^n$ s'écrit $m' = m + e$ où e est l'un des éléments de la sphère $S([0 \dots 0], 1)$, centrée sur le mot nul. Le décodage à distance minimale revient à trouver ce e. *Une astuce :* supposons que l'on a classé les colonnes de H par ordre croissant en binaire, i.e. la i^{eme} colonne correspond au codage binaire de $i-1$. Le calcul du syndrome $m' \cdot H^T = e \cdot H^T$ sélectionne alors une seule colonne de H (car e n'a qu'un seul bit à 1). Il suffit donc de retraduire le syndrome ($\in \mathcal{A}^{n-k} = \mathbb{F}_2^l$) du binaire aux entiers et d'ajouter 1 pour retrouver la position i du bit erroné dans m'.

5.4.2 Codes de Reed-Muller

À compléter...

5.4.3 Code de Golay

À compléter...

5.5 Transformations de codes

La construction algébrique de bons codes linéaires, comme on va le voir plus loin, impose des valeurs particulières aux paramètres (n, k, d). Il existe donc des techniques permettant de modifier un code afin d'ajuster ses paramètres au contexte d'une application. Nous allons donner le principe de chaque transformation, et ses propriétés générales. Garder à l'esprit que l'on peut obtenir des propriétés plus fortes, en appliquant les transformations à tel ou tel code particulier.

5.5.1 Extension

$\mathcal{C}'$ est obtenu en ajoutant un bit de parité à chaque mot de $\mathcal{C}$, d'où

$$n' = n + 1 \qquad k' = k \qquad d \leq d' \leq' d + 1$$

En fait, si d est pair, $d' = d$, et si d est impair, $d' = d + 1$ (**exercice**). Si H est une matrice de contrôle de $\mathcal{C}$, alors on obtient une matrice de contrôle du code étendu par (**exercice**)

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

5.5.2 Sélection d'un sous-code

Cela consiste simplement à ne garder qu'un sous-espace vectoriel de dimension $k' = k - 1$ dans $\mathcal{C}$, de façon à augmenter d.

$$n' = n \qquad k' = k - 1 \qquad d' \geq d$$

Exercice. Si d est impair, sélectionner les mots de poids pair dans $\mathcal{C}$ donne un code $\mathcal{C}'$ pour lequel $d' > d$.

5.5.3 Augmentation

C'est l'opération duale : toujours à n fixe, on augmente la dimension de l'espace vectoriel. $\mathcal{C}'$ est l'espace engendré par $\mathcal{C}$ et un élément $\mu \in \mathcal{A}^n \setminus \mathcal{C}$.

$$\mathcal{C}' \;=\; sp\{\mathcal{C}, \mu\} \;=\; \mathcal{C} \cup (\mu + \mathcal{C})$$

On a intérêt, bien entendu, à choisir μ "loin" de $\mathcal{C}$, i.e. tel que $d(\mu, \mathcal{C}) = \min_{m \in \mathcal{C}} d(\mu, m)$ est maximal. On appelle *rayon de recouvrement* cette distance maximale

$$\rho(\mathcal{C}) \;\triangleq\; \max_{\mu \in \mathcal{A}^n} \min_{m \in \mathcal{C}} d(\mu, m)$$

En choisissant μ tel que $d(\mu, \mathcal{C}) = \rho(\mathcal{C})$, il vient (**exercice**)

$$n' = n \qquad k' = k + 1 \qquad d' = min(d, \rho)$$

5.5.4 Perforation

C'est une technique très répandue qui permet de diminuer n. Cela consiste simplement à effacer une lettre du code, par exemple la dernière, ce qui revient à projeter $\mathcal{C}$ sur $\mathcal{A}^{n-1}$. Il faut prendre garde à ne pas faire décroître k, c'est à dire à conserver la dimension de $\mathcal{C}$.

$$n' = n - 1 \qquad k' = k \qquad d' \geq d - 1$$

En partant de bons codes, par exemple les codes cycliques de longueur $n = 2^m - 1$, on peut s'adapter aux contraintes de l'application en perforant i fois. Il existe des techniques pour raffiner la borne $d \geq d - i$, fort heureusement !

5.5.5 Raccourcissement

C'est une combinaison de la sélection d'un sous-code et de la perforation. On sélectionne le sous-code de $\mathcal{C}$ formé des mots ayant un 0 dans une position donnée, par exemple la dernière ; $\mathcal{C}'$ est alors le préfixe de ces mots (on enlève le zéro final).

$$n' = n - 1 \qquad k' \geq k - 1 \qquad d' \geq d$$

La technique de raccourcissement de codes cycliques donne en général de bons codes.

5.6 Combinaisons de codes

5.6.1 Produit cartésien

Définition 34 *Le* produit cartésien *des codes $\mathcal{C}_1$ et $\mathcal{C}_2$, de paramètres (n_1, k_i, d_i) est obtenu en concaténant les mots de code associés à* deux entrées différentes

$$\mathcal{C}' \;=\; \mathcal{C}_1 \times \mathcal{C}_2 \;\triangleq\; \{(m_1|m_2) \;:\; m_1 \in \mathcal{C}_1, m_2 \in \mathcal{C}_2\}$$

C'est un code de paramètres $(n' = n_1 + n_2, k' = k_1 + k_2, d' = \min(d_1, d_2))$.

Cette technique se révèle utile lorsque les deux codes sont de même longueur, $n_1 = n_2 = n$, on peut alors construire un code par

$$\mathcal{C}' \;=\; \{(m_1|m_1 + m_2) \;:\; m_1 \in \mathcal{C}_1, m_2 \in \mathcal{C}_2\} \tag{5.1}$$

Lemme 31 *Le code $\mathcal{C}'$ défini par (5.1) est de paramètres* $(n' = 2n, k' = k_1 + k_2, d' = \min(2d_1, d_2))$.

La preuve est laissée en exercice.

5.6.2 Juxtaposition

Cela consiste à juxtaposer les mots de code associés à *une même entrée* de $\mathcal{A}^k$. On suppose donc que $\mathcal{C}_1$ et $\mathcal{C}_2$ sont de même dimension, $k_1 = k_2 = k$. Si G_1 et G_2 sont des matrices génératrices de $\mathcal{C}_1$ et $\mathcal{C}_2$ alors $\mathcal{C}'$ est engendré par $G' = [G_1|G_2]$ et de paramètres

$$n' = n_1 + n_2 \qquad k' = k \qquad d' \geq \min(d_1, d_2)$$

Noter que si G_1 et G_2 sont sous forme systématique, alors les bits utiles apparaissent deux fois dans les mots de $\mathcal{C}'$. On sait que la répétition n'est pas très efficace pour lutter contre les erreurs, il vaut donc mieux se contenter de juxtaposer les parties redondantes, en ne gardant qu'une fois la partie utile.

5.6.3 Concaténation

C'est en fait la mise en série de deux codes : la sortie du premier est l'entrée du second. Il faut donc avoir $k_2 = n_1$

Différence avec juxtaposition : ici on recode la partie redondante 1, en plus de la partie utile.

à revoir et à compléter

5.6.4 Produit de codes

C'est une technique très utilisée en pratique. $\mathcal{C}' = \mathcal{C}_1 \otimes \mathcal{C}_2$, de paramètres $(n' = n_1 n_2, k' = k_1 k_2)$ procède au codage de la façon suivante : on écrit[6] les $k_1 k_2$ bits à coder dans un tableau $k_2 \times k_1$. On procède ensuite au codage de chaque ligne par $\mathcal{C}_1$, ce qui donne un tableau $k_2 \times n_1$. On code ensuite chaque colonne de ce tableau par $\mathcal{C}_2$, ce qui donne un tableau $n_2 \times n_1$, qui est le mot de code associé aux $k_1 k_2$ bits d'entrée. Si on procède au codage dans l'ordre inverse, on obtient le même résultat (**exercice**).

rajouter une figure

Lemme 32 *Le produit $\mathcal{C}' = \mathcal{C}_1 \otimes \mathcal{C}_2$ est de paramètres $(n' = n_1 n_2, k' = k_1 k_2, d' = d_1 d_2)$.*

Lemme 33 *$(\mathcal{C}_1 \otimes \mathcal{C}_2)^\perp$ a pour paramètres $(n_1 n_2, n_1 n_2 - k_1 k_2, \min[d(\mathcal{C}_1^\perp), d(\mathcal{C}_2^\perp)])$.*

à revoir et à compléter ; rajouter des preuves

[6] avec pour convention d'écriture : de droite à gauche et de haut en bas

Chapter 6

Codes cycliques

Les codes cycliques forment une sous-classe des codes linéaires, et sont les plus utilisés en pratique. Ils conjuguent en effet de nombreux avantages : leur mise en oeuvre (codage/décodage) est facile, ils offrent une gamme étendue de codes, avec de nombreux choix de paramètres (n, k, d), et enfin permettent de corriger différents types d'erreurs, isolées ou par paquets.

Dans ce chapitre, on suppose que $\mathcal{A}$ est le corps fini à q éléments $\mathbb{F}_q$. Pour les cas pratiques, q est de la forme 2^m.

6.1 Définition

Définition 35 *Le code linéaire $\mathcal{C} \subset (\mathbb{F}_q)^n$ est* cyclique *ssi il est stable par permutation circulaire des lettres dans chaque mot, c'est à dire ssi (en notant m_i la ième lettre de m, $m_i \in \mathcal{A}$)*

$$m = (m_0, m_1, \ldots, m_{n-1}) \in \mathcal{C} \quad \Rightarrow \quad m' = (m_{n-1}, m_0, m_1, \ldots, m_{n-2}) \in \mathcal{C}$$

Cette propriété est plus facile à caractériser en adoptant une *notation polynômiale* pour décrire les éléments de $\mathbb{F}_q^n$.

Notation : L'élément $m = (m_0, m_1, \ldots, m_{n-1})$ de $\mathcal{A}^n$ est représenté par le polynôme $m(X) = m_0 + m_1 X + m_2 X^2 + \ldots + m_{n-1} X^{n-1}$ de $\mathbb{F}_q[X]$.

On rappelle que $\mathbb{F}_q[X]$ est l'anneau des polynômes à coefficients dans $\mathbb{F}_q$. Considérons le produit de $m(X)$ par X. Il vient

$$X \cdot m(X) = m_0 X + m_1 X^2 + \ldots + m_{n-2} X^{n-1} + m_{n-1} X^n$$

Si on calcule le reste de $X \cdot m(X)$ modulo $X^n - 1$ il vient

$$m'(X) \; = \; X \cdot m(X) \; {}_{[X^n-1]} \; = \; m_{n-1} + m_0 X + m_1 X^2 + \ldots + m_{n-2} X^{n-1}$$

car $X^n = 1 \ _{[X^n-1]}$. Il vient que le code $\mathcal{C}$ est cyclique ssi il est stable par multiplication par X modulo $X^n - 1$, c'est à dire ssi il est stable par multiplication par X dans l'anneau quotient $\mathbb{F}_q[X]/X^n-1$. Par suite, $\mathcal{C}$ reste stable par multiplication par X^2, par X^3, etc. Comme il est déjà stable par addition et produit par un scalaire $b \in \mathbb{F}_q$ (c'est un espace vectoriel), il est donc stable par multiplication par $b_0 + b_1X + b_2X^2 + \cdots$. En d'autres termes

$$m(X) \in \mathcal{C} \quad \Rightarrow \quad \forall\, b(X) \in \mathbb{F}_q[X], \quad b(X) \cdot m(X) \ _{[X^n-1]} \ \in \mathcal{C}$$

ce qui peut se dire de façon savante

Lemme 34 *Dans la notation polynômiale, un code cyclique de $(\mathbb{F}_q)^n$ est un idéal de l'anneau $\mathbb{F}_q[X]/X^n-1$.*

6.2 Polynôme générateur

Tout idéal de $\mathbb{F}_q[X]/X^n-1$ est engendré par un seul élément[1] ; cela tient à l'existence d'une division euclidienne dans cet anneau.

Définition 36 *Le* polynôme générateur *du code cyclique $\mathcal{C}$ est le polynôme normalisé de plus bas degré de $\mathcal{C}$. On le note génériquement $g(X)$.*

"Normalisé" signifie que le coefficient du monôme de plus haut degré vaut 1. Cette normalisation garantit l'unicité de $g(X)$. Noter que $g(X)$ ne peut être le polynôme nul avec cette définition.

Lemme 35 *Les mots de $\mathcal{C}$ sont les multiples de $g(X)$ dans $\mathbb{F}_q[X]/X^n-1$. Plus précisément, si* $\mathrm{d}^{\mathrm{o}}\ g(X) = n - k$,

$$\forall\, m(X) \in \mathcal{C}, \ \ \exists !\, a(X) \in \mathbb{F}_q[X], \ \ \mathrm{d}^{\mathrm{o}}\ a(X) < k \ \ : \quad m(X) = a(X) \cdot g(X)$$

et donc $\mathrm{d}^{\mathrm{o}}\ g(X) + \dim \mathcal{C} = n$.

En effet, soit $m(X) \in \mathcal{C}$. Calculons sa division euclidienne par $g(X)$

$$m(X) \ = \ q(X) \cdot g(X) + r(X), \qquad \mathrm{d}^{\mathrm{o}}\ r(X) < \mathrm{d}^{\mathrm{o}}\ g(X)$$

On a vu que tout multiple de $g(X)$ est dans $\mathcal{C}$. Il vient que $r(X) = m(X) - q(X) \cdot g(X)$ est dans $\mathcal{C}$, ce qui contredit la définition de $g(X)$ comme étant de degré minimal dans $\mathcal{C}$, sauf si $r(X)$ est nul. Le second point du lemme vient en remarquant que $\mathrm{d}^{\mathrm{o}}\ q(X) < k$ car $\mathrm{d}^{\mathrm{o}}\ m(X) < n$. Pour l'unicité, on suppose $m(X) = a(X)g(X) = a'(X)g(X)$, d'où $[a(x) - a'(X)]g(X) = 0$. il suffit alors de raisonner sur les degrés.

[1] Un tel anneau est dit *principal*.

Lemme 36 *Soit $\mathcal{C}$ un code cyclique de $(\mathbb{F}_q)^n$, de polynôme générateur*

$$g(X) = g_0 + g_1 X + g_2 X^2 + \ldots + g_{n-k} X^{n-k}$$

on obtient une matrice génératrice de $\mathcal{C}$ par

$$G \;=\; \begin{pmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k} & \ddots & \vdots \\ \vdots & \ddots & \ddots & & & \ddots & 0 \\ 0 & \cdots & 0 & g_0 & g_1 & \cdots & g_{n-k} \end{pmatrix} \;\in\; (\mathbb{F}_q)^{k\times n}$$

Ce résultat n'est que la traduction vectorielle du lemme précédent. Il met en évidence la nature beaucoup plus structurée des codes cycliques. On observera que G est bien de rang k, car la sous-matrice $k \times k$ de droite est tridiagonale, avec des 1 sur la diagonale.

Lemme 37 *Soit $g(X)$ le polynôme générateur du code cyclique $\mathcal{C}$ dans $(\mathbb{F}_q)^n$, $g(X)$ est un diviseur de $X^n - 1$, ce que l'on note $g(X) \,|\, X^n - 1$.*

Ce résultat nous permettra de construire les codes cycliques à partir des diviseurs de $X^n - 1$. La preuve est identique à celle du lemme 35 : on calcule la division euclidienne de $X^n - 1$ par $g(X)$, et on conclut en passant modulo $X^n - 1$, ce qui donne $0 = q(X)g(X) + r(X) \in \mathcal{C}$, donc $r(X) \in \mathcal{C}$, etc.

6.3 Codage systématique

On souhaite coder les k lettres utiles $u_1, \ldots, u_k$ avec le code cyclique $\mathcal{C}$ de sorte que le mot de code se contente de rajouter des bits de redondance. On pose pour cela

$$u(X) \;=\; u_1 X^{n-1} + u_2 X^{n-2} + \ldots + u_k X^{n-k}$$

et l'on calcule la division euclidienne de $u(X)$ par $g(X)$:

$$u(X) \;=\; q(X) \cdot g(X) + r(X) \qquad \text{d}^\text{o}\; r(X) < \text{d}^\text{o}\; g(X) = n - k$$

Il vient que $m(X) = u(X) - r(X)$ est un mot du code $\mathcal{C}$, et les polynômes $u(X)$ et $r(X)$ n'ont aucun monôme en commun : les coefficients de $r(X)$ vont occuper les $n - k$ premiers monômes e $m(X)$, et ceux de $u(X)$ les k derniers. On a donc bien un codage systématique, les bits de redondance étant rajoutés en tête du mot.

Exemple. Sur $(\mathbb{F}_2)^7$, on considère le code engendré par $g(X) = 1+X+X^3$. Il s'agit bien d'un diviseur de $X^7 + 1$ car on a la décomposition en polynômes irréductibles

$$X^7 + 1 \;=\; (1 + X)(1 + X + X^3)(1 + X^2 + X^3)$$

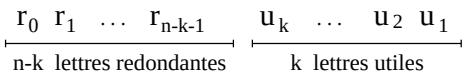

Figure 6.1: Forme systématique pour un mot de code cyclique.

$g(X)$ étant de degré 3, le code $\mathcal{C}$ est de dimension 4 et *une* de ses matrices génératrices est

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

où les points représentent des 0. Soit $u = (1, 1, 0, 0)$ l'entrée du codeur, on construit alors $u(X) = X^6 + X^5$ et l'on calcule son reste modulo $g(X)$:

$$X^6 + X^5 = (X^3 + X^2 + X)g(X) + X$$

ce qui donne le mot de code $m = (0, 1, 0, 0, 0, 1, 1)$ pour u. On peut ainsi calculer une matrice génératrice sous forme systématique pour $\mathcal{C}$ en calculant les restes de X^6, X^5, X^4 et X^3 par $g(X)$. Elle a la forme (**exercice**)

$$G' = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot G$$

ce qui "mélange" un peu les bits utiles en fin de mot. Une forme systématique plus conventionnelle peut être obtenue en observant que la sous-matrice 4×4 gauche de G, notée M, est de rang 4. Donc les 4 premières lettres du code forment un ensemble d'information. En prémultipliant G par M^{-1} on a la matrice génératrice

$$G'' = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot G$$

où il apparaît que $\mathcal{C}$ est équivalent au code de Hamming $(7, 4)$.

6.4 Matrice de contrôle

On a vu au lemme 37 que $g(X)|X^n - 1$, on peut donc écrire la factorisation $g(X)h(X) = X^n - 1$ dans $\mathbb{F}_q[X]$, où $h(X)$ est aussi un polynôme normalisé. $h(X)$ joue un rôle très important pour les codes cycliques. Notamment il engendre le code dual de $\mathcal{C}$, qui est aussi cyclique (voir section suivante). Par conséquent il permet aussi de construire une matrice de contrôle de $\mathcal{C}$.

Lemme 38 *Soit $\mathcal{C}$ un code cyclique de $(\mathbb{F}_q)^n$, de polynôme générateur $g(X)$, et soit $h(X)$ tel que $g(X)h(X) = X^n - 1$. Soit m un mot quelconque de $(\mathbb{F}_q)^n$, alors*

$$m \in \mathcal{C} \quad \Leftrightarrow \quad m(X)h(X) = 0 \ _{[X^n-1]}$$

La CN est immédiate puisque tout mot de $\mathcal{C}$ s'écrit $m(X) = a(X)g(X)$. Pour la CS, on écrit $m(X)h(X) = (X^n-1)f(X)$ dans $\mathbb{F}_q[X]$, avec $\mathrm{d}^\mathrm{o}\ f(X) \leq \mathrm{d}^\mathrm{o}\ h(X)-1 = k-1$. En divisant chaque membre par $g(X)$, il vient $m(X) = f(X)g(X)$, qui caractérise $m \in \mathcal{C}$.

Observons en détail le produit $p(X) = m(X)h(X)\ _{[X^n-1]}$. En notant $p(X) = p_0 + p_1X + \ldots + p_{n-1}X^{n-1}$ et $h(X) = h_0 + h_1X + \ldots + h_kX^k$, on a

$$\begin{array}{rcl} p_{n-1} & = & h_0m_{n-1} + h_1m_{n-2} + \ldots + h_km_{n-k-1} \\ p_{n-2} & = & h_0m_{n-2} + h_1m_{n-3} + \ldots + h_km_{n-k-2} \\ & \vdots & \\ p_k & = & h_0m_k + h_1m_{k-1} + \ldots + h_km_0 \end{array}$$

Formons alors la matrice H

$$H \ = \ \begin{pmatrix} h_k & h_{k-1} & \ldots & h_0 & 0 & \ldots & 0 \\ 0 & h_k & h_{k-1} & \ldots & h_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & & & \ddots & 0 \\ 0 & \ldots & 0 & h_k & h_{k-1} & \ldots & h_0 \end{pmatrix} \ \in \ (\mathbb{F}_q)^{(n-k)\times n}$$

H est une matrice circulante construite à partir du polynôme $h(X)$ *retourné*. L'équation $p(X) = 0$ se traduit en $m \cdot H^T = 0$. H est donc une matrice de contrôle du code $\mathcal{C}$.

Le polynôme $h(X)$ peut aussi servir au décodage du code $\mathcal{C}$. Supposons que le mot m a été émis, et que l'on reçoit $m' = m + e$, où e est le vecteur d'erreur. On calcule alors $s(X) = m'(X)h(X)\ _{[X^n-1]} = m(X)h(X)\ _{[X^n-1]} + e(X)h(X)\ _{[X^n-1]} = e(X)h(X)\ _{[X^n-1]}$. Cette équation ressemble beaucoup à un calcul de syndrome... Si $s(X) = 0$, le mot $m'(X)$ est accepté. Sinon, il faut estimer l'erreur e, c'est à dire choisir un e' de poids minimal tel que $s(X) = e'(X)h(X)$. On peut tenter de calculer la division euclidienne de $s(X)$ par $h(X)$: cela donne un vecteur d'erreur possible, mais pas forcément celle de poids minimal.

6.5 Code dual

Il est commode ici de changer légèrement la définition du produit scalaire de deux éléments de $(\mathbb{F}_q)^n$.

$$\langle m, m' \rangle_2 \ = \ \sum_{i=0}^{n-1} l_i \, l'_{n-1-i}$$

où $m = l_0 \cdots l_{n-1}$ et $m' = l'_0 \cdots l'_{n-1}$. Cela correspond au produit scalaire de la section 5.3.3 à un retournement près des lettres du second terme. Le dual $\mathcal{C}^{\perp_2}$ que nous définissons ici est donc obtenu en retournant tous les mots de $\mathcal{C}^{\perp}$. Le lemme 29 sur les dimensions reste donc valable.

Remarque importante : dans la notation polynômiale, $\langle m, m' \rangle_2$ est le coefficient du monôme X^{n-1} dans le produit $m(X)m'(X)$, qui reste inchangé dans $m(X)m'(X)_{\ [X^n-1]}$.

Lemme 39 *Soit $g(X)$ le polynôme générateur de $\mathcal{C}$ dans $(\mathbb{F}_q)^n$, alors $\mathcal{C}^{\perp_2}$ est le code cyclique engendré par $h(X)$, avec $g(X)h(X) = X^n - 1$.*

Cela tient de la remarque ci-dessus. Soit $\mathcal{C}'$ engendré par $h(X)$ et soient $m(X) = a(x)g(X) \in \mathcal{C}$ et $m'(X) = a'(X)h(X) \in \mathcal{C}'$. Comme $m(X)m'(X) = a(X)a'(X)g(X)h(X) = 0_{\ [X^n-1]}$, on a $\langle m, m' \rangle_2 = 0$, soit $\mathcal{C}' \subset \mathcal{C}^{\perp_2}$. Par ailleurs, d'après le lemme 29 $\dim \mathcal{C}^{\perp_2} = n - \dim \mathcal{C} = n - k$ et d'après le lemme 35 $\dim \mathcal{C}' = n - \text{d}^\text{o}\ h(X) = \text{d}^\text{o}\ g(X) = n - k$, on conclut $\mathcal{C}' = \mathcal{C}^{\perp_2}$. Pour avoir le dual $\mathcal{C}^{\perp}$ au sens de la section 5.3.3, il suffit de retourner tous les mots de $\mathcal{C}' = \mathcal{C}^{\perp_2}$.

Corollaire 3 *Si $g(X) \wedge h(X) = 1$, alors $\mathcal{C} \oplus \mathcal{C}^{\perp_2} = (\mathbb{F}_q)^n$.*

En effet, $g(X) \wedge h(X)$, "pgcd" de $g(X)$ et $h(X)$, est le polynôme générateur de $\mathcal{C} + \mathcal{C}^{\perp_2}$ (en utilisant l'égalité de Bezout). Si $g(X) \wedge h(X) = 1$, alors $\mathcal{C} + \mathcal{C}^{\perp_2} = (\mathbb{F}_q)^n$. En considérant les dimensions, on voit que l'intersection est réduite à $\{0\}$. Ce corollaire efface donc pour les codes cycliques la bizarrerie introduite par les vecteurs isotropes. La condition $g(X) \wedge h(X) = 1$ est vérifiée lorsque $n \wedge q = 1$, choix que l'on fait généralement pour la construction des codes cycliques (voir plus loin).

Exemple. Poursuivons l'exemple de la section précédente. On a $h(X) = (1 + X)(1 + X^2 + X^3) = 1 + X + X^2 + X^4$. La matrice

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

engendre $\mathcal{C}^{\perp_2}$. H est une matrice de contrôle pour $\mathcal{C}$ et le produit scalaire $\langle \cdot, \cdot \rangle_2$; autrement dit, sa retournée[2] H' est une matrice de contrôle pour $\mathcal{C}$ au sens $m \in \mathcal{C} \Leftrightarrow m(H')^T = 0$. Comme les colonnes de H' contiennent tous les triplets de taille 3, $\mathcal{C}$ est bien équivalent au code de Hamming $(7, 4)$.

6.6 Factorisation de $X^n - 1$ sur $\mathbb{F}_q[X]$

L'existence de codes cycliques repose sur la décomposition de $X^n - 1$ en un produit de polynômes irréductibles à coefficients dans le corps $\mathbb{F}_q$ (il suffit ensuite de choisir

[2] On signifie par là que H' s'obtient en lisant les lignes de H de droite à gauche.

un sous-ensemble de ces facteurs pour construire $g(X)$). Nous donnons ici un bagage algébrique minimal permettant de construire cette factorisation. Le lecteur intéressé trouvera plus de développements en annexe.

6.6.1 Structure du corps $\mathbb{F}_q$

On désigne par $\mathbb{F}_q$ le corps fini à q éléments. Il admet un sous-corps élémentaire $\mathbb{F}_p$, appelé son *sous-corps premier*, obtenu en considérant $\{0, 1, 1+1, 1+1+1, \ldots\}$, et qui est isomorphe à $\mathbb{Z}/p\mathbb{Z}$. Il s'ensuit que p est premier. Comme $\mathbb{F}_q$ est un $\mathbb{F}_p$ espace vectoriel, on a $q = p^m$: le cardinal d'un corps fini n'est donc pas quelconque. Noter la linéarité[3] de la puisance p dans $\mathbb{F}_p$: $(x+y)^p = x^p + y^p$, qui reste vraie dans $\mathbb{F}_q$. En fait, $\mathbb{F}_q$ est caractérisé par

$$x \in \mathbb{F}_q \quad \Leftrightarrow \quad x^q = x$$

L'*ordre* d'un élément de $\mathbb{F}_q$ est la plus petite puissance r telle que $x^r = 1$. Il existe dans $\mathbb{F}_q$ au-moins un élément α d'ordre $q-1$, que l'on appelle un *élément primitif* de $\mathbb{F}_q$, ce qui permet de montrer que

$$\mathbb{F}_q \;=\; \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$$

Cela induit l'unicité du corps $\mathbb{F}_q$ à un isomorphisme près.

6.6.2 Construction du surcorps $\mathbb{F}_{q^m}$

La construction du surcorps $\mathbb{F}_{q^m}$ est similaire à celle de $\mathbb{C}$ depuis $\mathbb{R}$:

$$\mathbb{C} \sim \mathbb{R}[X]/_{X^2+1}$$

On choisit donc un polynôme $f(X)$ de degré m dans $\mathbb{F}_q[X]$ et *irréductible*, c'est à dire non factorisable (en termes savants, il est générateur d'un idéal maximal de $\mathbb{F}_q[X]$). L'anneau quotient $\mathbb{F}_q[X]/f(X)$, formé des restes de la division euclidienne des polynômes par $f(X)$, est un corps[4] contenant $\mathbb{F}_q[X]$. C'est aussi un $\mathbb{F}_q$ espace vectoriel de base $(1, X, X^2, \ldots, X^{m-1})$, il contient donc q^m éléments.

$$\mathbb{F}_{q^m} \sim \mathbb{F}_q[X]/f(X)$$

$\mathbb{F}_{q^m}$ est le plus petit surcorps de $\mathbb{F}_q$ contenant une racine de $f(X)$; en effet, $X \in \mathbb{F}_{q^m}$ est racine de $f(X)$. En fait, $\mathbb{F}_{q^m}$ contient *toutes* les racines[5] de $f(X)$:

$$\beta \in \mathbb{F}_{q^m} \;:\; f(\beta) = 0 \;\Rightarrow\; \beta, \beta^q, \beta^{q^2}, \ldots, \beta^{q^{m-1}} \quad \text{sont racines de } f(X)$$

[3] Cela vient du fait que p divise les coefficients du binôme C_p^k. Il en découle le petit théorème de Fermat : $x^p \equiv x[p]$.

[4] c'est pour cela que $f(X)$ doit être irréductible

[5] Cela se montre en utilisant la linéarité de la puissance p, donc de la puissance q, sur tout corps contenant F_p, et le fait que $a^q = a, \forall a \in \mathbb{F}_q$. On tire $f(x^q) = f(x)^q, \forall x \in \mathbb{F}_{q^m}$.

On dit que ces éléments sont les *conjugués* de β dans $\mathbb{F}_{q^m}$. On notera que pour tout $\beta \in \mathbb{F}_{q^m}$, le polynôme

$$g(X) \;=\; \prod_{k=0}^{m}(X - \beta^{q^k})$$

est à coefficients dans $\mathbb{F}_q$, et non $\mathbb{F}_{q^m}$ (remarquer le parallèle avec $\mathbb{C}$). En particulier, pour α élément primitif de $\mathbb{F}_{q^m}$, on a $\beta = \alpha^i$, et donc $g(X)$ s'écrit

$$g(X) \;=\; \prod_{k=0}^{m}(X - \alpha^{i\cdot q^k})$$

Soit $M_\beta(X)$ le *polynôme minimal* de β dans $\mathbb{F}_q[X]$, c'est à dire le polynôme irréductible, normalisé[6] et admettant β comme racine, alors $g(X) = M_\beta(X)^l$. Le polynôme $M_\beta(X)$ est de degré d, de racines $\beta, \beta^q, \ldots, \beta^{q^{d-1}}$, et $ld = m$.

6.6.3 Factorisation de $X^n - 1$ dans $\mathbb{F}_q[X]$

On suppose que n et q sont premiers entre eux : $n \wedge q = 1$, et l'on cherche à décomposer $X^n - 1$ en polynômes irréductibles de $\mathbb{F}_q[X]$. La méthode est similaire à la factorisation de $X^n - 1$ sur $\mathbb{R}[X]$: on se place dans $\mathbb{C}$ pour faire apparaître toutes les racines de $X^n - 1$, puis on regroupe les racines conjuguées, qui donnent des polynômes de $\mathbb{R}[X]$. Les racines de $X^n - 1$ dans $\mathbb{C}$ sont en fait les puissances d'une *racine primitive* $n^{\text{ème}}$ de l'unité, par exemple $e^{2i\pi/n}$; cela reste vrai pour les corps finis.

Soit m le plus petit entier tel que

$$n \mid q^m - 1$$

On écrit $m = \mathcal{O}_q(n)$: m est l'*ordre multiplicatif de q modulo n*. Alors $\mathbb{F}_{q^m}$ est le plus petit surcorps de $\mathbb{F}_q$ contenant toutes les racines de $X^n - 1$. Certaines de ces racines sont d'ordre n ; ce sont des *racines primitives $n^{\text{èmes}}$ de l'unité*. Soit α une telle racine primitive, toute puissance de α est aussi racine de $X^n - 1$, or α étant d'ordre n, les éléments $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ sont tous distincs. Ils forment donc l'ensemble des racines de $X^n - 1$ dans $\mathbb{F}_{q^m}$. On peut les rassembler en groupes d'éléments conjugués ; les puissances de α rassemblées forment alors une *classe cyclotomique*. Les puissances i et j sont dans la même classe ssi il existe un k tel que $j = i \cdot q^k$, ce qui signifie que α^i et α^j sont conjugués.

Exemple. Supposons $q = 2$ et $n = 7$. Les classes cyclotomiques binaires modulo 7 sont alors

$$\begin{aligned} C_0 &= \{0\} \\ C_1 &= \{1, 2, 4\} \\ C_3 &= \{3, 6, 5\} \end{aligned}$$

[6] i.e. le coefficient du monôme de plus haut degré est 1

À chaque classe cyclotomique C_i, où i est le plus petit élément de la classe, correspond un polynôme irréductible de $\mathbb{F}_q[X]$, qui n'est autre que le polynôme minimal de α^i. Cela nous donne la factorisation de $X^n - 1$ en polynômes irréductibles

$$X^n - 1 \quad = \quad \prod_{\text{classes } C_i} M_{\alpha^i}(X)$$

Exemple. Pour $q = 2$ et $n = 7$, on trouve $m = 3$ car $n = 7$ divise $q^m - 1 = 2^3 - 1 = 7$. Il faut donc se placer dans $\mathbb{F}_{2^3} = \mathbb{F}_8$ pour factoriser $X^7 - 1$ en monômes. Soit α un élément primitif de $\mathbb{F}_8$. On a donc $\mathbb{F}_8 = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^6\}$. α est aussi une racine primitive $7^{\text{ème}}$ de l'unité, puisque les éléments de $\mathbb{F}_8$ sont les racines de $X_8 - X$. Formons les polynômes correspondant aux classes cyclotomiques vues plus haut :

$$\begin{aligned} P_0(X) &= X - \alpha^0 \\ P_1(X) &= (X - \alpha^1)(X - \alpha^2)(X - \alpha^4) \\ P_3(X) &= (X - \alpha^3)(X - \alpha^6)(X - \alpha^5) \end{aligned}$$

Pour remettre ces polynômes sous forme d'éléments de $\mathbb{F}_2[X]$, il faut les développer, en utilisant les règles de calcul[7] dans $\mathbb{F}_8$. On obtient alors

$$\begin{aligned} P_0(X) &= X + 1 \\ P_1(X) &= X^3 + X + 1 \\ P_3(X) &= X^3 + X^2 + 1 \end{aligned}$$

ce qui donne la factorisation en éléments irréductibles dans $\mathbb{F}_2[X]$:

$$X^7 + 1 \quad = \quad (X + 1)(X^3 + X + 1)(X^3 + X^2 + 1)$$

6.7 Exemples de codes cycliques classiques

Un code cyclique de longueur n, formé à partir des lettres du corps $\mathcal{A} = \mathbb{F}_q$, s'obtient en choisissant un polynôme générateur $g(X)$ de $\mathbb{F}_q[X]$ diviseur de $X^n - 1$. Pour choisir $g(X)$, on commence par décomposer $X^n - 1$ en facteurs irréductibles. Cela suppose de se placer dans un *surcorps* $\mathbb{F}_{q^m}$ de $\mathbb{F}_q$ qui contient toutes les racines de $X^n - 1$, puis de regrouper les monômes correspondant à des racines conjuguées. Le m caractérisant un tel surcorps est le plus petit entier tel que

$$n \mid q^m - 1$$

(on suppose toujours $n \wedge q = 1$). Les différents codes cycliques classiques correspondent à des choix particuliers de ces paramètres q, n et donc m.

[7] Il suffit en fait d'établir la table d'addition, sous la forme $\alpha^i + \alpha^j = \alpha^k$. Cette table peut s'établir à partir de la construction de $\mathbb{F}_8$ comme $\mathbb{F}_2[X]/P(X)$ où $P(X)$ est un polynôme irréductible de degré 3 dans $\mathbb{F}_2[X]$, par exemple $1 + X + X^3$.

6.7.1 Codes BCH (Bose-Chaudhury-Hocquenghem)

Ces codes, par un choix judicieux du polynôme générateur, garantissent une distance minimale au-moins égale à d_{BCH}. Ils permettent donc de corriger au-moins $(d_{BCH}-1)/2$ erreurs. La structure BCH n'impose pas d'autre contrainte sur le choix de n, q et m que $n \wedge q = 1$ et $n | q^m - 1$. Lorsque $n = q^m - 1$, on parle toutefois de *code BCH primitif*. Dans la suite, on appelle β une racine primitive $n^{\text{ème}}$ de l'unité dans $\mathbb{F}_{q^m}$.

Supposons que le polynôme $g(X)$, de degré $n-k$, admet pour racines $\alpha_1, \alpha_2, \ldots, \alpha_{n-k}$ dans $\mathbb{F}_{q^m}$. Un élément μ de $(\mathbb{F}_q)^n$ est alors un mot du code $\mathcal{C}$ ssi le polynôme $\mu(X)$ est un multiple de $g(X)$, c'est à dire s'il admet les α_i pour racines :

$$\mu \in \mathcal{C} \quad \Leftrightarrow \quad \mu(\alpha_i) = 0, \;\; i = 1 \ldots n-k$$

En d'autres termes, la matrice H suivante est une matrice de contrôle pour $\mathcal{C}$ (les calculs se font dans $\mathbb{F}_{q^m}$) :

$$H \;=\; \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha_{n-k} & \alpha_{n-k}^2 & \cdots & \alpha_{n-k}^{n-1} \end{pmatrix}$$

On a en effet $\mu \in \mathcal{C}$ ssi $\mu \cdot H^T = 0$. On a vu au lemme 28 (chapitre précédent) que le rang minimal des colonnes de H caractérise la distance minimale du code. On peut choisir les racines α_i de façon à rendre ce rang minimal élévé.

Lemme 40 *Si $g(X)$ admet $\beta^{i+1}, \beta^{i+2}, \ldots, \beta^{i+\delta-1}$ pour racines dans $\mathbb{F}_{q^m}$, alors il définit un code cyclique de distance minimale $d \geq \delta$.*

On va supposer $i = 0$ pour la preuve, et poser $\alpha_1 = \beta$, $\alpha_2 = \beta^2$, …, $\alpha_{\delta-1} = \beta^{\delta-1}$. Il faut montrer que tout choix de $\delta - 1$ colonnes de H définit une matrice M de rang plein. Prenons les colonnes correspondant aux puissances $j_1, j_2, \ldots, j_{\delta-1}$ des α_i, et ne conservons que les $\delta - 1$ premières lignes, il vient

$$M \;=\; \begin{pmatrix} \alpha_1^{j_1} & \alpha_1^{j_2} & \cdots & \alpha_1^{j_{\delta-1}} \\ \alpha_2^{j_1} & \alpha_2^{j_2} & \cdots & \alpha_2^{j_{\delta-1}} \\ \vdots & \vdots & & \vdots \\ \alpha_{\delta-1}^{j_1} & \alpha_{\delta-1}^{j_2} & \cdots & \alpha_{\delta-1}^{j_{\delta-1}} \end{pmatrix} = \begin{pmatrix} \beta^{j_1} & \beta^{j_2} & \ldots & \beta^{j_{\delta-1}} \\ \beta^{2j_1} & \beta^{2j_2} & \ldots & \beta^{2j_{\delta-1}} \\ \vdots & \vdots & & \vdots \\ \beta^{(\delta-1)j_1} & \beta^{(\delta-1)j_2} & \ldots & \beta^{(\delta-1)j_{\delta-1}} \end{pmatrix}$$

L'expression de droite indique que M est une matrice de Vandermonde, dont le déterminant vaut

$$\det(M) \;=\; \beta^{\sum_{k=1}^{\delta-1} j_k} \prod_{1 \leq u < v \leq \delta-1} \left(\beta^{j_v} - \beta^{j_u}\right) \;\neq\; 0 \qquad \text{cqfd}$$

Définition 37 *Le polynôme $g(X)$ de $\mathbb{F}_q[X]$ génère un code BCH de distance construite δ ssi $g(X)$ est le produit des polynômes minimaux dans $\mathbb{F}_q[X]$ de $\beta, \beta^2, \ldots, \beta^{\delta-1}$, ou plus généralement de $\beta^{i+1}, \beta^{i+2}, \ldots, \beta^{i+\delta-1}$. La distance BCH de $\mathcal{C}$, notée $d_{BCH}(\mathcal{C})$, est la plus grande distance construite que l'on peut associer à $\mathcal{C}$.*

6.7.2 Codes BCH primitifs binaires

On choisit ici $q = 2$ et $n = q^m - 1 = 2^m - 1$. Ces longueurs n "primitives" sont les plus employées en pratique. On appelle toujours β une racine primitive $n^{\text{ème}}$ de l'unité ; β correspond ici à un élément primitif de $\mathbb{F}_{q^m} = \mathbb{F}_{2^m}$.

On veut construire un code c-correcteur, donc de distance BCH $d_{BCH} \geq 2c + 1$. En utilisant le lemme 40 et la définition 37, on voit qu'il suffit de choisir un polynôme générateur $g(X)$ admettant $\beta, \beta^2, \ldots, \beta^{2c}$ comme racines. Soit $m_{\beta^i}(X)$ le polynôme minimal de β^i dans $\mathbb{F}_2[X]$. β^{2i}, conjugué de β^i, est une autre racine de $m_{\beta^i}(X)$. Il suffit donc de choisir $g(X)$ tel que $m_{\beta^i}(X)$ soit un de ses diviseurs pour $i = 1, 2, \ldots, c$. On peut prendre le ppcm

$$g(X) \quad = \quad \vee_{i=1}^{c} \, m_{\beta^i}(X)$$

qui comporte au plus c facteurs. Par ailleurs, le degré de chacun des $m_{\beta^i}(X)$ est inférieur[8] à m. On a donc

$$\mathrm{d}^{\mathrm{o}}\, g(X) \;\leq\; c \cdot m$$

En se souvenant que $\mathrm{d}^{\mathrm{o}}\, g(X) = n - k$, il vient que le nombre de bits utiles vérifie $k \geq n - cm = 2^m - 1 - cm$.

Lemme 41 *Il existe des codes BCH primitifs binaires c-correcteurs dont le nombre de bits utiles vérifie $k \geq 2^m - 1 - cm$ (construction ci-dessus).*

Il faut donc au plus m bits de redondance pour corriger une erreur, dans ces codes. La borne donnée par ce lemme est assez grossière. On peut facilement l'améliorer pour des choix de c et m particuliers.

Exemple. Prenons $n = 15 = 2^4 - 1$, donc $q = 2$ et $m = 4$. On a la factorisation en polynômes irréductibles suivante

$$\begin{aligned} X^{15} - 1 \quad &= \quad (X+1) \cdot (X^4 + X + 1) \cdot (X^4 + X^3 + X^2 + X + 1) \cdot \\ &\qquad (X^2 + X + 1) \cdot (X^4 + X^3 + 1) \\ &= \quad g_0(X) \cdot g_1(X) \cdot g_3(X) \cdot g_5(X) \cdot g_7(X) \end{aligned}$$

[8] Car $\mathbb{F}_2[X]/m_{\beta^i}(X)$ définit un corps de cardinal 2^δ où $\delta = \mathrm{d}^{\mathrm{o}}\, m_{\beta^i}(X)$. Ce corps contient toutes les racines de $m_{\beta^i}(X)$. C'est donc un sous-corps de $\mathbb{F}_{2^m}$, d'où nécessairement $\delta \leq m$. EN fait on a même $\delta | m$.

où les $g_i(X)$ correspondent aux classes cyclotomiques

$$\begin{aligned} C_0 &= \{0\} \\ C_1 &= \{1,2,4,8\} \\ C_3 &= \{3,6,12,9\} \\ C_5 &= \{5,10\} \\ C_7 &= \{7,14,13,11\} \end{aligned}$$

En choisissant $g(X) = g_1(X) \cdot g_3(X) = X^8+X^7+X^6+X^4+1$ on a donc $\beta^1, \beta^2, \ldots, \beta^4$ comme racines, d'où $d_{BCH} = 5$. On a $d = d_{BCH}$ car $g(X)$ est un mot du code et a pour poids 5. Ce $(15, 7, 5)$-code est 2-correcteur.

Exercice. Poursuivre l'exemple en construisant un code 3-correcteur.

6.7.3 Codes RS (Reed-Solomon)

Il s'agit d'une classe de codes très populaire, peut-être la plus utilisée en pratique (parmi les codes blocs). Il s'agit de codes BCH pour lesquels $m = 1$; on a donc $n = q - 1$. En d'autres termes, tout consiste à choisir le corps de référence $\mathbb{F}_q$. Le plus souvent, on choisit pour q une puissance de 2 : $q = 2^l$, de sorte que les éléments de $\mathbb{F}_q$ se représentent sous forme d'un l-uplet de bits. Cela permet un codage très simple à partir d'un train binaire : $n \cdot l = (2^l - 1)l$ bits donnent un mot de $n = 2^l - 1$ lettres de $\mathbb{F}_q$. On a en fait un isomorphisme de groupes additifs entre $\mathbb{F}_q$ et $(\mathbb{F}_2)^l$, qui facilite grandement les calculs dans $\mathbb{F}_q$ (voir annexe : calcul dans les corps finis).

Soit β un élément primitif de $\mathbb{F}_q$, donc $\mathbb{F}_q = \{0, 1, \beta, \beta^2, \ldots, \beta^{q-2}\}$, le polynôme générateur $g(X)$ est tout simplement un produit de monômes formé à partir de $n-k$ puissances consécutives[9] de β, par exemple

$$g(X) = \prod_{i=1}^{n-k} (X - \beta^i)$$

de sorte que la distance construite vérifie $d_{BCH} \geq n - k + 1$ par le lemme 40. On a vu par ailleurs que dans tous les cas, la distance minimale d'un code linéaire vérifie $d \leq n - k + 1$ (borne de Singleton, lemme 25). Donc

Lemme 42 *La distance minimale d'un code de Reed-Solomon est égale à sa distance BCH. Elle atteint par ailleurs la borne de Singleton $d = n - k + 1$: un code de Reed Solomon réalise donc la meilleure distance minimale possible, à n et k fixés.*

[9] Plus besoin de prendre des polynômes minimaux, car $X^n - 1$ se décompose complètement dans $\mathbb{F}_q[X]$.

6.7.4 Code de Golay

6.8 Méthodes de décodage

Chapter 7

Codes convolutifs

Chapter 8

Applications