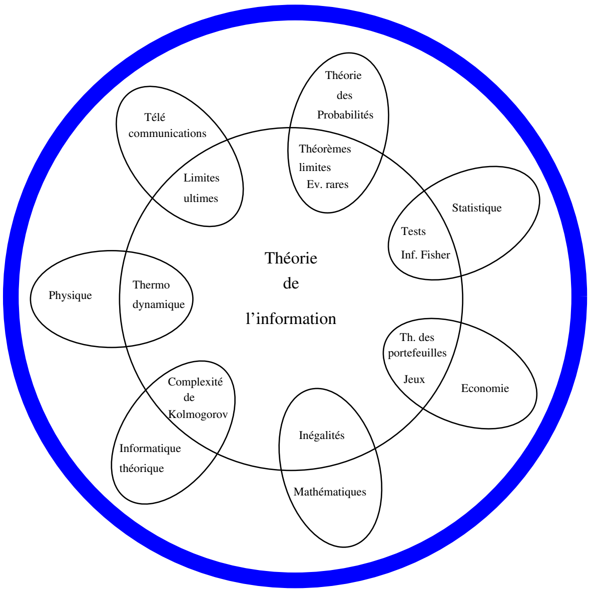
UNIVERSITÉ DE LIÈGE
FACULTÉ DES SCIENCES APPLIQUÉES

THÉORIE DE L'INFORMATION ET DU CODAGE



NOTES DE COURS

Louis WEHENKEL

SEPTEMBRE 2003.

Table des matières

1 INTRODUCTION

Comme son intitulé l'indique, ce cours est à vocation théorique et traite de deux sujets complémentaires : la notion d'information et les méthodes de codage de l'information. Ces deux sujets ont une relation très étroite, similaire à celle qui existe entre la théorie des systèmes et l'automatique. La théorie de l'information fournit essentiellement des outils de modélisation et d'analyse dans le domaine du traitement de l'information. La théorie du codage développe des techniques de synthèse, visant à concevoir des systèmes de stockage et de transmission de l'information fiables et/ou efficaces.

Mais il est également vrai que les répercussions de la théorie de l'information dépassent largement le seul domaine du codage, et, symétriquement, le codage des données fait appel à d'autres disciplines que la théorie de l'information. Dans cette introduction nous allons clarifier les questions abordées dans ces deux domaines très vastes et nous apportons aussi les précisions qui s'imposent sur la partie de cette matière que nous voulons couvrir dans ce cours ainsi que la démarche que nous adopterons pour y arriver.

La théorie de l'information donne des bases formelles et quantitatives à la notion d'information, de façon à ce que celle-ci devienne techniquement utilisable dans un certain nombre de disciplines qui traitent de l'information. Cette théorie fut conçue par Claude E. Shannon peu après la seconde guerre mondiale pour répondre à certaines interrogations fondamentales dans le domaine des techniques de communication. Comme le suggère l'intitulé de l'article fondateur de Shannon (*The mathematical theory of communication*, 1948), cette théorie peut être vue comme une discipline mathématique. Les trois principales questions auxquelles elle apporte une réponse sont les suivantes : (i) quelle est la limite ultime en matière de compression des données digitales réversible (réponse : l'entropie H de la source d'information); (ii) quel est le débit maximal de transmission fiable de ce type d'information sur un canal bruité (réponse: la capacité C du canal); (iii) sous quelles conditions un code de chiffrement est-il sûr (réponse : ...).

Le codage vise, quant à lui, à développer des méthodes pratiques (algorithmes) de représentation de l'information qui visent, certes, à atteindre les limites du possible définies par la théorie de l'information, mais qui répondent aussi à des préoccupations techniques, telles que vitesse de traitement, besoins réduits de mémoire, robustesse vis-à-vis des hypothèses de travail. Nous verrons que les différentes techniques de codage résultent essentiellement de différents compromis entre ces besoins antagonistes. Le codage est donc essentiellement une science appliquée, et suit de près les besoins pratiques et les moyens matériels qui sont à la disposition des ingénieurs. Notons que les langues parlées et écrites sont des méthodes de codage, inventées longtemps avant l'émergence de la théorie de l'information; d'autres méthodes ont été mises au point très récemment grâce à l'éclairage apporté par cette théorie et font maintenant partie des développements les plus importants de l'informatique.

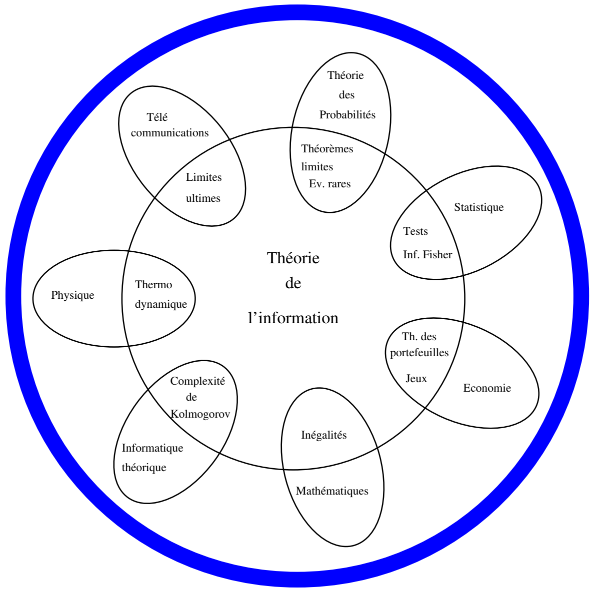


Figure 1.1. Relation de la théorie de l'information avec d'autres discipline scientifiques

Claude Shannon, qui est certainement un des esprits scientifiques les plus brillants de ce siècle, s'intéressait à de nombreuses questions fondamentales dans le domaine de l'informatique, et ses propres travaux de recherche dépassent de loin la théorie de l'information à proprement parler. Shannon a commencé sa carrière scientifique en montrant comment appliquer l'algèbre booléenne à la conception et à la minimisation de circuits logiques (à base de relais électromécaniques, à l'époque de ses travaux). Il est devenu célèbre à la publication de son article fondateur de la théorie de l'information, dont certains comparent l'importance avec celle de la théorie de la relativité générale d'Einstein. Bien que mathématicien hors pair, Shannon est aussi un ingénieur hautement créatif. Il a conçu de nombreuses machines "intelligentes" et notamment le premier programme de jeu d'échecs, dont les idées principales sont encore à la base des logiciels contemporains dans ce domaine.

Cependant, même si la théorie de l'information fut essentiellement conçue pour les besoins de l'informatique et des télécommunications, ces apports vont bien au-delà. En effet, on peut citer les contributions fondamentales en physique statistique (thermodynamique), en informatique théorique et en inférence statistique (complexité de Kolmogorov, théorie de l'apprentissage), dans le domaine de la génétique et de la neurophysiologie, et dans bien d'autres disciplines scientifiques. La figure 1.1 illustre, de façon non exhaustive, la relation entre la théorie de l'information et un certain nombre d'autres disciplines scientifiques, mettant en évidence les contributions les plus importantes de cette théorie. Comme on peut le voir ces contributions recouvrent un panel extrêmement vaste comprenant physique, informatique, mathématique et économie, tous domaines où la notion *d'information* joue un rôle capital.

Une autre caractéristique de cette théorie est que ses bases et la plupart de ses résultats théoriques fondamentaux furent ébauchés essentiellement par une seule personne (Claude Shannon) dans une période de maturation très courte. Il en résulte une théorie conceptuellement simple et élégante. Mais les limites prédites par Shannon n'ont pu être approchées de façon pratique qu'après une longue période de recherche (à laquelle d'ailleurs Shannon n'a pas participé directement). Dans le domaine de la compression de données les meilleures techniques ont été inventées seulement pendant les années 70-80 (codage arithmétique, algorithmes de Lempel-Ziv). Dans le domaine du codage de canal les résultats les plus spectaculaires sont encore plus récents (invention des turbo-codes au début des années 90). Nul doute que cette théorie continuera de jouer un rôle prédominant dans les techniques de l'information qui sont de plus en plus omniprésentes.

Dans le cadre de ce cours nécessairement introductif nous nous focaliserons essentiellement sur les questions qui ont des répercussions dans le domaine des communications ou de l'informatique. Nous suggérons au lecteur désireux d'en savoir plus sur le plan purement théorique de consulter le livre de T. M. Cover et J. A. Thomas [CT91], qui offre une très belle introduction à l'ensemble de la théorie de l'information. Par ailleurs, nous développerons la plupart des concepts et méthodes dans le cadre d'un monde où l'information est discrète et finie, ce qui nous semble justifié dans le cadre d'un cours introductif. Cependant, nous discuterons le cas échéant les interfaces avec le monde des signaux et systèmes physiques continus. A la fin de ce cours nous reviendrons sur cette distinction discret vs continu et nous analyserons à la lumière de la théorie de l'information pourquoi la représentation discrète de l'information a supplanté de manière quasi-universelle les systèmes analogiques.

Le cours a trois objectifs principaux, correspondant aux trois parties de ces notes.

Premièrement, nous voulons fournir aux étudiants les bases nécessaires à la mise en oeuvre d'approches probabilistes. La théorie de l'information est en effet fondée sur des modèles probabilistes et permet de donner une définition précise et quantitative aux notions d'incertitude, d'information et de dépendance statistique. Une fois bien assimilée, la théorie de l'information fournit un certain nombre de raisonnements intuitifs qui facilitent énormément l'analyse et la synthèse de modèles probabilistes. Ces modèles trouvent aujourd'hui un nombre croissant d'applications, que ce soit dans le domaine de l'informatique (par exemple en intelligence artificielle) ou celui de la théorie des systèmes complexes (systèmes stochastiques, réglages adaptatifs, fiabilité et sécurité de systèmes).

Le second objectif du cours concerne l'étude des grands théorèmes de la théorie de l'information. Ces théorèmes, énoncés et démontrés par Shannon lorsqu'il a fondé la théorie, définissent essentiellement les limites du possible en matière de stockage et de transmission de l'information, et sont dès lors fondamentaux dans le contexte de la mise au point des systèmes informatiques et des techniques de télécommunications. Cette étude passe par la modélisation des sources d'information et des canaux de communication, et débouche sur l'analyse des propriétés de ces modèles en ce qui concerne les limites de faisabilité en matière de codage efficace de l'information. Les notions d'efficacité qui sont étudiées ici concernent les performances de vitesse et de volume de stockage, et surtout de fiabilité face aux imperfections des systèmes physiques utilisés. Shannon a en effet montré que grâce au codage de l'information il est possible d'utiliser des systèmes physiques limités du point de vue capacité et fiabilité afin de stocker et de transmettre de l'information digitale de manière aussi fiable que souhaité. On peut difficilement surestimer l'importance de ces résultats dans le cadre du développement de l'informatique et des télécommunications de ces dernières décennies.

Le troisième objectif du cours est de donner un aperçu représentatif des diverses techniques de codage de données (compression de données, codes correcteurs d'erreurs, codes de chiffrement) qui sont aujourd'hui mises en oeuvre dans la plupart des applications. Cependant, ce domaine est trop vaste pour pouvoir être couvert dans son intégralité, et nous nous limiterons la plupart du temps à la discussion des méthodes les plus représentatives.

Enfin, un objectif omniprésent dans ce cours est de mettre en évidence les différentes questions en rapport avec le cours qui font aujourd'hui l'objet de nombreuses recherches. Nous souhaitons aussi donner envie aux étudiants d'en savoir plus sur les facettes de la théorie de l'information et du codage non abordées dans ce cours et nous consacrerons donc une partie du temps à évoquer un certain nombre de ces faces "cachées".

Ces notes de cours sont structurées en trois parties correspondant plus ou moins aux trois objectifs que nous venons de discuter.

La première partie (intitulée *Modélisation probabiliste de l'information*) fournit une introduction aux mesures d'information et au divers aspects du raisonnement probabiliste : quantification de la notion d'incertitude et de dépendance statistique, modèles probabilistes graphiques, raisonnement (inférence déductive) et apprentissage automatique (inférence inductive) à l'aide de ces modèles. Notre but ici est de donner un aperçu assez représentatif des applications modernes des méthodes probabilistes pour les informaticiens et électroniciens. Cependant, l'objet du cours n'est pas l'approfondissement des algorithmes d'inférence déductive et inductive qui sera laissé au soin d'enseignements plus spécialisés.

La seconde partie des notes est constituée d'une série de chapitres qui couvrent les principaux résultats théoriques de la théorie de l'information relatifs à la modélisation et à l'utilisation des sources d'information et des canaux de communication. Il s'agit incontestablement de la partie la plus difficile de la matière, mais aussi de celle dont les résultats portent le plus à conséquence. En particulier, les théorèmes de Shannon reposent essentiellement sur la loi des grands nombres et leur énoncé même est souvent mal compris. Afin de permettre aux étudiants de comprendre les raisonnements théoriques qui sont à la base de ces théorèmes, nous avons décidé d'inclure les démonstrations dans ces notes, même si elles sont ardues. Cependant, à nos yeux il est bien plus important de bien comprendre le sens des énoncés des théorèmes que d'apprendre ces démonstrations.

La troisième partie des notes couvre les différents aspects pratiques et théoriques des algorithmes de codage de l'information : compression de données digitales, quantification et compression de données analogiques (images, son), codes correcteurs et détecteurs d'erreurs. Logiquement, on devrait trouver dans cette partie également des éléments de cryptographie, mais cette matière est laissée au soin d'autres enseignements.

Les notes se terminent par un chapitre récapitulatif qui discute également de questions qui sortent du cadre strict des techniques de représentation et de transmission de l'information, mais qui font néanmoins partie des retombées de la théorie de l'information au sens large (complexité de Kolmogorov, raisonnements à base entropique en traitement du signal, gestion de portefeuilles, théorie des jeux. . .).

Enfin, nous mettons à la disposition des étudiants un certain nombre d'exercices collationnés à la fin de chaque chapitre, et quelques appendices (placés à la fin des chapitres qui y font appel) qui couvrent des bases mathématiques auxquelles on fait particulièrement appel dans le cadre de ce cours. Nous suggérons aux étudiants de prendre connaissance du contenu des ces appendices dès le début de la lecture de chaque chapitre, puis d'en faire usage seulement en cas de besoin réel.

Le cours oral est composé d'une partie théorique qui couvre essentiellement la matière des deux premières parties des notes et de certains chapitres de la troisième partie. Nous accordons une importance considérable aux travaux pratiques, nécessaires pour assimiler la matière vue au cours théorique. En ce qui concerne la première partie, il s'agit de répétitions sous forme d'exercices simples qui permettent d'utiliser et de compléter la matière reprise dans ces notes. Pour ce qui concerne la deuxième et la troisième partie, les travaux pratiques seront du type laboratoire informatique; les étudiants pourront étudier le comportement de divers modèles de sources et de canaux et comparer les performances de divers algorithmes de codage.

I MODELISATION PROBABILISTE DE L'INFORMATION

2 MOTIVATION

La théorie de l'information, telle qu'elle fut élaborée par C.E. Shannon peu après la fin de la seconde guerre mondiale, avait pour objet principal d'évaluer les performances limites ("optimales") des systèmes de télécommunications *en présence de perturbations aléatoires* (désignées par le terme générique de bruit). Elle se compare donc, par exemple, aux énoncés de la thermodynamique qui définissent le rendement limite des cycles thermiques.

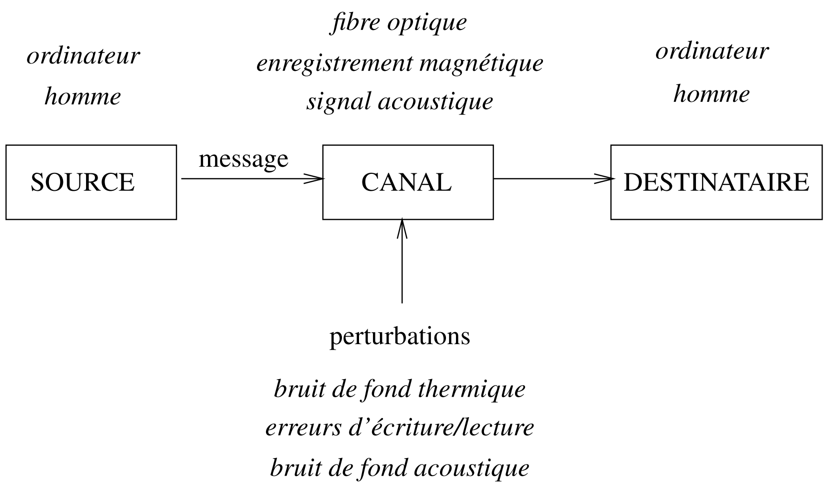


Figure 2.1. Schéma fondamental d'un système de communication

La figure 2.1 représente le schéma de communication désigné sous le nom de *paradigme de Shannon*. Une *source* engendre un *message* à l'intention d'un *destinataire*. La source et le destinataire sont deux entités séparées (éventuellement distantes) qui sont reliées par un *canal* qui est le support de la communication d'une part, mais qui d'autre part est le siège de *perturbations*. Les perturbations ont pour effet de créer une différence entre le message émis et celui qui est reçu. Ces perturbations sont de nature aléatoire, c'est-à-dire qu'il n'est pas possible (ni pour la source, ni pour le destinataire) de prévoir de manière certaine leur effet. La figure 2.1 indique quelques exemples physiques correspondant au paradigme de Shannon. Il est important de noter que le message émis par la source est également dans une certaine mesure imprévisible de la part du destinataire, car si le destinataire avait une connaissance totale du message a priori il n'y aurait pas besoin d'établir une communication.

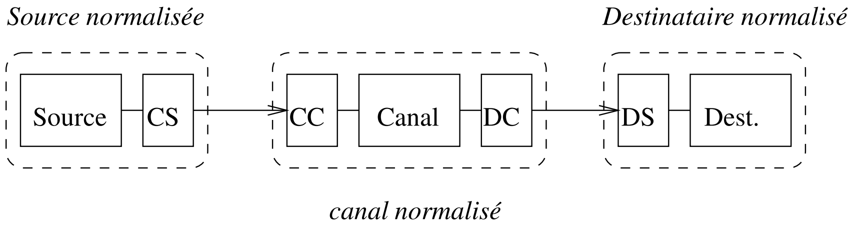


Figure 2.2. Chaîne de transmission avec codage/décodage source/canal

Les résultats fondamentaux de la théorie de l'information ont été établis par Shannon dès 1948 (dans l'optique télécommunications), et en particulier le fait capital, et totalement imprévu à l'époque, qu'il est possible de réaliser une transmission d'information exempte d'erreurs, malgré l'existence de bruit de fond, mais cela suppose une représentation appropriée de l'information (on utilisera dans la suite le terme de *codage*) et impose des contraintes sur le débit (ou la densité) de l'information transmise, qui dépendent des caractéristiques du canal. Mais il a fallu attendre les développements récents en informatique et en télécommunications pour voir apparaître des systèmes qui se rapprochent effectivement des performances limites annoncées par Shannon.

La théorie de l'information, telle qu'elle sera développée dans les chapitres qui suivent, donne tout d'abord une définition quantitative à la notion d'information. Nous montrerons que cette définition répond d'une certaine manière aux desiderata qu'on peut intuitivement imposer à une telle mesure. Ensuite la théorie met en évidence comment les limites de Shannon pourraient être réalisées, au moyen de codes appropriés. Cependant, ces développements ne donnent pas lieu à des codes pratiquement réalisables. Nous verrons donc dans la seconde partie des notes un certain nombre de codes pratiquement utilisables, et qui visent à s'approcher au mieux des limites de Shannon.

Nous verrons dans la suite que le schéma de communication "idéal" peut être schématisé tel que représenté à la figure 2.2. On y voit apparaître deux ensembles de codage CS et CC (et les éléments de décodage correspondants DS et DC) : CS code les messages émis par la source de façon à en éliminer (ou diminuer) la redondance; CC code les messages à l'entrée du canal en introduisant de la redondance sous une forme appropriée pour permettre l'utilisation du canal sans erreurs (ou avec un taux d'erreurs répondant aux spécifications techniques du système). Il est à noter que dans ce schéma les conversions physiques nécessaires à l'utilisation du canal sont implicitement représentées dans la boîte "canal". Notons que l'étude des canaux physiques et des techniques de conversion associées constitue une part importante des télécommunications, dont nous ne nous préoccupons cependant pas dans le cadre de ce cours.

Nous venons de voir que les deux éléments essentiels d'un schéma de communication (source et canal) ont un comportement aléatoire. L'outil de base en théorie de l'information est dès lors le calcul des probabilités. Dans le mesure où la plus grande partie de la théorie s'intéresse au monde digital, nous ferons intensivement appel aux probabilités discrètes, et seulement occasionnellement aux probabilités continues.

Cette première partie du cours est composée de quatre chapitres qui visent essentiellement à définir et illustrer les concepts de base utilisés pour la modélisation et l'analyse probabiliste du paradigme de Shannon (variables aléatoires discrètes, probabilités conditionnelles, indépendance, entropie, information mutuelle) et que nous utiliserons dans la seconde partie pour modéliser sources et canaux et expliquer les résultats fondamentaux établis par Shannon.

Nous commençons par un bref rappel de calcul de probabilités ce qui nous permettra aussi d'introduire les notations que nous utilisons dans ces notes. Ensuite, nous introduirons les notions d'entropie et d'information mutuelle de variables aléatoires discrètes ainsi que l'algèbre qui permet de manipuler ces notions efficacement (chapitre 4 des notes).

Enfin, nous terminerons cette première partie par deux chapitres qui discutent de l'exploitation de modèles probabilistes discrets (inférence déductive) et de leur obtention à partir de données observées (apprentissage automatique). Nous pourrons ainsi également mettre en évidence les liens entre la théorie de l'information et l'intelligence artificielle.

2.1 EXERCICES

Voici quelques exercices de réflexion. Nous reviendrons ultérieurement sur ces exercices et montrerons comment la théorie de l'information permet de répondre aux questions posées.

1. **Information et incertitude**

   Deux personnes jouent aux devinettes. La première lance une pièce à pile ou face sans montrer le résultat à la seconde. Pouvez vous mesurer le déficit en quantité d'information de la seconde personne sur la première ?

   Combien d'information la première personne communique-t-elle à la seconde lorsqu'elle lui révèle le côté sur lequel la pièce est tombée ? Quid, si au lieu de lancer la pièce une seule fois, on la lance deux fois de suite avant de révéler “simultanément” les deux résultats ?

   Quid, si on lance un dé à 4 faces ?

   Quid, si au lieu de lancer une pièce équilibrée, on lance une pièce qui retombe presque toujours sur la même face (en supposant que les deux personnes connaissent les caractéristiques de la pièce) ?

2. **Problème de pesée**

   On donne 12 billes, toutes de même poids à l'exception d'une seule qui est de poids différent (on ne dit pas si elle est plus légère ou plus lourde). On dispose d'une balance à deux plateaux qui permet de comparer les poids de deux ensembles de billes. A chaque pesée, on peut mettre un certain nombre de billes sur le plateau de gauche et le même nombre sur le plateau de droite : soit le plateau reste à l'horizontale, soit il penche à gauche, soit il penche à droite.

   On demande de définir une stratégie d'utilisation de la balance, dans le but d'identifier la bille anormale **et** de déterminer si elle est plus légère ou plus lourde que les autres. En essayant de résoudre ce problème, réfléchissez au questions suivantes :

   (a) Comment peut on mesurer l'information apportée par une pesée ?

   (b) Quel est le maximum d'information qu'on peut obtenir au moyen d'une seule pesée ?

   (c) Lorsque l'exercice de pesée est terminé (bille identifiée et réponse à la question concernant son poids relatif), combien d'information avez vous obtenu ?

   (d) Quel est le nombre minimal de pesées qui permet dans tous les cas de résoudre la devinette ?

   (e) Au fur et à mesure de la mise au point de la stratégie de pesée, vous pouvez dessiner un arbre dont les branches correspondront aux issues possibles. Quelle est la probabilité des trois issues possibles de la première pesée ?

   (f) Combien d'information obtient-on si au premier stade on pèse 6 billes contre 6 billes ? Combien d'information obtient-on si on pèse 4 billes contre 4 ?

3 CALCUL DES PROBABILITÉS

"La théorie des probabilités n'est rien d'autre que le bon sens réduit sous forme de calcul."
- Pierre Simon Laplace, 1819

Guide de lecture

La théorie de l'information traite de phénomènes imprévisibles (ou aléatoires) et repose essentiellement sur le calcul des probabilités. Ce chapitre fournit les bases dans le domaine du calcul des probabilités nécessaires dans le cadre de ce cours et introduit également les notations que nous utiliserons dans la suite.

La plus grande partie du cours se limite au traitement de modèles probabilistes discrets (variables aléatoires en nombre fini et ayant un nombre fini de valeurs possibles). En première lecture de ce chapitre il n'est donc pas nécessaire d'approfondir les questions relatives aux variables aléatoires continues.

Bien que ce chapitre constituera pour la plupart des étudiants un rappel, il est vivement recommandé de le lire le plus tôt possible et de s'entraîner en faisant les exercices (petites démonstrations laissées au soin du lecteur).

3.1 PROBABILITES VERSUS STATISTIQUE

La théorie des probabilités est une branche des mathématiques qui étudie les propriétés des structures (mathématiques) permettant de représenter les phénomènes où le "hasard" intervient.

Cette théorie permet donc de modéliser efficacement certains phénomènes aléatoires et d'en faire l'étude théorique. Elle fournit également une approche pour la formalisation du "raisonnement" en présence d'informations partielles et/ou contradictoires. Comme toute théorie mathématique, la théorie des probabilités est une science déductive, qui se base sur un certain nombre d'axiomes et utilise les techniques usuelles en mathématiques pour la démonstration de théorèmes. On y déduit donc des propriétés spécifiques, à partir d'hypothèses générales. Ses domaines d'application sont nombreux : la physique, l'intelligence artificielle, la théorie des systèmes, le traitement du signal, la statistique ... pour n'en citer que quelques uns.

La statistique, au sens le plus général, est une discipline qui consiste dans le recueil, le traitement et l'interprétation de données d'observations sur des systèmes physiques (réels ou simulés). En particulier, elle permet de construire des modèles (probabilistes ou non) qui représentent correctement la réalité mesurable du monde physique. Il s'agit d'une discipline faisant souvent appel au raisonnement inductif : à partir d'un certain nombre d'observations élémentaires on cherche à construire des lois générales qui "expliquent" ces observations. Etant donné ce caractère inductif, les résultats obtenus par la statistique peuvent être remis en question par de nouvelles observations, comme c'est le cas dans le domaine des sciences naturelles en général. Pour cette raison, une utilisation correcte de la statistique dans un domaine donné, va nécessairement de pair avec une bonne compréhension physique de ce domaine. Aussi, les résultats obtenus sont justifiés dans la mesure où ils sont opérationnels, et non pas parce qu'ils représenteraient la vérité absolue. Qu'on ne s'y trompe pas cependant, car l'utilisation des outils statistiques fait autant appel à la rigueur scientifique que les autres sciences expérimentales. Néanmoins, ces outils ne permettent de vérifier que la "plausibilité" (et non la "réalité") de la plupart des modèles utilisés par les ingénieurs et scientifiques de nombreuses disciplines. Etant donné la diversité des problèmes rencontrés en pratique, la statistique est un domaine extrêmement vaste dont l'appréhension d'ensemble nécessite du temps et de l'expérience pratique. Elle est basée sur le calcul des probabilités, qui sert d'outil de raisonnement, et fait appel à de nombreuses autres parties des mathématiques (analyse, algèbre, géométrie...).

Il y a donc une interdépendance forte entre les deux disciplines, mais également une différence fondamentale dans leur approche : déductive pour le calcul des probabilités; inductive pour la statistique.

Ce chapitre ne s'intéresse qu'au calcul des probabilités pour en rappeler les bases et les résultats les plus fondamentaux qui doivent être maîtrisés. Dans ce cours nous ne ferons appel qu'à des résultats élémentaires de statistique qui sont rappelés dans un appendice séparé. Pour une très bonne introduction aux probabilités et à la statistique nous recommandons vivement [Sap90]. Pour en savoir plus sur les fondements mathématiques du calcul des probabilités nous suggérons la lecture de la référence [Bil79].

Pour conclure cette introduction, insistons sur le fait que la séparation probabilités/statistique que nous faisons volontairement n'est pas justifiée d'un point de vue fondamental, mais bien pour des raisons pédagogiques. Nous commençons en quelque sorte par analyser quelques arbres sans nous préoccuper de la forêt. Parmi les différentes possibilités qui s'offrent pour aborder un domaine comme celui-ci, le choix que nous avons fait est celui d'une rupture minimale (mais nécessaire) avec la façon habituelle de présenter probabilités et statistique dans un même cours, et sans séparation claire.

Nous souhaitons ainsi limiter la confusion entre les aspects déductifs et inductifs complémentaires du calcul des probabilités et de la statistique. Au fur et à mesure de sa familiarisation avec les méthodes stochastiques, l'étudiant prendra conscience comment ces deux disciplines couvrent les deux pans du *raisonnement en présence d'informations incomplètes*. Enfin, pour terminer ces commentaires introductifs, notons que le domaine des méthodes stochastiques est étroitement apparenté à la logique et à la philosophie des sciences, même si l'approche probabiliste ne constitue pas la seule réponse possible aux problèmes abordés dans ces domaines.

3.2 NOTION DE PROBABILITE - INTERPRETATIONS

Dans cette section nous allons introduire, tout d'abord intuitivement puis plus formellement la notion de probabilité. Ensuite nous discuterons très brièvement de ses différentes interprétations logiques et physiques.

3.2.1 Intuitivement

Comme mentionné plus haut, le calcul des probabilités est un outil mathématique qui permet de représenter et de manipuler des situations/expériences dont l'issue est aléatoire et/ou au sujet desquelles on dispose de connaissances incomplètes/incertaines.

Une connaissance (c'est-à-dire une affirmation logique) est dite *incertaine* dans un contexte donné si, dans ce contexte, il est impossible aussi bien de réfuter sa véracité que de la prouver. La notion de probabilité permet d'ordonner de telles connaissances par ordre de *plausibilité* croissante, et de remettre à jour cet ordonnancement lorsque de nouvelles informations deviennent disponibles.

Expériences aléatoires. Une *expérience* est qualifiée d'aléatoire si on ne peut pas prévoir par avance son résultat, et donc si, répétée dans des conditions apparemment identiques, elle pourrait donner lieu à des *résultats* différents. Le calcul des probabilités permet de modéliser et d'analyser des expériences aléatoires.

Pour étudier une telle expérience on s'intéresse tout d'abord à l'univers de tous les résultats (ou objets) possibles : on note usuellement Ω cet ensemble fondamental, et ω un élément particulier de Ω, c'est-à-dire un résultat particulier parmi ceux possibles.

Exemple 1. Par exemple, si on s'intéresse au diagnostic médical, l'expérimentateur pourrait être un médecin particulier, et l'expérience le premier diagnostic de l'année (disons, le 2 janvier au matin, en l'an 2000). Nous pourrions alors définir l'ensemble Ω pour ce problème comme l'ensemble de tous les patients que ce médecin est susceptible de diagnostiquer (le médecin ne peut évidemment pas prévoir quel sera le patient particulier qui va se présenter devant lui).

Exemple 2. Un autre exemple intéressant concerne les réseaux informatiques. Plaçons nous en un noeud particulier du réseau Internet, et observons les messages par courrier électronique qui y transitent pendant une journée donnée. Un résultat particulier est alors la suite particulière de messages qui ont transité pendant la période d'observation. Avant d'avoir effectué l'expérience on ne peut évidemment pas prévoir quelle suite sera observée, et l'ensemble Ω est alors l'ensemble de toutes les suites de messages possibles pouvant transiter sur une journée, un ensemble certes très compliqué à caractériser mais néanmoins de taille finie.

Exemple 3. Un dernier exemple d'expérience aléatoire, plus directement pertinent dans le contexte de ce cours, concerne l'utilisation d'un système source-canal tel qu'illustré à la figure 2.1 (chapitre 2, page 7). Ici on pourrait définir un résultat de l'expérience comme l'observation d'un message émis par la source suivi de la déformation du message lors de la transmission par le canal et de la réception du message déformé par le destinataire. Dans la mesure ou nous ne chercherons pas à modéliser le fonctionnement interne du canal de communication, notre univers représenterait alors par exemple l'ensemble des couples possible du type "message émis par la source, message reçu par le destinataire" (nous appellerons un tel couple une utilisation du canal). Le calcul de probabilités pourrait alors être utilisé pour déterminer la fiabilité et l'efficacité du système de communications en fonction du système de codage utilisé.

Il faut noter que l'univers est défini en fonction de l'objectif particulier poursuivi. Ainsi, dans le premier exemple ci-dessus on aurait pu définir l'univers comme étant l'ensemble des maladies diagnostiquées par le médecin, ou encore l'ensemble des médicaments prescrits. Dans le second exemple, on aurait pu s'intéresser uniquement à l'expéditeur des messages et définir le résultat de l'expérience comme étant l'ensemble des adresses email des expéditeurs ayant envoyé au moins un message email pendant la journée : l'univers serait alors l'ensemble de tous les sous-ensembles d'expéditeurs possibles de messages électroniques susceptibles de transiter par le noeud. Enfin, dans le troisième exemple, nous aurions pu nous intéresser seulement aux messages émis par la source (nous verrons qu'en matière de compression de données c'est ce qui compte).

Evénements. Dans la terminologie usuelle, un **événement** désigne une assertion logique vérifiable relative au résultat d'une expérience. Ainsi, dans le cadre de notre troisième exemple l'assertion logique suivante

le message reçu est différent du message envoyé

définit un événement. Cette assertion logique est soit vraie soit fausse, et définit en fait un sous-ensemble des utilisations possibles du canal. Nous allons noter un tel ensemble $A \subset \Omega$.

Un autre événement correspondrait à l'assertion logique suivante

Le message émis comporte moins de cent mots

auquel on peut également associer un sous-ensemble $B \subset \Omega$, a priori différent de A.

A tout événement correspond donc une assertion logique à laquelle on peut faire correspondre un sous-ensemble de Ω. En particulier, à tout $\omega \in \Omega$ on peut associer un **événement élémentaire** correspondant au singleton $\{\omega\}$.

Probabilités. La notion de probabilité qui sera formalisée ci-dessous est une fonction qui mesure l'importance (la vraisemblance, ou la crédibilité) des événements : elle associe à un événement un nombre positif (entre 0 et 1) qui représente le degré de certitude qu'on peut associer à celui-ci a priori. Il traduit l'état de connaissance dans lequel on se trouve avant de réaliser une expérience.

Notons que si l'univers comprend un nombre fini d'éléments, les événements sont forcément des ensembles finis. Dans ce cas, un événement auquel on associe une probabilité égale à un est un événement dit certain (on est certain qu'il se réalisera); symétriquement, un événement auquel on associe une probabilité nulle est un événement impossible : on est certain qu'il ne se réalisera pas. Ces deux cas extrêmes sont les limites où le raisonnement probabiliste rejoint la logique classique : en ce qui nous concerne la partie intéressante concerne cependant tous les événements auxquels on associe des probabilités intermédiaires. La mesure de probabilité permet de trier l'ensemble des événements par ordre croissant de leur probabilité a priori. Le calcul des probabilités permet de s'assurer que les raisonnements effectués à l'aide de ces nombres restent cohérents, d'une part, d'autre part, il permet de mettre à jour les probabilités en fonction des informations obtenues.

Remarque sur la notion d'expérience répétable. Dans le monde réel, il est difficile d'imaginer des expériences qui puissent être répétées dans des conditions parfaitement identiques, l'aspect aléatoire vient alors du fait qu'il n'est pas possible de quantifier les variations des conditions d'une réalisation de l'expérience à la suivante. Cet aspect aléatoire peut être négligeable ou non, et ce n'est que dans ce dernier cas que l'utilisation du calcul de probabilités se justifie.

Par ailleurs, d'un point de vue strictement logique certaines expériences ne peuvent pas en principe être répétées : le 2 janvier 2000 au matin il n'y aura qu'un seul patient qui sera le premier et on ne peut donc pas répéter cette expérience. De même, au cours d'une journée donnée un seul ensemble de messages transitera en un noeud donné d'Internet. Le caractère non répétable d'un expérience n'empêche pas que les assertions logiques concernant de telles situations puissent être entachées d'incertitudes, soit parce qu'il s'agit de prédire l'avenir d'un système complexe, soit parce que la personne (ou l'ordinateur) qui effectue le raisonnement n'est pas complètement informé de tous les éléments pertinents concernant la situation.

Il est cependant souvent possible de supposer que les propriétés de certaines expériences ne changent pas au cours du temps : on peut alors répéter (éventuellement indéfiniment) cette expérience. Par exemple, on peut supposer qu'un dé ne s'use pas au cours du temps et que le résultat d'une expérience de lancer de dé ne dépend pas du temps, ou du nombre de fois qu'il a déjà été lancé. Similairement, lorsqu'on utilise plusieurs fois de suite un canal de communication pour transmettre de l'information, on peut supposer que ni la source ni le canal ne modifient leur comportement au fil du temps et que donc l'ensemble des résultats possibles ne change pas d'une fois à la suivante et que les probabilités des événements restent constantes.

La notion d'expérience répétable est donc en soi une vue de l'esprit, c'est-à-dire une abstraction qui ne se réalise jamais parfaitement en pratique : il n'est pas possible d'observer un système physique sans le perturber. Nous verrons plus loin que cette abstraction est souvent vérifiée approximativement et est à la base d'une grande partie des statistiques. Nous allons pour le moment au moins admettre qu'une expérience peut être répétée. Nous discuterons à la section 3.2.3 plus finement pourquoi il n'en est pas toujours ainsi, et pourquoi il est néanmoins intéressant de se servir du calcul des probabilités lorsque ce n'est pas le cas.

3.2.2 Formellement

3.2.2.1 Espace probabilisé.

La définition formelle d'un espace probabilisé est la suivante.

Définition : espace probabilisé

Un espace probabilisé est défini par un triplet $(\Omega, \mathcal{E}, P(\cdot))$ où

- Ω représente l'ensemble de résultats possibles d'une expérience aléatoire,
- $\mathcal{E}$ un σ-algèbre d'assertions logiques relatives aux résultats de l'expérience, et
- $P(\cdot)$ une loi de probabilité qui associe à chaque assertion logique A de $\mathcal{E}$ un nombre $P(A) \in [0; 1]$.

Nous allons préciser ci-dessous les notions de σ-algèbre d'événements et de loi de probabilité.

Notations. Ci-dessous nous utiliserons

- des lettres minuscules grecques $(\alpha, \beta, \ldots)$ pour désigner les éléments de Ω
- des lettres majuscules latines (p.ex. $A, B, \ldots$) pour désigner des sous-ensembles de Ω, et 2^{Ω} désigne alors l'ensemble de tous les sous-ensembles de Ω
- des lettres rondes (p.ex. $\mathcal{A}, \mathcal{B}, \ldots$) pour désigner des parties de 2^{Ω}, c'est-à-dire des ensembles de sous-ensembles de Ω.

- $f(\cdot), g(\cdot), \ldots$ pour désigner une fonction définie sur (une partie de) Ω,
- $F(\cdot), G(\cdot), \ldots$ pour désigner des fonctions définies sur (une partie de) 2^Ω.

Enfin, nous désignerons par $\neg A$ le complémentaire dans Ω de A.

3.2.2.2 σ-Algèbre des événements.

Un σ-algèbre est un ensemble de parties de Ω qui correspondent à toutes les assertions logiques dont on se donne le droit de discuter dans le cadre d'un problème. Cet ensemble doit vérifier un certain nombre de propriétés de complétude qui assurent que les combinaisons logiques usuelles d'assertions figurant dans cet ensemble (et, ou, négation) produisent encore des assertions logiques de l'ensemble.

Définition : σ-algèbre

Un σ-algèbre $\mathcal{E}$ d'événements[1] défini sur un univers Ω est une partie de 2^Ω (i.e. un ensemble de sous-ensembles de Ω) qui vérifie les propriétés suivantes :

1. $\Omega \in \mathcal{E}$;
2. $A \in \mathcal{E} \Rightarrow \neg A \in \mathcal{E}$;
3. $\forall A_1, A_2, \ldots \in \mathcal{E}$ (en nombre fini ou dénombrable[2]) : $\bigcup_i A_i \in \mathcal{E}$.

Les éléments de $\mathcal{E}$ sont désignés par le terme d'événements. Il s'agit des (seules) parties de Ω auxquelles nous conférons le statut particulier de partie "probabilisée", comme nous le verrons ci-dessous. Deux événements A et B sont dits *incompatibles* si $A \cap B = \emptyset$.

Remarques.

1. Les deux premières propriétés ci-dessus impliquent que l'ensemble vide (désigné par $\emptyset$) fait nécessairement partie de tout σ-algèbre d'événements.
2. La seconde et la troisième propriété impliquent également que $\bigcap_i A_i \in \mathcal{E}$.
3. $\{\emptyset, \Omega\}$ est un σ-algèbre d'événements : c'est le plus petit de tous.
4. 2^Ω est un σ-algèbre d'événements : c'est le plus grand de tous.
5. Si Ω est un ensemble fini, alors $\mathcal{E}$ l'est également.
6. Par contre, si Ω est infini (dénombrable ou non), $\mathcal{E}$ peut être non-dénombrable, dénombrable, et même fini.

3.2.2.3 Système complet d'événements.

Définition : système complet d'événements.

Une partie $\mathcal{A} = \{A_1, \ldots, A_n\} \subset \mathcal{E}$ forme un système complet d'événements

- si $\forall i \neq j\ : A_i \cap A_j = \emptyset$ (les A_i sont incompatibles deux à deux)
- et si $\bigcup_i^n A_i = \Omega$ (ils couvrent Ω).

On dit aussi que les A_i forment une partition de Ω, et on supposera la plupart du temps que tous les A_i sont non-vides. Nous verrons qu'un système complet d'événements correspond à une variable aléatoire discrète.

[1]Le terme consacré est en réalité "σ-algèbre de Boole" ou "tribu". Le terme "σ-algèbre" est normalement réservé au cas où la troisième propriété est relaxée à l'union finie. Cependant, dans la suite nous utiliserons la plupart du temps simplement le terme "σ-algèbre" étant entendu que dans le cas infini il faut comprendre σ-algèbre de Boole.

[2]Dorénavant nous utiliserons la notation $A_1, A_2, \ldots$ pour désigner une suite dénombrable (éventuellement finie) d'ensembles.

Exemple. Dans le cadre de notre schéma de communications on peut définir les 26 événements suivants

- A : l'ensemble des réalisations dont le message source commence par la lettre a,
- B : l'ensemble des réalisations dont le message source commence par la lettre b,
- ...
- Z : l'ensemble des réalisations dont le message source commence par la lettre z.

En supposant que les messages source sont écrits à l'aide des 27 symboles (26 lettres de l'alphabet plus l'espace) et qu'un message source ne peut pas commencer par un espace, cet ensemble forme un système complet d'événements.

3.2.2.4 Probabilités.

Le statut particulier des événements (par rapport aux parties de Ω qui ne seraient pas des événements) est qu'il est possible de leur attribuer une probabilité, c'est-à-dire un nombre positif compris entre 0 et 1, qui doit répondre aux axiomes suivants.

Axiomes de Kolmogorov

On appelle probabilité sur $(\Omega, \mathcal{E})$ (ou loi de probabilité) une fonction $P(\cdot)$ définie sur $\mathcal{E}$ telle que :

1. $P(A) \in [0, 1], \forall A \in \mathcal{E}$;
2. $P(\Omega) = 1$;
3. $\forall A_1, A_2, \ldots \in \mathcal{E}$ incompatibles : $P(\bigcup_i A_i) = \sum_i P(A_i)$.

Discussion.

On voit que l'utilisation du calcul des probabilités passe par trois étapes successives de modélisation : définition de l'univers Ω, choix d'un σ-algèbre d'événements $\mathcal{E}$, et enfin quantification par le choix de la mesure de probabilité. Les propriétés de base qui sont requises pour que ces trois opérations donnent lieu à un ensemble cohérent (c'est-à-dire qui permet de faire des raisonnements cohérents) résultent du fait que $\mathcal{E}$ est σ-algèbre et que $P(\cdot)$ satisfait les axiomes de Kolmogorov.

On constate également que le calcul des probabilités est compatible avec la logique classique (en tout cas, si Ω est fini). Il suffit de considérer le cas particulier où $P(\cdot)$ est définie sur $\{0, 1\}$, et associer à la valeur 1 la valeur de vérité "vrai" et à 0 la valeur "faux".

3.2.2.5 Propriétés remarquables.

Des axiomes de Kolmogorov on peut déduire immédiatement les propriétés suivantes (qu'on démontrera à titre d'exercice, en se restreignant au cas où Ω est fini).

1. $P(\emptyset) = 0$.
2. $P(\neg A) = 1 - P(A)$.
3. $A \subset B \Rightarrow P(A) \leq P(B)$.
4. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.
5. $P(\bigcup_i A_i) \leq \sum_i P(A_i)$.
6. $A_i \downarrow \emptyset \Rightarrow \lim_i P(A_i) = 0$ (voir note en bas de page [3])

On a également :

1. $P(A) = 1 \Rightarrow P(A \cup B) = 1, \forall B \in \mathcal{E}$.

[3] La notation $A_i \downarrow A$ désigne une suite d'ensembles, telle que $A_{i+1} \subset A_i$ et $\bigcap_i A_i = A$.

2. $P(A) = 1 \Rightarrow P(A \cap B) = P(B), \forall B \in \mathcal{E}$.
3. $P(A) = 0 \Rightarrow P(A \cap B) = 0, \forall B \in \mathcal{E}$.
4. $P(A) = 0 \Rightarrow P(A \cup B) = P(B), \forall B \in \mathcal{E}$.

3.2.2.6 Théorème des probabilités totales. Soit $\mathcal{B} = \{B_1, \ldots, B_n\}$ un système complet d'événements, alors

$$\forall A \in \mathcal{E} \ : P(A) = \sum_{i=1}^{n} P(A \cap B_i).$$

Exemple. Soit, pour notre système de communications le système complet d'événements $\{A, B, \ldots, Z\}$ décrit ci-dessus et soit E' l'événement qui désigne les réalisations correspondant à une erreur de transmission (message reçu différent du message émis). On a

$$P(E') = P(E' \cap A) + P(E' \cap B) + \cdots + P(E' \cap Z).$$

Le théorème des probabilités totales permet de décomposer un problème en sous-problèmes : ici la détermination du taux d'erreurs en 26 problèmes de détermination du taux d'erreurs dans des situations spécifiques.

Remarque. Certains auteurs définissent un système complet d'événements de façon légèrement différente de celle que nous avons adoptée ci-dessus. Au lieu d'exiger que $\bigcup_i^n A_i = \Omega$ ils imposent la condition plus faible $P(\bigcup_i^n A_i) = 1$. Ce type de système obéit également au théorème des probabilités totales.

D'ailleurs, on peut évidemment compléter un tel système avec $A_{n+1} = \neg \bigcup_i^n A_i$, avec $P(A_{n+1}) = 0$.

3.2.3 Différentes interprétations de la notion de probabilité

Il faut faire la distinction entre la formulation mathématique d'une théorie et l'utilisation que nous en faisons pour étudier des problèmes du monde réel qui nous entoure, c'est-à-dire son interprétation. Ceci est particulièrement vrai pour une théorie telle que le calcul des probabilités qui vise entre autres à modéliser une certaine forme du raisonnement humain, et qui s'adresse à des problèmes où l'incertitude joue un rôle fondamental, c'est-à-dire des problèmes où il pourrait être difficile de valider la théorie. En particulier, la théorie ne nous aide pas lorsqu'il s'agit de définir en pratique la loi de probabilité à associer aux événements choisis.

En réalité, depuis son origine, le calcul des probabilités a donné lieu à des débats intenses entre scientifiques, logiciens, physiciens, philosophes, en ce qui concerne la ou les interprétations physiques à donner à la notion même de probabilité. Ces débats sont encore d'actualité, et le resteront certainement encore longtemps; c'est la raison pour laquelle nous voulons mettre en évidence ici les différents points de vue qui s'opposent dans ce débat d'idées.

3.2.3.1 Le point de vue objectiviste.

La vision classique. La vision classique est héritée des jeux de hasard. Dans cette vision Ω est en général fini, et on considère alors comme σ-algèbre d'événements 2^{Ω}, fini lui aussi. Nous décrivons en détail la démarche adoptée pour alors définir la mesure de probabilité, car cette démarche est également utilisée dans la conception subjectiviste.

Dans ce cas, il suffit d'attribuer des probabilités aux événements élémentaires $P(\omega), \forall w \in \Omega$; les probabilités des autres événements s'en déduisent par application du troisième axiome de Kolmogorov. En particulier , on aura $P(\Omega) = \sum_{\omega} P(\omega)$ ce qui en vertu du second axiome de Kolmogorov impose évidemment que $\sum_{\omega} P(\omega) = 1$.

Sous cette contrainte, la vision classique impose des arguments de symétrie, posant que les événements élémentaires sont équiprobables. Par conséquent, $P(\omega) = |\Omega|^{-1}$ (où $|\Omega|$ désigne la taille finie de l'univers).

C'est cette démarche qui conduit à associer aux 6 faces d'un dé "parfait", une probabilité de $\frac{1}{6}$.

La principale faiblesse de cette approche est qu'elle repose sur un postulat de symétrie idéal (donc irréalisable en pratique) et ne permet pas la remise en question des probabilités en fonction d'informations supplémentaires (obtenues par exemple en effectuant des expériences de lancer de dé). Une autre faiblesse est que cette approche ne s'étend pas au cas ou Ω est non-dénombrable (voir à ce sujet la discussion au §3.2.4).

La vision fréquentiste. Elle repose sur la loi des grands nombres et sur une autre idéalisation, à savoir la notion d'expérience indéfiniment répétable dans les mêmes conditions. La loi des grand nombres assure en effet que dans une telle expérience la fréquence relative observée d'un événement converge vers la probabilité de celui-ci.

Notons que cette vision est aussi appelée la vision "orthodoxe" : toute comme dans la vision classique, la notion de probabilité est supposée définie de façon unique (c'est-à-dire indépendamment de l'observateur), et tous les observateurs doivent se soumettre à une expérience similaire pour en déterminer la valeur.

Il est clair que la procédure expérimentale n'est pas pratiquement réalisable. D'autre part, elle n'autorise pas l'utilisation du calcul des probabilités pour raisonner sur des événements incertains mais non répétables (et en pratique, aucun événement n'est parfaitement répétable). Enfin, elle est basée sur un cercle vicieux logique : cette définition repose sur la loi des grands nombres qui elle-même suppose déjà défini le concept de probabilité.

3.2.3.2 Le point de vue subjectiviste.

Les faiblesses des deux approches précédentes et le fait que d'un point de vue logique il soit souhaitable de permettre la remise en question de la probabilité d'un événement suite à l'obtention de nouvelles informations (par exemple, si nous apprenons que le dé est imparfait) conduisent à nier l'existence de la notion de probabilité "objective".

Ainsi, dans la conception subjectiviste on modélise l'état de connaissance d'un observateur. On peut alors argumenter que pour être cohérent avec lui-même, un observateur doit assigner des probabilités aux événements qui respectent les axiomes de Kolmogorov, mais, différents observateurs, ayant éventuellement des connaissances différentes, peuvent aboutir à des assignations différentes. De plus, un même observateur peut remettre à jour ces probabilités lorsque de nouvelles informations se présentent.

Mesure d'incertitude. La probabilité objective n'existe pas et n'est donc pas une grandeur mesurable; la probabilité subjective est simplement une mesure d'incertitude, qui peut varier avec les circonstances et avec l'observateur.

Puisque la notion d'expérience répétable n'est plus nécessaire, on peut étendre le domaine d'application du calcul des probabilités aux événements non répétables, c'est-à-dire au raisonnement en présence d'incertitudes (par exemple en intelligence artificielle, pour modéliser le raisonnement humain).

La vision bayesienne. Nous reviendrons plus loin sur cette approche, après avoir développé le calcul des probabilités. Pour le moment, contentons nous d'indiquer que cette approche consiste à attribuer des probabilités à tout ce qui est incertain. En particulier, cette approche consiste à attribuer des lois de probabilités aux probabilités des événements, si les informations disponibles ne sont pas suffisantes pour déterminer leurs valeurs exactes.

Ainsi, en présence d'un problème de jeu de "pile ou face", un bayesien va commencer par admettre qu'il ne connaît pas suffisamment bien la pièce pour fixer a priori la probabilité de "pile". En d'autre mots, il admet avoir une incertitude sur la valeur de cette probabilité, qu'il va modéliser par une (meta)loi de probabilités. Ensuite, il va utiliser cette loi de probabilités pour faire des prédictions, et si des expériences sont effectuées (par exemple des lancers de pièce) il va utiliser le calcul des probabilités (la formule de Bayes) pour remettre à jour la valeur des méta-probabilités en fonction de l'issue de l'expérience.

Il faut remarquer que cette approche n'est pas non plus entièrement satisfaisante (au grand dam de ses défenseurs) puisqu'il reste une phase arbitraire qui consiste à choisir les méta-probabilités. Signalons simplement que certains arguments de symétrie et d'"esthétique" sont utilisés par les bayesiens pour fixer de façon "objective" les méta-probabilités...

Cependant, sans prendre parti disons qu'il nous semble que l'approche bayesiennne présente une certaine souplesse qui va de pair avec la démarche scientifique.

3.2.4 Ensembles infinis, voire non-dénombrables

Pour terminer ces commentaires philosophiques, et avant de nous attaquer au coeur du calcul des probabilités, nous voulons faire ici quelques remarques générales sur la nécessité de pouvoir manipuler des lois de probabilités définies sur des univers de taille infinie ou dénombrables.

Lorsque l'ensemble Ω est fini, l'algèbre des événements l'est également. Par contre, lorsque Ω est infini, il est possible d'y définir des algèbres d'événements finis, dénombrables ou non-dénombrables. Par exemple, si Ω est infini mais reste dénombrable (c'est-à-dire peut être mis en bijection avec l'ensemble $\mathbb{N}$ des entiers naturels), l'algèbre complet est non-dénombrable (il peut être mis en bijection avec l'ensemble $\mathbb{R}$).

Dans la pratique, l'application du calcul des probabilités aux ensembles infinis peut conduire à un certain nombre de difficultés conceptuelles, dues aux passages à la limite. En particulier, un événement de probabilité nulle n'est pas nécessairement impossible, et corrolairement un événement de probabilité égale à un n'est pas nécessairement certain. Par exemple, si nous prenons comme univers l'intervalle $]0, 1]$ de la droite réelle muni de l'algèbre induit par les semi-intervalles $]a, b]$ ($a < b \in [0, 1]$) [4] et muni de la loi de probabilité uniforme qui associe à un intervalle $]a, b]$ la probabilité $b - a$, les événements élementaires de $]0, 1]$ sont des singletons de probabilité nulle et non moins possibles. Dans de telles situations il faut utiliser quelques précautions oratoires et parler d'événements presque impossibles ou presque certains.

D'un point de vue conceptuel, il est cependant important de se souvenir que ces ensembles infinis sont des constructions mathématiques obtenues par passage à la limite sur des ensembles finis. Le langage mathématique associé à ces ensembles permet d'écrire de façon synthétique des propriétés qui sont vraies pour les ensembles finis et qui le restent lors du passage à la limite. Mais, ce langage ne doit pas changer la signification de propriétés, ni nous induire en erreur.

D'un point de vue pratique, on peut donc adopter le point de vue que le monde tel qu'il est accessible à l'expérimentation physique est essentiellement fini (c'est d'ailleurs évident en ce qui concerne le monde de l'informatique digitale). On pourrait donc parfaitement justifier une approche qui consisterait à développer les théories sur base de modélisations par ensembles finis, et qui expliciterait les passages à la limite sur les résultats plutôt que sur les concepts de départ. On pourrait alors se débarasser des difficultés engendrées par l'analyse moderne (calcul infinitésimal, théorie de la mesure, des distributions...) au prix d'une lourdeur d'écriture accrue (et souvent excessive) d'un certain nombre de propriétés et de raisonnements.

Nous pensons que l'utilisation de l'analyse classique est un outil mathématique qui est non seulement intéressant du point de vue conceptuel, mais également justifié par son caractère opérationnel. Cependant, la compréhension des principes de base importants dans le domaine du calcul de probabilités peut par contre très bien se faire sans y faire appel à tour de bras. En clair, nous suggérons aux étudiants d'effectuer leurs raisonnements dans le cadre d'univers finis, afin de bien assimiler la signification mathématique et physique des principales notions. Une fois bien maîtrisé le cas fini, ils pourront ensuite se poser la question de savoir ce qui se passe lors du passage à la limite.

3.3 ELEMENTS DE BASE DU CALCUL DE PROBABILITES

3.3.1 Loi de probabilité conditionnelle

Partons d'un espace probabilisé $(\Omega, \mathcal{E}, P(\cdot))$, et supposons qu'un événement B soit réalisé, par exemple parce qu'une observation le confirme ou simplement à titre hypothétique.

Cherchons à savoir ce que devient alors la probabilité qu'un événement A quelconque soit également réalisé sous cette hypothèse. Nous allons noter cette quantité par $P(A|B)$.

Si A et B sont incompatibles il est clair que A devient impossible et $P(A|B) = 0$. Par contre, si $A \cap B \neq \emptyset$, alors la réalisation de A est compatible avec notre hypothèse, mais seulement les éléments de A qui sont également dans B nous intéressent. Si $A \cap B = B$ alors nous sommes certains que A se réalisera : $P(A|B) = 1$ dans ce cas. Tout se passe donc comme si nous avions restreint notre univers à l'événement B et que nous nous intéressions uniquement aux probabilités *relatives* des parties des événements situées dans B.

[4] C'est-à-dire le σ-algèbre de tous les ensembles qui peuvent s'exprimer sous la forme d'une union ou d'une intersection finie ou dénombrable de semi-intervalles. On appelle cet algèbre la tribu Borelienne.

Définition : loi de probabilité conditionnelle

Nous supposerons que B est de probabilité non-nulle (dans le cas fini il est ridicule d'envisager qu'un événement de probabilité nulle se soit réalisé), et nous *définissons* la probabilité conditionnelle de A sachant que B est réalisé par

$$P(A|B) \triangleq \frac{P(A \cap B)}{P(B)}. \tag{3.1}$$

Notons que $A \supset B \Rightarrow P(A|B) = 1$, mais (attention) la réciproque est fausse!

Il s'agit bien d'une loi de probabilité. En effet, elle vérifie les axiomes de Kolmogorov puisque :

- $A, B \in \mathcal{E} \Rightarrow A \cap B \in \mathcal{E}$ et donc $P(A|B)$ est bien définie sur $\mathcal{E}$.
- $P(A|B) \geq 0$.
- $A \cap B \subset B \Rightarrow P(A \cap B) \leq P(B) \Rightarrow P(A|B) \leq 1$.
- $P(\Omega|B) = 1$ (trivial).
- $P\left(\bigcup_i A_i|B\right) = \frac{P((\bigcup_i A_i)\cap B)}{P(B)} = \frac{P(\bigcup_i A_i \cap B)}{P(B)} = \sum_i \frac{P(A_i \cap B)}{P(B)} = \sum_i P(A_i|B)$, car si les A_i sont incompatibles les $A_i \cap B$ le sont également.

3.3.2 Notion d'indépendance d'événements

Définition : événements indépendants
On dit que A est indépendant de B si $P(A|B) = P(A)$, c'est-à-dire si le fait de savoir que B est réalisé ne change en rien la probabilité de A. On utilise souvent la notation

$$A \perp B \tag{3.2}$$

pour indiquer que A est indépendant de B.

Si $P(B) \in]0,1[$, on a A indépendant de B si, et seulement si, $P(A|B) = P(A|\neg B)$.
Suggestion : calculer alors $P(A)$ par le théorème des probabilités totales.

Définition : indépendance conditionnelle

Soient A, B, C trois événements, et $P(C) \neq 0$. Alors, on dit que A *est indépendant de B conditionnellement à C*, que l'on note par

$$A \perp B \mid C \tag{3.3}$$

si $P(A|B \cap C) = P(A|C)$.

Notons que

$$P(A|B \cap C) \triangleq \frac{P(A \cap (B \cap C))}{P(B \cap C)} = \frac{P((A \cap B) \cap C)}{P(C)} \frac{P(C)}{P(B \cap C)} = \frac{P(A \cap B|C)}{P(B|C)}.$$

L'indépendance conditionnelle de deux événements est donc équivalente à l'indépendance des ces deux événements vis-à-vis de la loi conditionnelle $P(\cdot|C)$.

Discussion. Il est important de remarquer que la loi de probabilité conditionnelle est bien définie sur l'algèbre de départ $\mathcal{E}$, et ceci bien que ses valeurs ne dépendent en fait que de probabilités de sous-ensembles de B.

Pour deux événements donnés A et B non indépendants on peut avoir soit $P(A|B) < P(A)$ ou $P(A|B) > P(A)$. Un événement peut donc devenir plus ou moins certain lorsqu'on dispose d'informations nouvelles.[5]

Dans le cas d'un univers fini tous les événements possibles sont de probabilité strictement positive, et définissent par conséquent une loi de probabilité conditionnelle. Nous avons déjà illustré un cas d'univers infini, où le fait qu'un événement se réalise n'implique pas nécessairement que sa probabilité a priori soit non-nulle; il est alors nécessaire de recourir à un artifice pour définir la notion de probabilité conditionnelle vis-à-vis de tels événements (passage à la limite).

3.3.3 Sur la notion d'indépendance

La notion d'indépendance est une notion centrale en théorie des probabilités. Aussi allons nous détailler les diverses propriétés immédiates qui découlent de sa définition. Nous supposerons ci-dessous que $P(A) \in]0,1[$ (resp. $P(B) \in]0,1[$), et dans le cas contraire nous dirons que A (resp. B) est un événement trivial. Nous laissons au lecteur le soin de vérifier dans quels cas (et comment) ces conditions peuvent être relaxées à des événements triviaux.

Propriétés positives. Nous demandons au lecteur de démontrer, à titre d'exercice immédiat, celles parmi les propriétés suivantes dont nous ne donnons pas la preuve.

- $\emptyset$ est indépendant de tout autre événement.
- Un événement de probabilité nulle est indépendant de tout autre événement :
  $P(A) = 0 \Rightarrow P(A \cap B) = 0 \Rightarrow P(A|B) = 0$.
- Tout événement est indépendant de Ω.
- Tout événement est indépendant de tout événement certain :
  $P(A) = 1 \Rightarrow P(A \cap B) = P(B)$ et donc $P(B|A) = P(B)$.
- "A indépendant de B" $\Leftrightarrow P(A \cap B) = P(A)P(B)$
  (conséquence directe de la définition).
- "A indépendant de B" $\Rightarrow$ "B indépendant de A".
- "A indépendant de B" $\Rightarrow$ "$\neg A$ indépendant de B".
- "A indépendant de B" $\Rightarrow$ "A indépendant de $\neg B$".
- "A indépendant de B" $\Rightarrow$ "$\neg A$ indépendant de $\neg B$".

On peut donc utiliser en lieu et place de la définition de l'indépendance la définition suivante.

Définition alternative de l'indépendance. Deux événements A et B sont indépendants si $P(A \cap B) = P(A)P(B)$.

Il est à noter que cette définition peut être étendue au cas où $P(A)$ et/ou $P(B)$ sont nulles, la propriété étant trivialement vérifiée dans ce cas (car $P(A \cap B) \leq \min\{P(A), P(B)\}$).

Propriétés négatives. L'assimilation de celles-ci sont aussi importantes pour la bonne compréhension de la notion d'indépendance.

Remarquons tout d'abord que des événements indépendants pour une loi de probabilité donnée peuvent très bien être non indépendants pour une autre loi de probabilité. En d'autres mots, la propriété d'indépendance dépend bien du choix de la loi de probabilité et pas seulement des propriétés ensemblistes.

[5] Cependant, dans le cours de théorie de l'information on montrera qu'en moyenne l'incertitude concernant une expérience aléatoire diminue, lorsqu'on utilise de l'information complémentaire.

Nous suggérons au lecteur de chercher des contre-exemples pour démontrer les propriétés négatives suivantes.

- Un événement quelconque non trivial n'est jamais indépendant de lui-même!
- A indépendant de B et B indépendant de $C \not\Rightarrow A$ indépendant de C
  (suggestion : prendre A et B non-triviaux indépendants et $C = A$).
- A dépendant de B et B dépendant de $C \not\Rightarrow A$ dépendant de C
  (suggestion : prendre A et C indépendants, et $B = A \cap C$ non trivial.)
- A indépendant de $B \not\Rightarrow A$ indépendant de B conditionnellement à C
  (suggestion : prendre comme exemple le double "pile ou face" avec une pièce équilibrée, comme événements A "face au premier lancer", B "face au second lancer", C "même issue aux deux lancers").

Indépendance mutuelle. On peut étendre la deuxième définition de l'indépendance au cas de n événements. On dira que les événements $A_1, A_2, \ldots, A_n$ sont mutuellement indépendants si pour toute partie I de l'ensemble des indices allant de 1 à n on a :

$$P\left(\bigcap_I A_i\right) = \prod_I P(A_i). \tag{3.4}$$

Il est important de noter que l'indépendance mutuelle est une condition plus forte que l'indépendance deux à deux.

Pour s'en convaincre il suffit de reconsidérer notre double lancer de pile ou face ci-dessus. Dans cet exemple on a en effet, A indépendant de B, B indépendant de C, et C indépendant de A, alors que C n'est pas indépendant de $A \cap B$.

Autres formules utiles.

$$P(A \cap B \cap C) = P(A|B \cap C)P(B|C)P(C) \tag{3.5}$$

$$P(A \cap B|C) = P(A|C)P(B|C \cap A) \tag{3.6}$$

Notations. Dans la suite nous utiliserons de façon interchangeable les notations suivantes pour désigner l'occurence simultanée de plusieurs événements :

- $A_1 \cap A_2 \cap \ldots \cap A_n$: la notation ensembliste (on insiste sur le fait que les A_i sont vus comme des ensembles).
- $A_1 \wedge A_2 \wedge \ldots \wedge A_n$: la notation logique (on insiste sur le fait que les A_i sont vus comme des formules logiques).
- $A_1, A_2, \ldots, A_n$: notation indifférente (plus légère).

3.3.4 Formules de Bayes

Les formules de Bayes permettent d'exprimer $P(A)$ et $P(B|A)$ en fonction de $P(A|B)$ et $P(B)$. Elles s'énoncent comme suit

Première formule de Bayes

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}. \tag{3.7}$$

Théorème des probabilités totales

Si $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ est un système complet d'événements non triviaux alors le théorème des probabilités totales peut s'écrire sous la forme suivante

$$P(A) = \sum_i P(A|B_i)P(B_i) \tag{3.8}$$

Théorème sur la "probabilité des causes" (deuxième formule de Bayes)

$$P(B_i|A) = \frac{P(A|B_i)P(B_i)}{\sum_k P(A|B_k)P(B_k)}, \tag{3.9}$$

Cette dernière formule permet de calculer les probabilités des causes possibles d'un événement sachant qu'une conséquence s'est réalisée, connaissant la probabilité de cette dernière sous l'hypothèse de chaque cause et connaissant la probabilité des causes a priori.

Il ne faut pas confondre la (ou les) **formules** *Bayes avec la notion de* **règle** *de Bayes utilisée en théorie de la décision. La règle de Bayes est une régle de décision qui minimise une probabilité d'erreurs. Nous en verrons des exemples plus loin dans ce cours.*

Discussion. Le théorème de Bayes (on donne souvent le nom de théorème de Bayes aux deux formules de Bayes) joue un rôle très important dans le cadre du calcul des probabilités. Il sert de fondement au raisonnement incertain probabiliste et est à la base de toute une branche de la statistique appelée *statistique bayesienne*.

Il permet de remettre à jour les probabilités d'un certain nombre d'alternatives B_i en fonction d'informations nouvelles (le fait que A soit réalisé). On utilise souvent le terme de *probabilités a priori* pour désigner les $P(B_i)$ et le terme de *probabilités a posteriori* pour désigner les $P(B_i|A)$.

Par exemple, dans le cadre du diagnostic médical cette formule permet à un médecin de remettre à jour la plausibilité de certaines maladies (désignées par les B_i) à partir des symptômes observés (désignés conjointement par A), partant d'une connaissance de la probabilité a priori d'observer les différentes maladies (obtenues par exemple en effectuant des statistiques) et une connaissance des probabilités d'observer les symptômes A pour chacune de ces maladies (obtenues également par application de méthodes statistiques). Il s'agit d'une extension de la logique classique au raisonnement plausible.

3.4 ESPACE PRODUIT

Nous introduisons ci-dessous quelques notions et terminologies qui seront utilisées et illustrées plus loin, notamment dans le cadre des variables aléatoires.

3.4.1 Définition

Etant donné un nombre fini d'espaces probabilisés $(\Omega_i, \mathcal{E}_i, P_i(\cdot))$ $(i = 1, \ldots, n)$ on peut définir un espace probabilisé produit $(\Omega, \mathcal{E}, P(\cdot))$ obtenu de la manière suivante :

- $\Omega = \Omega_1 \times \ldots \times \Omega_n$, (produit cartésien).
- $\mathcal{E} = \{A = \bigcup_j A_j \subset \Omega | \forall j, \forall i = 1, 2, \ldots, n, \exists A_{i,j} \in \mathcal{E}_i \ : A_j = A_{1,j} \times \ldots \times A_{n,j}\}$,
(extension des événements par produit cartésien et union dénombrable, où on peut exiger sans perte de généralité que les A_j soient disjoints)
- $P(A) = \sum_j P(A_j)$ et $P(A_j) = \prod_i P_i(A_{i,j})$ (factorisation de la loi de probabilités).

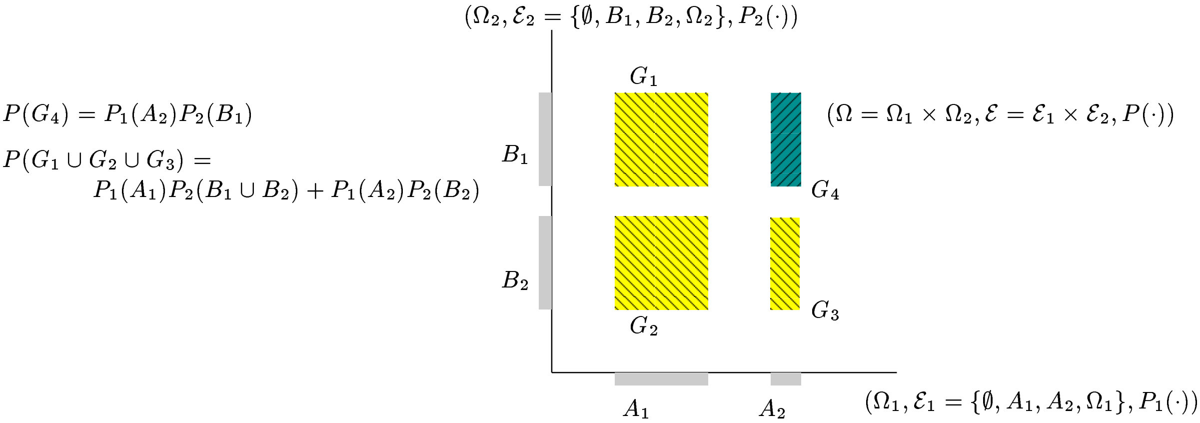


Figure 3.1. Espace probabilisé produit de deux espace binaires

On peut se convaincre que cette définition conduit bien à un espace probabilisé. On dira que les Ω_i sont les axes "orthogonaux" de l'espace produit et on parlera de la projection d'un événement A sur les axes pour désigner les A_i, et d'événements parallèles à un axe i si $A_i = \Omega_i$.

La figure 3.1 illustre la construction d'un espace produit à partir de deux espaces probabilisés de départ binaires (seulement deux événements non-triviaux dans le σ-algèbre). On voit que le σ-algèbre produit comprend les 4 événements de base décrits sur la figure. Ces derniers forment un système complet d'événements tel que tout événement de l'espace produit puisse s'écrire sous la forme d'une union de ces éléments (on parle d'une base). Voici les 16 ($= 2^4$) événements de l'espace produit

$$
\begin{array}{lll}
G_0 &=& \emptyset \\
G_1 &=& (A_1, B_1) \quad (\Rightarrow P(G_1) = P_1(A_1)P_2(B_1)) \\
G_2 &=& (A_1, B_2) \quad (\Rightarrow P(G_2) = P_1(A_1)P_2(B_2)) \\
G_3 &=& (A_2, B_2) \quad (\Rightarrow P(G_3) = P_1(A_2)P_2(B_2)) \\
G_4 &=& (A_2, B_1) \quad (\Rightarrow P(G_4) = P_1(A_2)P_2(B_1)) \\
G_5 &=& G_1 \cup G_2 = (A_1, \Omega_2) \triangleq A_1 \\
G_6 &=& G_3 \cup G_4 = (A_2, \Omega_2) \triangleq A_2 \\
G_7 &=& G_1 \cup G_4 = (\Omega_1, B_1) \triangleq B_1
\end{array}
\qquad
\begin{array}{lll}
G_8 &=& G_2 \cup G_3 = (\Omega_1, B_2) \triangleq B_2 \\
G_9 &=& G_1 \cup G_3 \\
G_{10} &=& G_2 \cup G_4 \\
G_{11} &=& G_1 \cup G_2 \cup G_3 \\
G_{12} &=& G_1 \cup G_2 \cup G_4 \\
G_{13} &=& G_1 \cup G_3 \cup G_4 \\
G_{14} &=& G_2 \cup G_3 \cup G_4 \\
G_{15} &=& G_1 \cup G_2 \cup G_3 \cup G_4 = \Omega
\end{array}
$$

où les événements G_5, G_6 sont parallèles à l'axe Ω_2 et G_7, G_8 sont parallèles à l'axe Ω_1. On pourra se convaincre que dans l'espace produit les événements $G_5 = G_1 \cup G_2$ et $G_7 = G_1 \cup G_4$ sont indépendants. En effet on a $P(G_5) = P_1(A_1)P_2(\Omega_2) = P_1(A_1)$ et $P(G_7) = P_2(B_1)P_1(\Omega_1) = P_2(B_1)$. D'autre part, on a $P(G_5 \cap G_7) = P(G_1) = P_1(A_1)P_2(B_1)$.

Plus généralement, il est vrai que si deux événements d'un espace produit sont parallèles à des ensembles d'axes complémentaires, ils sont indépendants. En d'autres termes, si un premier événement ne spécifie rien selon un certain nombres d'axes, alors le fait de savoir qu'un second événement qui ne spécifie que de l'information relative à ces axes soit réalisé ne fournit aucune information sur le premier événement.

3.4.2 Séries d'épreuves identiques et indépendantes

Un cas particulièrement intéressant en pratique d'espace produit est celui où tous les $(\Omega_i, \mathcal{E}_i, P_i(\cdot))$ sont identiques. Un tel type d'espace produit permet de modéliser les séries d'épreuves identiques et indépendantes, rencontrées en théorie de l'échantillonnage et à la base des statistiques.

3.4.3 Factorisation

Dans certains cas il est possible de factoriser un espace de départ en effectuant l'opération inverse, c'est-à-dire de l'écrire sous la forme du produit cartésien d'espaces indépendants (non nécessairement identiques).

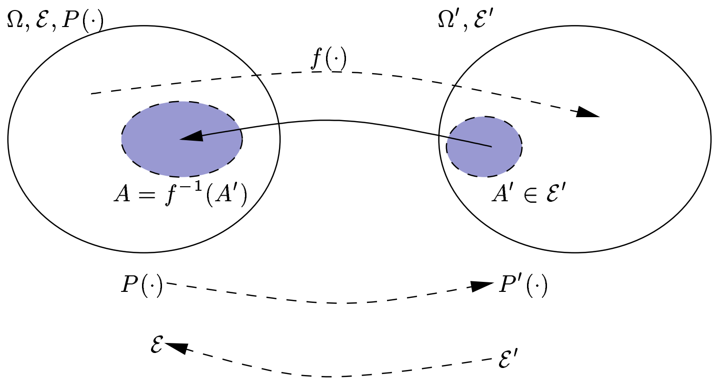


Figure 3.2. Variable aléatoire

(Suggestion : partir d'un espace fini dont on suppose que l'algèbre est engendré par deux événements indépendants A et B, et montrer qu'il peut se factoriser en deux axes correspondant à ces événements.)

3.4.4 Marginalisation

Partant d'un espace produit, il est possible de reconstituer les espaces produits correspondant à certains de ces axes par une opération de projection. En calcul de probabilité on dit qu'on "marginalise" les autres axes. Notons que cette opération peut s'effectuer sur un espace produit dont la loi de probabilité n'est pas factorisable.

3.5 VARIABLES ALEATOIRES

3.5.1 Définition générale

Soient un espace probabilisé $(\Omega, \mathcal{E}, P(\cdot))$ de départ et un univers Ω' de destination muni d'un algèbre d'événements $\mathcal{E}'$. Une fonction $f(\cdot)$ de Ω dans Ω' est une variable aléatoire si elle possède la propriété suivante :

$$\forall A' \in \mathcal{E}' \ : f^{-1}(A') \in \mathcal{E}, \tag{3.10}$$

où $f^{-1}(A')$ désigne $\{w \in \Omega | f(w) \in A'\}$. On dit alors que la fonction $f(\cdot)$ est $(\mathcal{E}, \mathcal{E}')$-mesurable, ou simplement mesurable si aucune confusion sur les σ-algèbres n'est possible.

Si cette propriété (peu restrictive en pratique) est vérifiée alors la variable aléatoire induit une loi de probabilité sur $(\Omega', \mathcal{E}')$ définie de la manière suivante :

$$P_f(A') = P\left(f^{-1}(A')\right). \tag{3.11}$$

Une variable aléatoire est donc une fonction définie sur un espace probabilisé qui est compatible avec les algèbres d'événements définis dans son espace d'origine et de destination, et qui induit donc une loi de probabilité sur l'espace de destination à partir de la loi de probabilité définie sur l'espace de départ. *(Suggestion : montrer que cette loi vérifie bien les axiomes de Kolmogorov.)*

Notons d'emblée que si les deux univers sont finis et munis des algèbres maximaux (et même si seulement l'espace de départ vérifie cette propriété), alors toute fonction de Ω dans Ω' est une variable aléatoire.

Par conséquent, si nous avons pris la précaution de formuler les restrictions ci-dessus, c'est que nous voulons pouvoir appliquer le concept de variable aléatoire dans des situations où l'espace de départ est infini (p.ex. $\Omega = \mathbb{R}^p$).

La figure 3.2 représente schématiquement la notion de variable aléatoire.

Interprétation. Il est à noter que toute fonction de Ω dans Ω' induit à partir de $\mathcal{E}'$ un nouvel algèbre d'événements sur l'espace de départ Ω. *(Nous suggérons au lecteur de s'en convaincre en montrant que l'ensemble des parties de Ω qui peuvent s'écrire sous la forme $f^{-1}(A')$ avec $A' \in \mathcal{E}'$ est un σ-algèbre si $\mathcal{E}'$ est un σ-algèbre.)*

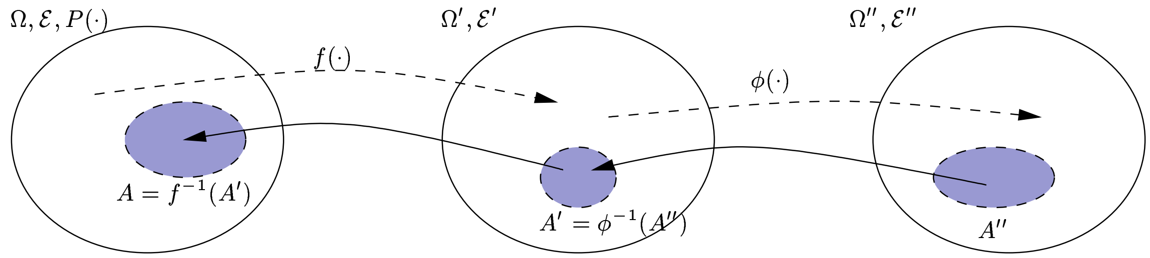


Figure 3.3. Variable aléatoire composée

Le sens profond de la condition (3.10) est donc que l'algèbre des événements induit par la fonction $f(\cdot)$ à partir de $\mathcal{E}'$ sur Ω doit être inclus dans l'algèbre sur lequel la loi de probabilité $P(\cdot)$ est connue. Donc, une variable aléatoire a pour effet de remplacer l'algèbre $\mathcal{E}$ par l'algèbre $f^{-1}(\mathcal{E}')$ qui contient en général moins de sous-ensembles que $\mathcal{E}$. Elle opère donc sur un espace probabilisé en condensant l'information dont on disposait au départ en une information plus grossière.

Il s'agit bien là du sens physique profond de la notion de variable aléatoire : l'observation d'une variable aléatoire fournit une information généralement partielle sur les événements de l'univers de départ.

En résumé, une variable aléatoire transpose une loi de probabilité d'un espace de départ vers un espace de destination et elle transpose une structure d'algèbre de l'espace de destination vers l'espace de départ.

3.5.2 Fonction d'une variable aléatoire

Soit une variable aléatoire X à valeurs dans Ω' défini par une fonction $f(\cdot)$, et une certaine fonction $\phi(\cdot)$ définie sur Ω' et à valeurs dans Ω''. Alors, si $\phi(\cdot)$ a le statut de variable aléatoire sur Ω' (compatibilité de $\mathcal{E}'$ et $\mathcal{E}''$), la fonction composée $\phi \circ f(\cdot) = \phi(f(\cdot))$ définit également une v.a. sur Ω (voir figure 3.3).

Nous verrons ci-dessous le cas particulièrement intéressant en pratique où les deux fonctions sont des v.a. réelles.

3.5.3 Variable aléatoire discrète

Une variable aléatoire est discrète si l'ensemble de ses valeurs possibles est fini. Une telle variable aléatoire définit un système complet d'événements mutuellement exclusifs; en fait elle est équivalente à la donnée d'un système complet d'événements discrets. Aussi nous ne distinguerons plus dans la suite ces deux notions. Notons que si l'espace Ω est fini, alors toute variable aléatoire est nécessairement discrète.

On utilisera communément la notation $\mathcal{X} = \{X_1, \ldots, X_k\}, \mathcal{Y} = \{Y_1, \ldots, Y_l\}, \ldots$ pour désigner l'ensemble des valeurs possibles de telles variables aléatoires, et par extension la variable aléatoire elle-même sera désignée par $\mathcal{X}(\cdot)$ ou simplement $\mathcal{X}$.

D'autre part, nous assimilerons dans nos notations les valeurs prises par la variable aléatoire avec les événements (sous-ensembles de Ω) qui leur correspondent. Par exemple la notation $P(X_i)$ désignera la probabilité de $\{\omega \in \Omega \ : \mathcal{X}(\omega) = X_i\}$.

Enfin, pour alléger nos écritures nous utiliserons dans la mesure du possible la notation $P(\mathcal{X})$ au lieu de $P_{\mathcal{X}}(\cdot)$ pour désigner la loi de probabilité induite par une variable aléatoire discrète.

La figure 3.4 illustre la manière dont une variable aléatoire discrète induit un système complet d'événements sur Ω.

Remarque. Dans la littérature on désigne souvent par variable aléatoire discrète une v.a. pouvant prendre un nombre dénombrable (éventuellement infini) de valeurs. Comme nous ne nous intéresserons que très marginalement au cas particulier infini, nous sous-entendrons (sauf mention explicite du contraire) que la variable discrète est aussi finie.

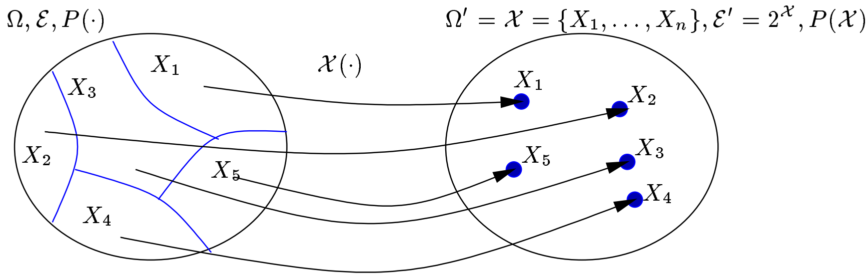


Figure 3.4. Variable aléatoire discrète

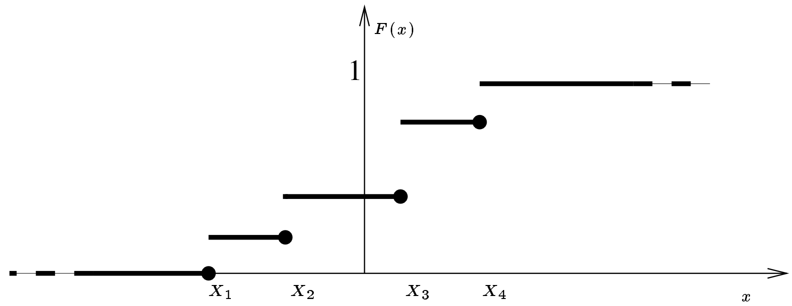


Figure 3.5. Fonction de répartition d'un v.a. réelle discrète

3.5.4 Variables aléatoires réelles

Une v.a. est dite réelle si $\Omega' = \mathbb{R}$. Elle peut être discrète ou non. Evidemment si Ω est fini, elle sera nécessairement discrète.

3.5.4.1 Fonction de répartition. Par définition, la fonction de répartition $F_{\mathcal{X}}$ d'une v.a. réelle $\mathcal{X}$ est une fonction de $\mathbb{R}$ dans $[0, 1]$ définie par

$$F(x) = P(] - \infty, x[), \tag{3.12}$$

Elle peut donc en principe avoir des discontinuités (à droite), si certaines des valeurs sont de probabilité non-nulle. Elle est monotonément croissante, et $F(-\infty) = 0$ et $F(+\infty) = 1$.

Cette fonction *caractérise* la variable aléatoire et permet de calculer la probabilité de tout intervalle de $\mathbb{R}$ par

$$P(a \leq \mathcal{X} < b) = F(b) - F(a). \tag{3.13}$$

Remarque. On peut tout aussi bien définir la fonction de répartition de façon à ce qu'elle soit continue à droite, on a alors

$$F'(x) = P(] - \infty, x]). \tag{3.14}$$

C'est la convention qu'on trouve généralement dans la littérature anglo-saxonne.

Lorsque la v.a. réelle est aussi discrète, il n'y a qu'un nombre fini de points de $\mathbb{R}$ de probabilité non-nulle. La fonction de répartition prend alors l'allure indiquée à la figure 3.5.

3.5.4.2 Variable aléatoire (réelle) continue. Une variable aléatoire est dite continue s'il existe une fonction $f(\cdot)$ (appelée densité de probabilités) définie sur $\mathbb{R}$ telle que $\forall a \leq b$ on a

$$P(]a, b[) = P([a, b]) = P([a, b[) = P(]a, b]) = \int_a^b f(x)dx. \tag{3.15}$$

Dans ce cas, $F(\cdot)$ est dérivable (et donc continue) et admet $f(\cdot)$ comme dérivée. Par conséquent, $f(\cdot)$ est positive et d'intégrale sur $\mathbb{R}$ égale à 1. La figure 3.6 représente graphiquement la fonction de répartition de ce type de variable aléatoire.

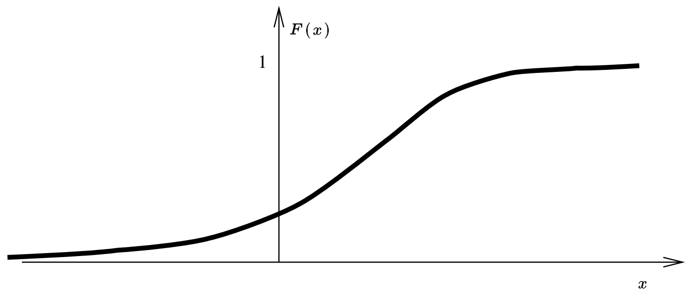


Figure 3.6. Fonction de répartition d'un v.a. réelle continue

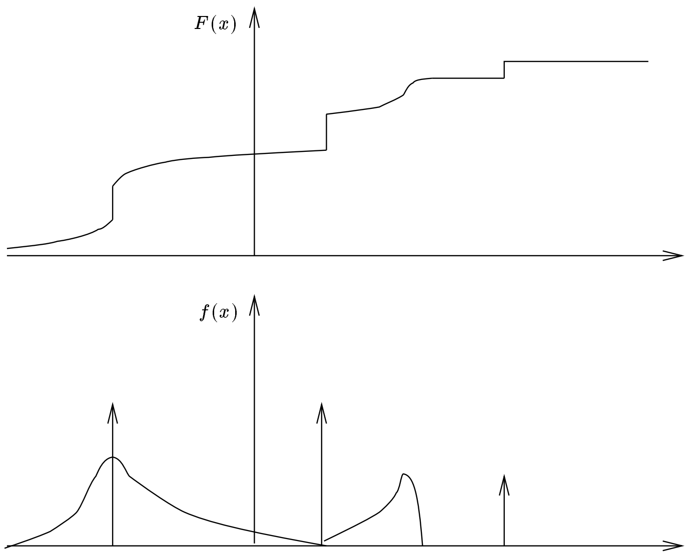


Figure 3.7. Fonction de réparition et distribution de probabilité

Il est évident qu'une variable aléatoire peut n'être ni discrète, ni continue. Elle ne peut cependant qu'avoir au plus un nombre dénombrable de points de discontinuité.

Dans la suite nous utiliserons la notation $\mathcal{X} \sim f(x)$ pour indiquer qu'une v.a. suit la densité de probabilités $f(x)$.

3.5.4.3 Cas général. Dans le cas général on peut séparer la v.a. réelle en la somme d'une composante continue et d'une composante discrète[6].

Notons qu'on peut dans le cas général aussi faire appel à la théorie des distributions pour définir cette fois la densité comme une *distribution* qui s'écrit sous la forme d'une combinaison linéraire d'une fonction et d'une série d'impulsions de Dirac. Cette situation est schématisée graphiquement à la figure 3.7.

3.5.5 Indépendance de variables aléatoires

Deux variables aléatoires $\mathcal{X}$ et $\mathcal{Y}$ définies sur un même espace probabilisé sont indépendantes si et seulement si, $\forall A \in \mathcal{E}_X, \forall B \in \mathcal{E}_Y$ on a

$$P(\mathcal{X}^{-1}(A) \cap \mathcal{Y}^{-1}(B)) = P(\mathcal{X}^{-1}(A))P(\mathcal{Y}^{-1}(B)). \tag{3.16}$$

[6]En toute généralité, on peut montrer que toute fonction de répartition peut se décomposer en une somme de trois termes ($F(x) = F_c(x) + F_d(x) + F_s(x)$) tels que F_c soit absolument continue (continue et dérivable), F_d est discrète, et F_s (composante singulière) est continue mais ne possède pas de dérivée. Nous supposons que $F_s = 0$.

En d'autres termes, la loi induite par la v.a. $\mathcal{X}\mathcal{Y}(\cdot) = (\mathcal{X}(\cdot), \mathcal{Y}(\cdot))$ sur l'espace produit $\Omega_{\mathcal{X}} \times \Omega_{\mathcal{Y}}$, est factorisable, et on a

$$P_{\mathcal{X}\mathcal{Y}} = P_{\mathcal{X}} P_{\mathcal{Y}}. \tag{3.17}$$

Dans le cas où les variables aléatoires sont réelles cette condition se traduit par

$$H(x,y) \stackrel{\triangle}{=} P(X < x \wedge Y < y) = F(x)G(y), \tag{3.18}$$

où $F(\cdot)$ et $G(\cdot)$ sont les fonctions de répartition respectivement de $\mathcal{X}$ et $\mathcal{Y}$. Si de plus X et Y admettent les densités $f(\cdot)$ et $g(\cdot)$, alors il en est de même pour le couple, dont la densité est le produit :

$$h(x,y) = f(x)g(y).$$

Notons que nous pouvons étendre ces notions et propriétés par induction au cas d'un nombre fini quelconque de v.a. On parle alors de vecteurs aléatoires et le cas particulier intéressant est celui où celui-ci appartient à $\mathbb{R}^p$.

Dans le cas de deux variables aléatoires discrètes $\mathcal{X} = \{X_1, \ldots, X_k\}$ et $\mathcal{Y} = \{Y_1, \ldots, Y_l\}$ on peut se convaincre que la condition ci-dessus est équivalente à

$$\forall i,j\ : P(X_i, Y_j) = P(X_i)P(Y_j). \tag{3.19}$$

Plus généralement, un ensemble de variables aléatoires discrètes $\mathcal{X}^1 = \{X_1^1, \ldots, X_{k^1}^1\}, \mathcal{X}^2 = \{X_1^2, \ldots, X_{k^2}^2\}, \ldots, \mathcal{X}^p = \{X_1^p, \ldots, X_{k^p}^1\}$ sont mutuellement indépendantes si et seulement si on a

$$\forall i^1, i^2, \ldots, i^p\ : P(X_{i^1}^1, X_{i^2}^2, \ldots, X_{i^p}^p) = P(X_{i^1}^1)P(X_{i^2}^2)\cdots P(X_{i^p}^p), \tag{3.20}$$

condition qu'on écrit de manière plus synthétique sous la forme

$$P(\mathcal{X}^1, \mathcal{X}^2, \ldots, \mathcal{X}^p) = P(\mathcal{X}^1)P(\mathcal{X}^2)\cdots P(\mathcal{X}^p), \tag{3.21}$$

où la notation signifie que les variables aléatoires en argument peuvent être remplacées par une quelconque combinaison de leurs valeurs possibles.

3.5.6 Espérance mathématique

Pour une variable réelle aléatoire discrète on définit l'espérance mathématique (on dit aussi sa moyenne) par

$$E\{\mathcal{X}\} = \sum_{i=1}^{k} X_k P(X_k). \tag{3.22}$$

Notons que dans le cas fini cette grandeur existe toujours. Dans le cas infini (dénombrable) il est facile de construire des exemples tels que cette série ne converge pas.

Pour une variable continue on a

$$E\{\mathcal{X}\} = \int_{\mathbb{R}} x f(x) dx. \tag{3.23}$$

Dans le cas plus général on peut combiner les deux formules. L'écriture générale est alors la suivante

$$E_P\{\mathcal{X}\} = \int_{\Omega} X(\omega) dP(\omega), \tag{3.24}$$

où le dP indique que l'intégrale est prise par rapport à la mesure P définie sur l'espace de départ Ω, ce qui est équivalent à

$$E\{\mathcal{X}\} = \int_{\mathbb{R}} x dP_{\mathcal{X}}(x), \tag{3.25}$$

où le $dP_{\mathcal{X}}$ indique que l'intégrale est prise par rapport à la mesure induite sur l'espace d'arrivée.

Il faut souligner, même si c'est évident, que l'intégrale n'est pas toujours définie. Il existe des densités de probabilités continues pour lesquelles l'espérance mathématique n'est pas définie.

Par exemple, la distribution de *Cauchy*

$$f(x) = \frac{1}{\pi(1+x^2)}, \tag{3.26}$$

n'admet pas d'espérance.

Cependant, toute fonction continue et à support compact étant intégrable, toute variable aléatoire réelle continue et bornée admet une espérance mathématique.

L'espérance mathématique d'une fonction ($\phi(\cdot)\ :\mathbb{R}\to\mathbb{R}$) est définie par

$$E_P\{\phi(\mathcal{X})\} = \int_\Omega \phi(X(\omega))dP(\omega), \tag{3.27}$$

et dans le cas continu on a

$$E_P\{\phi(\mathcal{X})\} = \int_\mathbb{R} \phi(x)f(x)dx. \tag{3.28}$$

Cas particuliers :

1. Fonction constante ($\phi(x) = a$) : $E\{\phi\} = a$.
2. Fonction linéaire ($\phi(x) = ax + b$) : $E\{\phi\} = aE\{\mathcal{X}\} + b$.
3. Somme de deux v.a. ($\phi(x, y) = x + y$) : $E\{\phi\} = E\{\mathcal{X}\} + E\{\mathcal{Y}\}$.
4. Produit de deux v.a. ($\phi(x, y) = xy$) :

   $$E\{\mathcal{X}\mathcal{Y}\} = \int_{\mathbb{R}^2} xydP_{\mathcal{X}\mathcal{Y}}(x, y).$$

   Lorsque $\mathcal{X}$ et $\mathcal{Y}$ sont indépendantes, la mesure $dP_{\mathcal{X}\mathcal{Y}}(x, y)$ se factorise et l'intégrale double peut se décomposer en produit des deux intégrales simples :

   $$E\{\mathcal{X}\mathcal{Y}\} = \int_\mathbb{R} xdP_\mathcal{X}(x) \int_\mathbb{R} ydP_\mathcal{Y}(y) = E\{\mathcal{X}\}E\{\mathcal{Y}\}.$$

   La réciproque n'est pas vraie.

3.5.7 Variance et écart-type

Lorsque l'espérance existe, la **variance** est définie par

$$V\{\mathcal{X}\} = \sigma^2 = E\{(\mathcal{X} - m)^2\}, \tag{3.29}$$

lorsque cette grandeur existe, où $m = E\{\mathcal{X}\}$.

L'écart-type est la racine carrée σ de la variance.

Propriétés de la variance. On a

$$E\{(\mathcal{X} - a)^2\} = V\{\mathcal{X}\} + (E\{\mathcal{X}\} - a)^2, \tag{3.30}$$

et par conséquent, la variance est la valeur minimale de $E\{(\mathcal{X} - a)^2\}$, et la valeur $a = E\{\mathcal{X}\}$ minimise l'expression $E\{(\mathcal{X} - a)^2\}$. Cette propriété est exploitée très largement en statistiques, dans le domaine de l'estimation au sens des moindres carrés.

On en déduit, en prenant $a = 0$ que

$$V\{\mathcal{X}\} = E\{X^2\} - (E\{\mathcal{X}\})^2. \tag{3.31}$$

L'espérance et l'écart-type sont reliés par *l'inégalité de Bienaymé-Tchebyshev* :

$$P(|X - E\{\mathcal{X}\}| > \epsilon) \leq \frac{\sigma^2}{\epsilon^2}, \tag{3.32}$$

dont on déduit que si $\sigma = 0$ la v.a. est presque sûrement égale à sa moyenne, c'est-à-dire constante. La variance mesure donc bien le caractère aléatoire d'une v.a.

Par ailleurs, on a

- $V\{\mathcal{X} + a\} = V\{\mathcal{X}\}$.
- $V\{a\mathcal{X}\} = a^2 V\{\mathcal{X}\}$.
- $V\{\mathcal{X} + \mathcal{Y}\} = V\{\mathcal{X}\} + V\{\mathcal{Y}\} + 2\mathrm{cov}\{\mathcal{X}, \mathcal{Y}\}$, où

  $$\mathrm{cov}\{\mathcal{X}, \mathcal{Y}\} \stackrel{\triangle}{=} E\{(\mathcal{X} - E\{\mathcal{X}\})(\mathcal{Y} - E\{\mathcal{Y}\})\} = E\{\mathcal{X}\mathcal{Y}\} - E\{\mathcal{X}\}E\{\mathcal{Y}\}$$

  désigne la covariance.
  Si les v.a. sont indépendantes, on a $\mathrm{cov}\{\mathcal{X}, \mathcal{Y}\} = 0$ et donc

  $$V\{\mathcal{X} + \mathcal{Y}\} = V\{\mathcal{X}\} + V\{\mathcal{Y}\}.$$

  La réciproque n'est pas vraie.

3.5.8 Inégalité de Jensen

Si $\phi(\cdot)$ est une fonction convexe (voir cours oral, pour la définition de la convexité), et si $\mathcal{X}$ est une v.a. réelle, alors

$$E_P\{\phi(\mathcal{X})\} \geq \phi(E_P\{\mathcal{X}\}), \tag{3.33}$$

et si la fonction est strictement convexe, alors l'égalité implique $X = cnste$, sauf éventuellement sur un ensemble de probabilité nulle.

Cette inégalité est très pratique dans le contexte d'un certain nombre de démonstrations en théorie de l'information et dans le domaine des processus aléatoires.

3.6 COUPLES DE V.A. ET CONDITIONNEMENT

Dans cette section on se focalisera en première lecture sur les sections 3.6.1 et 3.6.3.

3.6.1 Cas discret

Nous étudions ici les couples de v.a. $(\mathcal{X}, \mathcal{Y})$ tels que $\mathcal{X}$ et $\mathcal{Y}$ prennent leurs valeurs dans un ensemble fini désigné respectivement par $\mathcal{X} = \{X_1, \ldots, X_k\}$ et $\mathcal{Y} = \{Y_1, \ldots, Y_l\}$ munis de leur $\sigma-$algèbre maximal. Dans ce cas, le couple prend ses valeurs dans $\mathcal{X} \times \mathcal{Y}$ muni du $\sigma-$algèbre produit, qui est également maximal. On suppose que la fonction ainsi induite de $\Omega \to \mathcal{X} \times \mathcal{Y}$ est bien $\mathcal{E}$-mesurable.

3.6.1.1 Lois associées.

Loi (con)jointe. La loi de probabilité du couple $P_{\mathcal{X},\mathcal{Y}}$ est déterminée complètement par la connaissance des kl nombres

$$p_{i,j} \stackrel{\triangle}{=} P([\mathcal{X} = X_i] \cap [\mathcal{Y} = Y_j]) = P(X_i, Y_j), \forall i = 1, \ldots, k, \forall j = 1, \ldots, l. \tag{3.34}$$

On a bien sûr $\sum_{i=1}^{k} \sum_{j=1}^{l} p_{i,j} = 1$.

$$
\begin{array}{c|ccccc|c}
 & Y_1 & \cdots & Y_j & \cdots & Y_l & \\
\hline
X_1 & & & \vdots & & & \\
\vdots & & & \vdots & & & \\
X_i & \cdots & \cdots & p_{i,j} & \cdots & \cdots & p_{i,\cdot} \\
\vdots & & & \vdots & & & \\
X_k & & & \vdots & & & \\
\hline
 & & & p_{\cdot,j} & & &
\end{array}
$$

Figure 3.8. Table de contingences

Lois marginales. La loi marginale de $\mathcal{X}$ est évidemment

$$p_{i,\cdot} \triangleq P(\mathcal{X} = X_i) = \sum_{j=1}^{l} p_{i,j}. \tag{3.35}$$

De même, la loi marginale de $\mathcal{Y}$ est

$$p_{\cdot,j} \triangleq P(\mathcal{Y} = X_j) = \sum_{i=1}^{k} p_{i,j}. \tag{3.36}$$

On représente souvent un couple de v.a. à l'aide d'une *table de contingences*, telle qu'illustrée à la figure 3.8.

Lois conditionnelles. La loi conditionnelle de $\mathcal{X}$ connaissant $\mathcal{Y}$ est définie par

$$p_{X_i|Y_j} \triangleq P(\mathcal{X} = X_i | \mathcal{Y} = Y_j) = \frac{p_{i,j}}{p_{\cdot,j}}, \tag{3.37}$$

et celle de $\mathcal{Y}$ connaissant $\mathcal{X}$ par

$$p_{Y_j|X_i} \triangleq P(\mathcal{Y} = Y_j | \mathcal{X} = X_i) = \frac{p_{i,j}}{p_{i,\cdot}}. \tag{3.38}$$

3.6.1.2 Moments conditionnels.

Supposons que $\mathcal{Y}$ soit une v.a. réelle (où éventuellement complexe).

Espérance conditionnelle. Alors on définit l'espérance conditionnelle de $\mathcal{Y}$ par

$$E\{\mathcal{Y}|X\} \triangleq \sum_{j=1}^{l} Y_j p_{Y_j|X}. \tag{3.39}$$

$E\{\mathcal{Y}|X\}$ est donc une fonction (réelle où complexe) de X. Cette fonction s'appelle **fonction de régression** de $\mathcal{Y}$ en X. Comme $\mathcal{X}$ est une v.a. cette fonction définit une v.a. réelle ou complexe. Cette variable aléatoire présente un certain nombre de propriétés remarquables que nous allons énumérer.

En premier lieu elle est linéaire (il s'agit d'une espérance). Donc,

$$E\{\mathcal{Y}_1 + \mathcal{Y}_2|\mathcal{X}\} = E\{\mathcal{Y}_1|\mathcal{X}\} + E\{\mathcal{Y}_2|\mathcal{X}\}. \tag{3.40}$$

Mais surtout, elle satisfait au **théorème de l'espérance totale** :

$$E\{E\{\mathcal{Y}|\mathcal{X}\}\} = E\{\mathcal{Y}\}. \tag{3.41}$$

En effet, on peut calculer son espérance mathématique ce qui donne

$$E\{E\{\mathcal{Y}|\mathcal{X}\}\} = \sum_{i=1}^{k} p_{i,\cdot} \sum_{j=1}^{l} Y_j p_{Y_j|X_i} \tag{3.42}$$

$$= \sum_{j=1}^{l} Y_j \sum_{i=1}^{k} p_{i,\cdot} p_{Y_j|X_i} \tag{3.43}$$

$$= \sum_{j=1}^{l} Y_j p_{\cdot,j}. \tag{3.44}$$

Variance conditionnelle. On définit similairement la variance conditionnelle comme une v.a. qui prend la valeur

$$V\{\mathcal{Y}|\mathcal{X}\} \triangleq E\left\{(\mathcal{Y} - E\{\mathcal{Y}|\mathcal{X}\})^2 |\mathcal{X}\right\}. \tag{3.45}$$

On a le **théorème de la variance totale** qui s'écrit comme suit :

$$V\{\mathcal{Y}\} = E\{V\{\mathcal{Y}|\mathcal{X}\}\} + V\{E\{\mathcal{Y}|\mathcal{X}\}\}. \tag{3.46}$$

3.6.2 Variables continues

3.6.2.1 Une des deux variables est continue. On peut directement étendre ce qui précède au cas où $\mathcal{Y}$ est une variable continue en remplaçant les probabilités par des fonctions de répartition ou des densités. On note

$$G(y|X_i) \triangleq P(\mathcal{Y} < y|\mathcal{X} = X_i). \tag{3.47}$$

La fonction de répartition marginale s'écrit alors

$$G(y) \triangleq \sum_{i=1}^{k} p_{i,\cdot} G(y|X_i) \tag{3.48}$$

qui dérivée terme à terme donne la densité marginale

$$g(y) \triangleq \sum_{i=1}^{k} p_{i,\cdot} g(y|X_i). \tag{3.49}$$

Les théorèmes de l'espérance et de la variance totales restent également d'application.

On peut également écrire

$$P(\mathcal{X} = x|\mathcal{Y} < y) = \frac{G(y|x)P(x)}{G(y)}, \tag{3.50}$$

mais nous ne pouvons pas pour le moment écrire

$$P(\mathcal{X} = x|\mathcal{Y} = y) = \frac{g(y|x)P(x)}{g(y)}, \tag{3.51}$$

car $Y = y$ est un événement de probabilité nulle par rapport auquel on ne peut pas en principe conditionner. Nous allons indiquer ci-dessous sous quelles conditions un conditionnement de ce type est permis.

3.6.2.2 Cas le plus général. Nous référons le lecteur intéressé par les conditions d'existence de mesures de probabilités conditionnelles vis-à-vis d'événements de probabilité nulle à [Bil79] et à [Sap90] pour une discussion des implications en terme de conditionnement vis-à-vis de v.a. quelconques.

On peut résumer la situation de la manière suivante : si $\mathcal{Y}$ est une variable aléatoire réelle, et si $\mathcal{X}$ est une variable aléatoire soit discrète, soit à valeurs dans $\mathbb{R}^p$, alors il est permis de conditionner Ω et donc $\mathcal{Y}$ par rapport à $\mathcal{X}$ localement. De plus, si $E\{\mathcal{Y}\}$ existe alors il existe une v.a. aléatoire "espérance conditionnelle" qui satisfait au théorème de l'espérance totale. Enfin, si $V\{\mathcal{Y}\}$ existe aussi alors cette v.a. satisfait aussi au théorème de la variance totale.

Enfin, les formules de conditionnement des densités s'obtiennent par analogie au cas discret.

En particulier on a

$$g(y|x) = \frac{h(x,y)}{f(x)} \tag{3.52}$$

$$E\{\mathcal{Y}|x\} = \int yg(y|x)dy \tag{3.53}$$

et la formule de Bayes

$$g(y|x) = \frac{f(x|y)g(y)}{f(x)}. \tag{3.54}$$

3.6.3 Illustration

Un exemple pratique important où on considère les dépendances entre variables continues et discrètes est fourni par la théorie de la décision, qui intervient dans les problèmes de classification en apprentissage automatique, et également dans les problèmes de transmission de données numériques à l'aide de signaux analogiques.

Prenons par exemple, le problème de l'allocation de crédit bancaire qui se ramène à celui de l'étude des relations entre variables numériques et supposées continues (montant du crédit souhaité, niveau de salaire, endettement, âge ...) et discrètes décrivant la situation financière et sociale d'un demandeur de crédit (état civil, propriétaire, statut professionnel...), et la décision optimale d'une banque (Accord ou non du crédit).

Du point de vue du banquier non altruiste, la décision optimale est en l'occurence celle qui maximise l'espérance mathématique du bénéfice de la banque. Si le crédit est accordé, ce bénéfice dépendra du fait que le demandeur sera capable de rembourser les mensualités ou non. Si le crédit n'est pas accordé, le bénéfice est nul. Le banquier fera donc appel à un logiciel qui déterminera, sur base des informations fournies par le demandeur, la probabilité de remboursement complet du crédit (disons $P(\mathcal{R} = V|\mathcal{I})$, où $\mathcal{R}$ désigne une variable qui vaut V s'il y a remboursement et F sinon, et $\mathcal{I}$ symbolise les informations propres au demandeur), à partir de laquelle on pourra déterminer l'espérance mathématique du bénéfice par la formule de l'espérance totale (conditionnée par l'information fournie par le demandeur)

$$E\{\mathcal{B}|\mathcal{I}\} = E\{\mathcal{B}|\mathcal{R} = V, \mathcal{I}\}P(\mathcal{R} = V|\mathcal{I}) + E\{\mathcal{B}|\mathcal{R} = F, \mathcal{I}\}P(\mathcal{R} = F|\mathcal{I}). \tag{3.55}$$

Dans cette formule, on a évidemment

$$P(\mathcal{R} = F|\mathcal{I}) = 1 - P(\mathcal{R} = V|\mathcal{I}),$$

et le chiffre $E\{\mathcal{B}|\mathcal{R} = V, \mathcal{I}\}$ correspond au gain de la banque calculé au moyen de formules d'actualisation tenant compte des conditions du crédit (intérêt, type de remboursement, ...), du coût de l'argent immobilisé que la banque doit assumer, et est évidemment proportionnel au montant du crédit. D'autre part, le terme $E\{\mathcal{B}|\mathcal{R} = F, \mathcal{I}\}$ est quant à lui un "bénéfice" negatif.

Par conséquent, le crédit sera alloué si

$$E\{\mathcal{B}|\mathcal{I}\} > 0 \Leftrightarrow P(\mathcal{R} = V|\mathcal{I}) > \frac{-E\{\mathcal{B}|\mathcal{R} = F, \mathcal{I}\}}{E\{\mathcal{B}|\mathcal{R} = V, \mathcal{I}\} - E\{\mathcal{B}|\mathcal{R} = F, \mathcal{I}\}}, \tag{3.56}$$

et le problème se ramène donc essentiellement au calcul de $P(\mathcal{R} = V|\mathcal{I})$ et à la comparaison de celle-ci à un certain seuil, $\mathcal{R}$ étant une variable discrète et $\mathcal{I}$ un ensemble de variables généralement mixtes discrètes/continues. Nous verrons au cours d'apprentissage automatique que les méthodes utilisées par les banquiers se fondent essentiellement sur une approximation de $P(\mathcal{R} = V|\mathcal{I})$ obtenue à partir de bases de données des clients antérieurs de la banque et grâce aux méthodes d'apprentissage.

Notons que nous avons utilisé la notation explicite $\mathcal{R} = V$ ou $\mathcal{R} = F$ pour bien mettre en évidence le conditionnement sur des valeurs prises par la v.a. discrète $\mathcal{R}$. Selon notre convention, la notation $f(\mathcal{I}|\mathcal{R})$ désigne en effet une fonction à deux arguments définie par

$$f(\mathcal{I}|\mathcal{R}) = \begin{cases} f(\mathcal{I}|\mathcal{R} = V) \text{ si } \mathcal{R} = V \\ f(\mathcal{I}|\mathcal{R} = F) \text{ si } \mathcal{R} = F \end{cases}, \tag{3.57}$$

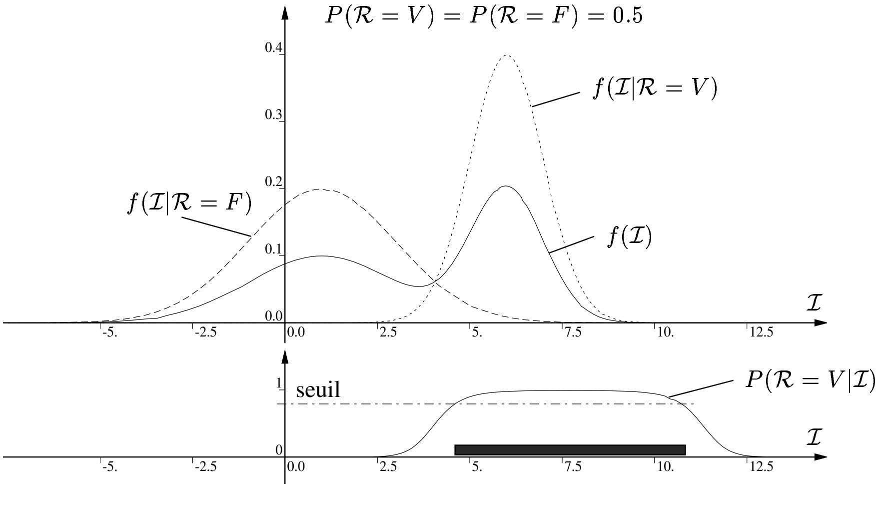


Figure 3.9. Illustration des densités conditionnelles

et $f(\mathcal{I}, \mathcal{R})$ est définie par

$$f(\mathcal{I}|\mathcal{R})P(\mathcal{R}) \tag{3.58}$$

où $\mathcal{R}$ peut désigner soit la valeur V soit la valeur F.

Cette remarque étant faite, illustrons ces idées graphiquement pour un cas simple où $\mathcal{I}$ se réduit à une seule variable numérique (disons un chiffre magique obtenu en combinant les différentes informations selon une formule pré-établie) et faisons l'hypothèse que cette variable est continue. La figure 3.9 représente graphiquement la situation, en termes des densités de probabilité $f(\mathcal{I}), f(\mathcal{I}|\mathcal{R} = V), f(\mathcal{I}|\mathcal{R} = F)$ et la probabilité conditionnelle $P(\mathcal{R} = V|\mathcal{I})$.

Notons qu'à la figure 3.9 les distributions conditionnelles $f(\mathcal{I}|\mathcal{R} = V)$ et $f(\mathcal{I}|\mathcal{R} = F)$ sont gaussiennes, de moyennes et de variances différentes. On suppose donc que les bons et les mauvais clients présentent des valeurs assez différentes de notre variable “magique”. On suppose également qu'a priori dans la population qui s'adresse aux banques pour obtenir des crédits on a la même proportion de bons et de mauvais clients, ce qui se traduit par l'égalité des probabilités a priori $P(\mathcal{R} = V)$ et $P(\mathcal{R} = F)$. On a,

$$f(\mathcal{I}) = f(\mathcal{I}|\mathcal{R} = V)P(\mathcal{R} = V) + f(\mathcal{I}|\mathcal{R} = F)P(\mathcal{R} = F), \tag{3.59}$$

et la probabilité a posteriori $P(\mathcal{R} = V|\mathcal{I})$ représentée sur la partie inférieure de la figure 3.9 est obtenue par la formule de Bayes

$$P(\mathcal{R} = V|\mathcal{I}) = \frac{f(\mathcal{I}|\mathcal{R} = V)P(\mathcal{R} = V)}{f(\mathcal{I})}. \tag{3.60}$$

On voit qu'elle vaut 0.5 au point de croisement des trois courbes du haut, c'est-à-dire au point où

$$f(\mathcal{I}) = f(\mathcal{I}|\mathcal{R} = V) = f(\mathcal{I}|\mathcal{R} = F), \tag{3.61}$$

parce que les classes sont a priori équiprobables.

Sur la partie inférieure de la figure on a illustré la règle de décision du banquier par un seuil supposé indépendant de $\mathcal{I}$ (ce qui n'est pas nécessairement vrai en pratique comme il ressort des formules générales indiquées ci-dessus). L'ensemble des valeurs de $\mathcal{I}$ pour lesquelles le crédit est alloué est l'intervalle indiqué sur la figure.

(Suggestion : trouver l'expression générale en fonction de $P(\mathcal{R} = V)$, de la relation entre $f(\mathcal{I}|\mathcal{R} = V)$ et $f(\mathcal{I}|\mathcal{R} = F)$ au point où $P(\mathcal{R} = V|\mathcal{I}) =$ seuil.)

3.7 MODÈLES PROBABILISTES

Bien souvent, dans le cadre de nos raisonnements théoriques, plutôt que de passer par la définition d'une expérience aléatoire, d'un univers de résultats possibles, d'une loi de probabilités, et de variables aléatoires telles que nous venons de le décrire, on postulera qu'une ou un certain nombre de variables aléatoires relatives à un problème sont distribuées selon une loi de probabilité donnée (p.ex. une loi uniforme, Gaussienne, binomiale,...). Parfois, on postule seulement la structure de cette loi de probabilités et laisse indéfinis les paramètres (p.ex. la moyenne et l'écart type dans le cas d'un loi Gaussienne), sans même décrire explicitement la nature de l'expérience.

Par exemple, dans le cadre du paradigme de Shannon lorsque nous étudierons les sources d'information nous serons dans certains cas amenés à postuler que les lettres successives des message de source sont distribuées de façon uniforme et indépendante. Ensuite, lorsque nous nous intéresserons à l'étude des erreurs de communication, nous introduirons une hypothèse supplémentaire en ce qui concerne la relation entrée-sortie du canal, par exemple en supposant que celui-ci transmet les caractères successifs en choisissant pour chacun un caractère de sortie selon une loi de probabilités conditionnelle constante, qui ne fait intervenir que le caractère présenté à l'entrée.

En fait, lorsqu'on étudie les relations entre un certain nombre de variables aléatoires définies à partir d'une expérience aléatoire, il n'est pas nécessaire de connaître tous les détails relatifs à cette expérience. Il suffit en effet de connaître la loi de probabilité conjointe des variables aléatoires en considération. Nous allons dans les chapitres suivants développer un certain nombre de modèles probabilistes qui peuvent être utilisés à cette fin, et nous en étudierons les propriétés de façon détaillée.

En cas de doute, nous encourageons le lecteur à réfléchir le cas échéant à une définition raisonnable de l'expérience aléatoire sous-jacente.

3.8 SUITES DE V.A. ET NOTIONS DE CONVERGENCE

Nous serons amenés dans la seconde partie de ce cours à manipuler des ensembles infinis de variables aléatoires, par exemples des suites de variables indicées par le temps appelées processus stochastiques.

Un certain nombre de résultats théoriques considèrent ces suites comme le passage à la limite d'un nombre fini de variables et reposent sur différentes notions de convergence lors de ce passage à la limite.

En particulier, on fera appel à plusieurs reprises à la loi des grands nombres pour démontrer les théorèmes de la théorie de l'information.

Les rappels et notations relatives à cette matière seront fournis ultérieurement sous la forme d'un appendice au chapitre 8.

3.9 EXERCICES

1. A partir des axiomes de Kolmogorov, montrer que :
   (a) $P(\emptyset) = 0$,
   (b) $P(\neg A) = 1 - P(A)$,
   (c) $A \subset B \Rightarrow P(A) \leq P(B)$, et
   (d) $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.
2. Montrer que $A \supset B \Rightarrow P(A|B) = 1$, et montrer que la réciproque est fausse ($P(B) > 0$, par hypothèse).
3. Montrer que si $P(B) > 0$, alors $P(\cdot|B)$ définit bien une loi de probabilité sur $(\Omega, \mathcal{E})$, c'est-à-dire vérifie les axiomes de Kolmogorov.
4. Montrer que $A \perp B \Leftrightarrow B \perp A$ (par hypothèse, $P(B), P(A) > 0$) et que $(A \perp B) \Leftrightarrow (\neg A \perp \neg B)$.
5. Propriéte négative (contre-exemple) : montrer que $(A \perp B) \not\Rightarrow (A \perp B|C)$.

4 MESURE QUANTITATIVE DE L'INFORMATION

Avertissement

La matière de ce chapitre constitue l'algèbre de base de la théorie de l'information, sur lequel repose tout le reste. Il est primordial de bien assimiler ces notions le plus rapidement possible pour tirer profit du cours oral. Les formules peuvent paraître très nombreuses à première vue, mais au fur et à mesure qu'on les assimilera on se rendra compte que l'ensemble des propriétés forme un tout cohérent et somme toute très intuitif.

Afin de permettre aux étudiants de s'entraîner aux raisonnements probabilistes et entropiques, nous avons décidé de faire les raisonnements de manière aussi explicite que possible. A diverses reprises nous mettons en évidence deux chemins permettant de démontrer telle ou telle propriété : le chemin "sûr", qui recourt aux définitions et n'exploite que des propriétés élémentaires de la fonction logarithme et du calcul de probabilités; un chemin plus direct, qui s'appuie sur les propriétés précédemment établies des fonctions d'entropie et de quantité d'information. Pour certains, cela peut paraître fastidieux et redondant. Dans ce cas, nous vous suggérons de tester vos connaissances en cherchant vous même les démonstrations et en essayant les exercices donnés en fin de chapitre.

A la fin de ce chapitre un formulaire collationne la plupart des formules utiles.

4.1 QUANTITE D'INFORMATION DE MESSAGES

Un problème fondamental en informatique est de reproduire à un moment donné et à un endroit particulier un message qui a été produit antérieurement et éventuellement à un endroit différent. Il faut donc pouvoir stocker et transporter (ou transmettre) des messages et la première question qui se pose est de décider comment quantifier de façon correcte, c'est-à-dire du point de vue des techniques à mettre en oeuvre, l'information contenue dans un message. De ce point de vue, la *quantité* d'information associée à un message doit être en relation avec la *quantité* de ressources physiques à mettre en oeuvre pour le stocker ou le transporter.

Souvent (certainement pas toujours...) les messages manipulés en informatique ont un *sens*; c'est-à-dire qu'ils sont à interpréter dans un contexte particulier en relation avec un système physique, ou des entités conceptuelles. Cependant, ces questions sémantiques ne sont pas pertinentes du point de vue de l'ingénieur qui doit développer des systèmes de stockage ou de transmission des messages. Ce qui compte de ce point de vue c'est que le message est *choisi* parmi un ensemble de messages possibles selon un certain procédé. Le système de stockage ou de transmission de l'information doit alors être conçu de manière à ce qu'il fonctionne correctement quel que soit le message sélectionné, et pas seulement celui qui sera effectivement choisi, car ce choix est inconnu au moment de la conception du système.

Notre but dans cette section est d'introduire de manière intuitive la mesure d'information logarithmique proposée par Shannon. Nous commençons par discuter du choix d'une mesure logarithmique purement *algébrique*, c'est-à-dire qui ne prend pas en compte des connaissances de nature probabiliste sur le mécanisme de choix des messages. Ensuite nous argumenterons de différents points de vue comment généraliser cette mesure au cas où nous voulons prendre en compte la probabilité de choix des messages, ce qui nous permettra de formuler une mesure probabiliste de la quantité d'information qui dans le cas de messages équiprobables dégénère en la mesure algébrique. On peut dire que l'oeuvre de Shannon a été de proposer cette mesure probabiliste et de montrer comment elle permet effectivement de quantifier les ressources physiques (taille, vitesse, densité, énergie, coût) à mettre en oeuvre pour traiter de manière efficace et fiable le stockage et la transmission de messages.

4.1.1 Information algébrique logarithmique

Si on ne veut pas tenir compte des probabilités de choix des messages, et si le nombre de messages possibles est fini, alors ce nombre ou n'importe quelle fonction monotone pourrait servir de mesure d'information.

Cependant, il est très naturel d'utiliser en tant que mesure de la quantité d'infomation le *logarithme* du nombre de messages possibles. En effet, soit alors Ω, un ensemble fini de messages possibles et désignons par $|\Omega|$ la taille de cet ensemble. Une définition purement "algébrique" de la quantité d'information d'un message choisi dans Ω serait alors :

Information algébrique

$$\mathrm{Info}_{\mathrm{alg}}(\Omega) = \log |\Omega|, \tag{4.1}$$

Bien que cette définition doive encore être généralisée pour tenir compte des probabilités de sélection des différents messages, en tout cas lorsque ceux-ci ne sont pas équiprobables, discutons d'abord pourquoi la fonction logarithme est en effet un choix raisonnable.

1. Les paramètres physiques de systèmes informatiques (temps, largueur de bande, taille) tendent tous à varier de manière logarithmique avec le nombre de possibilités. Par exemple, l'ajout d'une porte logique dans un circuit double le nombre d'états possibles de ce circuit, ce qui se traduit par une augmentation de 1 en base $\log_2$. De même, le nombre de messages différents qui peuvent être transmis pendant une durée T sur un canal de communication de largeur de bande F croît de façon exponentielle avec le produit TF.

2. Cette mesure est plus proche de notre perception intuitive "linéaire" de l'information. Par exemple, il semble intuitivement normal que la quantité d'information qu'on puisse stocker dans deux disquettes soit le double de la quantité d'information stockable dans une seule disquette.

3. Ce choix est mathématiquement plus approprié que tout autre choix. Un grand nombre des opérations qui deviennent linéaires (additives) lorsqu'on utilise le logarithme seraient autrement très lourdes à formuler.

Nous allons voir dans ce chapitre que la théorie de l'information *logarithmique* donne lieu à un algèbre (règles de combinaison) extrêmement simple. Le choix de la base du logarithme correspond à un choix d'unités (on peut passer d'un base a à une base b en multipliant les quantités exprimées dans la première base par $\log_b a$). Nous allons ici choisir la base deux, qui est la plus intuitive dans le cadre d'un monde où l'information est essentiellement représentée par des nombres binaires représentant l'état de systèmes dont les composants élémentaires n'ont que deux états stables.

4.1.2 Prise en compte des probabilités

Comment tenir compte dans la mesure d'information de probabilités non-uniformes de sélection de messages ?

Il nous faut définir une mesure fonction des probabilités des messages et qui soit cohérente avec ce que nous venons de discuter. Nous allons noter cette fonction d'information probabiliste pour le moment par

$$\mathrm{Info}_{\mathrm{prob}}$$

pour la distinguer de l'information algébrique définie plus haut. Cependant, si les messages sont équiprobables on doit avoir

$$\mathrm{Info}_{\mathrm{prob}} = \mathrm{Info}_{\mathrm{alg}}.$$

Voyons comment on peut tenir compte du fait que les messages ne sont pas équiprobables.

Continuité et passage à la limite. Supposons que notre mécanisme de sélection des messages consiste à diviser Ω en deux parties de même taille, Ω_1 et Ω_2, et à choisir un message de Ω_1 avec une probabilité $P(\Omega_1) = 1 - \epsilon$ ou un message de Ω_2 avec la probabilité $P(\Omega_2) = \epsilon$.

Alors, pour bien faire il faudrait que $\mathrm{Info}_{\mathrm{prob}}(\Omega)$ varie de manière continue en fonction de ϵ et que

$$\lim_{\epsilon \to 0} \mathrm{Info}_{\mathrm{prob}}(\Omega) = \mathrm{Info}_{\mathrm{prob}}(\Omega_1). \tag{4.2}$$

En d'autres termes, si certains messages n'ont aucune chance d'être choisis, il ne faudra pas comptabiliser ces messages dans la mesure d'information. Et si on fait varier le mécanisme de sélection des messages légèrement (dans notre exemple, si on change légèrement ϵ) il faut que la mesure d'information varie légèrement (continuité).

Additivité. Comme dans notre exemple (quelle que soit la valeur supposée de ϵ) on décompose le choix d'un message de Ω en deux étapes successives

1. choix d'un des ensembles Ω_1 ou Ω_2 : nous noterons par $\mathrm{Info}_{\mathrm{prob}}(\Omega_1, \Omega_2)$ l'information associée à ce choix.

2. choix d'un message dans l'ensemble Ω_i choisi à la première étape : $\mathrm{Info}_{\mathrm{prob}}(\Omega_i)$,

la quantité d'information totale doit pouvoir se décomposer en deux contributions associées à ces deux étapes. Par exemple, si Ω ne contient que deux éléments, les ensembles Ω_i sont des singletons et la seconde étape ne sert à rien (et ne produit donc pas d'information).

Plus généralement, en tenant compte des probabilités associées aux Ω_i on peut proposer la formule suivante pour combiner les informations associées aux deux étapes :

$$\mathrm{Info}_{\mathrm{prob}}(\Omega) = \mathrm{Info}_{\mathrm{prob}}(\Omega_1, \Omega_2) + P(\Omega_1)\mathrm{Info}_{\mathrm{prob}}(\Omega_1) + P(\Omega_2)\mathrm{Info}_{\mathrm{prob}}(\Omega_2). \tag{4.3}$$

Si nous supposons qu'à l'intérieur de chacun des deux ensembles les messages sont équiprobables et que $|\Omega_i| = \frac{|\Omega|}{2}$) on doit avoir

$$\mathrm{Info}_{\mathrm{prob}}(\Omega_i) = \mathrm{Info}_{\mathrm{alg}}(\Omega_i) = \log |\Omega_i| = \log |\Omega| - 1.$$

Par conséquent on obtient

$$\mathrm{Info}_{\mathrm{prob}}(\Omega) = \mathrm{Info}_{\mathrm{prob}}(\Omega_1, \Omega_2) + \log |\Omega| - 1. \tag{4.4}$$

Considérons alors les deux situations suivantes

- $\epsilon = 0.5$ (les deux ensembles sont équiprobables) : dans ce cas tous les messages de Ω sont équiprobables et si nous souhaitons être cohérent avec la mesure algébrique, il faut que dans ce cas on ait $\mathrm{Info}_{\mathrm{prob}}(\Omega) = \log |\Omega|$ ce qui implique que

$$\mathrm{Info}_{\mathrm{prob}}(\Omega_1, \Omega_2) = 1.$$

- $\epsilon \to 0$: il faut dans ce cas seulement comptabiliser les messages de Ω_1, ce qui implique que

$$\lim_{\epsilon \to 0} \mathrm{Info}_{\mathrm{prob}}(\Omega_1, \Omega_2) \to 0.$$

D'autre part, il est clair que $\mathrm{Info}_{\mathrm{prob}}(\Omega_1, \Omega_2)$ ne peut être qu'une fonction de $P(\Omega_1)$ et $P(\Omega_2)$, c'est-à-dire une fonction $f(\epsilon)$ telle que

$$\lim_{\epsilon \to 0} f(\epsilon) = 0.$$

De plus, il faut que $f(\cdot)$ soit une fonction symétrique par rapport à 0.5, car on peut évidemment inverser le rôle joué par Ω_1 et Ω_2 dans le raisonnement que nous venons de faire. D'autre part, puisque $\text{Info}_{\text{prob}}(\Omega)$ varie continument en fonction de ϵ, $f(\epsilon)$ doit être une fonction continue.

On peut proposer pour $f(\epsilon)$ la fonction suivante

$$f(\epsilon) = -[\epsilon \log \epsilon + (1-\epsilon)\log(1-\epsilon)]. \tag{4.5}$$

En effet, cette fonction est symétrique par rapport à 0.5, vaut (par passage à la limite) 0 lorsque $\epsilon = 0$ ou $\epsilon = 1$, et on a aussi $f(0.5) = 1$. Notons qu'en termes des probabilités de Ω_1 et Ω_2 on a

$$f(\epsilon) = \text{Info}_{\text{prob}}(\Omega_1, \Omega_2) = -[P(\Omega_1)\log P(\Omega_1) + P(\Omega_2)\log P(\Omega_2)]. \tag{4.6}$$

Supposons alors que Ω ne contienne au départ que deux messages, disons ω_1 et ω_2. On doit alors avoir, en fonction des formules (4.4) et (4.6)

$$\text{Info}_{\text{prob}}(\Omega) = -[P(\omega_1)\log P(\omega_1) + P(\omega_2)\log P(\omega_2)], \tag{4.7}$$

où nous notons par $P(\omega_i)$ la probabilité de sélection du message i. La généralisation à un nombre quelconque de messages de cette formule donne

$$\text{Info}_{\text{prob}}(\Omega) = -\sum_{i=1}^{|\Omega|} P(\omega_i)\log P(\omega_i). \tag{4.8}$$

Si les messages sont équiprobables on a

$$P(\omega_i) = \frac{1}{|\Omega|}$$

et la formule précédente donne bien la même valeur que la formulation purement algébrique :

$$\text{Info}_{\text{prob}}(\Omega) = \log|\Omega| = \text{Info}_{\text{alg}}. \tag{4.9}$$

4.1.3 Discussion

La mesure *quantitative* d'information proposée par Shannon est définie dans le cas d'un ensemble fini de messages par la forme (4.8). Etant donné la similarité de la forme mathématique de cette fonction avec la mesure d'entropie définie en thermodynamique, cette mesure porte aussi le nom d'*entropie*. On peut montrer (voir §4.5) qu'il s'agit de la seule fonction qui satisfait aux exigences de continuité et d'additivité que nous venons de discuter. Nous verrons aussi à la fin de ce chapitre comment cette notion peut se généraliser au cas continu, lorsque $\Omega = \mathbb{R}^n$.

Nous allons, dans les sections suivantes introduire de manière précise la notation et la terminologie que nous utiliserons dans la suite et nous étudierons de manière approfondie les propriétés des mesures d'information logarithmiques. Il faut cependant savoir que dans la littérature d'autres mesures (non logarithmiques) ont été proposées (elles ne répondent alors évidemment pas à toutes les exigences esthétiques formulées ci-dessus). Il semble donc opportun, avant de procéder à l'étude approfondie des propriétés mathématiques de nos mesures de justifier au moins de façon intuitive en quoi cette mesure est appropriée dans le cadre de nos préoccupations d'ingénieur.

Notons tout d'abord que la mesure (4.8) est exclusivement basée sur la notion d'imprévisibilité d'un événement et ne fait en aucun cas intervenir le sens des messages contenus dans Ω. D'ailleurs, pour la théorie que nous développons le fait que ces messages soient utiles ou inutiles, vrais ou faux, intéressants ou banals, n'intervient pas, ni d'ailleurs le fait que le message soit reçu par une seule personne ou par toute la planète. En effet, toutes ces propriétés dépendent du contexte et de la perception des sources et destinataires de la communication, et par souci de démocratie, nous ne voulons pas nous limiter à certains contextes, par exemple ceux où les messages émis seraient tous intéressants, pour un grand nombre d'utilisateurs. Mais il est vrai qu'un message même extrêmement intéressant pour tout le monde n'apporte pas d'information si c'est le seul message possible, c'est-à-dire si tout le monde connaît a priori le contenu de ce message avant qu'il ne soit partagé.

Caractérisons un message ω_i par la grandeur

$$i(\omega_i) \stackrel{\triangle}{=} -\log P(\omega_i). \tag{4.10}$$

La fonction $i(\cdot)$ définit en fait une variable aléatoire sur Ω (fonction mesurable à valeurs réelles positives définie sur Ω). On voit que la quantité d'information associée à $(\Omega, P(\cdot))$ est l'espérance mathématique de cette variable aléatoire

$$\text{Info}_{\text{prob}}(\Omega) = E\{i\}. \tag{4.11}$$

Nous dirons que l'observation d'un message ω_i apporte une information *propre* égale à $i(\omega_i)$. Plus généralement, si $A \subset \Omega$ désigne un événement de probabilité $P(A)$ nous dirons que la quantité d'information associée reçue par un observateur qui est informé de la réalisation de cet événement vaut

$$-\log P(A).$$

On voit que cette information est d'autant plus grande que l'événement était a priori improbable du point de vue de l'observateur. Inversement, si un événement est a priori très probable (à la limite certain) l'information qui lui est associée sera faible (à la limite nulle). La formule (4.11) montre que l'information associée au processus de choix des messages $(\Omega, P(\cdot))$ est la valeur moyenne des informations propres lorsqu'on répète un grand nombre de fois le processus de communication.

4.2 INFORMATION PROPRE ET MUTUELLE D'ÉVÉNEMENTS

Dans cette section nous revenons sur la notion d'information propre associée à l'observation d'un événement probabiliste, en partant d'un exemple simple de source d'information binaire. Nous introduirons ensuite quelques définitions utiles par la suite.

Supposons que nous observions une source binaire qui émet des symboles ω successifs indépendants ($\omega \in \{0, 1\}$). Nous noterons par $P(1)$ la probabilité a priori que le symbole suivant émis soit 1, et $P(0) = 1 - P(1)$ la probabilité qu'il s'agisse d'un 0. Nous excluons pour le moment le cas limite *certain*, c'est-à-dire où $P(1) \in \{0, 1\}$.

Comme nous venons de le dire, l'information apportée par l'observation d'un symbole ω émis par cette source sera intuitivement d'autant plus grande que ce symbole est improbable. En d'autres termes, nous mesurons cette quantité d'information par une fonction

$$h(\omega) = f(\frac{1}{P(\omega)}),$$

où $f(\cdot)$ est une fonction croissante. $f(\cdot)$ doit aussi satisfaire à la condition limite suivante :

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

de façon à ce que l'information apportée par un événement certain soit nulle.

D'autre part, nous souhaitons que l'observation conjointe de deux événements indépendants apporte une information égale à la somme des informations apportées par l'observation individuelle de ces deux événements. En d'autres termes,

$$h(\omega_1, \omega_2) = h(\omega_1) + h(\omega_2)$$

si ω_1 et ω_2 sont indépendants. Par exemple si nous observons deux symboles successifs ω_1, ω_2 de notre source (qui sont par hypothèse indépendants) nous aurons

$$h(\omega_1, \omega_2) = f(\frac{1}{P(\omega_1, \omega_2)}) = f(\frac{1}{P(\omega_1)P(\omega_2)}).$$

D'autre part,

$$h(\omega_1) + h(\omega_2) = f(\frac{1}{P(\omega_1)}) + f(\frac{1}{P(\omega_2)}),$$

et la fonction $f(\cdot)$ doit donc satisfaire la relation

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

ce qui nous conduit à utiliser la fonction logarithme.

Définition de l'information propre. Dans le cas général d'un univers Ω discret, nous nous intéressons à l'observation d'événements qui sont des sous-ensembles quelconques de Ω. On associe à la réalisation d'un événement $A \subset \Omega$ la *quantité d'information propre*

$$h(A) = -\log(P(A)). \tag{4.12}$$

Cette grandeur est toujours positive (elle est nulle seulement si l'événement A est certain). Elle tend vers l'infini pour un événement dont la probabilité tend vers zéro.

On peut en principe utiliser une base quelconque pour le logarithme. Nous utiliserons dans la suite toujours la base 2; dans ce cas l'unité d'information est le *Shannon*. Nous verrons plus loin que la quantité d'information moyenne mesure alors le nombre minimum de *bits* nécessaires en moyenne pour coder de façon binaire les symboles émis par une source. On peut donc interpréter la quantité d'information comme étant le nombre minimal de questions dichotomiques qu'il faut poser pour identifier un événement.

Définition de l'information conditionnelle. La formule (4.12) s'applique évidemment aussi dans le cas où on considère un événement qui peut s'écrire sous la forme $A \cap B$, c'est-à-dire qui résulte de la réalisation conjointe des événements A et B. On a alors

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

Lorsque A et B sont indépendants on a évidemment $P(B|A) = P(B)$ et on retrouve $h(A \cap B) = h(A) + h(B)$. Dans le cas général on définit la *quantité d'information conditionnelle* de B sachant que A est réalisé par

$$h(B|A) = -\log(P(B|A)), \tag{4.13}$$

et on peut donc écrire

$$h(A \cap B) = h(A) + h(B|A) = h(B) + h(A|B). \tag{4.14}$$

L'information conditionnelle $h(B|A)$ (resp. $h(A|B)$) mesure donc l'information fournie par l'observation de B (resp. A) qui n'était pas déjà fournie par l'observation de A (resp. B).

Illustration1. Le cas pratique qui nous intéressera dans la suite est celui où Ω désigne l'ensemble des paires entrée/sortie d'un canal. Et nous nous intéresserons alors plus particulièrement aux situations où A correspond à l'observation d'un message émis à l'entrée d'un canal, et B à l'observation du message reçu à la sortie de celui-ci.

Nous verrons qu'un canal pourra être modélisé par une loi de probabilité conditionnelle $P(B|A)$ qui décrit le caractère non-déterministe de la communication lié au bruit du canal. D'autre part, la probabilité $P(A|B)$ représente alors la connaissance que l'observateur situé à la sortie du canal peut avoir sur le message émis (A), lorsqu'il vient d'observer B. Dans cette vision, un canal parfait (sans bruit) correspondrait à une relation entrée/sortie déterministe, qui associerait à chaque message à son entrée un message unique à la sortie, différent pour chaque entrée. Cela se traduira, comme nous le verrons par une loi où les $P(B|A)$ (et aussi $P(A|B)$) sont soit égales à un, soit nulles.

Illustration2. Dans le cadre de l'apprentissage automatique, on cherche à construire des modèles prédictifs, c'est-à-dire des règles qui permettent à partir de l'observation de certaines caractéristiques d'un système de "deviner" d'autres grandeurs relatives à celui-ci. Par exemple, en diagnostic médical on cherche à mettre au point des règles qui permettent à partir de l'observation de certains *symptômes* de faire une hypothèse sur les causes possibles de ces symptômes (le diagnostic). Dans ce cas, B désigne un ensemble de symptômes et A un diagnostic; $h(A|B)$ désigne l'information apportée par la connaissance du diagnostic lorsque les symptômes sont connus. En d'autres termes, $h(A|B) = 0$ indique que les symptômes observés permettent de poser le diagnostic A avec certitude.

Définition de l'information mutuelle. On définit *l'information mutuelle* par

$$i(A;B) = h(A) - h(A|B). \tag{4.15}$$

On a donc,

$$i(A;B) = \log \frac{P(A|B)}{P(A)} = \log \frac{P(A \cap B)}{P(A)P(B)} = i(B;A),$$

et il s'agit bien d'une grandeur symétrique, comme son nom le sous-entend.

Discussion. L'information conditionnelle $h(A|B)$ peut être plus grande, égale, où plus petite que l'information propre $h(A)$. En effet, un événement a priori très probable peut devenir moins probable lorsqu'on observe un autre événement. Par exemple, l'issue d'un tournoi sportif peut paraître jouée à l'avance, mais les événements imprévus apparaissant en cours de partie peuvent renverser la tendance à la mi-temps. En conséquence, l'information mutuelle peut être positive, nulle ou négative.

Nous verrons cependant ci-dessous, que lorsqu'on considère l'information conditionnelle *moyenne* d'un *ensemble* complet d'événements mutuellement exclusifs, celle-ci est inférieure ou égale à l'information propre moyenne de cet ensemble d'événements. Par conséquent, nous verrons que (corollairement) l'information mutuelle *moyenne* ne peut pas être négative.

Revenons à l'equation (4.15) pour en discuter quelques cas particuliers. Par exemple si $A \supset B$, alors lorsque l'on a observé B on peut en déduire avec certitude que A se réalisera. En d'autres termes $A \supset B \Rightarrow P(A|B) = 1$ (noter que la réciproque est fausse !). On a alors, $h(A|B) = 0$ et

$$i(A;B) = h(A).$$

En d'autres termes, lorsqu'un des deux événements contient l'autre, l'information mutuelle est égale à l'information propre du plus grand des deux, et on aura nécessairement $h(B) \geq h(A)$.

Si de plus les deux événements sont équivalents $A = B$, on aura $i(A;B) = h(A) = h(B)$. Plus généralement, si les événements sont probabilistiquement équivalents, c'est-à-dire si $P(A) = P(A \cap B) = P(B)$, alors $i(A;B) = h(A) = h(B)$.

Information, incertitude et complexité. L'information quantitative associée à un événement, telle que nous l'avons définie ci-dessus, est essentiellement (et uniquement) liée au caractère plus ou moins incertain de celui-ci. On peut donc argumenter, et à juste titre, que le terme de quantité d'information est un peu surfait. Néanmoins, l'usage historique a depuis longtemps consacré l'utilisation de ce terme en technique.

Mais il est utile de se rappeler que la notion d'information est intrinsèquement plus générale et plus fondamentale que la notion d'incertitude modélisée par le calcul des probabilités. De nombreux travaux ont conduit à donner à la notion d'information des définitions plus générales, qui ne nécessitent pas de faire appel au calcul des probabilités. Par exemple, dans le domaine de l'informatique théorique on peut définir la notion d'information associée à une chaîne de caractères par le biais de la complexité du programme le plus court permettant à une machine de Turing de générer cette chaîne de caractères (notion de complexité de Kolmogorov [CT91]). Un avantage potentiel de ce type de démarche est de permettre une définition a priori de la notion d'information, à partir de laquelle on peut alors bâtir le calcul des probabilités. La discussion plus détaillée de ces idées sort cependant du cadre de ce cours.

Remarquons néanmoins ici que la quantité d'information augmente avec l'incertitude et qu'on utilise souvent de façon interchangeable les deux termes *incertitude* et *quantité d'information*. La quantité d'information peut être vue comme une mesure de la réduction de l'incertitude, suite à l'observation d'un événement. Nous introduirons ci-dessous le terme d'entropie qui mesure l'incertitude moyenne d'une variable aléatoire discrète; son nom se justifie par l'identité de sa définition mathématique avec l'entropie thermodynamique. Nous verrons dans la suite du cours que la notion d'entropie est en relation directe avec le nombre minimum de bits nécessaires par symbole source pour représenter des messages longs.

4.3 ENTROPIE ET INFORMATION MUTUELLE DE V.A.

Supposons que nous soyons en présence d'une source qui émet à des instants successifs $t \in \{t_0, t_1, \ldots\}$ des symboles successifs $s(t)$ faisant partie d'un *alphabet* fini $S = \{s_1, \ldots, s_n\}$. Nous supposerons également pour le moment que les symboles successifs sont indépendants et émis selon une même distribution de probabilité. Nous noterons par p_i la probabilité $P(s(t) = s_i)$. Nous dirons qu'il s'agit d'une source *stationnaire et sans mémoire*, ou simplement sans mémoire. Nous donnerons au chapitre 8 une définition précise de la notion de source (sous la forme de processus aléatoire en temps discret) et des notions de stationnarité et de mémoire. Pour le moment, disons que la stationnarité traduit le fait que la distribution de probabilité qui caractérise l'émission d'un symbole ne dépend pas du temps, et la "non mémoire" signifie qu'un symbole est statistiquement indépendant des symboles émis précédemment.

Autrement dit, nous avons défini une suite de variables aléatoires indépendantes et identiques qui correspondent aux symboles successifs émis par cette source.

4.3.1 Entropie ou quantité d'information moyenne de la source sans mémoire

La quantité d'information moyenne associée à chaque symbole de la source sans mémoire est définie comme l'espérance mathématique (notée $E\{\cdot\}$) de l'information propre fournie par l'observation de chacun des symboles possibles $\{s_1, \ldots, s_n\}$. Elle est notée

$$H(S) = E\{h(s)\} = -\sum_{i=1,n} p_i \log p_i. \tag{4.16}$$

Il s'agit de l'information qu'on obtiendrait en moyenne en observant en parallèle les symboles émis par un très grand nombre de sources sans mémoire identiques[1]. Comme la source est stationnaire et sans mémoire[2], il s'agit également de l'information moyenne par symbole, qu'on obtiendrait en observant une suite de symboles très longue émise par une seule source.

On appelle généralement $H(S)$ *l'entropie* de la source, ou de la variable aléatoire discrète qui lui correspond. Plus précisément il s'agit de l'entropie par symbole de cette source. On peut multiplier cette grandeur par la fréquence (moyenne s'il y a lieu) d'émission des symboles pour obtenir un *débit d'entropie*, mesuré en *Shannon/s*.

Plus généralement, étant donné une variable aléatoire discrète définie sur un univers Ω, c'est-à-dire une partition $\mathcal{X} = \{X_1, \ldots, X_n\}$ de Ω, on définit l'entropie de cette variable aléatoire par

Entropie d'une variable aléatoire

$$H(\mathcal{X}) \stackrel{\triangle}{=} -\sum_{i=1,n} P(X_i) \log P(X_i). \tag{4.17}$$

Remarque. Les formules (4.16, 4.17) supposent que les probabilités p_i et $P(X_i)$ soient non nulles. Si cependant certains événements (ou symboles) étaient de probabilité nulle, on peut calculer l'entropie en excluant ceux-ci de la somme. On peut aussi étendre la définition de l'entropie en effectuant le passage à la limite vers des probabilités nulles, et en exploitant le fait que

$$\lim_{x \to 0} x \log x = 0. \tag{4.18}$$

Les deux façons de faire conduisent évidemment au même résultat et nous étendrons donc la définition de l'entropie en autorisant des probabilités nulles, ce qui simplifiera l'écriture d'un certain nombre de formules dans la suite.

Fonction $H_2(p)$. Prenons le cas d'une variable aléatoire $\mathcal{X}$ binaire valant 1 avec une probabilité p. L'entropie de cette v.a. vaut

$$H(\mathcal{X}) = -p \log p - (1-p)\log(1-p) \stackrel{\triangle}{=} H_2(p). \tag{4.19}$$

La fonction $H_2(p)$ nous sera utile à de nombreuses reprises dans la suite. Nous avons représenté à la figure 4.1 comment cette fonction varie en fonction de p. En particulier, si p vaut 1 ou 0, $H_2(p) = 0$. La fonction est symétrique par rapport à $p = 0.5$, et maximale pour $p = 0.5$ et vaut alors $H_2(p) = 1$. Il est également important de remarquer que $H_2(p)$ est (strictement) concave (convexe $\cap$), ce qui se traduit mathématiquement par la propriété

$$\forall p_1 \neq p_2 \in [0,1], \forall q \in]0,1[: H_2(qp_1 + (1-q)p_2) > qH_2(p_1) + (1-q)H_2(p_2), \tag{4.20}$$

illustrée schématiquement sur le graphique de la figure 4.1.

[1]Ceci est une conséquence de la loi des grands nombres (voir chapitre 8).
[2]En fait, il s'agit ici de la propriété d'ergodicité (voir chapitre 8).

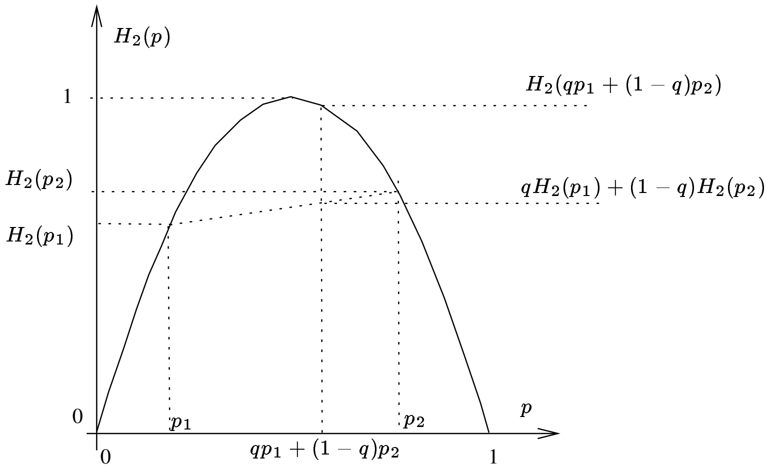


Figure 4.1. Fonction $H_2(p)$: entropie d'une v.a. binaire

4.3.2 Entropie conjointe de deux variables aléatoires

Soient $\mathcal{X} = \{X_1, \ldots, X_n\}$ et $\mathcal{Y} = \{Y_1, \ldots, Y_m\}$ deux variables aléatoires discrètes définies sur un même univers Ω. Alors *l'entropie conjointe* de $\mathcal{X}$ et $\mathcal{Y}$ est définie par

Entropie conjointe

$$H(\mathcal{X}, \mathcal{Y}) \triangleq -\sum_{i=1}^{n} \sum_{j=1}^{m} P(X_i, Y_j) \log P(X_i, Y_j). \tag{4.21}$$

Cette formule revient simplement à poser $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$ et à appliquer la définition de l'entropie (4.17) à cette nouvelle variable aléatoire, dont les valeurs possibles sont les combinaisons des valeurs de $\mathcal{X}$ et de $\mathcal{Y}$.

Il résulte de cette définition que l'entropie conjointe est une grandeur symétrique, c'est-à-dire que

$$H(\mathcal{X}, \mathcal{Y}) = H(\mathcal{Y}, \mathcal{X}).$$

La définition (4.21) peut évidemment s'étendre à un nombre quelconque de variables aléatoires :

Entropie conjointe (version générale)

$$H(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_m) \triangleq -\sum_{i_1=1}^{n_1} \cdots \sum_{i_m=1}^{n_m} P(X_{1,i_1}, \ldots, X_{m,i_m}) \log P(X_{1,i_1}, \ldots, X_{m,i_m}). \tag{4.22}$$

4.3.3 Entropie conditionnelle moyenne

De façon analogue, on définit *l'entropie conditionnelle* moyenne de $\mathcal{X}$ connaissant $\mathcal{Y}$ par

Entropie conditionnelle moyenne

$$H(\mathcal{X}|\mathcal{Y}) \triangleq -\sum_{i=1}^{n} \sum_{j=1}^{m} P(X_i, Y_j) \log P(X_i|Y_j). \tag{4.23}$$

Cette grandeur porte également le nom *d'équivocation* de $\mathcal{X}$ par rapport à $\mathcal{Y}$. Elle mesure en effet l'incertitude résiduelle ou le caractère ambigu ou équivoque de $\mathcal{X}$, lorsqu'on connaît $\mathcal{Y}$.

Montrons que l'entropie conditionnelle moyenne porte bien son nom, c'est-à-dire qu'il s'agit bien d'une moyenne d'entropies conditionnelles. L'entropie conditionnelle de $\mathcal{X}$ sachant que $\mathcal{Y} = Y_j$ est notée $H(\mathcal{X}|Y_j)$.

Elle est obtenue à partir de l'équation (4.17) où on remplace la loi de probabilités a priori de $\mathcal{X}$ par la loi de probabilités conditionnelles $P(\mathcal{X}|Y_j)$. Cela donne

$$H(\mathcal{X}|Y_j) = -\sum_{i=1}^{n} P(X_i|Y_j) \log P(X_i|Y_j), \tag{4.24}$$

et par conséquent

$$H(\mathcal{X}|\mathcal{Y}) = \sum_{j=1}^{m} P(Y_j) H(\mathcal{X}|Y_j). \tag{4.25}$$

On pourrait donc utiliser la formule (4.25) aussi bien que l'équation (4.23) comme définition de l'entropie conditionnnelle moyenne. Nous verrons plus loin que l'équation (4.25) permet de transposer un certain nombre de propriétés de l'entropie à l'entropie conditionnelle moyenne.

Notons ici que l'entropie conditionnelle moyenne *n'est pas* une grandeur symétrique, c'est-à-dire qu'on peut avoir $H(\mathcal{X}|\mathcal{Y}) \neq H(\mathcal{Y}|\mathcal{X})$.

Entropie a priori et entropie a posteriori. On utilise également souvent le terme *d'entropie a priori* pour désigner $H(\mathcal{X})$ et le terme *d'entropie a posteriori* (moyenne) pour désigner $H(\mathcal{X}|\mathcal{Y})$.

Nous verrons plus loin que l'entropie a posteriori $H(\mathcal{X}|\mathcal{Y})$ est inférieure ou égale à l'entropie a priori $H(\mathcal{X})$, l'égalité ayant lieu si et seulement si les deux variables aléatoires sont indépendantes. Au stade actuel de nos connaissances, nous pouvons vérifier que si $\mathcal{X} \perp \mathcal{Y}$ alors $H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{X})$ et donc aussi $H(\mathcal{Y}|\mathcal{X}) = H(\mathcal{Y})$. En effet, $\mathcal{X} \perp \mathcal{Y}$ implique que $P(X_i|Y_j) = P(X_i), \forall i, j$, ce qui implique que $H(\mathcal{X}|Y_j) = H(\mathcal{X}), \forall j$, ce qui implique aussi que $H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{X})$.

Le caractère naturel des définitions des entropies a priori et a posteriori résulte de la propriété suivante

Additivité

$$H(\mathcal{X}, \mathcal{Y}) = H(\mathcal{X}) + H(\mathcal{Y}|\mathcal{X}) = H(\mathcal{Y}) + H(\mathcal{X}|\mathcal{Y}), \tag{4.26}$$

qui se lit en disant que l'entropie conjointe de deux v.a. est égale à l'entropie a priori de l'une plus l'entropie conditionnelle de l'autre.

Cette propriété résulte des définitions (4.17), (4.21), (4.23) et de la formule de Bayes. Calculons $H(\mathcal{X}) + H(\mathcal{Y}|\mathcal{X})$ en faisant appel aux formules (4.17) et (4.23). On a

$$\begin{aligned}
H(\mathcal{X}) + H(\mathcal{Y}|\mathcal{X}) &= -\left(\sum_{i=1}^{n} P(X_i)\log P(X_i) + \sum_{i=1}^{n}\sum_{j=1}^{m} P(X_i, Y_j)\log P(Y_j|X_i)\right) && (4.27)\\
&= -\left(\sum_{i=1}^{n}\sum_{j=1}^{m} P(X_i, Y_j)\log P(X_i) + \sum_{i=1}^{n}\sum_{j=1}^{m} P(X_i, Y_j)\log P(Y_j|X_i)\right) && (4.28)\\
&= -\sum_{i=1}^{n}\sum_{j=1}^{m} P(X_i, Y_j)\log P(X_i)P(Y_j|X_i) && (4.29)\\
&= -\sum_{i=1}^{n}\sum_{j=1}^{m} P(X_i, Y_j)\log P(X_i, Y_j). && (4.30)
\end{aligned}$$

L'autre égalité se démontre de façon analogue.□

Un corollaire de (4.26) et de (4.25) est la propriété suivante :

Additivité (version générale)

$$H(\mathcal{X}, \mathcal{Y}|\mathcal{Z}) = H(\mathcal{X}|\mathcal{Z}) + H(\mathcal{Y}|\mathcal{X}, \mathcal{Z}). \tag{4.31}$$

En effet, calculons d'abord $H(\mathcal{X},\mathcal{Y}|Z_i)$. On a, en vertu de (4.26)

$$H(\mathcal{X},\mathcal{Y}|Z_i) = H(\mathcal{X}|Z_i) + H(\mathcal{Y}|\mathcal{X}, Z_i)$$

où le dernier terme vaut

$$H(\mathcal{Y}|\mathcal{X}, Z_i) = \sum_k P(X_k|Z_i)H(\mathcal{Y}|X_k, Z_i).$$

En multipliant les deux membres par $P(Z_i)$ et en sommant sur i on obtient donc

$$\sum_i P(Z_i)H(\mathcal{X},\mathcal{Y}|Z_i) = \sum_i P(Z_i)H(\mathcal{X}|Z_i) + \sum_i P(Z_i) \sum_k P(X_k|Z_i)H(\mathcal{Y}|X_k, Z_i),$$

où le dernier terme peut s'écrire aussi

$$\sum_i \sum_k P(X_k, Z_i)H(\mathcal{Y}|X_k, Z_i),$$

c'est-à-dire $H(\mathcal{Y}|\mathcal{X},\mathcal{Z})$.$\square$

Notons que la formule (4.31) est plus générale que (4.26) et englobe cette dernière comme cas particulier. En effet en considérant (4.31) dans le cas où la v.a. $\mathcal{Z}$ est une v.a. constante (une seule valeur Z_1 avec $P(Z_1) = 1$), on a $P(X_i, Y_j|Z_1) = P(X_i, Y_j)$, et (4.31) devient (4.26).

Remarque.

La formule (4.31) est aussi une conséquence directe de la formule (4.26). En effet, on a $H(\mathcal{X},\mathcal{Y},\mathcal{Z}) = H(\mathcal{X},\mathcal{Y}|\mathcal{Z}) + H(\mathcal{Z})$ *et aussi* $H(\mathcal{X},\mathcal{Y},\mathcal{Z}) = H(\mathcal{X},\mathcal{Z}) + H(\mathcal{Y}|\mathcal{X},\mathcal{Z})$*, et comme* $H(\mathcal{X},\mathcal{Z}) = H(\mathcal{Z}) + H(\mathcal{X}|\mathcal{Z})$ *on en déduit que* $H(\mathcal{X},\mathcal{Y}|\mathcal{Z}) = H(\mathcal{X}|\mathcal{Z}) + H(\mathcal{Y}|\mathcal{X},\mathcal{Z})$.

4.3.4 Information mutuelle moyenne

Une grandeur très importante associée à un couple de v.a. est *l'information mutuelle moyenne* définie par

Information mutuelle moyenne

$$I(\mathcal{X};\mathcal{Y}) \triangleq \sum_{i=1}^{n}\sum_{j=1}^{m} P(X_i, Y_j) \log \frac{P(X_i, Y_j)}{P(X_i)P(Y_j)}. \tag{4.32}$$

Ici aussi, il résulte de la définition que l'information mutuelle est une grandeur symétrique, c'est-à-dire que

$$I(\mathcal{X};\mathcal{Y}) = I(\mathcal{Y};\mathcal{X}).$$

Notons que l'équation (4.32) pourrait être généralisée à plusieurs variables aléatoires en restant symétrique. Cependant, cette généralisation ne s'avère pas utile en pratique. (A la section 4.4.5.2 nous reviendrons sur l'analyse des différentes grandeurs relatives à trois variables aléatoires et nous discuterons d'une grandeur que nous noterons $I(\mathcal{X};\mathcal{Y};\mathcal{Z})$.)

En recourant aux définitions, on pourra vérifier à ce stade qu'on a

$$I(\mathcal{X};\mathcal{Y}) = H(\mathcal{X}) - H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X}) = H(\mathcal{X}) + H(\mathcal{Y}) - H(\mathcal{X},\mathcal{Y}). \tag{4.33}$$

En termes du paradigme de Shannon, et en supposant que $\mathcal{X}$ représente le message émis par la source et $\mathcal{Y}$ celui reçu par le destinataire, cette décomposition revient à dire que l'information mutuelle moyenne est égale

- à l'information émise, diminuée de l'indétermination quant au symbole émis qui subsiste quand le symbole reçu est connu;
- symétriquement, à l'information reçue, diminuée de l'indétermination quant au symbole reçu qui subsiste quand le symbole émis est connu.

Note sur le point-virgule. Dans nos développements futurs nous allons considérer le cas où ces deux variables aléatoires sont des variables composites (p.ex. $\mathcal{X} = (\mathcal{X}_1, \ldots, \mathcal{X}_k)$ et $\mathcal{Y} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_p)$). Le point-virgule permet alors de donner un sens précis à la notation $I(\mathcal{X}_1, \ldots, \mathcal{X}_k; \mathcal{Y}_1, \ldots, \mathcal{Y}_p) = I(\mathcal{X}; \mathcal{Y})$.

4.3.5 Information mutuelle moyenne conditionnelle

On définit *l'information mutuelle moyenne conditionnelle* par

Information mutuelle moyenne conditionnelle

$$I(\mathcal{X}; \mathcal{Y}|\mathcal{Z}) \triangleq H(\mathcal{X}|\mathcal{Z}) - H(\mathcal{X}|\mathcal{Y}, \mathcal{Z}). \tag{4.34}$$

On peut aussi se convaincre à partir de (4.34) que

$$I(\mathcal{X}; \mathcal{Y}|\mathcal{Z}) = \sum_{i,j,k} P(X_i, Y_j, Z_k) \log \frac{P(X_i, Y_j|Z_k)}{P(X_i|Z_k)P(Y_j|Z_k)}, \tag{4.35}$$

ce qui implique que cette grandeur est également symétrique.

Nous verrons ci-dessous que cette grandeur a les mêmes propriétés que l'information mutuelle moyenne.

Exercice. Montrons que $H(\mathcal{X}, \mathcal{X}) = H(\mathcal{X})$, que $H(\mathcal{X}|\mathcal{X}) = 0$ et que $I(\mathcal{X}; \mathcal{X}) = H(\mathcal{X})$.

Il suffit en fait de montrer que $H(\mathcal{X}|\mathcal{X}) = 0$ car les autres égalités s'en déduisent. Or on a $P(X_i|X_j) = \delta_{i,j}$ (symbole de Kronecker), car si on sait que $\mathcal{X} = X_j$ toutes les autres valeurs de $\mathcal{X}$ sont impossibles. On aura donc, $\forall j$, $H(\mathcal{X}|X_j) = 0$ et donc aussi $H(\mathcal{X}|\mathcal{X}) = 0$ (c.q.f.d).

4.3.6 Règles de chaînage

Ci-dessous nous donnons les versions générales des formules permettant de développer les entropies et quantités d'information sur plusieurs variables en une somme de termes.

Probabilités. On peut développer une loi de probabilité conjointe de la manière suivante

$$\begin{aligned} P(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n) &= P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1)P(\mathcal{X}_3|\mathcal{X}_2, \mathcal{X}_1) \cdots P(\mathcal{X}_n|\mathcal{X}_{n-1}, \ldots, \mathcal{X}_1) \\ &\triangleq \prod_{i=1}^{n} P(\mathcal{X}_i|\mathcal{X}_{i-1}, \ldots, \mathcal{X}_1), \end{aligned} \tag{4.36}$$

où le premier facteur du membre de droite constitue un léger abus de notation. Cette formule est évidemment applicable quel que soit l'ordre choisi des variables et puisque le premier membre est invariant par rapport aux permutations des variables, c'est aussi le cas du second membre.

Entropies. La généralisation de la formule (4.26) au cas de n v.a. $\mathcal{X}_i, i = 1, \ldots, n$ donne lieu au résultat important suivant

$$\begin{aligned} H(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n) &= H(\mathcal{X}_1) + H(\mathcal{X}_2|\mathcal{X}_1) + H(\mathcal{X}_3|\mathcal{X}_2, \mathcal{X}_1) + \ldots + H(\mathcal{X}_n|\mathcal{X}_{n-1}, \ldots, \mathcal{X}_1) \\ &\triangleq \sum_{i=1}^{n} H(\mathcal{X}_i|\mathcal{X}_{i-1}, \ldots, \mathcal{X}_1). \end{aligned} \tag{4.37}$$

Cette formule s'obtient par application répétitive de la formule (4.31).

Entropies conditionnelles. A partir de l'équation (4.37) en faisant passer $H(\mathcal{X}_1)$ dans le premier membre on en déduit directement une règle de chaînage pour les entropies conditionnelles, à savoir

$$H(\mathcal{X}_2,\ldots,\mathcal{X}_n|\mathcal{X}_1) = H(\mathcal{X}_1,\mathcal{X}_2,\ldots,\mathcal{X}_n) - H(\mathcal{X}_1) = H(\mathcal{X}_2|\mathcal{X}_1) + H(\mathcal{X}_3|\mathcal{X}_2,\mathcal{X}_1) + \ldots + H(\mathcal{X}_n|\mathcal{X}_{n-1},\ldots,\mathcal{X}_1)$$

qui peut aussi se ré-écrire (en rebaptisant les variables et en en prenant une de plus)

$$\boxed{\begin{aligned} H(\mathcal{X}_1,\mathcal{X}_2,\ldots,\mathcal{X}_n|\mathcal{Y}) &= H(\mathcal{X}_1|\mathcal{Y}) + H(\mathcal{X}_2|\mathcal{X}_1,\mathcal{Y}) + \ldots + H(\mathcal{X}_n|\mathcal{X}_{n-1},\ldots,\mathcal{X}_1,\mathcal{Y}) \\ &\triangleq \sum_{i=1}^{n} H(\mathcal{X}_i|\mathcal{X}_{i-1},\ldots,\mathcal{X}_1,\mathcal{Y}) \qquad (4.38) \end{aligned}}$$

Informations. En exploitant la définition de l'information mutuelle conditionnnelle (4.34), on déduit des deux règles précédentes la formule suivante, dite de chaînage des informations

$$\boxed{I(\mathcal{X}_1,\mathcal{X}_2,\ldots,\mathcal{X}_n;\mathcal{Y}) = \sum_{i=1}^{n} I(\mathcal{X}_i;\mathcal{Y}|\mathcal{X}_{i-1},\ldots,\mathcal{X}_1). \qquad (4.39)}$$

La démonstration découle de l'application de la définition de l'information mutuelle, à savoir

$$I(\mathcal{X}_1,\mathcal{X}_2,\ldots,\mathcal{X}_n;\mathcal{Y}) = H(\mathcal{X}_1,\mathcal{X}_2,\ldots,\mathcal{X}_n) - H(\mathcal{X}_1,\mathcal{X}_2,\ldots,\mathcal{X}_n|\mathcal{Y})$$

et des règles de chaînage (4.37) et (4.38). On a en effet,

$$\begin{aligned} I(\mathcal{X}_1,\mathcal{X}_2,\ldots,\mathcal{X}_n;\mathcal{Y}) &= \sum_{i=1}^{n} H(\mathcal{X}_i|\mathcal{X}_{i-1},\ldots,\mathcal{X}_1) - \sum_{i=1}^{n} H(\mathcal{X}_i|\mathcal{X}_{i-1},\ldots,\mathcal{X}_1,\mathcal{Y}) && (4.40) \\ &= \sum_{i=1}^{n} [H(\mathcal{X}_i|\mathcal{X}_{i-1},\ldots,\mathcal{X}_1) - H(\mathcal{X}_i|\mathcal{X}_{i-1},\ldots,\mathcal{X}_1,\mathcal{Y})] && (4.41) \\ &= \sum_{i=1}^{n} I(\mathcal{X}_i;\mathcal{Y}|\mathcal{X}_{i-1},\ldots,\mathcal{X}_1). && (4.42) \end{aligned}$$

Nous verrons bientôt que tous les termes des membres de droite des formules de chaînage (4.37,4.38,4.39) sont positifs.

4.4 POSITIVITÉ, CONVEXITÉ ET MONOTONICITÉ

Nous venons de définir les principales grandeurs de la théorie de l'information et nous avons montré un certain nombre de relations qui existent entre ces grandeurs. Voyons quelles en sont les propriétés mathématiques principales.

Pour simplifier les écritures nous utiliserons ci-dessous la notation

$$H_n(p_1,p_2,\ldots,p_n) \triangleq -\sum_{i=1}^{n} p_i \log p_i, \qquad (4.43)$$

où il est sous-entendu que les p_i décrivent une loi de probabilité discrète, c'est-à-dire que $p_i \in [0,1]$ et $\sum_{i=1}^{n} p_i = 1$. Nous désignerons cette fonction par le terme *fonction d'entropie*, pour la distinguer conceptuellement des *mesures d'entropie* de variables aléatoires, construites à partir de cette fonction.

Nous étudions ci-dessous les propriétés de cette fonction et nous en déduisons un certain nombre de propriétés vérifiées par les entropies et informations moyennes (conditionnelles ou non).

4.4.1 Positivité de la fonction d'entropie

Cette propriété résulte de la définition de H_n. On a

$$H_n(p_1, p_2, \ldots, p_n) \geq 0. \tag{4.44}$$

De plus, $H_n = 0$, si, et seulement si, l'une des probabilités est égale à 1 (et donc toutes les autres valent 0). Nous noterons cette condition par $p_i = \delta_{i,j}$. L'événement j est alors certain et les autres sont impossibles.

Conséquences. On déduit de cette propriété les inégalités suivantes

$$H(\mathcal{X}) \geq 0 \quad \text{et} \quad H(\mathcal{X}, \mathcal{Y}) \geq 0 \quad \text{et} \quad H(\mathcal{X}|\mathcal{Y}) \geq 0. \tag{4.45}$$

L'entropie $H(\mathcal{X})$ est nulle si et seulement si une seule des valeurs de $\mathcal{X}$ est possible; $H(\mathcal{X}, \mathcal{Y})$ est nulle ssi une seule des combinaisons de valeurs de $\mathcal{X}$ et de $\mathcal{Y}$ est possible.

L'entropie conditionelle moyenne est nulle lorsque chacune des $H(\mathcal{X}|Y_j)$ est nulle, c'est-à-dire si $\mathcal{X}$ est une fonction de $\mathcal{Y}$.

Le fait que les entropies conditionnelles moyennes sont non négatives implique que

$$H(\mathcal{X}, \mathcal{Y}) \geq \max\left(H(\mathcal{X}), H(\mathcal{Y})\right). \tag{4.46}$$

Cette propriété est aussi appelée *monotonicité de l'entropie conjointe* que nous énoncerons de manière générale plus loin.

En anticipant sur la suite, on voit que la combinaison des deux équations (4.57) et de (4.26) donne

$$H(\mathcal{X}, \mathcal{Y}) \leq H(\mathcal{X}) + H(\mathcal{Y}). \tag{4.47}$$

De plus, (4.46) donne

$$2H(\mathcal{X}, \mathcal{Y}) \geq H(\mathcal{X}) + H(\mathcal{Y}), \tag{4.48}$$

ce qui se résume par

$$H(\mathcal{X}, \mathcal{Y}) \leq H(\mathcal{X}) + H(\mathcal{Y}) \leq 2H(\mathcal{X}, \mathcal{Y}) \tag{4.49}$$

la première égalité ayant lieu en cas d'indépendance, la seconde lorsque les deux v.a. sont probabilistiquement équivalentes.

4.4.2 Maximum de la fonction d'entropie

Pour n fixé, le maximum de $H_n(p_1, p_2, \ldots, p_n)$ est atteint pour une distribution uniforme, c'est-à-dire lorsque $\forall i \;: p_i = \frac{1}{n}$. Pour démontrer cette propriété nous allons d'abord démontrer le lemme suivant que nous réutiliserons à diverses reprises dans la suite.

Lemme (inégalité de Gibbs). Soient $(p_1, p_2, \ldots, p_n)$ et $(q_1, q_2, \ldots, q_n)$ deux distributions de probabilité sur un même univers fini. Alors,

$$\sum_{i=1}^{n} p_i \log \frac{q_i}{p_i} \leq 0, \tag{4.50}$$

l'égalité ayant lieu si, et seulement si, $\forall i \;: p_i = q_i$.

Pour démontrer (4.50), remarquons tout d'abord que la propriété est équivalente à

$$\sum_{i=1}^{n} p_i \ln \frac{q_i}{p_i} \leq 0, \tag{4.51}$$

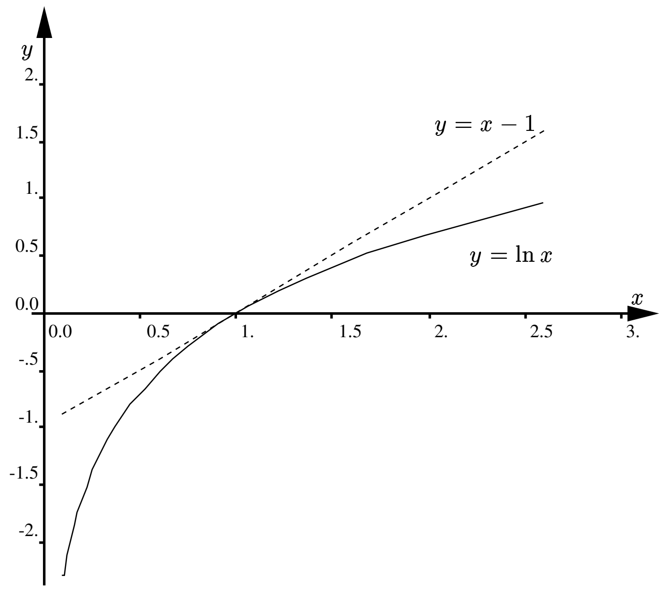


Figure 4.2. Relation entre $\ln x$ et $x - 1$

où les logarithmes sont exprimés en base e.

Nous supposerons tout d'abord tous les p_i non nuls et nous nous baserons sur la propriété suivante du logarithme népérien (voir figure 4.2) :

$$\ln x \leq x - 1, \tag{4.52}$$

l'égalité ayant lieu seulement pour $x = 1$. En remplaçant dans (4.52) x par $\frac{q_i}{p_i}$, en multipliant par p_i et en sommant sur i on obtient

$$\sum_{i=1}^{n} p_i \ln \frac{q_i}{p_i} \leq \sum_{i=1}^{n} p_i(\frac{q_i}{p_i} - 1) = \sum_{i=1}^{n} q_i - \sum_{i=1}^{n} p_i = 1 - 1 = 0,$$

ce qui démontre l'inégalité de Gibbs lorsque les p_i sont tous non nuls. On montrera à titre d'exercice que la propriété (4.50) reste vraie si certains des p_i sont nuls. □

Théorème. $H_n(p_1, p_2, \ldots, p_n) \leq \log n$, l'égalité ayant lieu si, et seulement si, $\forall i \; : p_i = \frac{1}{n}$.

En effet, appliquons l'inégalité de Gibbs à $q_i = \frac{1}{n}$. On trouve

$$\sum_{i=1}^{n} p_i \log \frac{1}{np_i} = \sum_{i=1}^{n} p_i \log \frac{1}{p_i} - \sum_{i=1}^{n} p_i \log n \leq 0 \Leftrightarrow$$

$$H_n(p_1, p_2, \ldots, p_n) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i} \leq \sum_{i=1}^{n} p_i \log n = \log n,$$

l'égalité ayant lieu si, et seulement si, tous les $p_i = q_i = \frac{1}{n}$ □

On interprète ce résultat en disant que le cas où tous les événements sont équiprobables correspond à l'incertitude maximale. Remarquons que nous trouvons la propriété relative au maximum de la fonction $H_2(p)$ en $p = 0.5$ sous forme de cas particulier.

Conséquences. $H(\mathcal{X})$ est donc maximale si les valeurs de $\mathcal{X}$ sont équiprobables; $H(\mathcal{X}, \mathcal{Y})$ sera maximale si toutes les combinaisons de valeurs de $\mathcal{X}$ et $\mathcal{Y}$ sont équiprobables, ce qui implique aussi que $\mathcal{X} \perp \mathcal{Y}$ et que $\mathcal{X}$ et $\mathcal{Y}$ sont des variables distribuées de manière uniforme. Enfin, $H(\mathcal{X}|\mathcal{Y})$ sera maximale si chacune des distributions conditionnelles $P(\mathcal{X}|Y_i)$ est uniforme, ce qui implique aussi que $\mathcal{X} \perp \mathcal{Y}$.

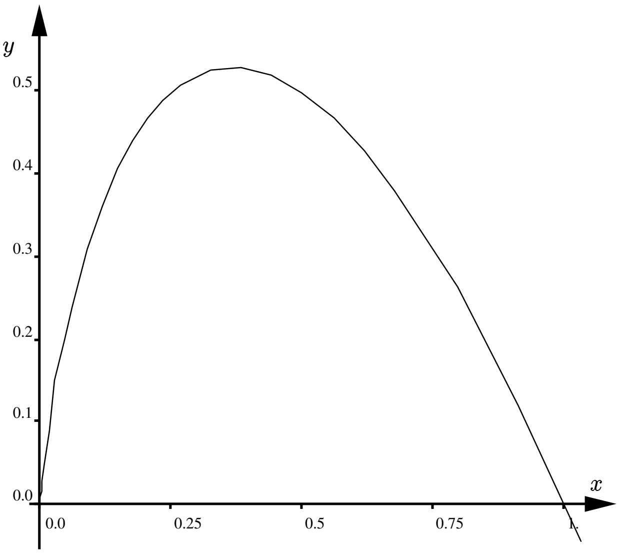


Figure 4.3. Fonction concave $f(x) = -x \log x$

4.4.3 Concavité de la fonction d'entropie

La généralisation de la propriété de concavité de la fonction $H_2(p)$ se formule de la manière suivante. Soient $(p_1, p_2, \ldots, p_n)$ et $(q_1, q_2, \ldots, q_n)$ deux distributions de probabilités discrètes et $\lambda \in [0, 1]$ alors

$$\begin{array}{c} H_n(\lambda p_1 + (1-\lambda)q_1, \ldots, \lambda p_n + (1-\lambda)q_n) \\ \geq \\ \lambda H_n(p_1, \ldots, p_n) + (1-\lambda)H_n(q_1, \ldots, q_n), \end{array} \tag{4.53}$$

en d'autres mots l'entropie de la moyenne est supérieure ou égale à la moyenne des entropies.

L'égalité est vérifiée lorsque soit $\lambda \in \{0, 1\}$ ou lorsque $\forall i \ : p_i = q_i$, et cela uniquement dans ce cas.

On en déduit facilement par induction que la propriété reste vraie pour la moyenne d'un nombre quelconque de distributions de probabilité, c'est-à-dire

$$\begin{array}{c} H_n(\sum_{j=1}^m \lambda_j p_{1j}, \ldots, \sum_{j=1}^m \lambda_j p_{nj}) \\ \geq \\ \sum_{j=1}^m \lambda_j H_n(p_{1j}, \ldots, p_{nj}), \end{array} \tag{4.54}$$

où $\lambda_i \in [0, 1]$, $\sum_{i=1}^m \lambda_i = 1$, et $\forall j \ : \sum_{i=1}^n p_{ij} = 1$. Dans ce cas, la condition nécessaire et suffisante pour que l'égalité ait lieu est que toutes les m distributions de probabilités mélangées soient identiques. En d'autres mots, que $\forall i, j \leq m \ : [\forall k \leq n \ : p_{ik} = p_{jk}]$.

La démonstration de cette propriété est obtenue aisément en faisant appel à la concavité de la fonction $f(x) = -x \log x$, dont on peut se convaincre graphiquement (voir figure 4.3). En effet, utilisant cette fonction on peut écrire

$$H_n(\sum_{j=1}^m \lambda_j p_{1j}, \ldots, \sum_{j=1}^m \lambda_j p_{nj}) = \sum_{i=1}^n f(\sum_{j=1}^m \lambda_j p_{ij}),$$

et par concavité de $f(\cdot)$ on a

$$f(\sum_{j=1}^m \lambda_j p_{ij}) \geq \sum_{j=1}^m \lambda_j f(p_{ij}),$$

ce qui donne

$$H_n(\sum_{j=1}^m \lambda_j p_{1j}, \ldots, \sum_{j=1}^m \lambda_j p_{nj}) \geq \sum_{i=1}^n \sum_{j=1}^m \lambda_j f(p_{ij}) = \sum_{j=1}^m \lambda_j \sum_{i=1}^n f(p_{ij}),$$

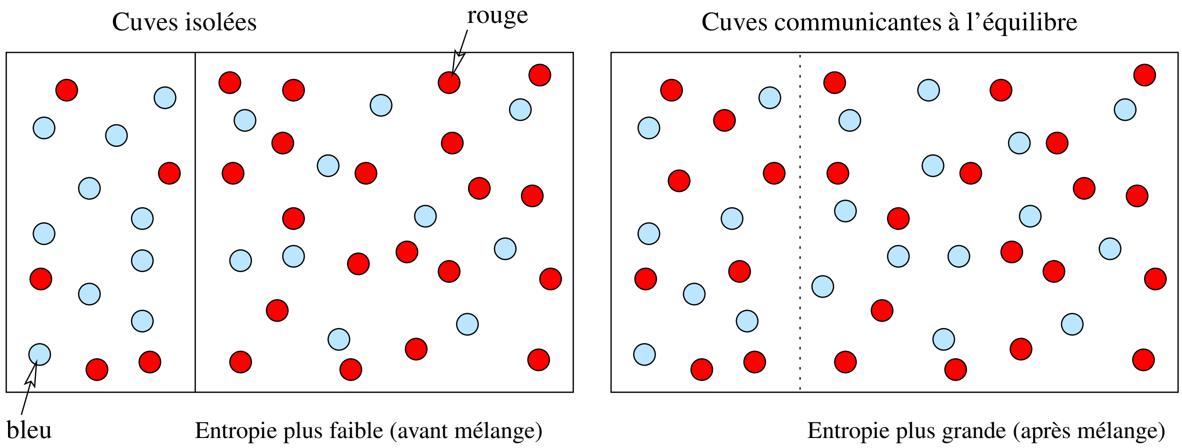


Figure 4.4. Second principe de la thermodynamique et concavité

or ce dernier terme vaut précisément $\sum_{j=1}^{m} \lambda_j H_n(p_{1j}, \ldots, p_{nj})$. □

Interprétation. La concavité de l'entropie donne lieu à une interprétation intéressante en rapport avec la thermodynamique. Supposons que nous disposions d'une grande cuve composée de m compartiments contenant chacun un mélange de n gaz inertes à la même pression, les différents mélanges ayant des compositions différentes. Nous savons que si nous enlevons les parois de séparation les gaz vont se mélanger avec un accroissement de l'entropie (second principe de la thermodynamique).

Ceci est illustré à la figure 4.4 où nous avons considéré le cas où $n = m = 2$. Avant mélange, le compartiment de gauche contient une proportion de 1/3 de molécules rouges (foncées) et celui de droite en contient une proportion de 2/3. Comme le compartiment de droite est deux fois plus grand que celui de gauche, il contient deux foix plus de molécules (puisque les pressions sont identiques). Avant mélange l'entropie moyenne de la cuve vaut donc $P(Gauche)H(Gauche) + P(Droite)H(Droite)$. Les entropies dans cette formule sont identiques et valent :

$$H(Droite) = H(Gauche) = 1/3 \log 3 + 2/3 \log 3/2 = 0.91829586 = H(Cuve).$$

Après mélange, la proportion de molécules rouges vaut

$$P(Rouge) = P(Gauche)P(Rouge|Gauche) + P(Droite)P(Rouge|Droite) = 1/9 + 4/9 = 5/9.$$

L'entropie de la cuve vaut donc

$$H(Cuve) = 4/9 \log 9/4 + 5/9 \log 9/5 = 0.99107605.$$

4.4.4 Propriétés de monotonicité

La positivité des entropies (conditionnelles ou non) entraîne la propriété générale suivante

Monotonicité (croissante) de l'entropie conjointe.

$$H(\mathcal{X}) \leq H(\mathcal{X}, \mathcal{Y}) \leq H(\mathcal{X}, \mathcal{Y}, \mathcal{Z}) \leq \ldots \tag{4.55}$$

avec

$$H(\mathcal{X}) = H(\mathcal{X}, \mathcal{Y}) \Leftrightarrow \mathcal{Y} = f(\mathcal{X}) \quad ; \quad H(\mathcal{X}, \mathcal{Y}) = H(\mathcal{X}, \mathcal{Y}, \mathcal{Z}) \Leftrightarrow \mathcal{Z} = g(\mathcal{X}, \mathcal{Y}) \quad ; \ldots \tag{4.56}$$

La propriété de convexité entraîne quant à elle que

Monotonicité (décroissante) de l'entropie vis-à-vis du conditionnement.

$$H(\mathcal{X}) \geq H(\mathcal{X}|\mathcal{Y}) \geq H(\mathcal{X}|\mathcal{Y},\mathcal{Z}) \geq \dots \tag{4.57}$$

avec

$$H(\mathcal{X}) = H(\mathcal{X}|\mathcal{Y}) \Leftrightarrow \mathcal{X} \perp \mathcal{Y} \quad ; \quad H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{X}|\mathcal{Y},\mathcal{Z}) \Leftrightarrow \mathcal{X} \perp \mathcal{Z}|\mathcal{Y} \quad ; \dots \tag{4.58}$$

Cette propriété se lit en disant que le conditionnement ne peut que diminuer l'entropie. La réalisation d'une variable aléatoire qui en conditionne une autre ne peut que réduire l'incertitude de cette dernière, c'est-à-dire en diminuer la quantité d'information.

Pour démontrer le cas $H(\mathcal{X}) \geq H(\mathcal{X}|\mathcal{Y})$ il suffit d'appliquer (4.54) au cas où, $p_{ij} = P(X_i|Y_j)$ et $\lambda_j = P(Y_j)$.

La démonstration de la seconde inégalité ($H(\mathcal{X}|\mathcal{Y}) \geq H(\mathcal{X}|\mathcal{Y},\mathcal{Z})$) découle du fait qu'on a $\forall Y_i$ la propriété

$$H(\mathcal{X}|Y_i) \geq H(\mathcal{X}|Y_i,\mathcal{Z})$$

puisque la première inégalité de (4.57) peut être appliquée aux distributions conditionnelles sous l'hypothèse de la réalisation de l'événement $\mathcal{Y} = Y_i$. Dans cette équation, l'égalité aura lieu seulement si $P(X_i, Z_k|Y_j) = P(X_i|Y_j)P(Z_k|Y_j), \forall i, j, k$, c'est-à-dire s'il y a indépendance conditionnelle $\mathcal{X} \perp \mathcal{Z}|\mathcal{Y}$.

Une conséquence de la monotonicité vis-à-vis du conditionnement est la positivité des quantités d'information.

Positivité des informations mutuelles (conditionnelles ou non).

$$I(\mathcal{X};\mathcal{Y}) \geq 0 \quad \text{et} \quad I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) \geq 0, \tag{4.59}$$

avec

$$I(\mathcal{X};\mathcal{Y}) = 0 \Leftrightarrow \mathcal{X} \perp \mathcal{Y} \quad \text{et} \quad I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) = 0 \Leftrightarrow \mathcal{X} \perp \mathcal{Y}|\mathcal{Z}. \tag{4.60}$$

Enfin, la règle de chaînage des informations mutuelles (4.39) et (4.59) entraînent une propriété de monotonicité de la quantité d'information

Monotonicité (croissante) de l'information mutuelle.

$$I(\mathcal{X};\mathcal{Y}) \leq I(\mathcal{X};\mathcal{Y},\mathcal{Z}) \leq I(\mathcal{X},\mathcal{T};\mathcal{Y},\mathcal{Z}) \dots \tag{4.61}$$

qui se lit en disant que la quantité d'information augmente lorsque l'un ou l'autre des deux ensembles de variables concernées augmente.

Suggestion. Montrer à partir de la définition de l'information mutuelle (eq. 4.32) et de l'inégalité de Gibbs que l'information mutuelle moyenne est positive ou nulle, et qu'elle est nulle si et seulement si les deux variables aléatoires sont indépendantes.

On pourrait alors "espérer" que l'information mutuelle diminue lorsqu'on conditionne, et notamment que $I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) \leq I(\mathcal{X};\mathcal{Y})$. Mais, cette propriété n'est pas vraie en général. Nous en verrons d'ailleurs des contre-exemples dans la suite. Cependant, nous discuterons à la section 4.4.6 d'un cas particulier où cette propriété est vraie.

4.4.5 Diagrammes de Venne

4.4.5.1 Deux variables. Le diagramme de Venne de la figure (4.5) résume de manière mnémotechnique les relations que nous venons de montrer, dans le cas de deux variables. Insistons ici sur le fait que ce diagramme ne constitue en rien une justification des formules; il s'agit simplement d'une représentation graphique des propriétés que nous avons démontrées, par analogie avec des notions ensemblistes.

Les propriétés que nous venons de montrer nous assurent que les trois parties du diagramme correspondent bien à des aires positives.

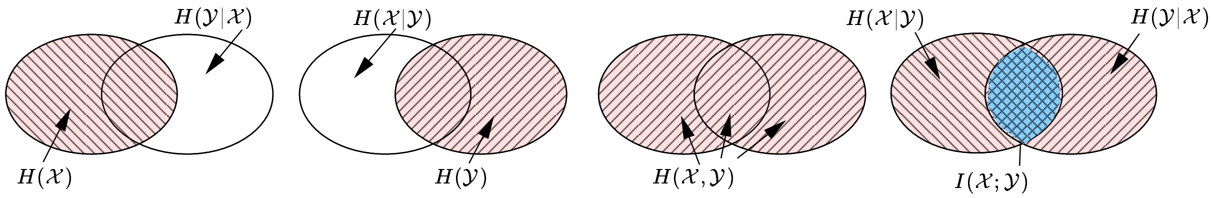


Figure 4.5. Diagramme de Venne des entropies relatives à deux variables

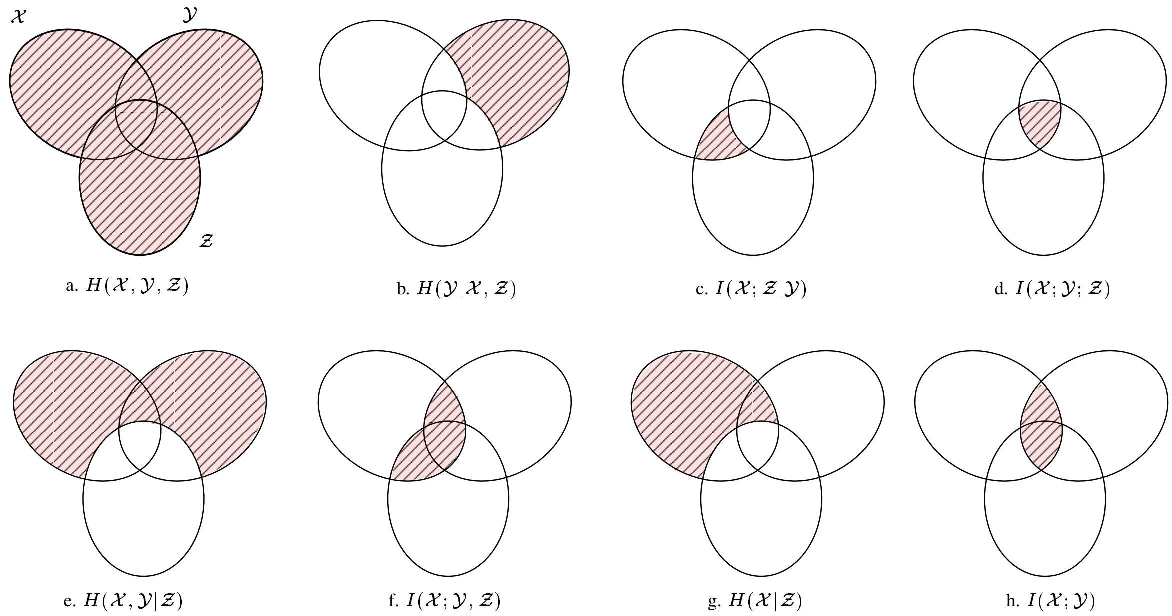


Figure 4.6. Diagramme de Venne pour trois variables aléatoires

4.4.5.2 Trois variables. Voyons ce que devient la figure 4.5 lorsqu'on considère trois variables aléatoires $\mathcal{X},\mathcal{Y},\mathcal{Z}$.

Les figures 4.6 représentent différentes entropies et informations mutuelles faisant intervenir trois variables aléatoires $\mathcal{X},\mathcal{Y},\mathcal{Z}$. Les figures 4.6.g et 4.6.h reprennent simplement deux grandeurs déjà représentées à la figure 4.5. Notons que les figures 4.6 b, c, e, f sont représentatives chacune de trois figures obtenues en intervertissant les rôles des trois variables aléatoires.

Essayons de voir si cette figure est cohérente avec les différentes propriétés démontrées ci-avant.

Par exemple, les figures 4.6e,b et g donnent la formule $H(\mathcal{X},\mathcal{Y}|\mathcal{Z}) = H(\mathcal{X}|\mathcal{Z}) + H(\mathcal{Y}|\mathcal{X},\mathcal{Z})$.

Les figures 4.6f, c et h donnent une version de la règle de chaînage des informations :

$$I(\mathcal{X};\mathcal{Y},\mathcal{Z}) = I(\mathcal{X};\mathcal{Y}) + I(\mathcal{X};\mathcal{Z}|\mathcal{Y}).$$

La règle de chaînage des entropies conjointes s'obtient à partir des figures 4.6a, b, g et du complémentaire (i.e. $H(\mathcal{Z})$) de la figure 4.6e.

La figure 4.6d introduit une nouvelle grandeur que nous avons notée $I(\mathcal{X};\mathcal{Y};\mathcal{Z})$. On peut la définir par exemple à partir des figures par

$$I(\mathcal{X};\mathcal{Y};\mathcal{Z}) \triangleq I(\mathcal{X};\mathcal{Y},\mathcal{Z}) - I(\mathcal{X};\mathcal{Z}|\mathcal{Y}) - I(\mathcal{X};\mathcal{Y}|\mathcal{Z}). \tag{4.62}$$

Il faut cependant vérifier que, conformément à ce qui est suggéré par la figure, cette grandeur est bien symétrique, c'est-à-dire qu'elle est invariante lorsqu'on permute les rôles de $\mathcal{X},\mathcal{Y},\mathcal{Z}$. Il découle de l'équation (4.62) qu'on peut permuter les rôles de $\mathcal{Y}$ et de $\mathcal{Z}$ sans changer le second membre. On a donc bien

$$I(\mathcal{X};\mathcal{Y};\mathcal{Z}) = I(\mathcal{X};\mathcal{Z};\mathcal{Y}).$$

D'autre part, on peut faire appel à la règle de chaînage pour reformuler le second membre de (4.62) par

$$I(\mathcal{X};\mathcal{Y};\mathcal{Z}) = I(\mathcal{X};\mathcal{Y}) + I(\mathcal{X};\mathcal{Z}|\mathcal{Y}) - I(\mathcal{X};\mathcal{Z}|\mathcal{Y}) - I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) = I(\mathcal{X};\mathcal{Y}) - I(\mathcal{X};\mathcal{Y}|\mathcal{Z}).$$

Or, nous savons que les deux termes du membre de droite de cette équation sont symétriques en $\mathcal{X}$ et $\mathcal{Y}$, et il s'en suit que nous pouvons affirmer que

$$I(\mathcal{X};\mathcal{Y};\mathcal{Z}) = I(\mathcal{Y};\mathcal{X};\mathcal{Z}).$$

Toutes les autres permutations pouvant être obtenues au moyen de ces deux permutations, nous avons bien montré que la grandeur $I(\mathcal{X};\mathcal{Y};\mathcal{Z})$ est symétrique.

Est-ce que cette grandeur est aussi positive ou nulle ? La réponse est non, ce qui veut donc dire que l'aire centrale de la figure 4.6d peut être négative. Il s'agit ici de la même propriété négative que nous avons déjà rencontrée à la fin du §4.4.4 en ce qui concerne la valeur relative de $I(\mathcal{X};\mathcal{Y}|\mathcal{Z})$ et $I(\mathcal{X};\mathcal{Y})$. Cette aire est en effet négative si $I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) > I(\mathcal{X};\mathcal{Y})$, c'est-à-dire si $\mathcal{X}$ et $\mathcal{Y}$ deviennent plus dépendantes lorsqu'on conditionne sur $\mathcal{Z}$. Dans la suggestion ci-dessous on rencontre le cas le plus extrême où cela se produit : deux variables sont indépendantes a priori et deviennent équivalentes après conditionnement.

Suggestion. Considérer un double lancer de pile ou face d'une pièce équilibrée, le second étant supposé indépendant du premier. Calculer ensuite les grandeurs de la figure 4.6 en considérant les trois variables aléatoires suivantes : $\mathcal{X} \in \{P, F\}$ *désigne le résultat du premier lancer;* $\mathcal{Y} \in \{P, F\}$ *désigne le résultat du second lancer;* $\mathcal{Z} \in \{T, F\}$ *vaut* T *si les deux lancers donnent le même résultat,* F *sinon. Trouver les valeurs des 7 aires élémentaires de la figure 4.6.*

4.4.6 Principe de non-création d'information

Nous allons terminer l'énumération des propriétés fondamentales des mesures d'information par la démonstration du principe dit de *non-création d'information*. Ce principe dit simplement qu'il est impossible de créer de l'information au moyen de manipulations quelconques de traitement de données.

Plus précisément, supposons que nous nous intéressions à une v.a. $\mathcal{X}$, et que nous disposions d'une v.a. $\mathcal{Y}$ à l'aide de laquelle nous voulons réduire notre incertitude sur $\mathcal{X}$. Alors le principe de non-création d'information dit que quels que soient les traitements (déterministes ou non) effectués sur $\mathcal{Y}$, l'information apportée sur $\mathcal{X}$ par le résultat de ces traitements ne peut pas augmenter.

Soit en effet une v.a. $\mathcal{Z}$ résultant d'une telle manipulation. On a nécessairement $P(\mathcal{Z}|\mathcal{X},\mathcal{Y}) = P(\mathcal{Z}|\mathcal{Y})$ car sinon il serait impossible de construire cette variable à l'aide de la seule connaissance de $\mathcal{Y}$. Par conséquent on a

$$P(\mathcal{X},\mathcal{Y},\mathcal{Z}) = P(\mathcal{X})P(\mathcal{Y}|\mathcal{X})P(\mathcal{Z}|\mathcal{Y}); \tag{4.63}$$

on dit que les v.a. $\mathcal{X},\mathcal{Y},\mathcal{Z}$ *forment une chaîne de Markov*.

On peut se convaincre facilement que (4.63) implique également que

$$P(\mathcal{X},\mathcal{Z}|\mathcal{Y}) = P(\mathcal{X}|\mathcal{Y})P(\mathcal{Z}|\mathcal{Y}), \tag{4.64}$$

c'est-à-dire que $\mathcal{X}$ et $\mathcal{Z}$ sont indépendantes lorsque $\mathcal{Y}$ est connu.
(Suggestion : appliquer la définition des probabilités conditionnelles à $P(\mathcal{X},\mathcal{Z}|\mathcal{Y})$*, puis se servir de (4.63).)*

On en déduit (en multipliant (4.64) par $P(\mathcal{Y})$ et en tenant compte du fait que $P(\mathcal{Z})P(\mathcal{Y}|\mathcal{Z}) = P(\mathcal{Y})P(\mathcal{Z}|\mathcal{Y})$) que

$$P(\mathcal{X},\mathcal{Y},\mathcal{Z}) = P(\mathcal{Z})P(\mathcal{Y}|\mathcal{Z})P(\mathcal{X}|\mathcal{Y}), \tag{4.65}$$

en d'autres termes que $\mathcal{Z},\mathcal{Y},\mathcal{X}$ forment également une chaîne de Markov. On résume ces deux propriétés par la notation $\mathcal{X} \leftrightarrow \mathcal{Y} \leftrightarrow \mathcal{Z}$.

Le principe de non-création d'information peut alors se formuler sous la forme du théorème suivant.

Théorème : non-création d'information
Si $\mathcal{X} \leftrightarrow \mathcal{Y} \leftrightarrow \mathcal{Z}$ *forment une chaîne de Markov alors*

$$I(\mathcal{X};\mathcal{Y}) \geq I(\mathcal{X};\mathcal{Z}) \quad \textit{et} \quad I(\mathcal{Y};\mathcal{Z}) \geq I(\mathcal{X};\mathcal{Z}). \tag{4.66}$$

En effet la règle de chaînage des quantités d'information appliquée de deux façons au calcul de $I(\mathcal{X};\mathcal{Y},\mathcal{Z})$ donne

$$I(\mathcal{X};\mathcal{Z}) + I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) = I(\mathcal{X};\mathcal{Y},\mathcal{Z}) = I(\mathcal{X};\mathcal{Y}) + I(\mathcal{X};\mathcal{Z}|\mathcal{Y}). \tag{4.67}$$

Mais, puisque $\mathcal{X}$ et $\mathcal{Z}$ sont conditionnellement indépendantes si $\mathcal{Y}$ est connu, on a $I(\mathcal{X};\mathcal{Z}|\mathcal{Y}) = 0$, et donc $I(\mathcal{X};\mathcal{Z}) \leq I(\mathcal{X};\mathcal{Y})$.

D'autre part, puisque nous pouvons intervertir le rôle des deux variables extrêmes on en tire aussi que $I(\mathcal{X};\mathcal{Z}) \leq I(\mathcal{Y};\mathcal{Z})$. □

4.4.6.1 Corollaires.

1. Puisque ce théorème s'applique en particulier lorsque $\mathcal{Z} = g(\mathcal{Y})$, on a

$$\boxed{I(\mathcal{X};\mathcal{Y}) \geq I(\mathcal{X};g(\mathcal{Y})),} \tag{4.68}$$

quelle que soit la fonction $g(\cdot)$.

2. Un autre corrollaire qui se déduit du théorème de non-création d'information est

$$\boxed{[\mathcal{X} \leftrightarrow \mathcal{Y} \leftrightarrow \mathcal{Z}] \Rightarrow I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) \leq I(\mathcal{X};\mathcal{Y}).} \tag{4.69}$$

En effet, la formule (4.67) donne, puisque $I(\mathcal{X};\mathcal{Z}|\mathcal{Y}) = 0$ pour une chaîne de Markov,

$$I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) = I(\mathcal{X};\mathcal{Y}) - I(\mathcal{X};\mathcal{Z}) \leq I(\mathcal{X};\mathcal{Y}).$$

On a donc également ici $I(\mathcal{X};\mathcal{Y};\mathcal{Z}) \geq 0$. *Mais, nous ré-insistons sur le fait qu'en général cette propriété n'est pas vraie. Le conditionnement peut augmenter l'information mutuelle.*

3. Du théorème de non-création d'information on peut déduire une propriété intéressante en ce qui concerne l'indépendance de deux v.a. Elle s'énonce comme suit

$$[\mathcal{X} \perp \mathcal{Y}] \Leftrightarrow [\forall f(\cdot), g(\cdot)\ : f(\mathcal{X}) \perp g(\mathcal{Y})]. \tag{4.70}$$

En d'autres termes, si deux variables aléatoires sont indépendantes alors des fonctions quelconques de ces deux v.a. sont encore indépendantes, et réciproquement.

La réciproque est triviale : il suffit de prendre pour $f(\cdot)$ et $g(\cdot)$ la fonction identité. Dans l'autre sens, la propriété découle directement du théorème de non-création d'information et du fait que l'information mutuelle de deux v.a. est nulle si et seulement si elles sont indépendantes. En effet, en vertu de ces propriétés on peut faire le raisonnement suivant

$$[\mathcal{X} \perp \mathcal{Y}] \Leftrightarrow [I(\mathcal{X};\mathcal{Y}) = 0] \Rightarrow [I(f(\mathcal{X});\mathcal{Y}) = 0] \Rightarrow [I(f(\mathcal{X});g(\mathcal{Y})) = 0] \Leftrightarrow [f(\mathcal{X}) \perp g(\mathcal{Y})], \tag{4.71}$$

en tenant compte aussi du fait que l'information mutuelle ne peut pas être négative.

4.4.6.2 Généralisation à un nombre quelconque de variables aléatoires.

Note. Cette section constitue un bon exercice de manipulation de la notion d'indépendance conditionnelle, et aussi une introduction aux processus de Markov que nous investiguerons avec plus d'attention dans les chapitres suivants. Nous ferons ici la plupart des raisonnements de manière explicite, pour permettre au lecteur de s'entraîner aux raisonnements probabilistes auxquels nous ferons souvent appel dans la suite de ces notes.

Considérons un nombre quelconque de variables aléatoires $\mathcal{X}_1, \ldots, \mathcal{X}_n$ et introduisons les notations suivantes : si $1 \leq k \leq l \leq n$

$$\mathcal{X}_k^l \triangleq \mathcal{X}_k, \ldots, \mathcal{X}_l \tag{4.72}$$

que nous simplifions lorsque $k = 1$ en

$$\mathcal{X}^l \triangleq \mathcal{X}_1, \ldots, \mathcal{X}_l. \tag{4.73}$$

On dit que les variables $\mathcal{X}_1, \ldots, \mathcal{X}_n$ forment une chaîne de Markov (de longueur n) si

$$P(\mathcal{X}^n) = P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1)P(\mathcal{X}_3|\mathcal{X}_2)\cdots P(\mathcal{X}_n|\mathcal{X}_{n-1}), \tag{4.74}$$

ce qu'on note par

$$\mathcal{X}_1 \leftrightarrow \mathcal{X}_2 \leftrightarrow \cdots \leftrightarrow \mathcal{X}_n. \tag{4.75}$$

En réalité cette notation suggère plus que ce qui est exprimé par la formule (4.74): d'une part elle suggère qu'on peut inverser l'ordre des variables, c'est-à-dire qu'on a aussi

$$\mathcal{X}_n \leftrightarrow \mathcal{X}_{n-1} \leftrightarrow \cdots \leftrightarrow \mathcal{X}_1; \tag{4.76}$$

d'autre part, elle suggère (transitivité du symbole $\leftrightarrow$) qu'on aurait également la même propriété pour un sous-ensemble quelconque des variables à condition de garder leur ordre original, en d'autres termes que si $i_1, i_2, \ldots, i_k$ est une suite croissante d'indices extraits de $1, 2, \ldots n$, on aurait également

$$\mathcal{X}_{i_1} \leftrightarrow \mathcal{X}_{i_2} \leftrightarrow \cdots \leftrightarrow \mathcal{X}_{i_k}. \tag{4.77}$$

Nous allons montrer que la seule propriété (4.74) implique bien toutes ces propriétés induites par notre notation $\leftrightarrow$, et que donc cette notation est bien licite.

Commençons par montrer par induction que (4.74) implique que $\forall i \leq n$ on a

$$P(\mathcal{X}^i) = P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1)P(\mathcal{X}_3|\mathcal{X}_2)\cdots P(\mathcal{X}_i|\mathcal{X}_{i-1}). \tag{4.78}$$

En effet, il suffit de *marginaliser* (4.74) par rapport à la variable $\mathcal{X}_n$. On a

$$\begin{aligned}
P(\mathcal{X}^{n-1}) &\overset{a}{=} \sum_{\mathcal{X}_n} P(\mathcal{X}^n) && (4.79)\\
&\overset{b}{=} \sum_{\mathcal{X}_n} P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1)P(\mathcal{X}_3|\mathcal{X}_2)\cdots P(\mathcal{X}_{n-1}|\mathcal{X}_{n-2})P(\mathcal{X}_n|\mathcal{X}_{n-1}) && (4.80)\\
&\overset{c}{=} P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1)P(\mathcal{X}_3|\mathcal{X}_2)\cdots P(\mathcal{X}_{n-1}|\mathcal{X}_{n-2})\sum_{\mathcal{X}_n} P(\mathcal{X}_n|\mathcal{X}_{n-1}) && (4.81)\\
&\overset{d}{=} P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1)P(\mathcal{X}_3|\mathcal{X}_2)\cdots P(\mathcal{X}_{n-1}|\mathcal{X}_{n-2}), && (4.82)
\end{aligned}$$

où le passage (a) est ce qu'on appelle la propriété de marginalisation du calcul de probabilités (le second membre décompose en fait la loi de probabilité du premier en une somme de termes relatifs à un système complet d'événements), le passage (b) applique (4.74), (c) est une mise en évidence des facteurs qui ne dépendent pas de l'indice de sommation, et (d) résulte du fait que $P(\mathcal{X}_n|\mathcal{X}_{n-1})$ est une loi de probabilité quelle que soit la valeur de $\mathcal{X}_{n-1}$ et doit donc sommer sur 1. La propriété (4.78) s'obtient alors par induction sur n.

Montrons maintenant qu'on a aussi $1 \leq k \leq l \leq n$

$$P(\mathcal{X}_k^l) = P(\mathcal{X}_k)P(\mathcal{X}_{k+1}|\mathcal{X}_k)P(\mathcal{X}_{k+2}|\mathcal{X}_{k+1})\cdots P(\mathcal{X}_l|\mathcal{X}_{l-1}). \tag{4.83}$$

On peut partir de (4.78) en utilisant $i = l$, puis marginaliser sur $\mathcal{X}_1$. Cela donne

$$\begin{aligned}
P(\mathcal{X}_2^l) &= \sum_{\mathcal{X}_1} P(\mathcal{X}^l) && (4.84)\\
&= \sum_{\mathcal{X}_1} P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1)P(\mathcal{X}_3|\mathcal{X}_2)P(\mathcal{X}_4|\mathcal{X}_3)\cdots P(\mathcal{X}_l|\mathcal{X}_{l-1}) && (4.85)\\
&= P(\mathcal{X}_3|\mathcal{X}_2)P(\mathcal{X}_4|\mathcal{X}_3)\cdots P(\mathcal{X}_l|\mathcal{X}_{l-1})\sum_{\mathcal{X}_1} P(\mathcal{X}_1, \mathcal{X}_2) && (4.86)\\
&= P(\mathcal{X}_3|\mathcal{X}_2)P(\mathcal{X}_4|\mathcal{X}_3)\cdots P(\mathcal{X}_l|\mathcal{X}_{l-1})P(\mathcal{X}_2) && (4.87)\\
&= P(\mathcal{X}_2)P(\mathcal{X}_3|\mathcal{X}_2)P(\mathcal{X}_4|\mathcal{X}_3)\cdots P(\mathcal{X}_l|\mathcal{X}_{l-1}) && (4.88)
\end{aligned}$$

et, à nouveau la thèse se déduit par induction.

Nous venons donc de démontrer que toute sous-suite contigüe de variables aléatoires de notre chaîne de Markov de départ forme encore une chaîne de Markov. En particulier on a

$$P(\mathcal{X}_{i-1}, \mathcal{X}_i, \mathcal{X}_{i+1}) = P(\mathcal{X}_{i-1})P(\mathcal{X}_i|\mathcal{X}_{i-1})P(\mathcal{X}_{i+1}|\mathcal{X}_i) \tag{4.89}$$

et donc aussi

$$P(\mathcal{X}_{i+1}|\mathcal{X}_i, \mathcal{X}_{i-1}) = P(\mathcal{X}_{i+1}|\mathcal{X}_i). \tag{4.90}$$

Mais, peut-on marginaliser sur des variables intermédiaires sans détruire la propriété de Markov ? Essayons d'éliminer de la sorte une variable quelconque, disons $\mathcal{X}_i$. En utilisant la notation $[\mathcal{X}_i]$ pour indiquer que la variable $\mathcal{X}_i$ est exclue, on a

$$\begin{aligned}
P(\mathcal{X}^n[\mathcal{X}_i]) &\stackrel{\triangle}{=} \sum_{\mathcal{X}_i} P(\mathcal{X}^n) && (4.91)\\
&= \sum_{\mathcal{X}_i} P(\mathcal{X}^{i+1}) P(\mathcal{X}_{i+2}|\mathcal{X}_{i+1}) \cdots P(\mathcal{X}_n|\mathcal{X}_{n-1}) && (4.92)\\
&= P(\mathcal{X}_{i+2}|\mathcal{X}_{i+1}) \cdots P(\mathcal{X}_n|\mathcal{X}_{n-1}) \sum_{\mathcal{X}_i} P(\mathcal{X}^{i+1}) && (4.93)\\
&= P(\mathcal{X}_{i+2}|\mathcal{X}_{i+1}) \cdots P(\mathcal{X}_n|\mathcal{X}_{n-1}) P(\mathcal{X}^{i+1}[\mathcal{X}_i]) && (4.94)
\end{aligned}$$

or

$$\begin{aligned}
P(\mathcal{X}^{i+1}[\mathcal{X}_i]) &= P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1) \cdots P(\mathcal{X}_{i-1}|\mathcal{X}_{i-2}) \sum_{\mathcal{X}_i} P(\mathcal{X}_i|\mathcal{X}_{i-1}) P(\mathcal{X}_{i+1}|\mathcal{X}_i) && (4.95)\\
&\stackrel{a}{=} P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1) \cdots P(\mathcal{X}_{i-1}|\mathcal{X}_{i-2}) \sum_{\mathcal{X}_i} P(\mathcal{X}_i|\mathcal{X}_{i-1}) P(\mathcal{X}_{i+1}|\mathcal{X}_i, \mathcal{X}_{i-1}) && (4.96)\\
&\stackrel{b}{=} P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1) \cdots P(\mathcal{X}_{i-1}|\mathcal{X}_{i-2}) \sum_{\mathcal{X}_i} P(\mathcal{X}_i, \mathcal{X}_{i+1}|\mathcal{X}_{i-1}) && (4.97)\\
&\stackrel{c}{=} P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1) \cdots P(\mathcal{X}_{i-1}|\mathcal{X}_{i-2}) P(\mathcal{X}_{i+1}|\mathcal{X}_{i-1}), && (4.98)
\end{aligned}$$

où en (a) nous avons exploité (4.90), en (b) la définition des probabilités conditionnelles, en (c) la marginalisation. En combinant (4.98) avec (4.94) on obtient bien la propriété (4.74) pour l'ensemble de variables dont on a exclu $\mathcal{X}_i$. Par conséquent, nous avons bien montré qu'on peut enlever un nombre quelconque de variables placées en position quelconque d'une suite de Markov sans détruire la propriété de Markov.

On peut déduire de ce qui précède la condition suivante. Si $i_1 < i_2 < \cdots < i_k$ est un sous-ensemble quelconque d'indices croissants extraits de $1, \ldots, n$ et si (4.74) est vraie, alors

$$\boxed{P(\mathcal{X}_{i_k}|\mathcal{X}_{i_1}, \ldots, \mathcal{X}_{i_{k-1}}) = P(\mathcal{X}_{i_k}|\mathcal{X}_{i_{k-1}}), \qquad (4.99)}$$

ce qui veut dire que l'information fournie par la variable la plus proche contient toute l'information des variables précédentes. La réciproque se formule comme suit : si (4.99) est vraie quelle que soit la suite d'indices consécutifs $i_1 < i_2 = i_1 + 1 < \cdots < i_k = i_1 + k - 1$ extraits de $1, \ldots, n$, alors (4.74) est vraie.

Mais, pouvons nous inverser l'ordre des variables tout en conservant la propriété de Markov ? Nous allons le montrer par induction (sachant que pour $n = 3$ on peut inverser l'ordre des variables). Supposons donc que

$$P(\mathcal{X}^{n-1}) = P(\mathcal{X}_{n-1})P(\mathcal{X}_{n-2}|\mathcal{X}_{n-1})P(\mathcal{X}_{n-3}|\mathcal{X}_{n-2}) \cdots P(\mathcal{X}_1|\mathcal{X}_2). \tag{4.100}$$

Alors

$$\begin{aligned}
P(\mathcal{X}^n) &= P(\mathcal{X}_n|\mathcal{X}^{n-1})P(\mathcal{X}^{n-1}) && (4.101)\\
&\stackrel{a}{=} P(\mathcal{X}_n|\mathcal{X}_{n-1})P(\mathcal{X}^{n-1}) && (4.102)\\
&\stackrel{b}{=} P(\mathcal{X}_n|\mathcal{X}_{n-1})P(\mathcal{X}_{n-1})P(\mathcal{X}_{n-2}|\mathcal{X}_{n-1})P(\mathcal{X}_{n-3}|\mathcal{X}_{n-2}) \cdots P(\mathcal{X}_1|\mathcal{X}_2) && (4.103)\\
&\stackrel{c}{=} P(\mathcal{X}_n, \mathcal{X}_{n-1})P(\mathcal{X}_{n-2}|\mathcal{X}_{n-1})P(\mathcal{X}_{n-3}|\mathcal{X}_{n-2}) \cdots P(\mathcal{X}_1|\mathcal{X}_2) && (4.104)\\
&\stackrel{d}{=} P(\mathcal{X}_n)P(\mathcal{X}_{n-1}|\mathcal{X}_n)P(\mathcal{X}_{n-2}|\mathcal{X}_{n-1})P(\mathcal{X}_{n-3}|\mathcal{X}_{n-2}) \cdots P(\mathcal{X}_1|\mathcal{X}_2), && (4.105)
\end{aligned}$$

où en (a) nous avons exploité (4.99), en (b) l'hypothèse inductive, et en (c) et (d) la définition des probabilités conditionnelles.

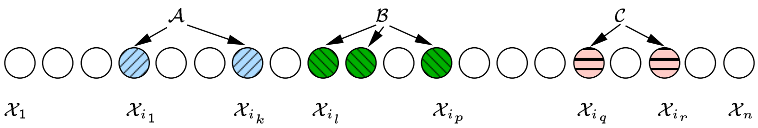


Figure 4.7. Sous-ensembles de variables d'une chaîne de Markov

De ce qui précède on peut déduire une autre propriété générale vérifiée par une chaîne de Markov. Prenons trois (ou plus) sous-ensembles de variables $\mathcal{A}, \mathcal{B}, \mathcal{C}$ qui ne se recouvrent pas (tels qu'illustrés à la figure 4.7). Nous avons

$$\mathcal{A} = \mathcal{X}_{i_1}, \mathcal{X}_{i_2}, \ldots \mathcal{X}_{i_k}, \tag{4.106}$$
$$\mathcal{B} = \mathcal{X}_{i_l}, \mathcal{X}_{i_{l+1}}, \ldots \mathcal{X}_{i_p}, \tag{4.107}$$
$$\mathcal{C} = \mathcal{X}_{i_q}, \mathcal{X}_{i_{q+1}}, \ldots \mathcal{X}_{i_r}, \tag{4.108}$$

où les indices $i_1, \ldots, i_k, i_l, \ldots, i_p, i_q \ldots i_r$ forment une sous-suite croissante de $1, \ldots, n$. Alors on a

$$P(\mathcal{A}|\mathcal{B},\mathcal{C}) = P(\mathcal{A}|\mathcal{B}) \quad \text{et} \quad P(\mathcal{C}|\mathcal{B},\mathcal{A}) = P(\mathcal{C}|\mathcal{B}), \tag{4.109}$$
$$P(\mathcal{A},\mathcal{B},\mathcal{C}) = P(\mathcal{A})P(\mathcal{B}|\mathcal{A})P(\mathcal{C}|\mathcal{B}) = P(\mathcal{C})P(\mathcal{B}|\mathcal{C})P(\mathcal{A}|\mathcal{B}), \tag{4.110}$$

ce qui veut dire que la propriété de Markov reste vérifiée après regroupement des variables. On a aussi

$$P(\mathcal{A},\mathcal{C}|\mathcal{B}) = P(\mathcal{A}|\mathcal{B})P(\mathcal{C}|\mathcal{B}) \tag{4.111}$$
$$P(\mathcal{A}|\mathcal{B}) = P(\mathcal{A}|\mathcal{X}_{i_l}) \tag{4.112}$$
$$P(\mathcal{C}|\mathcal{B}) = P(\mathcal{C}|\mathcal{X}_{i_p}). \tag{4.113}$$

En fait, on a même

$$P(\mathcal{B}|\mathcal{A},\mathcal{C}) = P(\mathcal{B}|\mathcal{X}_{i_k}, \mathcal{X}_{i_q}) \tag{4.114}$$

ce qui veut dire que seules les variables les plus proches doivent être gardées dans le conditionnement.

Conséquences sur les informations mutuelles et entropies conditionnelles. Nous allons énoncer sans démontrer les propriétés de monotonicité qui découlent de ce qui précède et du théorème de non-création d'information. Nous encourageons le lecteur à se persuader que ces propriétés sont bien vraies (les démonstrations sont immédiates).

Erosion de l'information
Si $k \leq l \leq p \leq q$ alors
$$I(\mathcal{X}_k; \mathcal{X}_q) \leq I(\mathcal{X}_l; \mathcal{X}_p). \tag{4.115}$$

Croissance de l'entropie conditionnelle (oubli)
Si $i \leq k \leq l$ alors

$$H(\mathcal{X}_l|\mathcal{X}_i) \geq H(\mathcal{X}_l|\mathcal{X}_k) = H(\mathcal{X}_l|\mathcal{X}_k, \mathcal{X}_i) \quad \text{et} \quad H(\mathcal{X}_i|\mathcal{X}_l) \geq H(\mathcal{X}_i|\mathcal{X}_k) = H(\mathcal{X}_i|\mathcal{X}_k, \mathcal{X}_l). \tag{4.116}$$

4.5 UNICITE DE L'EXPRESSION DE L'ENTROPIE

Nous avons introduit au début de ce chapitre la mesure d'information logarithmique à partir de quelques arguments intuitifs sur les propriétés que nous attendions de cette mesure. En particulier, nous avons souhaité l'additivité des entropies de v.a. indépendantes ce qui nous a conduit à proposer la mesure logarithmique. Les principales propriétés des différentes quantités introduites sont ensuite apparues comme conséquence de la convexité de la fonction $x \log x$. On peut cependant se demander si la formulation logarithmique que nous avons obtenue est unique à posséder ces propriétés, ou s'il existe d'autres fonctions susceptibles de servir comme mesure d'information. Or, on peut montrer qu'en postulant un certain nombre (très réduit) de propriétés sous la forme d'axiomes, la fonction logarithmique apparaît comme la seule mesure d'information acceptable. En fait, on trouve dans la littérature plusieurs axiomatisations qui ont été proposées pour l'entropie.

Par exemple, on peut montrer que la suite de fonctions $H_n(p_1, \ldots, p_n)$ qui satisfont à

Normalisation.
$H_2(\frac{1}{2}, \frac{1}{2}) = 1$;

Continuité.
$H_2(p, 1-p)$ est une fonction continue de $p \in [0, 1]$;

Groupement.
$H_n(p_1, \ldots, p_n) = H_{n-1}(p_1 + p_2, \ldots, p_n) + (p_1 + p_2) H_2(\frac{p_1}{p_1+p_2}, \frac{p_2}{p_1+p_2}), \forall n > 2$,

doivent avoir comme forme $H_n = -\sum_{i=1}^{n} p_i \log p_i$.

Notons que la relaxation de certaines de ces propriétés donne lieu à des familles de fonctions plus générales : on peut citer par exemple les entropies d'ordre β, définies comme suit

$$H_n^{\beta}(p_1, \ldots, p_n) \triangleq \sum_{i=1}^{n} p_i \cdot u^{\beta}(p_i), \tag{4.117}$$

où

$$u^{\beta}(p_i) \triangleq \frac{2^{\beta-1}}{2^{\beta-1} - 1}(1 - p_i^{\beta-1}). \tag{4.118}$$

Ces mesures comprennent comme cas particulier l'entropie de Shannon lorsque $\beta \to 1$ et l'indice *Gini* (ou entropie quadratique) lorsque $\beta = 2$.

Notons également que la propriété essentielle est l'additivité de l'entropie de v.a. indépendantes (qui apparaît ci-dessus sous la forme du groupement). C'est d'ailleurs cette propriété d'additivité (on dit aussi d'extensivité) qui conduit à adopter une mesure analogue en thermodynamique (l'entropie de l'union de deux systèmes indépendants est la somme des entropies de ces systèmes pris isolément).

4.6 ENTROPIE RELATIVE

Avant de terminer ce chapitre introductif, nous allons discuter d'une autre mesure d'entropie qui est utilisée fréquemment et dont certaines propriétés seront très utiles dans la suite. Il s'agit de *l'entropie relative* de deux lois de probabilités. Nous en profitons pour introduire quelques notations que nous utiliserons aussi dans la suite.

Définition de l'entropie relative. L'entropie *relative* ou *divergence de Kullback-Leibler* d'une loi de probabilité P par rapport à une loi Q, P et Q étant définies sur un même univers discret Ω, est définie par

Divergence (distance K-L)

$$D(P||Q) \triangleq \sum_{\omega \in \Omega} P(\omega) \log \frac{P(\omega)}{Q(\omega)}, \tag{4.119}$$

où on impose (par continuité) $0 \log \frac{0}{Q} = 0, \forall Q$ et $P \log \frac{P}{0} = +\infty, \forall P > 0$.

Propriétés. L'entropie relative est positive; elle est nulle si seulement si $\forall \omega \in \Omega \; : P(\omega) = Q(\omega)$. Il s'agit d'une conséquence immédiate de l'inégalité de Gibbs (4.50).

Bien que cette mesure ne soit pas symétrique elle est souvent interprétée comme une mesure de distance entre lois de probabilité, et on l'appelle communément *distance KL.*

On montre que $D(P||Q)$ est convexe vis-à-vis de la paire $(P(\omega), Q(\omega))$: si $(P_1(\omega), Q_1(\omega))$ et $(P_2(\omega), Q_2(\omega))$ sont deux paires de lois sur Ω, et si $P(\omega) \stackrel{\triangle}{=} \lambda P_1(\omega) + (1-\lambda)P_2(\omega)$ et $Q(\omega) \stackrel{\triangle}{=} \lambda Q_1(\omega) + (1-\lambda)Q_2(\omega)$, alors

$$D(P||Q) \leq \lambda D(P_1||Q_1) + (1-\lambda)D(P_2||Q_2), \tag{4.120}$$

$\forall \lambda \in [0,1]$.

On déduit aisément de ces propriétés quelques-unes des propriétés déjà démontrées pour les entropies et informations mutuelles.

En particulier, on a

$$I(\mathcal{X};\mathcal{Y}) = D(P(\mathcal{X},\mathcal{Y})||P(\mathcal{X})P(\mathcal{Y})), \tag{4.121}$$

et par conséquent on retrouve $I(\mathcal{X};\mathcal{Y}) \geq 0$.

De même, on a

$$H(\mathcal{X}) = \log|\mathcal{X}| - D(P(\mathcal{X})||\mathcal{U}), \tag{4.122}$$

où $|\mathcal{X}|$ représente le nombre de valeurs distinctes prises par la v.a. $\mathcal{X}$, et $\mathcal{U}$ désigne la distribution uniforme sur l'ensemble des valeurs possibles de la v.a. $\mathcal{X}$. On retrouve donc que $H(\mathcal{X})$ est maximale et vaut $\log|\mathcal{X}|$ pour une distribution uniforme, et on peut dire que l'écart de $H(\mathcal{X})$ par rapport à ce maximum mesure la distance de la loi de probabilité à la loi uniforme.

4.7 GENERALISATION AU CAS CONTINU

Les notions introduites ci avant l'ont été dans le cas simple de variables aléatoires définies sur un univers Ω fini (cas d'un alphabet fini). Lorsque le nombre de valeurs possibles devient infini dénombrable on peut en principe étendre les définitions avec un minimum de précautions (il faut s'assurer que les sommes infinies correspondent à des séries sommables, ce qui n'est pas nécessairement le cas).

Au-delà de ce cas assez académique, nous nous intéresserons dans la suite au cas de variables aléatoires à valeurs réelles et dont les distributions de probabilité sont continues. Dans ce cas, il n'est pas possible d'extrapoler directement les formules des entropies, celles-ci tendant vers l'infini.

4.7.1 Entropie relative

La distance de Kullback-Leibler se généralise au cas de v.a. continues de la manière suivante. Si $\mathcal{X}$ et $\mathcal{Y}$ sont deux v.a. continues sur $\mathbb{R}^n$, de densités $p_{\mathcal{X}}$ et $p_{\mathcal{Y}}$ telles que

$$p_{\mathcal{Y}}(x) = 0 \stackrel{p.p.}{\Rightarrow} p_{\mathcal{X}}(x) = 0$$

(on dit que $p_{\mathcal{Y}}$ est absolument continue par rapport à $p_{\mathcal{X}}$) l'entropie relative est définie par

$$\boxed{D(p_{\mathcal{X}}||p_{\mathcal{Y}}) \stackrel{\triangle}{=} \int \cdots \int p_{\mathcal{X}}(x_1,\ldots,x_n) \log \frac{p_{\mathcal{X}}(x_1,\ldots,x_n)}{p_{\mathcal{Y}}(x_1,\ldots,x_n)} dx_1 \ldots dx_n.} \tag{4.123}$$

4.7.2 Quantité d'information mutuelle

Comme la quantité d'information mutuelle est une entropie relative elle se généralise directement au cas continu.

En effet, en supposant que les variables aléatoires $\mathcal{X}$ et $\mathcal{Y}$ soient à valeurs réelles et admettent des *distributions* de probabilités continues de densité conjointe $p_{\mathcal{X},\mathcal{Y}}(\cdot,\cdot)$ et marginales $p_{\mathcal{X}}(\cdot)$ et $p_{\mathcal{Y}}(\cdot)$, on peut "passer à la limite" à partir du cas discret :

$$P(X \in [x, x+dx[) = p_{\mathcal{X}}(x)dx, P(Y \in [y, y+dx[) = p_{\mathcal{Y}}(y)dy$$

et

$$P(X \in [x, x+dx[, Y \in [y, y+dx[) = p_{\mathcal{X},\mathcal{Y}}(x,y)dxdy$$

et

$$I(\mathcal{X};\mathcal{Y}) = \int\int p_{\mathcal{X},\mathcal{Y}}(x,y) \log \frac{p_{\mathcal{X},\mathcal{Y}}(x,y)}{p_{\mathcal{X}}(x)p_{\mathcal{Y}}(y)} dxdy \tag{4.124}$$

et on retrouve le fait que sous l'hypothèse d'indépendance $I(\mathcal{X};\mathcal{Y}) = 0$.

Remarquons que, comme dans le cas discret, l'information mutuelle entre deux v.a. continues est égale à la distance KL de la distribution conjointe au produit des distributions marginales. Elle est positive en général et elle est nulle lorsque les deux v.a. sont indépendantes.

4.7.3 Entropie différentielle

Si nous effectuons le même passage à la limite dans la définition de l'entropie, nous voyons qu'elle tend vers l'infini, les événements élémentaires devenant de moins en moins probables au fur et à mesure que $\Delta x \to 0$.

Par conséquent, pour généraliser les entropies au cas continu on a recours à la notion *d'entropie différentielle* définie comme suit

$$H_d(\mathcal{X}) \stackrel{\triangle}{=} -\int \log\left[p_{\mathcal{X}}(x)\right] p_{\mathcal{X}}(x)dx, \tag{4.125}$$

pour autant que cette intégrale soit définie (et ce n'est pas nécessairement le cas).

On définit alors de manière similaire

$$H_d(\mathcal{X},\mathcal{Y}) = -\int\int \log\left[p_{\mathcal{X},\mathcal{Y}}(x,y)\right] p_{\mathcal{X},\mathcal{Y}}(x,y)dxdy, \tag{4.126}$$

$$H_d(\mathcal{X}|\mathcal{Y}) = -\int\int \log\left[p_{\mathcal{X}|\mathcal{Y}}(x|y)\right] p_{\mathcal{X},\mathcal{Y}}(x,y)dxdy. \tag{4.127}$$

Ces entropies différentielles héritent de la plupart des propriétés de leurs homologues discrètes, en particulier

$$I(\mathcal{X};\mathcal{Y}) = H_d(\mathcal{X}) + H_d(\mathcal{Y}) - H_d(\mathcal{X},\mathcal{Y}) \tag{4.128}$$

et

$$I(\mathcal{X};\mathcal{Y}) = H_d(\mathcal{X}) - H_d(\mathcal{X}|\mathcal{Y}) = H_d(\mathcal{Y}) - H_d(\mathcal{Y}|\mathcal{X}). \tag{4.129}$$

Exemples. Nous allons donner quelques exemples d'entropies différentielles et de quantités d'information. Nous laissons le soin au lecteur de faire les calculs, mais nous insistons sur quelques constatations immédiates que l'on peut faire. Avant d'essayer de comprendre ce qui suit, nous recommandons au lecteur de lire la partie de l'appendice B relatif aux variables aléatoires continues et en particulier la partie qui traite des vecteurs aléatoires Gaussiens.

1. L'entropie différentielle est invariante par rapport à une translation : si $\mathcal{Y} = a + \mathcal{X}$ alors $H_d(\mathcal{Y}) = H_d(\mathcal{X})$.

2. L'entropie différentielle est sensible à une dilatation de l'espace : si $\boldsymbol{\mathcal{Y}} = \boldsymbol{A}\boldsymbol{\mathcal{X}}$ (où $\boldsymbol{A}$ est une matrice $n \times n$ non-singulière, on a $H_d(\boldsymbol{\mathcal{Y}}) = \log|\boldsymbol{A}| + H_d(\boldsymbol{\mathcal{X}})$, où $|\boldsymbol{A}|$ désigne la valeur absolue du déterminant de la matrice $\boldsymbol{A}$.

3. L'entropie différentielle d'une v.a. uniforme sur l'intervalle $[0, a]$ vaut

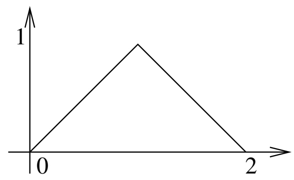


Figure 4.8. Distribution de la somme de v.a. indépendantes $\mathcal{U}_{[0,1]}$

Entropie d'une v.a. uniforme sur $[0, a]$

$$H_d(\mathcal{X}) = -\int_0^a \frac{1}{a} \log \frac{1}{a} dx = \log a. \tag{4.130}$$

On remarque en particulier que cette entropie est négative si $a < 1$. On retrouve le fait que si on multiple la variable aléatoire par une constante $b \neq 0$, la densité de probabilité reste uniforme mais vaut $\frac{1}{a|b|}$, par conséquent l'entropie différentielle est "augmentée" par un terme $\log |b|$.

4. Prenons deux v.a. indépendantes, $\mathcal{X}$ et $\mathcal{Y}$ distribuées uniformément sur un intervalle $[0, 1]$ et $\mathcal{Z} = \mathcal{X} + \mathcal{Y}$. On a $H_d(\mathcal{X}) = 0$ et $H_d(\mathcal{Y}) = 0$. D'autre part, calculons l'information mutuelle entre $\mathcal{Z}$ et disons $\mathcal{X}$:

$$I(\mathcal{X}; \mathcal{Z}) = H_d(\mathcal{Z}) - H_d(\mathcal{Z}|\mathcal{X}).$$

Or $H_d(\mathcal{Z}|\mathcal{X}) = 0$, puisque $\forall X$ la distribution conditionnelle de $\mathcal{Z}$ est une loi uniforme sur l'intervalle $[X, 1+X]$. D'autre part, $\mathcal{Z}$ suit une loi triangulaire telle que réprésentée à la figure 4.8. L'entropie de cette distribution vaut

$$H_d(\mathcal{Z}) = -\frac{2}{\ln 2} \int_0^1 x \ln x dx = \frac{1}{2 \ln 2},$$

et on trouve donc que

$$I(\mathcal{X}; \mathcal{Z}) = \frac{1}{2 \ln 2}.$$

5. L'entropie d'une v.a. Gaussienne $\mathcal{X} \sim \mathcal{N}(\mu, \sigma^2)$ vaut

Entropie d'une v.a. Gaussienne $\mathcal{N}(\mu, \sigma^2)$

$$H_d(\mathcal{X}) = \frac{1}{2} \log \left(2\pi e \sigma^2\right). \tag{4.131}$$

6. Considérons alors deux variables aléatoires conjointement Gaussiennes, et calculons leur information mutuelle. Nous pouvons supposer que les variables sont centrées et de matrice de variance-covariance

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}. \tag{4.132}$$

On a

$$H_d(\mathcal{X}_1) = \frac{1}{2} \log \left(2\pi e \sigma_1^2\right)$$

et

$$H_d(\mathcal{X}_1|\mathcal{X}_2) = \frac{1}{2} \log \left(2\pi e \sigma_1^2 (1 - \rho^2)\right),$$

et donc la quantité d'information mutuelle vaut

$$I(\mathcal{X}_1; \mathcal{X}_2) = H_d(\mathcal{X}_1) - H_d(\mathcal{X}_1|\mathcal{X}_2) = \frac{1}{2} \log \frac{1}{1 - \rho^2}. \tag{4.133}$$

On trouve, logiquement que la quantité d'information est nulle lorsque les deux v.a. sont indépendantes ($\rho = 0$) et qu'elle tend vers l'infini si elles sont linéairement dépendantes ($\rho = 1$). On constate également que l'information mutuelle est invariante si on modifie les échelles (par exemple, si on multiplie $\mathcal{X}_i$ par un nombre quelconque) puisqu'elle ne dépend que du nombre ρ qui est adimensionnel.

Exemple pratique.

Supposons que $\mathcal{X}$ soit une variable aléatoire Gaussienne de moyenne nulle et d'écart-type $\sqrt{P}$ qui représente un signal (nous dirons que P est la puissance moyenne du signal), et supposons que ce signal soit perturbé par un bruit indépendant $\mathcal{Z}$, lui aussi de nature Gaussienne et de puissance N (écart-type $\sqrt{N}$), donnant lieu au signal corrompu $\mathcal{Y} = \mathcal{X} + \mathcal{Z}$.

On peut calculer la quantité d'information mutuelle entre $\mathcal{X}$ et $\mathcal{Y}$ en calculant leur coefficient de corrélation. On a

$$\rho_{\mathcal{X},\mathcal{Y}} = \frac{\mathrm{cov}(\mathcal{X},\mathcal{Y})}{\sigma_{\mathcal{X}}\sigma_{\mathcal{Y}}}, \tag{4.134}$$

où $\mathrm{cov}(\mathcal{X},\mathcal{Y}) = E\{\mathcal{X}\mathcal{Y}\} = E\{\mathcal{X}(\mathcal{Z}+\mathcal{X})\} = P$, puisque les variables sont centrées et que $\mathcal{X} \perp \mathcal{Z}$. On a aussi $\sigma_{\mathcal{Y}} = \sqrt{P+N}$ et on en déduit finalement que

$$\rho = \sqrt{\frac{P}{P+N}}, \tag{4.135}$$

et donc que

$$\boxed{I(\mathcal{X};\mathcal{Y}) = \frac{1}{2}\log\left(1+\frac{P}{N}\right).} \tag{4.136}$$

Nous analyserons cet exemple plus en détail au chapitre 10 où nous verrons qu'il s'agit d'un modèle fréquemment utilisé dans le domaine des communications sur canaux continus.

Discussion. Les quelques exemples ci-dessus montrent clairement que les entropies différentielles peuvent être négatives et qu'elles ne sont pas invariantes lorsqu'on effectue une transformation non-singulière de la variable aléatoire. Par contre, on peut se convaincre que l'information mutuelle est invariante pour une telle transformation (ceci est illustré par l'exemple 6). On peut donc résumer la situation en disant que l'information mutuelle supporte le passage au cas continu.

Interprétation de l'entropie différentielle : discrétisation. Supposons que l'entropie différentielle de la v.a. $\mathcal{X}$ existe et discrétisons la variable $\mathcal{X}$ à pas constants Δ (voir figure 4.9). Si la densité de probabilité est continue dans chaque petit intervalle, on peut y écrire (théorème des valeurs intermédiaires) que

$$p_{\mathcal{X}}(x_i)\Delta = \int_{i\Delta}^{(i+1)\Delta} p_{\mathcal{X}}(x)dx,$$

en choisissant bien $x_i \in [i\Delta; (i+1)\Delta]$.

Par ailleurs, la valeur $p_{\mathcal{X}}(x_i)\Delta$ est la probabilité p_i de la $i-$ème valeur de la v.a. discrétisée $\mathcal{X}^{\Delta}$ avec le pas Δ. On a donc

$$H(\mathcal{X}^{\Delta}) = -\sum_{-\infty}^{+\infty} p_i \log p_i = -\sum_{-\infty}^{+\infty} p_{\mathcal{X}}(x_i)\Delta \log p_{\mathcal{X}}(x_i)\Delta,$$

c'est-à-dire

$$H(\mathcal{X}^{\Delta}) = -\sum_{-\infty}^{+\infty} p_{\mathcal{X}}(x_i)\Delta \log p_{\mathcal{X}}(x_i) - \log\Delta.$$

Si maintenant nous faisons tendre Δ vers 0, et en supposant que la limite du premier terme existe, il converge vers l'entropie différentielle. Par conséquent, on peut interpréter l'entropie différentielle de la manière suivante : l'entropie d'une quantification en n bits d'une variable aléatoire continue est approximativement égale à $H_d(\mathcal{X}) + n$ (on suppose que $\Delta = 2^{-n}$, et que l'intervalle de variation de x est fini.).

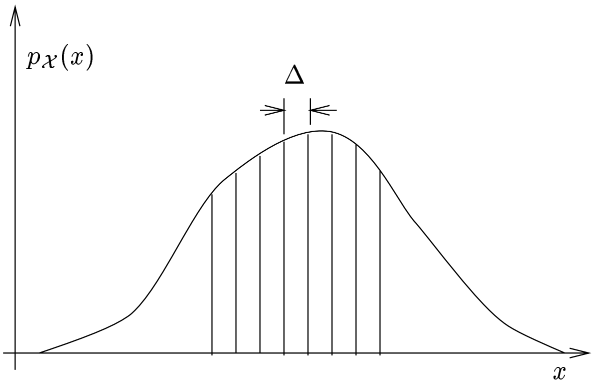


Figure 4.9. Quantification d'une v.a. continue

4.8 RESUME

Dans ce chapitre nous avons introduit les principales notions nouvelles en théorie de l'information, qui nous seront utiles dans la suite.

Premièrement nous avons défini, de façon plausible, mais sans autre forme de justification, la mesure d'incertitude et d'information logarithmique. Comme nous le verrons plus loin dans ce cours, cette mesure se justifie par le fait qu'elle permet en effet de quantifier les ressources physiques nécessaires à la conception de systèmes de traitement de l'information. Du point de vue mathématique, c'est l'additivité de cette mesure qui rend son exploitation aussi simple que possible du point de vue mathématique. Cette propriété élémentaire rend également cette mesure intuitivement acceptable comme mesure de l'incertitude.

Ces notions ont été introduites tout d'abord dans le cas discret, et nous avons vu que la plupart de leurs propriétés résultaient de quelques propriétés simples de la fonction logarithme, et en particulier de la concavité de celle-ci et de l'additivité vis-à-vis de la multiplication des probabilités. Enfin, nous avons généralisé les notions au cas continu, en passant par la notion d'entropie relative (distance KL) qui partage une bonne partie des propriétés de l'entropie de Shannon, avec en plus la possibilité d'être généralisée facilement au cas continu et/ou mixte. L'entropie différentielle a ensuite été introduite, de façon à ce que les différences d'entropies différentielles puissent toujours être interprétées comme des quantités d'information.

Nous avons terminé ce chapitre par un certain nombre d'exemples de calculs d'entropies et de quantités d'information dans le cas continu.

A la fin de ce chapitre nous vous proposons un certain nombre d'exercices ainsi qu'un formulaire qui collationne la plupart des formules utiles pour la suite.

Appendice 4.A: Exercices

Notions d'entropie et d'information

1. Soit la table de contingences

	Y_1	Y_2
X_1	$\frac{1}{3}$	$\frac{1}{3}$
X_2	0	$\frac{1}{3}$

   Calculer (logarithmes en base 2) :

   (a) $H(\mathcal{X}), H(\mathcal{Y})$
   (b) $H(\mathcal{X}|\mathcal{Y}), H(\mathcal{Y}|\mathcal{X})$
   (c) $H(\mathcal{X},\mathcal{Y})$
   (d) $H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X})$
   (e) $I(\mathcal{X};\mathcal{Y})$
   (f) dessiner un diagramme de Venne qui résume la situation.

2. Un match de Tennis entre deux équipes consiste en une série de 5 sets au maximum qui se termine dès que l'une des équipes a remporté trois sets. Soient a et b les deux équipes et soit $\mathcal{X}$ une variable aléatoire qui représente l'issue d'un match entre a et b. Par exemple, $X = aaa, babab, bbaaa$ sont des valeurs possibles de $\mathcal{X}$ (il y en a d'autres). Soit $\mathcal{Y}$ la variable aléatoire qui désigne le nombre de sets effectivement joués avec $\mathcal{Y} = \{3, 4, 5\}$.

   Représenter les parties au moyen d'un arbre dont les branches successives sont les issues des sets joués et les noeuds terminaux sont décorés au moyen de deux variables qui désignent le nombre de sets et l'issue du match.

   En supposant que les deux équipes sont de même niveau et que les sets sont indépendants, calculer $H(\mathcal{X})$, $H(\mathcal{Y})$, $H(\mathcal{X}|\mathcal{Y})$ et $H(\mathcal{Y}|\mathcal{X})$. Quelles probabilités faut-il associer aux noeuds terminaux de l'arbre ?

   Soit $\mathcal{Z} = \{a, b\}$ la variable aléatoire qui désigne l'équipe qui remporte le match. Trouver $H(\mathcal{X}|\mathcal{Z})$, comparer à $H(\mathcal{X})$ et justifier ce qu'on trouve. Trouver $H(\mathcal{Z}|\mathcal{X})$, et justifier.

3. Montrer que

   (a) $H(\mathcal{X},\mathcal{Y}) = H(\mathcal{Y}) + H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{X}) + H(\mathcal{Y}|\mathcal{X})$
   (b) $H(\mathcal{X},\mathcal{Y}) \geq \max\{H(\mathcal{X}), H(\mathcal{Y})\}$
   (c) $H(\mathcal{X}|\mathcal{Y}) \leq H(\mathcal{X})$
   (d) $H(\mathcal{X},\mathcal{Y}) \leq H(\mathcal{X}) + H(\mathcal{Y})$
   (e) $I(\mathcal{X};\mathcal{Y}) = H(\mathcal{X}) + H(\mathcal{Y}) - H(\mathcal{X},\mathcal{Y}) = H(\mathcal{X}) - H(\mathcal{X}|\mathcal{Y})$

4. Soit $\Delta(\mathcal{X},\mathcal{Y}) \stackrel{\triangle}{=} H(\mathcal{X},\mathcal{Y}) - I(\mathcal{X};\mathcal{Y})$, où $\mathcal{X}$ et $\mathcal{Y}$ sont des v.a. définies sur un $(\Omega, P, \mathcal{E})$ discret.

   Montrer que

   (a) $\Delta(\mathcal{X},\mathcal{Y}) = H(\mathcal{X}|\mathcal{Y}) + H(\mathcal{Y}|\mathcal{X})$
   (b) $\Delta(\mathcal{X},\mathcal{Y}) \geq 0$,
   (c) $\Delta(\mathcal{X},\mathcal{X}) = 0$,
   (d) $\Delta(\mathcal{X},\mathcal{Y}) = \Delta(\mathcal{Y},\mathcal{X})$
   (e) $\Delta(\mathcal{X},\mathcal{Z}) \leq \Delta(\mathcal{X},\mathcal{Y}) + \Delta(\mathcal{Y},\mathcal{Z})$ (plus difficile; partir de (a)).

   Est-ce que $\Delta(\mathcal{X},\mathcal{Y})$ définit une distance sur l'ensemble des v.a. définies sur $(\Omega, P, \mathcal{E})$? Pourquoi ?

5. Soient $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ trois variables aléatoires binaires. On donne les informations suivantes :

   - $P(\mathcal{X} = 0) = P(\mathcal{Y} = 0) = 0.5$,

- $P(\mathcal{X},\mathcal{Y}) = P(\mathcal{X})P(\mathcal{Y})$
- $\mathcal{Z} = (\mathcal{X}+\mathcal{Y})$mod2 (i.e. $\mathcal{Z} = 1 \Leftrightarrow \mathcal{X} \neq \mathcal{Y}$).

(a) Imaginez une expérience physique qui correspondrait à ces hypothèses.
(b) Que vaut $P(\mathcal{Z} = 0)$?
(c) Que valent $H(\mathcal{X}), H(\mathcal{Y}), H(\mathcal{Z})$?
(d) Que valent $H(\mathcal{X},\mathcal{Y}), H(\mathcal{X},\mathcal{Z}), H(\mathcal{Y},\mathcal{Z}), H(\mathcal{X},\mathcal{Y},\mathcal{Z})$?
(e) Que valent $I(\mathcal{X};\mathcal{Y}), I(\mathcal{X};\mathcal{Z}), I(\mathcal{Y};\mathcal{Z})$?
(f) Que valent $I(\mathcal{X};\mathcal{Y},\mathcal{Z}), I(\mathcal{Y};\mathcal{X},\mathcal{Z}), I(\mathcal{Z};\mathcal{X},\mathcal{Y})$?
(g) Que valent $I(\mathcal{X};\mathcal{Y}|\mathcal{Z}), I(\mathcal{Y};\mathcal{X}|\mathcal{Z}), I(\mathcal{Z};\mathcal{X}|\mathcal{Y})$?
(h) Dessiner un diagramme de Venne qui résume la situation.
(i) Lesquelles, parmi les relations d'indépendance suivantes

$$\mathcal{X} \perp \mathcal{Y}, \mathcal{X} \perp \mathcal{Z}, \mathcal{Y} \perp \mathcal{Z}, \mathcal{X} \perp \mathcal{Y}|\mathcal{Z}, \mathcal{Y} \perp \mathcal{Z}|\mathcal{X}$$

sont vérifiées dans ce problème ?
(j) En modifiant légèrement les données du problème, à savoir en supposant $P(\mathcal{Y} = 0) = p \neq 0.5$ et toujours $P(\mathcal{X} = 0) = 0.5$, lesquelles des relations d'indépendance précédentes restent vraies (respectivement, restent fausses) ?

6. Soient $\mathcal{X},\mathcal{Y},\mathcal{Z}$ trois variables aléatoires discrètes quelconques. Montrer que

(a) $H(\mathcal{X},\mathcal{Y}|\mathcal{Z}) \geq H(\mathcal{X}|\mathcal{Z})$;
(b) $I(\mathcal{X},\mathcal{Y};\mathcal{Z}) \geq I(\mathcal{X};\mathcal{Z})$;
(c) $H(\mathcal{X},\mathcal{Y},\mathcal{Z}) - H(\mathcal{X},\mathcal{Y}) \leq H(\mathcal{X},\mathcal{Z}) - H(\mathcal{X})$;
(d) $I(\mathcal{X};\mathcal{Z}|\mathcal{Y}) \geq I(\mathcal{Z};\mathcal{Y}|\mathcal{X}) - I(\mathcal{Z};\mathcal{Y}) + I(\mathcal{X};\mathcal{Z})$.

Applications illustratives

1. *Entropie d'un chien à la recherche d'un os.* Un chien se déplace le long d'une droite. Désignons par $\mathcal{X}_i$ la variable aléatoire (un entier) qui représente la position du chien à l'instant i. Le chien démarre sa recherche en $\mathcal{X}_0 = 0$ (la position initiale du chien), ensuite, à chaque pas de temps il fait un pas, soit à gauche ($\mathcal{X}_{i+1} = \mathcal{X}_i - 1$), soit à droite ($\mathcal{X}_{i+1} = \mathcal{X}_i + 1$). Le premier pas est +1 avec une probabilité de 0.5. Ensuite, à chaque pas de temps il choisit soit de poursuivre dans la direction courante (probabilité de 0.9) soit de changer de direction (probabilité de 0.1). Un trajet typique du chien pourrait être comme suit

$$(\mathcal{X}_0, \mathcal{X}_1, \ldots) = (0, +1, +2, +3, +2, +1, 0, -1, -2 \ldots)$$

(a) Pour, $i \geq 1$, exprimez $\mathcal{X}_{i+1}$ sous forme de deux fonctions, $f_+(\mathcal{X}_i, \mathcal{X}_{i-1})$ et $f_-(\mathcal{X}_i, \mathcal{X}_{i-1})$ correspondant respectivement au cas où le chien poursuit dans la direction courante et au cas où il change de direction.
(b) Que valent, pour $i \geq 1$, $H(\mathcal{X}_{i+1}|\mathcal{X}_{i-1}, \mathcal{X}_i)$ et $H(\mathcal{X}_{i+1}|\mathcal{X}_1\mathcal{X}_2, \ldots, \mathcal{X}_i)$.
(c) Calculer l'entropie $H(\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_n)$.
(d) Calculer l'entropie moyenne par pas de temps de la position du chien, c'est-à-dire

$$\lim_{n\to\infty} \frac{H(\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_n)}{n+1}.$$

(e) Désignons par S la variable aléatoire qui désigne l'instant où le chien change de direction pour la première fois. Quelle est la valeur moyenne de S (espérance mathématique) ? (Notez que le chien doit faire au moins un pas avant de pouvoir changer de direction).

2. *Entropie du temps au premier succès.* Une pièce équilibrée est lancée jusqu'à ce qu'on obtienne face pour la première fois. Soit $\mathcal{X}$ la variable aléatoire qui désigne le nombre de lancers ainsi obtenus ($\mathcal{X} \in \{1, 2, \ldots\}$).

(a) Trouver l'entropie $H(\mathcal{X})$.

(b) Supposons maintenant qu'on lance la pièce jusqu'à obtenir une deuxième fois face, et désignons par $\mathcal{X}'$ la variable aléatoire entière qui compte le nombre de lancers supplémentaires et $\mathcal{Y}$ la variable aléatoire qui désigne le nombre total de lancers. Montrer que $H(\mathcal{Y}) < H(\mathcal{X}, \mathcal{X}')$ et que $H(\mathcal{X}, \mathcal{X}') = 2H(\mathcal{X})$.

(c) Pouvez-vous décrire une séquence "efficace" de questions dichotomiques du type "Est-ce que $\mathcal{X} \in \mathcal{S}_i$?" pour déterminer (deviner) la valeur de $\mathcal{X}$, c'est-à-dire telle que le nombre moyen de questions vaille $H(\mathcal{X})$?

3. *Jeu de cartes.* Un jeu de cartes qui comprend 26 cartes rouges et 26 cartes noires est mélangé puis retourné une carte après l'autre. Soit $\mathcal{X}_k$ la v.a. qui désigne la couleur R ou N de la i-ème carte qui est ainsi retournée $(k = 1, \ldots, 52)$

   (a) Déterminez $P(\mathcal{X}_i = R)$, $H(\mathcal{X}_1)$ et $H(\mathcal{X}_{52})$.

   (b) Trouvez la relation entre les lois $P(\mathcal{X}_1, \ldots, \mathcal{X}_k)$ et $P(\mathcal{X}_{1+i}, \ldots, \mathcal{X}_{k+i})$, avec $i, k \geq 1$ et $i + k \leq 52$.

   (c) Déduisez la relation entre $P(\mathcal{X}_k | \mathcal{X}_1 \ldots, \mathcal{X}_{k-1})$ et $P(\mathcal{X}_{k+1} | \mathcal{X}_2 \ldots, \mathcal{X}_k)$.

   (d) Est-ce que $H(\mathcal{X}_k | \mathcal{X}_1, \ldots, \mathcal{X}_{k-1})$ croît ou decroît, lorsque k augmente ? (preuve à l'appui...)

   (e) Calculez $H(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_{52})$.

Appendice 4.B: Formules utiles

Entropie, information, divergence

$$H(\mathcal{X}) \triangleq -\sum_{i=1}^{n} P(X_i) \log P(X_i) \tag{F.1}$$

$$H(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_m) \triangleq -\sum_{i_1=1}^{n_1} \cdots \sum_{i_m=1}^{n_m} P(X_{1,i_1}, \ldots, X_{m,i_m}) \log P(X_{1,i_1}, \ldots, X_{m,i_m}) \tag{F.2}$$

$$H(\mathcal{X}|Y_j) = -\sum_{i=1}^{n} P(X_i|Y_j) \log P(X_i|Y_j) \tag{F.3}$$

$$H(\mathcal{X}|\mathcal{Y}) \triangleq -\sum_{i=1}^{n} \sum_{j=1}^{m} P(X_i, Y_j) \log P(X_i|Y_j) \tag{F.4}$$

$$H(\mathcal{X}|\mathcal{Y}) = \sum_{j=1}^{m} P(Y_j) H(\mathcal{X}|Y_j) \tag{F.5}$$

$$I(\mathcal{X};\mathcal{Y}) \triangleq \sum_{i=1}^{n} \sum_{j=1}^{m} P(X_i, Y_j) \log \frac{P(X_i, Y_j)}{P(X_i)P(Y_j)} \tag{F.6}$$

$$I(\mathcal{X};\mathcal{Y}) = H(\mathcal{X}) - H(\mathcal{X}|\mathcal{Y}) \tag{F.7}$$

$$= H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X}) = H(\mathcal{X}) + H(\mathcal{Y}) - H(\mathcal{X},\mathcal{Y}) \tag{F.8}$$

$$I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) \triangleq H(\mathcal{X}|\mathcal{Z}) - H(\mathcal{X}|\mathcal{Y},\mathcal{Z}) \tag{F.9}$$

$$I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) = \sum_{i,j,k} P(X_i, Y_j, Z_k) \log \frac{P(X_i, Y_j|Z_k)}{P(X_i|Z_k)P(Y_j|Z_k)} \tag{F.10}$$

$$D(P||Q) \triangleq \sum_{\omega \in \Omega} P(\omega) \log \frac{P(\omega)}{Q(\omega)} \tag{F.11}$$

Additivité et chaînage

$$H(\mathcal{X},\mathcal{Y}) = H(\mathcal{X}) + H(\mathcal{Y}|\mathcal{X}) = H(\mathcal{Y}) + H(\mathcal{X}|\mathcal{Y}) \tag{F.12}$$

$$H(\mathcal{X},\mathcal{Y}|\mathcal{Z}) = H(\mathcal{X}|\mathcal{Z}) + H(\mathcal{Y}|\mathcal{X},\mathcal{Z}) \tag{F.13}$$

$$\begin{aligned} H(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n) &= H(\mathcal{X}_1) + H(\mathcal{X}_2|\mathcal{X}_1) + H(\mathcal{X}_3|\mathcal{X}_2, \mathcal{X}_1) + \ldots + H(\mathcal{X}_n|\mathcal{X}_{n-1}, \ldots, \mathcal{X}_1) \\ &\triangleq \sum_{i=1}^{n} H(\mathcal{X}_i|\mathcal{X}_{i-1}, \ldots, \mathcal{X}_1) \end{aligned} \tag{F.14}$$

$$\begin{aligned} H(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n|\mathcal{Y}) &= H(\mathcal{X}_1|\mathcal{Y}) + H(\mathcal{X}_2|\mathcal{X}_1, \mathcal{Y}) + \ldots + H(\mathcal{X}_n|\mathcal{X}_{n-1}, \ldots, \mathcal{X}_1, \mathcal{Y}) \\ &\triangleq \sum_{i=1}^{n} H(\mathcal{X}_i|\mathcal{X}_{i-1}, \ldots, \mathcal{X}_1, \mathcal{Y}) \end{aligned} \tag{F.15}$$

$$I(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n; \mathcal{Y}) = \sum_{i=1}^{n} I(\mathcal{X}_i; \mathcal{Y}|\mathcal{X}_{i-1}, \ldots, \mathcal{X}_1) \tag{F.16}$$

Positivité, convexité et monotonicité

$$H(\mathcal{X}) \geq 0 \quad \text{et} \quad H(\mathcal{X},\mathcal{Y}) \geq 0 \quad \text{et} \quad H(\mathcal{X}|\mathcal{Y}) \geq 0 \tag{F.17}$$

$$H(\mathcal{X},\mathcal{Y}) \leq H(\mathcal{X}) + H(\mathcal{Y}) \leq 2H(\mathcal{X},\mathcal{Y}) \tag{F.18}$$

$$0 \leq H(\mathcal{X}) \leq H(\mathcal{X},\mathcal{Y}) \leq H(\mathcal{X},\mathcal{Y},\mathcal{Z}) \leq \ldots \tag{F.19}$$

$$H(\mathcal{X}) \geq H(\mathcal{X}|\mathcal{Y}) \geq H(\mathcal{X}|\mathcal{Y},\mathcal{Z}) \geq \ldots \geq 0 \tag{F.20}$$

$$I(\mathcal{X};\mathcal{Y}) \geq 0 \quad \text{et} \quad I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) \geq 0 \tag{F.21}$$

$$0 \leq I(\mathcal{X};\mathcal{Y}) \leq I(\mathcal{X};\mathcal{Y},\mathcal{Z}) \leq I(\mathcal{X},\mathcal{T};\mathcal{Y},\mathcal{Z}) \tag{F.22}$$

$$D(P||Q) \geq 0 \tag{F.23}$$

Non-création d'information

Si $\mathcal{X} \leftrightarrow \mathcal{Y} \leftrightarrow \mathcal{Z}$ forment une chaîne de Markov alors

$$I(\mathcal{X};\mathcal{Y}) \geq I(\mathcal{X};\mathcal{Z}) \quad \textit{et} \quad I(\mathcal{Y};\mathcal{Z}) \geq I(\mathcal{X};\mathcal{Z}) \tag{F.24}$$

En particulier $\mathcal{Z} = g(\mathcal{Y})$

$$I(\mathcal{X};\mathcal{Y}) \geq I(\mathcal{X};g(\mathcal{Y})) \tag{F.25}$$

5 MODELES GRAPHIQUES ET INFÉRENCE PROBABILISTE

5.1 INTRODUCTION

Dans ce chapitre nous allons décrire quelques outils de modélisation permettant d'appliquer les notions que nous venons d'introduire à des problèmes pratiques. Les différents modèles que nous introduisons ont la caractéristique commune de reposer sur une repésentation graphique. Nous nous bornons ici à la représentation de lois de probabilités discrètes, mais nous signalons au lecteur qu'il existe bien entendu de nombreux modèles graphiques pour la représentation de densités de probabilités de variables continues.

Nous commençons par les réseaux Bayesiens qui constituent à l'heure actuelle un des outils principaux de modélisation et de raisonnement probabliste utilisés notamment en intelligence artificielle. Les réseaux Bayesiens se focalisent sur la modélisation et l'exploitation de la notion d'indépendance conditionnelle.

Ensuite nous introduisons les modèles de type chaîne de Markov, qui sont en réalité un type particulier hautement structuré de réseau Bayesien. Les chaînes de Markov permettent notamment la modélisation de processus aléatoires en temps discret, et interviennent dans de nombreuses questions relatives à la modélisation du langage parlé et écrit (reconnaissance de la parole, compression de textes). Ce type de modèle sera intensivement utilisé dans ce cours pour la modélisation de sources et de canaux. Ajoutons que ce type de structure peut se généraliser à plusieurs dimensions sous la forme de réseaux de Markov, utilisés notamment en traitement d'images.

Enfin nous terminerons par la description d'un modèle simple permettant de représenter efficacement des lois de probabilités conditionnelles, sous la forme d'un arbre de décision. Ce type de modèle est utilisé dans de nombreux systèmes d'aide à la prise de décision, mais également dans le cadre du codage de données.

5.2 RESEAUX BAYESIENS

Dans les sections qui précèdent nous avons vu le rôle important joué par la notion d'indépendance et certaines subtilités autour de la notion d'indépendance conditionnelle. Nous avons également vu que certaines hypothèses structurantes du point de vue indépendance conditionnelle, telle que la propriété de Markov, peuvent être représentées graphiquement. Les réseaux Bayesiens constituent un outil très puissant pour la représentation et la manipulation de la notion d'indépendance conditionnelle. Ils servent notamment dans le cadre de la représentation de connaissances probabilistes dans les systèmes à base de connaissances (en Intelligence Artificielle) et permettent de formuler des algorithmes d'inférence efficaces.

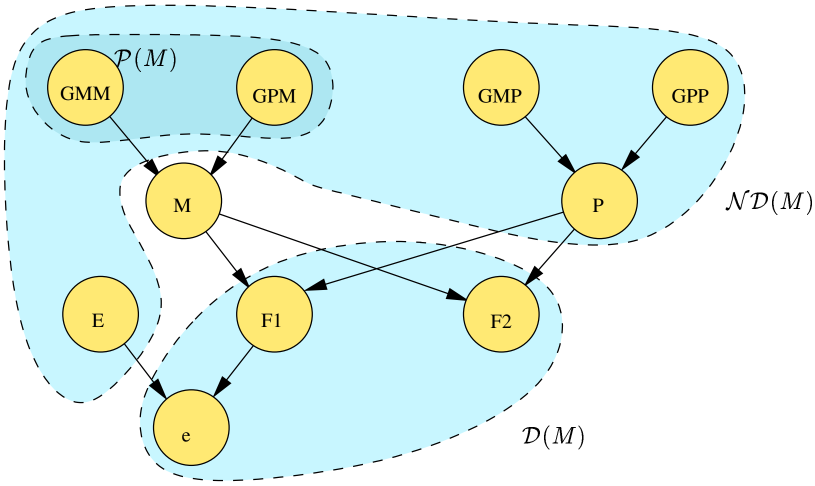


Figure 5.1. Réseau Bayesien

Nous ne pouvons pas dans le cadre de ce cours nous étendre sur la théorie de ces réseaux Bayesiens. Mais nous allons illustrer ici de quoi il s'agit en tirant profit des notions que nous venons d'introduire. Nous suggérons au lecteur intéressé de consulter les références [Fre99, Pea88] sur lesquelles nous avons basé ce texte.

Un réseau Bayesien est un graphe permettant de modéliser et de raisonner sur une loi de probabilité conjointe d'un certain nombre de variables aléatoires (nous considérons ici seulement le cas discret). Dans ce graphe, les noeuds correspondent aux variables aléatoires et les arcs, qui sont orientés, aux relations entre ces variables. Un arc $\mathcal{X} \to \mathcal{Y}$ signifie que la variable $\mathcal{X}$ influence (directement) la variable $\mathcal{Y}$. On dit aussi que $\mathcal{X}$ est une *cause directe* de $\mathcal{Y}$, et on appelle aussi ces graphes des graphes de causalités. Une restriction importante en ce qui concerne la topologie du graphe orienté est qu'il ne doit pas contenir de cycles orientés.

5.2.1 Exemple illustratif

La figure 5.1 montre un réseau Bayesien illustratif. Il s'agit d'un exemple de génétique : les différentes variables représentent la couleur des yeux de différentes personnes d'une même famille (P signifie père, M mère, GPP grand-père paternel, GMP grand-mère paternelle, GPM grand-père maternel, GMM grand-mère maternelle, F1 et F2 sont des fils, e enfant et E épouse.) Chacune des variables peut prendre comme valeurs $Bleu, Brun, Vert$ correspondant à la couleur des yeux de la personne désignée par la variable. Le graphe représente intuitivement les relations entre les couleurs des yeux des différentes personnes de la famille.

Remarque. Le réseau Bayesien de la figure 5.1 fournit une description intuitive, mais en réalité incorrecte de notre problème de génétique. Une description plus correcte du phénomène fait en effet appel à la distinction entre génotype (qui se transmet selon une structure similaire à notre graphe) et phénotype (qui correspond aux variables observables telles que la couleur des yeux). Nous reviendrons sur ce modèle plus correct plus loin. Pour le moment nous allons faire comme s'il n'y avait pas de différences entre les deux types de variables, afin de simplifier notre discussion qui a pour seul but d'introduire les idées générales relatives aux réseaux Bayesiens à l'aide d'une exemple concret et intuitif.

5.2.2 Terminologie

Nous rappelons ci-dessous la terminologie usuelle relative aux graphes orientés et introduisons les notations que nous utiliserons dans ce chapitre.

1. Les noeuds du graphe sont identifiés par les noms des v.a. qui leur correspondent (disons $\mathcal{X}_1$ à $\mathcal{X}_n$).

2. Les "pères d'un noeud $\mathcal{X}_k$" (notés $\mathcal{P}(\mathcal{X}_k)$) désignent l'ensemble des origines des arcs qui pointent vers le noeud $\mathcal{X}_k$.

3. Les “fils d’un noeud $\mathcal{X}_k$” (notés $\mathcal{F}(\mathcal{X}_k)$) désignent l’ensemble des noeuds extrémités des arcs qui émergent du noeud $\mathcal{X}_k$.

4. Les “ascendants d’un noeud $\mathcal{X}_k$” (notés $\mathcal{A}(\mathcal{X}_k)$) sont les pères de celui-ci et leurs ascendants.

5. Les “descendants d’un noeud $\mathcal{X}_k$” (notés $\mathcal{D}(\mathcal{X}_k)$) en sont les fils et les descendants de ceux-ci.

6. Nous désignons aussi par $\mathcal{G}$ l’ensemble des noeuds (et des v.a.) du graphe.

7. Les “non-descendants d’un noeud $\mathcal{X}_k$” (notés $\mathcal{ND}(\mathcal{X}_k)$) désigne alors l’ensemble $\mathcal{G} \cap \neg(\{\mathcal{X}_k\} \cup \mathcal{D}(\mathcal{X}_k))$, c’est-à-dire tous les noeuds du graphe sauf le noeud $\mathcal{X}_k$ et ses descendants.

8. Un chemin (non-orienté) dans le graphe est une suite de noeuds du graphe $\mathcal{X}_{i_1}, \mathcal{X}_{i_2}, \ldots, \mathcal{X}_{i_k}$ tels que $\forall j = 1, \ldots, k-1$, $\mathcal{X}_{i_j}$ est soit père, soit fils de $\mathcal{X}_{i_{j+1}}$.

9. Un chemin *orienté* est une suite de noeuds du graphe $\mathcal{X}_{i_1}, \mathcal{X}_{i_2}, \ldots, \mathcal{X}_{i_k}$ tels que $\forall j = 1, \ldots, k-1$, $\mathcal{X}_{i_j}$ est père de $\mathcal{X}_{i_{j+1}}$.

10. Un chemin (orienté ou non) est dit sans cycle si tous les noeuds qui en font partie y apparaissent une seule fois.

5.2.3 Modèle probabiliste et sémantique

Le modèle de réseau Bayesien est complété (on dit aussi *instantié*) par un ensemble de lois de probabilités conditionnelles de la manière suivante : pour chaque variable $\mathcal{X}$ on spécifie la loi $P(\mathcal{X}|\mathcal{P}(\mathcal{X}))$ où $\mathcal{P}(\mathcal{X})$ désigne l’ensemble des variables correspondant aux parents directs (pères) de $\mathcal{X}$ dans le graphe. Si $\mathcal{X}$ n’a pas de parents alors on donne simplement la loi a priori $P(\mathcal{X})$.

Par exemple, le réseau de la figure 5.1 est complété en donnant les probabilités a priori $P(GMM), P(GPM), P(GMP), P(GPP), P(E)$ et les lois conditionnelles $P(M|GMM, GPM), P(P|GMP, GPP), P(F1|M, P), P(F2|M, P)$ et $P(e|E, F1)$.

Une structure donnée d’un réseau Bayesien peut évidemment être complétée de multiples façons. Lorsqu’on étudie les propriétés de tels modèles il est donc fondamental de distinguer entre les propriétés *structurelles* qui seront vraies pour toutes les instantiations possibles, et celles dont la vérification dépend de l’instantiation particulière envisagée. Puisque les tables de probabilités conditionnelles sont difficiles à déterminer de façon très précise dans la plupart des situations pratiques, ce sont évidemment les propriétés structurelles qui nous intéressent le plus.

Du point de vue probabiliste, la sémantique associée aux réseaux Bayesiens est la suivante

Indépendance conditionnelle :
Pour toute variable $\mathcal{X}_k$ du graphe et quel que soit l’ensemble de variables $\mathcal{W} \subset \mathcal{ND}(\mathcal{X}_k)$ on a

$$P(\mathcal{X}_k|\mathcal{P}(\mathcal{X}_k), \mathcal{W}) = P(\mathcal{X}_k|\mathcal{P}(\mathcal{X}_k)). \tag{5.1}$$

Cela veut dire qui si les pères d’une variable sont donnés alors celle-ci devient indépendante de tout ensemble de non-descendants de cette variable (et en particulier de chacun de ses non-descendants).

Par exemple dans le graphe de la figure 5.1 les variables GMM, GPM, GMP et GPP sont idépendantes (ce qui est réaliste si nous excluons les mariages consanguins). De même, lorsque E et $F1$ sont connus, alors e est indépendant des autres variables du réseau. Le graphe exprime donc l’hypothèse que si nous connaissions la couleur des yeux des parents de e alors la connaissance de la couleur des yeux des autres membres de la famille (à l’exception des futurs enfants, petits enfants ... de e) ne nous apprendrait rien sur la couleur des yeux de e.

De la définition (5.1) découle le fait que la loi de probabilité conjointe des variables du graphe se factorise sous la forme du produit des lois conditionnelles associées aux différentes variables du graphe

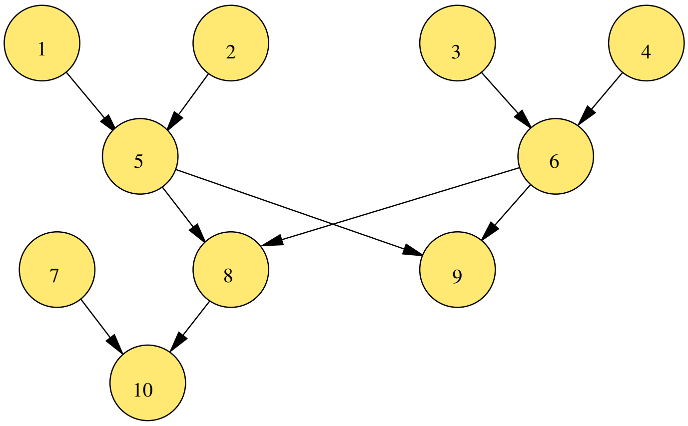


Figure 5.2. Réseau Bayesien numéroté

Distribution conjointe :

$$P(\mathcal{G}) = \prod_{\mathcal{X} \in \mathcal{G}} P(\mathcal{X}|\mathcal{P}(\mathcal{X})). \tag{5.2}$$

En effet, raisonnons sur notre exemple pour nous en convaincre. La distribution conjointe des variables du graphe $\mathcal{G}$ est déduite de (5.1) de la manière suivante :

1. Comme le graphe orienté est acyclique, il est possible de numéroter les noeuds de manière à ce que les pères de toute variable soient de numéro inférieur à cette variable.

   Par exemple la figure 5.2 fournit une numérotation des noeuds du graphe de la figure 5.1 qui respecte cette propriété. (Dans notre exemple il aurait suffit de trier les variables par ordre croissant de la date de naissance des personnes correspondant aux noeuds du graphe pour obtenir un ordre qui respecte la propriété, éventuellement différent de celui indiqué sur la figure.)

2. On applique ensuite la règle de chaînage des probabilités conjointes, en utilisant cet ordre de variables. Cela donne pour notre exemple

$$P(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_{10}) = P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1)P(\mathcal{X}_3|\mathcal{X}_1, \mathcal{X}_2) \cdots P(\mathcal{X}_{10}|\mathcal{X}_1, \cdots, \mathcal{X}_9). \tag{5.3}$$

3. Enfin, on remarque que pour chacun des facteurs dans la formule (5.3) le conditionnement se fait par rapport à un ensemble de non-descendants qui comprend l'ensemble des pères de la variable conditionnée. On peut donc exploiter la propriété de base (5.1) pour simplifier la formule en ne gardant dans les conditionnements que les pères. Dans le cas de notre exemple on obtient

$$\begin{aligned} P(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_{10}) \quad = \quad & P(\mathcal{X}_1)P(\mathcal{X}_2)P(\mathcal{X}_3)P(\mathcal{X}_4)P(\mathcal{X}_5|\mathcal{X}_1, \mathcal{X}_2)P(\mathcal{X}_6|\mathcal{X}_3, \mathcal{X}_4) \\ & P(\mathcal{X}_7)P(\mathcal{X}_8|\mathcal{X}_5, \mathcal{X}_6)P(\mathcal{X}_9|\mathcal{X}_5, \mathcal{X}_6)P(\mathcal{X}_{10}|\mathcal{X}_7, \mathcal{X}_8). \end{aligned}$$

On conclut bien que la distribution conjointe des variables du réseau Bayesien est le produit des distributions conditionnelles (ou a priori) associées à chacun des noeuds du graphe.

A partir de cette distribution conjointe on peut évidemment effectuer tous les raisonnements probabilistes souhaités, en utilisant les techniques habituelles de marginalisation et le théorème de Bayes. Cependant, lorsque le nombre de variables est important, ces opérations deviennent rapidement infaisables en pratique. En anticipant sur la suite, notons que les réseaux Bayesiens correspondant aux problèmes de codage/décodage de canal contiennent en pratique des dizaines de milliers de variables, et dans ce cas les opérations explicites sont parfaitement impossibles (il n'est même pas possible de représenter explicitement les tables de probabilités conjointes).

Il est donc capital d'exploiter la structure du réseau pour effectuer des raisonnements probabilistes. Des algorithmes généraux ont été développés dont nous discuterons ultérieurement des variantes simplifiées utiles dans le cadre de ce cours.

Marginalisation par rapport aux feuilles. A titre d'exemple d'une manipulation qui peut se faire très simplement en exploitant la structure du graphe, citons la marginalisation par rapport aux feuilles du graphe. On peut en effet se convaincre que la marginalisation sur une variable qui ne possède pas de descendants dans le graphe (une feuille) équivaut à enlever cette variable du réseau (ainsi que les arcs qui pointent vers cette variable). Plus généralement, la marginalisation par rapport à un ensemble de noeuds comprenant tous ses descendants équivaut à effacer cette partie du graphe.

Conditionnement par rapport aux racines. Un autre exemple intéressant concerne le conditionnement par rapport à une variable racine du graphe (qui ne dispose pas de père). Par exemple, supposons que nous nous plaçons sous l'hypothèse que la valeur de la variable GPP est connue (disons "Bleu"). Quel est le réseau Bayesien qui représente la loi $P(\mathcal{G}'|GPP = Bl)$, où $\mathcal{G}'$ désigne l'ensemble des autres variables ?

Il suffit d'effacer la variable GPP du graphe de départ, et de modifier les lois de probabilités conditionnelles de ses fils en les "instantiant" correctement. Ici, la loi relative à P serait remplacée par $P(P|GMP, GPP = Bleu)$, c'est à dire la colonne de la table intitiale correspondant à la valeur $GPP = Bleu$.

Avant de discuter plus en détail l'inférence, introduisons la propriété structurelle de base des réseaux Bayesiens, à savoir la notion de $d-$séparation.

5.2.4 D-séparation

Dans cette section nous allons caractériser les relations d'indépendance (conditionnelle, dans le cas général) entre ensembles de variables d'un réseau Bayesien. Précisons que nous recherchons les relations d'indépendance structurelles, c'est-à-dire celles qui découlent seulement de la topologie du réseau et qui sont donc vérifiées quelles que soient les valeurs associées aux tables de probabilités conditionnnelles associées aux noeuds.

Avant de commencer, il faut observer (le fait assez évident) que si deux ensembles de noeuds ne sont reliés par aucun chemin, les ensembles correspondants de variables sont évidemment indépendants. C'est donc des propriétés des chemins qui relient deux ensembles de noeuds que découlera la possibilité d'une dépendance. Désignons par $\mathcal{A}, \mathcal{B}, \mathcal{C}$ trois ensembles disjoints de variables d'un réseau Bayesien $\mathcal{G}$, et essayons de caractériser les conditions (topologiques) sous lesquelles la relation d'indépendance conditionnelle

$$\mathcal{A} \perp \mathcal{C}|\mathcal{B}, \tag{5.4}$$

est satisfaite. En dehors du cas trivial où n'existe aucun chemin reliant $\mathcal{A}$ à $\mathcal{B}$, cette condition sera satisfaite si la connaissance des valeurs de toutes les variables de l'ensemble $\mathcal{B}$ n'autorise pas l'information à circuler entre les deux ensembles $\mathcal{A}$ et $\mathcal{C}$. Il faut donc que tous les chemins reliant ces deux ensembles soient inactifs du point de vue de cette circulation d'information. Voyons les propriétés que doit vérifier un chemin pour qu'il y ait blocage.

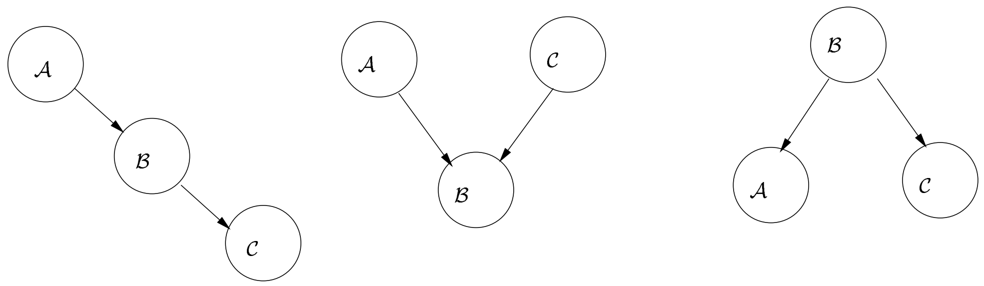
Chemins bloquants. Rappelons qu'un chemin (non-orienté) dans un réseau est une suite de noeuds reliés par des arcs. On dit que le chemin relie le premier noeud de la suite au dernier (et réciproquement). Notons qu'il est possible qu'un chemin contienne des cycles, mais que le nombre de chemins sans cycle qui peuvent être déduits d'un réseau Bayesien est forcément fini.

Considérons la figure 5.3 qui reprend trois configurations possibles de chemins reliant deux (ensembles de) variables et passant par une troisième variable. La configuration (a) correspond à une structure de type "chaîne de Markov", pour laquelle nous savons déjà qu'il y a indépendance conditionnelle entre les deux extrèmes, lorsque le milieu est donné.

La configuration (b) correspond à un problème où les deux variables A et C sont indépendantes a priori et où B est dépendante des valeurs de ces deux variables (voir par exemple, l'exercice 5 de l'appendice 4.A). Pour ce problème, la connaissance de B peut rendre les deux variables A et B dépendantes (alors qu'elles sont nécessairement indépendantes a priori).

Enfin, la situation (c) correspond explicitement à l'indépendance conditionnelle. Ici la connaissance de B rend les variables A et C indépendantes, alors qu'en l'absence de cette connaissance elles peuvent être dépendantes.

Les configurations (a) et (c) sont donc des configurations où la connaissance de B bloque la circulation d'information entre A et C alors que la situation (b) correspond au cas où au contraire elle active la circulation d'information.



(a) $P(A, B, C) = P(A)P(B|A)P(C|B)$ (b) $P(A, B, C) = P(A)P(C)P(B|A, C)$ (c) $P(A, B, C) = P(B)P(A|B)P(C|B)$

Figure 5.3.

La généralisation de ces idées est donnée par la définition suivante.

Bloquage : définition

On dit qu'un chemin qui relie un noeud quelconque de $\mathcal{A}$ à un noeud quelconque de $\mathcal{C}$ est bloqué par l'ensemble de noeuds $\mathcal{B}$ si ce chemin passe par au moins une variable $\mathcal{X}_k$ qui vérifie au moins l'une des trois conditions suivantes :

1. $\mathcal{X}_k$ joue à la fois le rôle de père et de fils dans le chemin et $\mathcal{X}_k \in \mathcal{B}$.
   Il s'agit d'une structure de type $\mathcal{A} \rightarrow \mathcal{B} \rightarrow \mathcal{C}$ ou $\mathcal{A} \leftarrow \mathcal{B} \leftarrow \mathcal{C}$.
2. $\mathcal{X}_k$ joue le rôle de père de deux variables dans le chemin et $\mathcal{X}_k \in \mathcal{B}$.
   Il s'agit d'une structure de type $\mathcal{A} \leftarrow \mathcal{B} \rightarrow \mathcal{C}$.
3. $\mathcal{X}_k$ joue le rôle de fils de deux variables dans le chemin mais ni $\mathcal{X}_k$ ni aucun de ses descendants (dans le graphe) ne figurent dans $\mathcal{B}$.
   Il s'agit d'une structure de type $\mathcal{A} \rightarrow \mathcal{X}_k \leftarrow \mathcal{C}$, où $\mathcal{B}$ ne contient ni $\mathcal{X}_k$ ni aucun de ces descendants.

Dans le cas contraire on dit que le chemin est actif.

Notons que si un chemin sans cycle est bloqué, alors il en sera de même pour tous les chemins dérivés de ce chemin en y ajoutant des variables de manière à former un cycle. Pour vérifier l'existence de chemins actifs, il suffit donc de considérer l'ensemble fini des chemins sans cycle.

On peut alors définir la d-séparation de la manière suivante.

D-séparation : définition

On dit que $\mathcal{A}$ et $\mathcal{C}$ sont d-séparés par $\mathcal{B}$ si tous les chemins reliant un noeud de $\mathcal{A}$ à un noeud de $\mathcal{C}$ sont bloqués par $\mathcal{B}$.

Notons que cette définition implique que si $\mathcal{A}$ et $\mathcal{C}$ sont d-séparés par $\mathcal{B}$, et si $\mathcal{A}' \subset \mathcal{A}$ et $\mathcal{C}' \subset \mathcal{C}$, alors $\mathcal{A}'$ et $\mathcal{C}'$ sont aussi d-séparés par $\mathcal{B}$.

On a la propriété fondamentale suivante (dont nous ne donnerons pas la démonstration)

D-séparation : propriété fondamentale

Si $\mathcal{A}, \mathcal{B}, \mathcal{C}$ sont trois ensembles disjoints de variables d'un réseau Bayesien alors on a $\mathcal{A} \perp \mathcal{C}|\mathcal{B}$ si $\mathcal{A}$ et $\mathcal{C}$ sont d-séparés par $\mathcal{B}$.

Cela veut dire qu'à partir de la structure du graphe on peut directement tester les relations d'indépendance conditionnelle. Notons que la réciproque de cette propriété n'est pas vraie. Il peut exister des relations d'indépendance conditionnelle qui ne sont pas "visibles" graphiquement (voir ci-dessous).

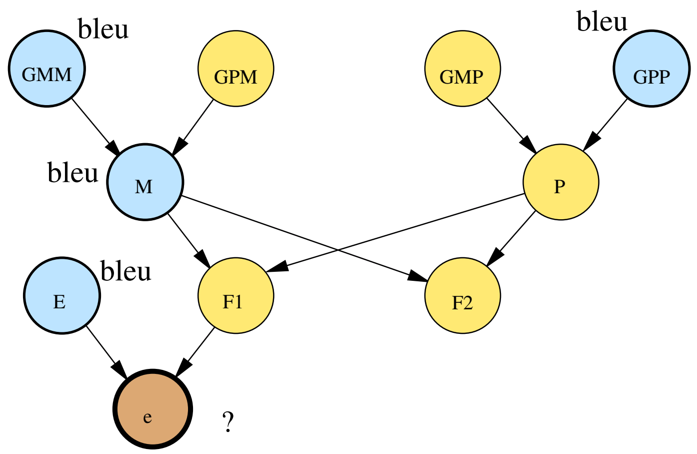


Figure 5.4. Exemple de problème d'inférence à partir d'un réseau Bayesien

Insistons sur le fait que $\mathcal{B}$ dans cette propriété désigne un ensemble quelconque de variables (disjoint de $\mathcal{A}$ et $\mathcal{C}$), et donc en particulier $\mathcal{B}$ peut être l'ensemble vide. La notion de d-séparation “par l'ensemble vide” permet donc aussi de caractériser les indépendances non conditionnelles.

Enfin, remarquons que la propriété de base qui définit les réseaux Bayesiens, à savoir qu'une variable est indépendante de ses non-descendants lorsque on connaît ses pères, est un cas particulier de d-séparation.

Exemple. Dans le réseau Bayesien de la figure 5.4, les ensembles de variables $\{GMM, GPM, GMP, GPP\}$ et $\{E, F1, F2, e\}$ sont d-séparés par l'ensemble de variables $\{M, P\}$. En effet, les chemins qui relient ces deux ensembles de variables sont tous bloqués. Par conséquent, si nous connaissons la couleur des yeux de M et de P, et que nous cherchons à deviner la couleur des yeux de leurs ascendants, il est inutile de tenir compte de la couleur des yeux de $\{E, F1, F2, e\}$.

On voit aussi que $\{GMM, GPM\}$ et $\{GMP, GPP, P, E, F1, F2, e\}$ sont d-séparés par $\{M\}$, ce qui justifie bien la simplification du raisonnement

Par contre, on voit que $\{E\}$ et $\{F1\}$ ne sont pas d-séparés par $\{e\}$: le fait d'observer la couleur des yeux de e rend la couleur des yeux de ses parents dépendantes. Par exemple, si e a des yeux bruns, il est impossible que ses deux parents aient des yeux bleus.

La réciproque de ce théorème n'est pas vraie. En clair, il existe des relations d'indépendance conditionnelle qui ne peuvent pas être représentées graphiquement à l'aide d'un réseau Bayesien.

Cependant, on a les deux propriétés suivantes :

Pouvoir de représentation

N'importe quelle loi de probabilité conjointe peut être représentée (en fait de multiples façons) à l'aide d'un réseau Bayesien, et les inférences sur les relations d'indépendance faites à partir du graphe sont toujours correctes, mais en général pas complètes.

Complétude

Pour tout réseau Bayesien $\mathcal{G}$ il existe une loi de probabilité telle que la structure graphique du graphe représente de manière complète les relations d'indépendance conditionnelle de cette loi.

5.2.5 Inférence

L'inférence consiste à exploiter le modèle probabiliste pour tirer des conclusions sur les valeurs probables de certaines variables étant donné l'observation de certaines autres variables. Il y a plusieurs types de problèmes d'inférence : trouver l'explication la plus plausible d'une série d'observations, calculer la loi de probabilité conditionnelle d'une ou plusieurs variables étant donné un certain nombre d'observations (conditionnement plus marginalisations)...

De façon générale, il s'agit donc de calculer des lois de probabilités conditionnelles de certaines variables non-observées étant donné certaines observations.

En principe ce calcul peut être effectué sans exploiter la structure du réseau en faisant simplement appel à la définition de la probabilité conditionnelle et à la marginalisation. Mais, il se trouve qu'il est possible d'exploiter la structure du réseau Bayesien pour rendre ce type d'opération à la fois plus naturel (plus proche du raisonnement humain) et, dans de nombreux cas, beaucoup plus efficace. Les réseaux Bayesiens sont donc non seulement utiles pour représenter de manière intuitive les relations entre variables mais aussi pour structurer les calculs de façon efficace.

Par exemple, la figure 5.4 illustre une situation particulière. On suppose avoir observé les couleurs de yeux d'un certain nombre de personnes (GMM, GPP, M, et E), et on se pose la question de savoir quelle est la couleur la plus probable des yeux d'une autre personne (e). Pour le besoin de l'exemple, supposons que les yeux de GMM, GPP, M, et E soient bleus, et essayons de calculer la probabilité que e ait des yeux bruns. Nous supposerons que, dans la population qui nous intéresse, la probabilité a priori d'avoir des yeux bleus est 0.6 et celle d'avoir des yeux bruns est 0.4 (nous supposerons que dans cette population il n'y a pas d'yeux verts).

Nous utiliserons ci-dessous la notation $X[br]$ (resp. $X[bl]$) pour indiquer que X est brun (resp. bleu), où X représente une variable quelconque du réseau Bayesien de la figure 5.4.

5.2.5.1 Approche brutale.

Nous pouvons calculer $P(e[br]|GMM[bl], GPP[bl], M[bl], E[bl])$ en calculant

$$P(e[br], GMM[bl], GPP[bl], M[bl], E[bl])$$

et

$$P(GMM[bl], GPP[bl], M[bl], E[bl])$$

et en divisant le premier par le second.

$P(e[br], GMM[bl], GPP[bl], M[bl], E[bl])$ peut être obtenue en marginalisant

$$P(e[br], F1, F2, P, GPM, GMP, GMM[bl], GPP[bl], M[bl], E[bl])$$

sur les variables non-impliquées dans notre raisonnement $F1, F2, P, GPM, GMP$ (2^5 termes). Puis on peut calculer $P(e[bl], GMM[bl], GPP[bl], M[bl], E[bl])$ de la même manière, et sommer ces deux nombres pour obtenir $P(GMM[bl], GPP[bl], M[bl], E[bl])$.

On voit que l'effort de calcul de cette méthode brutale croît exponentiellement avec le nombre de variables non-observées (qui doivent être marginalisées). Dans nos applications futures, nous verrons que le problème du décodage à la sortie d'un canal revient essentiellement à choisir pour un ensemble de variables non-observées (décrivant le message envoyé sur le canal) la combinaison de valeurs la plus probable étant donné une série de variables observées correspondant aux sorties du canal. Dans les systèmes de codage les plus sophistiqués ces messages sont composés de l'ordre de quelques milliers de variables (symboles). L'approche par calcul direct reviendrait dès lors à manipuler des tables de probabilités ayant un nombre de termes de l'ordre 2^{1000} et il est clair que la méthode brutale décrite ci-dessus devient complètement inapplicable.

Il nous faut donc développer des techniques beaucoup plus efficaces et nous allons introduire progressivement de telles techniques dans les sections suivantes. Notons d'emblée que le problème général d'inférence probabiliste sur réseau Bayesien est NP-complet. C'est donc seulement dans le cas de réseaux de structure particulière qu'on peut espérer développer des algorithmes réellement efficaces. Nous verrons que les structures en chaîne (chaînes de Markov) ou en arbre (ce qui en constitue la généralisation) permettent l'inférence efficace.

5.2.5.2 Approche structurée par le graphe.

Le problème fondamental de l'inférence probabiliste revient à marginaliser la distribution de probabilités conjointes représentée par le graphe par rapport aux variables qui ne nous intéressent pas (non-observées).

Elimination des noeuds non-observés. Ce problème est similaire au problème du calcul d'un schéma équivalent d'un circuit électrique, vu d'un certain nombre de noeuds auxquels on s'intéresse. Dans le domaine

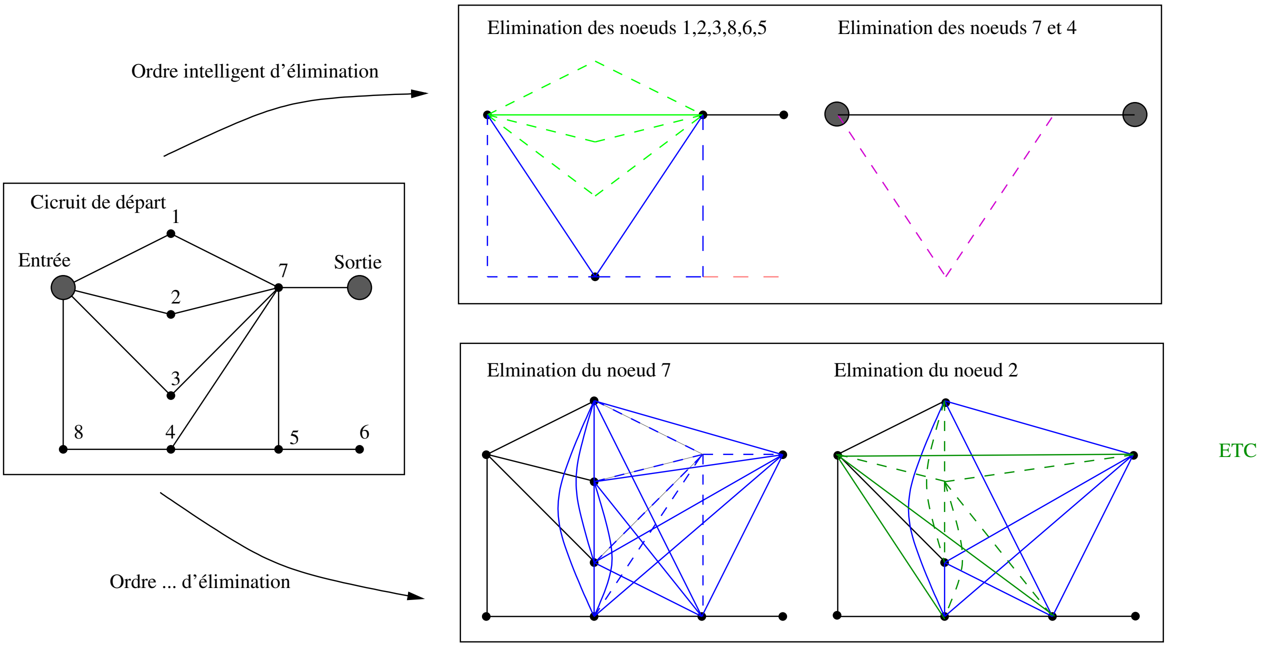


Figure 5.5. Réduction d'un circuit électrique par transfiguration et effet de l'ordre d'élimination des noeuds

de la théorie des circuits électriques il existe essentiellement deux approches à ce problème : (i) l'approche matricielle (globale) qui se sert de calculs directs et en particulier de l'inversion de matrice d'admittances; (ii) l'approche par transfiguration, qui transforme le circuit de départ au moyen d'une suite d'opérations simples de transformation "étoile - triangle". Cette dernière approche consiste à éliminer à chaque étape un noeud intérieur du circuit en ajoutant des liens entre chacun de ses voisins (voir figure 5.5). Puisqu'à chaque étape ce procédé ajoute un certain nombre de connexions au circuit, il est important d'éliminer les noeuds dans un ordre judicieux, en tenant compte de la structure creuse (non complètement connectée) du circuit. A cette fin, il existe diverses techniques d'ordonnancement de noeuds, qui visent à exploiter la structure creuse du circuit pour minimiser le nombre de termes à calculer dans le processus. La figure 5.5 illustre ce processus d'élimination en choisissant d'abord un ordre "efficace" (partie haute de la figure) et un ordre "inefficace". On voit que le choix de l'ordre d'élimination des noeuds peut avoir un impact considérable sur le nombre de calculs. Notons également que dans l'approche matricielle on retrouve ces techniques sous la forme de méthodes de pivotage, qui visent, lors de la décomposition LU, à préserver autant que faire se peut la structure creuse de la matrice d'admittances.

Dans le domaine de l'inférence probabiliste, ces techniques de transfiguration ont également été appliquées. Ici le but est de transfigurer le réseau Bayesien en éliminant les variables non-observées dans le bon ordre. Chaque fois qu'une variable $\mathcal{X}_k$ est éliminée, il faut alors ajouter des liens entre toutes les paires de variables voisines du noeud éliminé qui sont d-séparées (voir ci-dessus) par celui-ci, et remettre à jour les tables de probabilités conditionnelles. Ce type d'approche est intéressante, mais présente comme désavantage essentiel le fait qu'il est *centralisé* et intrinsèquement séquentiel.

Nous allons voir ci-dessous que la propagation de croyances peut quant à elle être réalisée de façon distribuée et asynchrone.

Propagation locale de croyances. L'approche *de propagation locale de croyances* a été développée dans le domaine de l'intelligence artificielle. Comme son nom l'indique, cette technique utilise la structure du graphe pour faire circuler de l'information le long des arcs, le processus étant initié à partir des variables observées. L'avantage de cette approche est de respecter totalement la structure du graphe : elle est donc intéressante à la fois du point de vue interprétabilité et du point de vue efficacité de calcul.

Dans le cadre limité de ce cours, nous ne pouvons malheureusement pas approfondir ces techniques dans toute leur généralité. Cependant, nous expliquerons et illustrerons plus loin un certain nombre de ces techniques dans

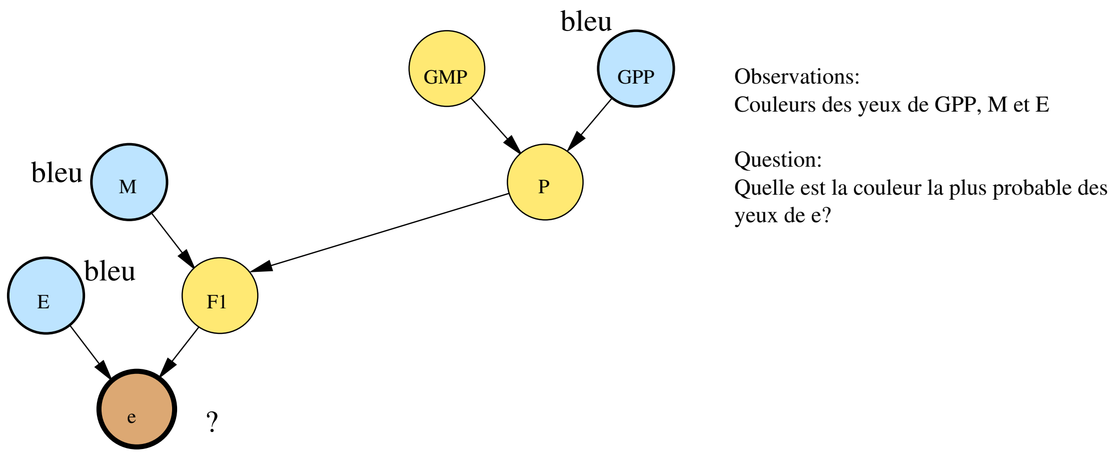


Figure 5.6. Réseau Bayesien simplifié

le cadre plus spécifique des chaînes de Markov auxquelles nous ferons appel pour la modélisation de sources et de canaux de communication. A la fin de ce chapitre nous donnerons quelques indications sur la généralisation de ces techniques pour les structures de réseaux Bayesiens quelconques. Ici, nous allons nous contenter d'illustrer dans le cas de notre exemple comment il est possible d'exploiter la structure d'un réseau Bayesien pour calculer de proche en proche l'effet de la marginalisation sur les variables non-observées.

Illustration. Nous utilisons ci-dessous la notation $X[br]$ (resp. $X[bl]$) pour indiquer que X est brun (resp. bleu), où X représente une variable quelconque du réseau Bayesien de la figure 5.4. La notation X sera utilisée losrque nous voulons désigner une valeur quelconque de cette variable. Notre objectif est de déterminer la probabilité $e[br]$ étant donné les observations $GMM[bl]$, $GPP[bl]$, $M[bl]$, et $E[bl]$.

Commençons par simplifier le réseau Bayesien : pour déterminer la probabilité $e[br]$ étant donné nos observations, on voit intuitivement qu'il n'est pas nécessaire de considérer la partie du réseau qui se situe en amont de M puisque la couleur des yeux de M est connue :

$$P(e[br]|GMM[bl], GPP[bl], M[bl], E[bl]) = P(e[br]|GPP[bl], M[bl], E[bl]). \tag{5.5}$$

On dit que cette observation isole la variable e de la partie du graphe représentant les ascendants de la variable M. (Nous verrons ci-dessous qu'il s'agit d'un cas particulier de *d-séparation*.)

De même, la variable $F2$, n'étant pas observée, peut être éliminée d'emblée du raisonnement (marginalisation sur une feuille). La figure 5.6 montre le réseau bayesien simplifié.

En partant de cette structure et en remontant à partir de e, on calcule

$$\begin{aligned} P(e[br]|GPP[bl], M[bl], E[bl]) &= P(e[br], F1[br]|GPP[bl], M[bl], E[bl]) && (5.6)\\ &\quad +P(e[br], F1[bl]|GPP[bl], M[bl], E[bl]). && (5.7)\end{aligned}$$

Or on a

$$\begin{gathered} P(e[br], F1[br]|GPP[bl], M[bl], E[bl]) \qquad (5.8)\\ = \qquad (5.9)\\ P(e[br]|F1[br], GPP[bl], M[bl], E[bl])P(F1[br]|GPP[bl], M[bl], E[bl]) \qquad (5.10)\\ = \qquad (5.11)\\ P(e[br]|F1[br], E[bl])P(F1[br]|GPP[bl], M[bl], E[bl]), \qquad (5.12)\end{gathered}$$

et une relation similaire pour $P(e[br], F1[bl]|GPP[bl], M[bl], E[bl])$.

Et par le même raisonnement on trouve que

$$P(F1|GPP, M, E) = P(F1, P[br]|GPP, M, E) + P(F1, P[bl]|GPP, M, E) \tag{5.13}$$

où

$$\begin{aligned} P(F1, P|GPP, M, E) &= P(F1|P, GPP, M, E)P(P|GPP, M, E) && (5.14)\\ &= P(F1|P, M)P(P|GPP, M, E) && (5.15)\end{aligned}$$

avec

$$P(P|GPP, M, E) = P(P, GMP[br]|GPP, M, E) + P(P, GMP[bl]|GPP, M, E) \qquad (5.16)$$

où

$$\begin{aligned} P(P, GMP|GPP, M, E) &= P(P|GMP, GPP, M, E)P(GMP|GPP, M, E) && (5.17)\\ &= P(P|GMP, GPP)P(GMP). && (5.18)\end{aligned}$$

Supposons que les lois de probabilité conditionnelles soient les suivantes pour l'ensemble des relations parents-enfant :

$$P(enfant[bl]|parent1[br], parent2[br]) = 0.3, \qquad (5.19)$$
$$P(enfant[bl]|parent1[br], parent2[bl]) = 0.4, \qquad (5.20)$$
$$P(enfant[bl]|parent1[bl], parent2[bl]) = 1.0. \qquad (5.21)$$

On peut alors calculer la probabilité que P ait des yeux bleus (sachant que l'un de ses parents a aussi des yeux bleus) par

$$P(P[bl]|GPP[bl]) = P(P[bl], GMP[bl]|GPP[bl]) + P(P[bl], GMP[br]|GPP[bl]) \qquad (5.22)$$

avec

$$\begin{aligned} P(P[bl], GMP[bl]|GPP[bl]) &= P(P[bl]|GMP[bl], GPP[bl])P(GMP[bl]) && (5.23)\\ &= 0.6 && (5.24)\end{aligned}$$

et

$$\begin{aligned} P(P[bl], GMP[br]|GPP[bl]) &= P(P[bl]|GMP[br], GPP[bl])P(GMP[br]) && (5.25)\\ &= 0.4 \times 0.4 = 0.24 && (5.26)\end{aligned}$$

et donc

$$P(P[bl]|GPP[bl]) = 0.84 \quad \text{et aussi} \quad P(P[br]|GPP[bl]) = 0.16. \qquad (5.27)$$

Nous pouvons ensuite propager cette information vers $F1$ en calculant

$$\begin{aligned} P(F1[bl]|M[bl], GPP[bl]) &= P(F1[bl]|P[bl], M[bl])P(P[bl]|GPP[bl]) && (5.28)\\ &\quad +P(F1[bl]|P[br], M[bl])P(P[br]|GPP[bl]) && (5.29)\\ &= 1.0 \times 0.84 + 0.4 \times 0.16 = 0.904. && (5.30)\end{aligned}$$

On a donc aussi $P(F1[br]|M[bl], GPP[bl]) = 0.096$. Enfin, on trouve

$$\begin{aligned} P(e[br]|M[bl], GPP[bl], E[bl]) &= P(e[br]|F1[bl], E[bl])P(F1[bl]|M[bl], GPP[bl]) && (5.31)\\ &\quad +P(e[br]|F1[br], E[bl])P(F1[br]|M[bl], GPP[bl]) && (5.32)\\ &= 0.0 + 0.096 \times 0.6 = 0.0576. && (5.33)\end{aligned}$$

Exégèse. On voit que la mise à jour de $P(e[br])$ en fonction des observations a été effectuée par un mécanisme de propagation d'informations à partir des variables observées. Il se trouve que notre exemple correspond à un cas particulier, où toutes les observations sont relatives à des ascendants de la variable de sortie. C'est cette propriété qui nous a permis de calculer les probabilités en propageant l'information le long des arcs du graphe de façon *unidirectionnelle* (en suivant le sens des flèches).

Cependant, si par exemple nous avions aussi observé la variable $F2$, ou bien des variables relatives à des descendants de $F2$ (non représentés sur le graphe), ce processus unidirectionnel n'aurait pas été suffisant. Il aurait alors été nécessaire de propager aussi les informations en remontant les flèches. Par exemple, l'observation de $F2$ doit modifier nos croyances en ce qui concerne la couleur des yeux de P et influencerait donc aussi indirectement les croyances en ce qui concerne e.

Il n'est pas évident a priori (c'est-à-dire sans une analyse plus fouillée des relations d'indépendance conditionnelle modélisées par un réseau Bayesien) de se convaincre qu'une technique de propagation purement locale continuerait de fonctionner dans ce type de situation. Cependant, on montre que la méthode d'inférence générale par propagation de croyances peut être réalisée en propageant deux types d'informations relatives aux deux types de liens de causalité entre variables :

1. Cause à effet : l'information transite dans le sens des flèches;

2. Effet à la cause : l'information transite dans le sens inverse des flèches.

Nous verrons plus loin que la loi de probabilité conditionnelle de chaque variable est obtenue par le produit de ces deux types d'informations.

5.2.6 Cas particuliers importants et exemples

Nous allons illustrer au moyen d'un certain nombre d'exemples pratiques le pouvoir de représentation des réseaux Bayesiens.

5.2.6.1 Double "pile ou face". Cet exemple, académique, permet de mieux cerner les limitations et les possibilités théoriques des réseaux Bayesiens. Il s'agit de lancer deux fois de suite une même pièce.

Le problème de double pile ou face est représenté par trois variables binaires $\mathcal{X}$ (0 si pile au premier lancer, 1 sinon), $\mathcal{Y}$ (0 si pile au second lancer, 1 sinon), $\mathcal{Z}$ (0 si résultat identique aux deux lancers, 1 sinon). Les deux variables $\mathcal{X}$ et $\mathcal{Y}$ sont indépendantes a priori (la pièce n'a pas de mémoire) et nous supposerons que la probabilité d'avoir pile vaut $p \in]0;1[$ (la pièce n'est pas nécessairement équilibrée). La loi de probabilité conjointe des trois variables peut naturellement s'écrire sous la forme

$$P(\mathcal{X},\mathcal{Y},\mathcal{Z}) = P(\mathcal{X})P(\mathcal{Y})P(\mathcal{Z}|\mathcal{X},\mathcal{Y}),$$

où la loi conditionnelle $P(\mathcal{Z}|\mathcal{X},\mathcal{Y})$ est représentée par une table à trois dimensions dont les deux plans sont donnés par les deux tableaux

$P(Z=0\|X,Y)$	$Y=0$	$Y=1$
$X=0$	1	0
$X=1$	0	1

$P(Z=1\|X,Y)$	$Y=0$	$Y=1$
$X=0$	0	1
$X=1$	1	0

.

Il s'agit d'un cas particulier où la relation de cause à effet est déterministe qui se traduit par une distribution de Dirac pour la loi conditionnnelle $P(\mathcal{Z}|\mathcal{X},\mathcal{Y})$.

Le réseau Bayesien qui représente les relations entre ces trois variables est représenté à gauche sur la figure 5.7. Il modélise le fait que les variables $\mathcal{X}$ et $\mathcal{Y}$ sont deux causes indépendantes de la variable $\mathcal{Z}$. L'indépendance des deux variables $\mathcal{X}$ et $\mathcal{Y}$ se traduit par un lien manquant dans le graphe, par rapport au réseau Bayesien complet représenté à droite sur la figure 5.7.

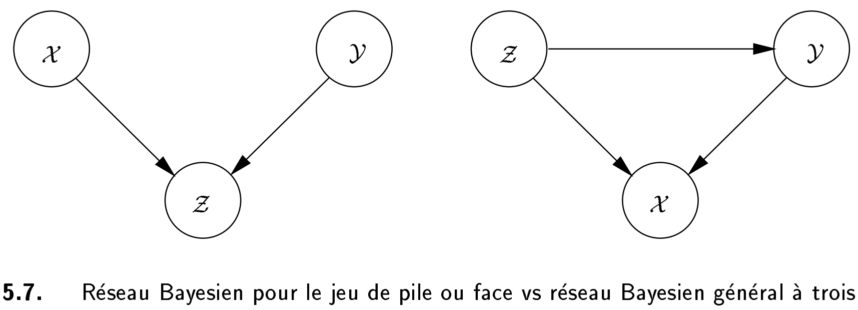


Figure 5.7. Réseau Bayesien pour le jeu de pile ou face vs réseau Bayesien général à trois variables

Calculons la probabilité a priori d'avoir un résultat identique sur les deux lancers $P(Z = 0)$. On a

$$P(Z = 0) = P(X = Y) = P(X = 0, Y = 0) + P(X = 1, Y = 1) = p^2 + (1 - p)^2 = 1 - 2p(1 - p).$$

Par ailleurs, on voit que $P(Z = 0|X = 0) = p$. Par conséquent, on a[1]

$$\mathcal{Z} \perp \mathcal{X} \Leftrightarrow p = 1 - 2p(1 - p) \Leftrightarrow p = \frac{1}{2}.$$

On voit que, si la pièce est équilibrée on aura $\mathcal{Z} \perp \mathcal{X}$ et aussi par symétrie $\mathcal{Z} \perp \mathcal{Y}$. Par contre, si on ne connaît pas la valeur de p on ne peut rien dire sur ces relations d'indépendance conditionnelle. En fait, si $p \neq \frac{1}{2}$ le graphe de gauche de la figure 5.7 représente de manière complète les relations d'indépendance entre les trois variables de notre problème. Par contre, si $p = \frac{1}{2}$ alors il n'existe aucun réseau Bayesien à trois variables qui exprime de façon complète les trois relations d'indépendance

$$\mathcal{X} \perp \mathcal{Y}\ ,\ \mathcal{Z} \perp \mathcal{X}\ ,\ \mathcal{Z} \perp \mathcal{Y}$$

de notre problème, tout en représentant correctement le fait que

$$\mathcal{Z} \not\perp (\mathcal{X}, \mathcal{Y}).$$

Ce problème illustre de la manière la plus simple les possibilités et les limitations de pouvoir de représentation de réseaux Bayesiens, qui sont incapables de représenter certaines relations d'indépendance enfouies dans les valeurs numériques de lois de probabilité.

Enfin, remarquons que le réseau Bayesien de droite de la figure 5.7 représente de manière aussi correcte la loi de probabilité conjointe des trois variables de notre problème. Mais, les variables ayant été mises dans un ordre différent ne prenant pas en compte leurs relations de causalité, on se retrouve avec une représentation graphique où cette fois *toutes* les relations d'indépendance sont cachées dans les valeurs numériques des lois de probabilité associées au réseau. Cependant, lorsque $p = \frac{1}{2}$ on pourrait supprimer l'arc qui relie les variables $\mathcal{Z}$ et $\mathcal{Y}$ puisque ces deux variables deviennent alors indépendantes a priori. On en déduit qu'il existe, lorsque toutes les relations d'indépendance ne peuvent pas être représentées par un seul réseau, plusieurs réseaux Bayesiens qui repésentent différents sous-ensembles de relations d'indépendance.

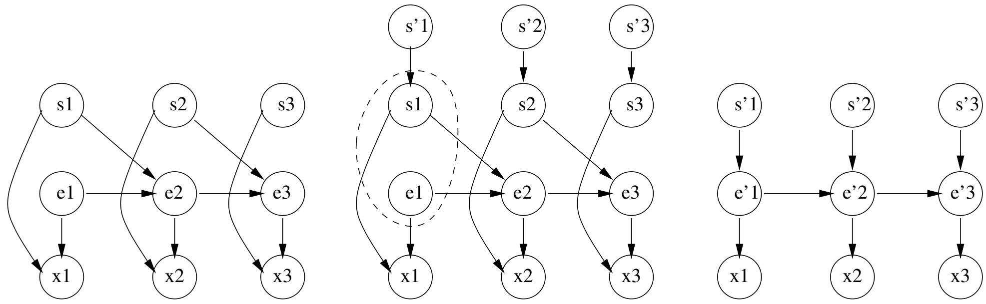
Cependant, même si $p = 0.5$, la seule représentation intuitivement correcte est le réseau de gauche, car il exprime les relations de causalité entre les variables, qui se déduisent directement des propriétés physiques du mécanisme qui produit les valeurs des variables. La structure logique de ce mécanisme ne dépend pas de la valeur de p. Il s'ensuit que les relations qualitatives exprimées par le graphe de gauche sont robustes vis-à-vis de la valeur de p. Dans le cadre d'applications pratiques il est évidemment fondamental de choisir des représentations robustes vis-à-vis des valeurs de paramètres. Dans ce sens, les réseaux Bayesiens offrent un langage de représentation complet, comme l'affirme le théorème de complétude de la section 5.2.4.

5.2.6.2 Graphes non connexes. Si un réseau Bayesien est composé d'un certain nombre de sous-graphes non reliés entre eux, alors les variables correspondant à ces sous-réseaux sont indépendantes, et les problèmes d'inférence peuvent être décomposés a priori en sous-problèmes indépendants relatifs aux sous-graphes connexes. Dans ce qui suit nous ne discuterons donc que de graphes connexes.

5.2.6.3 Chaînes de Markov. La structure la plus simple de réseau Bayesien est linéaire : toutes les variables ont alors au maximum un père et un fils. Ce type de modèle sera utilisé amplement dans le chapitre relatif à la modélisation de sources d'information. On montre que tous les processus physiques ayant une mémoire finie peuvent être modélisés au moyen de ce type de structure. A la section 5.3 nous discuterons d'une variante plus générale de ce type de modèle appelée chaîne de Markov cachée.

5.2.6.4 Arbres. On dit que le réseau Bayesien a une structure d'arbre si tous les noeuds à l'exception d'un seul (appelé racine) disposent exactement d'un seul père. Les chaînes de Markov et les chaînes de Markov cachées forment une sous-classe de cet ensemble de réseaux.

[1]La solution $p = 1$ est exclue par hypothèse, puisqu'elle correspond à une situation dégénérée où toutes les “variables” seraient constantes.



(a) graphe non-arborescent (b) duplication des entrées (b) graphe arborescent

Figure 5.8. Réseau Bayesiens équivalents pour le problème de codage de canal

5.2.6.5 Structures arborescentes (poly-arbres). Les structures les plus générales de réseaux Bayesiens donnant lieu à un algorithme d'inférence efficace sont les structure dites arborescentes. On dit qu'un réseau Bayesien est arborescent si dans le graphe *non-dirigé* obtenu à partir de celui-ci il n'existe pas de cycles. (Le graphe ne doit pas nécessairement être connexe.) Ce type de structure permet à un noeud d'avoir plusieurs pères. On appelle aussi ce genre de structure des poly-arbres car elles peuvent être vues comme la superposition de plusieurs arbres soudés aux noeuds ayant plusieurs pères.

5.2.6.6 Duplication et fusion de noeuds d'un réseau Baysien. Lorsqu'un réseau n'est pas à structure arborescente, il est souvent possible de modifier ce réseau au moyen d'opérations simples qui préservent son pouvoir de réprésentation tout en rendant la structure aborescente.

Fusion de variables. Deux ou plusieurs variables peuvent être remplacées par une seule variable complexe (les valeurs de cette variable correspondent aux combinaisons des valeurs des variables fusionnées) de la manière suivante : la nouvelle variable aura comme parents l'union des parents des variables fusionnées et comme fils l'union de leurs fils. L'opération peut se faire par regroupement successifs de deux variables. Notons que la fusion de deux variables donnera lieu à un réseau licite (sans cycles orientés) pour autant que le réseau de départ n'ait pas de cycles orientés (ce qui est le cas par hypothèse) et que les deux variables ne soient pas descendants indirects l'une de l'autre.

Duplication d'une variable. Une variable $\mathcal{Z}$ peut être dupliquée de la manière suivante : on ajoute une variable $\mathcal{Z}'$ et on impose : (i) le seul parent de la variable $\mathcal{Z}$ est $\mathcal{Z}'$; (ii) les parents de $\mathcal{Z}'$ sont les anciens parents de la variable $\mathcal{Z}$.

Exemple : codage de canal. La figure 5.8 illustre ces idées sur un exemple relatif au problème du codage de canal. La figure 5.8(a) décrit le comportement d'un encodeur de type machine d'état : les variables e_i représentent les états sucessifs; les variables s_i et x_i représentent respectivement les entrées et les sorties aux mêmes instants. La structure du graphe indique qu'à chaque instant l'encodeur produit une sortie fonction de l'état courant et de l'entrée courante, puis effectue une transition vers un nouvel état, fonction également de ces deux variables. Notons qu'en pratique on étudie ce type de système sur des intervalles de temps de l'ordre de 1000 à 10000 instants (voir chapitre 15). Sur la figure 5.8 nous ne représentons que les trois premiers instants.

La structure du réseau n'est évidemment pas arborescente mais on voit sur les figures 5.8(b) et (c) comment en combinant duplication et fusion on peut la rendre aborescente. Notons que tout réseau Bayesien peut être rendu arborescent par ce type d'opération (par exemple en fusionnant plusieurs variables), mais en pratique certaines transformations conduisent à des structures plus efficaces que d'autres (il s'agit de minimiser la taille des variables en termes de nombre de valeurs possibles). Il existe des algorithmes qui effectuent ce genre de transformation automatiquement, mais dans ce qui suit nous nous limiterons au cas des réseaux arborescents et nous ne discuterons pas ces algorithmes.

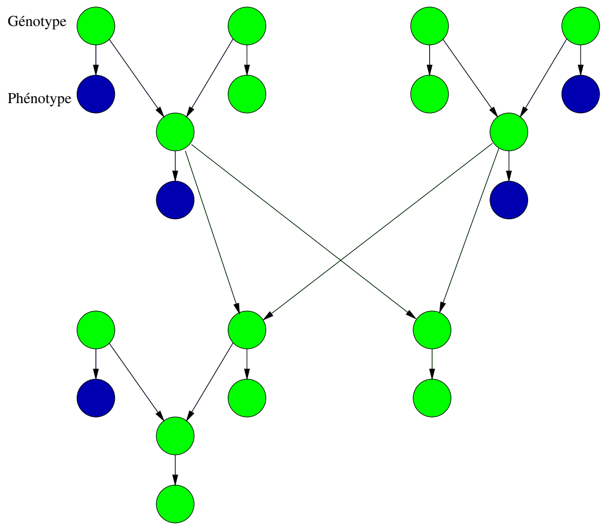


Figure 5.9. Modèle plus complet d'héritage génétique

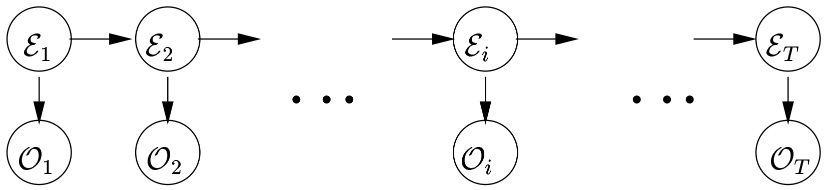


Figure 5.10. Réseau Bayesien représentant une chaîne de Markov cachée

5.2.6.7 Autres exemples pratiques. Nous encourageons le lecteur à consulter la page WEB suivante qui donne accès à un logiciel de simulation de réseaux Bayesiens et un certain nombre d'exemples utilisés dans le cadre de ce cours : http://www.montefiore.ulg.ac.be/∼lwh/javabayes/

En particulier, on peut y trouver la version plus complète (comprenant la modélisation des génotypes) représentée à la figure 5.9 du réseau Bayesien relatif aux couleurs des yeux de notre famille.

5.3 CHAINES DE MARKOV CACHÉES

En général, une chaîne de Markov cachée est constituée comme suit.

- Un ensemble d'indices $i = 1, 2, \ldots, T$ que nous appellerons les instants successifs.
- Les états successifs de la chaîne, que nous noterons $\mathcal{E}_i$ dans cette section. Cet ensemble de variables aléatoires forme une chaîne de Markov au sens précédemment défini.
- Les observations $\mathcal{O}_i$ qui sont supposées vérifier la relation

$$P(\mathcal{O}_i|\mathcal{E}_i, \mathcal{W}) = P(\mathcal{O}_i|\mathcal{E}_i)$$

quel que soit l'ensemble de variables $\mathcal{W}$ composé d'observations et/ou d'états à des instants différents de i. Nous utiliserons aussi le terme de *sorties* pour désigner ces variables.

La figure 5.10 illustre le réseau Bayesien "naturel" qui représente ce genre de modèle explicitement. On voit qu'il s'agit d'une structure d'arbre.

5.3.1 Exemples

L'appellation de "cachée" est due au fait qu'on utilise ces structures pour modéliser des systèmes dont il est impossible d'observer directement l'état; celui-ci est alors caché.

Il se trouve qu'en pratique un très grand nombre de processus peuvent se modéliser à l'aide de tels types de modèles. Notamment, les processus physiques de mémoire finie dont on n'observe que partiellement (ou indirectement) l'état peuvent être modélisés de cette façon.

Ce type de modèle est très populaire en reconnaissance de la parole, et dans le domaine du codage de canal au moyen de codes convolutionnels. Par exemple, nous verrons ultérieurement que les sorties d'un canal avec mémoire finie alimenté par une chaîne de Markov peuvent être modélisées au moyen d'une chaîne de Markov cachée.

5.3.2 Invariance temporelle

On dit que la chaîne de Markov cachée est invariante dans le temps si les lois de probabilités $P(\mathcal{E}_{i+1}|\mathcal{E}_i)$ et $P(\mathcal{O}_i|\mathcal{E}_i)$ ne dépendent pas de l'indice i.

Bien que pour ce qui concerne la discussion présente cette hypothèse ne soit pas fondamentale, nous supposerons néanmoins qu'elle est satisfaite, ce qui nous simplifiera considérablement les notations dans ce qui suit. Nous reviendrons au cas général à la fin de ce chapitre.

Lorsque la chaîne est invariante, les ensembles de valeurs possibles des variables $\mathcal{E}_i$ et $\mathcal{O}_i$ sont évidemment indépendants du temps i. Notons par N le nombre de valeurs possibles des $\mathcal{E}$ et désignons par $1, \ldots, N$ ces valeurs; notons par M le nombre de valeurs possibles des $\mathcal{O}$ et désignons par $1, \ldots, M$ ces valeurs.

Lorsque le modèle est invariant dans le temps, il suffit alors pour le spécifier de définir les nombres suivants

$$\pi_i \triangleq P(\mathcal{E}_1 = i), \quad \Pi_{i,j} \triangleq P(\mathcal{E}_{k+1} = j|\mathcal{E}_k = i), \forall k, \text{et} \quad \Sigma_{i,j} \triangleq P(\mathcal{O}_k = j|\mathcal{E}_k = i), \forall k.$$

Le vecteur π (de dimension N) définit la loi de probabilité des conditions initiales, la matrice Π de dimensions $N \times N$ est appelée matrice de transition, et la matrice Σ de dimensions $N \times M$ est appelée matrice d'observation.

5.3.3 Deux problèmes d'inférence

Nous allons considérer deux problèmes d'inférence particuliers dans le cadre des chaînes de Markov cachées. Des variantes de ces problèmes seront utilisées à plusieurs reprises dans les chapitres ultérieurs de ces notes. Pour ces variantes l'obtention d'algorithmes efficaces suit une démarche analogue à celle que nous allons décrire en détails ci-dessous.

5.3.3.1 Probabilité d'une suite d'observations.

Enoncé. Etant donné une suite d'observations $O^T = O_1 O_2 \cdots O_T$ (une réalisation des v.a. $\mathcal{O}^T = \mathcal{O}_1, \ldots, \mathcal{O}_T$) et la spécification du modèle (π, Π, Σ) calculer la probabilité $P(O^T)$.

Solution. Le but est de trouver un algorithme efficace qui calcule cette probabilité. Notons que le calcul direct (par marginalisation sur les valeurs possibles de $\mathcal{E}^T$) revient à calculer une somme comprenant N^T termes, chacun de ces termes étant le produit de $2T$ facteurs.

La procédure efficace est basée sur le calcul récursif "gauche $\rightarrow$ droite" des coefficients suivants

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, \mathcal{E}_t = i). \tag{5.34}$$

La procédure est la suivante

1. Initialisation :
$$\alpha_1(i) = P(O_1, \mathcal{E}_1 = i) = \pi_i \Sigma_{i,O_1}, \ 1 \leq i \leq N \tag{5.35}$$

2. Induction :
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) \Pi_{i,j}\right] \Sigma_{j,O_{t+1}}. \tag{5.36}$$

3. Terminaison
$$P(O^T) = \sum_{i=1} \alpha_T(i). \tag{5.37}$$

La justification de l'induction est la suivante :
$$\begin{aligned}
\alpha_{t+1}(j) &= P(O_1 O_2 \cdots O_t, \mathcal{E}_{t+1} = j, O_{t+1}) && (5.38)\\
&= P(O_1 O_2 \cdots O_t, \mathcal{E}_{t+1} = j) P(O_{t+1} | O_1 O_2 \cdots O_t, \mathcal{E}_{t+1} = j) && (5.39)\\
&= P(O_1 O_2 \cdots O_t, \mathcal{E}_{t+1} = j) P(O_{t+1} | \mathcal{E}_{t+1} = j) && (5.40)\\
&= P(O_1 O_2 \cdots O_t, \mathcal{E}_{t+1} = j) \Sigma_{j, O_{t+1}}, && (5.41)
\end{aligned}$$
à cause de l'indépendance conditionnelle de $\mathcal{O}_{t+1}$ par rapport aux autres variables du graphe lorsque $\mathcal{E}_{t+1}$ est donnée. D'autre part,
$$\begin{aligned}
P(O_1 O_2 \cdots O_t, \mathcal{E}_{t+1} = j) &= \sum_{i=1}^{N} P(O_1 O_2 \cdots O_t, \mathcal{E}_t = i, \mathcal{E}_{t+1} = j) && (5.42)\\
&= \sum_{i=1}^{N} P(O_1 O_2 \cdots O_t, \mathcal{E}_t = i) P(\mathcal{E}_{t+1} = j | O_1 O_2 \cdots O_t, \mathcal{E}_t = i) && (5.43)\\
&= \sum_{i=1}^{N} P(O_1 O_2 \cdots O_t, \mathcal{E}_t = i) P(\mathcal{E}_{t+1} = j | \mathcal{E}_t = i) && (5.44)\\
&= \sum_{i=1}^{N} \alpha_t(i) \Pi_{i,j}, && (5.45)
\end{aligned}$$
où le passage de (5.43) à (5.44) est justifié par le fait que $\mathcal{E}_t$ est le (seul) père de $\mathcal{E}_{t+1}$ et que les variables $\mathcal{O}_1, \mathcal{O}_2 \ldots \mathcal{O}_t$ sont des non-descendants de $\mathcal{E}_{t+1}$.
La dernière équation (terminaison) résulte simplement d'un marginalisation par rapport à $\mathcal{E}_T$.$\square$

Interprétation en termes de propagation locale d'informations dans le graphe. On peut voir que cet algorithme consiste à propager de l'information d'une part dans le sens des arcs $\mathcal{E}_{t-1} \stackrel{\Pi}{\rightarrow} \mathcal{E}_t$ d'autre part dans le sens inverse des arcs $\mathcal{O}_t \stackrel{\Sigma}{\rightarrow} \mathcal{E}_t$. Ces informations sont multipliées par les matrices représentant les lois de probabilités conditionnelles reliant les deux noeuds.

En effet désignons par $Bel_{\mathcal{O}_t}(i) = \delta_{O_t,i}$ les composantes d'un vecteur de longueur M ayant un 1 en position O_t et un zéro partout ailleurs[2]. Ces vecteurs identifient les observations aux différents instants. La valeur Σ_{j,O_t} peut dès lors être vue comme le résultat de l'opération
$$\mu_t(j) = \sum_{i=1}^{M} \Sigma_{j,i} Bel_{\mathcal{O}_t}(i),$$
qu'on peut interpréter comme la propagation arrière de $Bel_{\mathcal{O}_t}(i)$ au travers du lien caractérisé par $\Sigma_{j,i}$.

Par ailleurs, la formule (5.45) peut être vue comme une formule de propagation avant au travers du lien $\mathcal{E}_{t-1} \stackrel{\Pi}{\rightarrow} \mathcal{E}_t$
$$\lambda_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \Pi_{i,j},$$
en utilisant la matrice de transition Π comme fonction de transfert.

Enfin, le calcul de $\alpha_t(i)$ est obtenu simplement en multipliant terme à terme les vecteurs λ et μ reçus en chaque noeud :
$$\alpha_t(i) = \lambda_t(i) \mu_t(i).$$

Il faut ajouter, pour être complet que les valeurs $\lambda_1(i)$ sont initialisées aux probabilités a priori π_i, puisque ce noeud ne dispose d'aucun père. Le schéma de propagation le long du graphe est représenté à la figure 5.11.

[2]Nous utilisons cette notation pour désigner le fait que ce vecteur représente notre croyance (Belief) en ce qui concerne les valeurs de $\mathcal{O}_t$ compte tenu des observations

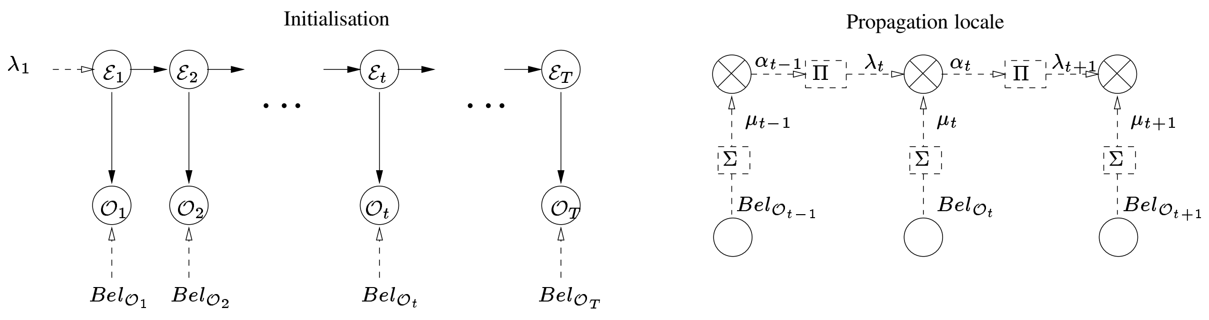


Figure 5.11. Illustration graphique de la propagation de croyances

Variantes.

Cas non-invariant. Si la chaîne est variante dans le temps, il suffit de modifier le schéma de calcul en faisant intervenir des matrices de transition et d'observation Π_t et Σ_t dépendant du temps.

Sorties partiellement observées. Supposons que la sortie $\mathcal{O}_k$ ne soit pas observée. On souhaite alors calculer la probabilité des observations $P(O_1, \ldots, [O_k], \ldots O_T)$. On pourra se convaincre qu'il suffit pour cela de modifier la formule (5.36) au moment du calcul de la valeur de $\alpha_k(j)$ de la manière suivante :

$$\alpha_k(j) = \sum_{i=1}^{N} \alpha_{k-1}(i)\Pi_{i,j},$$

c'est-à-dire qu'on ne prend pas en compte l'information relative à cette observation au moment de la propagation. Un autre façon de réaliser la même opération consiste à utiliser pour $\mathcal{O}_k$ un vecteur de croyance neutre ($Bel_{\mathcal{O}_k}(i) = 1$), et on obtient

$$\mu_k(i) = \sum_{i=1}^{M} \Sigma_{j,i} Bel_{\mathcal{O}_k}(i) = 1.$$

Observation de certains états. Supposons qu'en plus d'observer les sorties (ou certaines d'entre-elles), nous observions également la valeur d'un état, disons $\mathcal{E}_k = E_k$. Dans ce cas, nous souhaiterions calculer la valeur de $P(O_T, E_k)$. Il faut modifier la formule de propagation de façon à éviter la marginalisation sur $\mathcal{E}_k$ au moment du calcul de $\alpha_{k+1}(i)$. Cela peut être réalisé en multipliant $\alpha_k(i)$ par une fonction de croyance $Bel_{\mathcal{E}_k}(i) = \delta_{E_k,i}$, avant l'application de la formule (5.36).

On déduit de ce qui précède une version généralisée de l'algorithme de calcul de la probabilité conjointe d'une série d'observations dans une chaîne de Markov cachée (donc aussi dans une chaîne de Markov) : on associe à chaque variable observée (état ou sortie) une fonction de croyance (impulsion de Dirac placée à la valeur observée), et à chaque variable non-observée une fonction de croyance neutre (un vecteur de 1), puis on propage cette information selon les règles que nous venons d'expliquer.

5.3.3.2 Explication des sorties observées. Le second problème d'inférence que nous allons considérer consiste à déterminer la séquence d'états qui explique le mieux une suite d'observations.

Il faut d'abord remarquer que lorsqu'on parle de la *meilleure* explication d'une série d'obervations cela suppose qu'on se définisse un critère permettant de mesurer la qualité d'une explication. En pratique ces critères sont basés sur une mesure de la probabilité de se tromper (de choisir la mauvaise explication), et il y a deux variantes classiques que nous allons considérer.

Enoncé 1 (BCJR)
Etant donnée une suite d'observations O^T, déterminer pour chaque instant t la loi conditionnelle $P(\mathcal{E}_t|O^T)$. La solution de ce problème est donnée par l'algorithme BCJR (Bahl, Cocke, Jelinek, Raviv). La meilleure explication est alors la suite d'états définie par

$$E_t = \arg\max_i P(\mathcal{E}_t = i|O^T),$$

dans le sens où ce choix minimise la probabilité d'erreur moyenne pour chacun des états E_t choisis.

Algorithme BCJR. Cet algorithme est aussi appelé algorithme "avant-arrière", parce qu'il est basé sur la combinaison des formules d'induction "gauche-droite" pour le calcul des $\alpha_t(i)$ données plus haut et une procédure analogue "droite-gauche" pour le calcul des coefficients suivants

$$\beta_t(i) = P(O_{t+1}, \ldots, O_T|\mathcal{E}_t = i). \tag{5.46}$$

Montrons d'abord que la connaissance des valeurs de $\alpha_t(i)$ et $\beta_t(i)$ suffit pour le calcul des valeurs de $P(\mathcal{E}_t = i|O^T)$. En effet, on a

$$\begin{aligned}
P(O^T, \mathcal{E}_t = i) &= P(O_1, \ldots, O_t, \mathcal{E}_t = i, O_{t+1}, \ldots O_T) && (5.47)\\
&= P(O_1, \ldots, O_t, \mathcal{E}_t = i) P(O_{t+1}, \ldots O_T|O_1, \ldots, O_t, \mathcal{E}_t = i) && (5.48)\\
&\stackrel{a}{=} P(O_1, \ldots, O_t, \mathcal{E}_t = i) P(O_{t+1}, \ldots O_T|\mathcal{E}_t = i) && (5.49)\\
&= \alpha_t(i)\beta_t(i), && (5.50)
\end{aligned}$$

où le passage (a) est justifié par la proprieté de d-séparation des ensembles $\mathcal{O}_1, \ldots, \mathcal{O}_t$ et $\mathcal{O}_{t+1}, \ldots, \mathcal{O}_T$ par la variable $\mathcal{E}_t$.

De l'équation (5.50) on déduit que

$$P(O^T) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i), \tag{5.51}$$

et comme

$$P(\mathcal{E}_t = i|O^T) = \frac{P(O^T, \mathcal{E}_t = i)}{P(O^T)}, \tag{5.52}$$

on obtient finalement que

$$P(\mathcal{E}_t = i|O^T) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}. \tag{5.53}$$

La formule d'induction pour les $\beta_t(i)$ est la suivante :

1. Initialisation :
$$\beta_T(i) = 1,\ 1 \leq i \leq N \tag{5.54}$$
2. Induction :
$$\beta_t(j) = \sum_{i=1}^{N} \Pi_{j,i} \Sigma_{i,O_{t+1}} \beta_{t+1}(i). \tag{5.55}$$

Cela revient donc à multiplier terme à terme les vecteurs β_{t+1} et μ_{t+1} (en provenance des deux noeuds aval) et à les propager au travers de la matrice de transition Π vers le noeud amont. Nous laissons le soin au lecteur de se convaincre que ces formules sont bien correctes.

Remarques. Des variantes plus générales de cet algorithme peuvent être obtenues au moyen de raisonnements analogues à ceux de la section 5.3.3.1. L'algorithme avant arrière peut aussi être vu comme une généralisation des règles de propagation locale en sens inverse des arcs du graphe. On voit que chaque variable $\mathcal{E}_t$ peut calculer la probabilité conjointe de ses propres valeurs avec les observations effectuées sur l'ensemble du réseau, au moyen de la combinaison des informations locales α_{t-1}, $Bel_{\mathcal{O}_t}$ et β_{t+1} qui lui sont communiquées par ses trois voisins. A partir de cette table, il est possible de calculer la probabilité des observations (par marginalisation) et la loi conditionnelle des états par conditionnement.

Enoncé 2 (Viterbi)
Etant donné une suite d'observations O^T, trouver une suite d'états E^T telle que $P(E^T|O^T)$ soit maximale. Il s'agit donc de trouver la *combinaison* d'états qui soit conjointement l'explication la plus probable. On peut en effet reprocher à la première formulation le fait qu'elle ne produit pas nécessairement une suite d'états possibles de la chaîne de Markov sous-jacente (le modèle probabiliste peut en effet exclure certaines transitions). Le présent énoncé permet de remédier à ce problème. La solution est donnée par l'algorithme de VITERBI sur lequel nous reviendrons tout à la fin de ce cours.

Algorithme de Viterbi. Notons tout d'abord que la séquence d'états E^T qui maximise $P(E^T|O^T)$ est aussi celle qui maximise $P(E^T, O^T)$, puisque O^T est fixée.

L'algorithme de Viterbi est basé sur le principe de la programmation dynamique, qui permet de calculer itérativement la grandeur suivante

$$\delta_t(i) = \max_{\mathcal{E}_1,\ldots,\mathcal{E}_{t-1}} P(\mathcal{E}_1,\ldots,\mathcal{E}_{t-1},\mathcal{E}_t = i, O_1,\ldots,O_t), \tag{5.56}$$

qui est la probabilité de la suite d'états la plus probable menant à l'état i à l'instant t prise conjointement avec les t premières observations. De toute évidence la connaissance de la valeur $\max_i \delta_T(i)$ et de la suite d'états correspondante résoudrait notre problème.

Le calcul est basé sur l'induction suivante où on mémorise en même temps dans le vecteur $\psi_t(i)$ les suites optimales d'états.

1. Initialisation :

$$\delta_1(i) = P(O_1, \mathcal{E}_1 = i) = \pi_i \Sigma_{i,O_1},\ 1 \le i \le N, \tag{5.57}$$
$$\psi_1(i) = 0. \tag{5.58}$$

2. Induction :

$$\delta_{t+1}(j) = \left[\max_{i=1}^{N} \delta_t(i)\Pi_{i,j}\right] \Sigma_{j,O_{t+1}}, \tag{5.59}$$
$$\psi_t(j) = \arg\max_{i=1}^{N} \left[\delta_{t-1}(i)\Pi_{i,j}\right]. \tag{5.60}$$

3. Terminaison

$$P^* = \max_{i=1}^{N} \left[\delta_T(i)\right], \tag{5.61}$$
$$E_T^* = \arg\max_{i=1}^{N} \left[\delta_T(i)\right]. \tag{5.62}$$

4. Reconstitution (arrière) de la suite d'états :

$$E_t^* = \psi_{t+1}(E_{t+1}^*). \tag{5.63}$$

Avant d'expliquer pourquoi cet algorithme reconstitue bien la suite optimale d'états, remarquons que les formules d'induction sont identiques aux formules de calcul des $\alpha_t(i)$ à cela près que l'opérateur $\sum$ est remplacé par l'opérateur $\max$. C'est pour cela qu'on appelle dans la littérature l'algorithme de calcul des $\alpha_t(i)$ l'algorithme "somme-produit" et l'algorithme de calcul des $\delta_t(i)$ l'algorithme "max-produit".

Justification de l'algorithme de Viterbi. On a d'une part :

$$P(\mathcal{E}_1, \ldots \mathcal{E}_{t+1}, O_1, \ldots, O_{t+1}) = P(\mathcal{E}_1, \ldots, \mathcal{E}_t, O_1, \ldots, O_t) P(\mathcal{E}_{t+1}, O_{t+1} | \mathcal{E}_t), \tag{5.64}$$

à cause de la propriété de d-séparation ($\mathcal{E}_t$ bloque tous les chemins qui relient les variables qui lui sont en aval aux variables qui lui sont en amont).

D'autre part, on a

$$\begin{aligned}
\delta_{t+1}(j) &\stackrel{\triangle}{=} \max_{\mathcal{E}_1, \ldots, \mathcal{E}_t} P(\mathcal{E}_1, \ldots, \mathcal{E}_t, \mathcal{E}_{t+1} = j, O_1, \ldots, O_t, O_{t+1}) && (5.65)\\
&= \max_i \max_{\mathcal{E}_1, \ldots, \mathcal{E}_{t-1}} P(\mathcal{E}_1, \ldots, \mathcal{E}_t = i, \mathcal{E}_{t+1} = j, O_1, \ldots, O_t, O_{t+1}) && (5.66)\\
&\stackrel{a}{=} \max_i \max_{\mathcal{E}_1, \ldots, \mathcal{E}_{t-1}} P(\mathcal{E}_1, \ldots, \mathcal{E}_t = i, O_1, \ldots, O_t) P(\mathcal{E}_{t+1} = j, O_{t+1} | \mathcal{E}_t = i) && (5.67)\\
&= \max_i [\delta_t(i) P(\mathcal{E}_{t+1} = j, O_{t+1} | \mathcal{E}_t = i)] && (5.68)\\
&= \max_i [\delta_t(i) P(\mathcal{E}_{t+1} = j | \mathcal{E}_t = i) P(O_{t+1} | \mathcal{E}_t = i, \mathcal{E}_{t+1} = j)] && (5.69)\\
&= \max_i [\delta_t(i) \Pi_{i,j}] \, P(O_{t+1} | \mathcal{E}_{t+1} = j) && (5.70)\\
&= \max_i [\delta_t(i) \Pi_{i,j}] \, \Sigma_{j, O_{t+1}}. && (5.71)
\end{aligned}$$

Nous laissons le soin au lecteur de vérifier que le stockage des valeurs de $\psi_t(i)$ et l'algorithme de retour arrière reconstituent bien la séquence optimale d'états.

5.4 INFÉRENCE EN GÉNÉRAL DANS LES RÉSEAUX ARBORESCENTS

Les idées que nous venons de décrire peuvent être généralisées à des réseaux généraux pour autant qu'ils soient arborescents, donnant lieu à des algorithmes locaux de propagation d'évidences (de type message-passing), qui peuvent être exécutés en parallèle et en mode asynchrone, au fur et à mesure de l'arrivée de nouvelles observations.

Nous invitons le lecteur à consulter les références [Pea88, Fre99] pour plus de détails.

5.5 INFÉRENCE DANS LES RÉSEAUX BAYSIENS QUELCONQUES

L'inférence est réalisée au moyen de deux étapes :

1. Transformation du réseau en poly-arbre (algorithme dit de l'arbre de jonction).
2. Utilisation de la propagation locale sur l'arbre de jonction.

5.6 ARBRES DE DECISION

Considérons un problème de diagnostic médical et supposons que nous disposions d'un modèle probabiliste qui décrit la loi conjointe d'un certain nombre de variables qui décrivent les symptômes $\mathcal{S}_i$ et les antécédents $\mathcal{A}_j$ d'un patient ainsi que d'une variable $\mathcal{D}$ dont les valeurs correspondent aux diagnostics possibles dans le contexte considéré.

Par exemple, la figure 5.12 illustre une structure typique de réseau Bayesien pour ce problème. Cette structure met en évidence la nature des relations entre les différentes variables : les antécédents sont des causes potentielles d'une maladie alors que les symptômes en sont les conséquences. Les deux types de variables sont utiles au médecin pour effectuer son diagnostic. Typiquement, le médecin vérifiera certains symptômes (concentrations de divers types de cellules sanguines, couleur des yeux, de la peau, température, douleurs,...) et posera également des questions concernant les antécédents (âge, habitudes alimentaires, maladies familiales, ...). Nous allons supposer qu'il acquiert ces informations en posant successivement des questions au patient lui demandant d'indiquer chaque fois la valeur d'une des variables.

Une façon d'effectuer le diagnostic serait de déterminer pour chaque patient les valeurs de toutes les variables observables $\mathcal{S}_i$ et $\mathcal{A}_j$, puis d'utiliser le modèle probabiliste pour déterminer la loi conditionnelle

$$P(\mathcal{D} | A_1, \ldots, A_n, S_1, \ldots, S_m).$$

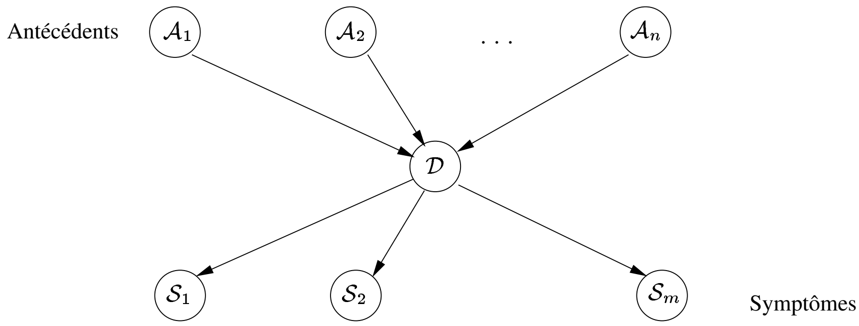


Figure 5.12. Réseau Bayesien pour un problème de diagnostic médical

Cette loi contient en effet toute l'information relative au diagnostic et doit permettre au médecin de trancher. Par exemple, en supposant que la variable $\mathcal{D}$ est binaire (présence ou absence d'une certaine maladie), la décision du médecin pourrait être :

Si $P(\mathcal{D}|\ldots) \leq \epsilon$ **:** on décide que le patient est sain.

Si $P(\mathcal{D}|\ldots) \geq 1 - \epsilon$ **:** on décide que le patient est malade.

Sinon : on décide de faire appel à un spécialiste.

Pour justifier le prix de la consultation, on espère que la dernière alternative est peu fréquente, ce qui en termes de mesures d'entropie signifie que l'entropie conditionnelle moyenne

$$H(\mathcal{D}|A_1, \ldots, A_n, S_1, \ldots, S_m)$$

est faible. Tout l'art de la médecine est de déterminer pour chaque type de maladie une série pertinente de questions à poser au malade.

Bien que cette approche systématique puisse convenir, en général toutes les questions n'ont pas le même niveau de pertinence et certaines questions ne seront posées qu'à certains patients, en fonction de réponses obtenues aux questions ou analyses précédentes. En effet, le médecin souhaite ne poser que les questions utiles et éviter tout examen qui n'apporterait pas d'information. Le problème qui se pose est dès lors le suivant :

Comment construire une stratégie pour effectuer le diagnostic au moyen d'un nombre minimal de questions ?

5.6.1 Construction d'un questionnaire

Pour résoudre ce problème, nous allons utiliser la théorie de l'information afin de construire un questionnaire sous la forme d'un arbre de décision.

5.6.1.1 Une seule question. Supposons d'abord que le médecin ne puisse poser qu'une seule question portant sur une seule des variables observables $\mathcal{S}_i$ ou $\mathcal{A}_j$, et cherchons à déterminer laquelle de ces variables il devrait choisir.

Avant de poser sa question, l'incertitude du médecin est mesurée par l'entropie a priori $H(\mathcal{D})$. S'il choisit de tester la valeur de la variable $\mathcal{X} \in \{A_1, \ldots, A_n, S_1, \ldots, S_m\}$, son incertitude sera en moyenne égale à

$$H(\mathcal{D}|\mathcal{X}).$$

La variable la plus utile dans ce contexte est donc la variable qui minimise cette grandeur, ou de manière équivalente la variable qui apporte une information maximale :

$$\mathcal{X}_* = \arg \max_{\mathcal{X} \in \{A_1, \ldots, A_n, S_1, \ldots, S_m\}} I(\mathcal{D}; \mathcal{X}). \tag{5.72}$$

En théorie, on a évidemment

$$H(\mathcal{D}|\mathcal{X}_*) \geq H(\mathcal{D}|A_1, \ldots, A_n, S_1, \ldots, S_m),$$

cependant s'il se trouvait que

$$H(\mathcal{D}|\mathcal{X}^*) \simeq H(\mathcal{D}|A_1, \ldots, A_n, S_1, \ldots, S_m)$$

alors il ne serait pas nécessaire de poser d'autres questions, toute (ou presque) l'information pertinente étant fournie par la question optimale.

5.6.1.2 Série de questions suffisantes. Il est rare qu'une seule information soit suffisante dans ce type de problème.

Supposons donc qu'ayant posé la première question et ayant obtenu la réponse $\mathcal{X}_* = X$, l'incertitude sur $\mathcal{D}$ soit trop grande pour effectuer le diagnostic, en d'autres termes :

$$H(\mathcal{D}|\mathcal{X}_* = X) \gg H(\mathcal{D}|A_1, \ldots, A_n, S_1, \ldots, S_m). \tag{5.73}$$

Il faut donc trouver une seconde question, et cela peut se faire selon la même procédure de recherche exhaustive en déterminant la variable

$$\mathcal{X}'_* = \arg \max_{\mathcal{X} \in \{A_1, \ldots, A_n, S_1, \ldots, S_m\}} I(\mathcal{D}; \mathcal{X}|\mathcal{X}_* = X), \tag{5.74}$$

où cette fois on utilisera la loi conditionnelle par rapport aux observations déjà effectuées. Pour un patient donné, ce processus s'arrête lorsqu'une des deux conditions suivantes est remplie :

1. Toutes les variables ont été testées.
2. L'entropie conditionnelle par rapport aux observations déjà effectuées est suffisamment faible.

Ce processus permet donc, grâce à l'utilisation du modèle probabiliste de déterminer une bonne stratégie pour effectuer le diagnostic sous la forme d'une suite de questions pertinentes.

5.6.1.3 Arbre de décision. Analysons les suites de questions qui seront posées si on utilise cette stratégie pour différents patients.

- La première question est toujours la même.
- La question suivante peut varier en fonction du cas rencontré, car la solution de (5.74) dépend de la valeur de $\mathcal{X}_*$ (réponse à la première question) qui doit forcément varier d'un patient à l'autre (sinon cette variable n'apporterait aucune information).
- Cependant, si deux patients ont répondu de la même manière aux $k-1$ premières questions, le médecin réagira de manière identique à l'étape k (soit il ne pose plus de questions et pose son diagnostic, soit il leur posera la même question à tous les deux.)

L'ensemble de toutes les suites de questions posées pour tous les patients possibles peut se structurer sous la forme d'un arbre. La figure 5.13 illustre ceci dans un cas simplifié où nous avons supposé que toutes les variables sont binaires (et que l'arbre se résume à deux questions). Chaque diagnostic correspondra à un chemin dans cet arbre qui s'arrête au moment où on atteint une feuille (noeud sans successeur noté T_i). De même qu'on peut déterminer a priori cet arbre on peut déterminer a priori les lois de probabilités conditionnelles $P(\mathcal{D}|T_i)$ correpondant aux feuilles.

Puisqu'à chaque combinaison de valeurs des variables aléatoires de départ $\mathcal{S}_i$ et $\mathcal{A}_j$ correspond un et un seul chemin dans l'arbre se terminant en une feuille T_i, on peut considérer que l'arbre définit une variable aléatoire discrète $\mathcal{T}$ en fonction des variables de départ

$$\mathcal{T} = f(\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{S}_1, \ldots, \mathcal{S}_m), \tag{5.75}$$

et encode la loi conditionnelle

$$P(\mathcal{D}|\mathcal{T}). \tag{5.76}$$

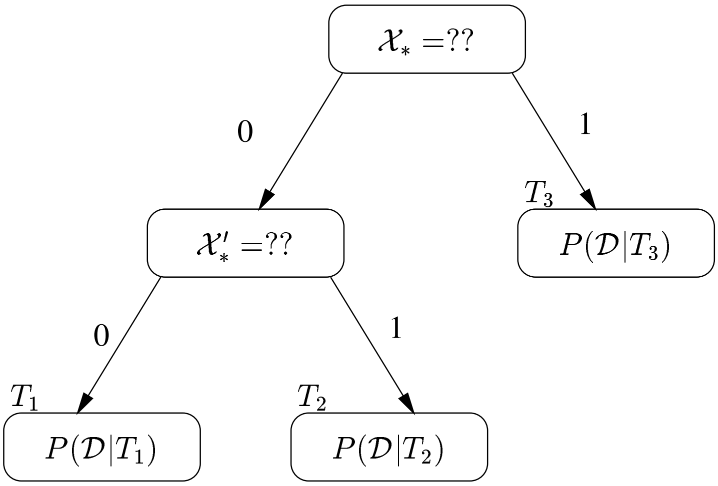


Figure 5.13. Arbre de décision pour un problème de diagnostic médical

Si l'arbre est complètement développé (on force l'épuisement des variables lors de sa construction) la variable $\mathcal{T}$ sera en bijection avec la combinaison de variables de départ et l'arbre founira toute l'information disponible. Si nous supposons que les variables sont toutes binaires, l'arbre complet aura une profondeur de $n+m$ et comportera exactement 2^{n+m} feuilles. Il conduira à une incertitude résiduelle de

$$H(\mathcal{D}|\mathcal{T}) = H(\mathcal{D}|\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{S}_1, \ldots, \mathcal{S}_m). \tag{5.77}$$

Il est cependant possible que l'arbre complet soit jugé trop complexe et qu'on souhaite le simplifier de manière optimale en élagant certaines parties. On peut alors avoir

$$H(\mathcal{D}|\mathcal{T}) > H(\mathcal{D}|\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{S}_1, \ldots, \mathcal{S}_m), \tag{5.78}$$

ce qui veut dire qu'un arbre élagé fournira en général moins d'information sur la variable $\mathcal{D}$ que notre réseau bayesien de départ. Pour décider de combien il faut élaguer l'arbre, on peut juger que l'arbre optimal serait celui qui minimiserait une mesure du type

$$Q(\mathcal{T}) = H(\mathcal{D}|\mathcal{T}) + \beta|\mathcal{T}|, \tag{5.79}$$

où $|\mathcal{T}|$ désigne le nombre de valeurs de la variable $\mathcal{T}$ définie par l'arbre, c'est-à-dire le nombre de feuilles de celui-ci, et où $\beta \geq 0$ est un coefficient de pondération qui définit le compromis souhaité entre information apportée et complexité.

Notons qu'il existe des algorithmes d'élagage qui effectuent ce travail automatiquement pour toutes les valeurs possibles de β (voir par exemple cours d'apprentissage inductif appliqué).

5.6.2 Synthèse

Les arbres de décision fournissent une représentation compacte d'une loi de probabilité conditionnelle. En pratique il s'agit d'une outil extrêmement intéressant, notamment parce qu'il permet de représenter de manière explicite et interprétable les relations entre une série de variables et une (ou plusieurs) variables de sortie.

Nous ferons également appel à ce type de structure à plusieurs reprises dans le domaine du codage de source (compression de données).

Nous verrons au chapitre suivant qu'il est possible de construire de manière efficace un arbre de décision à partir non pas d'un modèle probabiliste mais d'un échantillon représentatif de la loi conjointe.

Un des problèmes pratiques qui se posent dans le cadre des arbres de décision concerne le traitement des valeurs manquantes. Que faut-il faire si le patient ne sait pas répondre à l'une des questions posées par le médecin ?

5.7 RESUME

Dans ce chapitre nous avons introduit une classe de modèles probabilistes qui repose sur une représentation graphique. D'une part, les réseaux bayesiens fournissent un outil de modélisation d'une loi de probabilité conjointe, d'autre part, les arbres de décision fournissent une représentation d'une loi conditionnelle. Les deux types de structures sont utilisés largement dans le cadre du raisonnement probabiliste et notamment dans le cadre de ce cours.

Bien que nous ayons limité notre discussion au cas de variables aléatoires discrètes et en nombre fini, il est possible d'étendre ces notions au cas de variables continues et en nombre dénombrable.

Dans ce chapitre nous nous sommes attachés à l'exploitation de ces modèles pour l'inférence. Le chapitre suivant traite de la construction de tels modèles à partir d'observations, c'est-à-dire l'apprentissage automatique.

Appendice 5.A: Exercice

La figure 5.A.1 reprend un réseau bayesien relatif à un problème simple de diagnostic.

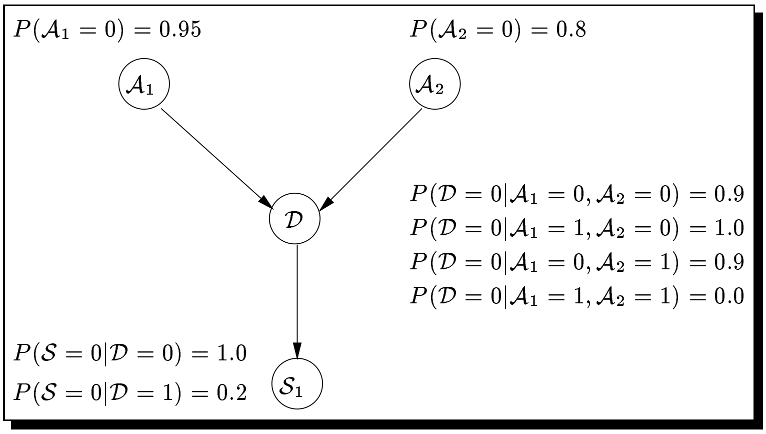


Figure 5.A.1. Réseau Bayesien pour un petit problème de diagnostic médical

Toutes les variables sont binaires (valeurs $\in \{0, 1\}$), et les tables de probabilités de ce réseau sont indiquées sur la figure. Le réseau est disponible sous format "javabayes" à la page WEB

http://www.montefiore.ulg.ac.be/∼lwh/javabayes/mini-diagnostic-medical.html

On demande de calculer par voie analytique les probabilités (marginales) $P(\mathcal{D} = 0)$ et $P(\mathcal{S}_1 = 0)$ ainsi que la loi conditionnelle $P(\mathcal{D}|\mathcal{S})$. On pourra vérifier les résultats obtenus à l'aide de l'applet "javabayes" en suivant le lien indiqué ci-dessus.

On demande ensuite de calculer les entropies conditionnelles $H(\mathcal{D}|\mathcal{A}_1)$, $H(\mathcal{D}|\mathcal{A}_2)$ et $H(\mathcal{D}|\mathcal{S}_1)$ et de vérifier les calculs à l'aide de l'applet.

Que signifie la variable $\mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{S}$ sur le réseau bayesien fourni à la page WEB indiquée ci-dessus ?

Si on devait construire un arbre de décision pour ce problème de diagnostic, quelle serait la variable testée au noeud racine ?

En vous servant de l'applet "javabayes", construisez un arbre de décision complet pour ce problème et expliquez-en la signification.

Vérifiez si $\mathcal{A}_1 \perp \mathcal{A}_2|\mathcal{D}$, $\mathcal{A}_1 \perp \mathcal{A}_2|\mathcal{S}$ et $\mathcal{A}_1 \perp \mathcal{S}|\mathcal{D}$ et expliquez la manipulation ainsi que le résultat trouvé.

Que pouvez-vous en déduire sur $\mathcal{A}_2 \perp \mathcal{S}|\mathcal{D}$?

6 APPRENTISSAGE AUTOMATIQUE DE MODÈLES GRAPHIQUES

6.1 INTRODUCTION

Dans le précédent chapitre nous avons introduit quelques modèles graphiques et les algorithmes d'inférence probabiliste associés.

Dans de nombreuses situations pratiques, on ne dispose pas a priori de connaissances suffisantes pour la spécification complète d'un modèle probabiliste. Dans certains cas, la structure est connue mais pas les paramètres (p.ex. les matrices de transition d'une chaîne de Markov, ou bien les lois de probabilités conditionnelles d'un réseau Bayesien). Dans d'autres cas, ni la structure exacte ni les paramètres ne sont connus.

Dans de telles situations il faut faire appel à l'expérimentation et à l'observation du système qu'on souhaite modéliser et exploiter ces observations afin de construire un modèle ad hoc. Les techniques qui permettent d'automatiser ce processus portent le nom générique "d'algorithmes d'apprentissage automatique", parce qu'elles visent à apprendre automatiquement un modèle d'un système à partir de l'observation de son comportement. La théorie qui permet d'étudier et de construire des algorithmes d'apprentissage automatique est essentiellement fondée sur des développements récents dans le domaine de la statistique.

Les algorithmes de codage de données font un usage intensif de modèles probabilistes et certaines techniques dites 'adaptatives' font appel à l'apprentissage automatique de ces modèles. Réciproquement, la théorie de la compression de données (codage de source) est à la base d'un principe général utilisé par bon nombre de méthodes d'apprentissage. Ce pan de l'informatique qu'est l'apprentissage automatique est donc en relation étroite avec le sujet de ce cours, et il semble utile, même dans un cours introductif comme celui-ci, de lui accorder une petite place.

Dans cette introduction nous ne ferons que décrire quelques principes généraux des méthodes utilisées pour apprendre les modèles graphiques décrits au chapitre précédent. Nous laissons l'étude détaillée de ces techniques à d'autres enseignements plus spécialisés.

Notons, pour terminer cette introduction, que les méthodes d'apprentissage automatique sont également à la base d'un nouveau domaine appelé "fouille de données" (en anglais Data Mining) qui vise à extraire des connaissances synthétiques à partir de bases de données. Etant donnée la croissance exponentielle des données informatisées, on peut se douter de l'importance pratique de ce domaine. Parmi les méthodes de data mining on retrouve les algorithmes d'apprentissage d'arbres de décision et de réseaux Bayesiens discutés ci-dessous, mais également de nombreuses autres méthodes telles que les réseaux de neurones artificiels, les techniques dites du plus proche voisin, et les méthodes de régression linéaire.

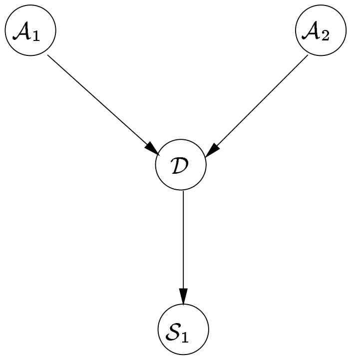
6.2 APPRENTISSAGE DE RESEAUX BAYESIENS

L'apprentissage de réseaux Bayesiens peut se présenter sous différentes formes, plus ou moins complexes, en fonction des informations dont on dispose a priori sur un problème. Nous allons discuter ces problèmes par ordre croissant de complexité, en ne détaillant la solution que pour la version la plus simple.

6.2.1 Structure du réseau complètement spécifiée et données totalement observées

Il s'agit de la version la plus simple de l'apprentissage de réseaux Bayesiens. La structure du réseau est donnée et le problème d'apprentissage consiste à déterminer à partir d'une base de données (composée d'observations d'échantillons relatifs à la distribution conjointe des variables du réseau) les tables de probabilités à associer au réseau.

Prenons comme exemple le réseau Bayesien de la figure 6.1, et supposons que nous disposions d'une base de données composée de N observations des valeurs des variables $\mathcal{A}_1, \mathcal{A}_2, \mathcal{D}, \mathcal{S}_1$ pour N cas de diagnostics résolus.



Base de données DB

No	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{D}$	$\mathcal{S}_1$
1	F	F	F	F
2	F	F	F	F
3	F	F	F	F
4	F	T	T	T
5	F	F	F	F
6	T	F	F	F
7	F	T	F	F
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
N-1	F	F	T	F
N	F	F	F	F

Figure 6.1. Réseau Bayesien et base de données pour un petit problème de diagnostic médical

L'estimation des paramètres du réseau Bayesien vise alors à déterminer les lois de probabilités $P(\mathcal{A}_1), P(\mathcal{A}_2), P(\mathcal{D}|\mathcal{A}_1, \mathcal{A}_2)$ et $P(\mathcal{S}_1|\mathcal{D})$ qui expliquent le mieux les données observées. Les variables étant binaires, il s'agit de déterminer les 8 paramètres suivants

$$\lambda_1 = P(\mathcal{A}_1 = T) \quad (6.1)$$
$$\lambda_2 = P(\mathcal{A}_2 = T) \quad (6.2)$$
$$\lambda_3 = P(\mathcal{D} = T|\mathcal{A} = TT) \quad (6.3)$$
$$\lambda_4 = P(\mathcal{D} = T|\mathcal{A} = TF) \quad (6.4)$$
$$\lambda_5 = P(\mathcal{D} = T|\mathcal{A} = FT) \quad (6.5)$$
$$\lambda_6 = P(\mathcal{D} = T|\mathcal{A} = FF) \quad (6.6)$$
$$\lambda_7 = P(\mathcal{S}_1 = T|\mathcal{D} = T) \quad (6.7)$$
$$\lambda_8 = P(\mathcal{S}_1 = T|\mathcal{D} = F) \quad (6.8)$$

les autres valeurs s'en déduisant puisque les lois de probabilités doivent sommer sur 1.

Intuitivement l'estimation de ces probabilités peut être effectuée au moyen des fréquences relatives correspondantes observées dans la base de données. Celles-ci convergent en effet vers les probabilités lorsque la taille de la base de données devient infinie (conséquence de la loi des grands nombres).

Estimation des paramètres au maximum de vraisemblance. Nous allons montrer que ces estimées (fréquences relatives) sont celles obtenues par la méthode du maximum de vraisemblance. Cela nous permettra de de dévlopper en détails cette méthode qui peut se généraliser à de nombreux autres problèmes d'apprentissage automatique (dont la solution n'est pas nécessairement aussi évidente a priori).

Désignons par $\lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7, \lambda_8)$ un vecteur de valeurs particulières pour ces paramètres, par $o_i = (A_1^i, A_2^i, D^i, S_1^i)$ la i-ème observation (i-ème ligne de la table) et par $P(o_i|\lambda)$, la probabilité associée par le réseau Bayesien instantié avec le vecteur λ à l'observation o_i.

Alors, en supposant que la base de données est composée d'échantillons indépendants tirés au moyen de la loi conjointe définie par le réseau Bayesien instantiée avec λ comme vecteur de paramètres, la probabilité d'observer la suite composée des N échantillons de la base de données DB vaut

$$P(DB|\lambda) = \prod_{i=1}^{N} P(o_i|\lambda). \tag{6.9}$$

Cette grandeur est aussi appelée la *vraisemblance de la base de données* (sous-entendu de la base de données DB sous l'hypothèse du modèle caractérisé par la vecteur de paramètres λ).

Un critère classique utilisé en estimation statistique consiste à choisir le vecteur de paramètre λ^* qui maximise cette grandeur :

$$\lambda^* = \arg\max_{\lambda} P(DB|\lambda). \tag{6.10}$$

Voyons comment se traduit ce critère du maximum de vraisemblance dans notre cas.

Décomposition en sous-problèmes indépendants. Remarquons tout d'abord que le critère du maximum de vraisemblance revient également au critère suivant

$$\lambda^* = \arg\min_{\lambda} \left(-\log P(DB|\lambda)\right). \tag{6.11}$$

D'autre part, on a

$$\begin{aligned}
-\log P(DB|\lambda) &= -\log\left(\prod_{i=1}^{N} P(o_i|\lambda)\right) && (6.12)\\
&= -\sum_{i=1}^{N} \log P(o_i|\lambda) && (6.13)\\
&= -\sum_{i=1}^{N} \log\left[P(A_1^i|\lambda)P(A_2^i|\lambda)P(D^i|A_1^i, A_2^i, \lambda)P(S_1^i|D^i, \lambda)\right] && (6.14)\\
&= -\sum_{i=1}^{N} \log P(A_1^i|\lambda) - \sum_{i=1}^{N} \log P(A_2^i|\lambda) - \sum_{i=1}^{N} \log P(D^i|A_1^i, A_2^i, \lambda)\\
&\quad -\sum_{i=1}^{N} \log P(S_1^i|D^i, \lambda). && (6.15)
\end{aligned}$$

Cette formule montre que le problème d'optimisation peut être décomposé en sous-problèmes simples. En effet, les quatre termes du membre de droite dépendent chacun d'un sous-ensemble de paramètres qui n'influence pas les autres termes. Par conséquent, le membre de gauche peut être minimisé en minimisant séparément les termes du second membre par rapport au jeu de paramètres qui leur est associé.

Estimation de probabilités au maximum de vraisemblance. Considérons, par exemple, le premier terme

$$-\sum_{i=1}^{N} \log P(A_1^i|\lambda).$$

Ce terme ne dépend en réalité que du paramètre λ_1 qui n'intervient dans aucun des trois autres termes. On a donc

$$\lambda_1^* = \arg\min_{\lambda_1 \in [0,1]} \left(-\sum_{i=1}^{N} \log P(A_1^i|\lambda_1)\right). \tag{6.16}$$

Voyons quelle est la valeur optimale de λ_1. On a

$$-\sum_{i=1}^{N} \log P(A_1^i|\lambda_1) = -N_1^T \log \lambda_1 - N_1^F \log(1-\lambda_1), \tag{6.17}$$

où N_1^T (resp. N_1^F) désigne le nombre de lignes de la base de données telles que $A_1 = T$ (resp. $A_1 = F$). En posant[1]

$$f_1 = \frac{N_1^T}{N},$$

on a donc aussi

$$\lambda_1^* = \arg \min_{\lambda_1 \in [0,1]} -[f_1 \log \lambda_1 + (1-f_1)\log(1-\lambda_1)], \tag{6.18}$$

et on montre[2] que la valeur optimale est donnée par

$$\lambda_1^* = f_1, \tag{6.19}$$

c'est-à-dire que l'estimée au maximum de vraisemblance des probabilités est donnée par les fréquences relatives.

Un raisonnement analogue donne

$$\lambda_2^* = f_2, \tag{6.20}$$

où f_2 désigne la proportion des cas de la base de données tels que $A_2 = T$.

Estimation de probabilités conditionnelles. En ce qui concerne les probabilités conditionnelles, faisons le raisonnement pour les paramètres λ_7 et λ_8. Ces paramètres n'interviennent que dans le quatrième terme de l'equation (6.15), et doivent donc être choisis de manière à minimiser celui-ci. Or on a,

$$\begin{aligned} -\sum_{i=1}^{N} \log P(S_1^i|D^i,\lambda) &= -\sum_{i=1}^{N} \log P(S_1^i|D^i,\lambda_7,\lambda_8) && (6.21)\\ &= -\sum_{i\,:D_i=T} \log P(S_1^i|D^i=T,\lambda_7) - \sum_{i\,:D_i=F} \log P(S_1^i|D^i=F,\lambda_8), && (6.22) \end{aligned}$$

ce qui veut dire qu'ici aussi on peut séparer l'estimation des paramètres λ_7 et λ_8. Par exemple, l'estimation du paramètre λ_7 revient à l'estimation de la proportion de cas tels que $S_1 = T$ dans la sous-base de données des objets tels que $D_i = T$, et, symétriquement, λ_8 est estimé par la proportion des cas tels que $S_1 = T$ dans l'autre moitié de la base de données.

Cela se généralise directement à l'estimation de probabilités conditionnelles vis-à-vis d'un nombre quelconque de variables aleátoires, en divisant la base de données en autant de parties qu'il y a de combinaisons de valeurs possibles pour les variables sur lesquelles on conditionne.

Remarques. La procédure d'estimation au maximum de vraisemblance est très simple à mettre en oeuvre. La précision des estimées des probabilités est directement proportionnelle au carré de la taille de la base de données (il faut multiplier par 4 la taille N pour diviser par deux l'écart-type des estimées).

Si la base de données est de faible taille et/ou si une variable est conditionnée sur un grand nombre de variables, il se peut que certaines des sous-bases de données soient de très faible taille (voire vides), ce qui posera des problèmes lors de l'estimation des paramètres correspondants. Il existe dans la littérature principalement deux approches possibles pour contourner ce problème (à savoir l'approche Bayesienne et l'utilisation de modèles avec un nombre réduit de paramètres), mais nous ne nous y attarderons pas dans le cadre de ce cours.

Dans ce qui précède nous avons décrit une formulation analytique directe permettant d'optimiser la vraisemblance de la base de données vis-à-vis des paramètres du modèle. Une approche alternative consisterait à utiliser un algorithme d'optimisation numérique itératif, par exemple un algorithme de descente de gradient. L'avantage de cette approche est qu'on peut la généraliser à d'autres critères d'optimisation et qu'on peut traiter des contraintes entre les valeurs des paramètres relatifs à différentes tables. Ce type d'algorithme intervient dans de nombreuses méthodes d'apprentissage automatique.

[1] f_1 désigne la proportion de cas observés pour lesquels $A_1 = T$

[2] par application de l'inégalité de Gibbs

6.2.2 Données partiellement observées

On parle de données partiellement observées (ou de données manquantes) lorsque certaines valeurs des variables n'ont pas été observées pour certains objets. Tout se passe comme si certaines des valeurs de la base de données étaient remplacées par un marqueur "?" signalant que cette valeur est inconnue.

L'approche du maximum de vraisemblance doit alors être légèrement modifiée, en supprimant dans les ensembles d'objets ceux pour lesquels au moins une des variables (conditionnée ou conditionnante) est inconnue. En supposant que les valeurs manquantes sont distribuées de façon indépendante des valeurs des variables observées (ce qui n'est pas toujours plausible), on obtient encore une procédure d'estimation non biaisée.

6.2.3 Structure inconnue et données totalement observées

Si la structure du réseau Bayesien est inconnue a priori, l'algorithme d'apprentissage a alors la responsabilité de choisir à la fois une structure et les tables de probabilités pour celle-ci. Découvrir la structure revient à identifier les relations d'indépendance conditionnelle entre les variables du problème.

L'ensemble des structures possibles sur un nombre fini de variables étant fini, ceci pourrait en principe se faire en essayant toutes les structures possibles, en déterminant pour chacune d'elles le jeu optimal de paramètres (par exemple en utilisant le critère du maximum de vraisemblance) et en retenant la structure qui maximise la vraisemblance de la base de données. Plusieurs problèmes se posent cependant :

Structures équivalentes. En général, il n'y a pas une seule structure optimale. Par exemple, si nous supposons que les données sont de type chaîne de Markov, il n'est pas possible à partir des données de définir l'orientation des arcs, puisque la structure $\mathcal{X} \rightarrow \mathcal{Y} \rightarrow \mathcal{Z}$ implique aussi $\mathcal{X} \leftarrow \mathcal{Y} \leftarrow \mathcal{Z}$.

Biais en faveur des structures les plus complexes. Le critère du maximum de vraisemblance tend à privilégier les structures les plus complexes. En effet, tout modèle qui peut être représenté à l'aide d'un réseau Bayesien donné peut aussi être représenté à l'aide de tous les réseaux dérivés de celui-ci en ajoutant des arcs (en évitant l'introduction de cycles orientés). Comme en ajoutant des arcs on augmente le nombre de paramètres du modèle, l'optimisation au sens du maximum de vraisemblance va en général pénaliser les modèles simples.

La solution à ce problème consiste essentiellement à introduire dans le critère d'évaluation un terme qui pénalise les modèles complexes, de façon à éviter ce qu'on appelle le surapprentissage. Nous reviendrons plus loin sur cette idée maîtresse en apprentissage automatique qui est aussi appelée "le rasoir d'Occam", et qui dit que dans le domaine de la modélisation il faut toujours chercher le modèle le plus simple qui est compatible avec les données observées.

Complexité de calcul. Le nombre de structures candidates croit exponentiellement avec le nombre de variables, et en pratique, dès que le nombre de variables dépasse 10, il devient impossible de les considérer de manière exhaustive[3].

La solution à ce problème consiste essentiellement à développer des algorithmes de recherche heuristique qui ne parcourent qu'une faible proportion des structures candidates. Dans ce domaine la recherche est encore très active à l'heure actuelle.

6.2.4 Variables cachées

Dans certains problèmes, la structure est connue mais certaines des variables qui interviennent dans cette structure ne peuvent pas être observées.

Le réseau Bayesien de la figure 5.9 en fournit un exemple : ici on peut facilement observer les variables relatives au phénotype (par définition) mais les variables qui représentent le génotype ne sont pas directement observables (même si les progrès récents dans le domaine de la génétique moléculaire permettent à l'heure actuelle d'observer de plus en plus d'informations relatives au génotype).

Pour un tel type de problème, l'algorithme d'apprentissage doit essentiellement deviner des valeurs plausibles pour les variables non-observées. Cela veut dire que le problème d'estimation inclut cette fois à la fois

[3]Pour un ensemble de 10 variables le nombre total de structures candidates vaut environ 10^{20} !

les paramètres (tables de probabilités conditionnelles) et les valeurs des variables cachées pour tous les objets d'apprentissage.

6.3 APPRENTISSAGE DE CHAINES DE MARKOV

L'apprentissage automatique de chaînes de Markov (cachées ou non) est évidemment un cas particulier de l'apprentissage des réseaux Bayesiens. Nous nous contentons de signaler l'existence d'algorithmes relativement efficaces qui tirent profit de la structure linéaire de ce type de réseaux, et en particulier de la propriété d'invariance temporelle qui permet de réduire très fortement le nombre de paramètres du modèle à apprendre, au prix d'une modification des algorithmes.

Nous invitons le lecteur intéressé à consulter la référence [Rab89] pour une introduction à ce sujet.

6.4 APPRENTISSAGE SUPERVISE PAR ARBRES DE DECISION

Au chapitre précédent nous avons montré comment un arbre de décision permet de représenter une loi de probabilités conditionnelles, et comment on pouvait construire automatiquement un tel arbre à partir d'une description d'un modèle probabiliste complet.

Ici nous nous intéressons à la détermination d'un tel modèle à partir d'une base de données telle que celle représentée à la figure 6.1 dont on a choisi une variable (c'est-à-dire une colonne) comme variable de sortie, et les autres comme variables d'entrée. Il s'agit d'un problème d'apprentissage *supervisé*, c'est-à-dire d'un problème où on cherche à construire un modèle entrée-sortie qui reflète le comportement d'un système à partir d'observations des variables d'entrée et de sortie de celui-ci.

Nous pouvons encore appliquer l'algorithme décrit au chapitre précédent, à condition d'estimer les lois de probabilité à partir de fréquences relatives telles qu'observées dans la base de données. Illustrons la méthode sur un exemple simple et voyons comment les notions d'entropie et de quantité d'information peuvent être utilisées en apprentissage automatique.

Problème d'apprentissage supervisé illustratif. Le dimanche matin, une personne veut planifier ses activités pour le dimanche après-midi, et une première étape consiste à évaluer, en fonction des informations dont elle dispose, si la journée est prometteuse pour jouer au tennis. Disons que la personne utilise à cette fin quatre types d'informations sur la situation météorologique

Outlook : qui décrit le type de couverture nuageuse, et peut avoir les 3 valeurs : sunny, overcast et rain.

Temperature : qui est un attribut discrétisé en trois catégories : hot, mild et cool.

Humidity : un attribut possédant deux valeurs possibles : high et normal.

Windy : un attribut de type logique (dont les valeurs possibles sont true et false).

La personne prend une décision en fonction de ces données, en classant la situation météorologique en deux classes possibles P ou N, P voulant dire que les informations disponibles indiquent que la situation est propice pour jouer au tennis, N qu'elle est non-propice.

Nous ne connaissons pas le raisonnement que fait la personne pour prendre sa décision, mais nous allons supposer qu'elle est cohérente, c'est-à-dire qu'en présence des mêmes données elle prend toujours la même décision. La décision est donc une fonction (au sens mathématique du terme) des quatre variables Outlook, Temperature, Humidity, Windy.

D'autre part, la personne s'étant déclarée incapable d'expliquer son raisonnement, nous lui avons demandé de nous indiquer, pendant quatorze semaines successives, les valeurs des variables et la décision qu'elle a prise. Cette information est réprésentée à la table 6.1. Est-il possible, à partir de ces données, d'inférer le raisonnement fait par la personne, et de prédire ses décisions futures ? En principe non, puisque les combinaisons des valeurs possibles des variables d'entrée sont au nombre de 36 ($3 \times 3 \times 2 \times 2$) et que nous n'en avons observé que 14. Nous devons donc nous contenter ici de faire une *hypothèse* sur le raisonnement de la personne, aussi *plausible* que possible au vu des informations dont nous disposons. Ce type de problème est ce qu'on appelle aussi un problème d'inférence inductive : *à partir d'un ensemble d'observations de cas particuliers, inférer une règle générale qui est compatible avec ces cas particuliers, et suffisamment plausible pour être utilisée pour prédire d'autres cas, non encore observés.*

Table 6.1. Ensemble d'apprentissage

No.	Décision	Attributs			
		Outlook	Temperature	Humidity	Windy
1	N	sunny	hot	high	false
2	N	sunny	hot	high	true
3	P	overcast	hot	high	false
4	P	rain	mild	high	false
5	P	rain	cool	normal	false
6	N	rain	cool	normal	true
7	P	overcast	cool	normal	true
8	N	sunny	mild	high	false
9	P	sunny	cool	normal	false
10	P	rain	mild	normal	false
11	P	sunny	mild	normal	true
12	P	overcast	mild	high	true
13	P	overcast	hot	normal	false
14	N	rain	mild	high	true

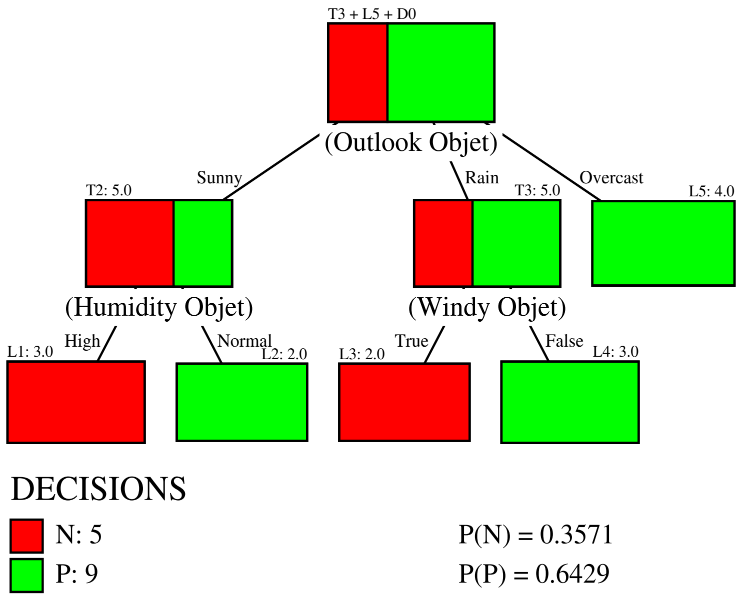


Figure 6.2. Arbre de décision construit par la méthode ID3

La figure 6.2 représente une telle règle sous la forme d'un arbre de décision. Une situation météorologique est classée par cet arbre en partant de la racine de l'arbre (le noeud situé le plus haut) en propageant la situation le long des arcs choisis en fonction des informations disponibles, jusqu'à aboutir en une feuille qui indique si la situation est propice ou non. On peut se convaincre que l'arbre de la figure 6.2 est bien cohérent avec les informations fournies à la table 6.1. D'autre part, il est relativement simple (d'ailleurs il ne fait intervenir que trois parmi les quatre informations) et, si on y réfléchit, exprime un critère tout à fait plausible.

Algorithme ID3. Comment construire un tel arbre automatiquement à partir des données dont nous disposons ? Nous allons décrire la méthode ID3, dont l'objectif est de construire un arbre de taille minimale compatible avec les données, et dont le principe repose sur l'utilisation de la notion de quantité d'information introduite dans ce chapitre.

La méthode construit l'arbre en partant de la racine, en "développant" progressivement les noeuds de test. Elle se sert de la base de données pour évaluer les probabilités des valeurs des variables d'entrée et des décisions prises par notre ami. Pour illustrer le principe de la méthode montrons comment le développement de la racine est effectué.

1. L'incertitude associée à la décision N ou P au noeud racine (c'est-à-dire sans aucune information sur la météo) peut être évaluée à partir des fréquences relatives des décisions P et N sur l'ensemble des lignes de la table 6.1. On obtient P(décision=N)= 5/14 et P(décision=P)=9/14, ce qui donne une entropie de 0.940 bit.

2. Evaluons ensuite la quantité d'information apportée par l'observation de chacune des variables météo en nous servant de la table. Pour chaque variable, il suffit de diviser la table en sous-ensembles correspondant aux valeurs possibles de la variable, de calculer les entropies conditionnelles moyennes et de calculer la quantité d'information au moyen de la formule

$$I(D;V) = H(D) - H(D|V) \tag{6.23}$$

   où D représente la décision et V la variable utilisée. On obtient les partages suivants (on peut vérifier à titre d'exercice les quantités d'informations affichées)

Attribut :	Partition	$I(D;V)$
Outlook :	(rain 14 10 6 5 4) (overcast 13 12 7 3) (sunny 11 9 8 2 1)	0.247
Temperature :	(cool 9 7 6 5) (mild 14 12 11 10 8 4) (hot 13 3 2 1)	0.029
Humidity :	(normal 13 11 10 9 7 6 5) (high 14 12 8 4 3 2 1)	0.152
Windy :	(true 14 12 11 7 6 2) (false 13 10 9 8 5 4 3 1)	0.048

3. La partition optimale est donc celle qui correspond à l'attribut "Outlook", qui est sélectionné pour le développement de la racine. On voit (cf. figure 6.2) que l'ensemble des observations correspondant à la valeur Outlook=Overcast correspondent toutes les 4 à une décision D=P : ce noeud ne doit donc pas être développé. En ce qui concerne les deux autres valeurs on voit qu'il subiste une incertitude sur la décision; il faut donc continuer le développement de l'arbre.

4. On peut vérifier que l'application de la même procédure (récursivement) aux sous-ensembles d'observations relatives aux deux noeuds en question conduit à sélectionner respectivement Humidity et Windy comme variables de test, et que les quatre sous-ensembles correspondants ont une entropie nulle (décisions identiques). La construction de l'arbre de décision est donc terminée à ce stade.

La procédure ID3 que nous venons d'illustrer permet de construire un arbre de décision de façon automatique, à partir d'une base de données quelconque. Son objectif est de produire un arbre aussi simple que possible qui soit compatible avec les observations disponibles. L'algorithme est heuristique, dans la mesure où il n'offre aucune garantie sur l'optimalité de son résultat. Mais il est très efficace, ce qui permet de l'appliquer à des problèmes de très grande taille (millions d'observations, milliers de variables), et sa démarche est intuitivement plausible.

De nombreuses variantes plus ou moins élaborées de cette méthode sont actuellement utilisées dans de nombreux logiciels d'analyse de données et de Data Mining. Parmi les nombreuses applications pratiques de cette méthode, citons l'apprentissage de stratégies de jeu d'échec, le diagnostic médical, le contrôle/commande de systèmes complexes, l'allocation de crédits bancaires, la prédiction de faillites...

Une version de cette méthode, progammée en langage JAVA, est accessible au travers de la page WEB
http://www.montefiore.ulg.ac.be/∼lwh/Gtdidt/.

6.4.1 Application au problème de pesées

L'algorithme que nous venons de décrire peut être appliqué au problème de pesées donné à la fin du chapitre 2. Il suffit pour cela construire une base de données comprenant la description des 24 solutions possibles à ce problème et de choisir comme ensemble de question candidates l'ensemble des pesées possibles en chaque noeud

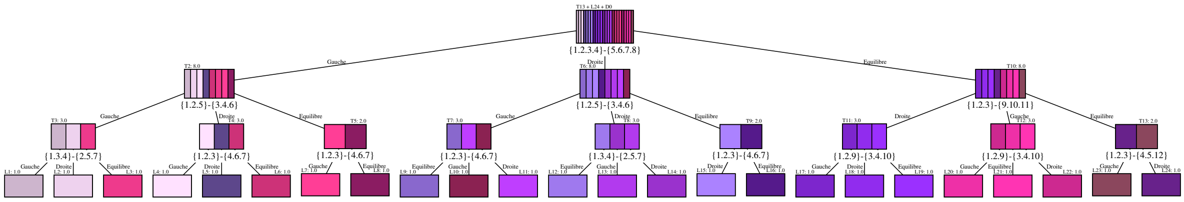


Figure 6.3. Arbre de pesées

de l'arbre (attention, le nombre de questions candidates est extrêmement élevé dans ce problème). Chacune de ces pesées candidates définit (implicitement) une variable aléatoire discrète ayant trois valeurs possibles à savoir : gauche, équilibre, droite. La variable de sortie de ce problème est une variable discrète ayant 24 valeurs possibles correspondant respectivement aux 24 solutions du problème.

La figure 6.3 représente la solution du problème sous la forme d'un arbre de décision. Les 24 noeuds terminaux correspondent aux 24 solutions du problème. Nous demandons au lecteur de trouver pour quelques noeuds terminaux la réponse à la question posée.

II LES GRANDS THEOREMES DE LA THEORIE DE L'INFORMATION

7 MOTIVATION

Cette seconde partie des notes est constituée d'une série de chapitres qui couvrent les principaux résultats théoriques de la théorie de l'information relatifs à la modélisation et à l'utilisation des sources d'information et des canaux de communication. Les théorèmes de Shannon énoncent les limites ultimes en matière de compression de données et définissent les débits maximaux atteignables pour la communication fiable à l'aide de canaux de communication bruités.

Au chapitre 8 nous nous intéresserons à la modélisation des sources discrètes au moyen de processus aléatoires (principalement les processus markoviens) et au codage de source, c'est-à-dire à la compression de données. Puis, au chapitre 9, nous étudierons quelques modèles simples de canaux discrets et établirons les théorèmes fondamentaux relatifs à la transmission sans erreurs (codes correcteurs d'erreurs) et au chapitre 10 nous nous consacrerons à l'étude de canaux continus, mais nous nous limiterons au cas particulier du canal à bruit additif gaussien qui est un modèle d'intérêt pratique important. Enfin, le chapitre 11 fournit une brêve introduction à la théorie de la distorsion, qui étudie la compression de données irréversible et le compromis entre débits de communication et taux d'erreurs.

Ces résultats reposent essentiellement sur les propriétés de stationnarité et d'ergodicité des processus aléatoires et la loi des grands nombres rappelée en appendice au chapitre 8.

En anticipant sur les chapitres suivants, nous allons illustrer sur un exemple imaginaire comment la mesure d'information probabiliste, qui vient d'être introduite, est en relation directe avec les ressources physiques nécessaires pour le traitement de l'information.

7.1 AU RESTAURANT "CHEZ BILL"

7.1.1 Problème de communication efficace de recettes

Penchons-nous sur un problème pratique relatif à la grande cuisine. Il s'agit d'un problème de communication de plats dans le restaurant "Chez Bill" situé dans la proche banlieue de Seattle.

Le chef Bill et son garçon de salle Claude doivent communiquer au sujet du choix par les clients de plats de cuisine. Tous deux disposent du même catalogue de plats de cuisine (p.ex. sous la forme d'un livre de recettes stocké sur un certain nombre de disquettes). Claude prend les commandes en salle (de l'ordre de n par aller-retour), et doit en informer le chef Bill. Le temps presse, comme souvent dans ces circonstances, et il s'agit de communiquer de manière fiable et efficace les n plats à préparer.

Table 7.1. Recettes de cuisine du restaurant "Chez Bill"

Plat	Probabilité	Info = $-\log P$	$-P\log P$
Moules marinières - frites	0.1250	3	$\frac{3}{8}$
Moules provencale - frites	0.1250	3	$\frac{3}{8}$
Filet américain - frites	0.2500	2	$\frac{2}{4}$
Glace enfants	0.1250	3	$\frac{3}{8}$
Glace adultes	0.1250	3	$\frac{3}{8}$
Tarte aux pommes	0.0625	4	$\frac{4}{16}$
Tarte au riz	0.0625	4	$\frac{4}{16}$
Gâteau au chocolat	0.1250	3	$\frac{3}{8}$
	Total : 1.0000		Entropie : $\frac{23}{8}$

On supposera que le nombre de recettes possibles est fini, et étant féru d'informatique, le chef a décidé que le nombre total de plats différents servis dans son restaurant serait une puissance de 2, disons 2^m. De cette façon, s'est-il dit, il suffira de m bits d'information pour communiquer chaque plat et il a déjà organisé son fichier de recettes sous la forme d'un arbre de profondeur m pour minimiser le temps d'accès.

Mais, le garçon de salle qui avait eu vent des travaux de Claude Shannon sur la théorie de l'information, et a consacré une partie (non-négligeable) de son temps libre à comprendre de quoi il retournait, s'est dit qu'il y avait là une opportunité d'épater son chef. Il lui parla de ses lectures sur les σ-algèbres, les fonctions mesurables, d'additivité et de continuité de l'information, et des trois grands théorèmes de Shannon, en insistant surtout sur le premier. Le chef n'y comprit pas grand-chose, mais décida avec sagesse et ouverture d'esprit de faire confiance au garçon de salle. En effet, celui-ci avait déjà optimisé avec grand succès le prix des consommations, en faisant appel à la théorie des portefeuilles. Il donna donc le feu vert pour une réorganisation entièrement nouvelle du schéma de communication et de stockage des plats.

Comme le garçon de salle aussi bien que le chef connaissaient bien les habitudes de choix des clients du restaurant suite à l'enquête précédente relative aux portefeuilles, ils consultèrent une table associant à chaque recette la probabilité a priori d'être choisie par un client quelconque, représentée à la table 7.1, où nous avons supposé que $m = 3$ pour des raisons évidentes (en fait, il semble d'après la carte que le chef se passionne beaucoup moins pour la cuisine que pour l'informatique). Nous supposerons aussi que la carte de boissons comprend 2^{10} boissons, mais que la clientèle de "Chez Bill" ne boit que du Coca-Cola.

Voyant cette table, le garçon de salle s'écria : "Chef, il y a moyen de coder les plats en utilisant seulement 23 huitièmes de bits par plat, soit un huitième de bit de mieux que votre schéma très naïf !". Le chef répondit : "C'est pas terrible, mais mieux vaut ça que rien", se disant que Claude allait avoir du mal avec des huitièmes de bits comme adresses sur son ordinateur. Cependant, le garçon savait que ça allait marcher (car les probabilités étaient diadiques) et il montra au chef l'arbre de la figure 7.1.

Cette fois-ci le chef marqua son impatience. En effet, la profondeur maximale de cet arbre était supérieure à son arbre équilibré et il demanda au garçon de s'expliquer. Celui-ci lui fit comprendre que ce qui comptait ici, en tout cas d'après Shannon, c'était non pas le temps maximal mais le temps moyen. Il lui expliqua que sur un nombre de commandes n suffisamment grand, la loi des grand nombres allait lui permettre d'économiser avec quasi certitude $\frac{n}{8}$ bits d'adresse au total et le chef ferait les mêmes économies en temps d'accès.

L'expérience lui donna raison, et bien que l'économie fut substantielle, le chef lui demanda s'il n'était pas possible d'améliorer encore le système. Là-dessus, le garçon répondit : "Non chef ! C'est interdit par le premier théorème de Shannon, car chacun des bits de mon code apporte exactement 1 bit d'information. D'ailleurs dans l'arbre de mon code on voit bien que chacune des deux possibilités codées à chaque niveau est équiprobable."

7.1.2 L'âge d'or

Fort de cette première expérience, Bill décida d'enrichir sa carte en ajoutant plus de plats, et en offrant également des entrées. La clientèle devint de plus en plus nombreuse et gastronome, et finit par consommer les nombreux vins et bières spéciales proposés chez Bill.

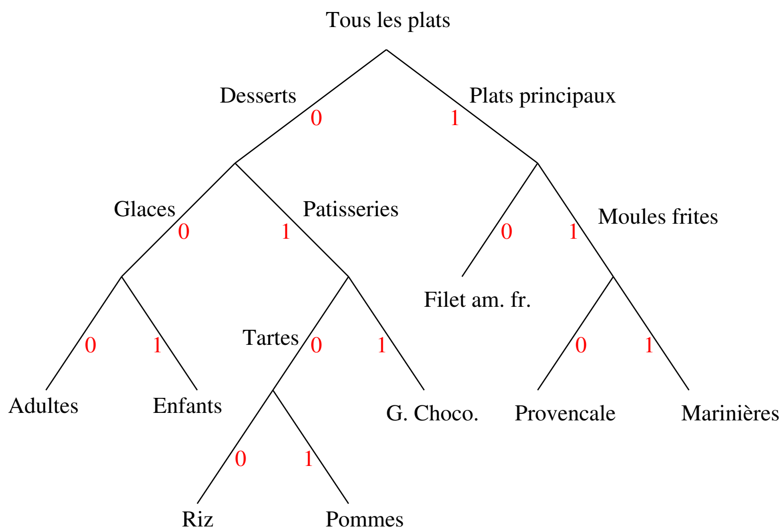


Figure 7.1. Organisation optimale des recettes de cuisine

Claude avait mis au point un programme informatique qui tous les soirs mettait à jour les statistiques de choix de plats et boissons et qui adaptait le code utilisé pour le lendemain. Au fur et à mesure de l'augmentation du nombre de plats il se rendit compte que les choix des entrées, plat principaux, desserts et boissons des personnes sur une même table étaient fortement corrélés et décida d'encore optimiser son système en prenant d'emblée les commandes de tous les plats pour une même table. Il fit appel successivement à différentes techniques de codage inventées par des amis cuisiniers, eux aussi versés dans l'informatique (Huffman, Gallager et plus tard Rissanen et les frères Lempel et Ziv).

Il réussit ainsi à économiser de plus en plus de bits dans son code. En même temps, le chef allait aussi de plus en plus vite pour faire ses plats car il avait mis au point des instruments spéciaux qui lui permettaient de réaliser très rapidement les recettes les plus fréquemment commandées et il s'était construit un frigo spécial à structure hiérarchique qui permettait un accès rapide aux ingrédients les plus souvent utilisés. D'ailleurs Claude, qui était à l'origine de ces idées breveta la première sous le sigle *RISC (Reduced Instrument Set Cooking)* (la seconde était déjà connue sous le nom de mémoire cash hiérarchisée). Il paraît même que c'est un ami de Bill qui appliqua l'idée de l'architecture RISC aux processeurs informatiques, ce qui permit des progrès considérables dans ce domaine, notamment en diminuant considérablement la complexité des puces et surtout la taille et le temps de traitement des programmes.

Enfin, Claude appliqua aussi les mêmes idées au stockage et à la commande des ingrédients (dont les distributions étaient loin d'être uniformes), à la disposition des tables dans la salle à manger, et même le menu était organisé sous la forme d'un arbre binaire, ce qui permettait aux clients de choisir plus rapidement leur commande et d'encoder directement leur choix.

Epilogue : bruit de cuisine. Tout se passa bien pendant les deux premières années qui suivirent la mise en service du nouveau système. Claude avait de moins en moins de travail en salle car maintentant les clients faisaient directement leurs choix à l'aide d'un questionnaire optimal connecté au site WEB de la cuisine.

Mais, au fur et à mesure que le restaurant gagnait en succès, il se trouva que de plus en plus souvent il y eut des erreurs dans les ingrédients. En effet, chaque fois qu'une commande arrivait après du serveur de la cuisine, celui-ci énonçait à voie haute le nom des plats et des ingrédients associés, et de plus en plus souvent il y eut des erreurs d'interprétation de la part des cuisiniers.

Claude diagnostiqua que le problème venait du bruit de cuisine qui, suite à l'augmentation du nombre de cuisiniers, entravait maintenant la fiabilité de la communication. Il y vit à nouveau une opportunité d'épater son chef, en faisant appel cette fois-ci au second théorème de Shannon. Celui-ci lui promettait en effet la possibilité de rendre la probabilité d'erreurs arbitrairement faible, sans pour autant devoir augmenter de façon inconsidérée la longueur de son code et grâce aux récents turbo-codes inventés par des cuisiniers bretons il allait quasiment pouvoir approcher cette limite de Shannon. Mais ceci est une autre histoire…

7.2 EXÉGÈSE

Cet exemple, bien que tiré par les cheveux, suggère en quoi la mesure d'information probabiliste est une mesure naturelle du point de vue de la conception des systèmes de traitement de l'information : elle permet de quantifier a priori les ressources nécessaires pour le stockage et la communication efficace de messages. L'histoire met également en évidence la très grande portée pratique des idées de Shannon dans le domaine de l'informatique, car une fois qu'on sait comment stocker et communiquer efficacement de l'information on peut appliquer ces idées à la conception de jeux d'instruction et de langages de progammation appropriés, et à la mise au point de systèmes de calcul physiques performants.

Si on en croit Claude, *l'entropie* ($\frac{23}{8}$) de la distribution de probabilités associée au choix de plats de cuisine mesure le nombre minimal de bits nécessaires en moyenne pour coder ces choix. Nous verrons que cette limite, qui dans le cas de notre exemple est atteinte par le code de Claude, peut toujours être approchée en pratique, du moins si on se donne la peine de coder l'information de façon appropriée.

Nous verrons également qu'il y a une grandeur similaire qui permet de caractériser le débit maximal d'information possible sur un canal de communications, en fonction du débit physique et du niveau de bruit de ce canal.

Ces deux résultats constituent le coeur de la théorie de l'information. Du point de vue de l'ingénieur on doit leur associer, comme nous le verrons, les techniques de codage de données qui visent à atteindre ces limites en pratique. Mais avant cela, il nous faut définir de manière plus précise notre vocabulaire et apprendre à l'utiliser correctement. C'est l'objet de la suite de cette partie du cours.

Figure 7.2. Claude Shannon

8 SOURCES DISCRETES

8.1 INTRODUCTION

Dans ce qui précède, nous avons introduit les notions mathématiques principales qui seront utilisées tout au long de ce cours, en essayant de leur donner une interprétation intuitive. Nous avons vu au chapitre 4, en particulier, que l'entropie d'une source émettant des symboles indépendants et distribués selon une même loi de probabilité (source stationnaire sans mémoire) utilisant un alphabet Q-aire était au maximum égale à $\log Q$. Si les symboles ne sont pas équiprobables on peut donc dire que l'alphabet est mal utilisé dans la mesure où l'information fournie par la source est inférieure au maximum théoriquement atteignable. Nous verrons dans ce chapitre que cette quantité d'information est encore réduite si les symboles successifs ne sont pas indépendants, c'est-à-dire si la source a une mémoire. On peut le comprendre intuitivement, puisque dans ce cas la connaissance d'une partie des symboles émis par la source réduit potentiellement l'entropie des symboles non encore observés.

Nous examinerons dans ce chapitre le problème du codage de source, c'est-à-dire celui de transformer le message engendré par une source redondante (dont les symboles successifs ne sont pas équiprobables et/ou ne sont pas indépendants, ce qui implique que son entropie est inférieure à la capacité $\log Q$) en un message dont les symboles sont équiprobables et indépendants. La transformation sera supposée réversible, en ce sens que le message original pourra être exactement reconstitué. Nous nous plaçons donc dans le cadre du codage *sans perte d'information*, et établirons dans ce contexte le *premier théorème de Shannon*, qui affirme que la longueur moyenne d'un code peut approcher d'aussi près qu'on le veut la limite $\frac{H(S)}{\log r}$ où $H(S)$ désigne l'entropie de la source et r la taille de l'alphabet utilisé pour le code. $H(S)$ peut donc bien s'interpréter comme le nombre de bits nécessaires pour coder de façon binaire et optimale les messages émis par la source S.

Nous voyons donc ce qui est théoriquement possible dans le domaine de la compression de données si on impose la condition de *réversibilité*, ce qui est souvent le souhait dans un monde purement digital; au chapitre 13 nous mettrons ces résultats en perspective avec les possibilités offertes par d'autres méthodes de compression de données qui n'imposent pas nécessairement la condition de réversibilité (compression d'images, sons, ...).

Ce chapitre est organisé comme suit. Nous commençons tout d'abord par discuter des sources de Markov : il s'agit d'un modèle de source avec mémoire qui est à la fois suffisamment général pour couvrir de nombreuses applications pratiques, et suffisamment simple pour permettre le traitement mathématique que nous visons. Ensuite, nous étudierons successivement les propriétés des messages émis par les sources sans mémoire ou par les sources de Markov, en montrant que ceux-ci peuvent se décomposer en deux ensembles disjoints : les *messages typiques* pour lesquels les symboles apparaissent avec une fréquence relative proche de leur probabilité, et les autres messages. Nous verrons que le nombre de messages typiques est limité par l'entropie de la source, alors que la probabilité totale de tous les messages typiques est voisine de 1. C'est cette propriété qui est alors à la base de la possibilité de compression de données par codage, comme nous le verrons. Enfin, nous terminerons cette introduction au codage de source par la description de l'algorithme de *Huffman*.

8.2 SOURCES DE MARKOV

Nous allons dans la suite utiliser comme exemples deux sources réelles et plusieurs modèles de source de chacune d'elles qui seront désignés dans la suite par $S_1, S_2, \ldots S_i$.

8.2.1 Sources réelles

La télécopie. Il s'agit d'une source binaire ($q = 2$) obtenue en balayant ligne par ligne, à vitesse constante, un document graphique comportant des parties noires sur fond blanc, en échantillonnant à une fréquence appropriée le signal ainsi obtenu et en associant les symboles '0'et '1' aux couleurs blanc et noir, respectivement, aux instants d'échantillonnage.

La télégraphie. Il s'agit d'une source à 27 symboles (correspondant aux 26 lettres de l'alphabet latin plus l'espace), correspondant à des textes écrits en français (ou en toute autre langue utilisant cet alphabet) et en négligeant, pour simplifier, les symboles de ponctuation et les accents. Ce type de source est également important puisqu'une bonne partie des données à transmettre ou à stocker sont des textes de ce type.

Nous mentionnerons occasionnellement des variantes de ce type de source, en faisant référence à des textes correspondant à des codes sources de programmes écrits dans un certain langage de programmation.

8.2.2 Modèles de sources

Nous allons préciser ci-dessous 4 modèles différents qui correspondent respectivement à un modèle sans mémoire et avec mémoire pour les deux exemples de sources réelles indiquées ci-dessus. Nous utiliserons quelques fois ces sources à titre illustratif et dans un certain nombre d'exercices. Notons d'emblée que l'extension d'ordre k d'une de ces sources S_i sera dans la suite désignée par le symbole S_i^k. Notre objectif ici est de mettre en évidence le fait que la modélisation des sources est un art en soi, qui repose sur un compromis simplicité/réalisme.

Modèles télécopie.

Source S_1. Il s'agit d'un modèle sans mémoire pour la télécopie. La source S_1 est dès lors complètement caractérisée par les probabilités des symboles '0'et '1', qui pourraient par exemple être déterminées par une analyse statistique d'un nombre suffisant et représentatif de documents et en comptant la fréquence relative des deux symboles. Nous noterons leurs probabilités par p_0 et p_1 et nous considérerons la source définie par les valeurs $p_0 = 0.9$ et $p_1 = 0.1$.

L'entropie de cette source est $H(S_1) = 0.469$ Shannon.

L'extension d'ordre 2 de cette source est notée S_1^2. Elle utilise un alphabet quaternaire ($q = 4$) dont les symboles correspondent aux paires '00', '01', '10', et '11' et dont les probabilités sont obtenues (hypothèse d'indépendance) $p_{00} = 0.81$, $p_{01} = 0.09$, $p_{10} = 0.09$ et $p_{11} = 0.01$.

L'entropie de cette source est $H(S_1^2) = 0.938$ Shannon, c'est-à-dire $2H(S_1)$. On peut remarquer que même si la loi de probabilité de la source S_1 correspond bien à la statistique observée, cela n'implique en aucune manière que la loi de probabilité de son extension d'ordre deux corresponde à la statistique des paires de symboles observés. En d'autres termes, si on mesure la probabilité d'apparition de paires de symboles '00", '01', '10', '00' on ne retrouve pas nécessairement les produits des probabilités p_0p_0 et p_0p_1,. . .

Source S_2. Il s'agit d'une source avec mémoire, qui serait obtenue si on souhaite que la statistique des paires de symboles soit également respectée. Plus précisément, nous supposons toujours que la source vérifie les statistiques relatives à l'émission de symboles isolés p_0 et p_1, et que de plus elle respecte les statistiques relatives aux

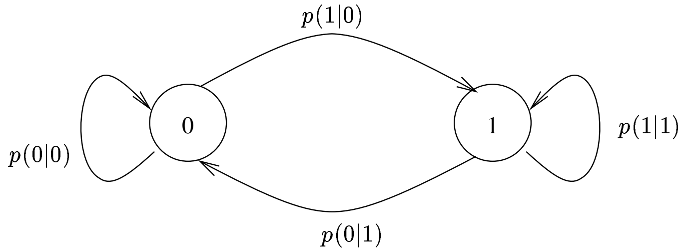


Figure 8.1. Source S_2 : chaîne de Markov à deux états

probabilités conditionnelles $p(j|i)$ (on les notera également $p(i \to j)$, ou encore simplement p_{ij}) qui décrivent la probabilité d'observer le symbole j sachant que le précédent est un i.

Plus exactement nous faisons l'hypothèse de Markov (d'ordre 1) qui postule que dans une séquence de symboles $s_1, s_2, \ldots, s_n$, on a $p(s_n|s_{n-1}, s_{n-2}, \ldots, s_1) = p(s_n|s_{n-1})$. Autrement dit, il suffit de connaître le symbole précédent pour déterminer la loi de probabilité du symbole courant.

Nous verrons à la section 8.5.2 que, tout comme la source S_1, S_2 est une source stationnaire et même ergodique.

Nous verrons ci-dessous que pour une source (ou une chaîne) de Markov, les probabilités conditionnelles $p(i|j)$ suffisent pour caractériser complètement la source, et les probabilités stationnaires $p(i)$ se déduisent de cette information.

Pour le moment, supposons que la source S_2 soit caractérisée par les probabilités conditionnelles $P(s_n|s_{n-1})$ suivantes : $p(0|0) = 0.93$, $p(1|0) = 0.07$, $p(0|1) = 0.63$ et $p(1|1) = 0.37$. Notons que cette distribution est plausible, dans la mesure où nous avons augmenté la probabilité d'observer deux symboles à la suite identiques par rapport à la source S_1 (cf. extension de celle-ci).

Nous verrons plus loin que les probabilités stationnaires de cette source sont encore $p_0 = 0.9$ et $p_1 = 0.1$, c'est-à-dire compatibles avec les statistiques supposées obtenues plus haut.

Nous verrons que l'entropie (par symbole) de cette source vaut $H(S_2) = 0.424$ Shannon; on a donc bien une réduction d'entropie par rapport à la source sans mémoire.

Modèles télégraphie.

Source S_3**.** Le modèle suppose à nouveau l'indépendance entre lettres successives, mais suppose que les lettres élémentaires ne sont pas équiprobables. Leur probabilité pourrait être déterminée par comptage statistique sur un certain nombre de documents. On trouverait sans doute que l'espace et la lettre E sont nettement plus probables que les autres lettres de l'alphabet.

Source S_4**.** En supposant que les lettres obéissent aux contraintes grammaticales de la langue française, et en supposant une certaine probabilité d'erreurs liée à la possibilité de faire des fautes ou des erreurs de frappe, on obtiendrait un modèle nettement plus réaliste, mais d'une complexité telle que son utilisation en pratique serait hautement compromise.

Signalons cependant que Markov était linguiste, et que c'est en essayant de développer des modèles linguistiques qu'il a développé les modèles qui portent son nom. Notons également que les modèles markoviens jouent un rôle important dans un grand nombre de domaines, tels que le traitement d'images, la reconnaissance de la parole, et plus généralement, la modélisation de processus aléatoires physiques.

8.3 DEBITS D'INFORMATION ET EXTENSIONS D'UNE SOURCE

Les grandeurs introduites pour mesurer l'information ont été définies *par symbole* (resp. émis ou reçu), en précisant sur le schéma de Shannon où ces symboles sont observés.

Lorsque la source émet des symboles avec un débit moyen constant en fonction du temps et que le reste de la chaîne suit (i.e. fonctionne en temps réel), ce qui est le cas le plus fréquent en communications, on peut considérer des *débits d'information* (ou d'entropie) : produit de l'entropie par symbole par la fréquence moyenne d'occurrence des symboles.

Nous nous intéresserons plus loin dans ce cours à des conversions d'alphabet, où il s'agira de remplacer des symboles par d'autres symboles qui sont obtenus au moyen d'un code à décodage unique, c'est-à-dire tel que l'on puisse retrouver de manière univoque les symboles de départ à partir de ceux du nouvel alphabet. En particulier, une conversion d'alphabet que nous utiliserons fréquemment consiste à remplacer une source par sa $k-$ème extension. Etant donné une source S engendrant des messages dont les symboles appartiennent à une alphabet Q-aire, sa $k-$ème extension, notée S^k, consiste à grouper ses symboles par blocs successifs de k symboles, chaque bloc étant interprété comme un symbole élémentaire d'un nouvel alphabet de taille Q^k. Si les symboles de la source initiale sont indépendants, l'entropie de la source S^k est multipliée par k. Comme la fréquence de cette nouvelle source est divisée par k, il s'ensuit que le débit d'information ne change pas.

Nous verrons, au chapitre suivant que la généralisation de la notion d'entropie aux sources avec mémoire (qui émettent des symboles successifs dépendants) est justement basée sur l'entropie des sources étendues, d'ordre $k \to +\infty$.

8.4 LA PROPRIETE D'EQUIPARTITION ASYMPTOTIQUE

Considérons, pour nous convaincre de la possibilité de codage efficace, tout d'abord l'exemple simple d'une source stationnaire et sans mémoire, et plus précisément les extensions d'ordre n d'une telle source.

Nous allons voir que l'ensemble de messages émis par une telle source peut se structurer en deux sous-ensembles : l'un qui comporte les messages dits *typiques*, qui possèdent essentiellement la caractéristique que l'on y retrouve les symboles élémentaires en fréquences proches de leurs probablités a priori (nous préciserons dans quel sens); l'autre composé du complémentaire de cet ensemble.

Nous verrons ensuite que le nombre de message typiques de longueur n est au plus de $2^{n(H+\epsilon)}$, et que ces messages sont approximativement équiprobables, de probabilité $2^{-n(H+\epsilon)}$. Par conséquent, l'ensemble des messages *atypiques* est de probabilité quasi nulle. Nous verrons que les égalités pourront être approchées d'aussi près que souhaité, pour autant qu'on choisisse de recourir à des extensions d'ordre n suffisamment élévé.

Partant de cette propriété *d'équipartition asymptotique* nous pourrons alors développer un code binaire très simple et dont la longueur moyenne par symbole de la source de départ approche $H(S)$ d'aussi près que souhaité.

8.4.1 Théorème AEP

Dans cette section nous faisons appel à la loi des grands nombres. Nous suggérons de commencer par la lecture de l'appendice 8.B situé à la fin de ce chapitre qui fournit les rappels et notations en la matière.

Le théorème AEP (de l'anglais *asymptotic equipartition property*) est formulé comme suit :

Si les v.a. discrètes $\mathcal{X}_1, \mathcal{X}_2, \ldots$ sont indépendantes et distribuées identiquement selon la loi $P(\mathcal{X})$, alors

$$-\frac{1}{n}\log P(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n) \xrightarrow{P} H(\mathcal{X}), \tag{8.1}$$

où la notation $\xrightarrow{P}$ indique une convergence en probabilité (voir appendices).

En effet, remarquons tout d'abord que

$$-\frac{1}{n}\log P(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n) = -\frac{1}{n}\sum_{i=1}^{n}\log P(\mathcal{X}_i), \tag{8.2}$$

et nous avons donc bien affaire à une somme de v.a. indépendantes (de plus identiquement distribuées). La loi des grands nombres s'applique et

$$-\frac{1}{n}\sum_{i=1}^{n}\log P(\mathcal{X}_i) \xrightarrow{P} -E\{\log P(\mathcal{X})\} = H(\mathcal{X}),$$

ce qui démontre le théorème. □

Interprétation. Désignons par $X_1, \ldots, X_{|\mathcal{X}|}$ les $|\mathcal{X}|$ différentes valeurs que chacune des v.a. $\mathcal{X}_i$ peut prendre. Pour un message quelconque de longueur n émis par la source, le membre de droite de l'équation (8.2) peut alors se décomposer en une somme

$$-\frac{1}{n}\sum_{i=1}^{n}\log P(\mathcal{X}_i) = -\frac{1}{n}\sum_{j=1}^{|\mathcal{X}|} n_j \log P(X_j) = -\sum_{j=1}^{|\mathcal{X}|}\frac{n_j}{n}\log P(X_j), \tag{8.3}$$

où nous comptabilisons par n_j le nombre de fois que la j-ème valeur de $\mathcal{X}$ apparaît dans le message.

La signification du théorème est donc qu'avec une presque certitude le membre de droite converge vers

$$H(\mathcal{X}) = -\sum_{j=1}^{|\mathcal{X}|} P(X_j)\log P(X_j),$$

ce qui n'est qu'une simple conséquence du fait que les $\frac{n_j}{n} \stackrel{P}{\to} P(X_j)$ étant donné la loi des grands nombres.

8.4.2 Ensemble de messages typiques $A_\epsilon^{(n)}$

Désignons par $\mathcal{X}^n = \mathcal{X} \times \cdots \times \mathcal{X}$ le produit cartésien de longueur n, identique à l'ensemble de tous les messages possibles de longueur n.

L'ensemble typique $A_\epsilon^{(n)}$ par rapport à $P(\mathcal{X})$ est défini comme l'ensemble des réalisations $(X_1, X_2, \ldots, X_n)$ de $(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n) \in \mathcal{X}^n$ qui satisfont à la double inégalité suivante :

$$2^{-n(H(\mathcal{X})+\epsilon)} \leq P(X_1, X_2, \ldots X_n) \leq 2^{-n(H(\mathcal{X})-\epsilon)}. \tag{8.4}$$

On peut montrer, par application du théorème AEP, que cet ensemble satisfait aux propriétés suivantes :

1. $(X_1, X_2, \ldots X_n) \in A_\epsilon^{(n)} \Rightarrow H(\mathcal{X}) - \epsilon \leq -\frac{1}{n}\log P(X_1, X_2, \ldots X_n) \leq H(\mathcal{X}) + \epsilon$, quelle que soit la valeur de n.
2. $P(A_\epsilon^{(n)}) > 1 - \epsilon$, pour n suffisamment grand.
3. $|A_\epsilon^{(n)}| \leq 2^{n(H(\mathcal{X})+\epsilon)}$, pour tout n (la notation $|A_\epsilon^{(n)}|$ désigne le nombre d'éléments de l'ensemble $A_\epsilon^{(n)}$).
4. $|A_\epsilon^{(n)}| \geq (1-\epsilon)2^{n(H(\mathcal{X})-\epsilon)}$, pour n suffisamment grand.

Donc l'ensemble typique a une probabilité proche de 1, et est formé d'éléments pratiquement équiprobables dont le nombre total est voisin de 2^{nH}. Comparant ce nombre total avec le nombre total de messages possibles qui vaut $|\mathcal{X}|^n = 2^{n\log|\mathcal{X}|}$ on voit que seulement dans le cas d'une distribution uniforme le nombre d'éléments de l'ensemble typique est comparable à la taille de l'alphabet de la source étendue. Si par contre $H(\mathcal{X}) < \log|\mathcal{X}|$ le rapport entre la taille de ces deux ensembles décroit exponentiellement vite, lorsque la distance $D(P(\mathcal{X})||\mathcal{U})$ augmente. La situation est résumée à la figure 8.2.

Interprétation : typicalité vs représentativité. Examinons de plus près les caractéristiques des messages typiques.

Désignons par *message représentatif*, un message tel que les fréquences relatives des symboles coïncident avec leurs probabilités a priori :

$$\frac{n_i}{n} = P(X_i).$$

Alors, il est évident que tous les messages représentatifs sont forcément typiques $\forall\epsilon$.

Montrons que de plus le nombre de messages représentatifs est du même ordre de grandeur que le nombre de messages typiques. En effet, calculons le nombre de messages représentatifs, lorsque $n \to \infty$. Si le message est représentatif, il doit comporter à peu près $n_i = nP(X_i)$ fois le symbole $X_i, \forall i = 1, \ldots |\mathcal{X}|$. Pour une longueur totale donnée n, le nombre de messages représentatifs que nous désignons par $|R_n|$ vaut

$$|R_n| = \frac{n!}{\prod_{i=1}^{|\mathcal{X}|} n_i!}. \tag{8.5}$$

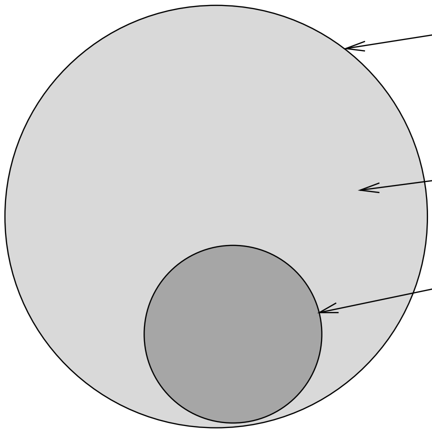


Figure 8.2. Ensembles typiques et atypiques

Introduisons la notation $a_n \doteq b_n$ comme suit :

$$a_n \doteq b_n \Leftrightarrow \lim_{n\to\infty} \frac{1}{n} \log \frac{a_n}{b_n} = 0; \tag{8.6}$$

on dit que a_n et b_n sont égaux au premier ordre dans l'exposant, car dans ce cas on peut écrire que

$$a_n = A^{n+\epsilon_n}; b_n = A^{n+\epsilon'_n} \tag{8.7}$$

avec

$$\frac{\epsilon_n - \epsilon'_n}{n} \to 0. \tag{8.8}$$

Cette propriété est indépendante de la base du logarithme et elle est évidemment symétrique.

Montrons que

$$|R_n| \doteq 2^{nH}, \tag{8.9}$$

ce qui revient à montrer que

$$\lim_{n\to\infty} \frac{1}{n} \log |R_n| = H(\mathcal{X}).$$

Rappelons la formule de Stirling pour le calcul de $n!$:

$$n! = n^n e^{-n} \sqrt{2\pi n}(1 + \epsilon_n), \tag{8.10}$$

où ϵ_n tend vers zéro lorsque n tend vers l'infini. On a donc

$$\frac{1}{n} \log |R_n| = \frac{1}{n} \log \frac{n!}{\prod_{i=1}^{|\mathcal{X}|} n_i!} \tag{8.11}$$

$$= \frac{1}{n} \left(\log n! - \sum_{i=1}^{|\mathcal{X}|} \log n_i! \right) \tag{8.12}$$

$$= \frac{1}{n} \left(n \log n - \sum_{i=1}^{|\mathcal{X}|} n_i \log n_i + A_n + B_n + C_n \right), \tag{8.13}$$

où

$$A_n = -\left(n \log e - \sum_{i=1}^{|\mathcal{X}|} n_i \log e \right) \tag{8.14}$$

$$= 0, \tag{8.15}$$

$$\begin{aligned} B_n &= \log\sqrt{\frac{2\pi n}{\prod_{i=1}^{|\mathcal{X}|} 2\pi n_i}} && (8.16)\\ &= \frac{1}{2}\left(\log n - \sum_{i=1}^{|\mathcal{X}|}\log nP(X_i)\right) + Cste && (8.17)\end{aligned}$$

et

$$C_n = \log(1+\epsilon_n) - \sum_{i=1}^{|\mathcal{X}|}\log(1+\epsilon_{n_i}). \quad (8.18)$$

On peut se convaincre facilement que $\frac{1}{n}B_n \to 0$ et que $\frac{1}{n}C_n \to 0$. Par conséquent,

$$\begin{aligned} \frac{1}{n}\log|R_n| &\to \frac{1}{n}\left(n\log n - \sum_{i=1}^{|\mathcal{X}|} n_i \log n_i\right) && (8.19)\\ &= -\frac{1}{n}\left(\sum_{i=1}^{|\mathcal{X}|} n_i(\log n_i - \log n)\right) && (8.20)\\ &= \frac{1}{n}\left(-n\sum_{i=1}^{|\mathcal{X}|} P(X_i)\log P(X_i)\right) && (8.21)\\ &= H(\mathcal{X}), && (8.22)\end{aligned}$$

ce qu'il fallait démontrer. □

Exemples. Considérons d'abord le cas où tous les symboles sont équiprobables. Dans ce cas $P(X_1, X_2, \ldots X_n) = |\mathcal{X}|^{-n}$ quel que soit le message émis par la source. Par conséquent, dans le cas d'une distribution uniforme, tous les messages sont typiques (et pas seulement les messages représentatifs). Le nombre de messages typiques est donc dans ce cas exactement égal à la taille de l'alphabet de la source étendue. De même, si nous considérons une source non uniforme, mais telle que certains symboles soient équiprobables, alors il suffira que la fréquence relative cumulée de ces symboles soit représentative pour qu'un message soit typique.

Considérons ensuite le cas d'une source binaire non uniforme de probabilité $P(1) = p_1 \neq 0.5$. Examinons un message non-représentatif M de longueur infinie, c'est-à-dire un message présentant une fréquence relative de 1 valant $f_1 \neq p_1$, pour n suffisamment grand. Alors, pour n suffisamment grand on aura

$$-\frac{1}{n}\log P(M^n) = -\frac{1}{n}\log P(M_1, \ldots, M_n) = -(f_1\log p_1 + (1-f_1)\log(1-p_1))$$

où M^n désigne les n premiers symboles de ce message. On a donc,

$$P(M^n) \neq 2^{-nH(\mathcal{X})}$$

et ce message ne restera pas typique si on fait tendre ϵ vers zéro.

On voit donc que la typicalité est une notion qui n'est sensible qu'aux différences entre les probabilités des différents symboles émis par une source. Si certains symboles sont équiprobables, la propriété de typicalité ne peut pas différencier ces symboles. La représentativité implique la typicalité, mais la réciproque n'est pas vraie en général.

On aboutit donc à la conclusion suivante : **l'ensemble des messages typiques contient les messages représentatifs, mais il est en général plus grand. Cependant, les messages typiques non-représentatifs sont en nombre négligeable, au premier ordre dans l'exposant.**

(Suggestion. Examiner sous quelles conditions sur les fréquences relatives les messages émis par une source dont l'alphabet comporte trois symboles (avec des probabilités différentes) sont typiques $\forall\epsilon > 0$. Construire des messages typiques non représentatifs.)

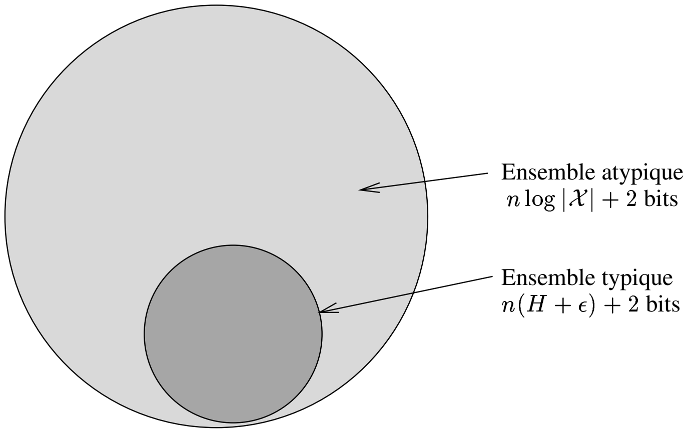


Figure 8.3. Codage de source utilisant l'ensemble typique

8.4.3 Résultat général

Le théorème AEP reste valable pour une source stationnaire ergodique quelconque (voir ci-dessous), et en particulier pour les chaînes de Markov irréductibles stationnaires (voir ci-dessous). Nous ne le démontrerons cependant pas.

Le théorème AEP *n'est pas valable* en général pour les sources stationnaires non ergodiques.

8.4.4 Compression de données

Nous allons maintenant exploiter les propriétés de l'ensemble des messages typiques pour construire un code très simpliste mais qui est néanmoins efficace.

Nous divisons l'ensemble des messages possibles $\mathcal{X}^n$ en deux sous-ensembles qui seront codés avec une stratégie énumérative : l'ensemble $A_\epsilon^{(n)}$ et son complémentaire. Nous trions ensuite les messages dans chaque ensemble selon un ordre quelconque (disons lexicographique).

Nous pouvons alors représenter chaque message figurant dans $A_\epsilon^{(n)}$ par son numéro d'ordre codé en base binaire. Puisqu'il y a moins de $2^{n(H(\mathcal{X})+\epsilon)}$ messages dans cet ensemble, ce codage nécessite au plus $n(H(\mathcal{X}) + \epsilon) + 1$ bits (le bit supplémentaire est nécessaire seulement si $n(H(\mathcal{X}) + \epsilon)$ n'est pas un nombre entier). Nous préfixons chacun de ces mots de code par un 0 pour indiquer qu'il s'agit d'un mot de code relatif à un message typique, ce qui donne au total une longueur de code de $n(H(\mathcal{X}) + \epsilon) + 2$.

Similairement, nous indexons les éléments du complémentaire de $A_\epsilon^{(n)}$, en utilisant au plus $n \log |\mathcal{X}| + 2$ bits et en utilisant le préfixe 1.

Le code est bien biunivoque et facilement décodable. Nous avons utilisé une technique naïve d'énumération, mais nous avons associé un nombre de bits plus faible à l'ensemble typique. Calculons la longueur moyenne de ce code.

Désignons par $\ell(X_1, X_2, \ldots X_n)$ la longueur du mot de code associé au message $(X_1, X_2, \ldots X_n)$, ou plus simplement $\ell(X^n)$. Si n est suffisamment grand (disons supérieur à n_0) alors $P(A_\epsilon^{(n)}) > 1 - \epsilon$, et la longueur vaut en moyenne

$$\begin{aligned}
E\{\ell(\mathcal{X}^n)\} &= \sum_{X^n} P(X^n)\ell(X^n) \\
&= \sum_{X^n \in A_\epsilon^{(n)}} P(X^n)\ell(X^n) + \sum_{X^n \notin A_\epsilon^{(n)}} P(X^n)\ell(X^n) \\
&\leq \sum_{X^n \in A_\epsilon^{(n)}} P(X^n)[n(H(\mathcal{X})+\epsilon)+2] + \sum_{X^n \notin A_\epsilon^{(n)}} P(X^n)(n\log|\mathcal{X}|+2) \\
&= P(A_\epsilon^{(n)})[n(H(\mathcal{X})+\epsilon)+2] + (1-P(A_\epsilon^{(n)}))(n\log|\mathcal{X}|+2) \\
&\leq n(H(\mathcal{X})+\epsilon) + \epsilon n \log|\mathcal{X}| + 2[P(A_\epsilon^{(n)}) + (1-P(A_\epsilon^{(n)}))] \\
&= n(H+\epsilon')
\end{aligned}$$

où nous pouvons rendre $\epsilon' = \epsilon + \epsilon\log|\mathcal{X}| + \frac{2}{n}$ aussi petit que souhaité en choisissant ϵ suffisamment petit et ensuite n suffisamment grand.

En conclusion nous avons démontré le théorème suivant, en faisant appel à la notion d'extension de source :

Pour toute source sans mémoire, il est possible de trouver un code à décodage unique qui transforme les messages de cette source en messages binaires de longueur moyenne arbitrairement proche de l'entropie de la source $H(S)$.

8.4.5 Discussion

L'idée principale de la démonstration qui précède est de faire appel à la notion de source étendue. Du fait de la loi des grands nombres, les messages d'une source étendue deviennent de plus en plus typiques, au fur et à mesure que l'ordre de cette source augmente. Alors que le nombre des messages typiques augmente avec un taux de croissance exponentiel fixé par l'entropie de la source de départ, on aboutit à une situation où la distribution de probabilités se concentre de plus en plus autour de ces messages typiques.

Nous avons démontré dans ce qui précède que la limite annoncée au début de ce chapitre pouvait être approchée pour un certain type de source. Reste à démontrer qu'elle ne peut pas être dépassée (qu'il est impossible de trouver des codes qui soient meilleurs) et qu'elle reste atteignable pour un ensemble plus large de sources que nous devrons caractériser. C'est l'objet des sections qui suivent; la démarche que nous allons adopter peut être résumée comme suit : (i) définition de l'entropie de sources discrètes stationnaires de manière à ce que le raisonnement *par extension* continue de fonctionner; (ii) étude de conditions nécessaires et suffisantes d'existence de codes déchiffrables; (iii) démonstration du premier théorème de Shannon; (iv) illustration sur la base du codage de Huffman.

8.5 SOURCES DISCRETES EN TEMPS DISCRET

Le modèle général pour les sources discrètes est celui d'un processus aléatoire discret en temps discret. Ci-dessous nous allons tout d'abord définir ce que nous entendons par là, puis allons préciser ce que signifient les propriétés de stationnarité et d'ergodicité. Ensuite, nous étudierons un peu plus en détail les chaînes de Markov, qui forment une famille de modèles fréquemment utilisés en pratique. Enfin, nous terminerons par la définition de l'entropie par symbole d'une telle source.

8.5.1 Processus aléatoires discrets en temps discret

Un processus aléatoire en temps discret est une suite de variables aléatoires $\mathcal{X}_i$, indexée par un paramètre entier (nous supposerons dans la suite que $i > 0$). Nous nous intéressons plus particulièrement au cas où les variables aléatoires sont elles-mêmes discrètes et nous supposerons que $\forall i, \mathcal{X}_i = \mathcal{X} = \{1, \ldots, q\}$ c'est-à-dire que les v.a. utilisent un même alphabet fini. Rappelons que la notation $\mathcal{X}_i$ représente à la fois la v.a. et son ensemble de valeurs possibles.

Un tel processus aléatoire est en principe caractérisé par les lois de probabilités conjointes $P(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ définies sur $\mathcal{X}^n, \forall n = 1, 2, \ldots$, qui permettent de calculer par marginalisation $P(\mathcal{X}_n, \ldots, \mathcal{X}_{n+k}), \forall n > 0, k \geq 0$.

La notion de processus aléatoire à valeurs discrètes et en temps discret est le modèle probabiliste le plus général qu'on puisse imaginer pour représenter la notion de source discrète. Nous allons tout de suite faire deux restrictions, qui imposent respectivement la notion de *stationnarité* et *d'ergodicité*.

Processus stationnaire. Un processus aléatoire $\mathcal{X}_i$ est dit *stationnaire* si les lois de probabilités sont indépendantes de l'origine des temps, c'est-à-dire si

$$P(\mathcal{X}_1 = i^1, \ldots, \mathcal{X}_n = i^n) = P(\mathcal{X}_{1+\ell} = i^1, \ldots, \mathcal{X}_{n+\ell} = i^n)$$

quels que soient n et ℓ positifs, et quels que soient $i^1, \ldots, i^n \in \mathcal{X}$.

Par exemple, si nous supposons que les $\mathcal{X}_i$ sont indépendantes et distribuées selon une même loi, alors

$$P(\mathcal{X}_1 = i^1, \ldots, \mathcal{X}_n = i^n) = P(\mathcal{X} = i^1) \ldots P(\mathcal{X} = i^n)$$

et le processus est évidemment stationnaire.

Processus ergodique. Nous dirons qu'un processus aléatoire stationnaire est *ergodique* si pour tout $k = 1, 2, \ldots$, pour toute suite d'indices $i_1, \ldots i_k$, et pour toute fonction $f(\cdot)$ de $\mathcal{X}^k$ dans $\mathbb{R}$ dont l'espérance mathématique $E\{f(\mathcal{X}_{i_1}, \ldots, \mathcal{X}_{i_k})\}$ est finie (condition d'existence), on a

$$\frac{1}{n} \sum_{t=1}^{n} f(\mathcal{X}_{i_1+t}, \ldots, \mathcal{X}_{i_k+t}) \xrightarrow{p.s.} E\{f(\mathcal{X}_{i_1}, \ldots, \mathcal{X}_{i_k})\}. \tag{8.23}$$

En d'autres termes, un processus stationnaire est dit ergodique si les espérances mathématiques de v.a. arbitraires définies sur le processus peuvent être approchées par des *moyennes temporelles* le long d'un réalisation (ou trajectoire) quelconque du processus de longueur infinie. Cela veut dire, qu'on peut étudier ce processus en observant une seule réalisation de longueur suffisante de ce processus.

Par exemple, la loi (forte) des grands nombres permet d'affirmer que la source stationnaire sans mémoire est aussi ergodique.

Nous renvoyons les étudiants au cours d'*Introduction aux processus stochastiques* pour une discussion plus fine et des exemples pratiques éclairant les notions de stationnarité et d'ergodicité. Nous supposerons, dans le cadre de ce cours, que toutes les sources sont des processus stationnaires et ergodiques.

Notons que dans le cas où le processus est à valeurs discrètes l'espérance mathématique de toute fonction (bornée) est toujours définie, et la condition d'existence relative à celle-ci devient dès lors redondante. On peut alors montrer que dans ce cas il faut (et il suffit) pour qu'un processus stationnaire soit ergodique que les fréquences relatives de toute combinaison de symboles observées le long d'une trajectoire convergent vers la probabilité de cette combinaison.

Réinsistons sur le fait que l'ergodicité (au sens où nous l'utilisons dans ce cours) n'est définie que pour les processus stationnaires. Un processus ergordique est donc implicitement aussi stationnaire.

8.5.2 Chaînes de Markov

Nous allons maintenant définir avec plus de précision la notion de chaîne de Markov introduite plus haut, et nous allons en particulier définir les conditions sous lesquelles une telle chaîne (source) est stationnaire et ergodique.

Remarque. Les chaînes de Markov fournissent un modèle important de processus aléatoire. Une source de Markov est une source dont les symboles émis sucessivement forment une chaîne de Markov. Nous utiliserons ces deux termes de façon interchangeable.

8.5.2.1 Définitions.

Chaîne de Markov. Un processus en temps discret et à valeurs discrètes est une *chaîne de Markov* si, $\forall n = 1, 2, \ldots,$

$$P(\mathcal{X}_{n+1} = i^{n+1} | \mathcal{X}_n = i^n, \ldots, \mathcal{X}_1 = i^1) = P(\mathcal{X}_{n+1} = i^{n+1} | \mathcal{X}_n = i^n), \tag{8.24}$$

$\forall i^{n+1}, i^n, \ldots i^1 \in \mathcal{X}$.

Dans ce cas on peut écrire

$$P(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n) = P(\mathcal{X}_1)P(\mathcal{X}_2|\mathcal{X}_1), \ldots, P(\mathcal{X}_n|\mathcal{X}_{n-1}). \quad (8.25)$$

L'espace d'états de la chaîne de Markov est par définition l'ensemble des valeurs que peut prendre le processus à chaque instant (noté ci-dessus par $\mathcal{X} = \{1, \ldots, q\}$). On dira que $q = |\mathcal{X}|$ est le nombre d'états de la chaîne.

Notons qu'on réserve généralement le terme de processus de Markov au cas où l'espace d'états est continu.

Invariance dans le temps. La chaîne de Markov est dite *invariante dans le temps* ou *homogène*, si $\forall n = 1, 2, \ldots,$ on a

$$P(\mathcal{X}_{n+1}|\mathcal{X}_n) = P(\mathcal{X}_2|\mathcal{X}_1), \quad (8.26)$$

c'est-à-dire si ces probabilités de transition ne dépendent pas du temps. Insistons ici sur le fait que cette condition est différente de la condition de *stationnarité*.

Une chaîne de Markov invariante dans le temps est donc complètement caractérisée par son vecteur de probabilités inititiales $\pi = (\pi_1, \ldots, \pi_q)$ défini par

$$\pi_i = P(\mathcal{X}_1 = i), \quad (8.27)$$

et sa matrice de transition Π de dimension $q \times q$ définie par

$$\Pi_{ij} = P(\mathcal{X}_2 = j|\mathcal{X}_1 = i). \quad (8.28)$$

La i-ème ligne de cette matrice correspond aux probabilités de départ de l'état i; la $j-$ème colonne correspond aux probabilités d'arrivée à l'état j. On doit évidemment avoir $\forall i \leq q$,

$$\sum_{j=1}^{q} \Pi_{ij} = 1 \quad (8.29)$$

Un telle chaîne de Markov peut donc être représentée par un graphe orienté dont les q noeuds représentent les états et les arcs orientés les transitions d'un état i vers un état j. Les arcs (i, j) sont étiquetés par les probabilités de transition Π_{ij} et les noeuds par le numéro de l'état correspondant (et éventuellement les probabilités initiales π_i, si celles-ci sont spécifiées). On ne représente graphiquement que les arcs correspondant à des transitions possibles, c'est-à-dire telles que $\Pi_{ij} > 0$ (voir figure 8.4).

Irréductibilité. La chaîne de Markov est dite *irréductible* s'il est possible de passer en un nombre fini d'étapes (avec une probabilité non nulle) de n'importe quel état à n'importe quel autre état. Sinon, la chaîne est dite réductible.

Disons que deux états communiquent si il est possible de passer de l'un à l'autre par un chemin de longueur finie dans le graphe de la chaîne dont tous les arcs ont une probabilité strictement positive, et si cette propriété reste vraie si on inverse le rôle joué par les deux noeuds. Il est facile d'imaginer des situations qui ne vérifient pas cette condition. Une chaîne est donc irréductible si tous ses états communiquent.

Il est clair que la relation de communication est une relation d'équivalence (transitive, symétrique et réflexive). Cette relation partitionne donc les états de la chaîne en classes d'équivalence. La chaîne est donc irréductible si le nombre de ses classes d'équivalence se réduit à une seule, qui comprend donc tous les états. Ces notions sont illustrées à la figure 8.4, où les états sont numérotés de 1 à 6.

Apériodicité. Soit P_{jj}^n la probabilité que partant de l'état j, la chaîne retourne à cet état après exactement n pas de temps, et soit

$$I_j \triangleq \{n > 0 \ : \ P_{jj}^n > 0\},$$

c'est-à-dire l'ensemble des temps de retour possibles pour l'état j. Si tous ces nombres sont multiples d'un entier $k > 1$ on dira que l'état est périodique. La période $d(j)$ de l'état j est alors le plus grand commun diviseur de l'ensemble des valeurs I_j. Si $d(j) = 1$ l'état est dit apériodique. (Si l'ensemble $I_j = \emptyset$ on pose $d(j) = 0$.)

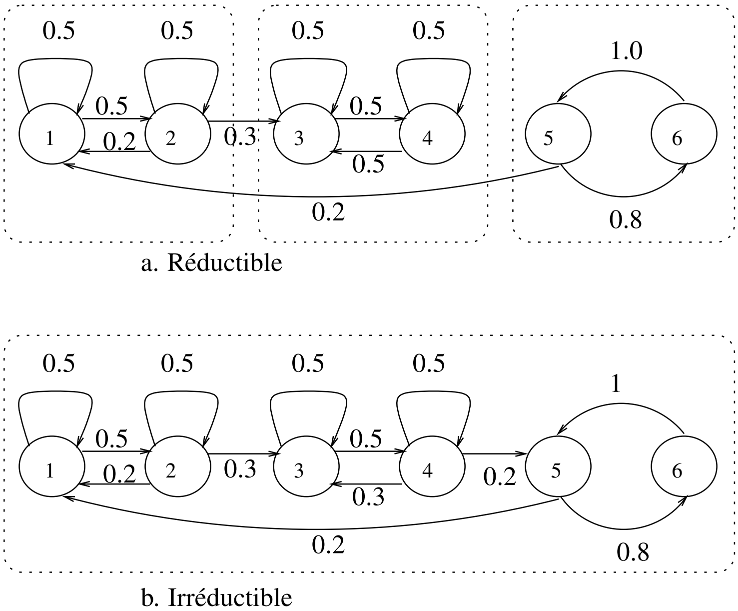


Figure 8.4. Exemples de chaînes réductibles et irréductibles

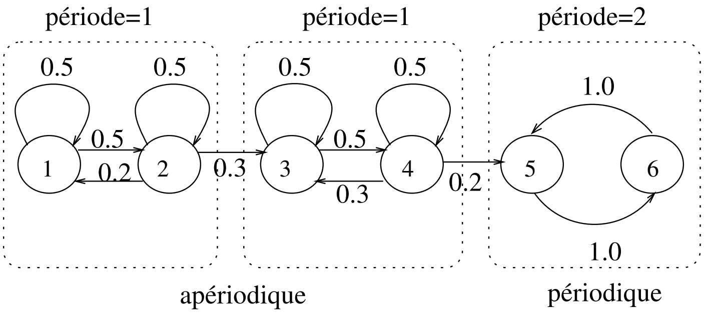


Figure 8.5. Exemples de chaînes périodique et apériodique

Une chaîne de Markov est dite apériodique si tous les états sont apériodiques. On montre que tous les états d'une classe de communication ont la même période. Par conséquent, si une chaîne est irréductible, tous ses états ont même période (voir figure 8.5).

Une chaîne de Markov irréductible peut néanmoins présenter un comportement périodique. Par exemple, à la figure 8.5 si on ne conserve que la partie de droite on obtient une chaîne de Markov irréductible de période 2.

Etats récurrents et transitoires. Un état est dit *récurrent* si la chaîne de Markov y retourne infiniment souvent (ou avec certitude). Autrement, l'état est dit *transitoire* : il s'agit d'un état tel que l'ensemble des séquences dans lesquelles cet état n'apparaît qu'un nombre fini de fois est de probabilité égale à 1.

Une chaîne est dite récurrente si tous les états sont récurrents. On montre que tous les états d'une classe de communication sont soit transitoires soit récurrents. Donc, les états d'une chaîne irréductible sont soit tous récurrents soit tous transitoires.

Par exemple, pour la chaîne de Markov de la figure 8.4(a) on voit aisément que les deux états du milieu sont récurrents, alors que les quatre autres états sont transitoires.

Notons que si l'espace d'état est fini, il y a au moins un état récurrent.

Régime(s) permanent(s). On peut caractériser le comportement asymptotique d'un processus de Markov de différentes façons.

On dit que la chaîne de Markov atteint un régime permanent si la limite

$$\lim_{n \to \infty} P(\mathcal{X}_n = i) = p_i \tag{8.30}$$

existe pour tout $i = 1, \ldots, q$.

On montre que si cette limite existe, elle doit satisfaire à l'équation vectorielle

$$\boldsymbol{p} = \boldsymbol{p}\boldsymbol{\Pi}, \tag{8.31}$$

où $\boldsymbol{p} = (p_1, \ldots, p_q)$, ou, plus explicitement,

$$p_j = \sum_{i=1}^{q} \Pi_{ij} p_i. \tag{8.32}$$

En effet, si la distribution $P(\mathcal{X}_n)$ est donnée on en déduit que

$$P(\mathcal{X}_{n+1} = j) = \sum_{i=1}^{q} P(\mathcal{X}_n = i) \Pi_{ij}, \tag{8.33}$$

et en prenant la limite pour $n \to \infty$ des deux membres de cette équation, on a

$$p_j = \lim_{n \to \infty} \sum_{i=1}^{q} P(\mathcal{X}_n = i) \Pi_{ij}, \tag{8.34}$$

ce qui donne en intervertissant la limite et la somme

$$p_j = \sum_{i=1}^{q} p_i \Pi_{ij}, \tag{8.35}$$

si les limites existent.

On dit que tout $\boldsymbol{p}$ qui satisfait (8.32) est une distribution stationnaire, car une chaîne de Markov initialisée avec une de ses distributions stationnaires forme un processus stationnaire.

Il est possible, pour une chaîne de Markov quelconque, d'avoir plusieurs distributions stationnaires. Il est en outre possible pour une chaîne de Markov d'avoir une distribution stationnaire, sans nécessairement atteindre le régime permanent.

Une autre façon de caractériser le comportement asymptotique d'un processus de Markov est basé sur le comportement de $P(\mathcal{X}_n | \mathcal{X}_1 = j)$. Pour dire que la chaîne est stable on aimerait bien que cette grandeur tende vers une limite quelles que soient les conditions initiales et que cette limite soit indépendante des conditions initiales. De cette façon, la chaîne aura un comportement asymptotique indépendant du choix des probabilités initiales.

Exemples.

1. Considérons une chaîne de Markov à deux états (numérotés 1 et 2) caractérisée par la matrice de transition suivante

$$\begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}.$$

Il s'agit d'une chaîne de Markov irréductible et apériodique. De plus quelle que soit la distribution de l'état à l'étape $n-1$, on a $P(X_n = 1) = 0.5P(X_{n-1} = 1) + 0.5P(X_{n-1} = 2) = 0.5$. Cette chaîne a donc un comportement stationnaire et elle admet comme distribution stationnaire $(0.5, 0.5)$. Elle atteint son régime permanent en une seule étape.

2. Considérons une chaîne de Markov à deux états caractérisée par la matrice de transition suivante

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Il s'agit d'une chaîne de Markov irréductible de période 2. Elle admet aussi la (seule) distribution stationnnaire $(0.5, 0.5)$. Cependant, quelles que soient les conditions initiales, cette chaîne ne peut pas atteindre un régime permanent, puisqu'on aura $P(X_n = 1|X_1 = 1) = 1$, pour n pair, et 0 pour n impair. En effet, la suite $P(\mathcal{X}_n|X_1 = 1)$ possède, quelles que soient les conditions initiales, deux sous-suites convergentes (constantes) qui ne convergent pas vers la même limite.

3. Considérons la chaîne de Markov relative au modèle d'ordre 1 de la télécopie (source S_2). Sa matrice de transition est

$$\Pi = \begin{bmatrix} 0.93 & 0.07 \\ 0.63 & 0.37 \end{bmatrix}.$$

On trouve que la distribution $(0.9, 0.1)$ est stationnaire. En effet, on a

$$0.9 * 0.93 + 0.1 * 0.63 = 0.9$$

et

$$0.9 * 0.07 + 0.1 * 0.37 = 0.1$$

et par ailleurs, on a

$$\Pi^2 = \begin{bmatrix} 0.909 & 0.091 \\ 0.819 & 0.181 \end{bmatrix},$$

et

$$\lim_{n\to\infty} \Pi^n = \begin{bmatrix} 0.9 & 0.1 \\ 0.9 & 0.1 \end{bmatrix},$$

et donc

$$\lim_{n\to\infty} P(\mathcal{X}_n) = \lim_{n\to\infty} \sum_j P(\mathcal{X}_n|\mathcal{X}_1 = j)P(\mathcal{X}_1 = j) = \sum_j P(\mathcal{X}_1 = j) \lim_{n\to\infty} \Pi^{n-1} = (0.9, 0.1),$$

quelle que soit la distribution initiale $P(\mathcal{X}_1)$. En effet, le terme $P(\mathcal{X}_n|\mathcal{X}_1 = j)P(\mathcal{X}_1 = j)$ désigne la $j-$ème ligne de la matrice Π^{n-1}, qui finit par devenir indépendante de la valeur de j.

Cette chaîne de Markov possède une distribution stationnaire unique et si on observe la chaîne suffisamment longtemps, la distribution des états tend vers cette distribution stationnaire, quelles que soient ses conditions initiales. Mais il lui faut un nombre infini d'étapes pour atteindre son régime permanent.

4. Considérons ensuite une chaîne de Markov réductible ayant deux classes de communication apériodiques, par exemple la chaîne constante caractérisée par la matrice identité

$$\Pi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Pour cette chaîne toutes les distributions initiales sont stationnaires. Cependant, selon que l'état initial est 0 où 1, cette chaîne présentera deux trajectoires différentes. On voit donc que les moyennes temporelles le long d'une trajectoire sont sensibles aux conditions initiales. Ce type de comportement, même s'il est stationnaire est non ergodique.

Existence et unicité d'une distribution stationnaire. On montre que toute chaîne de Markov avec un nombre fini d'états, *irréductible* et *apériodique*, admet une seule distribution stationnaire π_∞ solution de l'équation (8.32) et on a

$$\pi_\infty = \lim_{n\to\infty} P(\mathcal{X}_n) = \lim_{n\to\infty} P(\mathcal{X}_n|\mathcal{X}_1 = i), \tag{8.36}$$

$\forall i = 1, \dots q$.

On montre également que dans ce cas la matrice

$$\overline{\Pi} = \lim_{n\to\infty} \Pi^n, \tag{8.37}$$

est bien définie, et que toutes ses lignes sont identiques et égales au vecteur π_∞.

Lorsque le nombre d'états est fini, l'irréductibilité et l'existence d'une distribution stationnaire entraînent que les probabilités stationnaires de tous les états sont alors strictement positives. En effet, supposons que la probabilité stationnaire d'un état soit nulle, disons $\pi_{\infty,1} = 0$. Alors, on en déduit que

$$\sum_{j=1}^{q} \pi_{\infty,j} \Pi_{j1} = \pi_{\infty,1} = 0,$$

ce qui veut dire que les probabilités stationnaires des états qui présentent des transitions vers l'état 1 doivent être nulles (puisque les valeurs correspondantes des Π_{j1} sont toutes positives). On en déduit que si la probabilité stationnaire d'un état est nulle, il en est de même de tous les états qui autorisent des transitions vers cet état. En faisant le même raisonnement de proche en proche, on en déduit que tous les états à partir desquels il est possible d'atteindre l'état dont on a supposé la probabilité stationnaire nulle (en un nombre fini d'étapes) sont de probabilité stationnaire nulle. Mais puisque la chaîne est irréductible, tous les états sont concernés, et donc toutes les composantes de π_∞ sont nulles, ce qui est incompatible avec l'existence même de ce vecteur (dont la somme des composantes, toutes non négatives, doit valoir 1).

Il est cependant possible, pour une chaîne apériodique mais réductible de présenter une distribution stationnaire dont certaines composantes sont nulles. Mais, lorsqu'on s'intéresse au comportement asymptotique de ce type de chaînes de Markov les états transitoires peuvent être effacés, pour ne retenir que les composantes correspondant aux états dont les probabilités stationnaires sont strictement positives. Si la chaîne de Markov obtenue ainsi comporte une seule classe de communication apériodique, alors initialisée avec sa distribution stationnaire, elle présentera un comportement stationnaire et ergodique.

Par exemple, la chaîne de Markov de matrice de transition

$$\Pi = \begin{bmatrix} 0.5 & 0.5 \\ 0.0 & 1.0 \end{bmatrix},$$

présente cette caractéristique. On trouve que

$$\overline{\Pi} = \lim_{n\to\infty} \Pi^n = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix},$$

et que la distribution stationnaire est $\pi_\infty = (0, 1)$. La condition que l'état 1 soit parcouru un nombre infini de fois est dans ce cas identique à la condition que la chaîne reste tout le temps dans cet état. Elle vaut donc

$$\lim_{n\to\infty} P(\mathcal{X}_1 = 1, \ldots, \mathcal{X}_n = 1) = P(\mathcal{X}_1 = 1) \prod_{i=1}^{\infty} \Pi_{11} = 0,$$

et cet état est bien transitoire.

Pour terminer, mentionnons que dans le cas des chaînes de Markov ayant un nombre infini d'états, on introduit la notion de chaîne *positivement récurrente*, vérifiée si tous les états sont récurrents et ont une probabilité stationnaire strictement positive.

Ergodicité des chaînes de Markov. Une chaîne de Markov *irréductible apériodique*[1] initialisée selon sa distribution stationnaire forme un processus aléatoire stationnaire et ergodique.

Les exemples 2 et 4 montrent qu'il n'est par contre pas vrai que n'importe quelle source de Markov soit stationnaire et ergodique.

Si on observe suffisamment longtemps une chaîne de Markov *irréductible* et *apériodique* initialisée de manière quelconque elle finit par présenter un comportement quasi stationnaire et ergodique. L'initialisation avec la distribution stationnaire revient donc à négliger ce régime transitoire, dont le poids statistique finit par s'estomper de toutes façons.

[1]et positivement récurrente, mais nous supposons que nous travaillons avec un nombre fini d'états, ce qui fait que cette condition est un conséquence de l'irréductibilité.

8.5.2.2 Entropie. On définit l'entropie d'une source de Markov stationnaire par

$$H(S) = -\sum_{i=1}^{m}\sum_{j=1}^{m} \pi_i \Pi_{ij} \log \Pi_{ij}. \tag{8.38}$$

Nous reviendrons sur cette formule, un peu plus loin, après avoir défini l'entropie des processus stationnaires. Pour le moment, disons simplement qu'elle est plausible, puisqu'elle définit l'entropie par la moyenne des informations innovantes de tous les états, pondérées par la probabilité stationnaire des états.

8.5.2.3 Mémoire finie. On définit une source de Markov *de mémoire* $m \geq 0$, comme une source qui émet des suites de symboles vérifiant

$$P(S(t+m+1)|S(1),\ldots,S(t+m)) = P(S(t+m+1)|S(t+1),\ldots,S(t+m)). \tag{8.39}$$

La probabilité d'émission d'un symbole ne dépend donc que des m symboles précédents émis.

En particulier, lorsque $m = 0$, on parle de source sans mémoire, et lorsque $m = 1$, d'une source de Markov.

Les sources de mémoire finie m forment évidemment un ensemble en principe plus riche pour modéliser les sources réelles. En particulier elles permettent de modéliser assez bien les textes écrits en langue naturelle, ou les langages de programmation. Cependant, nous allons voir ci-dessous que l'étude de ces sources peut se ramener à l'étude des sources de mémoire $m = 1$.

8.5.2.4 Extensions d'ordre élevé. Nous allons justifier maintenant le fait que nous pouvons restreindre notre attention aux sources de Markov (sous-entendu d'ordre 1).

En effet, soit S une source Q-aire de mémoire m, et considérons son extension d'ordre n notée S^n (dont les symboles correspondent aux Q^n séquences possibles de n symboles consécutifs de la source S). Alors, la mémoire de cette source est égale au plus petit entier supérieur ou égal au rapport $\frac{m}{n}$ c'est-à-dire

$$m(S^n) = \left\lceil \frac{m(S)}{n} \right\rceil, \tag{8.40}$$

où la notation $\lceil\cdot\rceil$ signifie que nous arrondissons à l'entier supérieur ou égal.

Donc, si $n \geq m$, $m(S^n) = 1$. Par conséquent, toute source de Markov de mémoire finie peut se ramener à une source de Markov de mémoire valant 1.

On montre que l'entropie de l'extension d'ordre n d'une source de Markov vaut n fois l'entropie de celle-ci

$$H(S^n) = nH(S). \tag{8.41}$$

8.5.2.5 Source adjointe. La source adjointe S' d'une source de Markov S est la source sans mémoire définie par les probabilités stationnaires de la source de Markov. On a évidemment

$$H(S') \geq H(S), \tag{8.42}$$

et l'écart entre ces deux grandeurs mesure le degré de dépendance entre symboles successifs. Il est nul si, et seulement si, ceux-ci sont indépendants.

On peut montrer que

$$\lim_{n\to\infty} |H(S^{n\prime}) - H(S^n)| = 0, \tag{8.43}$$

ce qui veut dire qu'au fur et à mesure qu'on considère des extensions d'ordre de plus en plus élevé d'une source de Markov, on se rapproche de plus en plus d'un comportement sans mémoire.

(Suggestion : comparer ceci aux définitions de l'entropie d'une source stationnaire quelconque, données ci-dessous.)

8.5.3 Entropie de sources stationnaires

Le modèle général de source discrète coïncide avec celui d'une variable aléatoire en temps discret et à valeurs discrètes. Il est donc complètement défini par les lois de probabilités $P(\mathcal{X}_1,\ldots,\mathcal{X}_n), \forall n = 1,2,\ldots$.

Clairement, l'entropie $H(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ des messages de longueur n de ce type de source est définie, pour tout $n > 0$.

Nous définissons l'entropie (on dit aussi le taux d'entropie) de la source S par

$$H(S) = \lim_{n\to\infty} \frac{H(\mathcal{X}_1, \ldots, \mathcal{X}_n)}{n}, \tag{8.44}$$

qui représente l'entropie moyenne par symbole pour des chaînes de longueur infinie, si cette limite existe.

Notons d'emblée que cette définition étendue est bien compatible avec la définition de l'entropie de la source stationnaire sans mémoire. En effet, dans ce cas

$$H(\mathcal{X}_1, \ldots, \mathcal{X}_n) = nH(\mathcal{X}).$$

Notons aussi qu'il est parfaitement possible de construire des exemples de sources telles que la limite (8.44) ci-dessus n'existe pas (voir exercices).

Une autre définition possible pour l'entropie de la source aurait été

$$H'(S) = \lim_{n\to\infty} H(\mathcal{X}_n|\mathcal{X}_{n-1} \ldots, \mathcal{X}_1), \tag{8.45}$$

qui représente, si cette limite existe, l'information asympotiquement apportée par un symbole additionnel émis par la source. Notons que pour un processus stationnaire sans mémoire cette définition est également compatible avec la définition de base.

Théorème : l'existence de $H'(S)$ implique $H(S) = H'(S)$.

Montrons que si $H'(S)$ existe, alors $H(S)$ existe et on a $H(S) = H'(S)$. La démonstration repose sur le résultat suivant (théorème de la moyenne de Cesaro) : *Si $a_n \to a$ et $b_n = \frac{1}{n}\sum_{i=1}^{n} a_i$ alors $b_n \to a$.*

En effet, appliquons le théorème de Cesaro au cas où $a_n = H(\mathcal{X}_n|\mathcal{X}_{n-1} \ldots, \mathcal{X}_1)$. On déduit de l'existence de $H'(S)$ que

$$\frac{1}{n}\sum_{i=1}^{n} H(\mathcal{X}_i|\mathcal{X}_{i-1} \ldots, \mathcal{X}_1) \to H'(S).$$

Or le premier membre de cette limite est justement égal à $H(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ par application de la règle de chaînage (4.37).□

Notons que la réciproque de ce théorème n'est pas vraie. Néanmoins, le théorème suivant donne une condition suffisante d'existence de $H'(S)$ et donc de $H(S)$.

Théorème : entropie de processus stationnaires.

Pour toute source stationnaire $H'(S)$ et donc $H(S)$ existe.

En effet, en général on a

$$H(\mathcal{X}_{n+1}|\mathcal{X}_n, \ldots, \mathcal{X}_1) \leq H(\mathcal{X}_{n+1}|\mathcal{X}_n, \ldots, \mathcal{X}_2).$$

La condition de stationnarité implique que

$$H(\mathcal{X}_{n+1}|\mathcal{X}_n, \ldots, \mathcal{X}_2) = H(\mathcal{X}_n|\mathcal{X}_{n-1}, \ldots, \mathcal{X}_1).$$

On en déduit que la suite $H(\mathcal{X}_{n+1}|\mathcal{X}_n, \ldots, \mathcal{X}_1)$ est une suite décroissante si le processus est stationnaire. Comme cette suite est de plus minorée (car positive ou nulle) elle doit nécessairement converger. Donc $H'(S)$ existe bien et donc aussi $H(S)$.

On en déduit également que pour une source stationnaire on a

$$H(S) \leq H(\mathcal{X}_n|\mathcal{X}_{n-1}, \ldots, \mathcal{X}_1), \tag{8.46}$$

pour tout n.

Puisque

$$H(\mathcal{X}_n, \mathcal{X}_{n-1}, \ldots, \mathcal{X}_1) = \sum_{i=1}^{n} H(\mathcal{X}_i|\mathcal{X}_{i-1}, \ldots, \mathcal{X}_1),$$

on déduit également que la suite

$$\frac{H(\mathcal{X}_n, \mathcal{X}_{n-1}, \ldots, \mathcal{X}_1)}{n}$$

est monotonément décroissante. Par conséquent on a également $\forall n$

$$H(\mathcal{S}) \leq n^{-1} H(\mathcal{X}_n, \mathcal{X}_{n-1}, \ldots, \mathcal{X}_1). \tag{8.47}$$

Entropie d'une chaîne de Markov. Plaçons nous dans le cas d'une source stationnaire de Markov; l'entropie telle que définie ci-avant donne alors lieu à la formule suivante :

$$\begin{aligned} H(\mathcal{S}) = H'(\mathcal{S}) &= \lim_{n\to\infty} H(\mathcal{X}_n|\mathcal{X}_{n-1}, \ldots, \mathcal{X}_1) \\ &= \lim_{n\to\infty} H(\mathcal{X}_n|\mathcal{X}_{n-1}) = H(\mathcal{X}_2|\mathcal{X}_1), \end{aligned} \tag{8.48}$$

où on utilise la distribution stationnaire pour les états, ce qui conduit à la même définition que celle que nous avons introduite de but en blanc plus haut (formule 8.38, page 132).

8.6 CODAGE DE SOURCE

Le codage de source, encore appelé codage de canal sans erreur, vise à utiliser au mieux la capacité (de transmission, ou de stockage) d'un canal supposé sans erreur. Nous avons indiqué dans l'introduction, sans justification, qu'un schéma de communication pouvait être décomposé de manière à mettre en évidence (en partie centrale) un canal idéal (sans erreurs). Il est donc naturel de s'interroger sur la façon d'utiliser au mieux la capacité d'un tel canal pour transmettre de l'information en provenance d'une source aux propriétés connues. Nous verrons au chapitre suivant comment “emballer” un canal non-idéal de manière à ce qu'il puisse être utilisé de cette manière.

8.6.1 Position du problème

Le but du codage de source est de remplacer les messages émis par une source $\mathcal{S}$ utilisant un alphabet Q-aire $s_1, \ldots, s_Q$, par des messages écrits dans un alphabet donné q-aire qui est celui utilisé par un canal ou un système de stockage d'information. Dans la plupart des cas pratiques on utilise un alphabet binaire ($q = 2$), mais l'ensemble des résultats que nous allons développer est valable quel que soit $q > 1$. Par ailleurs, la taille de l'alphabet d'origine est également quelconque; il s'en suit que nous pouvons considérer de la même façon le codage symbole par symbole, ou blocs de symboles par blocs de symboles.

Par exemple, les sources $\mathcal{S}_3$ et $\mathcal{S}_4$ utilisent un alphabet de 27 symboles. Le code Morse (développé en 1837) réalisait une conversion d'alphabet vers un alphabet quaternaire ($q = 4$) : le point, le trait, l'espace court et l'espace long. Il s'agit d'un code de longueur variable qui associe la séquence la plus courte à la lettre la plus fréquente en anglais (le ‘E’).

Nous allons essentiellement étudier des codes de longueur variable et nous considérons pour la simplicité qu'un tel code associe à chaque symbole s_i ($i = 1, \ldots, Q$) de la source un *mot de code* q-aire, c'est-à-dire une suite m_i de n_i (éventuellement fonction de i) symboles de l'alphabet de destination q-aire.

Un tel code doit posséder d'une part des propriétés qui permettent de reconstruire (plus ou moins facilement) les messages envoyés par la source, d'autre part il doit permettre d'utiliser au mieux la capacité du canal en réalisant une adaptation statistique (qui dépend donc des propriétés statistiques de la source).

8.6.2 Propriétés souhaitées des codes

Régularité. Un code est dit régulier (ou non-singulier) si tous les mots de codes m_i sont distincts. Il est clair qu'un code doit être au moins régulier, ce qui implique que tout message de longueur 1 de la source peut être décodé.

Déchiffrabilité. Un code régulier est dit déchiffrable (ou encore à décodage unique) si pour toute suite $m^1, \ldots, m^n$ de mots de code il est possible de distinguer les mots m^i sans ambiguïté, et donc retrouver les symboles s^i correspondants. Il existe différentes façons d'assurer cette propriété :

Longueur fixe. En utilisant des mots tels que $n_i = n$, on peut reconstituer les messages aisément (exemple : code ASCII).

Séparateur. On consacre un des symboles de la source comme séparateur de fin de mot (exemple binaire : on code s_i par une suite de i '1' suivi d'un '0').

Sans préfixes. En évitant qu'un mot du code ne soit identique au début d'un autre (ne soit un préfixe d'un autre mot).

Notons que la dernière condition est plus générale que les deux premières : un code régulier de longueur fixe est nécessairement sans préfixes; de même un code avec séparateur est également sans préfixes.

On dit d'un code qu'il est à *décodage instantané* s'il est possible de décoder les mots de code dès lors que tous les symboles qui en font partie ont été reçus.

Il est cependant possible de construire des codes déchiffrables qui ne soient pas instantanés.

Condition nécessaire et suffisante de déchiffrabilité. Pour qu'un code soit à décodage unique il faut et il suffit que toutes ses extensions soient régulières.

(L'extension d'ordre n d'un code est le code formé par les séquences de n mots du code original.)

Condition nécessaire et suffisante d'un code instantané. Pour qu'un code soit instantané, il faut et il suffit qu'il soit sans préfixes. Cela justifie donc notre appellation ci-dessus.

8.6.3 Inégalité de Kraft

L'inégalité de Kraft constitue un résultat fondamental en théorie des codes. Elle fournit en effet une condition nécessaire et suffisante d'existence de codes déchiffrables et instantanés, exprimée en fonction de la longueur des mots du code. Historiquement, elle fut d'abord démontrée par McMillan pour les codes déchiffrables. Le fait qu'elle reste vraie pour les codes instantanés suggère qu'on ne réduit pas significativement les possibilités de codage en se restreignant à cette classe de codes.

Condition de McMillan. Soient $n_1, \ldots, n_Q$ des longueurs de mots candidates pour coder une source Q-aire dans un alphabet q-aire. Alors l'inégalité *de Kraft*

$$\sum_{i=1}^{Q} q^{-n_i} \leq 1 \tag{8.49}$$

est une condition nécessaire et suffisante d'existance d'un code déchiffrable respectant ces longueurs de mots.

La condition nécessaire signifie que :

- Si un code est à décodage unique (et donc a fortiori s'il est instantané), les longueurs de mots doivent satisfaire (8.49).
- Cela signifie donc aussi que si la condition n'est pas satisfaite par un code quelconque, ce code n'est pas déchiffrable (ni a fortiori instantané).

La condition suffisante signifie (plus subtilement) que : si on se donne un ensemble de longueurs de mots qui satisfont (8.49), alors il est possible de construire un code déchiffrable utilisant ces longueurs de mots.

Codes instantanés.

L'inégalité de Kraft est aussi une condition nécessaire et suffisante d'existence de codes instantanés.

Le fait qu'elle soit nécessaire est évidemment un corollaire direct de la condition de McMillan, puisque tout code instantané est également déchiffrable.

Pour démontrer la suffisance de cette condition, nous adopterons à la section 8.8 une démarche constructive qui établit comment construire un code instantané, une fois que sont donnés des n_i qui satisfont (8.49). Notons que nous aurons par là également démontré la suffisance de la condition de McMillan, puisque tout code instantané est également déchiffrable.

Il ne nous reste donc ici qu'à démontrer le fait que l'inégalité de Kraft est une condition nécessaire d'existence de code déchiffrable.

Avant de passer à la démonstration, remarquons qu'on déduit immédiatement des deux conditions nécessaires et suffisantes ci-dessus la propriété suivante

Il existe un code déchiffrable ayant pour longueurs de mots n_i, si, et seulement s'il existe un code instantané ayant les mêmes longueurs de mots.

Démonstration de la nécessité de la condition de McMillan. Considérons le développement de

$$D = \left(\sum_{k=1}^{s} r_k q^{-k}\right)^m, \tag{8.50}$$

où r_k est un entier positif ou nul, m entier positif.

Les termes du développement de D s'écrivent

$$r_{i_1} r_{i_2} \ldots r_{i_m} q^{-(i_1+i_2+\ldots+i_m)} \tag{8.51}$$

avec $1 \leq i_j \leq s, j = 1, 2, \ldots, m$ et $m \leq i_1 + i_2 + \ldots + i_m \leq sm$.

Soit alors un certain code déchiffrable, et supposons que les r_k dans D représentent le nombre de mots de codes de longueur k de ce code (et s la longueur maximale). Alors, le nombre de combinaisons de m mots donnant lieu à un message codé de n symboles du code est

$$\nu(n) = \sum_{i_1,\ldots,i_m\ :i_1+i_2+\ldots+i_m=n} r_{i_1} r_{i_2} \ldots r_{i_m}, \tag{8.52}$$

où la somme s'étend sur toutes les combinaisons d'indices dont la somme vaut n. Mais ce nombre est précisément égal au facteur de q^{-n} dans le développement de D. Autrement dit

$$D = \left(\sum_{k=1}^{s} r_k q^{-k}\right)^m = \sum_{n=m}^{n=sm} \nu(n) q^{-n}, \tag{8.53}$$

Or, le fait que le code soit déchiffrable impose évidemment que les messages possibles de longueur n soient tous différents. Leur nombre ne peut donc pas dépasser q^n, étant donnée la taille du vocabulaire. Par conséquent $\nu(n) \leq q^n$ et on a

$$\left(\sum_{k=1}^{s} r_k q^{-k}\right)^m \leq \sum_{n=m}^{n=sm} q^n q^{-n} = ms - m + 1 \leq ms \tag{8.54}$$

ou encore

$$\sum_{k=1}^{s} r_k q^{-k} \leq (ms)^{\frac{1}{m}}, \tag{8.55}$$

relation qui doit être vraie quel que soit m (toutes les extensions du code doivent être régulières), et par conséquent en passant à la limite pour $m \to \infty$ on a

$$\sum_{k=1}^{s} r_k q^{-k} \leq 1, \tag{8.56}$$

qui est identique à l'inégalité de Kraft (8.49) où on a groupé les termes correspondant à $n_i = k$ supposés en nombre égal à r_k. $\square$

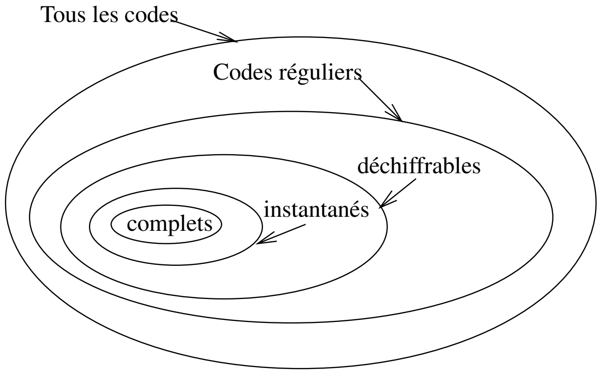


Figure 8.6. Types de codes

Codes complets. Un code instantané est dit complet si, après sa construction, il n'est plus possible de former de nouveaux mots de longueur inférieure ou égale à s tout en conservant le caractère instantané du code. On montre que la condition nécessaire et suffisante d'existence d'un code complet est l'égalité

$$\sum_{i=1}^{Q} q^{-n_i} = 1. \tag{8.57}$$

La figure 8.6 représente les relations entre les différents types de codes que nous venons de discuter.

8.7 PREMIER THEOREME DE SHANNON

8.7.1 Limite inférieure de la longueur moyenne du code

Soit une source stationnaire sans mémoire, montrons tout d'abord que la longueur moyenne des mots nécessaires au codage déchiffrable de cette source est nécessairement supérieure ou égale à $H(S)/\log q$. Autrement dit

$$H(S) \leq \log q \sum_{i=1}^{Q} P(s_i) n_i. \tag{8.58}$$

où $P(s_i)$ désigne la probabilité du symbole s_i de la source, et n_i la longueur du mot de code q-aire qui lui est attribué.

En effet, supposons disposer d'un code déchiffrable; celui-ci devant vérifier l'inégalité de Kraft, on a

$$0 < \mathcal{Q} = \sum_{i=1}^{Q} q^{-n_i} \leq 1, \tag{8.59}$$

On peut donc définir les nombres $q_i = \frac{q^{-n_i}}{\mathcal{Q}}$ dont la somme vaut 1 et appliquer l'inégalité de Gibbs

$$\sum_{i=1}^{Q} P(s_i) \log \frac{q_i}{P(s_i)} \leq 0 \tag{8.60}$$

ce qui revient à dire que

$$-\sum_{i=1}^{Q} P(s_i) \log P(s_i) \leq -\sum_{i=1}^{Q} P(s_i) \log \frac{q^{-n_i}}{\mathcal{Q}} = \log q \sum_{i=1}^{Q} P(s_i) n_i + \log \mathcal{Q}, \tag{8.61}$$

en d'autres mots

$$H(S) \leq \log q \sum_{i=1}^{Q} P(s_i) n_i, \tag{8.62}$$

puisque $\log \mathcal{Q} \leq 0$. $\square$

Nous noterons dans la suite la longueur moyenne du code par $\overline{n}$.

8.7.2 Borne supérieure de la limite supérieure (code de Shannon)

Montrons qu'il est possible de trouver un code déchiffrable tel que

$$\overline{n} < \frac{H(S)}{\log q} + 1, \tag{8.63}$$

En effet, prenons comme longueur n_i du mot de code réservé au symbole s_i le plus petit nombre entier n_i tel que $q^{-n_i} \leq P(s_i)$. Donc n_i est le plus petit entier supérieur à $-\log_q P(s_i)$. En introduisant la notation $\lceil x \rceil$ pour désigner le plus petit entier supérieur où égal à x, on a

$$n_i = \lceil -\log_q P(s_i) \rceil.$$

Par conséquent, $\forall i = 1, \ldots Q$ on a

$$\frac{-\log P(s_i)}{\log q} \leq n_i = \lceil -\log_q P(s_i) \rceil < \frac{-\log P(s_i)}{\log q} + 1 \tag{8.64}$$

et en multipliant par $P(s_i)$ et en sommant sur i on obtient

$$\frac{H(S)}{\log q} \leq \overline{n} < \frac{H(S)}{\log q} + 1, \tag{8.65}$$

et la longueur moyenne du code répond aux spécifications. Nous dirons dans la suite que tout code qui vérifie (8.65) est *compact*. Le code dont nous venons de spécifier les longueurs de mots s'appelle le *code de Shannon*.

De plus, les longueurs n_i ainsi construites sont conformes à l'inégalité de Kraft. En effet,

$$q^{-n_i} \leq P(s_i) \Rightarrow \sum_{i=1}^{Q} q^{-n_i} \leq \sum_{i=1}^{Q} P(s_i) = 1,$$

ce qui garantit bien l'existence d'un code déchiffrable et/ou instantané. □

Ce théorème nous indique donc qu'il est possible d'approcher (sans recourir à l'extension de la source) la borne inférieure à un symbole q-aire près. Il ne nous indique pas, cependant comment construire ce code, ni ne nous permet de dire s'il est possible d'approcher de plus près la limite en faisant appel à d'autres longueurs de mots. Cependant, nous allons voir que ce résultat suffit pour montrer qu'il est possible en faisant appel à l'extension de la source d'approcher la limite d'aussi près que souhaité, ce que nous savons déjà suite à l'exploitation du théorème AEP (dans le cas binaire).

Notons que l'inégalité de Kraft étant une condition suffisante d'existence de code instantané, on déduit de ce qui précède qu'il est également possible de trouver un code instantané *compact* (ayant les mêmes longueurs de mots que le code déchiffrable). Cela justifie donc l'interêt particulier accordé aux codes instantanés.

8.7.3 Code absolument optimal

On dit que le code est absolument optimal si

$$\overline{n} = \frac{H(S)}{\log q}. \tag{8.66}$$

8.7.4 Extension de la source et théorème du codage sans bruit

Le premier théorème de Shannon s'énonce comme suit.

Codage de source. *Pour toute source stationnaire, il existe un procédé de codage déchiffrable, où la longueur moyenne $\overline{n}$ des mots est aussi voisine que l'on veut de sa borne inférieure $H(S)/\log q$.*

Démonstration. Si la source est sans mémoire, on peut écrire (8.65) pour sa k-ème extension, ce qui donne

$$\frac{kH(S)}{\log q} \leq \overline{n^k} < \frac{kH(S)}{\log q} + 1, \tag{8.67}$$

où $\overline{n^k}$ est la longueur moyenne des mots de code utilisés pour coder la k-ème extension. Divisée par k, cette double inégalité montre que $\frac{\overline{n^k}}{k} \to \frac{H(S)}{\log q}$. □

Si la source est stationnaire, le même raisonnement peut s'appliquer au codage des messages de longueur s, à savoir que

$$\frac{H_s(S)}{\log q} \leq \overline{n^s} < \frac{H_s(S)}{\log q} + 1 \tag{8.68}$$

où $H_s(S) = H(S_1, \ldots, S_s)$ désigne l'entropie conjointe de messages composés de s symboles successifs émis par la source.

Donc, en divisant par s et en prenant la limite pour $s \to \infty$ on obtient

$$\lim_{s\to\infty} \frac{\overline{n^s}}{s} = \frac{H(S)}{\log q}. \tag{8.69}$$

pour autant que $H(S) \triangleq \lim_{s\to\infty} \frac{H_s(S)}{s}$ existe, ce qui est toujours le cas pour les processus stationnaires.□

8.7.5 Remarques

Débits. La capacité d'un canal sans erreurs (ou sans bruit) est définie comme le maximum d'information qu'il est possible de transmettre sur ce canal par symbole. On a pour un canal C utilisant un alphabet q-aire

$$C(C) = \log q. \tag{8.70}$$

Le codage déchiffrable permet en principe de coder une source stationnaire utilisant un alphabet fini quelconque de manière à ce que le nombre moyen de symboles émis par ce code pour un symbole de la source s'approche de la limite

$$\frac{H(S)}{\log q}. \tag{8.71}$$

Si la source émet en moyenne les symboles avec un débit de D_S symboles Q-aires par seconde, cela donne lieu à un débit de

$$D = D_S \frac{H(S)}{\log q} \tag{8.72}$$

symboles q-aires par seconde à transmettre sur le canal. En d'autres termes, pour que le canal puisse suivre le rythme de la source, il faut que son débit D_C soit supérieur ou égal à cette valeur. Il faut donc que

$$\frac{D_C}{D_S} \geq \frac{H(S)}{C(C)}. \tag{8.73}$$

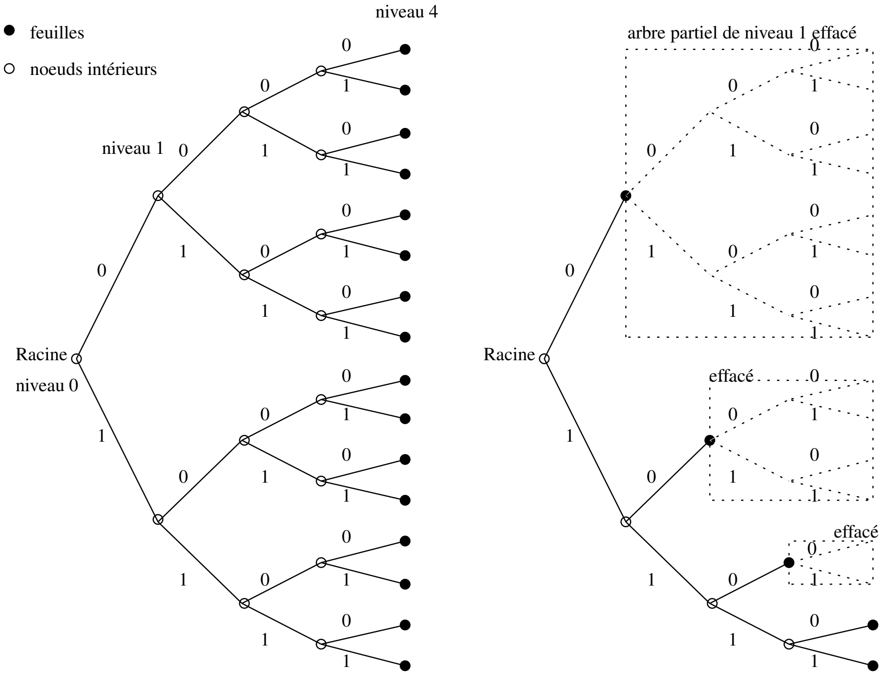
Nous verrons au chapitre suivant que pour un canal avec bruit cette inégalité reste valable à condition d'utiliser pour la capacité la définition suivante

$$C(C) = \max_{P(\mathcal{X})} I(\mathcal{X}; \mathcal{Y}) \tag{8.74}$$

où la quantité d'information mutuelle de la sortie $\mathcal{Y}$ et de l'entrée $\mathcal{X}$, est maximisée par rapport aux distributions de probabilités de l'entrée du canal. Cette formule dégénère en

$$C(C) = \log q - H(\mathcal{U}|\mathcal{Y}), \tag{8.75}$$

pour les canaux dits symétriques (qui réalisent ce maximum pour une entrée $\mathcal{X}$ distribuée uniformément : $\mathcal{X} = \mathcal{U}$). Notons d'emblée que le résultat satisfait bien les conditions limites du canal sans bruit puisque dans ce cas $\mathcal{X} = \mathcal{Y}$ et $H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{X}|\mathcal{X}) = 0$.



(a) arbre complet (b) arbre incomplet

Figure 8.7. Exemples d'arbres binaires de profondeur 4

8.8 CODAGE INSTANTANE OPTIMAL

Nous allons maintenant décrire la méthode mise au point par Huffman en 1952, pour la construction d'un code instantané compact. Nous passons par quatre étapes sucessives : (i) nous montrons que tout code instantané peut être representé par un arbre q-aire éventuellement incomplet; (ii) nous re-déduisons ensuite des propriétés des arbres incomplets le fait que l'inégalité de Kraft est une condition nécessaire d'existence de code instantané avant de démontrer qu'il s'agit aussi d'une condition suffisante; (iii) nous attachons des probabilités aux noeuds de l'arbre d'un code en fonction des probabilités de la source; (iv) nous montrons enfin comment obtenir un code optimal par la méthode de Huffman.

8.8.1 Codes instantanés et arbres q-aires

Soient $n > 0$ et $q > 1$ deux entiers. Dans la suite n représentera la longueur maximale des mots d'un code utilisant q symboles.

Arbre complet.

Un arbre q-aire *complet* de profondeur (on peut dire aussi hauteur) n est alors un arbre, tel qu'illustré à la figure 8.7(a) (pour $q = 2$ et $n = 4$). Il s'agit d'un graphe construit à partir d'un noeud de départ appelé *racine*. De la racine (noeud de niveau 0) partent q branches (ou arcs) vers q nouveaux noeuds dits de niveau 1; de chaque noeud de niveau 1 partent à nouveau q branches, donnant lieu à q^2 noeuds de niveau 2, et ainsi de suite, jusqu'à obtenir q^n noeuds de niveau n. On convient qu'une branche reliant un noeud de niveau $i - 1$ à un noeud de niveau i est orientée vers ce dernier, et on dit que celui-ci est l'extrémité de la branche et que le noeud de niveau $i - 1$ est son origine. Toute suite de branches consécutives dans l'arbre est appelée *chemin* (deux branches sont consécutives si l'extrémité de la première est l'origine de la seconde).

Un noeud de niveau i est donc un noeud relié à la racine par un chemin de longueur i. Il y a q^i noeuds de niveau $i \leq n$ dans un arbre q-aire complet de profondeur n, et q^i branches sont issues des noeuds de niveau $i - 1$. Une *feuille* d'un arbre est un noeud qui n'est l'origine d'aucune branche; dans un arbre complet de profondeur n les feuilles correspondent aux q^n noeuds de niveau n. Deux noeuds (différents) n et n' sont *reliés* s'il existe un chemin passant par n et n' : deux noeuds de même niveau ne sont jamais reliés. Si n et n' sont reliés on dit que celui de niveau supérieur est le successeur, l'autre le parent.

On peut définir une correspondance bijective entre les suites de n symboles construites sur un alphabet d'ordre q et les chemins reliant la racine aux feuilles d'un arbre complet, en étiquetant les q branches issues des noeuds intérieurs avec les q symboles de l'alphabet (voir figure 8.7).

Arbres incomplets.

Un arbre *incomplet* est le graphe obtenu en *effaçant* d'un arbre complet tous les successeurs d'un certain nombre de ses noeuds, et toutes les branches touchant ces noeuds. Les noeuds dont on a effacé les successeurs deviennent alors des feuilles. La figure 8.7(b) illustre un arbre incomplet obtenu en effaçant les successeurs de trois noeuds.

Notons qu'il existe des arbres qui ne sont ni complets ni incomplets. Mais dans un arbre complet ou incomplet le nombre de branches B et le nombre de feuilles F sont reliés par l'equation suivante

$$B = (F-1)\frac{q}{q-1}, \tag{8.76}$$

et réciproquement, un arbre qui vérifie (8.76) est complet ou incomplet. Donc, puisque les deux nombres B et F sont entiers, on en déduit que le nombre de branches est multiple de q (évident) et que le nombre de feuilles diminué de 1 est multiple de $q-1$ (c'est vrai pour un arbre de profondeur 1, et cela le reste à chaque fois qu'on développe une feuille, puisque cette opération remplace une feuille par un noeud interne et ajoute q nouvelles feuilles à l'arbre, soit au total une augmentation du nombre de feuilles par $q-1$). Nous suggérons au lecteur de démontrer la propriété (8.76) à titre d'exercice.

Par exemple, l'arbre de la figure 8.7(a) comporte $F = 16$ feuilles et $B = 30$ branches, et on vérifie que $30 = 15.2$. L'arbre de la figure 8.7(b) comporte $F = 5$ feuilles et $B = 8$ branches, et on vérifie à nouveau que $8 = 4.2$.

Nous allons voir que tout code instantané peut être représenté par un arbre complet ou incomplet, et que tout arbre complet ou incomplet définit un code instantané.

8.8.2 Arbres, codes instantanés et inégalité de Kraft

Rappelons d'abord qu'un code instantané est un code régulier tel qu'aucun mot ne soit un préfixe d'un autre.

Soient alors $m_i, i = 1, \ldots, Q$ les mots d'un code instantané de longueur maximale n. Chaque mot de ce code peut être (de toute évidence) réprésenté par un chemin partant de la racine de l'arbre complet. Soit alors m_i un des mots de ce code, et n_i sa longueur : aucun autre mot du code ne peut avoir ce mot comme préfixe, et nous pouvons effacer de l'arbre complet les successeurs du noeud correspondant au dernier symbole du mot m_i. Par exemple, l'arbre incomplet de la figure 8.7(b) est ainsi obtenu en supposant que le code contenait les cinq mots $0, 10, 110, 1111, 1110$. Chaque fois que nous introduisons un mot de longueur n_i nous effaçons ainsi q^{n-n_i} feuilles de l'arbre complet, qui sont remplacées par une seule feuille de l'arbre incomplet.

Par conséquent, si nous supposons que le code existe, c'est qu'il est possible d'effectuer cette opération jusqu'au dernier mot m_Q. Nous aurons alors effacé au total $\sum_{i=1}^{Q} q^{n-n_i}$ feuilles de l'arbre complet. Comme celui-ci en comportait au total q^n il faut que

$$\sum_{i=1}^{Q} q^{n-n_i} \leq q^n, \tag{8.77}$$

ce qui implique l'inégalité de Kraft (8.49) en divisant par q^n. Nous avons donc (re)démontré que l'inégalité de Kraft est une condition nécessaire d'existence de code instantané.

Montrons maintenant qu'elle est aussi une condition suffisante. Tout d'abord, l'inégalité de Kraft implique évidemment (8.77), qui peut se réécrire sous la forme

$$\sum_{i=1}^{n} r_i q^{n-i} \leq q^n, \tag{8.78}$$

où on a groupé les termes correspondant aux mots de code de longueur i (r_i désigne leur nombre).

L'inégalité de Kraft implique également, $\forall p \leq n$,

$$\sum_{i=1}^{p} r_i q^{p-i} \leq q^p. \tag{8.79}$$

Pour s'en convaincre il suffit de multiplier (8.78) par q^{p-n} et tronquer la somme au $p-$ème terme.

Montrons alors, en nous servant de (8.79), qu'il est possible de construire un code instantané. En effet, nous avons tout d'abord besoin de r_1 mots de longueur 1. Ce choix est possible puisque (8.79) pour $p = 1$ implique que $r_1 \leq q$, et nous pouvons réserver r_1 symboles du code. Ensuite, nous devons choisir r_2 mots de longueur 2, qui ne commencent pas par l'un des r_1 symboles déjà utilisés. Notons qu'au maximum il sera ainsi possible de construire $(q - r_1)q$ mots de longueur 2 qui vérifient cette condition. Or, pour $p = 2$ (8.79) implique que $r_1 q + r_2 \leq q^2$, autrement dit $r_2 \leq (q - r_1)q$.

Supposons alors que nous ayons pu ainsi choisir $r_1, \ldots, r_{p-1}$ mots de code de longueur $1, \ldots, p-1$, et montrons qu'il est encore possible de choisir r_p mots de longueur p. Nous avions au départ q^{p-1} séquences possibles de longueur $p-1$, et les $p-1$ premières séries de choix en ont éliminé exactement

$$\sum_{i=1}^{p-1} r_i q^{p-1-i} \tag{8.80}$$

et il en reste donc $q(q^{p-1} - \sum_{i=1}^{p-1} r_i q^{p-1-i})$ mots possibles de longueur p. Or (8.79) implique que ce nombre est supérieur ou égal à r_p. Nous avons donc bien démontré que l'inégalité de Kraft est une condition *suffisante* d'existence d'un code instantané et nous avons par la même occasion montré comment construire un tel code.□

Remarquons enfin que la condition

$$\sum_{i=1}^{n} r_i q^{-i} = 1 \tag{8.81}$$

implique qu'arrivé à la fin de la procédure il n'y a plus de mots de code disponibles. En d'autres termes cela veut dire que tout l'arbre incomplet est exploité, et toutes ses feuilles correspondent à un mot de code, lorsque celui-ci est complet.

8.8.3 Probabilités des noeuds

Nous allons ci-dessous raisonner sur les probabilités de parcourir certaines parties d'un arbre de codage. Nous introduisons donc la notion de probabilité d'un noeud d'un arbre de code.

Supposons que $Q-1$ (Q étant le nombre de symboles de la source qu'on souhaite coder) soit multiple de $q-1$ (q étant le nombre de symboles de l'alphabet du code). Si ce n'était pas le cas, on peut ajouter quelques symboles fictifs à la source (au plus $q-2$) de probabilité nulle. On suppose de plus que le nombre total de symboles de la source S de probabilité nulle est inférieur à $q-1$. Nous verrons plus loin ce que cette condition signifie.

Alors supposons disposer d'un arbre complet ou incomplet qui satisfait la relation (8.76), avec un nombre de feuilles F égal au nombre Q de symboles de la source. Puisque $Q-1$ est multiple de $q-1$, le nombre de branches B de l'arbre est bien un nombre entier. Il est donc possible d'associer à la source un arbre complet ou incomplet, dont les feuilles correspondent aux symboles de la source. Nous associons à ces feuilles la probabilité $P(S_i)$ du symbole en question, et nous propageons ces probabilités vers le haut en sommant. Nous associons donc à tout noeud intérieur une probabilité qui est égale à la somme de probabilités des feuilles qui sont des noeuds successeurs de celui-ci. En particulier, nous associons au noeud racine la probabilité $\sum_{i=1}^{Q} P(S_i) = 1$.

8.8.4 Algorithmes de construction de codes

Remarquons tout d'abord que si le code est optimal alors les mots les plus courts doivent être associés aux symboles les plus probables de la source. En d'autres mots, si on connaît les longueurs des mots du code, il suffit pour trouver le codage de les ranger par longueur croissante et de leur associer les symboles de la source par ordre de probabilité décroissante.

Par ailleurs, si nous supposons que les probabilités de la source peuvent s'écrire sous la forme

$$P(S_i) = q^{k_i - K}. \tag{8.82}$$

Alors un codage optimal est obtenu comme suit :

1. Construire l'arbre complet de profondeur K.
2. Associer au symbole S_i un noeud de profondeur $K - k_i$ et effacer les successeurs de ce noeud.

La longueur moyenne de ce code vaut

$$\overline{n} = \sum_{i=1}^{Q} (K - k_i) q^{k_i - K} \tag{8.83}$$

et la limite de Shannon vaut

$$\frac{H(S)}{\log q} = \frac{\sum_{i=1}^{Q} q^{k_i - K} \log q^{K - k_i}}{\log q} = \overline{n}. \tag{8.84}$$

D'autre part le fait que $\sum_{i=1}^{Q} P(S_i) = 1$ joint à la condition (8.82) assure que l'inégalité de Kraft est satisfaite (avec le signe $=$).

Passons maintenant aux choses sérieuses. Supposons que les probabilités prennent des valeurs quelconques, et donnons nous comme point de départ un arbre incomplet quelconque dont le nombre de feuilles est égal au nombre de symboles de la source (éventuellement augmentée au moyen de symboles "impossibles" pour rendre la chose possible). Associons aux feuilles de cet arbre les probabilités des symboles de la source de manière tout à fait quelconque.

Cet arbre n'a évidemment aucune raison d'être optimal. Cependant, nous pouvons l'améliorer petit à petit en échangeant ses sous-arbres de la manière suivante :

1. considérer toutes les paires de sous-arbres telles que la profondeur de la racine du premier soit strictement inférieure à celle du second, et trouver une paire telle que la probabilité de la racine du premier soit strictement inférieure à celle de la racine du second.
2. si une paire de sous-arbres vérifiant la condition a été trouvée, échanger les deux sous-arbres (en faisant transiter le plus probable vers le haut, le moins probable vers le bas), et recommencer toute l'opération avec le nouvel arbre ainsi produit.
3. sinon, arrêter la procédure et renvoyer comme résultat l'arbre ainsi modifié.

Notons tout d'abord que l'opération est en principe faisable et tous les arbres intermédiaires sont des arbres complets ou incomplets, car l'opération d'échange ne détruit pas cette propriété. De plus, les arbres ainsi parcourus forment une suite d'arbres dont les profondeurs moyennes sont strictement décroissantes (car l'opération d'échange est faite dans des conditions qui le garantissent). De ce fait tous ces arbres sont différents (pas de possibilité de cycles). Tous ces arbres correspondent donc à des codes instantanés dont la longueur moyenne est bornée inférieurement par la limite de Shannon $\frac{H(S)}{\log q}$. D'autre part, comme le nombre total d'arbres complets ou incomplets ayant Q feuilles est évidemment fini, ce procédé doit nécessairement s'arrêter après un nombre fini d'étapes. Par conséquent, la procédure renvoie un arbre après un nombre fini d'étapes, et cet arbre est tel qu'aucun échange de deux de ses sous-arbres ne permette de réduire la longueur moyenne du code.

Cet arbre "optimisé" possède la propriété suivante :

Pour toute paire de noeuds de niveaux différents, celui qui est le moins profond est au moins aussi probable que l'autre.

En d'autres mots, l'ordre de profondeur respecte l'ordre inverse des probabilités. En particulier, les mots (correspondant aux feuilles) triés par ordre décroissant de longueur doivent respecter l'ordre inverse des probabilités. Cette propriété donne une *condition nécessaire* d'optimalité.

A ce stade de notre raisonnement, rien ne garantit qu'un arbre optimisé de cette façon à partir d'un point de départ quelconque donne lieu à une longueur moyenne minimale. Tout ce que nous pouvons dire, c'est qu'il s'agit d'un minimum local, c'est-à-dire non améliorable au moyen d'une seule opération d'échange. Cela étant, un arbre globalement optimal doit néanmoins, et nécessairement, satisfaire à la même propriété. En effet, si ce n'était pas le cas on pourrait encore l'améliorer à l'aide de notre méthode d'optimisation, et il ne serait donc pas optimal.

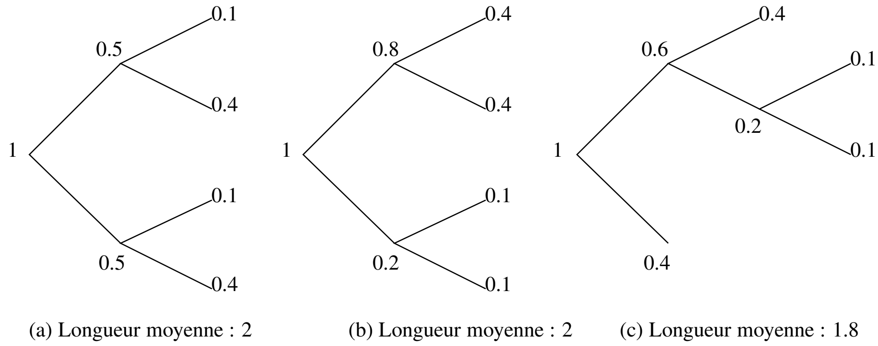


Figure 8.8. Arbres binaires pour un alphabet de source d'ordre 4

Montrons que l'algorithme proposé peut ne pas produire un arbre optimal à l'aide du contre-exemple représenté à la figure 8.8 (les chiffres associés aux noeuds correspondent aux probabilités des ensembles de mots attachés aux feuilles des sous-arbres correspondants). Ces trois arbres correspondent à 3 codes instantanés pour une source dont les probabilités de symboles sont $0.1, 0.1, 0.4, 0.4$ et dont l'entropie vaut 1.722 Shannon. On voit que les deux premiers codes sont équivalents du point de vue de leur longueur moyenne, et le troisième est légèrement meilleur. Or l'arbre de la figure 8.8(a) vérifie la critère d'arrêt de notre algorithme. Donc, si cet arbre est fourni comme point de départ à l'algorithme il ne peut être amélioré. De toute évidence il n'est pas optimal puisque l'arbre de la figure 8.8(c) est meilleur. Remarquons, par ailleurs, que l'arbre de la figure 8.8(b), qui peut être obtenu à partir de celui de la figure 8.8(a) par simple échange de deux feuilles de même niveau, est améliorable puisque son sous-arbre inférieur est de niveau inférieur aux deux feuilles supérieures, alors qu'il est moins probable. Si nous le donnons comme point de départ à notre algorithme, celui-ci va l'améliorer pour fournir un arbre similaire à celui de la figure 8.8(c).

Nous allons voir que l'arbre de la figure 8.8(c) est en fait un arbre optimal, tel que ceux produits par l'algorithme de Huffman, décrit ci-dessous dans le cas binaire. Notons d'abord qu'il vérifie notre condition nécessaire d'optimalité. De plus il possède d'autres propriétés intéressantes : les deux symboles les moins probables correspondent à des feuilles de profondeur maximale (conséquence directe de la précédente), et surtout ces deux feuilles ont un même père.

8.8.5 Algorithme de Huffman pour un alphabet de code binaire

Décrivons d'abord l'algorithme, puis montrons qu'il produit un code optimal. Remarquons tout d'abord que si $q = 2$, il n'est pas nécessaire d'augmenter la source en y ajoutant de nouveaux symboles (tout nombre $Q - 1$ étant multiple de 1).

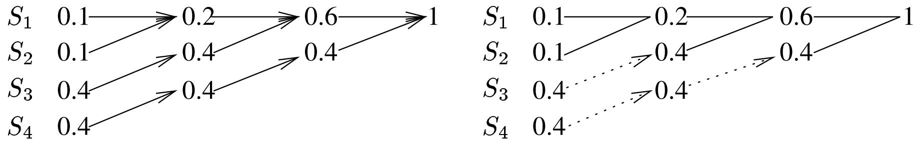
8.8.5.1 Algorithme.

La méthode de Huffman (publiée en 1952) procède en construisant l'arbre en partant des feuilles les plus profondes, c'est-à-dire en regroupant progressivement les symboles de la source en fonction de leurs probabilités, jusqu'à aboutir à un ensemble contenant tous les symboles, qui correspondra à la racine de l'arbre. A chaque étape la méthode fusionne les q feuilles les moins probables en les remplaçant par une feuille. Le parcours inverse fournit alors l'arbre du code.

Montrons sur l'exemple de la source codée sur la figure 8.8 comment l'algorithme procède ($P(S_1) = P(S_2) = 0.1, P(S_3) = P(S_4) = 0.4$)

On identifie tout d'abord les deux symboles les moins probables, soit ici S_1, S_2. On les regroupe pour former une nouvelle source (on dira qu'on réduit la source) de probabilités ($P(S_1') = 0.2, P(S_2') = P(S_3') = 0.4$). On applique ensuite l'algorithme récursivement à cette source : on regroupe donc S_1' avec l'un quelconque de S_2' et S_3', ceux-ci étant équiprobables (disons S_2'). On dispose alors d'une nouvelle source de deux symboles ($P(S_1'') = 0.6, P(S_2'') = 0.4$)), qui une fois regroupés terminent l'algorithme.

La figure 8.9 représente graphiquement le fonctionnement de l'algorithme dans le cas d'un code binaire. La figure (8.9)(a) décrit les regroupements successifs; la figure (8.9)(b) indique comment l'arbre de codage s'en déduit.



(a) Construction du code

(b) Obtention de l'arbre

Figure 8.9. Illustration de l'algorithme de Huffman binaire

On constate que l'arbre produit par cet algorithme est équivalent à celui de la 8.8(c); sa longueur moyenne vaut 1.8 bits. Sa redondance

$$\frac{\overline{n} - \frac{H(S)}{\log q}}{\overline{n}}$$

est de 4.3%.

8.8.5.2 Démonstration de l'optimalité du code de Huffman.

Puisque l'algorithme est récursif, nous allons utiliser un raisonnement par induction pour prouver qu'il est optimal. L'idée est de déduire de l'hypothèse que l'algorithme est optimal pour une source ayant $Q-1$ symboles, qu'il doit nécessairement alors aussi l'être pour une source ayant Q symboles. Ajouté au cas de base (une source de deux symboles), pour lequel l'algorithme produit évidemment le code trivial qui est aussi optimal, cela démontrera l'optimalité de l'algorithme.

Il est important de se rendre compte qu'il n'y a pas qu'un seul code optimal: par exemple partant d'un code optimal on peut en construire de nombreux autres en permutant les mots de même longueur et/ou en permutant les symboles du code. La méthode de Huffman produit un de ces codes optimaux. Pour le démontrer, commençons par résumer ce que nous pouvons dire des propriétés qu'un code optimal particulier doit satisfaire. Sans perte de généralité, nous supposerons que les symbole de source sont ordonnés par ordre décroissant de probabilités $P(S_1) \geq P(S_2) \geq \ldots \geq P(S_Q) > 0$.

Lemme. *Pour toute source, il existe un code optimal instantané (de longueur moyenne minimale) qui satisfait les propriétés suivantes:*

1. *Si $P(S_j) > P(S_k)$ alors $n_j \leq n_k$.*

2. *Les deux mots les plus longs ont la même longueur.*

3. *Les deux mots les plus longs ne diffèrent que d'un bit, et correspondent aux deux mots les moins probables.*

En effet :

1. Est un cas particulier de la propriété plus générale indiquée au §8.8.4. Si elle n'était pas vérifiée pour deux symboles on pourrait échanger les deux mots de codes correspondants et ainsi améliorer le code.

2. Si ce n'est pas le cas, alors une des deux feuilles de profondeur maximale de l'arbre incomplet correspondant au code ne correspondrait pas à un symbole. On pourrait alors effacer ces deux feuilles et associer le symbole source au noeud père, ce qui réduirait encore la longueur du code. On en déduit aussi que tous les mots de profondeur maximale doivent avoir un sibling (c'est-à-dire, qu'il doit exister un noeud de même profondeur qui partage le même père). De plus, la première propriété implique que les mots de longueur maximale correspondent aux symboles les moins probables.

3. Si le code est optimal, le mot le moins probable est de profondeur maximale. Soit alors le deuxième mot le moins probable, seulement trois possibilités sont compatibles avec l'optimalité du code : (i) il est le sibling du premier; (ii) il n'est pas le sibling mais est une feuille de même profondeur; (iii) il est de profondeur inférieure mais de même probabilité que le sibling. Dans les deux derniers cas on peut échanger ce symbole et le sibling du mot le moins probable sans changer la longueur moyenne du code. Il est donc toujours possible de s'arranger pour satisfaire la dernière condition.

Nous avons donc démontré que si $P(S_1) \geq P(S_2) \geq \ldots \geq P(S_Q) > 0$, alors il existe un code optimal avec des longeurs $n_1 \leq n_2 \leq \ldots \leq n_Q$, et tel que les mots de code m_{Q-1} et m_Q ne diffèrent que dans leur dernier bit.

Soit alors un code C_Q qui vérifie cette condition pour une source de taille Q, considérons la réduction de ce code, notée C_{Q-1}, obtenue en utilisant les $Q-2$ premiers mots de C_Q (associés aux symboles $S_1, \ldots, S_{Q-2}$), et comme $(Q-1)-$ème mot le préfixe commun des mots m_{Q-1} et m_Q, associé à un symbole (fictif) S'_{Q-1} de probabilité $P(S_{Q-1}) + P(S_Q)$.

La longueur moyenne du code C_Q (notons la $\overline{n_Q}$) peut alors s'écrire en fonction de la longueur moyenne du code réduit par

$$\overline{n_Q} = \overline{n_{Q-1}} + P(S_{Q-1}) + P(S_Q). \tag{8.85}$$

Cette différence ne dépend pas du choix du code utilisé pour la source réduite. On en déduit que le code C_Q est optimal si, et seulement si, le code C_{Q-1} l'est aussi. □

Nous avons donc démontré que l'algorithme *local* qui fusionne à chaque étape les deux symboles les moins probables conduit effectivement à un code optimal.

8.8.5.3 Extension au cas d'un alphabet d'ordre quelconque.

L'algorithme reste optimal pour un alphabet de code q-aire quelconque. On y regroupe alors à chaque étape les q symboles les moins probables. Avant de l'appliquer, il faut cependant augmenter le nombre de symboles de la source par des symboles fictifs de probabilité nulle, de façon à satisfaire la condition $Q-1$ multiple de $q-1$.

Par exemple, montrons comment on obtiendrait un code ternaire ($q = 3$) pour notre source ci-dessus. D'abord, ajoutons un symbole fictif de probabilité nulle ce qui donne les probabilités : $P(S_0) = 0, P(S_1) = P(S_2) = 0.1, P(S_3) = P(S_4) = 0.4$. On regroupe donc d'abord les trois premier symboles, donnant $P(S'_1) = 0.2, P(S'_2) = P(S'_3) = 0.4$. Le regroupement de ces trois derniers symboles donne le code. Sa longeur moyenne est 1.2 symboles ternaires. La limite inférieure de Shannon pour le code ternaire vaut $\frac{H(S)}{\log 3} = 1.086$. Sa redondance η est d'environ 9.5%.

Utilisant l'algorithme pour construire un code sur un alphabet quaternaire donne évidemment le code trivial qui associe un mot de code de longueur 1 à chaque symbole. Sa longueur moyenne vaut 1, à comparer avec $\frac{H(S)}{\log 4} = 0.861$. Sa redondance vaut par conséquent environ 14%.

On comparera, à titre d'exercice, pour les trois valeurs de q illustrées l'efficacité du code de Huffman au codage de Shannon (qui associe une longueur de mot de $\lceil -\log_q P(S_i) \rceil$ symboles q-aires au symbole de source S_i).

8.8.5.4 Codage et questionnaires.

Remarquons tout d'abord que l'algorithme de Huffman construit un code minimisant

$$\overline{n} = \sum_{i=1}^{Q} n_i P(S_i), \tag{8.86}$$

où les $P(S_i)$ somment à 1, mais qu'il peut évidemment être aussi appliqué à la minimisation de

$$\sum_{i=1}^{Q} n_i x_i \tag{8.87}$$

où les x_i sont quelconques positifs ou nuls.

D'autre part, le code de Huffman permet de construire des questionnaires optimaux pour l'identification d'un certain nombre d'objets. En effet, un questionnaire q-aire est équivalent à un code et le nombre moyen de questions est équivalent à la longueur moyenne du code.

Ce code est également à la base d'algorithmes en informatique. Par exemple, si nous voulons construire une table contenant un certain nombre d'objets qui puissent être accédés rapidement nous pouvons nous inspirer de cette idée. La longueur moyenne du code représenterait dans ce cas le nombre d'opérations élementaires nécessaires pour accéder un objet, c'est-à-dire le temps d'accès. On consacrerait alors plus d'opérations aux objets "ésotériques" et moins aux objets fréquemment accédés. Par exemple, on commencerait avec une table de profondeur constante, puis, au fur et à mesure de l'utilisation de la table, on relèverait les statistiques d'utilisation et réorganiserait la table pour minimiser les temps d'accès moyens.

8.9 RESUME

Nous avons consacré ce chapitre à l'étude de modèles de sources discrètes et au codage de source.

Nous nous sommes limités essentiellement à l'étude de sources stationnaires. Nous avons commencé par étudier le modèle le plus simple de source dit source sans mémoire, qui émet des symboles successifs indépendants et distribués identiquement. Nous avons dans ce contexte énoncé et démontré le théorème AEP, qui nous a permis de mieux entrevoir la structure de l'ensemble des messages émis par une source sans mémoire. A partir de cette compréhension nous avons une première fois démontré la possibilité d'atteindre la limite de Shannon de compression de données, sans toute fois démontrer qu'il s'agissait d'une limite ultime. Rappelons que ce raisonnement a fait appel au codage de messages longs, et rappelons également que le théorème AEP reste valable pour des sources stationnaires quelconques, pour autant qu'elles soient ergodiques.

Notre étude s'est ensuite focalisée sur la famille des sources de Markov, qui présentent la caractéristique de disposer d'une mémoire finie. Nous avons également donné les conditions sous lesquelles une telle source était stationnaire et dans quel cas elle était ergodique. Enfin, nous avons fourni une définition générale de l'entropie par symbole d'une source stationnaire, et en particulier d'une source de Markov.

Ensuite, nous nous sommes tournés vers le codage à proprement parler. Nous avons énuméré un certain nombre de propriétés souhaitables des codes et démontré que l'inégalité de Kraft était une condition nécessaire et suffisante d'existence de code déchiffrable et de code instantané. Partant de cette propriété, nous avons démontré ensuite le premier théorème de Shannon et sa réciproque, valable pour des sources stationnaires quelconques, à nouveau en faisant appel à des messages longs. Cependant, nous avons également mis en évidence une borne supérieure atteignable avec des messages de longueur finie.

Enfin, nous avons terminé ce chapitre par la construction d'un algorithme optimal (l'algorithme de Huffman) qui construit un code de longueur moyenne minimale pour une source donnée, ou une de ses extensions. Ce code est en général sous-optimal si on le compare à la limite de Shannon, et n'atteint en général cette limite que pour autant qu'on l'applique à des extensions d'ordre suffisamment élevé. Il produit en outre un code instantané, c'est-à-dire très facile à décoder.

Enfin, puisque le codage optimal exploite la redondance (issue soit de distributions non uniformes, soit de corrélations entre signaux successifs), on peut dire que le codage a comme sous-produit de transfomer la source en une source uniforme et sans mémoire, et cela d'autant mieux qu'il est proche des conditions d'optimalité énoncées par Shannon. En effet, si ce n'était pas le cas, il serait possible de réduire la longueur moyenne des mots du code en recodant celui-ci une deuxième fois.

On arrive donc à la conclusion que toutes les sources une fois codées de manière optimale se ressemblent et se comportent (vues de l'extérieur) comme si elles émettaient des symboles indépendants et équiprobables de l'alphabet du canal. Nous avons donc effectué notre premier pas vers la normalisation du paradigme de Shannon.

Appendice 8.A: Exercices

1. **Codes de Huffman**

   Soit la variable aléatoire $\mathcal{X}$ dont les valeurs possibles et probabilités sont comme suit :

$$\begin{pmatrix} X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 \\ 0.49 & 0.26 & 0.12 & 0.04 & 0.04 & 0.03 & 0.02 \end{pmatrix}.$$

   (a) Trouver un code de Huffman binaire et la longueur moyenne correspondante

   (b) Trouver un code de Huffman ternaire et la longueur moyenne correspondante

   (c) Comparer ces longueurs moyennes aux bornes prédites par le théorème de Shannon

2. **Mauvais codes.**

   Lesquels parmi les codes suivants ne peuvent pas être un code de Huffman binaire ? Justifier.

   (a) $\{0, 10, 11\}$

   (b) $\{00, 01, 10, 110\}$

   (c) $\{01, 10\}$.

3. **Compression optimale d'une source de Markov simple.**

   Soit une source de Markov (3 états) caractérisée par la matrice de transition

$$\left[\begin{array}{c|ccc} S_{n-1} \backslash S_n & s_1 & s_2 & s_3 \\ \hline s_1 & 1/2 & 1/4 & 1/4 \\ s_2 & 1/4 & 1/2 & 1/4 \\ s_3 & 0 & 1/2 & 1/2 \end{array}\right]$$

   (Donc la probabilité que s_1 suive s_3 est nulle)

   (a) Construire un code composé de 3 codes instantanés binaires C_1, C_2, C_3 pour coder la chaîne de Markov de façon optimale.

   (b) Quelle est la longueur moyenne du symbole suivant, conditionnellement à l'état précédent ?

   (c) Quelle est la longueur moyenne non-conditionnelle ?

   (d) Relier cela à l'entropie par symbole de la chaîne de Markov

4. **Codes déchiffrables et codes instantanés.**

   Soit $N = \sum_{i=1}^{m} p_i n_i^{100}$ l'espérance mathématique de la 100ème puissance des longueurs des mots d'un code pour une variable aléatoire discrète $\mathcal{X}$. Soit $N_1 = \min N$ sur tous les codes instantanés; et soit $N_2 = \min N$ sur tous les codes déchiffrables.

   (a) Quelle relation (inégalité) existe entre N_1 et N_2 ? Justifier.

5. **Mauvais vin.**

   On donne 6 bouteilles de vin. On sait que parmi ces bouteilles exactement une seule a tourné (vinaigre : le goût est infect). A partir de l'inspection visuelle des bouteilles on a pu déterminer que la probabilité p_i qu'il s'agisse de la i-ième bouteille est donnée par

$$(p_1, p_2, p_3, p_4, p_5, p_6) = (\frac{8}{23}, \frac{6}{23}, \frac{4}{23}, \frac{2}{23}, \frac{2}{23}, \frac{1}{23}).$$

   (a) Supposons que vous goutiez le vin, une bouteille à la fois. (Il est déconseillé de consommer la totalité des six bouteilles.)

      Choisissez **l'ordre** de dégustation qui minimise le nombre moyen de dégustations nécessaires pour déterminer la mauvaise bouteille. Notez qu'il n'est pas nécessaire de déguster la dernière bouteille.

      i. Quel est alors le nombre de dégustations nécessaires en moyenne ?

ii. Quelle bouteille doit être dégustée en premier lieu ?

(b) Supposons que vous soyez plus subtil (ayant suivi un cours de théorie de l'information).

Pour la première dégustation vous mélangez le vin issu de certaines bouteilles, et vous dégustez le mélange. Vous poursuivez le raisonnement, et dégustez à chaque étape des mélanges, en vous arrêtant dès que vous êtes convaincu d'avoir identifié la mauvaise bouteille.

i. Quel est maintenant le nombre moyen de dégustations ?

ii. Quel mélange testeriez-vous en premier ?

Appendice 8.B: Suites de v.a. et notions de convergence

Notions de convergence

Il existe différentes façons de définir la notion de convergence de suites de v.a.. Nous les rappelons brièvement ci-dessous en indiquant les relations qui existent entre ces notions, s'il y a lieu.

Convergence en probabilité

Notation: $\xrightarrow{P}$

La suite $(\mathcal{X}_n)$ de v.a. réelles converge en probabilité vers la constante a; si $\forall \epsilon$ et η (arbitrairement petits), $\exists n_0$ tel que $n > n_0$ entraîne

$$P(|\mathcal{X}_n - a| > \epsilon) < \eta. \tag{B.1}$$

On note alors $(\mathcal{X}_n) \xrightarrow{P} a$.

On définit la convergence en probabilité d'une suite de v.a. $(\mathcal{X}_n)$ vers une v.a. $\mathcal{X}$ comme la convergence vers 0 de la suite $(\mathcal{X}_n - \mathcal{X})$.

Convergence presque sûre ou convergence forte

Notation: $\xrightarrow{p.s.}$

La suite $(\mathcal{X}_n)$ de v.a. réelles converge presque sûrement vers $\mathcal{X}$ si :

$$P(\{\omega | \lim_{n\to\infty} \mathcal{X}_n(\omega) \neq \mathcal{X}(\omega)\}) = 0. \tag{B.2}$$

On note alors $(\mathcal{X}_n) \xrightarrow{p.s.} \mathcal{X}$.

La convergence presque sûre implique la convergence en probabilité, c'est pourquoi on l'appelle aussi convergence forte.

Convergence en moyenne d'ordre p

Si $E\{(\mathcal{X}_n - \mathcal{X})^p\}$ existe $\forall n$, alors on a

$(\mathcal{X}_n) \to \mathcal{X}$ en moyenne d'ordre p si $E\{(\mathcal{X}_n - \mathcal{X})^p\} \to 0$.

Le cas pratique usuel est la moyenne quadratique ($p = 2$).

La convergence en moyenne d'ordre p implique la convergence en probabilité.

Convergence en en loi

Notation: $\xrightarrow{\mathcal{L}}$

La suite $(\mathcal{X}_n)$ de v.a. réelles converge en loi vers $\mathcal{X}$ de fonction de répartition $F(\cdot)$ si en tout point de continuité x de $F(\cdot)$, la suite $(F_n(x))$ converge ponctuellement vers $F(x)$. On note

$$(\mathcal{X}_n) \xrightarrow{\mathcal{L}} \mathcal{X}. \tag{B.3}$$

Il s'agit de la convergence la plus faible. En particulier, la convergence en probabilité implique la convergence en loi. Cette dernière est très utilisée en pratique car elle permet d'approximer la fonction de répartition de $(\mathcal{X}_n)$ par celle de $\mathcal{X}$, et réciproquement.

On montre que si $F(\cdot)$ est continue alors la convergence est uniforme (plus que ponctuelle). De plus, si les $F_n(\cdot)$ admettent des densités alors la convergence en loi implique la convergence ponctuelle des densités.

Théoremes de convergence

Moivre-Laplace

Ce théorème utile en statistiques, permet d'approximer une loi binomiale par une loi Gaussienne. Il dit que, si $(\mathcal{X}_n)$ forme une suite de v.a. binomiales $\mathcal{B}(n, p)$, alors

$$\frac{\mathcal{X}_n - np}{\sqrt{np(1-p)}} \xrightarrow{\mathcal{L}} \mathcal{N}(0, 1). \tag{B.4}$$

Théorème central-limite

Ce théorème établit la convergence en loi vers la loi normale d'une somme de v.a. i.i.d. sous des hypothèses très peu contraignantes. Il dit que, si $(\mathcal{X}_n)$ forme une suite de v.a. i.i.d. de moyenne μ et d'écart-type σ (ces deux moments sont donc supposés exister), alors

$$\left(\frac{\sum_{i=1}^{n} \mathcal{X}_i - n\mu}{\sigma\sqrt{n}} \right) \xrightarrow{\mathcal{L}} \mathcal{N}(0, 1). \tag{B.5}$$

On retrouve comme cas particulier le théorème de Moivre-Laplace, en prenant des variables de Bernouilli.

Contre-exemple : loi de Cauchy.

Lois des grands nombres

Loi faible des grands nombres. Soient $\mathcal{X}_i, \forall i = 1, \ldots n$ indépendantes d'espérance μ_i finies et de variances σ_i finies, alors

Si $\frac{1}{n} \sum_{i=1}^{n} \mu_i \to \mu$ *et* $\frac{1}{n^2} \sum_{i=1}^{n} \sigma_i^2 \to 0$, *alors* $\overline{\mathcal{X}}_n \triangleq \frac{1}{n} \sum_{i=1}^{n} \mathcal{X}_i$ *est telle que*

$$\overline{\mathcal{X}}_n \xrightarrow{P} \mu. \tag{B.6}$$

Cas particulier : les v.a. $\mathcal{X}_i$ sont i.i.d. μ, σ. On a alors $\frac{1}{n} \sum_{i=1}^{n} \mu_i = \mu$ et $\frac{1}{n^2} \sum_{i=1}^{n} \sigma_i^2 = \frac{\sigma^2}{n}$.

Loi forte des grands nombres. *Si* $\frac{1}{n} \sum_{i=1}^{n} \mu_i \to \mu$ *et* $\sum_{i=1}^{n} \frac{\sigma_i^2}{i^2} \to a$, *alors* $\overline{\mathcal{X}}_n \triangleq \frac{1}{n} \sum_{i=1}^{n} \mathcal{X}_i$ *est telle que*

$$\overline{\mathcal{X}}_n \xrightarrow{p.s.} \mu. \tag{B.7}$$

Cas particulier : les v.a. $\mathcal{X}_i$ sont i.i.d. μ, σ. On a alors $\frac{1}{n} \sum_{i=1}^{n} \mu_i = \mu$ et $\sum_{i=1}^{n} \frac{{\sigma_i}^2}{i^2} = \sigma^2 \sum_{i=1}^{n} \frac{1}{i^2}$, qui converge.

9 CANAUX DISCRETS

Le contenu de ce chapitre a été annoncé à quelques reprises dans ce qui précède. Notre objectif ici est de nous convaincre qu'il est possible d'utiliser un canal de manière fiable et efficace, même si ce canal est le siège de perturbations aléatoires, qui le rendent partiellement imprévisible. Nous entendons ici par "communication fiable", la possibilité de reconnaître les symboles émis avec une probabilité d'erreur répondant à des spécifications données (éventuellement très contraignantes), et par "communication efficace" une communication qui exploite au mieux les bandes passantes mises à disposition. Bien que nous allons assez peu nous préoccuper pour le moment des moyens matériels (puissance de calcul, mémoire,. . .) à mettre en oeuvre pour effectivement réaliser cet objectif, nous entreverrons cependant que le prix à payer de ce point de vue est également fonction de la sévérité des spécifications. Nous verrons plus loin que la mise au point d'un système de communication consiste essentiellement en un compromis où interviennent le *taux d'erreurs*, le *débit*, et la *puissance de calcul* utilisée pour le codage/décodage.

Cependant, nous limiterons notre objectif ici à montrer la faisabilité de communications avec un niveau d'erreurs arbitrairement faible, et à en mettre en évidence le prix à payer en termes des deux autres paramètres. Les moyens pratiques actuellement mis au point pour se rapprocher de ces conditions idéales seront discutés au chapitre 15. Nous restons pour le moment dans le monde discret; le chapitre 10 viendra compléter notre analyse en s'intéressant au cas particulier de canaux continus utilisés avec une source discrète.

Le contenu de ce chapitre est structuré comme suit : (i) nous commençons par définir la capacité (en information) d'un canal discret de façon générale; (ii) nous illustrons cette notion à l'aide de quelques cas particuliers de modèles de canaux; (iii) nous montrons comment il est possible de transmettre des messages sur un canal bruité avec un taux d'erreurs arbitrairement petit et un débit arbitrairement proche de la capacité du canal, ce qui constitue le *second théorème de Shannon*; (iv) nous montrerons également la réciproque de ce théorème, à savoir qu'il n'est possible d'atteindre un taux d'erreurs arbitrairement petit que si le débit est inférieur à la capacité du canal; (v) il y a une dualité entre codage de source et codage de canal : le premier élimine la redondance pour former une version aussi compacte que possible des messages de la source, alors que le second introduit, comme nous le verrons, la redondance de manière à contrôler le taux d'erreurs; nous terminerons ce chapitre en montrant que ces deux problèmes peuvent être découplés, ce qui justifiera complètement notre décomposition du schéma de communication de Shannon en source et canal "normalisés", introduite au début de ce cours (figure 2.2).

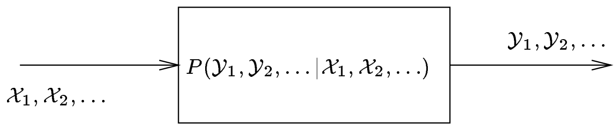


Figure 9.1. Canal : système entrée sortie stochastique

9.1 COMMUNICATIONS A TRAVERS UN CANAL

Nous commençons par définir les notions de canal et de capacité en information d'un canal en donnant quelques exemples illustratifs.

9.1.1 Canal discret

Un canal discret est un système stochastique en temps discret qui accepte à son entrée des suites de symboles définies sur un alphabet d'entrée $\mathcal{X}$ et émet à sa sortie des suites de symboles définies sur un alphabet de sortie $\mathcal{Y}$. Les deux alphabets sont supposés finis, de tailles $|\mathcal{X}|$ et $|\mathcal{Y}|$ éventuellement différentes.

La nature stochastique du canal fait que pour une suite d'entrée connue il n'est pas en général possible de déterminer avec certitude quelle sera la suite de sortie. Nous supposerons dès lors que la suite de sortie est reliée à l'entrée par un modèle probabiliste. Le plus général qu'on puisse imaginer est un modèle qui relie des séquences quelconques d'entrée et de sortie, c'est-à-dire la donnée des lois de probabilités conditionnelles

$$P(\mathcal{Y}_1, \ldots, \mathcal{Y}_m | \mathcal{X}_1, \ldots, \mathcal{X}_n), \tag{9.1}$$

définies $\forall m, n = 1, 2, \ldots$

Il s'agit donc d'un dispositif qui choisit de façon aléatoire des séquences de sortie en fonction des séquences d'entrée (voir figure 9.1). En d'autre mots, nous avons défini un processus stochastique $\mathcal{Y}_i$ en temps discret dont la loi de probabilité est choisie en fonction d'une séquence d'entrée choisie par le système qui alimente le canal.

Canal "causal". Le canal est dit causal, si $\forall m \leq n$

$$P(\mathcal{Y}_1, \ldots, \mathcal{Y}_m | \mathcal{X}_1, \ldots, \mathcal{X}_n) = P(\mathcal{Y}_1, \ldots, \mathcal{Y}_m | \mathcal{X}_1, \ldots, \mathcal{X}_m). \tag{9.2}$$

Cela implique (en marginalisant les deux membres sur $\mathcal{Y}_k$, c'est-à-dire en sommant sur $\mathcal{Y}_1, \ldots, \mathcal{Y}_{k-1}$) que $\forall k \leq m \leq n$

$$P(\mathcal{Y}_k, \ldots, \mathcal{Y}_m | \mathcal{X}_1, \ldots, \mathcal{X}_n) = P(\mathcal{Y}_k, \ldots, \mathcal{Y}_m | \mathcal{X}_1, \ldots, \mathcal{X}_m), \tag{9.3}$$

et en particulier que $\forall m \leq n$

$$P(\mathcal{Y}_m | \mathcal{X}_1, \ldots, \mathcal{X}_n) = P(\mathcal{Y}_m | \mathcal{X}_1, \ldots, \mathcal{X}_m), \tag{9.4}$$

c'est-à-dire que la sortie à tout instant m est conditionnellement indépendante des entrées futures, lorsque les entrées présente et passées (instants $i \leq m$) sont connues.

Remarque. Nous engageons le lecteur à se méfier de cette notion de "causalité" probabiliste, qui est en réalité une notion plus restrictive que la notion classique de causalité en théorie des systèmes. En particulier, il est parfaitement possible de construire un canal non causal au sens ou nous l'avons défini ci-dessus, à l'aide de systèmes physiques causaux, par exemple en utilisant des délais. Par ailleurs, notons que si $\mathcal{Y}_k$ dépend probabilistiquement de $\mathcal{X}_{k-1}$ alors $\mathcal{X}_{k-1}$ ne peut pas être statistiquement indépendant de $\mathcal{Y}_k$, donc l'entrée passée dépend bien de la sortie future, ce qui n'est nullement en contradiction avec la propriété de causalité.

On peut définir la notion de canal causal à mémoire finie, qui est tel que la sortie ne dépend que d'un nombre fini de symboles d'entrée passés.

On peut également définir la notion de canal stationnaire, dont les probabilités conditionnelles entrées/sorties restent invariantes lors d'un décalage dans le temps.

Nous allons voir ci-dessous le cas particulier qui nous intéresse, à savoir le *canal discret, causal, sans mémoire, et stationnaire*. L'ensemble de nos développements porte essentiellement sur ce cas particulier. Nous ferons néanmoins quelques remarques, ponctuellement, sur la généralisation de telle ou telle propriété.

Canal causal sans mémoire. Le canal causal est dit sans mémoire si $\forall k \geq 2$

$$P(\mathcal{Y}_k|\mathcal{X}_1,\ldots,\mathcal{X}_k,\mathcal{Y}_1,\ldots,\mathcal{Y}_{k-1}) = P(\mathcal{Y}_k|\mathcal{X}_k), \tag{9.5}$$

c'est-à-dire si la sortie à un instant donné ne dépend que de l'entrée au même instant. Le canal sans mémoire est donc aussi causal par hypothèse.

D'une part, lorsque la loi de probabilité (9.1) du canal est donnée, on peut calculer à partir de celle-ci les probabilités conditionnelles du membre de gauche de (9.5) et vérifier si le canal est effectivement sans mémoire, en vérifiant qu'elles ne dépendent pas des valeurs des variables $\mathcal{X}_1,\ldots,\mathcal{X}_{k-1},\mathcal{Y}_1,\ldots,\mathcal{Y}_{k-1}$.

D'autre part, lorsque le canal est supposé sans mémoire, la loi de probabilité conditionnelle (9.1) est entièrement déterminée par les probabilités conditionnelles instantanées $P(\mathcal{Y}_k|\mathcal{X}_k)$.

En effet, en posant $\mathcal{Z}^k \triangleq \mathcal{Z}_1,\ldots,\mathcal{Z}_k$, et en prenant $m \leq n$, on calcule

$$\begin{aligned}
P(\mathcal{Y}^m|\mathcal{X}^n) &\stackrel{a}{=} P(\mathcal{Y}^m|\mathcal{X}^m) && (9.6)\\
&\stackrel{b}{=} P(\mathcal{Y}_m|\mathcal{X}^m,\mathcal{Y}^{m-1})P(\mathcal{Y}^{m-1}|\mathcal{X}^m) && (9.7)\\
&\stackrel{c}{=} P(\mathcal{Y}_m|\mathcal{X}_m)P(\mathcal{Y}^{m-1}|\mathcal{X}^{m-1}) && (9.8)\\
&\stackrel{d}{=} \prod_{i=1}^{m} P(\mathcal{Y}_i|\mathcal{X}_i), && (9.9)
\end{aligned}$$

où (a) est obtenu par causalité, (b) par application de la formule $P(A,B|C) = P(A|C)P(B|A,C)$, (c) d'une part, par application de la propriété de non mémoire au premier facteur, d'autre part, par la propriété de causalité appliquée au second, et enfin (d) par induction sur m.

Canal discret sans mémoire stationnaire. Les probabilités $P(\mathcal{Y}_k|\mathcal{X}_k)$ peuvent éventuellement être dépendantes du temps. Le canal sans mémoire est donc dit stationnaire si $\forall k \geq 1$ on a

$$P(\mathcal{Y}_k|\mathcal{X}_k) = P(\mathcal{Y}|\mathcal{X}), \tag{9.10}$$

c'est-à-dire si la relation entrée sortie est indépendante du temps.

Ce type de canal est donc caractérisé par une matrice stochastique de dimension $|\mathcal{X}| \times |\mathcal{Y}|$, appelée *matrice de transition*. Les lignes de cette matrice correspondent aux symboles d'entrée $X_1,\ldots,X_{|\mathcal{X}|}$ et les colonnes aux symboles de sortie $Y_1,\ldots,Y_{|\mathcal{Y}|}$. L'élément i,j donne la probabilité conditionnelle $P(Y_j|X_i)$.

Nous le désignerons dans la suite simplement par le terme *canal sans mémoire*, les propriétés de causalité et de stationnarité étant sous-entendues.

9.1.2 Capacité en information du canal sans mémoire

Nous définissons la *capacité en information* (par symbole) du canal par

$$C = \max_{P(\mathcal{X})} I(\mathcal{X};\mathcal{Y}). \tag{9.11}$$

Il s'agit d'une grandeur calculée pour une utilisation du canal, c'est-à-dire pour une paire de symboles entrée/sortie.

Notons que $I(\mathcal{X};\mathcal{Y})$ dépend à la fois des propriétés statistiques de l'entrée et du canal. Mais du fait de la maximisation sur toutes les distributions de probabilité des symboles d'entrée, la capacité ne dépend que des propriétés du canal lui-même.

Nous donnerons plus loin une définition de la *capacité opérationnelle* d'un canal, liée directement au débit maximum de transmission de l'information avec un taux d'erreurs arbitrairement petit. Le second théorème de Shannon établit que cette notion est équivalente à la notion de capacité en information; nous utiliserons donc la plupart du temps le simple terme de *capacité*.

Suggestion : en s'inspirant de la définition générale de l'entropie d'un processus aléatoire, essayer de trouver une généralisation de la capacité par symbole pour le modèle général de canal causal. Sous quelles conditions particulières cette généralisation a-t-elle un sens ?

9.1.3 Exemples de capacités de canaux

9.1.3.1 Canal binaire sans bruit.

Il s'agit d'un canal qui utilise des alphabets d'entrée et de sortie binaires et reproduit à sa sortie le symbole d'entrée avec certitude. On a dans ce cas, $I(\mathcal{X};\mathcal{Y}) = H(\mathcal{X})$ qui est maximale et vaut 1 Shannon lorsque les deux signaux d'entrée sont équiprobables.

Par ailleurs, on peut évidemment transmettre un bit (symbole binaire) sans erreur par utilisation de ce canal. Les deux notions coïncident donc bien.

La matrice de transition de ce canal est la matrice identité

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

9.1.3.2 Canal bruité sans recouvrement des sorties.

Il s'agit d'un canal qui utilise 2 symboles d'entrée et quatre symboles de sortie, et qui lorsque le premier symbole est présenté à l'entrée, choisit aléatoirement, avec une probabilité de $(p, 1-p)$, un des deux premiers symboles de sortie. De même, pour le second symbole d'entrée il choisit l'un des deux derniers symboles de sortie, avec une probabilité de $(q, 1-q)$.

Dans ce cas aussi on peut retrouver avec certitude l'entrée connaissant la sortie. On dit que les symboles d'entrée sont distinguables. Donc, $H(\mathcal{X}|\mathcal{Y}) = 0$ (quels que soient p et q, d'ailleurs) et $I(\mathcal{X};\mathcal{Y}) = H(\mathcal{X})$. Par conséquent la capacité vaut à nouveau 1 Shannon et est réalisée avec une distribution d'entrée uniforme.

Ici également il est possible de transmettre un symbole binaire sans erreurs par symbole envoyé sur ce canal.

La matrice de transition de ce canal est

$$\begin{bmatrix} p & (1-p) & 0 & 0 \\ 0 & 0 & q & (1-q) \end{bmatrix}.$$

Remarque. Cet exemple, apparemment trivial, montre bien qu'il est possible de transmettre de l'information sans erreur sur un canal bruité. Nous allons voir dans la suite qu'en utilisant l'extension d'ordre n d'un canal, il est possible de s'arranger pour que les messages de sortie se partitionnent en un certain nombre de classes distinguables. Nous verrons que pour des séquences d'entrées de longueur n croissante, le nombre maximum de classes distinguables en sortie croît de façon exponentielle (disons 2^{nR}), pour autant que n soit suffisamment grand.

Sachant que les séquences d'entrées de longueur n émises par une source sans mémoire deviennent rapidement typiques et en nombre limité par $2^{nH(\mathcal{X})}$, on voit intuitivement que si $H(\mathcal{X}) \leq R$, le nombre de classes distinguables finira, pour n suffisamment grand, par dépasser le nombre de séquences typiques en entrée. On peut donc espérer qu'il devient possible de transmettre les messages typiques sans erreur, c'est-à-dire tous les messages possibles avec une probabilité d'erreur aussi petite que souhaité. Si on se place dans le cas d'un canal binaire, on sait qu'une séquence d'entrée de longueur n' pourrait être codée par un code binaire ne demandant pas plus de $n'H(\mathcal{X})$ bits en moyenne, chacune desquelles consommera en moyenne n symboles du canal. Si alors, $nR \geq n'H \Leftrightarrow \frac{n'}{n} \leq \frac{R}{H}$ le débit est atteignable, c'est-à-dire qu'il est possible de transmettre les symboles avec un débit moyen au moins égal à celui de la source.

9.1.3.3 La machine à écrire bruitée.

Les alphabets d'entrée et de sortie sont identiques, composés des 26 lettres de l'alphabet latin. Le canal opère de la manière suivante : lorsqu'une lettre est envoyée à l'entrée, la sortie reçoit cette lettre avec une probabilité p=0.5, ou la lettre suivante (dans l'ordre alphabétique) avec une probabilité p=0.5. En utilisant un sous-ensemble des symboles d'entrée composé d'une lettre sur deux, les sorties deviennent distinguables. Il sera donc possible de transmettre l'équivalent de $\log 13$ symboles binaires par utilisation du canal.

On peut calculer la capacité en information : on trouve $H(\mathcal{Y}|\mathcal{X}) = \log 2 = 1$ Shannon, quelle que soit la distribution de probabilité de l'entrée. Donc, $I(\mathcal{X};\mathcal{Y}) = H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X})$ est maximale lorsque les sorties sont

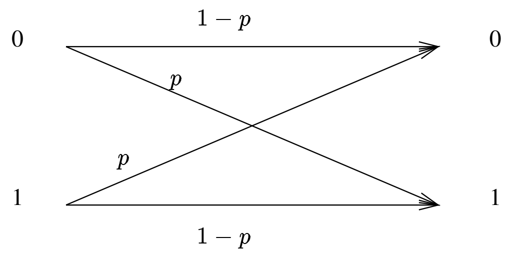


Figure 9.2. Canal symétrique binaire

équiprobables, ce qui est réalisé en utilisant une distribution uniforme pour les entrées. On a alors, $I(\mathcal{X};\mathcal{Y}) = \log 26 - \log 2 = \log 13$ Shannon.

9.1.3.4 Canal symétrique binaire.

Cet exemple très important en pratique est illustré à la figure 9.2. A gauche figurent les deux entrées, à droite les sorties; les flèches répésentent les transitions possibles avec leur probabilité.

La matrice de transition de ce canal est

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

Il s'agit du modèle le plus simple de canal avec erreurs; pourtant il est représentatif de la complexité du problème général. Lorsqu'un symbole est reçu, il est impossible de déterminer si il y a eu erreur de transmission ou non; il n'est pas non plus possible d'utiliser un alphabet d'entrée réduit, puisqu'il n'y a que deux symboles d'entrée.

Nous verrons plus loin dans ce chapitre qu'il est néanmoins possible de transmettre de l'information de manière fiable avec un taux (nombre de bits/symbole) non-nul. Mais il faudra recourir à la notion d'extension du canal et utiliser une partie seulement des séquences d'entrée de cette extension.

Calculons la capacité en information de ce canal :

$$\begin{aligned} I(\mathcal{X};\mathcal{Y}) &= H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X}) \\ &= H(\mathcal{Y}) - \sum_{X\in\mathcal{X}} P(X)H(\mathcal{Y}|X) \\ &= H(\mathcal{Y}) - \sum_{X\in\mathcal{X}} P(X)H_2(p) \\ &= H(\mathcal{Y}) - H_2(p) \\ &\leq 1 - H_2(p), \end{aligned} \tag{9.12}$$

puisque $\mathcal{Y}$ est une variable binaire. On pourra se convaincre que pour réaliser cette borne supérieure, il suffit d'utiliser une distribution uniforme à l'entrée, ce qui aura pour effet d'assurer une distribution uniforme en sortie également. Par conséquent, la capacité en information du canal symétrique binaire vaut

$$C = 1 - H_2(p). \tag{9.13}$$

En particulier, cette grandeur est symétrique par rapport à $p = 0.5$ et nulle pour $p = 0.5$. Elle vaut 1, lorsque $p = 0$, ce qui est compatible avec la capacité du canal sans erreurs.

9.1.3.5 Canal binaire avec effacement.

Il s'agit du modèle représentant un canal binaire où certains symboles sont perdus. Le recepteur est informé du fait qu'un symbole est perdu sans savoir de quel symbole il s'agissait. Les symboles non perdus sont supposés être reçus sans corruption. La probabilité de perte de symboles vaut α et est supposée indépendante du symbole émis, tel que schématisé à la figure 9.3.

La matrice de transition de ce canal est

$$\begin{bmatrix} 1-\alpha & \alpha & 0 \\ 0 & \alpha & 1-\alpha \end{bmatrix}.$$

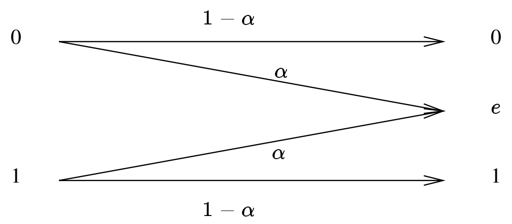


Figure 9.3. Canal binaire avec effacement

Le calcul de la capacité donne

$$\begin{aligned} C &= \max_{P(\mathcal{X})} I(\mathcal{X};\mathcal{Y}) \\ &= \max_{P(\mathcal{X})} H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X}) \\ &= \max_{P(\mathcal{X})} H(\mathcal{Y}) - H_2(\alpha). \end{aligned} \tag{9.14}$$

Notons qu'il est impossible de réaliser $H(\mathcal{Y}) = \log 3$ par un choix judicieux des probabilités d'entrée (sauf si $\alpha = 1/3$).

En général, soit $P(\mathcal{X} = 1) = \pi$; on trouve $H(\mathcal{Y}) = H_2(\alpha) + (1-\alpha)H_2(\pi)$ qui est maximale lorsque $\pi = 0.5$. Substituant dans (9.14) on trouve finalement

$$C = 1 - \alpha, \tag{9.15}$$

ce qui revient à dire que les symboles perdus n'apportent aucune information.

Notons, qu'il n'est pas évident a priori qu'il soit possible d'atteindre cette capacité. Mais nous pourrons le déduire à partir du second théorème de Shannon.

Supposons néanmoins que le récepteur soit en mesure de signaler à l'émetteur le fait qu'un symbole a été perdu, c'est-à-dire supposons que notre diagramme de communication comprenne une boucle de retour d'information. Dans ce cas, lorsqu'un symbole est perdu il suffit de le retransmettre et on se convainc aisément que le taux d'échec sera en moyenne de α, mais la transmission sans erreurs. On arrive donc effectivement ainsi à réaliser (au prix de l'installation d'une boucle de retour) un taux moyen de $1 - \alpha$ Shannon par symbole envoyé sur le canal. Nous reviendrons plus loin sur ce type de situation, et nous montrerons en fait que ce taux est le taux limite qui peut être réalisé avec ou sans boucle de retour.

9.1.4 Canaux symétriques

Un canal est dit symétrique, si les lignes de sa matrice de transition sont des permutations les unes des autres, et si les colonnes sont des permutations les unes des autres.

Dans ce cas $H(\mathcal{Y}|\mathcal{X})$ est indépendante de la distribution de probabilités de l'entrée, et, d'autre part, $\mathcal{Y}$ sera distribuée de façon uniforme dès lors que $\mathcal{X}$ l'est.

Par conséquent, la capacité de ce type de canal vaut

$$C = \log|\mathcal{Y}| - H(LMT) \tag{9.16}$$

où $H(LMT)$ désigne l'entropie d'une ligne de la matrice de transition.

Cette dernière propriété se généralise à un canal dit *faiblement* symétrique, qui est un canal tel que les lignes de sa matrice de transition soient des permutations les unes des autres, et tel que la somme des termes de chaque colonne soit égale à une constante indépendante de la colonne.

De nombreux modèles pratiques conduisent à des canaux symétriques ou faiblement symétriques.

Illustration : canal à bruit additif. Représentons les alphabets d'entrée et de sortie par les entiers $\mathcal{X} = \{0, 1, 2, \ldots, c-1\}$, et supposons que $Y = (X + Z)\mathrm{mod}c$ où $\mathcal{Z}$ est une variable aléatoire indépendante de $\mathcal{X}$, dont les valeurs possibles sont $\mathcal{Z} = \{0, 1, \ldots, c-1\}$.

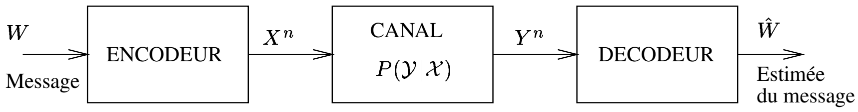


Figure 9.4. Système de communication

(Suggestion : se convaincre que ce canal est symétrique en supposant pour une valeur de $c = 4$ *une distribution de probabilité pour* $\mathcal{Z}$ *et en construisant explicitement la matrice de transition de ce canal.)*

9.1.5 Propriétés de la capacité en information d'un canal

Nous énumérons sans les démontrer les propriétés suivantes (elles sont immédiates)

1. $C \geq 0$.
2. $C \leq \min\{\log|\mathcal{X}|, \log|\mathcal{Y}|\}$.

De plus, on montre que, pour une distribution $P(\mathcal{Y}|\mathcal{X})$ donnée, $I(\mathcal{X};\mathcal{Y})$ est une fonction continue et concave de $P(\mathcal{X})$. Comme elle est concave, tout maximum local est un maximum global sur l'ensemble convexe des distributions de probabilités de $\mathcal{X}$. Comme elle est bornée, le maximum l'est aussi. On peut donc utiliser des méthodes d'optimisation locale ("descente" de gradient,...) pour trouver son maximum.

9.2 CODAGE DE CANAL

Nous définissons maintenant ce que nous entendons par *schéma de communication* à travers un canal.

9.2.1 Système de communication

Nous analyserons ci-dessous des systèmes de communication tels que réprésentés à la figure 9.4. Dans cette figure, la partie centrale représente le canal à proprement parler, tel que nous l'avons introduit ci-dessus. Nous supposons que dans notre problématique, cette partie est une *donnée* du problème. Rappelons qu'un canal est un dispositif physique qui permet de transmettre des symboles d'entrée vers un destinataire isolé dans l'espace (télécommunications) ou dans le temps (enregistrement de données). La construction de tels canaux à partir de supports physiques divers fait l'objet des techniques de télécommunications et d'enregistrement de données, qui ne sont pas abordées dans ce cours. Les parties externes dans la figure 9.4 concernent l'encodage et le décodage de *messages* sous la forme de séquences de symboles transmises sur le canal. Nous allons nous intéresser à la conception de tels systèmes dans le but d'utiliser au mieux les ressources offertes par le canal. Plus précisémement, dans ce chapitre nous nous intéressons à la faisabilité théorique et dans le chapitre 15 nous étudierons la faisabilité pratique.

Le système de communication de la figure 9.4 fonctionne de la manière suivante. Un message W (tiré dans un ensemble fini de messages possibles $\mathcal{W} = \{1, 2, \ldots, M\}$) est encodé par *l'encodeur* sous la forme d'une suite de n symboles d'entrée du canal, désignée par $X^n(W)$. Cette séquence est reçue de l'autre côté sous la forme d'une séquence aléatoire de symboles Y^n (distribuée selon la loi de probabilité $P(\mathcal{Y}^n|X^n(W))$. Cette séquence Y^n est ensuite décodée par le *décodeur*, qui choisit un élément $\hat{W}(Y^n) \in \mathcal{W}$. Le récepteur commet une *erreur* si $\hat{W}(Y^n) \neq W$.

Dans ce qui suit, nous supposons que l'encodeur et le décodeur opèrent de manière déterministe : $X^n(W)$ est la règle (ou fonction) d'encodage; $\hat{W}(Y^n)$ est la règle (ou fonction) de décodage.

Notations. Nous avons utilisé ci-dessus et continuerons d'utiliser dans la suite de ces notes la notation abrégée $\mathcal{X}^n$ pour désigner un vecteur aléatoire composé d'une suite de n variables aléatoires définies sur le même ensemble $\mathcal{X}$. Rappelons que selon notre convention, cette notation désigne à la fois l'ensemble[1] de valeurs possibles

[1] $\mathcal{X}^n$ est en effet le produit cartesien de n fois $\mathcal{X}$.

de cette variable et la variable elle-même. De même, nous utilisons la notation X^n pour désigner une réalisation quelconque de cette variable aléatoire, c'est-à-dire un élément quelconque de $\mathcal{X}^n$.

Il est capital de se rendre compte que nous ne faisons aucune hypothèse sur la distribution de probabilités $P(\mathcal{X}^n)$ en utilisant cette notation. En particulier, la notation n'implique certainement pas que les composantes du vecteur aléatoire sont distribuées de façon identique et/ou indépendante.

Nous utiliserons de plus les notations $\mathcal{X}_j^n$ (resp. X_j^n) pour désigner la j-ème composante ($j \leq n$) de ce vecteur aléatoire (resp. de sa réalisation). Lorsqu'aucune confusion n'est possible, nous utiliserons les notations abrégées $\mathcal{X}_j$ (resp. X_j).

Définition : données canal sans mémoire. Pour caractériser un canal, nous supposons seulement donnée la matrice de transition du canal $P(\mathcal{Y}_k|\mathcal{X}_k)$, pas nécessairement stationnaire.

Nous ne supposons pas non plus d'emblée que le canal est causal, car nous verrons ci-dessous que cette propriété peut dépendre de la façon dont on utilise le canal physique.

Définition : extension d'ordre n d'un canal. L'extension d'ordre n d'un canal discret sans mémoire $(\mathcal{X}, P(\mathcal{Y}|\mathcal{X}), \mathcal{Y})$ est le canal $(\mathcal{X}^n, P(\mathcal{Y}^n|\mathcal{X}^n), \mathcal{Y}^n)$, où

$$P(\mathcal{Y}_k|\mathcal{X}^k, \mathcal{Y}^{k-1}) = P(\mathcal{Y}_k|\mathcal{X}_k), \forall k = 1, 2, \ldots n. \tag{9.17}$$

En d'autres termes, lorsque la $k-$ème composante de l'entrée est connue, la $k-$ème sortie ne dépend plus ni des entrées ni des sorties précédentes, et est distribuée selon la même distribution de probabilité conditionnelle que le $k-$ème symbole du canal de départ.

Définition : canal utilisé sans boucle de retour. Si $\forall k < n$, $\mathcal{X}_{k+1}$ est conditionnellement indépendant de $\mathcal{Y}^k$ lorsque $\mathcal{X}^k$ est connu, on dit que le canal est *utilisé* sans boucle de retour (ou sans feedback). Rappelons que l'indépendance conditionnelle $\mathcal{X}$ de $\mathcal{Y}$ lorsque $\mathcal{Z}$ est connue signifie que

$$P(\mathcal{X}|\mathcal{Y}, \mathcal{Z}) = P(\mathcal{X}|\mathcal{Z}) \tag{9.18}$$

où $\mathcal{X}$ et $\mathcal{Y}$ peuvent évidemment être échangés.

Extension vs causalité. Nous avons vu plus haut que si nous imposons de plus la propriété de causalité, alors les propriétés statistiques de l'extension d'ordre n du canal sont entièrement déterminées par le produit des $P(\mathcal{Y}|\mathcal{X})$. Montrons qu'il en est de même si le canal est utilisé sans boucle de retour.

Donc l'indépendance de $\mathcal{X}_{k+1}$ et de $\mathcal{Y}^k$ lorsque $\mathcal{X}^k$ est connu est équivalente à

$$P(\mathcal{X}_{k+1}|\mathcal{Y}^k, \mathcal{X}^k) = P(\mathcal{X}_{k+1}|\mathcal{X}^k), \tag{9.19}$$

ou

$$P(\mathcal{Y}^k|\mathcal{X}_{k+1}, \mathcal{X}^k) = P(\mathcal{Y}^k|\mathcal{X}^k), \tag{9.20}$$

qui peut encore être écrite sous la forme

$$P(\mathcal{Y}^n|\mathcal{X}^{n+1}) = P(\mathcal{Y}^n|\mathcal{X}^n), \tag{9.21}$$

ce qui implique aussi, par marginalisation sur $\mathcal{Y}_{m+1}, \ldots, \mathcal{Y}_n$ (avec $m < n$)

$$P(\mathcal{Y}^m|\mathcal{X}^{n+1}) = P(\mathcal{Y}^m|\mathcal{X}^n), \tag{9.22}$$

et par induction sur n on a $\forall m < n$

$$P(\mathcal{Y}^m|\mathcal{X}^n) = P(\mathcal{Y}^m|\mathcal{X}^{m+1}). \tag{9.23}$$

D'autre part, l'équation (9.20) implique que

$$P(\mathcal{Y}^m|\mathcal{X}^{m+1}) = P(\mathcal{Y}^m|\mathcal{X}^m), \tag{9.24}$$

et au total on a bien $\forall m < leqn$

$$P(\mathcal{Y}^m|\mathcal{X}^n) = P(\mathcal{Y}^m|\mathcal{X}^m). \tag{9.25}$$

Le canal est donc causal au sens ou nous l'avons défini plus haut. Réciproquement, si on introduisait une dépendance entre la valeur de X_{k+1} et les sorties Y^k, le canal ne serait plus causal.

Puisque le canal sans feedback est causal on a $\forall n$

$$P(\mathcal{Y}^n|\mathcal{X}^n) = \prod_{i=1}^{n} P(\mathcal{Y}_i|\mathcal{X}_i). \tag{9.26}$$

Exercice. Redémontrons à titre d'exercice cette propriété, en utilisant un argument par induction (la propriété est trivialement vraie pour $n = 1$) et montrons que si la propriété est vraie pour $n = k$ elle l'est encore pour $n = k + 1$. En effet,

$$\begin{aligned} P(\mathcal{Y}^{k+1}|\mathcal{X}^{k+1}) &= P(\mathcal{Y}_{k+1}, \mathcal{Y}^k|\mathcal{X}^{k+1}) \\ &= P(\mathcal{Y}_{k+1}|\mathcal{Y}^k, \mathcal{X}^{k+1})P(\mathcal{Y}^k|\mathcal{X}^{k+1}) \\ &= P(\mathcal{Y}_{k+1}|\mathcal{X}_{k+1})P(\mathcal{Y}^k|\mathcal{X}^{k+1}) \\ &= P(\mathcal{Y}_{k+1}|\mathcal{X}_{k+1})P(\mathcal{Y}^k|\mathcal{X}^k), \end{aligned} \tag{9.27}$$

où la première identité est simplement une réécriture, la seconde est obtenue par application de la définition de la probabilité conditionnelle, la troisième en application de la définition de l'extension du canal (9.17) et enfin la dernière en nous servant de (9.21) qui exprime l'hypothèse “sans feedback”. Il est alors évident qu'en introduisant l'hypothèse inductive dans (9.27) on obtient la propriété (9.26) pour $n = k + 1$.

Résumé. Dorénavant, lorsque nous parlerons du canal discret sans mémoire, nous supposerons qu'il s'agit du canal sans mémoire et sans feedback, à moins de spécifier explicitement le contraire. Notons que dans ce qui précède nous n'avons pas requis que le canal soit stationnaire. Il le sera si les $P(\mathcal{Y}_i|\mathcal{X}_i)$ sont indépendantes de i.

Définition : code (M, n). Un code (M, n) pour un canal $(\mathcal{X}, P(\mathcal{Y}|\mathcal{X}), \mathcal{Y})$ est défini par

1. Un ensemble d'indices $\{1, \ldots, M\}$;
2. Une fonction d'encodage $X^n(\cdot) : \{1, \ldots, M\} \to \mathcal{X}^n$, qui donne les mots de code $X^n(1), \ldots, X^n(M)$ que nous appellerons la table du code (codebook).
3. Une fonction de décodage
$$g(\cdot) : \mathcal{Y}^n \to \{1, \ldots, M\}, \tag{9.28}$$
qui est une règle déterministe qui associe à chaque sortie possible du canal une entrée $g(Y^n)$.

Définition : probabilités d'erreur. Outre la probabilité d'erreur de décodage, nous allons définir trois notions de probabilité d'erreur qui seront utilisées ultérieurement.

Nous désignerons la *probabilité d'erreur conditionnelle* sachant que i fut envoyé sur le canal par

$$\lambda_i = P(g(\mathcal{Y}^n) \neq i|\mathcal{X}^n = X^n(i)) = \sum_{Y^n \in \mathcal{Y}^n} P(Y^n|X^n(i))(1 - \delta_{g(Y^n),i}), \tag{9.29}$$

où $\delta_{g(Y^n),i}$ est le symbole de Kronecker (qui vaut 1 si les deux indices sont identiques et 0 sinon), c'est-à-dire que la somme s'étend sur toutes les sorties dont le décodage ne restitue pas i.

Nous désignerons la *probabilité d'erreur maximale* d'un code (M, n) par

$$\lambda^{(n)} = \max_{i \in \{1, \ldots, M\}} \lambda_i. \tag{9.30}$$

Enfin, nous désignerons la *probabilité d'erreur moyenne (algébrique)* par

$$P_e^{(n)} = \frac{1}{M} \sum_{i=1}^{M} \lambda_i. \tag{9.31}$$

Notons que si $\mathcal{I}$ représente la variable aléatoire de choix des entrées, et si on suppose que $\mathcal{I}$ a une distribution uniforme sur $\{1, \ldots, M\}$

$$P_e^{(n)} = P(g(Y^n) \neq \mathcal{I}) \tag{9.32}$$

où le second membre désigne la probabilité d'erreur de décodage.

Règle de décodage optimale. En général, $\mathcal{I}$ n'est pas nécessairement distribuée uniformément.

On entend alors par règle de décodage optimale, une règle qui minimise la probabilité d'erreurs (moyenne au sens statistique du terme) $P(g(\mathcal{Y}^n) \neq \mathcal{I})$. Lorsque les probabilités des séquences d'entrée et de transition du canal sont connues, cette règle consiste simplement à choisir l'indice i tel que

$$P(X^n(i)|Y^n) = \frac{P(Y^n|X^n(i))P(X^n(i))}{\sum_{i=1}^{M} P(Y^n|X^n(i))P(X^n(i))}, \tag{9.33}$$

soit maximale. Comme le dénominateur du second membre de cette égalité ne dépend pas du choix effectué, cela revient donc à choisir l'indice i tel que

$$P(Y^n|X^n(i))P(X^n(i)) \tag{9.34}$$

soit maximale. Cette règle de décision optimale est bien connue sous le terme de *règle du maximum de probabilité a posteriori*, ou encore *règle de Bayes*.

L'inconvénient principal de cette règle est de nécessiter la connaissance des probabilités d'émission des symboles à l'entrée du canal, ce qui est rarement le cas en pratique. On utilise alors la règle *du maximum de vraisemblance* qui choisit l'indice i maximisant

$$P(Y^n|X^n(i)); \tag{9.35}$$

elle serait optimale si les symboles étaient distribués de façon équiprobable car elle minimise la probabilité d'erreur moyenne algébrique $P_e^{(n)}$.

Notons que si la source est codée de façon proche de l'optimum, alors les symboles à la sortie du code seront distribués selon une loi proche de la loi uniforme. La règle du maximum de vraisemblance sera alors quasi optimale.

Définition : taux (ou débit) de communication R. Le débit de communication R d'un code (M, n) est défini par

$$R = \frac{\log M}{n} \tag{9.36}$$

exprimé en Shannon par symbole transmis sur le canal. Le débit est égal à *l'entropie par symbole de canal* des messages d'entrée lorsque ceux-ci sont distribués de façon équiprobable.

Définition : débit réalisable. Un débit R est dit réalisable (ou atteignable), s'il existe une suite de codes $(M(n), n), n = 1, 2, \ldots$ avec $M(n) = \lceil 2^{nR} \rceil$, telle que

$$\lim_{n \to \infty} \lambda^{(n)} = 0. \tag{9.37}$$

Si un débit R est réalisable, cela veut donc dire qu'il est possible de transmettre $\lceil 2^{nR} \rceil$ messages distincts à l'aide de séquences de symboles d'entrée du canal de longueur n, de telle façon que lorsque n devient grand on s'approche de plus en plus des conditions sans erreurs. Notons que la distribution des messages d'entrée ne joue pas dans cette condition, puisque c'est la probabilité *maximale* qui tend vers zéro.

Définition : capacité opérationnelle. La capacité opérationnelle C_o d'un canal discret et sans mémoire est la borne supérieure de tous les débits réalisables.

Notons que $R = 0$ est réalisable, puisque trivialement on peut transmettre $M = 2^0 = 1$ symbole sans erreur sur le canal, on a donc $C_o \geq 0$. Cependant, au stade du raisonnement que nous avons atteint, rien ne garantit que C_o puisse être strictement supérieure à 0, ni qu'elle soit nécessairement finie.

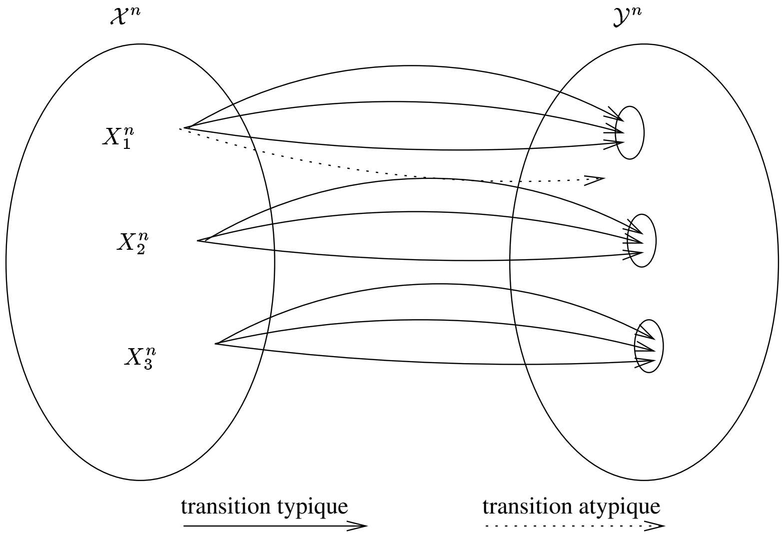


Figure 9.5. Utilisation de l'extension d'ordre n d'un canal

9.2.2 Preview du second théorème de Shannon

Avant de nous lancer dans la démonstration du second théorème de Shannon essayons de dégager le principe du raisonnement qui va suivre.

Tous d'abord, nous allons nous intéresser au cas où les entrées et les sorties du canal sont stationnaires et ergodiques, conditions pour lesquelles le théorème AEP est applicable. Cette condition est réalisée si le signal d'entrée est stationnaire et ergodique et si le canal est stationnaire et à mémoire finie. Cependant, dans nos démonstrations nous allons nous restreindre au cas du canal sans mémoire.

Supposons que nous souhaitions envoyer un ensemble de M messages, composés chacun de n symboles d'entrée du canal, de telle manière que les sorties soient distinguables. Construisons une table de code selon une méthode, qui peut sembler a priori surprenante : fixons une distribution de probabilités quelconque $P(\mathcal{X})$, et construisons les M mots de code par $M \times n$ tirages aléatoires successifs, indépendants selon $P(\mathcal{X})$, ce qui nous donne les symboles $X_i^n(M)$. Nous savons, suite au théorème AEP du chapitre 8 que les mots de code ainsi choisis ont de fortes chances d'être *typiques*, si n est suffisamment grand.

Faisant passer l'un quelconque de ces mots de code dans notre canal, nous aurons à la sortie une séquence aléatoire, qui elle aussi sera composée de symboles indépendants dont la distribution résulte du choix du mot de code (la séquence X^n) et des caractéristiques du canal. Cette séquence est donc également typique (conditionnellement) avec une grande probabilité. La situation est résumée à la figure 9.5.

Le nombre de séquences typiques à l'entrée est à peu près $2^{nH(\mathcal{X})}$; pour chacune de ces séquences typiques il existe environ $2^{nH(\mathcal{Y}|\mathcal{X})}$ messages de sortie typiques. Mais comme le nombre total de séquences de sortie typiques est de $2^{nH(\mathcal{Y})}$, cet ensemble doit être divisé en paquets comprenant $2^{nH(\mathcal{Y}|\mathcal{X})}$ séquences, soit au maximum $2^{nH(\mathcal{Y})-nH(\mathcal{Y}|\mathcal{X})}$ paquets. Il n'est donc possible d'envoyer qu'au maximum $2^{nI(\mathcal{X};\mathcal{Y})}$ séquences distinguables. En étant optimiste, nous pouvons espérer que parmi les $2^{nH(\mathcal{X})}$ message typiques d'entrée, il est possible d'en trouver $2^{nI(\mathcal{X};\mathcal{Y})}$ tels que les $2^{nI(\mathcal{X};\mathcal{Y})}$ ensembles typiques correspondants en sortie ne se recouvrent pas, de manière à pouvoir décoder les entrées sans erreur. En étant encore plus optimiste, nous pouvons espérer effectuer cette opération avec n'importe quel choix de $P(\mathcal{X})$, y compris celui qui maximise $I(\mathcal{X};\mathcal{Y})$. Si tel est le cas, cela voudra dire que nous sommes capables de transmettre 2^{nC} messages d'entrées distinguables en sortie, ce qui voudra dire que la capacité en information est atteignable. Pour terminer notre démonstration, il nous restera à vérifier qu'il n'est pas possible de faire mieux.

La discussion qui précède reste à être confirmée de manière plus rigoureuse. Elle fournit une idée intuitive des raisons qui permettent d'exploiter efficacement les canaux au moyen de messages longs : la loi des grands nombres dont les théorèmes AEP ne sont que la traduction dans le jargon de la théorie de l'information. Elle montre aussi que c'est en travaillant sur des espaces à grande dimension (vecteurs aléatoires de dimension n)

qu'on peut efficacement exploiter cette loi des grand nombres, qui fait qu'au fur et à mesure qu'on augmente la taille des vecteurs, les régions où peuvent se trouver les messages issus de la déformation aléatoire par le canal d'une séquence d'entrée donnée, se concentrent de plus en plus dans une fraction faible de l'espace de sortie. Une autre idée intéressante introduite ci-dessus est celle du codage aléatoire. Nous verrons qu'on peut choisir les mots de code indépendamment les uns des autres, tout en garantissant que la probabilité d'erreur maximale moyenne (de tous ces codes) soit inférieure à ϵ pour autant que n soit suffisamment grand. Par conséquent, il existe donc au moins (en réalité une grande proportion) des codes aléatoires qui satisfont ce taux maximal d'erreurs.

Avant de démontrer le théorème de Shannon (et sa réciproque), introduisons tout d'abord la notion de *séquences conjointement typiques*.

9.2.3 Séquences conjointement typiques

Notre règle de décodage consistera, grosso modo, à associer à une séquence de sortie Y^n l'indice i si le mot de code $X^n(i)$ est "conjointement typique" avec Y^n. Nous allons définir cette notion importante de *typicalité conjointe* et nous allons évaluer la probabilité que cette propriété soit vérifiée sous deux hypothèses : (i) $X^n(i)$ est effectivement la "cause" de Y^n; (ii) $X^n(i)$ n'est pas la "cause" de Y^n.

Supposons que nous observions des séquences de longueur n de couples de variables aléatoires $\mathcal{X}$ et $\mathcal{Y}$, et plaçons-nous, comme pour le théorème AEP du chapitre 8, dans les conditions où entrées et sorties forment conjointement une source sans mémoire, c'est-à-dire que $P(X^n, Y^n) = \prod_{i=1}^n P(X_i, Y_i)$, pour toute séquence entrée-sortie (X^n, Y^n). Cela implique évidemment que $P(X^n) = \prod_{i=1}^n P(X_i)$ et $P(Y^n) = \prod_{i=1}^n P(Y_i)$ (marginalisation).

Cette condition correspond par exemple au cas d'un canal sans mémoire alimenté par une source sans mémoire.

Définition : typicalité conjointe. L'ensemble $A_\epsilon^{(n)}$ de séquences conjointement typiques $\{(X^n, Y^n)\}$ par rapport à une distribution $P(\mathcal{X}, \mathcal{Y})$, est l'ensemble de telles séquences dont les entropies conjointes et marginales *empiriques* sont $\epsilon-$proches des entropies conjointes et marginales, c'est-à-dire

$$\begin{aligned} A_\epsilon^{(n)} \;=\; & \Big\{(X^n, Y^n) \in \mathcal{X}^n \times \mathcal{Y}^n \;: \\ & \left|-\frac{1}{n}\log P(X^n) - H(\mathcal{X})\right| \le \epsilon, \\ & \left|-\frac{1}{n}\log P(Y^n) - H(\mathcal{Y})\right| \le \epsilon, \\ & \left|-\frac{1}{n}\log P(X^n, Y^n) - H(\mathcal{X}, \mathcal{Y})\right| \le \epsilon\Big\}, \end{aligned} \tag{9.38}$$

où

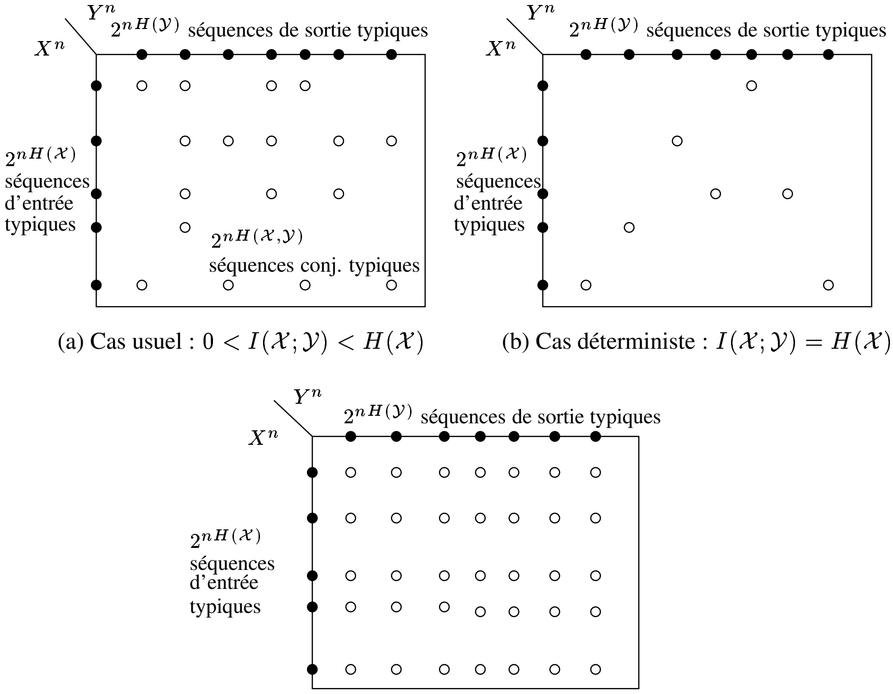
$$P(X^n, Y^n) = \prod_{i=1}^n P(X_i, Y_i). \tag{9.39}$$

Remarquons que l'ensemble est l'intersection de trois ensembles typiques, au sens du théorème AEP du chapitre 8.

Théorème : équipartition asymptotique (AEP) conjointe. Soit une suite de variables aléatoires $(\mathcal{X}^n, \mathcal{Y}^n)$ correspondant à des séquences entrée/sortie de longueur n tirées aléatoirement, de façon indépendante et distribuées selon une même loi de probabilité $P(\mathcal{X}^n, \mathcal{Y}^n)$ (i.e. $P(\mathcal{X}^n, \mathcal{Y}^n) = \prod_{i=1}^n P(\mathcal{X}_i, \mathcal{Y}_i)$). Alors

1. $\lim_{n\to\infty} P((\mathcal{X}^n, \mathcal{Y}^n) \in A_\epsilon^{(n)}) = 1$.
2. $(1-\epsilon)2^{n(H(\mathcal{X},\mathcal{Y})-\epsilon)} \le |A_\epsilon^{(n)}| \le 2^{n(H(\mathcal{X},\mathcal{Y})+\epsilon)}$.
3. si $(\tilde{\mathcal{X}}^n, \tilde{\mathcal{Y}}^n) \sim P(\mathcal{X}^n)P(\mathcal{Y}^n)$, i.e. si $\tilde{\mathcal{X}}^n$ et $\tilde{\mathcal{Y}}^n$ sont indépendantes et distribuées selon les lois marginales de $P(\mathcal{X}^n, \mathcal{Y}^n)$, alors

$$(1-\epsilon)2^{-n(I(\mathcal{X};\mathcal{Y})+3\epsilon)} \le P((\tilde{\mathcal{X}}^n, \tilde{\mathcal{Y}}^n) \in A_\epsilon^{(n)}) \le 2^{-n(I(\mathcal{X},\mathcal{Y})-3\epsilon)}. \tag{9.40}$$



(a) Cas usuel : $0 < I(\mathcal{X};\mathcal{Y}) < H(\mathcal{X})$ (b) Cas déterministe : $I(\mathcal{X};\mathcal{Y}) = H(\mathcal{X})$

(c) Cas totalement aléatoire : $I(\mathcal{X};\mathcal{Y}) = 0$

Figure 9.6. Séquences conjointement typiques : influence de l'information mutuelle

Nous ne donnons pas la démonstration de ce théorème, qui se base essentiellement sur une suite de majorations/minorations appliquées à la conséquence de la loi des grand nombres. Mais nous allons commenter la signification des différentes propriétés des ensembles conjointenement ou simplement typiques. (Par opposition à l'ensemble conjointement typique, nous appelons ensemble *simplement* typique l'ensemble des couples (X^n, Y^n) tels que $| -\frac{1}{n} \log P(X^n, Y^n) - H(\mathcal{X},\mathcal{Y})| \leq \epsilon$)

La figure 9.6 montre graphiquement comment les séquences conjointement typiques se placent en fonction des espaces d'entrée et de sortie. Sur cette figure, les ronds creux représentent l'ensemble de séquences conjointement typiques et les ronds pleins sur l'axe vertical (resp. horizontal) les ensembles des séquences (typiques) d'entrée (resp. de sortie) qui leur correspondent. La figure illustre trois différentes configurations en fonction de la quantité d'information mutuelle entre symboles d'entrée et de sortie.

Notons que la projection des séquences conjointement typiques sur les espaces d'entrée ou de sortie converge vers les séquences typiques de cet espace. Cependant, le nombre total de séquences conjointement typiques est en général inférieur au produit du nombre de séquences d'entrée typiques par le nombre de séquences typiques de sortie (voir figures 9.6). L'exception qui confirme cette règle se produit lorsque les séquences d'entrée et de sortie sont indépendantes.

Les deux premières propriétés de l'ensemble conjointement typique montrent que malgré le fait que celui-ci soit en principe plus petit que l'ensemble simplement typique, il continue néanmoins à contenir presque toute la probabilité, et reste du point de vue cardinalité comparable à l'ensemble simplement typique. Cela veut dire que l'ajout des contraintes sur la typicalité des probabilités marginales ne modifie pas les propriétés fondamentales de l'ensemble $A_\epsilon^{(n)}$.

La troisième condition est plus difficile à interpréter, mais c'est réellement elle qui caractérise la typicalité conjointe. Elle montre que si $I(\mathcal{X},\mathcal{Y})$ est strictement positive, et lorsqu'on s'intéresse à des messages longs, alors la probabilité que certaines combinaisons d'entrées et de sorties appartiennent à cet ensemble tend rapidement vers zéro. Il s'agit des paires dont les entrées et les sorties sont distribuées selon les lois marginales $P(\mathcal{X}^n)$ et $P(\mathcal{Y}^n)$ mais dont la partie sortie est choisie de façon indépendante de la partie entrée.

De telles paires sont évidemment avec une grande probabilité marginalement typiques. Donc, la probabilité de trouver dans l'ensemble conjointement typique une séquence qui ne soit pas typique par rapport à $P(\mathcal{X},\mathcal{Y})$, tout en restant marginalement typique sur $\mathcal{X}$ et $\mathcal{Y}$, tend vers zéro lorsque n tend vers l'infini. Notons que dans le cas où $I(\mathcal{X},\mathcal{Y}) = 0$, ces séquences sont évidemment conjointement typiques, puisque dans ce cas $P(\mathcal{X}^n)P(\mathcal{Y}^n) = P(\mathcal{X}^n,\mathcal{Y}^n)$. Dans ce cas, la troisième condition n'apporte rien de plus.

Les trois conditions peuvent donc s'interpréter en disant que la typicalité simple de $P(\mathcal{X}^n,\mathcal{Y}^n)$ est nécessaire et suffisante à la typicalité marginale de $P(\mathcal{X}^n)$ et $P(\mathcal{Y}^n)$, et donc aussi à la typicalité conjointe de $P(\mathcal{X}^n,\mathcal{Y}^n)$. D'autre part, lorsque on s'intéresse à des message longs, ceux-ci ont de très fortes chances d'être conjointement typiques.

Suggestion : montrer que la probabilité conditionnelle pour que (X^n, Y^n) *soient conjointement typiques sachant que* X^n *(resp.* Y^n*) est typique, peut être majorée par* $1-\epsilon'$*, où* ϵ' *peut être rendu arbitrairement petit en prenant* n *suffisamment grand.*

9.2.4 Second théorème de Shannon

Nous sommes mûrs pour nous lancer dans la démonstration du second théorème de Shannon, qui est formulé, en termes des notions introduites auparavant, comme suit :

Second théorème de Shannon : codage de canal. Tous les débits R tels que $R < C$ sont réalisables (condition suffisante). Réciproquement, si un débit est réalisable, on a $R \leq C$ (condition nécessaire).

9.2.4.1 Preuve de la condition suffisante de réalisabilité.

La démonstration que nous allons fournir est assez proche de celle que Shannon esquissa dans ces travaux originaux. La différence essentielle, qui rend la preuve plus simple, réside ici dans la règle de décodage utilisée dans la preuve : nous utilisons le decodage par énumération des ensembles typiques, alors que Shannon utilisa le décodage au maximum de vraisemblance.

La démonstration se décompose en trois étapes : (i) construction de codes aléatoires; (ii) analyse et majoration de la probabilité d'erreur moyenne des codes aléatoires; (iii) sélection de bons codes parmi les codes aléatoires.

Construction de codes aléatoires $(\lceil 2^{nR} \rceil, n)$ pour $R < C$. Le procédé est le suivant (données : alphabets d'entrée et de sortie, matrice de transition) :

1. choix arbitraire de $P(\mathcal{X})$ et calcul de $I(\mathcal{X};\mathcal{Y})$;
2. choix quelconque de $R < I(\mathcal{X};\mathcal{Y})$ et calcul de $M = \lceil 2^{nR} \rceil$;
3. construction de la table de code par $M \times n$ tirages aléatoires successifs et indépendants de $X_j(i)$ selon la loi $P(\mathcal{X})$, nous noterons la table de code par $\mathcal{C}$ qui peut s'écrire

$$\mathcal{C} = \begin{bmatrix} X_1(1) & X_2(1) & \cdots & X_n(1) \\ \vdots & \vdots & \ddots & \vdots \\ X_1(M) & X_2(M) & \cdots & X_n(M) \end{bmatrix} \tag{9.41}$$

La probabilité de choisir une table de code particulière est donc

$$P(\mathcal{C}) = \prod_{w=1}^{M} \prod_{i=1}^{n} P(\mathcal{X} = X_i(w)). \tag{9.42}$$

Le système de communication est ensuite mis en route de la manière suivante

1. Choix du code $\mathcal{C}$, comme décrit ci-dessus.
2. Les encodeurs et décodeurs reçoivent connaissance du code et de la matrice de probabilités de transition du canal $P(\mathcal{Y}|\mathcal{X})$. Le décodeur construit pour chaque ligne de la table de code une table de messages de sortie qui regroupe toutes les suites Y^n conjointement typiques avec cette entrée.
3. Un message W est choisi selon la distribution uniforme $P(W = w) = M^{-1}, \forall w = 1, \ldots, M$.

4. Le $w-$ème mot de code $X^n(w)$ est lu dans la table et envoyé sur le canal.

5. Le message est déformé sur le canal, il en sort une suite Y^n distribuée selon la loi conditionnelle $P(Y^n|X^n(w))$.

6. Le décodeur devine quelle séquence fut envoyée au moyen du procédé de recherche exhaustive suivant : il parcourt les M tables des messages conjointement typiques avec les M mots de code d'entrée, et identifie les mots d'entrée qui comportent dans leur table la séquence reçue Y^n : si le résultat est une liste vide il déclare forfait et renvoie $\hat{W} = 0$ (une erreur d'emblée); si la liste ne comporte qu'un seul indice il retourne celui-ci comme indice estimé $\hat{W}$; si la liste comporte plusieurs indices il déclare encore une erreur d'emblée et renvoie $\hat{W} = 0$.

7. Le mot $\hat{W}$ est comparé à W et une erreur est comptabilisée si les deux ne sont pas identiques. Notons qu'il y a donc erreur si, et seulement si, l'une au moins des deux conditions suivantes est réalisée : le message envoyé n'est pas conjointement typique avec la sortie, ou s'il existe au moins un mot de code erroné qui est conjointement typique avec la sortie.

Majoration de la moyenne des probabilités d'erreur (moyenne algébrique) des codes aléatoires. Au lieu de calculer la probabilité d'erreur moyenne (algébrique) d'un code particulier, nous allons calculer l'espérance mathématique de cette grandeur pour une distribution aléatoire de codes sélectionnés comme ci-dessus. L'avantage de ce procédé est de symétriser et donc de simplifier grandement les calculs. Une fois que nous aurons trouvé une borne supérieure de cette moyenne, nous pourrons alors en déduire qu'il existe forcément des codes dont la probabilité d'erreur est inférieure à cette borne et nous aurons donc, gratuitement, la preuve d'existence de codes acceptables du point de vue probabilité d'erreur *moyenne algébrique*. Nous verrons alors qu'il est possible d'en déduire l'existence de codes dont la probabilité d'erreur *maximale* est arbitrairement petite.

Dans ce qui suit nous allons désigner par E l'événement $\hat{W} \neq W$. On a donc, en désignant par $\mathcal{C}$ un code quelconque et par $P(E)$ l'espérance mathématique de la probabilité d'erreur moyenne algébrique des codes aléatoires

$$\begin{aligned} P(E) &= \sum_{\mathcal{C}} P(\mathcal{C}) P_e^{(n)}(\mathcal{C}) && (9.43)\\ &= \sum_{\mathcal{C}} P(\mathcal{C}) M^{-1} \sum_{w=1}^{M} \lambda_w(\mathcal{C}) && (9.44)\\ &= M^{-1} \sum_{w=1}^{M} \sum_{\mathcal{C}} P(\mathcal{C}) \lambda_w(\mathcal{C}), && (9.45)\end{aligned}$$

mais comme les codes sont générés aléatoirement, la somme $\sum_{\mathcal{C}} P(\mathcal{C})\lambda_w(\mathcal{C})$ ne dépend pas en fait du choix du mot w, et nous pouvons dès lors supposer, sans perte de généralité, que le message $W = 1$ fut envoyé. Nous poursuivons donc notre raisonnement sous cette hypothèse : on a

$$P(E) = M^{-1} \sum_{w=1}^{M} \sum_{\mathcal{C}} P(\mathcal{C})\lambda_1(\mathcal{C}) = \sum_{\mathcal{C}} P(\mathcal{C})\lambda_1(\mathcal{C}) = P(E|W = 1). \quad (9.46)$$

Définissons alors les événements suivants

$$E_i = \{(X^n(i), Y^n) \in A_\epsilon^{(n)}\}, \forall i \leq M, \quad (9.47)$$

c'est-à-dire l'événement “le $i-$ème mot de code est conjointenement typique avec la sortie”. Un erreur se produit donc si soit $\neg E_1$ soit $E_2 \cup E_3 \cup \ldots \cup E_M$ se réalisent. En d'autres termes

$$P(E|W = 1) = P(\neg E_1 \cup E_2 \cup E_3 \cup \ldots \cup E_M) \leq P(\neg E_1) + \sum_{i=2}^{M} P(E_i). \quad (9.48)$$

Mais, le théorème AEP implique, d'une part, que $P(\neg E_1) \to 0$ et donc pour n suffisamment grand on a

$$P(\neg E_1) \leq \epsilon. \quad (9.49)$$

D'autre part, puisque les mots de code ont été générés indépendamment (mais selon une même loi marginale), les $X^n(i), i \neq 1$ sont indépendantes de $X^n(1)$, et donc également de la sortie Y^n. Mais, les $X^n(i)$ sont distribués selon la même loi marginale que $X^n(1)$. Donc la probabilité pour que l'un de ces mots de code soit conjointement typique avec Y^n est bornée par $2^{-n(I(\mathcal{X};\mathcal{Y})-3\epsilon)}$ (propriété no 3 du théorème AEP conjoint).

Par conséquent, en substituant ces deux bornes dans (9.48), on trouve que

$$P(E) = P(E|W=1) \leq \epsilon + \sum_{i=2}^{M} 2^{-n(I(\mathcal{X};\mathcal{Y})-3\epsilon)} \tag{9.50}$$

ou encore

$$P(E) \leq \epsilon + (M-1)2^{-n(I(\mathcal{X};\mathcal{Y})-3\epsilon)} < \epsilon + 2^{-n(I(\mathcal{X};\mathcal{Y})-R-3\epsilon)}, \tag{9.51}$$

puisque $M = \lceil 2^{nR} \rceil$ entraîne que $M - 1 < 2^{nR}$.

Alors, pour autant autant que $I(\mathcal{X};\mathcal{Y}) - R > 3\epsilon$ l'exposant du deuxième terme sera strictement négatif. Par conséquent, en choisissant $R < I(\mathcal{X};\mathcal{Y}) - 3\epsilon$ et n suffisamment grand, on peut rendre $P(E) < 2\epsilon$.

Sélection de bons codes. Pour terminer la preuve, il faut encore effectuer une série de raffinements qui visent à choisir un code correctement :

1. Soit $P^*(\mathcal{X})$ la distribution de probabilité qui maximise $I(\mathcal{X};\mathcal{Y})$ et donc qui satisfait $I(\mathcal{X};\mathcal{Y}) = C$. Alors en utilisant cette distribution de probabilité nous pouvons remplacer $I(\mathcal{X};\mathcal{Y})$ par C donnant la condition $R < C - 3\epsilon$, qui est équivalente à $R < C$.

2. Débarrassons nous ensuite de la moyenne sur toutes les tables de code. Puisque $P(E) \leq 2\epsilon$ en moyenne sur tous les codes, il existe au moins un code aléatoire, disons $\mathcal{C}^*$ tel que $P_e^{(n)}(\mathcal{C}^*) \leq 2\epsilon$. Pour le trouver, il suffirait d'énumérer tous les codes possibles de façon exhaustive jusqu'à en trouver un qui réponde au critère.

3. Nous avons donc montré l'existence d'un bon code du point de vue de la probabilité d'erreur moyenne algébrique. Montrons comment en déduire un code bon du point de vue de la probabilité d'erreur *maximale*. Enlevons de la table de ce code la moitié la plus mauvaise des mots de code, du point de vue de leur probabilité d'erreur. Les mots restant auront alors une probabilité d'erreur $\lambda_i(\mathcal{C}^*) \leq 4\epsilon$, car sinon la probabilité d'erreur moyenne du code ne pourrait être inférieure à 2ϵ. Nous avons maintenant un code de longueur 2^{nR-1} d'erreur maximale $\leq 4\epsilon$.

En d'autre termes, en prenant $R' < C - \frac{1}{n}$ il est possible de s'arranger pour obtenir un code dont la probabilité d'erreur maximale est arbitrairement petite. En prenant n suffisamment grand on peut donc s'approcher arbitrairement près de C, tout en rendant arbitrairement petite la probabilité d'erreur maximale. Cela prouve donc bien l'atteignabilité $\forall R < C$. $\square$

9.2.4.2 Inégalité de Fano et réciproque du second théorème.

Avant de prouver la réciproque du second théorème de Shannon, nous allons d'abord démontrer quelques inégalités remarquables qui nous seront utiles ici, et qui présentent également un intérêt général.

Inégalité de Fano. Soient $\mathcal{X}$ et $\mathcal{Y}$ deux variables aléatoires discrètes, et supposons que nous souhaitions deviner la valeur de X connaissant Y. En d'autres termes nous cherchons une fonction $g(Y)$ telle que $P_e = P(g(\mathcal{Y}) \neq \mathcal{X})$ soit la plus faible possible, éventuellement nulle. L'inégalité de Fano fournit une relation entre P_e et l'entropie conditionnelle $H(\mathcal{X}|\mathcal{Y})$:

$$H_2(P_e) + P_e \log(|\mathcal{X}| - 1) \geq H(\mathcal{X}|\mathcal{Y}). \tag{9.52}$$

Pour établir cette inégalité, remarquons d'abord que si $\mathcal{X}$ est une fonction de $\mathcal{Y}$ on peut évidemment estimer X à l'aide de cette fonction sans erreurs. Dans ce cas on a évidemment $P_e = 0$. Cette situation est équivalente à $H(\mathcal{X}|\mathcal{Y}) = 0$. On a donc $H(\mathcal{X}|\mathcal{Y}) = 0 \Rightarrow P_e = 0$. Essayons alors de relier P_e à $H(\mathcal{X}|\mathcal{Y})$ lorsque celle-ci est non-nulle.

Soit $g(Y)$ une fonction quelconque de Y; alors $\mathcal{X} \leftrightarrow \mathcal{Y} \leftrightarrow g(\mathcal{Y})$ forme une chaîne de Markov. Calculons la probabilité d'erreur $P_e = P(g(\mathcal{Y}) \neq \mathcal{X})$. Désignons par $\mathcal{E}$ une variable aléatoire indicatrice de l'erreur, i.e.

$E = 1 - \delta_{g(Y),X}$. On a donc, $P_e = P(\mathcal{E} = 1)$, et par conséquent $H(\mathcal{E}) = H_2(P_e) = -(P_e \log P_e + (1 - P_e)\log(1 - P_e))$.

D'autre part, puisque E est une fonction de X et de Y on a $H(\mathcal{E}|\mathcal{X}, \mathcal{Y}) = 0$.

Par ailleurs

$$H(\mathcal{X}|\mathcal{E}, \mathcal{Y}) = (1 - P_e)H(\mathcal{X}|\mathcal{Y}, \mathcal{E} = 0) + P_e H(\mathcal{X}|\mathcal{Y}, \mathcal{E} = 1). \tag{9.53}$$

Mais lorsque $\mathcal{E} = 0$ on connaît X (pas d'erreur), et le premier terme de (9.53) est donc nul. D'autre part, lorque $\mathcal{E} = 1$ on peut exclure une valeur possible pour $\mathcal{X}$ (celle devinée par $g(Y)$), il ne reste donc plus que $|\mathcal{X}| - 1$ valeurs possibles pour $\mathcal{X}$ et l'entropie conditionnelle $H(\mathcal{X}|\mathcal{Y}, \mathcal{E} = 1)$ est majorée par $\log(|\mathcal{X}| - 1)$. Donc

$$H(\mathcal{X}|\mathcal{E}, \mathcal{Y}) \leq P_e \log(|\mathcal{X}| - 1). \tag{9.54}$$

Developpons alors $H(\mathcal{E}, \mathcal{X}|\mathcal{Y})$ de deux façons différentes

$$\begin{aligned} H(\mathcal{E}, \mathcal{X}|\mathcal{Y}) &= H(\mathcal{X}|\mathcal{Y}) + H(\mathcal{E}|\mathcal{X}, \mathcal{Y}) && (9.55)\\ &= H(\mathcal{E}|\mathcal{Y}) + H(\mathcal{X}|\mathcal{E}, \mathcal{Y}) && (9.56) \end{aligned}$$

puisque $H(\mathcal{E}|\mathcal{X}, \mathcal{Y}) = 0$ on en déduit que

$$H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{E}|\mathcal{Y}) + H(\mathcal{X}|\mathcal{E}, \mathcal{Y}) \tag{9.57}$$

et en y substituant les inégalités

$$\begin{aligned} H(\mathcal{E}|\mathcal{Y}) &\leq H(\mathcal{E}) = H_2(P_e), && (9.58)\\ H(\mathcal{X}|\mathcal{E}, \mathcal{Y}) &\leq P_e \log(|\mathcal{X}| - 1) && (9.59) \end{aligned}$$

et en réarrangeant on obtient *l'inégalité de Fano*

$$H_2(P_e) + P_e \log(|\mathcal{X}| - 1) \geq H(\mathcal{X}|\mathcal{Y}). \tag{9.60}$$

Notons que $P_e = 0$ implique bien $H(\mathcal{X}|\mathcal{Y}) = 0$ comme suggéré par l'intuition.

L'inégalité de Fano (9.60) peut être affaiblie en remplaçant $H_2(P_e)$ par 1 (sa borne supérieure), et $(|\mathcal{X}| - 1)$ par $|\mathcal{X}|$, pour donner

$$P_e \geq \frac{H(\mathcal{X}|\mathcal{Y}) - 1}{\log |\mathcal{X}|}. \tag{9.61}$$

Inégalité de Fano pour le canal discret sans mémoire. Nous supposons que les indices d'entrée $W \in \mathcal{W} = \{1, 2, \ldots, \lceil 2^{nR} \rceil\}$ de la table de code $\mathcal{C}$ sont distribués de façon uniforme, de telle façon que la probabilité d'erreur P_e de décodage soit égale à la moyenne arithmétique $P_e^{(n)}$. Nous cherchons à deviner W à l'aide d'une fonction quelconque $\hat{W} = g(Y^n)$. On a alors la transcription suivante de l'inégalité de Fano.

Pour un canal discret sans mémoire de table de code $\mathcal{C}$ et dont les messages d'entrée sont uniformément distribués, et utilisant une fonction de décodage $\hat{W} = g(Y^n)$ quelconque, on a nécessairement

$$H(\mathcal{W}|\mathcal{Y}^n) \leq 1 + P_e^{(n)} nR. \tag{9.62}$$

Cela résulte directement de (9.60), en tenant compte du fait que $H_2(P_e^{(n)}) \leq 1$ et que $\lceil 2^{nR} \rceil - 1 \leq 2^{nR}$.

Comme on a d'autre part, $H(\mathcal{X}^n(\mathcal{W})|\mathcal{Y}^n) \leq H(\mathcal{W}|\mathcal{Y}^n)$ on en déduit que

$$H(\mathcal{X}^n(\mathcal{W})|\mathcal{Y}^n) \leq 1 + P_e^{(n)} nR. \tag{9.63}$$

Capacité du canal étendu. Soient $\mathcal{X}^n$ *des chaînes d'entrée d'un canal discret sans mémoire,* $\mathcal{Y}^n$ *les sorties correspondantes, et* $P(\mathcal{X}^n)$ *une distribution de probabilité quelconque des entrées. Alors*

$$I(\mathcal{X}^n; \mathcal{Y}^n) \leq nC \tag{9.64}$$

où C désigne la capacité en information du canal.

En d'autres termes la capacité de l'extension d'ordre n du canal (notons la C^n) vérifie $C^n \leq nC$, puisque c'est le maximum du membre de gauche de (9.64). Comme, d'autre part, il est possible de trouver $P(\mathcal{X}^n)$ tels que $I(\mathcal{X}^n;\mathcal{Y}^n) = nC$ (on peut prendre les $\mathcal{X}_i$ indépendants et distribuées selon la loi qui réalise C), on a

$$C^n = nC, \tag{9.65}$$

ce qui est assez rassurant.

La démonstration de (9.64) est assez immédiate. On a

$$\begin{aligned}
I(\mathcal{X}^n;\mathcal{Y}^n) &= H(\mathcal{Y}^n) - H(\mathcal{Y}^n|\mathcal{X}^n) &(9.66)\\
&= H(\mathcal{Y}^n) - \sum_{i=1}^{n} H(\mathcal{Y}_i|\mathcal{Y}_1,\mathcal{Y}_2,\ldots,\mathcal{Y}_{i-1},\mathcal{X}^n) &(9.67)\\
&= H(\mathcal{Y}^n) - \sum_{i=1}^{n} H(\mathcal{Y}_i|\mathcal{X}_i) &(9.68)\\
&\leq \sum_{i=1}^{n} H(\mathcal{Y}_i) - \sum_{i=1}^{n} H(\mathcal{Y}_i|\mathcal{X}_i) &(9.69)\\
&= \sum_{i=1}^{n} I(\mathcal{X}_i;\mathcal{Y}_i) &(9.70)\\
&\leq nC, &(9.71)
\end{aligned}$$

où le passage de (9.66) à (9.67) résulte de l'application de la règle de chaînage des entropies, le passage de (9.67) à (9.68) provient de la définition du canal sans mémoire, le passage de (9.68) à (9.69) tient compte du fait que les entropies conjointes sont inférieures ou égales à la somme des entropies marginales, et enfin le passage de (9.70) à (9.71) par application de la définition de la capacité ($C \geq I(\mathcal{X}_i;\mathcal{Y}_i)$). □

Réciproque du second théorème. *N'importe quelle séquence de codes $(\lceil 2^{nR}\rceil, n)$ telle que $\lambda^{(n)} \to 0$ doit avoir $R \leq C$.*

Démonstration. Notons tout d'abord que $\lambda^{(n)} \to 0 \Rightarrow P_e^{(n)} \to 0$. Supposons donnée la fonction d'encodage quelconque $X^n(W)$ qui choisit une séquence d'entrée pour un indice d'entrée donné. Supposons que $\mathcal{W}$ soit distribuée uniformément (elle prend $\lceil 2^{nR}\rceil$ valeurs possibles).

On peut donc supposer que les entrées sont distribuées uniformément. On a

$$\begin{aligned}
nR \leq \log\lceil 2^{nR}\rceil &= H(\mathcal{W}) = H(\mathcal{W}|\mathcal{Y}^n) + I(\mathcal{W};\mathcal{Y}^n) &(9.72)\\
&\leq H(\mathcal{W}|\mathcal{Y}^n) + I(\mathcal{X}^n(\mathcal{W});\mathcal{Y}^n) &(9.73)\\
&\leq 1 + P_e^{(n)} nR + nC, &(9.74)
\end{aligned}$$

qui est justifié par les deux inégalités (9.62) et (9.64), et le principe de non-création d'information. En divisant par n on obtient

$$R \leq P_e^{(n)} R + \frac{1}{n} + C, \tag{9.75}$$

et en faisant tendre n vers l'infini on a finalement $R \leq C$. □

On peut aussi réécrire (9.75) sous la forme

$$P_e^{(n)} \geq 1 - \frac{C}{R} - \frac{1}{nR}, \tag{9.76}$$

ce qui donne asymptotiquement $R > C \Rightarrow P_e^{(\infty)} > 0$. Par conséquent, si $R > C$ et pour n suffisamment grand tout code finit par avoir une erreur strictement positive. Cela implique que tout code (même pour n fini) doit avoir une erreur strictement positive, car s'il existait un code parfait $(\lceil 2^{nR}\rceil, n)$ tel que $P_e^{(n)} = 0$, pour une valeur finie de n, alors on peut construire par simple concaténation de celui-ci autant de codes parfaits pour des valeurs de $n' = kn$ aussi grandes que souhaitées, tels que $P_e^{(n')} = 0$.

Cette propriété est parfois appelée la *réciproque faible* du second théorème. Il est en effet possible de montrer une réciproque forte qui dit que lorsque $R > C$, alors $P_e^{(n)} \to 1$ exponentiellement vite.

La capacité est donc un point de *bifurcation* très clair : en dessous on peut faire tendre $P_e^{(n)}$ vers 0, à vitesse exponentielle, au-dessus on a beau faire tout ce qu'on veut, elle tendra exponentiellement vers 1.

9.2.5 De l'égalité dans la réciproque

Examinons sous quelles conditions on peut atteindre la capacité avec une probabilité d'erreur nulle.

On peut répéter les différentes étapes ci-dessus :

$$\begin{aligned}
nR &\stackrel{a}{=} H(\mathcal{W}) && (9.77)\\
&\stackrel{b}{=} H(\mathcal{X}^n(\mathcal{W})) && (9.78)\\
&= H(\mathcal{X}^n|\mathcal{Y}^n) + I(\mathcal{X}^n;\mathcal{Y}^n) && (9.79)\\
&\stackrel{c}{=} I(\mathcal{X}^n;\mathcal{Y}^n) = H(\mathcal{Y}^n) - H(\mathcal{Y}^n|\mathcal{X}^n) && (9.80)\\
&\stackrel{d}{=} \sum_{i=1}^{n} H(\mathcal{Y}_i) - \sum_{i=1}^{n} H(\mathcal{Y}_i|\mathcal{X}_i) && (9.81)\\
&= \sum_{i=1}^{n} I(\mathcal{X}_i;\mathcal{Y}_i) && (9.82)\\
&= nC, && (9.83)
\end{aligned}$$

où on peut toujours s'arranger pour (a) soit possible, et, (b) est permis à condition que tous les mots de code soient distincts, (c) est permis car $H(\mathcal{X}^n|\mathcal{Y}^n) = 0$ si $P_e^{(n)} = 0$, le passage à (d) est possible si les $\mathcal{Y}_i$ sont indépendants, et enfin le dernier passage est possible en choisissant les $\mathcal{X}_i$ distribués de façon identique selon la loi qui réalise C.

On voit donc que les entrées doivent être codées de manière à ce que les $\mathcal{Y}_i$ aient l'air indépendants et distribuées de façon identique selon la loi induite par celle qui réalise la capacité :

$$P^*(Y) = \sum_{\mathcal{X}} P^*(X)P(Y|X). \qquad (9.84)$$

9.2.6 Généralisation

Il est possible d'étendre la notion de capacité au cas du canal causal de mémoire finie, nourri par une source stationnaire et ergodique.

Les variables aléatoires X^n, Y^n sont alors stationnaires et ergodiques ce qui donne d'une part un sens à la définition de la quantité d'information mutuelle et à la capacité (stationnarité), par passage à la limite et division par n des entropies de suites finies.

D'autre part, comme les théorèmes AEP restent essentiellement applicables aux processus stationnaires et ergodiques on peut espérer que les raisonnements effectués ci-dessus restent applicables, et donc le second théorème de Shannon également.

Nous verrons aux cours de séances d'exercices que l'introduction d'une mémoire (finie) au niveau du canal conduit à une augmentation de sa capacité. Celà réduit en effet l'équivocation à la sortie du canal, les perturbations aléatoires aux symboles successifs étant corrélées, et donc moins "efficaces" en entropie.

9.3 CONSTRUCTION DE BONS CODES DE CANAL

Les codes tels que ceux que nous avons vus ci-dessus, qui consistent à attribuer des blocs isolés de symboles de canal à des messages, sont appelés *codes en blocs*. Nous verrons au chapitre 15 qu'il existe également d'autres types de codes qui superposent (codes convolutionnels) ou entrelacent (codes entrelacés) les séquences de sym-

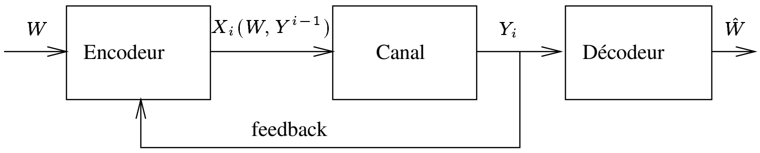


Figure 9.7. Canal discret sans mémoire avec feedback

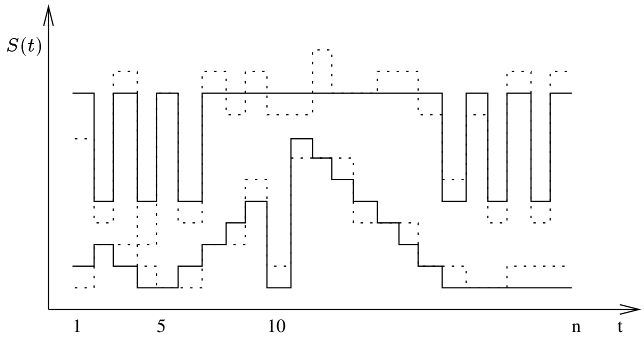


Figure 9.8. Représentation de messages par des signaux temporels discrets

boles relatives à des messages successifs. Pour le moment discutons un peu de la quête aux bons codes en blocs, provoquée par la publication du second théorème de Shannon.

Il est clair que la méthode utilisée ci-dessus pour démontrer l'atteignabilité pourrait permettre de construire des codes. Il suffirait en effet de construire une série de codes aléatoires, jusqu'à en trouver un qui ait une probabilité d'erreur suffisamment faible, en répétant l'opération pour des valeurs croissantes de n. Le problème de cette méthode est qu'on a de fortes chances d'aboutir à des longueurs de blocs très grandes, et donc à des tables de code de taille encore plus grande (elle croit exponentiellement avec la longueur des blocs). Ces tables seront difficiles à décoder car de structure quelconque (en fait aléatoire) : il faudra recourir à une recherche exhaustive pour identifier le mot d'entrée de vraisemblance maximale. Une amélioration des performances se payera donc par une augmentation très significative des temps de calculs au décodage.

Il ne faut donc pas s'étonner que depuis la publication de Shannon en 1948, de nombreux chercheurs se soient penchés sur la construction de codes correcteurs d'erreurs qui sont faciles à encoder/décoder. Depuis lors, de très nombreux codes ont été développés, mais les taux asymptotiques de ces codes ne s'approchent pas encore de la capacité. Nous en verrons quelques exemples dans le chapitre 15, où nous nous intéresserons plus particulièrement aux aspects d'implémentation des algorithmes d'encodage/décodage.

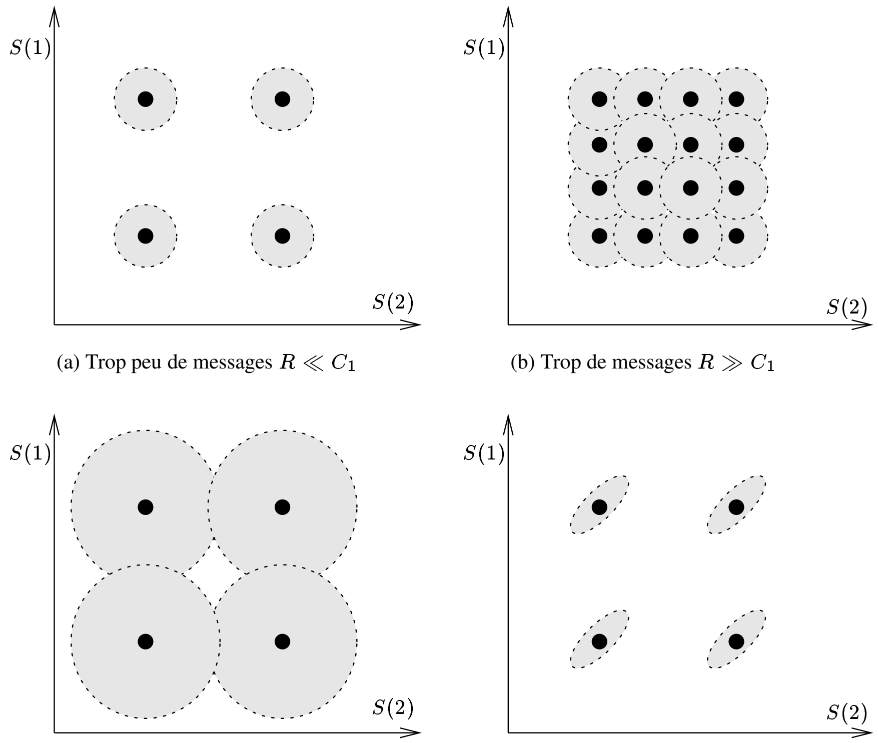
9.4 CAPACITE DE FEEDBACK

La figure 9.4 montre un schéma de communication avec feedback. Ici le symbole X_i envoyé sur le canal lorsqu'on veut transmettre un message W est une fonction $X_i(W, Y^{i-1})$ du message et des symboles précédemment reçus. Ce type de fonctionnement a comme cas particulier le mode étudié précédemment. Il doit donc être possible de faire au moins aussi bien. La question qui vient à l'esprit est de savoir si on peut faire mieux. La réponse que nous donnons sans démonstration est (malheureusement) négative.

9.5 INTERPRETATION GEOMETRIQUE DU CODAGE DE CANAL

Nous allons faire une analogie entre les messages considérés en théorie de l'information et les signaux temporels (en temps discret). Nous supposerons que les valeurs prises à chaque instant pour un tel signal appartiennent également à un ensemble discret, disons $\{1, 2, \ldots, M\}$. Nous supposerons que le bruit est additif dans notre exemple, c'est-à-dire que $\mathcal{Y} = \mathcal{X} + \mathcal{Z}$, où $\mathcal{Z}$ est une variable aléatoire qui représente le bruit.

La figure 9.8 montre deux signaux d'entrée, et les deux signaux de sortie qui en résultent après déformation par le bruit, qui est ici supposé sans mémoire.



(a) Trop peu de messages $R \ll C_1$

(b) Trop de messages $R \gg C_1$

(c) Bruit plus élevé : $R \gg C_2 \ll C_1$

(d) Bruit corrélé dans le temps : $C_3 \gg C_1$

Figure 9.9. Représentation des signaux sous forme géométrique

On voit que le bruit crée autour des signaux d'entrée une zone d'incertitude à l'intérieur desquelles les sorties qui lui correspondent peuvent se trouver. En d'autres mots, si nous nous représentons les signaux d'entrée et de sortie comme des vecteurs d'un espace à n dimensions, le bruit a pour effet de distribuer les signaux de sortie dans un certain voisinage du signal d'entrée. L'hypothèse d'additivité que nous avons supposée ci-dessus (§9.1.4), revient alors à dire que la forme de ces voisinages ne dépend pas de la localisation du point d'entrée. Si nous supposons de plus que le bruit est blanc (c'est-à-dire que sa distribution de probabilités à l'instant t est indépendante de t et du bruit aux instants précédents), cela veut dire que les voisinages ont une géométrie isotrope, c'est-à-dire identique dans toutes les directions de l'espace.

On voit intuitivement que pour un niveau de bruit donné, les signaux de sortie seront d'autant plus faciles à reconnaître, que les signaux d'entrée sont différents, c'est-à-dire d'autant plus que la "distance" entre les points correspondants est grande. De même, pour une configuration donnée des signaux d'entrée, le décodage sera d'autant moins facile que la taille du voisinage induit par le bruit est élevée, autrement dit que l'entropie $H(\mathcal{Z})$ de la variable de bruit est élevée, qui dans le cas du bruit additif est égale à l'equivocation $H(\mathcal{Y}|\mathcal{X})$. La figure 9.9 illustre quelques configurations possibles, où nous avons supposé que le nombre de valeurs possibles était très grand, et avons assimilé, pour les besoins de la cause, l'espace des signaux à une partie de $\mathbb{R}^2$ (en ne représentant que les deux premiers symboles émis par la source). Sur cette figure, les points noirs représentent les signaux d'entrée, et les régions en gris représentent les voisinages de sortie qui y sont induits par le bruit. Il s'agit de l'ensemble des messages conjointement typiques avec l'entrée correspondante.

Les différentes configurations entrée/sortie de la figure 9.9 mettent bien en évidence la problématique du codage de canal. Par exemple, les figures 9.9(a,b) montrent qu'il est possible de décoder les sorties correctement à condition que les voisinages induits par le bruit ne se recouvrent pas. Pour le niveau de bruit des figures 9.9(a,b), il est certainement possible de faire passer plus que 4 messages sur ce canal, et moins de 16. Les figures 9.9(c) montre l'influence d'une augmentation du niveau de bruit (qui se traduit par une réduction de la capacité du canal). Enfin, la figure 9.9(d) montre ce qui passe lorsque le bruit aux différents instants est corrélé (c'est-à-dire que le canal est à mémoire) : pour des distributions marginales identiques, la corrélation (i.e. la mémoire) se traduit par une réduction du volume des voisinages, donc par une augmentation de la capacité. Signalons également que dans le cas plus général où le bruit n'est pas indépendant des symboles d'entrée, la forme des voisinage serait variable d'un point à l'autre de l'espace des signaux.

Enfin, que devient le théorème AEP conjoint dans cette représentation ? Ce théorème dit que lorsque la dimension de l'espace croit, le volume des voisinages induits par le bruit décroit en relatif par rapport au volume complet disponible de l'espace des signaux, et ceci à une vitesse exponentielle. Il est alors possible de placer des signaux d'entrée en nombre croissant de façon exponentielle, sans que les voisinages ne se recouvrent, à condition que le taux de croissance R ne dépasse pas la capacité du signal. Si par contre, le taux de croissance du nombre de signaux d'entrée est supérieur (même légèrement), les voisinages vont de plus en plus encombrer l'espace, pour finir par se couvrir complètement.

Nous aurons l'occasion de revenir sur cette interprétation géométrique au chapitre suivant lorsque nous parlerons des canaux continus.

9.6 SYNTHESE : CODAGE CONJOINT SOURCE ET CANAL

Il est temps maintenant de combiner les deux principaux résultats obtenus pour le paradigme de Shannon : (i) compression de données ($\overline{n} = R_s \leq H$); (ii) transmission de données ($R_c < C$). Est-ce que la condition $H < C$ est nécessaire et suffisante pour permettre la transmission sans erreurs sur un canal des symboles émis par une source ?

Par exemple, considérons l'envoi de son digital au moyen d'un canal discret et sans mémoire. On pourrait construire un code pour transformer la suite d'échantillons sonores directement en suite de symboles d'entrée du canal, ou bien nous pouvons d'abord comprimer cette suite en une représentation plus compacte, puis envoyer celle-ci sur le canal au moyen d'un code adéquat. Il n'est pas évident, a priori, que nous ne perdons pas en efficacité en utilisant la méthode en deux étapes, puisque la compression de données ne fait pas référence aux propriétés du canal, et que le code canal ne prend pas en compte la distribution de la source.

Nous allons montrer dans cette dernière section que cette méthode en deux étapes est aussi bonne que toute autre méthode de transmission d'information sur un canal bruité. La conséquence pratique très importante est que nous pouvons concevoir le système de transmission (canal + encodeur/décodeur) indépendamment des propriétés des sources qui utiliseront celui-ci. D'autre part, nous pouvons développer un code approprié pour une source quelconque, sans nous préoccuper des propriétés du canal qui sera utilisé ultérieurement (le même algorithme de compression pourra être utilisé, que ce soit pour stocker des données sur un disque, ou que ce soit pour les envoyer sur Internet). Comme les systèmes digitaux utilisent pratiquement exclusivement la représentation binaire, il n'y a pas en pratique de nécessité de conversion d'alphabet.

Le fait que la méthode en deux étapes optimales soit en principe aussi bonne que n'importe quelle méthode en une seule étape semble tellement évident qu'il est bon de souligner qu'il existe des situations où ce n'est pas le cas. Ainsi, il y a des exemples de canaux à utilisateurs multiples où la décomposition conduit à une certaine sous-optimalité.

Plus intuitivement, bien souvent les caractéristiques des sources (leur redondance) est en fait parfaitement adaptée et permettent de corriger un grand nombre d'erreurs de transmission. Par exemple, si nous considérons la langue (française où anglaise) écrite, nous pouvons envoyer des message écrits sur un canal hautement bruité tout en étant capable de détecter et de corriger les erreurs, grâce à notre connaissance de la langue. Un code optimal pour la langue serait évidemment nettement plus compact[2], cependant l'envoi de tels messages compressés nécessiterait à son tour un code de canal très efficace pour combattre le bruit. Etant donné la difficulté de trouver de bons codes correcteurs d'erreurs, il n'est pas certain que le rendement de l'opération soit spectaculaire. Ces situations, où la structure de la source est en réalité adaptée aux caractéristiques du milieu physique utilisé pour la transmission sont fréquentes.

Décrivons le schéma de communication que nous allons considérer pour notre discussion. Nous avons une source S qui génère des symboles appartenant à un alphabet $\mathcal{S} = \{S_1, S_2, \ldots S_q\}$. La seule autre hypothèse que nous faisons est que le processus aléatoire correspondant aux symboles émis par cette source obéit aux conditions d'applicabilité du théorème AEP. Rappelons que les sources stationnaires sans mémoire, les sources de Markov stationnaires, et en fait n'importe quel processus stationnaire et ergodique, répondent à ce critère. D'autre part, le codage par blocs à l'entrée du canal sans mémoire et la transmission sur celui-ci préservent ces propriétés,

[2]Par exemple en utilisant des modèles markoviens d'ordre 100, on peut obtenir des facteurs de compression de l'ordre de 4 (environ 1bit/lettre).

et par conséquent le processus à la sortie satisfait encore à l'AEP, et de plus les entrées codées et sorties prises conjointement satisfont à l'AEP conjoint.

Nous voulons envoyer la suite de symboles $\mathcal{S}^n = \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ sur le canal de telle manière que le récepteur puisse la reconstruire. A cette fin, nous transformons la suite en un mot de code $\mathcal{X}^n(\mathcal{S}^n)$ et l'envoyons sur le canal. Le récepteur regarde la séquence reçue $\mathcal{Y}^n$ et fait une devinette $\hat{\mathcal{S}}^n = g(\mathcal{Y}^n)$ de $\mathcal{S}^n$; il commet une erreur si $\hat{\mathcal{S}}^n \neq \mathcal{S}^n$. Nous définissons la probabilité d'erreurs $P_e^{(n)}$ par

$$P_e^{(n)} \triangleq P(\hat{\mathcal{S}}^n \neq \mathcal{S}^n) = \sum_{\mathcal{Y}^n} \sum_{\mathcal{S}^n} P(\mathcal{S}^n) P(\mathcal{Y}^n | \mathcal{X}^n(\mathcal{S}^n))(1 - \delta_{g(\mathcal{Y}^n), \mathcal{S}^n}), \tag{9.85}$$

où $g(\mathcal{Y}^n)$ est la fonction de décodage et le dernier facteur comptabilise donc les cas où il y a erreur de décodage.

Nous pouvons alors formuler le théorème suivant.

Théorème : codage source-canal. *si $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ est un processus stochastique discret et à valeurs discrètes qui a la propriété d'équipartition asymptotique, alors il existe un code source-canal tel que $P_e^{(n)} \to 0$ dès lors que $H(\mathcal{S}) < C$.*

Réciproquement, pour n'importe quel processus stationnaire, si $H(\mathcal{S}) > C$, il est impossible d'envoyer ce processus sur le canal avec une probabilité d'erreurs arbitrairement petite.

Démonstration.

Atteignabilité : Nous allons utiliser le codage en deux étapes. Puisque le processus $\mathcal{S}$ répond à l'AEP, on peut prendre n suffisamment grand de telle manière que presque toutes la probabilité soit concentrée sur l'ensemble typique $A_\epsilon^{(n)}$ qui contiendra au maximum $2^{n(H(\mathcal{S})+\epsilon)}$ messages de longueur n. Nous encodons seulement ces messages, et associons un mot de code quelconque aux autres (qui seront donc décodés erronément). Puisqu'il y a au plus $2^{n(H(\mathcal{S})+\epsilon)}$ messages, $n(H + \epsilon)$ bits suffisent pour les identfier. Nous pouvons donc transmettre l'identificateur avec une probabilité d'erreur inférieure à ϵ si

$$H(\mathcal{S}) + \epsilon < C. \tag{9.86}$$

Le récepteur reconstruit les messages orginaux S^n en énumérant leur ensemble typique. La probabilité d'erreur totale sera

$$\begin{aligned} P_e^{(n)} &= P(\hat{\mathcal{S}}_n \neq \mathcal{S}^n) && (9.87)\\ &= P(\hat{\mathcal{S}}_n \neq \mathcal{S}^n | \mathcal{S}^n \notin A_\epsilon^{(n)})(1 - P(A_\epsilon^{(n)})) \\ &\quad + P(\hat{\mathcal{S}}_n \neq \mathcal{S}^n | \mathcal{S}^n \in A_\epsilon^{(n)}) P(A_\epsilon^{(n)}) && (9.88)\\ &\leq \epsilon + \epsilon && (9.89) \end{aligned}$$

pour n suffisamment grand. Donc, si $H(\mathcal{S}) < C$ il est possible de reconstruire les messages avec une probabilité d'erreur arbitrairement faible.

Réciproque : Nous voulons monter que $P_e^{(n)} \to 0$ implique $H(\mathcal{S}) \leq C$ pour toute suite de paires de codes canal-source :

$$\begin{aligned} X^n(S^n) &: \mathcal{S}^n \to \mathcal{X}^n, && (9.90)\\ g(Y^n) = \hat{S}^n(Y^n) &: \mathcal{Y}^n \to \mathcal{S}^n. && (9.91) \end{aligned}$$

L'inégalité de Fano donne :

$$H(\mathcal{S}^n | \hat{\mathcal{S}}^n) \leq 1 + P_e^{(n)} n \log |\mathcal{S}|. \tag{9.92}$$

Par conséquent

$$\begin{aligned} H(\mathcal{S}) &\overset{a}{\leq} \frac{H(\mathcal{S}^n)}{n} = \frac{H(\mathcal{S}^n|\hat{\mathcal{S}}^n)}{n} + \frac{I(\mathcal{S}^n; \hat{\mathcal{S}}^n)}{n} && (9.93)\\ &\overset{b}{\leq} \frac{1}{n}(1 + P_e^{(n)} n \log |\mathcal{S}|) + \frac{1}{n} I(\mathcal{S}^n; \hat{\mathcal{S}}^n) && (9.94)\\ &\overset{c}{\leq} \frac{1}{n}(1 + P_e^{(n)} n \log |\mathcal{S}|) + \frac{1}{n} I(\mathcal{X}^n; \mathcal{Y}^n) && (9.95)\\ &\overset{d}{\leq} \frac{1}{n} + P_e^{(n)} \log |\mathcal{S}| + C, && (9.96) \end{aligned}$$

où (a) résulte de la définition et de la monotonicité de l'entropie d'une source stationnaire, (b) est obtenu par l'application de l'inégalité de Fano, (c) est dû à une double application du principe de non-création d'information à la chaîne de Markov $\mathcal{S}^n \leftrightarrow \mathcal{X}^n \leftrightarrow \mathcal{Y}^n \leftrightarrow \hat{\mathcal{S}}^n$, et (d) résulte de la propriété de la capacité du canal sans mémoire étendu (9.64). En faisant $n \to \infty$, $P_e^{(n)} \to 0$ par hypothèse et donc

$$H(\mathcal{S}) \leq C. \tag{9.97}$$

Nous concluons que nous pouvons transmettre une source stationnaire et ergodique sur un canal sans mémoire, si, et seulement si l'entropie de cette source est inférieure à la capacité du canal.□

Nous avons donc réussi à rejoindre les deux bouts : compression et correction d'erreurs. Résumons les quelques points principaux de notre démarche.

- Les théorèmes AEP sont des conséquences de la loi des grand nombres.
- Le théorème de compression de données est basé sur la typicalité des messages de sources : on peut limiter son intérêt à un petit ensemble (en nombre) de messages typiques, de probabilité proche de 1 (sa taille vaut 2^{nH}). On peut donc coder ces messages avec H bits par symbole.
- Le théorème de transmission de données est basé sur l'AEP conjoint; il utilise le fait que pour des longueurs de blocs élevées la séquence de sortie du canal est très probablement conjointement typique avec l'entrée, alors qu'avec tout autre entrée possible il est conjointement typique avec une probabilité d'environ 2^{-nI}. On peut donc utiliser à peu près 2^{nI} mots de codes possibles tout en ayant une probabilité de confusion négligeable.
- Enfin, le théorème du codage source-canal montre que nous pouvons effectivement coder séparément source et canal et combiner le résultat pour obtenir des performances optimales.

9.7 RESUME

Ce chapitre s'est consacré à l'étude des limites de transmission de données sur canal bruité, couronnement des travaux de Shannon.

Nous avons commencé par étudier la notion abstraite de canal discret, vu comme un système entrée/sortie stochastique, c'est-à-dire un système où les sorties sont reliées aux entrées par des distributions de probabilités conditionnelles. Nous avons énuméré un certain nombre de propriétés mathématiques des canaux (causalité, mémoire finie ou nulle, stationnarité, symétrie) et donné un certain nombre d'exemples dont nous avons calculé la capacité en information. Bien que ce modèle couvre en particulier le cas de canaux déterministes, où la relation entrée/sortie est fonctionnelle, nous ne nous sommes pas intéressés à ce cas particulier en tant que tel. Mais les résultats établis dans le cas plus général restent évidemment applicables.

La seconde partie de ce chapitre a été consacrée à l'étude de la relation entre la capacité en information et le débit maximum réalisable (asymptotiquement sans erreurs), c'est-à-dire le nombre de messages distincts pouvant être transmis par utilisation du canal. Cette étude, effectuée pour le cas particulier du canal sans mémoire, nous a conduit à la démonstration du second théorème de Shannon, qui fournit une condition nécessaire et suffisante de communication (sans erreur) sur un canal bruité. Cette démonstration repose essentiellement sur le codage aléatoire, le théorème AEP conjoint, et l'inégalité de Fano.

Nous avons terminé ce chapitre en montrant (codage source-canal) que la décomposition du paradigme de Shannon en blocs normalisés était bien justifiée, puisque rien ne peut être gagné en effectuant un codage conjoint source canal.

Nous avons, à l'occasion, mentionné que ce théorème se généralisait au cas des sources stationnaires et ergodiques, et au cas des canaux causaux stationnaires et à mémoire finie. Voici donc une formulation de la version générale de ce théorème :

1. *Soient un canal stationnaire, causal de capacité en information ergodique C (maximum de l'information mutuelle entrée/sortie par symbole, pour une entrée ergodique) et de mémoire finie m, et une source $\mathcal{S}$ stationnaire et ergodique d'entropie $H(\mathcal{S}) < C$. Soit $\epsilon > 0$.*

2. *Pour* n *assez grand, il est possible de coder une suite* S^n *de* n *symboles émise par la source en une suite de symboles d'entrée du canal de longueur* $n + m$ *(où* m *est la taille de la mémoire), telle que l'observation des suites correspondantes en sa sortie permette avec une probabilité* $1 - \epsilon$ *de déterminer* S^n*.*

3. *Il existe un code tel que l'entropie de la source constituée par le décodeur connecté à la sortie du canal soit arbitrairement proche de* $H(\mathcal{S})$*.*

10 CANAL A BRUIT ADDITIF GAUSSIEN

Nous avons jusqu'ici concentré notre attention sur les modèles purement discrets, à la fois du point de vue du temps et des valeurs possibles émises par les sources et canaux.

Le but de ce chapitre est de donner un aperçu, nettement moins approfondi que dans les chapitres précédents, de ce que la théorie de l'information peut apporter dans le cas continu. Cette généralisation comporte donc deux aspects bien distincts : d'une part, l'utilisation d'un alphabet continu (par exemple un intervalle de nombres réels), d'autre part la considération de signaux (aléatoires ou non) définis pour des valeurs continues du temps. En ce qui concerne le premier aspect nous ferons appel à la notion d'entropie différentielle, déjà introduite au chapitre 4. En ce qui concerne le temps continu, sa considération en toute généralité passerait par le développement de la théorie des processus aléatoires, ce qui sort du cadre de ce cours introductif. Nous nous contenterons donc d'introduire intuitivement la notion de processus aléatoire, puis nous nous intéresserons aux cas particuliers où les signaux en temps continu peuvent être représentés par un nombre fini d'échantillons, ce qui nous ramènera au cas du temps discret.

Nous avons vu à la fin du chapitre 4 que les notions mathématiques d'entropie et d'information mutuelle peuvent, sous certaines conditions, être généralisées au cas de variables aléatoires continues. Qu'en est-il des théorèmes de compression et de transmission de données ?

En ce qui concerne la compression de données, nous avons vu que l'entropie différentielle ne mesure plus la quantité d'information propre. Celle-ci devient en réalité infinie lorsqu'on passe à la limite. Par conséquent, il ne sera plus possible de discuter du codage de source en terme d'information fournie par celle-ci, qui sera toujours infinie. Une théorie appropriée pour étudier le codage de sources continues devra donc tolérer des approximations, ce qui implique la définition de critères de fidélité. L'investigation de ce type de problèmes constitue la *théorie de la distorsion*, qui étudie le codage de sources continues sous des contraintes de distorsion, en d'autres termes la représentation d'une variable continue au moyen d'un nombre fini de mots de code. Cette théorie sera très brièvement abordée dans la troisième partie du cours.

Nous savons d'autre part que la notion d'information mutuelle conserve son sens habituel et les propriétés du cas discret. Par conséquent, en ce qui concerne la transmission d'informations sur un canal continu, nous allons voir qu'il est possible de généraliser le second théorème de Shannon. Le résultat essentiel de cette généralisation est que la capacité du canal limité en puissance est finie dès lors qu'il y a du bruit. Il n'est donc pas possible de transmettre de l'information continue de façon *réversible* même à l'aide d'un canal continu. Le canal continu peut cependant servir à la transmission d'informations discrètes de façon réversible, pour autant que la capacité ne soit pas dépassée. Cela veut dire qu'il est possible de représenter un nombre donné de signaux discrets par un nombre équivalent de signaux continus, de les transmettre à l'aide d'un canal continu, et de décoder les signaux reçus à la sortie avec une probabilité d'erreur arbitrairement petite. Nous étudierons essentiellement un seul modèle de canal, à savoir le canal à bruit additif Gaussien, dont le traitement est à la fois assez direct et qui revêt un intérêt pratique important.

Avant d'entamer l'étude du canal Gaussien nous allons introduire quelques notions de processus aléatoires en temps continu, qui constituent la base pour définir un modèle général de canal continu. Nous énoncerons ensuite le théorème AEP pour les v.a. continues et indiquerons quelques propriétés importantes de la distribution Gaussienne. Nous verrons ensuite quelques modèles de canaux continus, et en particulier le modèle de base du canal Gaussien (sans structure temporelle) ainsi que quelques modèles temporels qui en sont dérivés par application du théorème d'échantillonnage que nous rappelons par ailleurs.

10.1 PROCESSUS ALEATOIRES EN TEMPS CONTINU

10.1.1 Définitions et discussion intuitive

Nous allons très succinctement introduire la notion de processus aléatoire en temps continu, généralisation des notions introduites au chapitre 8 (§8.5). Avant tout insistons sur le fait que notre but ici est de donner une compréhension intuitive de cette notion sans nous attarder sur son étude théorique, par ailleurs assez complexe.

La notion de processus aléatoire est un cas particulier de la notion de fonction aléatoire que nous définissons d'abord.

Fonction aléatoire. Soient un espace probabilisé $(\Omega, \mathcal{E}, P)$, un ensemble $\mathcal{T}$, et un ensemble $\mathcal{F}$. Alors, une fonction aléatoire $\mathcal{X}(\cdot,\cdot)$ est une fonction à deux arguments définie comme suit

$$\begin{aligned} \mathcal{X}(\cdot,\cdot) \quad : \quad & \mathcal{T} \times \Omega \to \mathcal{F} \\ & (t,\omega) \longmapsto \mathcal{X}(t,\omega), \end{aligned} \tag{10.1}$$

où t est une variable variant dans un *ensemble d'indices* $\mathcal{T}$, et ω est le résultat d'une expérience aléatoire. Pour tout t la fonction doit être mesurable.

Notons qu'en principe les ensembles $\mathcal{T}$ et $\mathcal{F}$ peuvent être quelconques. Par exemple, $\mathcal{T}$ peut représenter le temps (continu si $\mathcal{T} = \mathbb{R}$, discret si $\mathcal{T} = \mathbb{N}$), une partie du plan (en traitement d'images), ou le produit cartésien des deux (images video). D'autre part, $\mathcal{F}$ peut lui aussi être quelconque, discret ou continu, mais nous nous intéresserons ici au seul cas où $\mathcal{F} = \mathbb{R}$.

Une fonction aléatoire peut donc être vue comme une famille de variables aléatoires à valeurs dans $\mathcal{F}$ indicées par $t \in \mathcal{T}$, c'est-à-dire

$$\boldsymbol{\mathcal{X}}(\cdot) = \{\mathcal{X}(t,\cdot) \ : t \in \mathcal{T}\}, \tag{10.2}$$

où les $\mathcal{X}(t,\cdot)$ sont des v.a. classiques définies sur $(\Omega, \mathcal{E}, P)$ à valeurs dans $\mathcal{F}$.

Si on fixe la valeur de $\omega \in \Omega$,

$$\mathcal{X}(\cdot,\omega) \tag{10.3}$$

devient simplement une fonction définie sur $\mathcal{T}$ à valeurs dans $\mathcal{F}$. On parle de *réalisation* de la fonction aléatoire, où encore de *trajectoire* du processus.

Pour une valeur de $t \in \mathcal{T}$ nous noterons par $\mathcal{X}(t)$ la variable aléatoire $\mathcal{X}(t,\cdot)$ définie sur Ω. On parle de valeur de la fonction aléatoire en t.

Les fonctions aléatoires interviennent dans bon nombre de domaines. Citons l'apprentissage automatique, qui s'intéresse à l'approximation de familles de fonctions aléatoires, le traitement de signal statistique qui s'intéresse essentiellement à la modélisation de divers types de bruits et au filtrage de celui-ci, la prédiction de séries temporelles, et plus généralement la théorie des systèmes stochastiques.

Remarques. L'intérêt d'étudier les fonctions aléatoires est évidemment de pouvoir modéliser les relations (éventuellement non-déterministes) qui existent entre les valeurs de celles-ci pour différentes valeurs de t.

On voit que lorsque $\mathcal{T}$ est un ensemble discret fini une fonction aléatoire est essentiellement la même chose qu'un vecteur aléatoire. Lorsque $\mathcal{T}$ est infini dénombrable il s'agit d'une suite aléatoire.

Processus aléatoires. On réserve le terme de processus aléatoire (ou stochastique) au cas où $\mathcal{T}$ représente physiquement le temps (p.ex. $\mathcal{T} = \mathbb{R}$). On parle aussi de signal aléatoire, et, lorsque le temps est discret on utilise le terme de série chronologique ou temporelle.

Pour l'étude théorique des processus aléatoires en temps continu et/ou à valeur réelles il est nécessaire de structurer les ensembles de trajectoires $\mathcal{X}(\cdot, \omega)$ auxquelles on s'intéresse et/ou les ensembles de variables aléatoires $\mathcal{X}(t, \cdot)$. Cela se fait en faisant appel à l'analyse fonctionnelle, et notamment aux espaces de Hilbert, qui permettent de généraliser aux espaces de fonctions les notions de distance, de produit scalaire, de projection ... Nous n'en dirons pas plus ici, mais nous suggérons aux étudiants de bien garder en tête l'analogie suivante

fonction de $\mathbb{R}$ dans $\mathbb{R} \approx$ vecteurs de $\mathbb{R}^n$, $n \to \infty$

analogie dans laquelle une valeur du temps correspond à une des dimensions de l'espace $\mathbb{R}^n$.

Nous verrons ci-dessous que dans certains cas particuliers cette analogie tient d'ailleurs au sens strict.

Exemples. Nous reprendrons à titre illustratif les exemples suivants parmi lesquels nous avons repris certains de ceux utilisés par Shannon dans son article fondant la théorie de l'information. Il montrent que dans certains cas un processus aléatoire peut être décrit explicitement par une famille de trajectoires, dans d'autres cas ce n'est pas possible.

1. Un ensemble fini de fonctions $f_i(t)$ de probabilités respectives p_i. C'est typiquement un exemple de ce qui se passerait si on utilisait un canal continu pour "moduler" des messages discrets en nombre fini. Chaque valeur de i correspondrait alors à l'un des messages discret, et $f_i(t)$ serait le signal correspondant envoyé sur le canal.

2. Un ensemble infini (mais de dimension finie) de fonctions définies au moyen d'un vecteur aléatoire de densité de probabilité connue. Par exemple

$$f_{\boldsymbol{w}}(t) = \sum_{i=1}^{p} w_i \sin(2\pi i f t) + w_{p+i} \cos(2\pi i f t)$$

avec disons $\boldsymbol{\mathcal{W}} \sim \mathcal{N}_{2p}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ (les symbole $\sim$ est à interpréter par "...est distribué selon la loi...").

Les deux premiers exemples forment ce qu'on appelle des processus "quasi-déterministes" : une connaissance réduite du passé d'une trajectoire permet d'identifier le signal (la valeur de i, où de $\boldsymbol{w}$) et donc de prédire tout le futur.

3. Soit t_k une suite (disons finie) de valeurs dans $\mathcal{T}$ distribuées selon une certaine loi de probabilités, et $f(\cdot)$ une fonction du temps, alors on peut définir $\forall t$ la variable aléatoire

$$\mathcal{X}(t) = \sum_{k=1}^{n} f(t - t_k), \tag{10.4}$$

ce qui définit bien un processus aléatoire. Ce type de processus permet de modéliser les bruits impulsionnels rencontrés dans un certain nombre d'applications. $f(\cdot)$ est alors une fonction qui réprésente la forme des impulsions, et les t_i représentent les instants d'arrivée des impulsions.

4. Soit $f(\cdot)$ une fonction du temps et $\mathcal{Y}$ une variable aléatoire réelle. Alors, $\forall t$ on peut définir la v.a.

$$\mathcal{X}(t) = f(t) + \mathcal{Y}, \tag{10.5}$$

qui est simplement la v.a. $\mathcal{Y}$ décentrée. Cet exemple fournit une illustration du fait qu'il n'est pas toujours possible de visualiser un processus comme un ensemble probabilisé de fonctions "normales". On peut qualifier la v.a. de "bruit additif blanc" dans la mesure où les valeurs du terme de bruit à des instants différents sont indépendantes. Nous nous servirons de ce type de processus un peu plus loin.

10.1.2 Modélisation probabiliste de processus aléatoires

Usuellement, on décrit un processus aléatoire par une famille de distributions marginales de probabilités, obtenues par projection sur un sous-ensemble fini de l'axe temporel.

Plus précisément, on appelle loi de probabilité fini-dimensionnelle du processus à l'ordre n, la loi du vecteur aléatoire $(\mathcal{X}(t_1), \mathcal{X}(t_2); \ldots, \mathcal{X}(t_n))^T$ donnée pour toute suite d'instants $(t_1, t_2, \ldots, t_n) \in \mathbb{R}^n$.

Ensuite, on appelle loi temporelle d'un processus aléatoire, l'ensemble de toutes les lois fini-dimensionnelles du processus à tout ordre.

Par exemple, on dit qu'un processus aléatoire est Gaussien, si toutes ses lois fini-dimensionnelles sont des lois Gaussiennes. On déduit du fait que toute combinaison linéaire d'une loi Gaussienne est encore Gaussienne, que la transformation d'un processus Gaussien par un filtre (système) linéaire est encore Gaussien.

Un processus Gaussien peut être blanc ou non. Il est blanc si toutes les lois fini-dimensionnelles sont des lois Gaussiennes dont la matrice de variance-covariance est diagonale. Physiquement, cela veut dire que la valeur du processus à un instant donné est indépendante des valeurs à d'autres instants, précédents ou ultérieurs.

10.1.3 Eléments de processus aléatoires

On appelle moyenne du processus aléatoire la fonction

$$m_{\mathcal{X}}(t) \stackrel{\triangle}{=} E\{\mathcal{X}(t)\}. \tag{10.6}$$

Le processus est dit centré si sa moyenne est nulle à tout instant. Si $\mathcal{X}(t)$ est un processus aléatoire non centré on désignera par

$$\mathcal{X}_c(t) \stackrel{\triangle}{=} \mathcal{X}(t) - m_{\mathcal{X}}(t), \tag{10.7}$$

la version centrée de ce processus.

On appelle fonction d'autocovariance (parfois aussi appelée à mauvais escient fonction d'autocorrélation) du processus aléatoire la fonction de deux arguments

$$R_{\mathcal{X}\mathcal{X}}(t_1, t_2) \stackrel{\triangle}{=} E\{\mathcal{X}_c(t_1)\mathcal{X}_c(t_2)\}. \tag{10.8}$$

Cette fonction jouit des propriétés suivantes

1. $R_{\mathcal{X}\mathcal{X}}(t,t) \geq 0$,
2. $R_{\mathcal{X}\mathcal{X}}(t_1, t_2) = R_{\mathcal{X}\mathcal{X}}(t_2, t_1)$,
3. $|R_{\mathcal{X}\mathcal{X}}(t_1, t_2)|^2 \leq R_{\mathcal{X}\mathcal{X}}(t_1, t_1)R_{\mathcal{X}\mathcal{X}}(t_2, t_2)$ (inégalité de Schwarz).

Stationnarité. Un processus aléatoire est dit *stationnaire au sens strict* si la loi temporelle est invariante par translation dans le temps.

On a en particulier dans ce cas les trois propriétés suivantes :

1. la loi de probabilité de $\mathcal{X}(t)$ est indépendante de t.
2. $m_{\mathcal{X}}(t) = m$ est constante,
3. $R_{\mathcal{X}\mathcal{X}}(t_1, t_2)$ ne dépend que de la différence $t_1 - t_2$ (comme en plus elle est symétrique elle ne dépend en fait que du module $|t_1 - t_2|$).

On dit que le processus est *stationnaire au sens large* si

1. $m_{\mathcal{X}}(t) = m$ est constante,
2. $R_{\mathcal{X}\mathcal{X}}(t_1, t_2)$ ne dépend que de la différence $t_1 - t_2 = \tau$,

ce qui est évidemment une condition plus faible que la précédente.

Dans le cas des processus stationnaires (au sens strict ou large) on désigne la fonction d'autocovariance par $R_{\mathcal{X}\mathcal{X}}(\tau)$.

Moments temporels et ergodicité. Dans cette section nous nous intéressons à une sous-classe des processus stationnaires pour lesquels l'information contenue dans une seule trajectoire est suffisante pour les caractériser.

De façon générale, pour une valeur de ω, la réalisation $X(\cdot,\omega)$ est une fonction du temps, éventuellement très bizarre (considérer par exemple le cas du bruit blanc). Nous allons supposer que ces signaux, quel que soit ω, sont de puissance moyenne finie, c'est-à-dire que

$$\frac{1}{2T}\int_{-T}^{T} X(t,\omega)dt \text{ et } \frac{1}{2T}\int_{-T}^{T} X(t+\tau,\omega)X(t,\omega)dt \tag{10.9}$$

sont finies et admettent des limites lorsque $T \to \infty$. Notons que ces quantités dépendent de ω et sont par conséquent des variables aléatoires.

On appelle *moyenne temporelle* la variable aléatoire

$$\overline{\mu}_{\mathcal{X}} \triangleq \lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} X(t,\omega)dt, \tag{10.10}$$

et on appelle *fonction d'autocovariance temporelle* la variable aléatoire

$$\overline{\mu}_{\mathcal{X}\mathcal{X}}(\tau) \triangleq \lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} (X(t+\tau,\omega)-\overline{\mu}_{\mathcal{X}})(X(t,\omega)-\overline{\mu}_{\mathcal{X}})dt. \tag{10.11}$$

Si le processus est ergodique (voir chapitre 8) on montre que

$$\overline{\mu}_{\mathcal{X}} = m_{\mathcal{X}} \text{ et } \overline{\mu}_{\mathcal{X}\mathcal{X}}(\tau) = R_{\mathcal{X}\mathcal{X}}(\tau). \tag{10.12}$$

Un cas particulier de signal ergodique est celui où les variables aléatoires à des instants différents sont i.i.d.

Densité spectrale de puissance. Pour un signal déterministe $x(t)$, on appelle puissance

$$P = \lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} |x(t)|^2 dt. \tag{10.13}$$

Pour un processus ergodique, la quantité apparaissant dans le membre de droite n'est pas aléatoire et est reliée à la fonction d'autocovariance par

$$P = R_{\mathcal{X}\mathcal{X}}(0) + |m_{\mathcal{X}}|^2. \tag{10.14}$$

On étend, par convention, cette formule au cas des processus stationnaires non ergodiques.

En supposant que la fonction d'autocovariance est de carré sommable, nous définissons la densité spectrale de puissance d'un processus stationnaire par

$$S_{\mathcal{X}\mathcal{X}}(f) \triangleq \int_{-\infty}^{+\infty} R_{\mathcal{X}\mathcal{X}}(\tau)e^{-2j\pi f\tau}d\tau, \tag{10.15}$$

c'est-à-dire la transformée de Fourier de $R_{\mathcal{X}\mathcal{X}}(\tau)$. On a donc

$$R_{\mathcal{X}\mathcal{X}}(\tau) \triangleq \int_{-\infty}^{+\infty} S_{\mathcal{X}\mathcal{X}}(f)e^{2j\pi f\tau}df, \tag{10.16}$$

et en particulier

$$R_{\mathcal{X}\mathcal{X}}(0) \triangleq \int_{-\infty}^{+\infty} S_{\mathcal{X}\mathcal{X}}(f)df, \tag{10.17}$$

et par conséquent

$$P = \int_{-\infty}^{+\infty} S_{\mathcal{X}\mathcal{X}}(f)df + |m_{\mathcal{X}}|^2. \tag{10.18}$$

Notons que si le processus est centré ces formules se simplifient évidemment.

Les notions précédentes se traduisent aisément pour le cas du temps discret ($t = \{\ldots, -1, 0, 1, \ldots\}$). Dans ce cas, $R(\tau)$ ne sera définie qu'aux instants discrets et la densité spectrale de puissance devient

$$S_{\mathcal{X}\mathcal{X}}(f) \triangleq \sum_{-\infty}^{+\infty} R_{\mathcal{X}\mathcal{X}}(\tau) e^{-2j\pi f\tau}, \tag{10.19}$$

qui est une fonction continue de f et périodique de période 1. La transformée inverse devient (on peut s'en convaincre)

$$R_{\mathcal{X}\mathcal{X}}(\tau) \triangleq \int_{-\frac{1}{2}}^{+\frac{1}{2}} S_{\mathcal{X}\mathcal{X}}(f) e^{2i\pi f\tau} df. \tag{10.20}$$

On peut interpréter la densité spectrale de puissance de la manière suivante. Soit

$$P_t(f) \triangleq \frac{1}{T}|d_t(f)|^2 \text{ avec } d_t(f) \triangleq \int_0^t X(\tau,\omega) e^{-2i\pi f\tau} d\tau. \tag{10.21}$$

Alors, $\forall t > 0, \forall f$, $P_t(f)$ est une variable aléatoire fonction de t, par définition positive. On peut montrer que

$$\lim_{t\to+\infty} E\{P_t(f)\} = S_{\mathcal{X}\mathcal{X}}(f).$$

Il est important de remarquer que comme la fonction $R_{\mathcal{X}\mathcal{X}}(\tau)$ est paire sa transformée de Fourier est réelle. Nous venons également de montrer que $S_{\mathcal{X}\mathcal{X}}(f) \geq 0, \forall f$ puisque c'est la limite d'une fonction positive. D'autre part, $R_{\mathcal{X}\mathcal{X}}(\tau)$ atteint son maximum absolu en l'origine. En effet, montrons que $\forall\tau \;:\; R_{\mathcal{X}\mathcal{X}}(\tau) \leq R_{\mathcal{X}\mathcal{X}}(0)$. D'après l'inégalité de Schwarz on a

$$|R_{\mathcal{X}\mathcal{X}}(t+\tau,t)|^2 \leq R_{\mathcal{X}\mathcal{X}}(t+\tau,t+\tau) R_{\mathcal{X}\mathcal{X}}(t,t),$$

$\forall t, \tau$. On a donc

$$|R_{\mathcal{X}\mathcal{X}}(\tau)|^2 \leq R_{\mathcal{X}\mathcal{X}}(0) R_{\mathcal{X}\mathcal{X}}(0)$$

ce qui implique (puisque que $R_{\mathcal{X}\mathcal{X}}(0)$ est non négatif)

$$R_{\mathcal{X}\mathcal{X}}(\tau) \leq R_{\mathcal{X}\mathcal{X}}(0).$$

□

10.1.4 Théorème d'échantillonnage

Rappelons que ce théorème dû conjointement à Shannon et Nyquist est formulé de la manière suivante.

Supposons qu'une trajectoire $\mathcal{X}(t)$ d'un processus (aléatoire ou non) soit limitée en fréquence. C'est-à-dire, supposons que la transformée de Fourier de $\mathcal{X}(\cdot)$

$$\mathcal{F}(f) \triangleq \int_{-\infty}^{+\infty} \mathcal{X}(\tau) e^{-2j\pi f\tau} d\tau, \tag{10.22}$$

existe et soit telle que

$$\mathcal{F}(f) = 0, \forall |f| > f_0, \tag{10.23}$$

où f_0 désigne la largeur de bande du signal (en Hz).

Le théorème d'échantillonage dit que la fonction $\mathcal{X}(t)$ est dans ces conditions complètement déterminée par des échantillons

$$\mathcal{X}(\frac{i}{2f_0}), \forall i = -\infty, \ldots, -2, -1, 0, 1, 2, \ldots, +\infty.$$

Nous ne donnons qu'une ébauche de la démonstration : $\mathcal{F}(\cdot)$ peut-être développée en série de Fourier de période fondamentale $[-f_0, \ldots, f_0]$ et les coefficients de ce développement sont entièrement déterminés par les échantillons de $\mathcal{X}(\cdot)$ aux instants sus-mentionnés. Ces coefficients déterminent donc la fonction $\mathcal{F}(\cdot)$ et donc la fonction $\mathcal{X}(\cdot)$ par l'intermédiaire de la transformée de Fourier inverse.

Nous allons montrer que la fonction $\mathcal{X}(\cdot)$ peut en fait être reconstituée directement sans faire appel à cet artifice. Considérons la fonction sinc$(\cdot)$ définie par

$$\mathrm{sinc}(x) \triangleq \frac{\sin(x)}{x}. \tag{10.24}$$

Cette fonction vaut 1 en $t = 0$ et $\mathrm{sinc}(i\pi) = 0, \forall i \neq 0$.

A partir de cette fonction on peut construire une base orthonormée de signaux limités en fréquence dans la bande $[-f_0, \ldots, f_0]$ comme suit :

$$\phi_i(t) = \sqrt{2f_0}\,\mathrm{sinc}\left(2\pi f_0\left(t - \frac{i}{2f_0}\right)\right), \forall i = \ldots -2, -1, 0, 1, 2, \ldots. \tag{10.25}$$

Cette base est orthonormée car

$$\langle \phi_i(\cdot), \phi_j(\cdot)\rangle = \int_{-\infty}^{\infty} \phi_i(t)\phi_j(t)dt = \delta_{i,j}. \tag{10.26}$$

Les transformées de Fourier de ces fonctions sont constantes en module dans la bande $[-f_0, \ldots, f_0]$ et nulles en dehors. D'autre part, la fonction $\phi_i(t)$ vaut $\sqrt{2f_0}$ en $t = \frac{i}{2f_0}$ et est nulle aux instants $t = \frac{j}{2f_0}, \forall j \neq i$. Nous appellerons dans la suite les ϕ_i "signaux de Shannon".

Donc la fonction

$$g(t) = \sum_{n=-\infty}^{n=+\infty} \frac{\mathcal{X}(\frac{n}{2f_0})}{\sqrt{2f_0}} \phi_n(t) \tag{10.27}$$

est évidemment identique à $\mathcal{X}(t)$ aux instants d'échantillonnage. Comme cette fonction est une superposition de fonctions à spectre limité dans une même bande, elle est aussi à spectre limité dans cette bande. Le théorème d'échantillonnage garantit donc qu'elle doit être identique à $\mathcal{X}(t), \forall t \in \mathbb{R}$.

Discussion. En limitant la largeur de bande d'un signal, on réduit le nombre de degrés de liberté à un nombre dénombrable. Pour que le signal puisse être représenté par un vecteur de dimension finie, il faudrait qu'il soit soit périodique, soit de durée limitée. Mais, il est bien connu, que la condition de bande limitée est incompatible (au sens strict) avec la durée limitée disons à $[0, T]$. Un signal continu *échantillonnable* ne peut donc pas être de durée limitée. D'autre part, puisque la réponse impulsionnelle d'un filtre à bande limitée est non-causale, il n'est pas non plus possible de réaliser exactement ce type de filtre en pratique.

Cependant, il est possible de définir la notion de fonction *presque limitée en temps* à $[0, T]$ et *presque limitée en fréquence* à $[0, f_0]$, et il est possible de décrire ces fonctions à l'aide d'une base de fonctions orthogonales contenant à peu près $2Tf_0$ éléments. Il est donc possible de décrire ces fonctions à l'aide de vecteurs de dimension finie $2Tf_0$. Cette discussion suffit pour nous convaincre que les conditions de spectre limité et de durée limitée ne sont pas si incompatibles que ça, comme le suggère par ailleurs le bon sens physique.

D'autre part, à la fin de ce chapitre, nous reviendrons sur l'utilisation de signaux continus pour représenter des suites de symboles, c'est-à-dire le cas très fréquent en pratique où on utilise un canal continu pour transmettre des messages numériques en nombre a priori fini. Nous verrons qu'il est possible de se servir d'espaces structurés de signaux d'entrée de dimension finie à cet effet. En particulier, nous pouvons utiliser un nombre fini d'échantillons associés aux signaux de Shannon, ce qui donnera cependant lieu à des signaux de durée illimitée dans le temps comme nous venons de le souligner.

Retenons pour le moment que certains types de signaux temporels peuvent être représentés au moyen d'un nombre fini de valeurs réelles. Par conséquent, certains types de processus aléatoires peuvent aussi être représentés par des vecteurs aléatoires en dimension finie.

10.2 ENTROPIES DIFFERENTIELLES DE V.A. CONTINUES ET AEP

Nous avons introduit, au chapitre 4, la notion d'entropie différentielle d'une v.a. continue par

$$H_d(\mathcal{X}) \triangleq -\int \log\left[p_{\mathcal{X}}(x)\right] p_{\mathcal{X}}(x)dx, \tag{10.28}$$

écriture qui se généralise directement aux vecteurs aléatoires, auquel cas l'intégrale devient une intégrale multiple.

Nous insistons sur deux hypothèses qui doivent être vérifiées (ce que nous supposerons implicitement être le cas dans la suite) : (i) les distributions de probabilités doivent être continues; (ii) l'intégrale qui définit H_d existe effectivement.

10.2.1 Propriétés de H_d vis-à-vis de transformations linéaires

Considérons une v.a. vectorielle $\boldsymbol{\mathcal{X}}$ et voyons comment son entropie est affectée par des transformations linéaires.

10.2.1.1 Translation. Rappelons que si $\boldsymbol{\mathcal{Y}} = \boldsymbol{a} + \boldsymbol{\mathcal{X}}$ alors $H_d(\boldsymbol{\mathcal{Y}}) = H_d(\boldsymbol{\mathcal{X}})$; l'entropie différentielle est invariante par rapport à une translation.

10.2.1.2 Produit par une matrice. D'autre part, si $\boldsymbol{\mathcal{Y}} = \boldsymbol{A}\boldsymbol{\mathcal{X}}$ (où $\boldsymbol{A}$ est une matrice $n \times n$ non-singulière), on a $H_d(\boldsymbol{\mathcal{Y}}) = \log|\boldsymbol{A}| + H_d(\boldsymbol{\mathcal{X}})$, où $|\boldsymbol{A}|$ désigne la valeur absolue du déterminant de la matrice $\boldsymbol{A}$. Notons que $|\boldsymbol{A}| = \prod_{i=1}^{n} |\lambda_i|$, où λ_i désigne la i-ème valeur propre de $\boldsymbol{A}$.

Il faut donc que la transformation soit non-singulière pour que cette entropie soit finie. Si la transformation tend vers une transformation singulière, l'entropie de $\boldsymbol{\mathcal{Y}}$ tend vers $-\infty$. On peut interpréter cela en disant qu'une transformation singulière concentre le vecteur $\boldsymbol{\mathcal{Y}}$ dans une région infiniment plus petite de l'espace que celle occupée par $\mathcal{X}$. Notons que, puisque $\boldsymbol{\mathcal{Y}}$ est une fonction de $\boldsymbol{\mathcal{X}}$, la quantité d'information $I(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}})$ est infinie, et en fait non-définie.

Ces deux propriétés peuvent être démontrées en effectuant un changement de variables dans la définition de l'entropie différentielle.

Suggestion : supposer que $\boldsymbol{A} = 2\boldsymbol{I}$ et calculer la quantité d'information au moyen des formules $I(\boldsymbol{\mathcal{X}}; \boldsymbol{\mathcal{Y}}) = H_d(\boldsymbol{\mathcal{Y}}) - H_d(\boldsymbol{\mathcal{Y}}|\boldsymbol{\mathcal{X}})$ et $I(\boldsymbol{\mathcal{X}}; \boldsymbol{\mathcal{Y}}) = H_d(\boldsymbol{\mathcal{X}}) - H_d(\boldsymbol{\mathcal{X}}|\boldsymbol{\mathcal{Y}})$, en supposant (naïvement) $H_d(\boldsymbol{\mathcal{Y}}|\boldsymbol{\mathcal{X}}) = 0$ et $H_d(\boldsymbol{\mathcal{X}}|\boldsymbol{\mathcal{Y}}) = 0$ (la connaissance de $\boldsymbol{\mathcal{X}}$ suffit pour connaître $\boldsymbol{\mathcal{Y}}$, et si $\boldsymbol{A}$ est invertible alors l'inverse est également vrai). Montrer qu'on arrive à une contradiction et essayer de justifier en quoi c'est normal. Quelle est la "valeur" de la quantité d'information ?

10.2.2 Entropie différentielle de quelques lois usuelles

Nous allons tout d'abord considérer quelques exemples de v.a. usuelles et indiquer leur entropie différentielle, sans démonstration.

10.2.2.1 Loi uniforme. Soit $\mathcal{X} \sim \mathcal{U}_{[0,a]}$ une v.a. uniforme sur l'intervalle $[0, a]$ alors

$$H_d(\mathcal{X}) = -\int_0^a \frac{1}{a} \log \frac{1}{a} dx = \log a. \tag{10.29}$$

On remarque en particulier que cette entropie est négative si $a < 1$.

Plus généralement, soit $\mathcal{X} \sim \mathcal{U}_S$ une variable aléatoire uniforme sur une région S de $\mathbb{R}^n$ de volume

$$\mathrm{Vol}(S) = \int_{\mathbb{R}^n} 1(x, S) dx,$$

où $1(x, S)$ est la fonction caractéristique de l'ensemble S qui vaut 1 dans cet ensemble et 0 en dehors, alors

$$H_d(\mathcal{X}) = \log \mathrm{Vol}(S). \tag{10.30}$$

On montre que tout comme dans le cas discret, la loi uniforme est la loi qui maximise l'entropie différentielle sous la contrainte $\int_S p(x)dx = 1$.

10.2.2.2 Lois Gaussiennes. Soit $\mathcal{X} \sim \mathcal{N}(\mu, \sigma^2)$ une v.a. Gaussienne dans $\mathbb{R}$, alors

$$H_d(\mathcal{X}) = \frac{1}{2} \log 2\pi e \sigma^2. \tag{10.31}$$

Plus généralement, soit $\boldsymbol{\mathcal{X}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ une v.a. Gaussienne dans $\mathbb{R}^n$, alors

$$H_d(\boldsymbol{\mathcal{X}}) = \frac{1}{2} \log\left((2\pi e)^n |\boldsymbol{\Sigma}|\right) \tag{10.32}$$

où $|\boldsymbol{\Sigma}|$ désigne le déterminant de $\boldsymbol{\Sigma}$ (nécessairement non-négatif).

On montre que la loi Gaussienne est celle qui maximise l'entropie différentielle, sous les contraintes $E\{\boldsymbol{\mathcal{X}}\} = \boldsymbol{\mu}$ et $E\{(\boldsymbol{\mathcal{X}} - \boldsymbol{\mu})(\boldsymbol{\mathcal{X}} - \boldsymbol{\mu})^T\} = \boldsymbol{\Sigma}$.

En particulier, dans le cas unidimensionel on trouve que la loi Gaussienne $\mathcal{N}(\mu, \sigma^2)$ maximise l'entropie sous les contraintes d'égalité $E\{\mathcal{X}\} = \mu$ et $\text{Var}\{\mathcal{X}\} = \sigma^2$, et son entropie vaut $\frac{1}{2}\log(2\pi e\sigma^2)$. Sachant que $E\{\mathcal{X}^2\} = \text{Var}\{\mathcal{X}\} + \mu^2$, on en déduit immédiatement que la loi qui maximise l'entropie sous la seule contrainte d'égalité $E\{\mathcal{X}^2\} = P$ est la loi Gaussienne $\mathcal{N}(0, P)$ d'entropie $\frac{1}{2}\log(2\pi eP)$. De cette dernière propriété et du fait que l'entropie de la loi Gaussienne est une fonction croissante de P, on déduit que la loi qui maximise l'entropie sous la contrainte d'inégalité $E\{\mathcal{X}^2\} \leq P_0$ est la loi Gaussienne $\mathcal{N}(0, P_0)$. Nous ferons appel à ce résultat un peu plus loin.

10.2.3 Théorème AEP pour des v.a. continues

Le théorème AEP reste valable à condition d'effectuer les changements suivants :

- $P \to p$ (remplacement des probabilités par des densités),
- $H \to H_d$ (remplacement de l'entropie par l'entropie différentielle),
- $|\cdot| \to \text{Vol}(\cdot)$ (remplacement des cardinalités par des volumes).

Il se formule donc de la manière suivante :

Soit $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n$ une suite de v.a. i.i.d. selon $p(\cdot)$. Alors

$$-\frac{1}{n} \log p(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n) \xrightarrow{P} E\{-\log p(x)\} = H_d(\mathcal{X}). \tag{10.33}$$

A nouveau, le théorème AEP est une conséquence directe de la loi faible des grands nombres. Le théorème est complété par l'énoncé des propriétés des ensembles typiques, définis comme suit.

Ensembles typiques. Pour $\epsilon > 0$ et $\forall n$ on définit l'ensemble typique $A_\epsilon^{(n)}$ par rapport à la densité $p(\cdot)$ par

$$A_\epsilon^{(n)} \triangleq \left\{(X_1, \ldots, X_n) \in \mathcal{X}^n \ : \left| -\frac{1}{n} \log p(X_1, \ldots, X_n) - H_d(\mathcal{X}) \right| \leq \epsilon \right\}, \tag{10.34}$$

où $p(X_1, \ldots, X_n) = \prod_{i=1}^n p(X_i)$.

L'ensemble typique a les propriétés fondamentales suivantes :

1. $P(A_\epsilon^{(n)}) > 1 - \epsilon$, pour n suffisamment grand.
2. $\text{Vol}(A_\epsilon^{(n)}) \leq 2^{n(H_d(\mathcal{X})+\epsilon)}$, pour tout n.
3. $\text{Vol}(A_\epsilon^{(n)}) \geq (1 - \epsilon)2^{n(H_d(\mathcal{X})-\epsilon)}$, pour n suffisamment grand.

De plus, on montre que le volume minimal de tout sous-ensemble de $\mathcal{X}^n$ de probabilité supérieure à $1 - \epsilon$ est essentiellement le volume de l'ensemble typique.

Ce théorème montre donc que dans $\mathbb{R}^n$, le plus petit volume qui contient essentiellement toute la probabilité est à peu près 2^{nH_d}, ce qui correspond à un hyper-cube de largeur 2^{H_d}. On peut donc dire que l'entropie différentielle est le logarithme de la largeur moyenne du plus petit ensemble qui contient à peu près toute la probabilité. Donc, une entropie différentielle faible implique que la v.a. est confinée dans un petit volume, et réciproquement, une entropie différentielle élevée implique que la v.a. est étalée sur un grand volume.

Illustrations.

1. Supposons que $\mathcal{X} \sim \mathcal{U}_{[0,a]}$. On a $H_d = \log a$, et donc $2^{H_d} = a$ et $\mathrm{Vol}(A_\epsilon^{(n)}) = 2^{nH_d} = a^n$, ce qui est bien conforme au fait qu'ici toute la probabilité est confinée dans un hyper-cube de largeur de côté a.

2. Supposons que $\mathcal{X} \sim \mathcal{N}(\mu, \sigma^2)$. Les séquences typiques ont alors une distribution sphérique centrée autour du point $(\mu, \mu, \ldots, \mu)$. Le volume de la sphère typique sera de $2^{n\frac{1}{2}\log 2\pi e\sigma^2}$. On montre que le rayon équivalent d'une sphère pleine ayant ce volume croît lentement mais indéfiniment lorsque $n \to \infty$.

AEP conjoint. On peut formuler et démontrer également le théorème AEP conjoint dans le cas de deux v.a. continues $\mathcal{X}$ et $\mathcal{Y}$, avec les mêmes modifications par rapport au cas discret que ci-dessus.

10.3 CANAUX CONTINUS

10.3.1 Le canal Gaussien

Le canal Gaussien est un canal ayant un alphabet continu en entrée et en sortie mais discret en temps. Sa relation entrée-sortie est définie de la manière suivante :

$$\mathcal{Y}_i = \mathcal{X}_i + \mathcal{Z}_i,\ \mathcal{Z}_i \sim \mathcal{N}(0, N) \tag{10.35}$$

où $\mathcal{X}$ désigne l'entrée et $\mathcal{Y}$ la sortie. La sortie est donc égale à l'entrée à un terme de bruit additif Gaussien près. Le symbole N (de l'anglais *noise*) désigne la variance du bruit. Les variables $\mathcal{Z}_i$ sont indépendantes des $\mathcal{X}_i$ et mutuellement indépendantes, en d'autres termes le bruit est blanc en plus d'être Gaussien.

Lorsque $N = 0$, le récepteur reçoit exactement le signal transmis. Puisque $\mathcal{X}$ peut prendre n'importe quelle valeur réelle, le canal sans bruit peut transmettre un nombre réel quelconque sans erreur; la capacité est donc illimitée.

Lorsque $N > 0$, mais qu'aucune contrainte n'existe sur les signaux d'entrée, il est possible de choisir une infinité de valeurs d'entrée espacées arbitrairement loin les unes des autres, de façon à ce qu'elles restent distinguables à la sortie malgré le bruit. La capacité est donc encore illimitée.

Donc, si soit la variance du bruit est nulle, soit l'entrée est non-contrainte, la capacité du canal Gaussien est infinie.

La limitation la plus commune imposée en pratique sur les entrées est une limitation de puissance (ou d'énergie). Nous allons supposer que la puissance moyenne (par symbole transmis) est limitée. Nous supposons donc que toute suite de symboles $(X_1, \ldots, X_n)$ transmise sur le canal satisfait à l'inégalité suivante

$$\frac{1}{n}\sum_{i=1}^{n} X_i^2 \leq P. \tag{10.36}$$

Si les signaux d'entrée sont ergodiques (ce que nous supposerons être le cas dans ce qui suit), cette contrainte devient lorsque $n \to \infty$ équivalente à la contrainte

$$E\{\mathcal{X}^2\} \leq P. \tag{10.37}$$

Ce type de canal modélise un grand nombre de canaux pratiques, incluant les liaisons par faisceau hertzien et les canaux satellite. La bruit additif dans ce type de canal peut être provoqué par différents phénomènes. Mais, le théorème de la limite centrale nous apprend que l'effet cumulatif d'un grand nombre de causes indépendantes donne lieu à une distribution Gaussienne, ce qui explique que l'hypothèse Gaussienne soit valable dans un grand nombre de situations réelles.

Analysons tout d'abord une première façon d'exploiter notre canal Gaussien. Elle est simple, mais sous-optimale. Supposons que nous voulions envoyer un symbole binaire sur le canal par utilisation du canal. Etant donné la limitation en puissance, le mieux que nous puissions faire est d'envoyer un des deux niveaux $+\sqrt{P}$ ou $-\sqrt{P}$. Le récepteur voit le résultat à la sortie, Y et essaye de décider lequel des deux symboles a été envoyé. En supposant que les deux niveaux soient équiprobables (ce qui serait le cas si nous voulions envoyer 1 bit d'information), la règle de décodage optimale est de décider que $+\sqrt{P}$ fut envoyé si $Y > 0$ et que $-\sqrt{P}$ fut envoyé si $Y < 0$. (On peut négliger ce qui se passe si $Y = 0$, puisque la probabilité de cet événement est nulle.) La probabilité d'erreur de ce schéma de communication est alors

$$\begin{aligned} P_e &= \frac{1}{2}P(Y < 0|X = +\sqrt{P}) + \frac{1}{2}P(Y > 0|X = -\sqrt{P}) && (10.38)\\ &= \frac{1}{2}P(Z < -\sqrt{P}|X = +\sqrt{P}) + \frac{1}{2}P(Z > +\sqrt{P}|X = -\sqrt{P}) && (10.39)\\ &= P(Z > \sqrt{P}) && (10.40)\\ &= 1 - \Phi\left(\sqrt{\frac{P}{N}}\right), && (10.41)\end{aligned}$$

où $\Phi(\cdot)$ est fonction de répartition de la loi Gaussienne

$$\Phi(x) \triangleq \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt. \tag{10.42}$$

En utilisant ce schéma, nous avons converti notre canal Gaussien en un canal binaire symétrique avec une probabilité de confusion P_e. De même en utilisant une quantification en 4 niveaux, nous pouvons convertir le canal Gaussien en canal discret avec $q = 4$. L'avantage principal de cette idée, par ailleurs utilisée dans un certain nombre de schémas de modulation pratiques, est qu'on peut ensuite utiliser les techniques de codage valables pour les canaux discrets pour réduire les erreurs de transmission. Cependant, comme nous allons le voir la quantification du signal conduit à une perte de capacité.

10.3.1.1 Définitions. Les définitions suivantes sont analogues à celles introduites au chapitre précédent dans le cas des canaux discrets.

Capacité en information. La *capacité en information* du canal (Gaussien) limité en puissance est définie par

$$C \triangleq \max_{p(x):E\{\mathcal{X}^2\}\leq P} I(\mathcal{X};\mathcal{Y}). \tag{10.43}$$

Nous pouvons effectuer le calcul de C de la manière suivante :

$$\begin{aligned} I(\mathcal{X};\mathcal{Y}) &= H_d(\mathcal{Y}) - H_d(\mathcal{Y}|\mathcal{X}) && (10.44)\\ &= H_d(\mathcal{Y}) - H_d(\mathcal{X}+\mathcal{Z}|\mathcal{X}) && (10.45)\\ &= H_d(\mathcal{Y}) - H_d(\mathcal{Z}|\mathcal{X}) && (10.46)\\ &= H_d(\mathcal{Y}) - H_d(\mathcal{Z}), && (10.47)\end{aligned}$$

puisque $\mathcal{Z}$ est indépendante de $\mathcal{X}$. Or $H_d(\mathcal{Z}) = \frac{1}{2}\log 2\pi eN$, et

$$E\{\mathcal{Y}^2\} = E\{\mathcal{X}^2 + \mathcal{Z}^2\} = E\{\mathcal{X}^2\} + 2E\{\mathcal{X}\mathcal{Z}\} + E\{\mathcal{Z}^2\} \leq P + N \tag{10.48}$$

car, comme $\mathcal{X}$ et $\mathcal{Z}$ sont indépendantes et que $E\{\mathcal{Z}\} = 0$, on a $E\{\mathcal{X}\mathcal{Z}\} = E\{\mathcal{X}\}E\{\mathcal{Z}\}$.

Or, nous savons que la distribution de $\mathcal{Y}$ qui maximise l'entropie différentielle $H_d(\mathcal{Y})$ sous la contrainte

$$E\{\mathcal{Y}^2\} \leq P + N \tag{10.49}$$

est la distribution Gaussienne de moyenne nulle et de variance $P + N$. On aura donc,

$$I(\mathcal{X};\mathcal{Y}) \leq \frac{1}{2}\log 2\pi e(P+N) - \frac{1}{2}\log 2\pi eN = \frac{1}{2}\log(1+\frac{P}{N}). \tag{10.50}$$

Si maintenant $\mathcal{X} \sim \mathcal{N}(0,P)$ on a bien $E\{\mathcal{X}^2\} = P$, $\mathcal{Y} \sim \mathcal{N}(0, N+P)$ et

$$I(\mathcal{X};\mathcal{Y}) = \frac{1}{2}\log(1+\frac{P}{N}). \tag{10.51}$$

La capacité en information vaut donc bien

$$C = \frac{1}{2}\log(1+\frac{P}{N}), \tag{10.52}$$

et on voit le rôle joué par le rapport signal bruit à la sortie $\log(1+\frac{P}{N})$.

Lorsque $N \to 0$ ou lorsque $P \to +\infty$ la capacité en information tend vers l'infini, ce qui correspond aux deux cas évoqués au début de cette section. Dans tous les autres cas, la capacité en information est finie. Nous verrons ci-dessous que cette notion coïncide bien avec la notion de capacité opérationnelle, et que par conséquent il n'est pas possible de transmettre des symboles d'un alphabet continu sans perte d'information.

Code (M, n). Un code (M, n) pour le canal Gaussien avec limitation de puissance P consiste en :

1. Un ensemble d'indices $\{1, 2, \ldots, M\}$.
2. Une fonction d'encodage $x^n(\cdot) \ : \{1, 2, \ldots, M\} \to \mathcal{X}^n$, produisant les mots de codes $x^n(1), \ldots, x^n(M)$ qui vérifient la contrainte de puissance moyenne pour chaque mot de code, i.e.

$$\sum_{i=1}^{n} x_i^2(w) \leq nP, \ \forall w \in \{1, 2, \ldots, M\}. \tag{10.53}$$

3. Une fonction de décodage

$$g(\cdot) \ : \mathcal{Y}^n \to \{1, 2, \ldots, M\}. \tag{10.54}$$

Le débit et la probabilité d'erreur du code sont définis comme dans le cas de canaux discrets (§9.2.1).

Débit réalisable. Un débit R est dit réalisable pour le canal Gaussien avec limitation de puissance P s'il existe une suite de codes $(\lceil 2^{nR} \rceil, n)$ (respectant la limitation de puissance) telle que la probabilité d'erreur maximale $\lambda^{(n)}$ tende vers zéro.

Capacité opérationnelle. La capacité opérationnelle est la borne supérieure des débits atteignables.

10.3.1.2 Capacité du canal Gaussien.

Théorème. La capacité opérationnelle du canal Gaussien avec limitation de puissance à P et variance de bruit N vaut

$$C = \frac{1}{2}\log(1+\frac{P}{N}) \text{ bits par utilisation du canal.} \tag{10.55}$$

Démonstration. Nous ne donnerons pas la démonstration détaillée de ce théorème, mais nous indiquerons les principales idées sous-jacentes, en commençant par un argument intuitif de plausibilité basé sur l'interprétation géométrique dans l'espace des signaux.

Empilement de sphères dans $\mathbb{R}^n$. Raisonnons géométriquement comme au chapitre précédent pour montrer qu'il est plausible de distinguer les mots d'un code $(\lceil 2^{nC} \rceil, n)$ à la sortie du canal.

Plaçons nous dans la situation qui maximise la capacité en information. Considérons qu'un mot de code à l'entrée est un vecteur donné de $\mathbb{R}^n$, et le mot reçu est alors un vecteur aléatoire avec une distribution Gaussienne centrée sur le vecteur d'entrée (la matrice de covariance est diagonale, de variance $N\boldsymbol{I}$, ou $\boldsymbol{I}$ représente la matrice identité).

A cause de la loi des grands nombres, $\frac{1}{n}\sum_{i=1}^{n} z_i^2 \xrightarrow{P} N$, et le signal reçu est donc confiné avec une probabilité proche de 1 (pour n suffisamment grand) dans une sphère de rayon $\sqrt{n(N+\epsilon)}$. Remarquons aussi qu'on assiste en même temps à un phénomène étrange appelé *durcissement des sphères* : puisque $\frac{1}{n}\sum_{i=1}^{n} z_i^2 \xrightarrow{P} N$, le signal tend à se concentrer de plus en plus à la surface de la sphère.

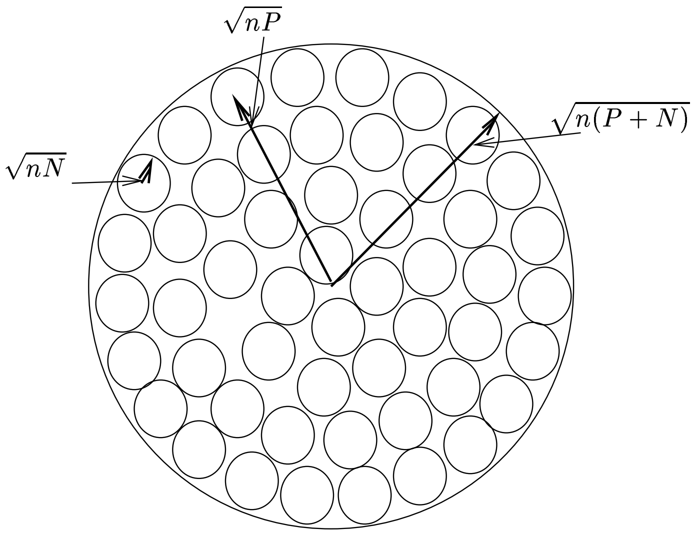


Figure 10.1. Empilement des sphères pour le canal Gaussien

Donc, si nous nous arrangeons pour placer les vecteurs d'entrée de manière à ce que ces sphères de sortie ne se recouvrent pas, nous pouvons décoder le signal reçu en affectant tout ce qui tombe dans une des sphères au mot d'entrée correspondant, ce qui donnera lieu à une probabilité d'erreur aussi faible que souhaité, en faisant tendre n vers l'infini.

L'évaluation de la capacité se ramène donc à la détermination du nombre de sphères que nous pourrons empiler sans qu'il y ait recouvrement. Puisque les signaux émis sont limités en puissance moyenne ils doivent se trouver dans une sphère de rayon $\sqrt{nP}$; puisque les signaux reçus sont le résultat de la superposition de signaux mis à l'entrée et d'un terme de bruit indépendant, leur puissance moyenne sera limitée à $(P + N)$, car

$$\sum_{i=1}^{n} y_i^2 = \sum_{i=1}^{n} (x_i + z_i)^2 = \sum_{i=1}^{n} x_i^2 + 2 \sum_{i=1}^{n} x_i z_i + \sum_{i=1}^{n} z_i^2, \tag{10.56}$$

où le terme central $\sum x_i z_i$ tend vers 0 (loi des grand nombres), le terme de gauche est majoré par nP et le terme de droite tend vers nN.

Donc les vecteurs reçus seront confinés dans une sphère de rayon $\sqrt{n(P+N)}$. Mais le volume d'une sphère de rayon r dans un espace de dimension n, peut s'écrire sous la forme

$$\text{Vol} = A_n r^n, \tag{10.57}$$

où A_n ne dépend pas du rayon. Donc le rapport du volume de la grande sphère (réception) à celui des petites sphères est minoré par

$$\frac{(n(P+N))^{\frac{n}{2}}}{(n(N))^{\frac{n}{2}}} = 2^{\frac{n}{2}\log(1+\frac{P}{N})}, \tag{10.58}$$

et ce nombre fixe donc une borne supérieure au nombre de mots de code qu'on peut envoyer sur le canal sans erreur. La situation est schématisée à la figure 10.1.

Cet argument géométrique, qui nous laisse un peu perplexe, montre qu'on ne peut pas espérer dépasser la capacité en information. Nous allons esquisser ci-dessous une démonstration plus rigoureuse qui montre que la capacité en information correspond à un débit atteignable à ϵ près, et que de plus il n'est possible de la dépasser qu'au prix d'une probabilité d'erreur de décodage non-nulle. Cela nous permettra entre autres d'affiner notre vision géométrique des choses.

Esquisse de la démonstration. On utilise les mêmes idées que dans le cas du canal discret, à savoir, codage aléatoire et AEP conjoint. Les modifications sont liées à la prise en compte de la limitation de la puissance moyenne.

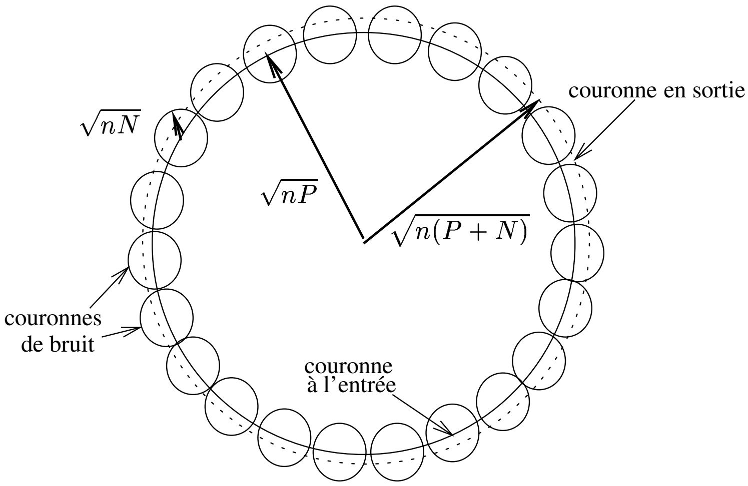


Figure 10.2. Empilement des sphères pour le canal Gaussien

Esquissons d'abord la preuve que la capacité en information est bien atteignable.

1. *Table de code.* Nous voulons construire un code qui respecte (au moins asymptotiquement) la limitation de puissance. Pour cela, nous générons M mots de code de longueur n en tirant $M \times n$ nombres aléatoires indépendants, distribués en loi normale $\mathcal{N}(0, P - \epsilon)$. La loi des grands nombres permet alors d'affirmer que pour n suffisamment grand et pour tout ϵ ces mots de codes finiront par respecter la contrainte de puissance. Nous n'en demandons pas plus.

   Cela étant, nous remarquons que cette façon de faire consiste à placer les mots de code dans un voisinage (d'autant plus fin que n est grand) de la surface de la sphère de rayon $\sqrt{n(P - \epsilon)}$. Les mots de code n'occupent donc apparemment pas tout l'espace. La situation géométrique est représentée à la figure 10.2.

   Pour une valeur finie de n on conserve tous les mots de code, même ceux qui ne respecteraient pas la contrainte (pour ces derniers on comptabilisera d'office une erreur de décodage).

2. Une fois le code construit il est révélé à l'émetteur et au récepteur. L'émetteur l'utilise pour encoder ses messages, le récepteur exploite le théorème AEP conjoint pour le décodage : il identifie le signal d'entrée qui est conjointement typique avec le mot reçu à la sortie. Si ce signal ne respecte pas la contrainte de puissance, ou bien s'il est ambigu, alors le récepteur déclare une erreur d'emblée, sinon il déclare (éventuellement erronément) qu'il a reçu le signal conjointement typique.

3. La suite de la démonstration consiste en une majoration de la probabilité d'erreur moyenne de tous les codes aléatoires (on y exploite des arguments de symétrie résultant du tirage aléatoire des mots de code).

4. Ayant montré qu'en moyenne ces codes sont bons du point de vue de la probabilité d'erreur moyenne, on est en position de dire qu'il existe au moins un code dont la probabilité d'erreur moyenne est faible. On peut alors prendre ce code, et filtrer la moitié la plus mauvaise des mots de ce code, et on contruit ainsi un code dont la probabilité d'erreur *maximale* est aussi faible que souhaité, ce qui termine la démonstration. □

L'esquisse de la démonstration de la réciproque est la suivante. Il faut montrer que les débits supérieurs à $\frac{1}{2}\log(1 + \frac{P}{N})$ ne sont pas atteignables. A nouveau la démonstration est très proche de celle du canal discret, et se sert de l'inégalité de Fano. On montre que pour un débit donné R et une distribution uniforme (équiprobable) des 2^{nR} séquences d'entrée, on a l'inégalité suivante

$$nR \leq \sum_{i=1}^{n} I(X_i; Y_i) + n\epsilon'_n + 1, \tag{10.59}$$

où $\epsilon'_n + \frac{1}{n} = \epsilon_n \to 0$, lorsque $P_e^{(n)} \to 0$. Or $I(X_i;Y_i) = H_d(Y_i) - H_d(Z_i)$. D'autre part, $H_d(Z_i) = \frac{1}{2}\log 2\pi e(N)$. Enfin, si nous désignons par

$$P_i = \frac{1}{2^{nR}} \sum_{j=1}^{2^{nR}} x_{ji}^2$$

la valeur moyenne des valeurs x_{ji}^2 correspondant à la i-ème colonne de la table de code (correspondant au i-ème symbole reçu) la puissance moyenne de Y_i vaut $P_i + N$. Son entropie ne peut donc pas dépasser celle de la distribution gaussienne de variance $P_i + N$, soit $\frac{1}{2}\log 2\pi e(P_i + N)$.

Par conséquent,

$$nR \leq \sum_{i=1}^{n} \frac{1}{2}\log\left(1 + \frac{P_i}{N}\right) + n\epsilon_n. \tag{10.60}$$

Puisque chaque mot de code satisfait à la contrainte de puissance on a

$$\forall j\ ,\ \sum_{i=1}^{n} x_{ji}^2 \leq nP, \tag{10.61}$$

ce qui implique[1]

$$\Rightarrow \sum_{j=1}^{2^{nR}} \sum_{i=1}^{n} x_{ji}^2 \leq n2^{nR}P. \tag{10.62}$$

Par conséquent, en intervertissant l'ordre de sommation et en divisant par $n2^{nR}$ on trouve que

$$\frac{1}{n}\sum_{i=1}^{n} P_i \leq P.$$

Comme la fonction $\log(1+x)$ est concave, on a (inégalité de Jensen) que

$$\frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\log\left(1 + \frac{P_i}{N}\right) \leq \frac{1}{2}\log\left(1 + \frac{1}{n}\sum_{i=1}^{n}\frac{P_i}{N}\right), \tag{10.63}$$

et le dernier terme est majoré par $\frac{1}{2}\log\left(1 + \frac{P}{N}\right)$ puisque la fonction $\log(1+x)$ est croissante.

Nous avons donc en fin de compte que

$$R \leq \frac{1}{2}\log\left(1 + \frac{P}{N}\right) + \epsilon_n, \tag{10.64}$$

ce qui implique que $R \leq C$.□

10.3.2 Canaux à bande passante limitée

Un modèle de canal très commun en télécommunications par faisceau hertzien ou ligne téléphonique est le canal à bande passante limitée et bruit blanc. Il s'agit d'un canal qui fonctionne en temps continu, dont la sortie peut être décrite par

$$Y(t) = (X(t) + Z(t)) * h(t), \tag{10.65}$$

où $X(t)$ est le signal temporel d'entrée, $Z(t)$ une réalisation de bruit blanc Gaussien et $h(t)$ est la réponse impulsionnelle d'un filtre passe-bas idéal qui coupe toute les composantes fréquentielles au-delà d'un seuil f_0 (le symbole $*$ désigne le produit de convolution). Remarquons tout d'abord que comme le canal filtre parfaitement toutes les fréquences en dehors de l'intervalle $[-f_0, +f_0]$ on peut se contenter d'analyser ce qui se passe avec des signaux d'entrée et de sortie limités en fréquence.

[1]Ce détour est nécessaire car la seule chose que nous savons sur le code, au niveau de la démonstration de la réciproque, c'est qu'il doit respecter la contrainte de puissance pour chaque mot.

Notons que la réponse impulsionnelle d'un filtre à bande limitée idéal est précisément la fonction sinc(t) définie plus haut. Notons également qu'on a évidemment

$$Y(t) = X(t) * h(t) + Z(t) * h(t), \tag{10.66}$$

et que, suite au théorème d'échantillonnage les signaux $X(t) * h(t)$ et $Z(t) * h(t)$ et $Y(t)$ sont déterminés par leurs échantillons aux instants $t_i = \frac{i}{2f_0}$.

Bruit blanc et spectre limité. Spécifions tout d'abord ce que nous entendons ici par bruit blanc Gaussien en temps continu. Un bruit blanc $\mathcal{X}(\omega, t)$ est un processus aléatoire stationnaire tel que $E\{\mathcal{X}(t)\} = 0, \forall t$, et tel que

$$R_{\mathcal{X}\mathcal{X}}(\tau) \stackrel{\triangle}{=} E\{\mathcal{X}(t)\mathcal{X}(t-\tau)\} = \frac{N_0}{2}\delta(\tau), \tag{10.67}$$

où $\delta(\cdot)$ représente l'impulsion de Dirac (idéalisation mathématique) et $\frac{N_0}{2}$ la variance du bruit. Le bruit blanc est dit Gaussien si de plus, les $\mathcal{X}(t)$ suivent une loi Gaussienne $\forall t$. Ce type de signal lorsqu'il est filtré par un filtre idéal donne lieu à ce qu'on appelle un bruit blanc Gaussien à bande limitée.

Une autre façon de procéder pour définir un bruit blanc Gaussien est la suivante : supposons que nous utilisions les signaux de Shannon (définis page 185, formule (10.25)) comme base pour représenter tous nos signaux. Supposons, pour commencer, que nous utilisions une base de largeur de fréquence très grande, de façon à pouvoir représenter tous les signaux physiquement réalisables. Dans ce cas, nous pouvons écrire n'importe quel signal $X(t)$ comme une superposition de fonctions

$$X_i \mathrm{sinc}(2\pi f_0(t - i\Delta t))$$

avec $\Delta t = \frac{1}{2f_0}$, qui peut être rendu aussi petit que voulu en prenant f_0 suffisamment grand.

En faisant tendre f_0 vers l'infini, Δt tend vers zéro, et on est ainsi capable de représenter tout type de signal. Supposons maintenant que, pour une valeur donnée de Δt, les valeurs de X_i soient i.i.d. selon une loi Gaussienne de variance $N_0 f_0$. Nous pouvons calculer

$$R_{\mathcal{X}\mathcal{X}}(\tau) = E\{\mathcal{X}(t)\mathcal{X}(t-\tau)\}, \tag{10.68}$$

où $\tau \in \mathbb{R}$. Puisque que le processus est stationnaire nous pouvons calculer cette corrélation en postulant que $t = 0$. On obtient

$$R_{\mathcal{X}\mathcal{X}}(\tau) = E\{\mathcal{X}(0)\mathcal{X}(-\tau)\}, \tag{10.69}$$

en remplaçant $\mathcal{X}(0)$ par $\mathcal{X}_0$ et $\mathcal{X}(-\tau)$ par $\sum_i \mathcal{X}_i \mathrm{sinc}(-2\pi f_0(i\Delta t + \tau))$ on obtient

$$R_{\mathcal{X}\mathcal{X}}(\tau) = E\{\sum_i \mathcal{X}_0\mathcal{X}_i \mathrm{sinc}(-2\pi f_0(i\Delta t + \tau))\}. \tag{10.70}$$

En intervertissant les opérations de sommation et de produit avec l'espérance mathématique on a

$$R_{\mathcal{X}\mathcal{X}}(\tau) = \sum_i E\{\mathcal{X}_0\mathcal{X}_i\}\mathrm{sinc}(-2\pi f_0(i\Delta t + \tau)), \tag{10.71}$$

et comme les $\mathcal{X}_i, \forall i \neq 0$ sont indépendantes de $\mathcal{X}_0$ et de moyennes nulles, on a finalement

$$R_{\mathcal{X}\mathcal{X}}(\tau) = E\{\mathcal{X}_0^2\}\mathrm{sinc}(-2\pi f_0\tau) = N_0 f_0 \mathrm{sinc}(2\pi f_0\tau). \tag{10.72}$$

On constate donc que la corrélation n'est nulle qu'aux instants d'échantillonnage. On a par ailleurs

$$\int_{-\infty}^{+\infty} \mathrm{sinc}(2\pi f_0\tau)d\tau = \frac{1}{2f_0},$$

ce qui implique que

$$\int_{-\infty}^{+\infty} R_{\mathcal{X}\mathcal{X}}(\tau) = \frac{N_0}{2},$$

quelle que soit la valeur de f_0.

D'autre part, pour tout $\tau \neq 0$, on a

$$\lim_{f_0 \to \infty} \mathrm{sinc}(2\pi f_0 \tau) = 0, \tag{10.73}$$

et on peut donc accepter l'idée que

$$\lim_{f_0 \to \infty} R_{\mathcal{X}\mathcal{X}}(\tau) = \frac{N_0}{2}\delta(\tau).$$

On peut donc caractériser un bruit aléatoire Gaussien limité en fréquence en disant qu'il s'agit d'un processus aléatoire Gaussien de moyenne nulle et dont la matrice de variance-covariance se déduit directement de la fonction $R_{\mathcal{X}\mathcal{X}}(\tau)$. En particulier, pour une suite de k instants périodiques de période $\Delta t = \frac{1}{2f_0}$ cette matrice est la matrice $N_0 f_0 \boldsymbol{I}$ où $\boldsymbol{I}$ désigne la matrice identité d'ordre k. Notons, pour terminer, que l'énergie par unité de temps (puissance) de ce type de signal croît évidemment linéairement avec la largeur de bande du signal f_0.

Capacité. Vu ce qui précède, pour un canal en temps continu limité en fréquence, nous pouvons représenter aussi bien les entrées que les sorties par leurs échantillons pris à des instants multiples de $\frac{1}{2f_0}$ (la partie du signal d'entrée qui n'est pas dans la bande de fréquence du canal étant filtrée de toutes façons, elle n'influence donc pas la sortie). On vient de voir que, dans ces conditions, les valeurs du bruit à ces instants sont indépendantes. Ajoutons que notre canal est invariant dans le temps, et donc notre raisonnement ne dépend absolument pas du choix de l'origine du temps.

Si le bruit est de densité spectrale égale à $\frac{N_0}{2}$, la puissance totale du bruit dans la bande de fréquence utile est $N_0 f_0$. Supposons que nous utilisions une fenêtre temporelle de durée T pour transmettre notre signal perturbé par le bruit. L'énergie moyenne du bruit sur cette durée est alors $N_0 f_0 T$, répartie sur $2f_0 T$ échantillons, ce qui donne une contribution de $\frac{N_0}{2}$ pour chaque échantillon. Il faut remarquer que cette contribution ne dépend pas de la largeur de bande du signal.

Pour le signal, que nous supposons limité en énergie à PT indépendamment de la largeur de la bande de fréquence, on obtient, en faisant le même raisonnement, une contribution de $\frac{P}{2f_0}$ par échantillon.

Nous pouvons utiliser un schéma aléatoire pour générer nos signaux en tirant au hasard les valeurs des échantillons avec une loi Gaussienne de variance P. Si nous regardons maintenant un échantillon à la sortie du canal, celui-ci est composé d'un terme relatif à la source de variance P et d'un terme de bruit additif Gaussien de variance $N_0 f_0$, ce qui nous ramène au cas discuté dans la section précédente. On a donc

$$C = \frac{1}{2}\log\left(1 + \frac{P}{N_0 f_0}\right) \text{ bits par échantillon,} \tag{10.74}$$

et comme il y a $2f_0$ échantillons par seconde, on obtient

$$C = f_0 \log\left(1 + \frac{P}{N_0 f_0}\right) \text{ bits par seconde.} \tag{10.75}$$

On voit que le fait d'augmenter la largeur de bande conduit à une diminution logarithmique de la capacité du canal par symbole. Mais, comme le fait d'augmenter la bande de fréquence permet d'augmenter de façon linéaire le nombre de symboles modulables, le rendement de l'opération est positif. La limite de la capacité, pour une largeur de bande infinie, vaut

$$\lim_{f_0 \to \infty} f_0 \log\left(1 + \frac{P}{N_0 f_0}\right) = \frac{P}{N_0}\log e \text{ bits par seconde.} \tag{10.76}$$

Donc, pour une largeur de bande infinie, la capacité croît linéairement avec la puissance d'émission.

Exemple.

La largeur de bande d'une ligne téléphonique est limitée à 3300Hz (ce qui convient parfaitement à la parole). En supposant que le rapport signal bruit soit de 20dB ($= 10\log_{10}\frac{P}{N_0 f_0}$) on a

$$\frac{P}{N_0 f_0} = 100,$$

et on calcule que $C = 21.972$ bits par seconde.

10.4 CANAUX PARALLELES ET BRUIT COLORE

Nous allons brièvement montrer comment le résultat précédent peut être utilisé pour traiter des situations dérivées.

10.4.1 Canaux Gaussiens parallèles

Dans certaines situations on dispose de plusieurs canaux Gaussiens qui peuvent être utilisés en parallèle pour transmettre de l'information. Il est intéressant de calculer la capacité de ce type de système sous une contrainte commune limitant la puissance totale. On a alors la situation suivante : pour chacun des k canaux $Y_i(t) = X_i(t) + Z_i(t)$ avec $\mathcal{Z}_i(t) \sim \mathcal{N}(0, N_i)$ (les $\mathcal{Z}_i$ étant indépendants) sous la contrainte

$$E\{\sum_{i=1}^{k} X_i^2\} \leq P. \tag{10.77}$$

Nous allons simplement (sans parler de la capacité opérationnelle) calculer la capacité en information de ce type de système, c'est-à-dire

$$C = \max_{p(X_1,\ldots,X_k)\,:E\{\sum X_i^2\}\leq P} I(\mathcal{X}_1,\ldots,\mathcal{X}_k;\mathcal{Y}_1,\ldots,\mathcal{Y}_k). \tag{10.78}$$

Supposons d'abord que les puissances individuelles $P_i = E\{X_i^2\}$ soient connues. Nous demandons au lecteur de se convaincre que le maximum est alors réalisé si les $\mathcal{X}_i$ sont indépendants et distribués de manière Gaussienne avec une variance P_i, et qu'on a sous ces conditions

$$I = \sum_{i=1}^{k} \frac{1}{2}\log(1 + \frac{P_i}{N_i}). \tag{10.79}$$

Le problème se ramène donc à choisir les P_i sous la contrainte

$$\sum_i P_i \leq P, \tag{10.80}$$

et les contraintes de faisabilité physique

$$P_i \geq 0, \tag{10.81}$$

de façon à maximiser (10.79). Formons le Lagrangien

$$J(P_1,\ldots,P_k) = \sum_{i=1}^{k} \frac{1}{2}\log\left(1 + \frac{P_i}{N_i}\right) + \lambda\left(\sum_i P_i\right) + \sum_{i=1}^{k} \mu_i P_i. \tag{10.82}$$

On doit donc avoir

$$\frac{\partial J(P_1,\ldots,P_k)}{\partial P_i} = 0, \forall i = 1,\ldots,k, \tag{10.83}$$

ce qui donne

$$\frac{1}{2}\frac{1}{P_i + N_i} + \lambda + \mu_i = 0. \tag{10.84}$$

Notre problème se ramène donc à la résolution du système d'équations

$$\frac{1}{2}\frac{1}{P_i + N_i} + \lambda + \mu_i = 0 \tag{10.85}$$

$$P_i \geq 0 \tag{10.86}$$

$$\sum_i P_i = P. \tag{10.87}$$

où λ et μ_i sont choisis de manière à maximiser (10.79).

Si nous supposons que les contraintes (10.86) sont inactives, les μ_i seront nuls $\forall i$. On peut se convaincre que la solution doit alors s'écrire sous la forme

$$P_i = \nu - N_i, \tag{10.88}$$

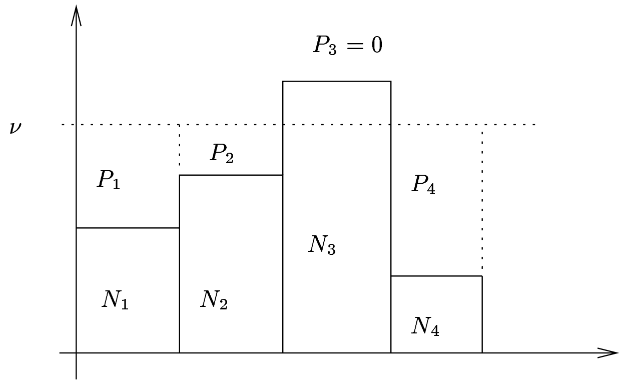


Figure 10.3. Allocation optimale de puissance sur canaux parallèles

avec ν choisi aussi grand que possible (de toute évidence le maximum de (10.79) doit correspondre à des valeurs élevées des P_i), c'est-à-dire de manière à satisfaire la contrainte (10.80). Notons que si les niveaux de bruit N_i sont identiques, on retrouve le résultat $P_i = \frac{P}{k}$ de la section précédente.

Si les niveaux de bruit sont différents d'un canal à l'autre, l'équation (10.88) nous apprend qu'il faut allouer plus de puissance aux canaux les moins bruités. En effet, on aura

$$P = \sum P_i = k\nu - \sum N_i \Rightarrow \nu = \frac{P + \sum N_i}{k} \Rightarrow P_i = \frac{P}{k} - \frac{kN_i - \sum N_i}{k}. \tag{10.89}$$

Lorsque les niveaux de bruit sont très différents, cette solution peut cependant conduire à des valeurs négatives des P_i pour les canaux les plus bruités. Dans ce cas, il faut réintroduire les contraintes de faisabilité (10.86) de manière à rendre la solution physiquement réalisable. Cela revient en fait à imposer $P_j = 0$ pour un certain nombre des canaux les plus bruités et on montre que la solution générale peut s'écrire

$$P_i = \max(\nu - N_i, 0), \tag{10.90}$$

avec ν choisi de manière à avoir $\sum P_i = P$. La situation est représentée graphiquement à la figure 10.3, où $P_3 = 0$ correspond au canal le plus bruité.

10.4.2 Canal à bruit coloré

Dans la section précédente nous avons considéré le cas d'un ensemble de canaux parallèles indépendants opérant avec une contrainte de puissance commune. Ici, nous allons considérer le cas où le bruit n'est pas indépendant. Cela correspond non seulement à une situation où les canaux parallèles partagent un même milieu physique, mais aussi au cas d'un canal à bruit Gaussien avec mémoire. Dans ce type de canal, nous pouvons considérer qu'un bloc de n utilisations successives du canal correspond à une utilisation de n canaux parallèles avec bruit coloré. En poursuivant le raisonnement à la limite vers le temps continu, nous obtenons un canal Gaussien en temps continu avec bruit coloré. Notre contrainte de puissance est ici une contrainte sur la puissance moyenne par utilisation du canal.

Comme ci-dessus, nous nous contentons du calcul de la capacité en information. Soit $\boldsymbol{K}_{\mathcal{Z}}$ la matrice de variance-covariance du bruit, et soit $\boldsymbol{K}_{\mathcal{X}}$ la matrice de variance-covariance du signal d'entrée. La contrainte de puissance peut alors s'écrire sous la forme

$$\frac{1}{n}\sum_{i=1}^{n} E\{X_i^2\} \leq P, \tag{10.91}$$

ou encore

$$\frac{1}{n}\mathrm{tr}\,(\boldsymbol{K}_{\mathcal{X}}) \leq P. \tag{10.92}$$

Nous avons

$$I(\mathcal{X}_1, \ldots, \mathcal{X}_n; \mathcal{Y}_1, \ldots, \mathcal{Y}_n) = H_d(\mathcal{Y}_1, \ldots, \mathcal{Y}_n) - H_d(\mathcal{Z}_1, \ldots, \mathcal{Z}_n), \tag{10.93}$$

où $H_d(\mathcal{Z}_1, \ldots, \mathcal{Z}_n)$ est seulement fonction des propriétés du bruit.

Il faut donc, pour réaliser la capacité, maximiser $H_d(\mathcal{Y}_1, \ldots, \mathcal{Y}_n)$. Celle-ci est maximale lorsque $\mathcal{Y}^n$ est distribuée de façon Gaussienne, ce qui est le cas si l'entrée l'est ($\mathcal{Y}^n = \mathcal{X}^n + \mathcal{Z}^n$, $\mathcal{X}^n$ et $\mathcal{Z}^n$ indépendants). Cela implique, étant donné l'indépendance de $\mathcal{X}^n$ et de $\mathcal{Z}^n$, que la matrice de variance-covariance de $\mathcal{Y}$ est $\boldsymbol{K}_{\mathcal{Y}} = \boldsymbol{K}_{\mathcal{X}} + \boldsymbol{K}_{\mathcal{Z}}$, et donc

$$H_d(\mathcal{Y}_1, \ldots, \mathcal{Y}_n) = \frac{1}{2} \log\left((2\pi e)^n \left|\boldsymbol{K}_{\mathcal{X}} + \boldsymbol{K}_{\mathcal{Z}}\right|\right). \tag{10.94}$$

Il faut donc ici choisir $\boldsymbol{K}_{\mathcal{X}}$ de façon à maximiser $|\boldsymbol{K}_{\mathcal{X}} + \boldsymbol{K}_{\mathcal{Z}}|$, sous la contrainte relative à la trace de $\boldsymbol{K}_{\mathcal{X}}$.

Diagonalisons alors $\boldsymbol{K}_{\mathcal{Z}}$, au moyen d'une matrice orthogonale $\boldsymbol{Q}$:

$$\boldsymbol{K}_{\mathcal{Z}} = \boldsymbol{Q}\Delta\boldsymbol{Q}^T, \text{ où } \boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}. \tag{10.95}$$

Alors,

$$\begin{aligned} |\boldsymbol{K}_{\mathcal{X}} + \boldsymbol{K}_{\mathcal{Z}}| &= \left|\boldsymbol{K}_{\mathcal{X}} + \boldsymbol{Q}\Delta\boldsymbol{Q}^T\right| && (10.96)\\ &= |\boldsymbol{Q}| \cdot |\boldsymbol{Q}^T\boldsymbol{K}_{\mathcal{X}}\boldsymbol{Q} + \Delta| \cdot |\boldsymbol{Q}^T| && (10.97)\\ &= |\boldsymbol{Q}^T\boldsymbol{K}_{\mathcal{X}}\boldsymbol{Q} + \Delta|. && (10.98) \end{aligned}$$

D'autre part, comme on a pour toute paire de matrices $\text{tr}(\boldsymbol{A}\boldsymbol{B}) = \text{tr}(\boldsymbol{B}\boldsymbol{A})$, on a

$$\text{tr}(\boldsymbol{Q}^T\boldsymbol{K}_{\mathcal{X}}\boldsymbol{Q}) = \text{tr}(\boldsymbol{K}_{\mathcal{X}}). \tag{10.99}$$

Le problème se ramène donc à trouver une matrice $\boldsymbol{A}$ symétrique définie positive, de trace majorée par nP et telle que le déterminant $|\boldsymbol{A} + \Delta|$ soit maximal. D'autre part, comme la somme de deux matrices symétriques définies positives est encore une matrice symétrique définie positive, la matrice $\boldsymbol{A} + \Delta$ l'est également.

Or, toute matrice symétrique définie positive $\boldsymbol{K}$ obéit à l'inégalité de *Hadamard* (la démonstration est donnée ci-dessous)

$$|\boldsymbol{K}| \leq \prod_{i=1}^{n} K_{i,i}, \tag{10.100}$$

avec égalité seulement si la matrice est diagonale. Donc

$$|\boldsymbol{A} + \Delta| \leq \prod_{i=1}^{n} (A_{i,i} + \lambda_i), \tag{10.101}$$

avec égalité seulement si $\boldsymbol{A}$ est diagonale. Puisque $\boldsymbol{A}$ est sujette à la contrainte sur sa trace on a

$$\frac{1}{n} \sum_{i=1}^{n} A_{i,i} \leq P, \tag{10.102}$$

et évidemment $A_{i,i} \geq 0$. Par conséquent, la valeur maximale de $|\boldsymbol{A} + \Delta|$ est atteinte si $\boldsymbol{A}$ est diagonale et telle que (en maximisant le log du membre de droite de (10.101) selon la technique de la section 10.4.1)

$$A_{i,i} = \max\{\nu - \lambda_i, 0\}, \tag{10.103}$$

où ν est choisi de manière à ce que

$$\sum_i A_{i,i} = nP. \tag{10.104}$$

Le résultat est donc analogue à ce que nous avons trouvé précédemment, à la nuance près que nous devons travailler dans un espace décorrélé. Nous avons, en diagonalisant la matrice de variance-covariance du bruit, transformé les canaux en n canaux parallèles à bruit indépendant. Dans cet espace, les entrées seront également indépendantes, et il suffit d'effectuer la transformation inverse pour obtenir les combinaisons de ces signaux indépendants avec lesquelles il faut alimenter les canaux physiques de départ.

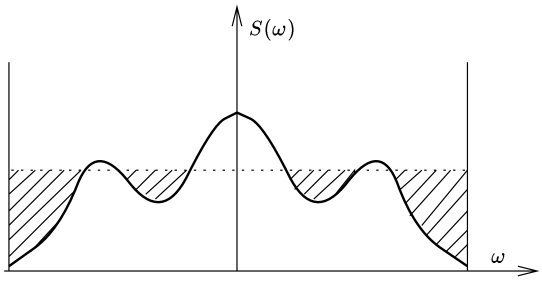


Figure 10.4. Remplissage du spectre de bruit

Inégalité de Hadamard. Démontrons l'inégalité de Hadamard en nous servant de ce que nous apprend la théorie de l'information. Soit, $\boldsymbol{K}$ une matrice symétrique définie positive d'ordre n. Alors, il existe un vecteur aléatoire Gaussien de dimension n qui a cette matrice comme matrice de variance-covariance. Soit, $\mathcal{X}^n$ ce vecteur. On a

$$H_d(\mathcal{X}_1, \ldots, \mathcal{X}_n) \leq \sum_i H_d(\mathcal{X}_i), \tag{10.105}$$

avec égalité si, et seulement si les $\mathcal{X}_i$ sont indépendantes ($\Leftrightarrow$ matrice diagonale). Il suffit alors de remplacer les deux membres de cette inégalité par leurs valeurs (distributions Gaussiennes) pour obtenir

$$\log |\boldsymbol{K}| \leq \sum_i \log K_{i,i} \tag{10.106}$$

équivalente à l'inégalité de Hadamard.

Passage à la limite du temps continu. Soit un canal en temps continu, à bruit Gaussien additif, dont la matrice de variance-covariance fini-dimensionnelle (en n instants d'échantillonnage successifs) est $\boldsymbol{K}_{\mathcal{Z}}^{(n)}$. Si le processus est stationnaire, les éléments i, j de cette matrice ne dépendent que de $i - j$ (on dit que la matrice est une matrice de Toeplitz), et on montre que ses valeurs propres tendent vers une limite lorsque n tend vers l'infini. Le spectre (ensemble de valeurs propres de la matrice avec leur multiplicité) tend alors vers une fonction continue. Plus précisément, lorsque nous faisons tendre $n \to \infty$ la matrice de variance-covariance tend vers un opérateur linéaire et stationnaire en temps discret, c'est-à-dire vers un système de convolution. L'action de cette matrice sur une suite de valeurs devient alors le produit de convolution, dont les vecteurs propres tendent vers les signaux harmoniques $\exp(j\omega t)$ et les valeurs propres de ces vecteurs propres sont $S(\omega)$, où $S(\omega)$ est la transformée de Fourier (purement réelle) de la fonction d'autocovariance $R_{\mathcal{X}\mathcal{X}}(\tau)$ (symétrique), c'est-à-dire la densité spectrale de puissance du bruit. Si le canal n'est pas limité en fréquence, on considérera dans le processus de passage à la limite également la limite pour $\Delta t \to 0$.

Donc, dans ce cas l'optimisation de la capacité se traduit par un schéma de "remplissage" du spectre du bruit, tel qu'illustré à la figure 10.4. Donc, pour un canal de ce type, le signal d'entrée devrait être choisi de manière à concentrer l'énergie dans les bandes de fréquence où le bruit est le plus faible. A nouveau, remarquons que cela revient à décorréler le bruit (équiper le canal de deux filtres (un en entrée, l'autre inversé en sortie) qui ont pour effet de présenter le canal comme si le bruit était blanc).

On montre que la capacité vaut

$$C = \int \frac{1}{2} \log \left(1 + \frac{\max\{\nu - N(f), 0\}}{N(f)} \right) df \tag{10.107}$$

où $N(f)$ désigne la densité de puissance dans la bande de fréquences $[f, f + df]$, et où ν est fixé de façon à ce que $\int \max\{\nu - N(f), 0\} df = P$.

10.5 DEFINITION PLUS GENERALE D'ESPACES DE SIGNAUX

Dans cette section nous allons discuter de façon plus générale de la démarche que nous venons d'adopter à la section 10.3.

10.5.1 Modulation par signaux temporels continus

Nous allons supposer que nous souhaitons transmettre sur un canal continu un message i, parmi un ensemble fini de messages possibles $\{1, \ldots, M\}$. Nous attribuons a priori à chaque message un signal différent $x_i(t)$ qui peut être transmis sur le canal et nous désignerons par $p(x_i)$ la probabilité d'émission du message i. Comme le canal est bruité, à chacun de ces messages correspond à la sortie un ensemble de messages possibles avec une certaine distribution de probabilité conditionnelle (c'est-à-dire un processus aléatoire $\mathcal{Y}_i(t)$). Nous supposons pour le moment que le nombre de signaux reconnaissables à la sortie est fini, en d'autre termes, $\mathcal{Y}_i(t)$ est un v.a. discrète $\forall t$, et désignons par $y_j(\cdot)$ les N sorties possibles.

A tout couple entrée sortie i, j, on peut associer le nombre $p(y_j|x_i)$ qui désigne la probabilité conditionnelle de recevoir le signal $y_j(t)$ sachant que $x_i(t)$ fut envoyé. Si nous voulons minimiser la probabilité d'erreur, notre règle de décision doit choisir lorsqu'un signal $y_j(t)$ est reçu, l'indice de message i tel que

$$p(x_i|y_j) = \frac{p(y_j|x_i)p(x_i)}{p(y_j)}$$

soit maximale. En l'absence d'information sur les $p(x_i)$ la stratégie usuelle consiste à supposer que les messages sont équiprobables, et donc à choisir l'indice i qui maximise la vraisemblance

$$p(y_j|x_i). \tag{10.108}$$

Ayant cette règle de décision, l'optimisation du système de communication consiste alors à choisir le répertoire de signaux $x_i(t)$ de façon à rendre aussi différentes que possibles, en moyenne, les vraisemblances associées d'une part à une décision correcte, et d'autre part à une décision erronée.

10.5.2 Bruit additif, signaux de durée et d'énergie finie

Dans notre schéma, un bruit additif se traduit par $y(t) = x_i(t) + n(t)$ où $n(t)$ est un processus aléatoire indépendant de l'entrée. Pour un message émis $x_i(t)$, les messages reçus sont alors distibués avec une distribution de probabilité semblable à celle du bruit, mais centrée autour $x_i(t)$.

D'autre part, nous supposons que les signaux sont de durée finie, définis sur un intervalle de temps $[0, T]$ et tels que leur énergie

$$E_i = \int_0^T x_i^2(t)dt < \infty, \forall i = 1, \ldots, M. \tag{10.109}$$

Ces fonctions peuvent être vues comme des vecteurs de l'espace de Hilbert de fonctions de carré sommable (voir appendice F). Le bruit additif est alors équivalent à l'addition vectorielle d'un vecteur de bruit. Même si le bruit n'est pas de carré sommable sur $[0, T]$ (ce qui est par exemple le cas si celui-ci est blanc et de spectre illimité), on montre que pour notre problème de décision optimale, les composantes du bruit qui ne font pas partie de l'espace de Hilbert sont "transparentes". Nous pouvons supposer que $n(t)$ appartient à cet espace, et que nous sommes capables de calculer la distribution de probabilité de celui-ci. Nous allons désigner par $\boldsymbol{x_i}, \boldsymbol{y}$ et $\boldsymbol{n}$ les vecteurs de cet espace de Hilbert qui représentent les signaux correspondants (nous utilisons des lettre grasses pour insister sur le fait que nous travaillons dans un espace vectoriel).

Nous pouvons donc identifier le vecteur de bruit (la partie utile) à la différence entre le signal reçu et le signal émis :

$$\boldsymbol{n} = \boldsymbol{y} - \boldsymbol{x_i}.$$

Si nous désignons par $p_n(\cdot)$ la densité du bruit on a

$$p(\boldsymbol{y}|\boldsymbol{x_i}) = p_n(\boldsymbol{y} - \boldsymbol{x}_i). \tag{10.110}$$

L'application du critère de vraisemblance maximale consiste alors à choisir l'indice i qui maximise $p_n(\boldsymbol{y} - \boldsymbol{x}_i)$. Si le bruit est isotrope, c'est-à-dire si sa densité de probabilité ne dépend que de la norme du vecteur $\boldsymbol{n}$, et si cette densité décroît lorsque la norme du vecteur bruit croît, cela revient à choisir l'indice i qui correspond au vecteur d'entrée le plus proche de la sortie.

Rappelons que la norme de notre espace de Hilbert est induite par le produit scalaire

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \int_0^T x(t)y(t)dt. \tag{10.111}$$

La distance entre la sortie et une certaine entrée vaut par conséquent

$$\|\boldsymbol{x}_i - \boldsymbol{y}\|^2 = \|\boldsymbol{y}\|^2 + E_i - 2\langle \boldsymbol{x}_i, \boldsymbol{y} \rangle. \tag{10.112}$$

Comme la sortie est fixée, celle-ci est donc minimale lorsque

$$E_i - 2\langle \boldsymbol{x}_i, \boldsymbol{y} \rangle \tag{10.113}$$

est minimal. En particulier, si tous les signaux d'entrée sont de même énergie on aura $E_i = E$, et le choix de vraisemblance maximale se ramène au choix de i qui maximise le produit scalaire

$$\langle \boldsymbol{x}_i, \boldsymbol{y} \rangle. \tag{10.114}$$

10.5.3 Récepteur optimal

On voit que le décodage se ramène au calcul du produit scalaire dans l'espace de Hilbert. Il suffit, pour cela, de disposer d'un filtre adapté pour chacun de signaux d'entrée, c'est-à-dire d'un filtre dont la réponse impulsionnelle est $h_i(t) = x_i(T - t)$. En effet, la sortie de ce type de filtre est

$$(y * h_i)(t) = \int_0^T y(u)h_i(t-u)du = \int_0^T y(u)x_i(T-t+u)du \tag{10.115}$$

et donc

$$(y * h_i)(T) = \int_0^T y(u)x_i(u)du = \langle \boldsymbol{x}_i, \boldsymbol{y} \rangle. \tag{10.116}$$

10.5.4 Représentation du bruit blanc dans un espace de signaux

On peut représenter les signaux utilisés à l'entrée d'un canal à l'aide d'une base de signaux orthonormés, disons $\phi_i(t)$, telles que

$$\langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle = \delta_{i,j}. \tag{10.117}$$

Tout signal d'entrée possible peut alors s'écrire sous la forme

$$x(t) = \sum_i x_i \phi_i(t), \tag{10.118}$$

avec

$$x_i = \langle \boldsymbol{x}, \boldsymbol{\phi}_i \rangle. \tag{10.119}$$

L'énergie d'un signal est alors obtenue par

$$E_x = \langle \boldsymbol{x}, \boldsymbol{x} \rangle = \sum_i x_i^2. \tag{10.120}$$

Partant d'un ensemble de signaux en nombre fini, la base peut être construite par une procédure d'orthogonalisation. L'espace de signaux engendré par ces vecteurs est un sous-espace de Hilbert de dimension finie de l'espace des fonctions de carré sommable sur $[0, T]$.

Un signal de bruit est alors vu sous le filtre formé par ce sous-espace de Hilbert. Il est, du point de vue de la décision optimale, entièrement déterminé par ces composantes

$$n_i = \int_0^T n(t)\phi_i(t)dt, \tag{10.121}$$

qui sont des variables aléatoires. Remarquons que ces variables aléatoires sont des combinaisons linéaires (en nombre infini) des variables conjointement Gaussiennes (les valeurs de $n(t)$ à différents instants).

On montre de façon assez directe que

$$E\{n_i n_j\} = \int_0^T \int_0^T R_{\mathcal{N}\mathcal{N}}(t_1 - t_2)\phi_i(t_1)\phi_j(t_2)dt_1 dt_2,$$

et si le bruit est blanc on a par définition $R_{\mathcal{N}\mathcal{N}}(\tau) = \frac{N_0}{2}\delta(\tau)$, ce qui donne

$$E\{n_i n_j\} = \frac{N_0}{2}\delta_{i,j}. \tag{10.122}$$

Par conséquent, dans un espace (ou sous-espace) de Hilbert de fonctions de carré intégrable, les composantes du bruit blanc sont i.i.d. de variance $\frac{N_0}{2}$ dans toute base orthonormée. Si le bruit est Gaussien, les composantes (qui sont des combinaisons linéaires de valeurs $n(t)$) sont conjointement Gaussiennes de matrice de variance-covariance $\frac{N_0}{2}\boldsymbol{I}$ diagonale. Elles sont donc aussi indépendantes.

On déduit de ce qui précède que les raisonnements effectués précédemment sur les signaux engendrés par la base de Shannon restent en réalité valables quel que soit l'espace de signaux utilisé.

10.6 RESUME

Dans ce chapitre nous avons entrepris d'étudier la généralisation du second théorème de Shannon au cas des canaux à signaux continus en temps et en valeurs.

Nous avons commencé par introduire intuitivement la notion de processus aléatoire en temps continu, qui constitue le modèle mathématique à la base de l'étude de ce type de systèmes physiques. Ensuite, nous nous somme focalisés sur l'étude du canal à bruit additif et Gaussien. Ce type de canal n'est pas un modèle universel, car dans de nombreuses applications le bruit n'est pas nécessairement additif ou Gaussien. Il constitue néanmoins un modèle très intéressant pour deux raisons : d'un part, la loi des grands nombres fait que le modèle additif Gaussien est souvent réaliste; d'autre part, comme nous l'avons vu dans ce qui précède, l'exploitation de ce type de canal conduit à des solutions mathématiques *linéaires*, élégantes et puissantes à la fois.

C'est d'ailleurs une des raisons pratiques qui fait que lorsqu'un canal n'est pas additif et Gaussien, on s'arrange souvent en pratique pour l'emballer de manière à ce que vu de l'extérieur il paraisse comme tel. Par exemple, les codes entrelacés utilisés lorsque le canal est sujet à du bruit ponctuel (par exemple les enregistrements sur disques compacts, certains canaux hertziens) ont précisément pour but de rendre le bruit Gaussien.

Par ailleurs, la démarche utilisée dans nos raisonnements faisant appel à des espaces de signaux est évidemment une démarche générale que nous n'avons fait qu'effleurer. Nous observons que l'augmentation de la dimension de cet espace de signaux permet d'atteindre la capacité. C'est donc en construisant des espaces de dimension aussi grande que possible, et en exploitant cette dimension pour le choix des signaux d'entrée qu'on lutte le plus efficacement contre le bruit. Ceci est évidemment très semblable aux résultats obtenus dans le cas discret. Alors que dans le cas discret la dimension de l'espace est équivalente à la longueur des blocs de symboles utilisés pour le codage de canal, ici, la dimension est définie par le nombre de fonctions orthonormées utilisées effectivement pour réprésenter les signaux d'entrée.

11 THEORIE DE LA DISTORSION

La description d'un nombre réel arbitraire nécessite un nombre infini de bits; donc une représentation finie d'une variable aléatoire continue ne peut pas être parfaite. On peut par contre se demander s'il est possible de choisir cette représentation de manière optimale, pour un nombre donné de bits, et quelle est la précision atteignable dans des conditions optimales. Pour préciser le problème, il est nécessaire de définir tout d'abord une mesure de qualité d'une représentation d'une source. Cela peut être fait en définissant une mesure de distorsion qui évalue la distance entre une variable aléatoire et sa représentation. Le problème de base de la théorie de la distorsion peut alors se formuler comme suit :

> *Etant donné une distribution de source et une mesure de distorsion, quelle est la valeur minimale de la distorsion atteignable pour un débit de source fixé a priori ?*
> ou de manière equivalente
> *Quelle est le débit minimal nécessaire pour atteindre un certain objectif en terme de distorsion ?*

Un des résultats les plus surprenants de cette théorie est qu'il est plus efficace de coder de façon conjointe un certain nombre de variables aléatoires que de les coder séparément, même lorsque celles-ci sont indépendantes.

La théorie de la distorsion (c'est-à-dire de la quantification optimale) peut être appliquée aussi bien à des variables aléatoires discrètes que continues. Nous en donnons ici une très brève introduction. Nous verrons que la théorie de la compression de données réversible apparaît comme un cas particulier de la théorie de la distorsion avec une contrainte de distorsion nulle.

11.1 QUANTIFICATION SCALAIRE

Commençons par le problème simple de répresenter un nombre réel par un nombre fini de bits. Soit $\mathcal{X}$ une variable aléatoire, et désignons par $\hat{X}(X)$ la représentation associée à la valeur X de $\mathcal{X}$. Si nous nous donnons R bits pour représenter $\mathcal{X}$, alors l'ensemble de valeurs de $\hat{X}$ peut contenir 2^R valeurs distinctes. La question est de savoir comment placer ces 2^R valeurs sur la droite réelle de façon à minimiser la distorsion et comment encoder un nombre réel en choisissant une des ces valeurs. Ce problème revient à définir une partition de la droite réelle en 2^R régions auxquelles on associe les 2^R valeurs de $\hat{X}$.

Soit $\mathcal{X} \sim \mathcal{N}(0, \sigma^2)$ et supposons que nous utilisions une mesure de distorsion quadratique. Cela veut dire que nous cherchons une fonction $\hat{X}(X)$ prenant au plus 2^R valeurs distinctes et telle que $E\{(\mathcal{X} - \hat{X}(\mathcal{X}))^2\}$ soit minimale.

Si nous nous donnons 1 seul bit pour représenter $\mathcal{X}$, il est clair que ce bit doit permettre de distinguer entre les valeurs positives et négatives. De plus, pour minimiser l'erreur quadratique, chacune des deux valeurs associées doit être égale à l'espérance de $\mathcal{X}$ dans la région qui lui correspond. On a donc

$$\hat{X}(X) = \begin{cases} +\sqrt{\frac{2}{\pi}}\sigma, & \text{si } x \geq 0 \\ -\sqrt{\frac{2}{\pi}}\sigma, & \text{si } x < 0 \end{cases} \tag{11.1}$$

L'erreur quadratique moyenne d'approximation de ce schéma vaut $\frac{\pi-2}{\pi}\sigma^2 \approx 0.363\sigma^2$.

Si nous nous donnons 2 bits, la situation est moins simple. Clairement, nous cherchons à diviser la droite réelle en quatre régions et nous voulons utiliser un point dans chaque région pour encoder les valeurs qui lui appartiennent. Mais il n'est pas évident quels régions et points utiliser. On peut cependant énoncer deux propriétés simples qui doivent être vérifiées pour une telle quantification optimale:

- Pour un ensemble de points de reconstruction donnés (codes), les régions associées doivent correspondre à la règle du plus proche voisin : une valeur est encodée par l'élément du code qui lui est le plus proche au sens de la distance utilisée pour la mesure de distorsion.
- Les points de reconstruction doivent être choisis de manière à minimiser la distorsion moyenne sur la région qui leur correspond.

Ces deux propriétés nous permettent de formuler un algorithme simple pour trouver un "bon" quantificateur: (i) on commence avec un ensemble de points de reconstruction quelconque (leur nombre étant fixé à priori); (ii) puis on construit les régions optimales correspondantes; (iii) on calcule pour chaque région le point de reconstruction optimal (il s'agit de l'espérance conditionnelle dans cette région, si on utilise une distorsion quadratique); (iv) on continue à itérer les étapes (ii) et (iii) tant que les points de reconstruction ne sont pas stabilisés. A chaque étape de cet algorithme la distorsion moyenne décroît; l'algorithme doit donc s'arrêter en un minimum local de la mesure de distorsion. Cet algorithme porte le nom *d'algorithme de Lloyd*. Il peut se généraliser à des variables aléatoires vectorielles. Dans la version où on utilise un échantillon de taille finie pour calculer les espérances conditionnelles, cet algorithme porte le nom *d'algorithme K-means*, bien connu en apprentissage automatique.

Supposons qu'au lieu de coder un seul nombre à la fois, nous cherchions à coder des suites de n valeurs d'une source sans mémoire à l'aide de nR bits. Comme ces nombres sont indépendants, on pourrait croire que la quantification optimale consisterait à coder les n nombres de façon indépendante sur R bits comme ci-dessus. Mais cela n'est pas vrai, comme nous allons le voir ci-dessous.

Notons que la procédure ci-dessus ne converge pas nécessairement vers des régions équiprobables; il s'en suit qu'après quantification il se pourrait qu'il soit possible de réduire le débit en codant de façon appropriée les symboles du code. Ci-dessous, nous allons analyser les limites ultimes atteignables au moyen de ces deux opérations : quantification + codage de source.

11.2 DÉFINITIONS

Soit une source qui émet n symboles successifs indépendants distribués selon une loi $p(x), x \in \mathcal{X}$. Nous allons supposer pour simplifier les démonstrations que l'alphabet de source $\mathcal{X}$ est fini, mais la plupart des raisonnements peuvent être étendus au cas continu.

L'encodeur décrit la séquence X^n au moyen d'un nombre entier $f_n(X^n) \in \{1, 2, \ldots, 2^{nR}\}$. Le décodeur repésente X^n au moyen d'une approximation $\hat{X}^n$.

Une fonction de distorsion est par définition une mesure

$$d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+ \tag{11.2}$$

telle que $d(X, \hat{X})$ mesure le coût associé à la représentation de X par $\hat{X}$. Typiquement les deux alphabets sont identiques et normalement $d(X, X) = 0$. Des exemples typiques de mesures de distorsion utilisées sont la mesure de Hamming (ou distance de Dirac)

$$d(X,\hat{X}) = \begin{cases} 0 & \text{si } X = \hat{X} \\ 1 & \text{si } X \neq \hat{X} \end{cases} \tag{11.3}$$

pour les sources discrètes, et la distance Euclidienne pour les sources continues

$$d(X,\hat{X}) = (X - \hat{X})^2. \tag{11.4}$$

La distorsion entre séquences de symboles est définie comme la moyenne algébrique des distorsions sur les symboles correspondants

$$d(X^n,\hat{X}^n) = \frac{1}{n}\sum_{i=1}^{n} d(X_i,\hat{X}_i). \tag{11.5}$$

Une code de quantification vectorielle $(2^{nR}, n)$ est défini par une fonction d'encodage

$$f_n \ : \mathcal{X}^n \to \{1, 2, \ldots, 2^{nR}\}, \tag{11.6}$$

et une fonction de décodage (ou de reconstruction)

$$g_n \ : \{1, 2, \ldots, 2^{nR}\} \to \hat{\mathcal{X}}^n. \tag{11.7}$$

La distorsion D de ce code est définie par

$$D = E\{d(X^n, g_n(f_n(X^n)))\}. \tag{11.8}$$

Une paire débit distorsion (R, D) est dite atteignable (à la limite) s'il existe une suite de codes de quantification vectorielle $(2^{nR}, n)$, (f_n, g_n), telle que

$$\lim_{n\to\infty} E\{d(X^n, g_n(f_n(X^n)))\} \leq D. \tag{11.9}$$

La région de distorsion de débit d'une source est par définition la fermeture de l'ensemble des paires (R, D) atteignables pour cette source. La fonction de *débit de distorsion* $R(D)$ est la borne inférieure des débits R tels que (R, D) soit atteignable. La fonction de *distorsion de débit* est la borne inférieure des distorsions D tels que (R, D) soit atteignable.

La fonction de *distorsion de débit en information* pour une source et une mesure de distorsion D est définie par

$$R^{(I)}(D) \triangleq \min_{p(\hat{X}|X) \,:\, \sum_{(X,\hat{X})} p(X)p(\hat{X}|X)d(X,\hat{X}) \leq D} I(X;\hat{X}). \tag{11.10}$$

Le théorème principal de la théorie de la distorsion peut alors être énoncé comme suit.

Théorème. *La fonction de distorsion de débit $R(D)$, pour une source sans mémoire et une fonction de distorsion d bornée, est identique à la fonction de distorsion d'information correspondante $R^{(I)}(D)$.*

Nous renvoyons le lecteur à la référence [CT91] pour une démonstration de ce théorème dans le cas discret.

11.3 EXEMPLES DE FONCTIONS DE DISTORSION DE DÉBIT

11.3.1 Source binaire

La fonction de distorsion de débit d'une source binaire (probabilités $p, 1-p$) avec la distance de Hamming (taux d'erreur) est donnée par

$$R(D) = \begin{cases} H_2(p) - H_2(D) & \text{si } 0 \leq D \leq \min\{p, 1-p\} \\ 0, & \text{si } D > \min\{p, 1-p\}. \end{cases} \tag{11.11}$$

Cette fonction est représentée à la figure 11.1, dans le cas où $p = 0.5$.

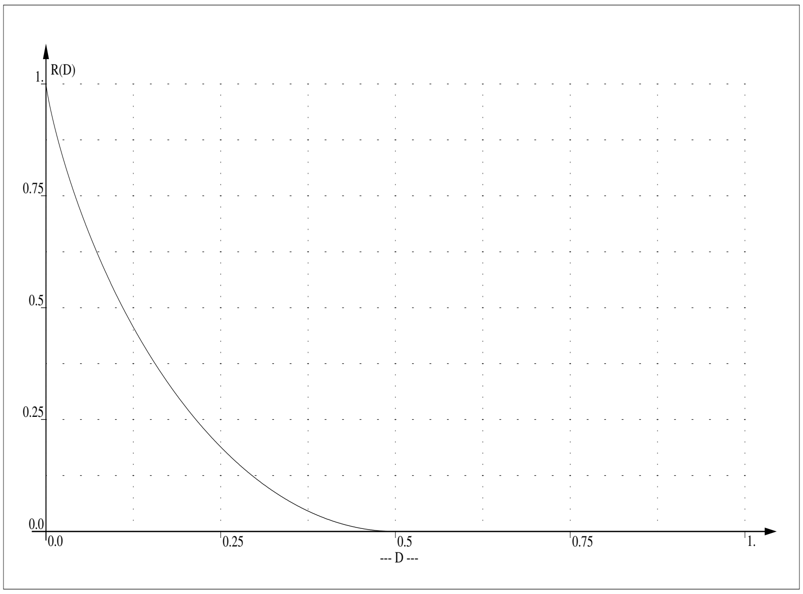


Figure 11.1. Fonction de distorsion de débit d'une source binaire

On voit que si on force une distorsion nulle ($D = 0$), ce qui correspond à un taux d'erreurs de décodage nul, on obtient la condition

$$R(0) = H_2(p), \tag{11.12}$$

conforme aux résultats relatifs au codage de source du chapitre 8.

Suggestion. Calculer le débit et le taux d'erreurs D qu'on obtiendrait si on codait n bits de la source ($p = 0.5$), en ne gardant que les $n - 1$ premiers bits, c'est-à-dire en laissant le décodeur choisir le n-ème bit de façon arbitraire.

Démonstration. Nous allons nous contenter de calculer la fonction de distorsion d'information correspondante $R^{(I)}(D)$, le théorème permettant alors de conclure sur $R(D)$ par la même occasion.

Nous pouvons supposer que $p < 0.5$. D'autre part, si $D > p$ il est trivial que le débit de $R = 0$ est atteignable. Calculons la fonction

$$R^{(I)}(D) \triangleq \min_{p(\hat{X}|X)\,:\sum_{(X,\hat{X})} p(X)p(\hat{X}|X)d(X,\hat{X})\leq D} I(X;\hat{X}). \tag{11.13}$$

Désignons par $\oplus$ l'opération d'addition modulo 2. Donc $X \oplus \hat{X} = 1$ représente la condition $X \neq \hat{X}$. Montrons tout d'abord que $H_2(p) - H_2(D)$ est un borne inférieure de $R^{(I)}(D)$; nous montrerons ensuite qu'elle est réalisable.

Nous avons

$$\begin{aligned} I(X;\hat{X}) &= H(X) - H(X|\hat{X}) && (11.14)\\ &= H_2(p) - H(X \oplus \hat{X}|\hat{X}) && (11.15)\\ &\geq H_2(p) - H(X \oplus \hat{X}) && (11.16)\\ &\geq H_2(p) - H_2(D), && (11.17)\end{aligned}$$

la dernière inégalité résultant du fait que $P(X \neq \hat{X}) \leq D$ et que la fonction $H_2(\cdot)$ est croissante sur l'intervalle $[0; 0.5]$. On a donc bien

$$R(D) \geq H_2(p) - H_2(D). \tag{11.18}$$

Il suffit donc mainenant de trouver une distribution conjointe de $(X, \hat{X})$ qui respecte la distribution de X et la contrainte de distorsion et telle que $I(X;\hat{X}) = R(D)$. Nous pouvons la construire de la manière suivante :

- Choisissons $P(X = 0|\hat{X} = 0) = 1 - D$ et $P(X = 1|\hat{X} = 1) = 1 - D$; il est clair que cette distribution donne lieu à une distorsion (taux d'erreur de décodage) qui vaut D.
- Choisissons maintenant $P(\hat{X} = 0)$ de telle manière que $P(X = 0) = p$,de façon à respecter le modèle de la source. Il suffit de prendre
$$P(\hat{X} = 0) = \frac{1 - p - D}{1 - 2D}, \tag{11.19}$$
ce qui est possible puisque $D < p < 0.5$.

Calculons alors $I(X; \hat{X})$ avec ce choix

$$\begin{aligned} I(X;\hat{X}) &= H(X) - H(X|\hat{X}) && (11.20) \\ &= H_2(p) - H_2(D). && (11.21) \end{aligned}$$

□

Suggestion. Montrer que le choix (11.19) donne bien lieu à $P(X = 0) = p$.

11.3.2 Source Gaussienne

La fonction de distorsion de débit d'une source sans mémoire qui émet des v.a. $\mathcal{X} \sim \mathcal{N}(0, \sigma^2)$ est

$$R(D) = \begin{cases} \frac{1}{2}\log\frac{\sigma^2}{D}, & \text{si } 0 \leq D \leq \sigma^2 \\ 0, & \text{si } D > \sigma^2. \end{cases} \tag{11.22}$$

On voit que si on fait tendre D vers 0, le débit doit tendre vers l'infini, ce qui est bien compatible avec notre intuition. D'autre part, si on admet une distorsion moyenne égale à la variance de la v.a. le débit peut être nul. En dehors de ces deux extrêmes, le débit est une image directe du rapport signal bruit.

En particulier, si on prend $D = \frac{\sigma^2}{4}$ on obtient $R = 1$. Cela veut donc dire que le codage de messages longs permet d'obtenir une distorsion à la limite égale à $\frac{\sigma^2}{4}$ pour un débit d'un bit par symbole de source. On peut comparer ce chiffre à la valeur de $0.363\sigma^2$ obtenue au début de ce chapitre pour le codage symbole par symbole.

Suggestion. Trouver la démonstration en suivant un cheminement analogue à celui de la section précédente.

11.3.3 Description simultanée de plusieurs sources Gaussiennes

Le codage simultané de sources Gaussiennes indépendantes donne lieu à une approche d'allocation de distorsion pour chaque source fonction de sa variance, partant d'un raisonnement similaire à celui utilisé pour les canaux Gaussiens parallèles. Nous demandons au lecteur intéressé de bien vouloir consulter la référence [CT91].

III APPLICATIONS AU CODAGE

12 MOTIVATION

Le but de cette partie du cours est de parcourir les algorithmes de codage de données les plus importants, avec une approche suffisamment analytique pour permettre à l'étudiant d'en assimiler les bases théoriques et pratiques et de se sentir à l'aise lorsqu'il consultera la littérature importante qui existe dans ce domaine.

La matière est divisée en 3 grands chapitres qui traitent respectivement de la compression *réversible* de textes (données discrètes), la compression *irréversible* de données analogiques et plus particulièrement d'images, et enfin des algorithmes de codage de canal (codes détecteurs et correcteurs d'erreurs) en insistant sur les progrès récents dans ce domaine (Turbo-codes).

On insiste beaucoup sur l'utilisation de modèles probabilistes des sources et canaux de données dans le cadre des algorithmes de codage/décodage et sur le compromis *efficacité de codage* vs *complexité computationnelle* qui est omniprésent.

Alors que la compression de textes a atteint un certain niveau de maturité depuis plus de dix ans, il reste encore beaucoup de travail à faire dans le domaine de la compression d'images et de sons et celui du codage de canal.

13 COMPRESSION DE DONNEES REVERSIBLE

13.1 INTRODUCTION

Dans ce chapitre nous nous intéressons au problème du codage de source, c'est-à-dire à la compression de données.

Nous allons supposer que les alphabets initiaux et finaux sont binaires ($s_i = \{0, 1\}$). A l'occasion nous mentionnerons les extensions des méthodes et résultats que nous décrirons dans le cas plus général d'alphabets quelconques. Mais, comme l'alphabet binaire est roi à ce stade de l'histoire informatique, il est plus pratique de particulariser directement au cas binaire que de s'encombrer de notations générales.

De nombreux problèmes de compression de données sont de la forme suivante : on dispose d'un long texte (ou fichier) binaire T qu'on souhaite transformer en un texte binaire U plus court, de telle manière que T soit exactement (ou au moins presque) recouvrable à partir de U. Dans d'autres cas, on ne dispose pas à l'avance du texte T, car celui-ci est produit en temps-réel sous la forme d'une suite de symboles et d'un marqueur (symbole spécial) indiquant la fin de la communication. Il est évidemment possible de stocker le texte, pour se ramener au cas précédent, mais cela nécessite des capacités de stockage éventuellement de grande taille, et surtout cela peut introduire des délais indésirables entre le moment ou un symbole est émis et le moment où il est décodé (cf multimédia).

Lorsque T est exactement recouvrable, nous parlerons de compression ou de code *réversible*, sinon nous dirons que le code est irréversible, ou avec pertes. Le *taux de compression* est défini par le rapport des longueurs[1] des deux textes

$$\frac{\ell(T)}{\ell(U)}.$$

Le taux de compression réalisé par une méthode est alors le taux moyen obtenu en utilisant cette méthode, ce qui veut dire que la moyenne est calculée sur tous les textes d'entrée T auxquels cette méthode est susceptible d'être appliquée.

Certaines méthodes de compression de données sont mieux adaptées à la situation où le texte T est complètement disponible au moment de la construction du code (il peut par exemple être stocké sous forme de fichier). Cela permet notamment l'étude statistique du texte afin de construire le code. Lorsque le fichier est obtenu en temps-réel sous la forme d'un flux de bits, et qu'on souhaite éviter le stockage de celui-ci, il n'est pas possible de connaître

[1]Cette formule reste encore valable pour des alphabets non binaires, à condition que les alphabets d'entrée et de sortie aient le même nombre de symboles. Dans le cas général, il faut utiliser la formule $\frac{\ell(T)\log q(T)}{\ell(U)\log q(U)}$ où $q(T)$ et $q(U)$ désignent respectivement les tailles des alphabets d'entrée et de sortie.

au moment de la construction du code et de l'encodage les propriétés des bits futurs. Il est alors nécessaire de faire appel à des méthodes de compression/décompression de flux de données. Par conséquent, nous ne distinguerons pas seulement nos méthodes de compression sur base de leur taux de compression, ou de leur réversibilité, mais aussi selon leur vitesse de traitement et selon la façon dont elles traitent les symboles source et les changements fondamentaux en cours de route concernant les propriétés des suites de bits à coder.

Enfin, un autre point qu'il faut garder à l'esprit lorsqu'on compare les méthodes de codage, concerne les *coûts cachés*. Ceux-ci se présentent souvent sous la forme d'instructions (informations) nécessaires pour recouvrir T à partir de U. De toute évidence, il n'est pas très utile d'obtenir un très bon taux de compression, si celui-ci s'accompagne d'un ensemble très important d'instructions nécessaires pour le décodage (qui doivent également être transmises sur le canal). Un autre type de coût caché est lié à la complexité computationnelle des méthodes de compression/décompression mises en oeuvre. Il est assez clair que les coûts cachés sont liés, généralement de façon inverse, à la vitesse de traitement, à l'adaptabilité, au taux de compression, et à la qualité de l'information récupérée après décompression.

Nous allons voir dans ce chapitre quatre types de méthodes réversibles, regroupées dans les quatre sections suivantes.

1. Méthodes d'ordre zéro, non-adaptatives :

   (a) schémas de remplacement (codage en blocs : Huffman, Shannon...);

   (b) codage arithmétique qui associe à un mot T un sous-intervalle de $[0,1[$ correspondant à un segment fini de l'expansion binaire d'un nombre réel $r \in [0,1[$.

2. Méthodes d'ordre élevé non-adaptatives.

3. Méthodes adaptatives.

4. Méthodes utilisant des dictionnaires (Lempel-Ziv, GNU zip ...).

Pour des raisons évidentes, nous ne pourrons pas traiter en détails l'ensemble de ces sujets. Plutôt que de consacrer beaucoup de temps à la description détaillée d'une technique particulière, nous avons préféré opter pour une couverture assez large du domaine de façon à en donner une vue d'ensemble, et à mettre en évidence les relations entre les différentes méthodes.

Nous attendrons la fin de ce chapitre pour donner une synthèse de ces méthodes, qui constituent aujourd'hui l'état de l'art dans le domaine de la compression de données réversible de texte. Au chapitre 14, nous nous intéresserons au problème de la compression de données d'images, et plus généralement, de données où les structures de corrélations sont multidimensionnelles. Le domaine étant extrêmement vaste, nous nous contenterons de décrire deux approches pour la compression de données irréversible dans ce cas.

13.2 METHODES D'ORDRE ZERO

13.2.1 Schémas de remplacement

Les méthodes de cette famille procèdent comme suit.

1. Choix d'un ensemble de mots binaires de source $s_1, s_2, \ldots, s_m$. Comme nous le verrons, cet ensemble de mots définit une grammaire (simple) pour les textes source.

2. Analyse du texte T et remplacement de chaque occurrence de s_i par un mot de code binaire w_i ce qui donne le texte encodé U.

La substitution des mots s_i par les w_i est appelée *remplacement d'ordre zéro.* Nous verrons plus loin d'où vient le "zéro". Evidemment le but est de choisir les s_i et les w_i de façon à maximiser le taux de compression.

Exemple. Supposons que les s_i soient choisis comme suit

$$\begin{aligned} s_1 &= 0 \\ s_2 &= 10 \\ s_3 &= 110 \\ s_4 &= 1110 \\ s_5 &= 1111, \end{aligned} \tag{13.1}$$

et que le texte T soit 111110111111011101111101110110. Alors, l'analyse de ce texte donne la séquence suivante

$$s_5 s_2 s_5 s_4 s_4 s_5 s_1 s_4 s_3.$$

La décomposition d'un texte en une suite de mots appartenant à une grammaire porte en anglais le nom de "parsing", et s'appelle en français l'"analyse grammaticale". Nous utiliserons la plupart du temps le terme "d'analyse".

On peut remarquer qu'il s'agit de la seule possibilité de générer le texte T au moyen des s_i que nous avons choisis. L'analyse grammaticale est non-ambiguë car les s_i forment un ensemble de mots sans préfixes ("prefix-free"). Nous avons aussi évité une certaine difficulté dans cet exemple, puisque le texte appartient à l'ensemble des textes générables à l'aide des mots s_i.

Supposons que nous associons les mots suivants w_i aux mots s_i

$$\begin{aligned} s_1 &\to 1111 \\ s_2 &\to 1110 \\ s_3 &\to 110 \\ s_4 &\to 10 \\ s_5 &\to 0. \end{aligned} \tag{13.2}$$

On obtient le texte codé $U = 01110010100111110110$, dont la longueur est de 20 bits, alors que $\ell(T) = 30$. Nous avons donc un taux de compression de $3/2$.

Est-ce que le texte T est recouvrable à partir de U ? Oui, car le code (13.2) est sans préfixes.

Nous reviendrons plus loin sur la condition "sans préfixes". Pour le moment, mettons en évidence une propriété un peu moins triviale des s_i.

Définition : SPP. On dit qu'une liste $s_1, s_2, \ldots, s_m$ de mots binaires vérifie la propriété d'analyse grammaticale forte (en anglais *strong parsing property* SPP), si, et seulement si tout texte binaire est une concaténation *unique*,

$$W = s_{i_1} \cdots s_{i_t} \nu, \tag{13.3}$$

de certains des s_i et d'un suffixe ν éventuellement vide, telle qu'aucun des s_i ne soit un préfixe de ν, et telle que $\ell(\nu) < \max_{1 \le i \le m} \ell(s_i)$.

Le mot ν est appelé la *feuille* de l'analyse de T au moyen des s_i. L'unicité signifie que si $T = s_{i_1} \cdots s_{i_t} \nu = s_{j_1} \cdots s_{j_r} \mu$ avec ν et μ n'ayant aucun des s_i comme préfixe et $\ell(\nu), \ell(\mu) < \max_{1 \le i \le m} \ell(s_i)$, alors $t = r$, $i_1 = j_1, \ldots, i_t = i_r$, et $\nu = \mu$.

De toute évidence, si les s_i vérifient la condition SPP, ils doivent tous être différents. De même, cette propriété est indépendante de l'ordre des s_i. Nous pouvons donc considérer la liste s_i comme un ensemble de mots binaires.

Exemple. Montrons que la liste (13.1) vérifie la propriété SPP. Si nous prenons un texte binaire quelconque T nous pouvons l'analyser de gauche à droite de la manière suivante : parcourir le texte jusqu'à rencontrer un zéro, ou jusqu'à avoir lu quatre 1 successifs, et sortir le mot s_i correspondant; arrêter la procédure lorsque la fin de T est rencontrée. Il est assez évident que cette procédure permet d'analyser un texte quelconque en produisant un suffixe $\nu = \lambda, 1, 11,$ où 111 (λ désigne le mot vide). L'analyse est toujours possible, et l'unicité est plausible, dans la mesure où notre algorithme n'est jamais confronté à une ambiguïté.

Nous verrons ci-dessous la formulation générale d'un théorème qui nous indiquera de manière précise sous quelles conditions cela est vrai.

13.2.2 Ensembles de mots ayant la propriété SPP

Quels sont les ensembles S de mots qui vérifient la propriété SPP ? Pourquoi est-ce important de le savoir ?

Nous allons voir que la très grande majorité des méthodes de compression réversibles commencent par décomposer le texte de départ en une suite de mots binaires, selon l'approche décrite dans notre exemple. Ces méthodes ne diffèrent donc pas de ce point de vue, mais seulement selon la manière dont elles encodent ensuite ces suites de mots en un texte de sortie. Pour une méthode de compression de données qui utilise un certain ensemble S, nous dirons qu'elle utilise S comme *alphabet de source*.

Les ensembles SPP constituent donc les alphabets de source utilisés par la plupart des progammes de compression de données.

Un premier choix assez fréquent est constitué par les alphabets $S = \{0,1\}^L$. Il s'agit d'alphabets de longueur fixe L. Par exemple, comme la plupart des ordinateurs utilisent une organisation des fichiers en bytes de longueur 8, le choix $L = 8$ est très courant car l'analyse grammaticale est alors immédiate. Une autre bonne raison pour choisir l'alphabet $S = \{0,1\}^8$ est que bien souvent les textes de départ n'exploitent pas complètement les 256 combinaisons possibles de 8 bits. Par exemple, la plupart des textes correspondant à des messages en langue anglaise utilisent moins de 60 caractères différents, bien que ceux-ci soient en général codés sur un byte. Lors de la compression de données il est alors possible de tirer profit de cela en réduisant d'emblée le nombre de bits nécessaires à 6.

Bien que les alphabets composés de blocs de longueur fixe soient les plus communs, nous ne voulons pas ici réduire d'emblée nos possibilités. C'est pourquoi nous allons caractériser l'ensemble des alphabets S vérifiant la propriété SPP au moyen d'une condition générale qui doit être satisfaite par tous ces alphabets.

Avant cela une remarque s'impose quant à l'impact de la feuille ν sur la longueur moyenne des textes encodés, et le calcul du taux de compression. En pratique la plupart des textes à analyser sont relativement longs; la mise en évidence et la représentation de la feuille ν demandera généralement plus de bits que la longueur $\ell(\nu)$. Cependant, l'effet global sur la longueur du texte encodé est généralement négligeable et nous allons dans la suite l'ignorer tout simplement.

Condition nécessaire et suffisante d'un alphabet SPP. Nous allons désigner dans la suite par *langage* engendré par un alphabet (noté aussi S^*) l'ensemble des textes T qui peuvent s'écrire sous la forme $T = s_{i_1} \cdots s_{i_t}$. Par ailleurs nous utiliserons également la notation $\overline{S^*}$ (fermeture du langage), l'ensemble des textes T qui peuvent s'écrire sous la forme $T = s_{i_1} \cdots s_{i_t} \nu$, avec ν n'ayant aucun des s_i comme préfixe et $\ell(\nu) < \max_{1 \leq i \leq m} \ell(s_i)$.

Rappelons qu'au chapitre 8 nous avons défini ce qu'est un code de source : il s'agit d'une règle qui associe à chaque symbole de la source s_i un mot m_i de n_i symboles de l'alphabet du code. Nous supposons ici que l'alphabet du code est binaire, et dans les notations nous identifions le symbole s_i et le mot binaire qui lui correspond et désignons par $\ell(s_i)$ le nombre de bits de ce mot.

Nous savons, depuis la fin du chapitre 8, qu'un code sans préfixes (c'est-à-dire tel qu'aucun mot ne soit un préfixe d'un autre), est à décodage unique. Cela veut dire que si nous construisons un texte T par concaténation d'une suite quelconque de mots de ce code, il est impossible de construire le même texte à partir d'une autre suite de mots de ce code. L'analyse de ce texte peut alors se faire de gauche à droite et de façon instantanée (ce que nous avons appelé le décodage instantané au chapitre 8).

Cette propriété ressemble fortement, mais pas exactement à la propriété SPP. En effet, pour le moment rien ne garantit qu'il soit possible de construire n'importe quel texte à l'aide d'un ensemble S qui vérifie la condition de préfixe.

Nous savons également que les longueurs des mots $\ell(s_i)$ doivent satisfaire l'inégalité de Kraft (cas binaire)

$$\sum_{s_i \in S} 2^{-\ell(s_i)} \leq 1. \tag{13.4}$$

Et nous savons aussi que réciproquement, si cette condition est vérifiée pour un ensemble spécifié de longueurs de mots, alors il existe forcément un code sans préfixes qui respecte ces longueur de mots.

Nous formulons maintenant la condition nécessaire et suffisante pour que $S = \{s_1, \ldots, s_m\}$ vérifie la propriété SPP.

Théorème SPP. *Si $s_1, \ldots, s_m$ est une liste de mots binaires, alors elle vérifie la propriété SPP si, et seulement si elle est sans préfixes et satisfait à **l'égalité** de Kraft*

$$\sum_{s_i \in S} 2^{-\ell(s_i)} = 1. \tag{13.5}$$

*En d'autres mots S forme un code sans préfixes **complet**.*

Esquisse de démonstration. Nous suggérons au lecteur courageux d'essayer de trouver la démonstration en suivant les étapes suivantes.

1. Montrer que la condition de préfixe est une condition nécessaire pour la propriété SPP.
   Sugestion : montrer que si la condition de préfixe n'est pas vérifiée alors la SPP ne peut pas l'être non plus.

2. Montrer que si le code S vérifie la condition de préfixe, alors pour un texte appartenant au langage S^*, l'analyse est non ambiguë (il n'existe qu'une seule suite $s_{i_1} \cdots s_{i_t} = W$).
   Suggestion : sinon, soit un $T = s_{i_1} \cdots s_{i_t} = s_{j_1} \cdots s_{j_r}$, et soit k le plus petit indice tel que $i_k \neq j_k$; on en déduit que $T' = s_{i_k} \cdots s_{i_t} = s_{j_k} \cdots s_{j_r}$, ce qui implique évidemment que les $\min\{\ell(s_{i_k}), \ell(s_{j_k})\}$ premiers bits de T' sont identiques, ce qui est contraire à l'hypothèse "sans préfixes".

3. Montrer que si S est tel que $\sum_{s_i \in S} 2^{-\ell(s_i)} < 1$, alors il ne peut pas satisfaire à la propriété SPP.
   Suggestion : montrer qu'on peut ajouter au code un mot supplémentaire s de longueur $\ell(s) \geq \max_{1 \leq i \leq m} \ell(s_i)$, sans détruire la condition de préfixe; en déduire que le texte $T = s$ ne peut pas être écrit sous la forme $s_{i_1} \cdots s_{i_t} \nu$, telle qu'aucun des s_i ne soit un préfixe de ν, et telle que $\ell(\nu) < \max_{1 \leq i \leq m} \ell(s_i)$.

4. Montrer que si S est sans préfixes et vérifie l'égalité de Kraft alors tout texte T de longueur $\ell(T) \geq \max_{1 \leq i \leq m} \ell(s_i)$ doit avoir un des mots s_i comme préfixe. En déduire que tout texte doit donc pouvoir s'écrire sous la forme $T = s_{i_1} \cdots s_{i_t} \nu$ avec $\ell(\nu) < \max_{1 \leq i \leq m} \ell(s_i)$ et n'ayant aucun des s_i comme préfixe.

5. Mettre les conditions (1)-(4) ensemble pour prouver la CNS de SPP.

Exercice. Montrer que tout ensemble S vérifiant la condition de préfixe peut être complété de manière à satisfaire la condition SPP.
Suggestion : revoir les démonstrations du chapitre 8 relatives à l'inégalité de Kraft.

13.2.3 Choix d'un schéma d'encodage

Supposons que l'alphabet de source S vérifiant la condition de SPP ait été choisi et que le texte de départ T ait été (au moins hypothétiquement) analysé pour former la suite de symboles de source $Z = s_{i_1} \cdots s_{i_n}$ et éventuellement une feuille ν non vide.[2]

Considérons maintenant le problème de décider comment choisir les mots binaires w_i à associer à chacun de nos mots s_i, c'est-à-dire le choix d'un schéma de remplacement $s_i \to w_i$, tel qu'on puisse recouvrir Z à partir du texte $U = w_{i_1} \cdots w_{i_n}$ et que la longueur $\ell(U)$ soit aussi petite que possible.

De toute évidence, la longueur de U ne sera fonction que des $\ell(w_i)$. D'autre part, depuis le chapitre 8 nous savons que nous ne perdons rien en exigeant que l'ensemble $W = \{w_1, \ldots, w_m\}$ soit sans préfixes, ce qui garantit le déchiffrement instantané. Le bon sens nous dit qu'il faut associer aux symboles s_i les plus fréquents les longueurs $\ell(w_i)$ les plus courtes, aux mots les plus rares les longueurs les plus grandes (sans les prendre inutilement trop longs).

Désignons par n_i le nombre de fois que le symbole s_i apparaît dans la suite $Z = s_{i_1} \cdots s_{i_n}$ et par $f_i = \frac{n_i}{n}$ sa fréquence relative. Nous pouvons considérer les f_i comme une loi de probabilité discrète correspondant à une expérience aléatoire ou nous mettons les $s_{i_1}, \ldots, s_{i_n}$ dans une urne et nous tirons un élément au hasard dans cette

[2]Comment doit-on choisir un bon alphabet de source S ? Peu de réflexion a été accordée à cette question. Rappelons que la solution "standard" est d'utiliser un alphabet $S = \{0, 1\}^8$.

urne (avec remise). D'autre part, si nous utilisons le schéma de remplacement $s_i \to w_i$, la longueur totale de U sera évidemment

$$\ell(U) = \sum_{j=1}^{n} \ell(w_{i_j}) = n \sum_{i=1}^{m} f_i \ell(w_i). \tag{13.6}$$

En conclusion, le problème auquel nous faisons face ici est en fait le même que celui que nous avons traité au chapitre 8. Il s'en suit que les méthodes que nous y avons discutées s'appliquent également ici. Nous allons brièvement rappeler ces différentes méthodes (en les traduisant dans le langage que nous utilisons ici).

Nous savons que la solution de notre problème est l'algorithme de Huffman[3]. Nous allons néanmoins décrire les méthodes de Shannon et de Fano, qui sont d'un intérêt historique et académique.

Méthode de Shannon. Soient les $s_1, \ldots, s_m$ et $f_1, \ldots, f_m$. Si nécessaire, on renumérote les s_i pour que $f_1 \geq f_2 \geq \cdots \geq f_m > 0$ et on fait disparaître les s_j qui n'apparaitraient pas du tout dans l'analyse de T. Définissons par $F_1 = 0$ et $F_k = \sum_{i=1}^{k-1} f_i, 2 \leq k \leq m$ et soit $\ell_k = \lceil \log_2 f_k^{-1} \rceil$.

Le schéma $s_i \to w_i$ de Shannon consiste alors à prendre pour w_i les ℓ_i premiers bits de l'expansion binaire du nombre F_i.

Expliquons ce que nous entendons par expansion binaire d'un nombre réel dans l'intervalle $[0, 1]$. Tout nombre compris dans cet intervalle peut en effet être écrit sous la forme d'une série

$$r = \lim_{n \to \infty} \sum_{j=1}^{n} a_j 2^{-j}, \tag{13.7}$$

où les $a_j \in \{0, 1\}$. Nous appelons n premiers bits de l'*expansion binaire* le mot $a_1 a_2 \cdots a_n$. On dit que le nombre réel est une fraction dyadique, s'il est possible de le représenter sous forme d'une expansion dont seulement un nombre fini de termes a_i sont différents de zéro.

En réalité, comme la série $\sum_{j=1}^{n} 2^{-j}$ tend vers 1, les fractions dyadiques non-nulles peuvent toutes s'écrire de deux façons : une expansion avec un nombre fini de termes non-nuls, disons $a_1 \cdots a_k 00 \cdots$ où $a_k = 1$, et une expansion $a_1 \cdots a_{k-1} 011 \cdots$. C'est pénible dans la mesure où dans la méthode de Shannon nous faisons référence à *La* (supposée seule) expansion binaire de F_i. Quid si F_i est une fraction dyadique ? La réponse est que nous allons prendre (ici et partout dans la suite) l'expansion qui est finie, c'est-à-dire celle qui se termine par une suite de 0. Nous demandons au lecteur de se convaincre que dans ces conditions, nous avons exclu les expansions binaires se terminant par une suite de longueur infinie de 1 (en d'autres mots, les seuls réels qui puissent se représenter de cette façon sont des fractions dyadiques, et pour celles-ci nous choisissons la représentation finie).

Nous savons que la méthode de Shannon donne lieu à des longueurs de mots compatibles avec l'inégalité de Kraft. Mais, est-ce que le code construit à partir des expansions binaires des F_i est effectivement sans préfixe ?

Reprenons le premier exemple utilisé dans ce chapitre. La suite de symboles de source était $s_5 s_2 s_5 s_4 s_4 s_5 s_1 s_4 s_3$: donc $f_1 = f_2 = f_3 = 1/9, f_4 = f_5 = 3/9$. Nous pouvons inverser l'ordre pour obtenir une séquence triée par ordre décroissant de fréquences relatives : $s'_i = s_{5-i+1}$. Donc, $\ell'_1 = \ell'_2 = 2$ et $\ell'_3 = \ell'_4 = \ell'_5 = 4$. Nous obtenons pour les w'_i

$$\begin{array}{rcl} F'_1 = & 0 & = (.00\ldots) \\ F'_2 = & 3/9 & = (.01\ldots) \\ F'_3 = & 6/9 & = (.1010\ldots) \\ F'_4 = & 7/9 & = (.1100\ldots) \\ F'_5 = & 8/9 & = (.1110\ldots). \end{array} \tag{13.8}$$

Donc la méthode de Shannon donne le code suivant

$$\begin{array}{l} s_5 = s'_1 \to 00 \\ s_4 = s'_2 \to 01 \\ s_3 = s'_3 \to 1010 \\ s_2 = s'_4 \to 1100 \\ s_1 = s'_5 \to 1110. \end{array} \tag{13.9}$$

[3] En 1951, Huffman, alors étudiant au M.I.T., découvrait cet algorithme en réponse à un problème posé par son professeur R. M. Fano, grand mathématicien qui enseignait à l'époque le cours de théorie de l'information au M.I.T. Ajoutons que le Professeur Fano avait judicieusement "oublié" de mentionner à ses étudiants que le problème était non résolu à l'époque.

On calcule que la longueur moyenne vaut $8/3$ et que le taux de compression par rapport au texte initial est de $5/4$ (il faut 24 bits pour représenter le texte U). Puisqu'avec le code sans préfixes que nous avons utilisé au début de la section 13.2.1 nous avions obtenu un taux de $3/2$, nous pouvons être deçus.

Revenons à notre question du départ. Comment justifier que le code construit à partir des expansions binaires des F_i est effectivement sans préfixe ? (Puisque la méthode porte le nom de "code de Shannon", nous pouvons raisonnablement supposer que la méthode est valide.)

Supposons que deux mots w_i et w_j du code de Shannon soient tels que $w_j = w_i\nu$ avec ν éventuellement vide. Nous pouvons toujours supposer que $j > i$, puisque les mots sont rangés par longueurs croissantes. On peut donc écrire que $F_i = (.w_i r_i)$ et $F_j = (.w_i r_j)$, ce qui implique que $F_j - F_i \leq F_j - (.w_i 00\ldots00) = (.00\ldots 0r_j)$; comme $(.00\ldots0r_j) < 2^{-\ell_i}$ (comme nous avons exclu les représentations infinies dyadiques) on en déduit que $F_j - F_i < 2^{-\ell_i}$.

D'autre part, comme $\ell_i = \lceil -\log f_i \rceil$ on a évidemment

$$-\log f_i \leq \ell_i < -\log f_i + 1 \tag{13.10}$$

ce qui implique que $f_i \geq 2^{-\ell_i}$. Or, on a évidemment aussi (par définition des F_i) que $F_j - F_i \geq f_i$ ce qui entraîne que $F_j - F_i \geq 2^{-\ell_i}$, ce qui est en contradiction avec $F_j - F_i < 2^{-\ell_i}$. □

Nous savons par ailleurs, depuis le chapitre 8, que la méthode de Shannon donne un code dont la longueur moyenne vérifie

$$\overline{\ell}_{\text{Shannon}} \leq H_m(f_1, \ldots, f_m) + 1. \tag{13.11}$$

Méthode de Fano. On se donne à nouveau les f_i et les s_i par ordre décroissant des f_i. La méthode de Fano est récursive : elle consiste à couper la suite s_i en deux, de telle manière que les sommes partielles $\sum_{i=1}^{k} f_i$ et $\sum_{i=k+1}^{m} f_i$ soient aussi proches que possibles. On associe un bit (0 ou 1) aux deux ensembles de symboles ainsi construits et on procède récursivement pour chacun de ceux-ci, jusqu'à obtenir des ensembles de taille 1.

Ce procédé produit de toute évidence un arbre binaire, équivalent au code de Fano. Ce code est par conséquent sans préfixe.

Voyons le code obtenu par cette méthode dans le cas de notre exemple.

$$\begin{array}{lllllll}
s_1 & : f_1 = & 3/9 & 0 & 0 & & \to 00 \\
s_2 & : f_2 = & 3/9 & 0 & 1 & & \to 01 \\
s_3 & : f_3 = & 1/9 & 1 & 0 & & \to 10 \\
s_4 & : f_4 = & 1/9 & 1 & 1 & 0 & \to 110 \\
s_5 & : f_5 = & 1/9 & 1 & 1 & 1 & \to 111.
\end{array} \tag{13.12}$$

On obtient une longueur moyenne de $20/9$. Il faut donc 20 bits pour coder le texte de neuf symboles de source $s_5 s_2 s_5 s_4 s_4 s_5 s_1 s_4 s_3$, soit un taux de compression de $3/2$.

Evidemment, nous aurions pu grouper les symboles différemment. On peut montrer que la méthode de Fano donne une longueur moyenne de code qui vérifie l'inégalité suivante :

$$\overline{\ell}_{\text{Fano}} \leq H_m(f_1, \ldots, f_m) + 2. \tag{13.13}$$

Nous invitons le lecteur à se poser les questions suivantes :

1. *Pour quelles valeurs des fréquences relatives la méthode de Fano donne-t-elle un code optimal, c'est-à-dire un code qui pour cet ensemble de fréquences relatives minimise $\overline{\ell}$?*

2. *Même question pour la méthode de Shannon.*

3. *Est-ce que la méthode de Fano est toujours au moins aussi bonne que la méthode de Shannon ?*

Méthode de Huffman. Rappelons que la méthode de Huffman se base également, comme celle de Fano, sur la construction d'un arbre de code (binaire dans le cas d'un alphabet de sortie binaire). La différence, essentielle, est que cette méthode construit l'arbre en partant des feuilles, plutôt que de la racine, et procède par regroupements successifs, plutôt que par divisions.

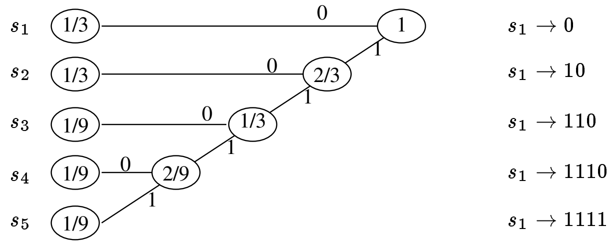


Figure 13.1. Codage de Huffman

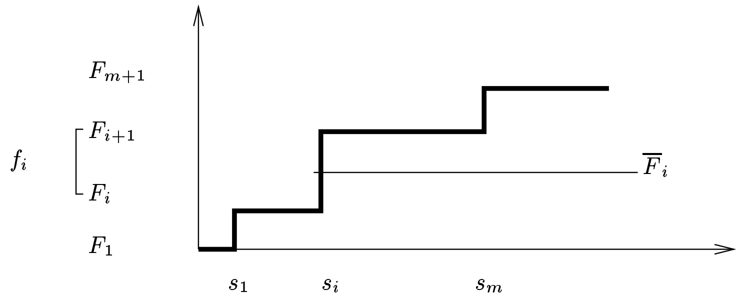


Figure 13.2. Diagramme de fréquences cumulatives

Nous savons que cette méthode produit le code optimal. Voyons, dans le cas de notre exemple ce que cela donne. La figure 13.1 illustre le procédé, qui regroupe toujours les deux feuilles les moins probables de l'arbre de code. On obtient le code que nous avions choisi comme exemple tout au début de ce chapitre. La longueur moyenne vaut $\overline{\ell} = 20/9$.

Remarques. Il est clair que les méthodes décrites ci-dessus s'appliquent quel que soit le choix de l'alphabet de source $s_1, \ldots, s_m$ vérifiant la SPP. Il doit également être clair, au moins pour le lecteur attentif, que le taux de compression obtenu avec l'une ou l'autre de ces méthodes est très fortement dépendant de ce choix. Pour s'en convaincre, il suffit de dire d'une part que si nous utilisons un alphabet de source binaire $s_1 = 0, s_2 = 1$, alors les trois méthodes de compression donneront un taux de compression de 1 (c'est-à-dire, en tenant compte des coûts cachés, en fait un taux inférieur à 1). Par ailleurs, nous savons qu'il est possible d'approcher de plus près la limite de Shannon en utilisant les extensions de la source, ce qui revient en fait à prendre des alphabets de taille plus grande.

Cependant, les méthodes que nous venons de discuter ne sont pas appropriées dans le cas où on veut adapter la taille des blocs à coder. Par exemple, dans l'algorithme de Huffman, si on veut passer de $S = \{0,1\}^L$ à $S = \{0,1\}^{L+1}$, on est obligé de reconstruire complètement l'arbre du code. De plus, comme la construction de l'arbre de code nécessite la connaissance des f_i, et le nombre des f_i à prendre en compte croît exponentiellement avec L, cette procédure est peu efficace pour des extensions d'ordre élevé.

13.2.4 Codage arithmétique

Dans cette section, nous allons voir qu'il est possible de construire des codes qui sont équivalents, en termes de taux de compression, à l'utilisation de l'algorithme de Huffman appliqué à des extensions d'ordre élevé, sans pour autant conduire à une explosion combinatoire de la complexité des algorithmes. Comme nous allons le voir, ces méthodes de codage "arithmétique" permettent de passer du code pour $S = \{0,1\}^L$ au code pour $S = \{0,1\}^{L+1}$, de façon très élégante et avec un effort de calcul faible.

Avant cela, voyons une variante de la méthode de Shannon, qui introduit une des idées principales du codage arithmétique, à savoir la représentation de mots (ou de textes) au moyen d'un sous-intervalle de $[0,1[$.

Shannon-Fano-Elias. Considérons le graphique de la figure 13.2. Comme nous n'allons pas nous servir dans notre raisonnement du fait que les mots sont triés par ordre décroissant de f_i nous n'avons pas non plus fait cette supposition sur la figure 13.2, qui représente simplement la relation entre les F_i et les s_i, dans un ordre quelconque : $F_{i+1} - F_i = f_i$. Désignons par $\overline{F}_i$ la valeur intermédiaire entre F_i et F_{i+1} :

$$\overline{F}_i = F_i + \frac{1}{2} f_i. \tag{13.14}$$

Puisque toutes les valeurs de f_i sont positives, les $\overline{F}_i$, tout comme les F_i sont toutes différentes (et même strictement croissantes). Connaissant la valeur de $\overline{F}_i$ on peut donc reconnaître le symbole s_i.

Mais, en général, les $\overline{F}_i$ sont des nombres réels, qui nécessitent éventuellement un nombre infini de bits pour être représentés. Cependant, comme ces nombres sont tous strictement différents, il y a certainement moyen de les arrondir en conservant la propriété de distinguabilité.

Notons par $\lfloor r \rfloor_\ell$ le nombre obtenu en gardant uniquement les ℓ premiers bits de l'expansion binaire du nombre réel $r \in [0, 1]$ (nous utiliserons aussi cette notation pour désigner le mot binaire composé de ces bits). Supposons que nous arrondissions $\overline{F}_i$ aux $\ell_i = \lceil -\log f_i \rceil + 1$ premiers bits. Cela donne lieu au code $s_i \to \lfloor F_i + \frac{1}{2} f_i \rfloor_{\ell_i}$. Notons que par rapport à la méthode de Shannon, non seulement nous ne demandons pas que les s_i soient triées par ordre décroissant de f_i, mais de plus nous gaspillons un bit par mot de code.

Montrons néanmoins que le code est sans préfixes. Nous laissons le soin au lecteur de vérifier que $\lfloor \overline{F}_i \rfloor_{\ell_i}$ appartient bien à l'intervalle $]F_i, F_{i+1}[$ (voir figure 13.2). Tous ces nombres sont donc bien distincts. Pour vérifier que le code est sans préfixes, nous allons considérer que chaque mot de code représente un intervalle de nombres réels défini par

$$[0.w_i, 0.w_i + 2^{-\ell_i}], \tag{13.15}$$

et noter que la condition "sans préfixes" est équivalente à la condition "intervalles disjoints". Comme nous savons que $0.w_i = \lfloor \overline{F}_i \rfloor_{\ell_i}$, le nombre $0.w_i$ est en fait dans la moitié inférieure de $[F_i, F_{i+1}]$, et donc $0.w_i + 2^{-\ell_i}$ doit aussi appartenir à l'intervalle $]F_i, F_{i+1}[$, car $2^{-\ell_i} < \frac{f_i}{2}$, c'est-à-dire strictement inférieur à la moitié de la largeur de l'intervalle $[F_i, F_{i+1}]$. $\square$

Etant donné les longueurs des mots de code, on trouve aisément que

$$\overline{\ell} < H_m(f_1, \ldots, f_m) + 2. \tag{13.16}$$

Suggestion : construire le code pour notre exemple en utilisant l'ordre initial des symboles s_i.

Codes arithmétiques. Malgré la sous-optimalité du codage par intervalles de la méthode de Shannon-Fano-Elias, c'est l'exploitation de cette idée qui donne lieu aux méthodes de compression de données d'ordre zéro les plus efficaces à ce jour, à savoir le codage arithmétique que nous allons décrire maintenant.

Pour une discussion détaillée de ce type de codes, nous conseillons la référence [HHJ97]. Ici nous reprenons la présentation de la référence [CT91], qui s'en tient aux idées essentielles.

Nous allons considérer que nous disposons d'un texte écrit dans un certain alphabet de source, et que nous souhaitons coder des suites de symboles de ce texte (éventuellement tout le texte) sur base des fréquences relatives déduites de l'étude statistique du texte. Les mêmes idées s'appliquent évidemment aussi, si nous voulons construire un code pour encoder des messages de longueur n quelconque d'une source dont nous avons modélisé les lois de probabilités, par exemple à l'aide d'un des modèles décrits au chapitre 8.

Désignons par $s^n = s_1 \cdots s_n$ une suite particulière de symboles source de longueur n. L'idée essentielle du codage arithmétique est le calcul efficace (récursif) des fréquences (ou probabilités) $f(s^n)$ et de la distribution cumulée $F(s^n)$. Ensuite, en utilisant les idées du codage de Shannon-Fano-Elias, nous pouvons utiliser un nombre réel dans l'intervalle $]F(s^n), F(s^n) + f(s^n)]$ décrit par un certain nombre de bits, comme mot de code pour s^n. Par exemple, en utilisant les $\lceil -\log f(s^n) \rceil$ premier bits de $F(s^n) + f(s^n)$ nous obtenons un code pour les blocs de longueur n. Le même raisonnement que ci-dessus permet de se convaincre que les mots de code ainsi obtenus sont tous différents.

Cependant, nous ne pouvons plus affirmer que le code est sans préfixes, car pour cela il aurait fallu arrondir à $\lceil -\log f(s^n) + 1 \rceil$ bits. Comme notre idée est ici de coder l'ensemble d'un texte de longueur n, il n'est cependant pas nécessaire d'utiliser un code sans préfixes. Du moment que le décodeur connaît la valeur de n, il suffit en effet que les mots de code associés à tous les textes de longueur n soient différents, pour que le code soit déchiffrable.

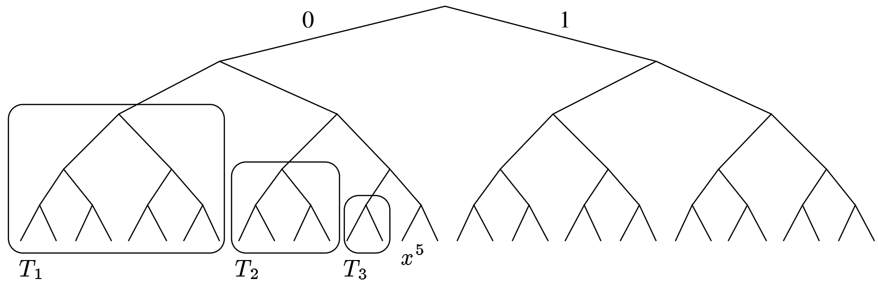


Figure 13.3. Arbre des textes source ($n = 5$) pour le codage arithmétique

Nous avons donc deux options : (i) utiliser un code de longueur moyenne légèrement plus faible, et indiquer au décodeur la longueur du texte source encodé (correspondant à un seul bloc); (ii) utiliser un code sans préfixes comme dans la méthode de Shannon-Fano-Elias, avec des longueurs légèrement plus grandes, et adopter une convention pour les longueurs de blocs source encodés. Notons que généralement, c'est la première option qui prévaut : on code à l'aide du codage arithmétique tout le texte à transmettre, et on transmet au décodeur la longueur de ce texte.

Dans la version simplifiée du codage arithmétique que nous allons décrire, la longueur du texte n est donc fixée a priori et connue par l'encodeur et le décodeur. Nous allons de plus nous placer dans le cas particulier où l'alphabet de source est binaire ($s_i \in \{0, 1\}$), afin de simplifier notre discussion. Nous supposons que nous disposons d'une procédure simple qui permet de calculer $f(s_1, \ldots, s_n)$ pour toute suite de n symboles de source, et nous allons considérer les mots binaires de longueur n triés par ordre lexicographique : si x^n et y^n sont deux chaînes différentes, alors $x^n > y^n$ si $x_i = 1$ et $y_i = 0$ pour le premier i tel que $x_i \neq y_i$. De manière équivalente, $x > y$ si $\sum_i x_i 2^{-i} > \sum_i y_i 2^{-i}$, c'est-à-dire si les nombres binaires vérifient $0.x^n > 0.y^n$.

Nous pouvons organiser les textes de n symboles de source sous la forme d'un arbre binaire complet de profondeur n, dont les feuilles correspondent aux chaînes de longueur n. Un tel arbre est représenté à la figure 13.3. Dans cet arbre, l'ordre $x^5 > y^5$ correspond au fait que x^5 est à la droite de y^5 à la même profondeur dans l'arbre. Il est clair que le passage de n à $n+1$ consistera à étendre l'arbre d'un niveau.

Si nous attachons à cet arbre les fréquences $f(x^n)$ correspondant aux feuilles, nous voyons que les valeurs $F(x^n)$ peuvent être aisément obtenues à partir des probabilités des noeuds de l'arbre. En effet, de la discussion du codage Shannon-Fano-Elias, il apparaît que nous devons, pour calculer $F(x^n)$, trouver les $f(y^n)$ de tous les mots $y^n < x^n$, et sommer ces valeurs pour calculer $F(x^n)$. Ensuite, le mot de code est obtenu en utilisant les $\lceil -\log f(x^n) \rceil$ premier bits de $F(x^n) + f(x^n)$. Cela revient, évidemment à calculer les $\lceil -\log f(x^n) \rceil$ premiers bits de $F'(x^n) = \sum_{y^n \leq x^n} f(y^n)$.

En considérant la figure 13.3, nous voyons qu'il s'agit de calculer la somme des probabilités des feuilles qui sont à la gauche de x^n, ce qui équivaut aussi à la somme des probabilités des arbres qui se situent à la gauche de x^n.

Soit $T_{x_1 x_2 \cdots x_{k-1} 0}$ un sous-arbre pointé par le chemin $x_1 x_2 \cdots x_{k-1} 0$, la fréquence de ce sous-arbre est

$$f(T_{x_1 x_2 \cdots x_{k-1} 0}) = \sum_{y_{k+1} \cdots y_n} f(x_1 x_2 \cdots x_{k-1} 0 y_{k+1} \cdots y_n) \tag{13.17}$$

$$= f(x_1 x_2 \cdots x_{k-1} 0), \tag{13.18}$$

qui peut être calculée facilement (par hypothèse). Notons que quelle que soit la valeur de $k > 0$, l'arbre $T_{x_1 x_2 \cdots x_{k-1} 0}$ est situé à gauche de x^n si, et seulement si $x_k = 1$.

On peut donc écrire $F(x^n)$ sous la forme

$$F(x^n) = \sum_{y^n < x^n} f(y^n) \tag{13.19}$$

$$= \sum_{T \,:\, T \text{est à gauche de } x^n} f(T) \tag{13.20}$$

$$= \sum_{k \,:\, x_k = 1} f(x_1 x_2 \cdots x_{k-1} 0). \tag{13.21}$$

Exemple. Supposons que nous ayons à faire à un texte binaire, tel que $f(1) = \theta$ et $f(0) = 1 - \theta$. Si nous faisons l'hypothèse d'ordre zéro, qui consiste à supposer que les symboles successifs sont indépendants, alors $f(s_1, \ldots, s_n) = f(s_1) \cdots f(s_n)$.

Considérons l'arbre de la figure 13.3 et essayons de trouver la valeur de $F(01110)$ (x^5 sur la figure). On trouve que

$$\begin{aligned} F(01110) &= f(T_1) + f(T_2) + f(T_3) \\ &= f(00) + f(010) + f(0110) \\ &= f(0)f(0) + f(0)f(1)f(0) + f(0)f(1)f(1)f(0) \\ &= f(0)(1 + f(1)(1 + f(1)))f(0) \\ &= (1-\theta)(1+\theta(1+\theta))(1-\theta), \end{aligned}$$

où nous avons mis en évidence (deux dernières lignes) la possibilité de calcul récursif des $f(\cdot)$, ce qui limite le nombre de multiplications/additions en ne recalculant pas à chaque fois les facteurs communs entre les différents f_i, et les termes communs aux différents $f(T_i)$.

Encodage. Les symboles de la source peuvent être traités séquentiellement :

1. Soit x^k le préfixe déja lu après k étapes, $f(x^k)$ la fréquence relative correspondante, et $F(x^k)$ la fréquence cumulée correspondante, et soit u_k le noeud courant dans l'arbre.
2. k est initialisé à zéro; x^0 est initialisé à la chaîne vide; la fréquence $f(x^0)$ à 1; $F(x^0)$ est initialisée à zéro, le noeud courant u_k est initialisé à la racine de l'arbre.
3. Soit b le $(k+1)$-ème bit lu.
   - si $b = 1$, $F(x^{k+1}) = F(x^k) + f(x^k 0)$.
   - si $b = 0$, $F(x^{k+1}) = F(x^k)$.
4. le préfixe est mis à jour $x^{k+1} = x^k b$, la fréquence $f(x^{k+1})$ est également mise à jour (cf. remarques ci-dessous), et le noeud courant u_{k+1} est mis à jour en descendant dans l'arbre en suivant la branche b émanant du noeud u_k.
5. si $k < n$, on poursuit la procédure, sinon on renvoie les valeurs $F(x^n)$ et $f(x^n)$.
6. à partir de $F(x^n)$ et $f(x^n)$ on construit le mot de code $\lfloor F(x^n) + f(x^n) \rfloor_{\lceil \log f(x^n) \rceil}$.

Calcul des $f(x^k)$. Nous avons vu que pour le codage d'un texte s^n, le parcours de l'arbre nécessite au plus deux calculs de fréquences relatives $f(x^k)$ par niveau de l'arbre, c'est-à-dire par symbole source. En fait, si $s_k = 0$ un seul calcul suffit, et si $s_k = 1$ deux calculs sont nécessaires. Cela veut dire que si nous sommes capables de calculer efficacement ces fréquences relatives, l'algorithme est essentiellement linéaire en fonction de la taille du texte, et cela indépendamment du modèle probabiliste utilisé. Voyons s'il est possible de faire ces calculs efficacement.

Le cas particulier où les symboles successifs sont considérés comme indépendants est particulièrement facile, puisque dans ce cas la mise à jour se fait au moyen d'une seule multiplication : $f(x^{k+1}) = f(x^k)f(b)$, et la probabilité de l'arbre est obtenue aussi par $f(x^k 0) = f(x^k)f(0)$.

De même, si nous modélisons le texte à l'aide d'une chaîne de Markov, la mise à jour est immédiate.

Dans le cas général, on peut utiliser la formule de récurrence suivante :

$$f(s^k b) = f(b|s^k)f(s^k),$$

où $f(x^k)$ est le résultat de l'étape k, et $f(b|s^k)$ est obtenu directement à partir du modèle probabiliste.

Suggestion : expliciter les formules dans le cas de la source de Markov.

Alphabet de source m-aire. Il est également assez facile de voir comment cet algorithme doit être modifié si nous avons une source qui utilise un alphabet m-aire avec m quelconque. L'arbre devient un arbre m-aire complet, et à chaque étape il faut sommer les probabilités des $m' < m$ sous-arbres laissés à gauche.

Décodage. Le décodage est également immédiat. On se déplace dans l'arbre en mettant à jour la fréquence cumulée, la fréquence et le préfixe comme ci-dessus. La différence vient du critère de décision utilisé pour se déplacer. Au lieu d'utiliser le bit suivant du texte initial (qui est inconnu), on se sert de la valeur du nombre réel qui correspond au mot de code reçu pour décider de la direction à prendre.

On exploite l'arbre comme un arbre de décision : à la racine on compare le nombre reçu à la valeur de $f(0)$: si celle-ci est plus petite que le nombre reçu, alors le sous-arbre correspondant au mot encodé doit se trouver à la droite du sous-arbre pointé par 0, et par conséquent $x_1 = 1$. En poursuivant ce procédé en descendant dans l'arbre on peut décoder la séquence, bit par bit.

A nouveau, la technique de décodage peut être facilement adaptée au cas d'un alphabet de source q-aire. Cette fois, il faut laisser à gauche tous les sous-arbres tels que leur fréquence **cumulée** soit inférieure au mot de code : le chemin qui est suivi, correspond au sous-arbre le plus à gauche tel que sa fréquence **cumulée** avec celle de ses voisins de gauche soit supérieure au mot de code reçu.

Discussion. Si on fait l'hypothèse d'ordre zéro, il est clair que la longueur moyenne du code arithmétique se situe dans une fourchette de $\frac{1}{n}$ bits de la limite d'entropie $H_m(f_1, \ldots, f_m)$.

La méthode permet de compresser et de décompresser des textes de longueur quelconque de manière séquentielle, à condition que le décodeur soit informé de la longueur du texte qu'il va recevoir. Si nous supposons que les f_i sont fixes d'un texte à l'autre, il ne sera nécessaire de fournir cette information au décodeur qu'une seule fois (elle peut donc être incorporée en "dur" dans l'algorithme de décodage). Si nous supposons que les f_i sont redéterminées pour chaque texte source, il faut également en transmettre les valeurs au décodeur. Comme il s'agit de nombres réels, nous serons obligés la plupart du temps de les arrondir, typiquement en n'utilisant que les $\lceil -\log f_i \rceil$ premiers bits. Il y a donc un coût caché au total de $\sum_{i=1}^{m} \lceil -\log f_i \rceil$ bits, qui devient négligeable si le texte est suffisamment long.

En ce qui concerne l'implémentation, nous attirons l'attention sur le fait que cette méthode fait appel à des multiplications/additions de nombres réels, qui doivent être effectuées avec une précision suffisamment grande pour éviter des erreurs de codage/décodage. Nous renvoyons le lecteur à la référence [HHJ97] pour une discussion de ces questions.

D'autre part, il est important de remarquer que ce type de code exploite un modèle de source (ici d'ordre zéro) : il n'est pas nécessaire pour autant que la source se comporte comme une source d'ordre zéro. Il suffit que les fréquences relatives des symboles utilisées correspondent effectivement aux probabilités d'émission de ces symboles (ou aux fréquences relatives effectivement présentes dans le texte encodé, si celui-ci est suffisamment long), pour que le taux de compression soit atteint. Evidemment, si nous utilisons un modèle d'ordre 0 alors que la source émet des symboles successifs dépendants, cela voudra dire que l'entropie par symbole de celle-ci est inférieure à $H_m(f_1, \ldots, f_m)$, et qu'il serait possible d'atteindre des taux de compression supérieurs en exploitant de l'information d'ordre supérieur.

D'ailleurs, dans la description de notre algorithme d'encodage/décodage, nous n'avons pas fait d'hypothèse explicite sur la manière dont les $f(s^n)$ sont déterminées. Et nous avons montré qu'il est parfaitement possible d'utiliser le codage arithmétique avec des modèles de source d'ordre supérieur. La section suivante s'intéresse à ce problème.

Enfin, remarquons que si les f_i utilisées pour l'encodage ne sont pas exactement égales aux probabilités p_i des symboles, mais néanmoins proches, alors le code reste bon. On montre en effet aisément que la longueur moyenne sous une loi de probabilité p_i d'un code qui assigne des longueurs $\lceil -\log f_i \rceil$ vérifie

$$H_m(p_1, \ldots, p_m) + D(p\|f) \leq \overline{\ell} \leq H_m(p_1, \ldots, p_m) + D(p\|f) + 1. \tag{13.22}$$

On constate que par rapport à la situation où $f_i = p_i$, les limites sont décalées vers le haut, d'autant plus que $D(p\|f)$ est grande. Il est clair que si $p_i \approx f_i$, $D(p\|f)$ sera petite et le code restera efficace, et cela justifie donc aussi l'utilisation de fréquences relatives "arrondies" comme suggéré ci-dessus.

Longueur du texte et caractère on-line. Nous avons indiqué ci-dessus que le codage arithmétique permettait de traiter de façon séquentielle lors du codage et du décodage les symboles source. Est-ce que la méthode est pour autant on-line ? En d'autres mots, est-ce que la transmission du texte codé peut commencer avant d'avoir parcouru celui-ci complètement ? Combien de symboles du texte source doivent être connus pour que la transmission puisse commencer ?

Nous allons voir que ces questions sont intimement liées à la nécessité de transmettre la longueur du texte source au décodeur. En effet, pour que la décompression reproduise le texte source il faut nécessairement lui transmettre l'information relative à la longueur de celui-ci. Nous allons discuter deux possibilités, dont une seule préserve le caractère on-line de la méthode.

La première solution consiste à transmettre pour chaque texte encodé , en préambule à la communication, la longueur du texte. Il y a un coût caché d'au moins $\log n$ bits, pour un texte de longueur n. Ce qui est plus gênant, c'est qu'il est nécessaire lors du codage de connaître cette longueur avant de commencer la transmission, ce qui implique la plupart du temps le stockage complet du texte encodé avant de pouvoir le transmettre.

Un autre solution consiste à ajouter un symbole supplémentaire (disons ".") à l'alphabet de source qui est réservé pour marquer la fin du texte. Il faut évidemment lui attribuer une probabilité non-nulle dans le modèle de source, ce qui conduit in fine également à un coût caché. Le décodeur poursuivra alors sa descente dans l'arbre des messages de source jusqu'à avoir décodé le symbole ".".

Cette façon de faire permet en plus de restituer au codage/décodage le caractère on-line, et donc de réduire les délais de transmission. En effet, intuitivement il est assez évident que les bits de poids le plus fort dans la représentation binaire du mot de code ne dépendent que des premiers symboles du texte. Cela veut donc dire qu'après un nombre fini (généralement petit, mais pas nécessairement égal à un) de symboles source, une première partie du mot de code est établie (un ou plusieurs bits), ensuite après quelques symboles supplémentaires quelques bits supplémentaires peuvent être transmis, et ainsi de suite. La méthode d'encodage est donc quasi on-line, et cela indépendamment du modèle probabiliste utilisé. Symétriquement, à la réception quelques bits de poids le plus fort suffisent pour tester les probabilités des sous-arbres, et le décodage peut donc se faire pratiquement au même rythme que l'encodage.

Suggestion : réfléchir à un moyen pratique de transmettre la longueur n du texte comme préambule au texte chiffré.

13.3 MODELES D'ORDRE SUPERIEUR

Pour bien comprendre les méthodes de compression de données statistiques, il est utile de se représenter le dispositif de codage comme composé de deux boîtes autonomes : l'*algorithme de codage* et le *modèle statistique*. Le modèle (ou processeur statistique) passe à l'algorithme de codage de l'information concernant la nature statistique du texte source. L'algorithme de codage utilise cette information pour encoder les textes efficacement.

La section précédente s'intéressait essentiellement à différents algorithmes de codage, et le modèle statistique utilisé était très rudimentaire; nous avons supposé qu'une étude statistique du texte nous avait permis d'évaluer les fréquences relatives $f_1, \ldots, f_m$ des m symboles de l'alphabet de source qui sont alors utilisées par l'encodeur. Dans cette section et dans la suivante nous allons investiguer l'utilisation de modèles plus sophistiqués. A la section 13.4 nous verrons les méthodes adaptatives, c'est-à-dire capables de traiter efficacement des sources non-stationnaires. Ici, nous nous focalisons sur les méthodes exploitant des modèles d'ordre supérieur, permettant de tenir compte des dépendances entre symboles successifs.

Tous ces modèles peuvent être exploités par les méthodes de codage (Huffman, arithmétique, Shannon-Fano-Elias, Shannon, Fano) décrites dans la section précédente. Tant que l'encodeur connaît les fréquences relatives $(f_1, \ldots, f_m)$ *courantes*, il sait comment encoder le texte source. L'encodeur ne doit donc pas, en principe, être ajusté pour exploiter tel ou tel modèle statistique. Bien sûr, en pratique il est souvent possible d'augmenter l'efficacité computationnelle en modifiant l'algorithme de codage pour mieux coller au modèle de source exploité. Nous allons donc présenter les modèles d'ordre supérieur (et plus loin les modèles adaptatifs) en alliance avec les codeurs de Huffman ou arithmétique, et faire comme s'il y avait des choses telles que "Codage de Huffman d'ordre k", ou "Codage arithmétique adaptatif", parce qu'il est pratique de faire ainsi et que nous pensons qu'il est plus concret d'étudier comment l'ensemble "modèle + encodeur" fonctionne. Mais nous voulons insister ici sur le fait que "l'ordre supérieur" est une qualité du modèle et non de l'algorithme de codage.

De quoi s'agit-il ? Soient un alphabet de source $S = \{s_1, \ldots, s_m\}$, et un entier $k \geq 0$. Dans un modèle d'ordre k, nous supposons que les fréquences relatives $f(i_1, \ldots, i_{k+1})$ des mots $s_{i_1} \cdots s_{i_{k+1}}$ parmi tous les mots de source de longueur $k+1$ sont données. Si $k = 0, 1, 2$ nous écrivons $f(i) = f_i$ (les fréquences relatives des symboles utilisées précédemment), $f(i,j) = f_{i,j}$ (digrammes) et $f(i,j,k) = f_{i,j,k}$ (trigrammes).

Notons que, puisque nous supposons pour le moment que le comportement modélisé est stationnaire, cela implique que les fréquences marginales des symboles obtenues à partir des multigrammes, telles que

$$f_{i_l} = \sum_{i_1 \leq m} \cdots \left[\sum_{i_l \leq m} \right] \cdots \sum_{i_{k+1} \leq m} f(i_1, \ldots, i_{k+1}),$$

sont indépendantes de la position dans le temps l du symbole marginalisé (la mise entre crochets $[\cdot]$ dans la formule signifie que la sommation sur i_l n'est pas effectuée).

Par exemple, dans le cas des digrammes cette propriété se traduit par le fait que $\sum_{i=1}^{m} f_{i,j} = \sum_{i=1}^{m} f_{j,i} = f_j$.

13.3.1 Codage de Huffman d'ordre supérieur

Nous avons deux options pour appliquer l'algorithme de Huffman. La première, déjà amplement discutée dans les chapitres précédents, consiste à coder la $k+1$-ème extension[4] de la source (S^{k+1}) en nous servant des fréquences relatives $f(i_1, \ldots, i_{k+1})$ des m^{k+1} symboles de cette source. Ce schéma donne évidemment lieu à une longueur moyenne $\overline{\ell}_{(k)}$ par symbole de la source S coincée entre les deux limites $\frac{H(S^{k+1})}{k+1}$ et $\frac{H(S^{k+1})+1}{k+1}$.

Mais nous allons utiliser une autre idée, plus souple en principe et susceptible de mieux exploiter les dépendances entre symboles successifs. Au lieu d'utiliser une table de code comprenant m^{k+1} lignes, nous allons utiliser m^k tables comprenant m mots de code[5]. Chaque table est adressée par un préfixe de longueur k correspondant aux k symboles qui précèdent celui qui doit être codé (ou décodé) à un instant particulier. Soit s_j le symbole courant, et soient $s_{i_1}, \cdots, s_{i_k}$ les k symboles précédents : nous appelons cette suite, le *contexte* du symbole s_j.

On peut alors, à partir des multigrammes $f(i_1, \ldots, i_{k+1})$, calculer la probabilité (ou fréquence) conditionnelle

$$f(j|i_1, \cdots, i_k) = \frac{f(i_1, \ldots, i_k, j)}{f(i_1, \ldots, i_k)}, \tag{13.23}$$

de rencontrer le symbole s_j connaissant le contexte. On peut donc appliquer l'algorithme de Huffman en utilisant les m valeurs stockées $f(j|i_1, \cdots, i_k)$. Nous appellerons *codage de Huffman d'ordre supérieur* ce schéma qui utilise les probabilités conditionnelles. Nous verrons ci-dessous quels arguments peuvent plaider en sa faveur, par rapport au premier schéma, que nous appellerons dans la suite *codage de Huffman de l'extension d'ordre* $k+1$.

De toute évidence le schéma décrit ci-dessus ne fonctionne qu'à partir du $(k+1)$-ème symbole de source. Comment traiter les k premier symboles ? On peut imaginer plusieurs solutions, comme par exemple

1. utilisation d'un préfixe artificiel $s_{-k+1} \cdots s_0$, qui précède le début du texte.
2. codage d'ordre $k(i)$, pour le i-ème symbole, avec $k(i) = \min\{i-1, k\}$.
3. utilisation d'un codage d'ordre 0 pour les k premiers symboles.

Dans les deux derniers cas, on peut déduire les fréquences relatives nécessaires au codage à partir des multigrammes. D'autre part, comme on peut supposer que le texte est suffisamment long pour que les effets de bords soient négligeables du point de vue du taux de compression obtenu, on peut se contenter d'utiliser le schéma le plus simple à implémenter et/ou celui qui conduit au coût caché le plus faible.

Nous allons supposer qu'on utilise le codage de Huffman d'ordre zéro pour les k premiers symboles, et qu'à partir du $(k+1)$-ème on utilise le codage d'ordre k. De plus, nous allons négliger dans nos calculs les effets de bords, en ne calculant que les taux de compression obtenus à partir du $(k+1)$-ème symbole de source.

Quid, en ce qui concerne le décodage ? Le décodeur reçoit, conjointement au texte codé les m^{k+1} multigrammes, ce qui lui permet de construire les $m^k + 1$ tables de code : 1 pour les k premier symboles; m^k correspondant aux m^k préfixes de longueur k possibles. A partir de là, il peut décoder séquentiellement les symboles de source, selon un schéma qui devrait maintenant être évident.

[4] Attirons l'attention sur le fait que le modèle de l'extension d'ordre $K+1$ est, selon notre terminologie, un modèle d'ordre k.

[5] Nous disons "table" sans pour autant supposer que les données sont organisées en mémoire de façon linéaire. On peut se convaincre que les deux schémas ne diffèrent ni en termes de volume de stockage, ni en termes de temps d'accès.

Intéressons nous aux taux de compression atteignables par cette méthode. La longueur moyenne d'un mot de code est obtenue par

$$\overline{\ell}^{(k)} = \sum_{1\leq i_1,\ldots,i_k\leq m} f(i_1,\ldots,i_k)\sum_{j=1}^{m} f(j|i_1,\ldots,i_k)\ell(i_1,\ldots,i_k,j) \tag{13.24}$$

$$= \sum_{1\leq i_1,\ldots,i_{k+1}\leq m} f(i_1,\ldots,i_{k+1})\ell(i_1,\ldots,i_{k+1}), \tag{13.25}$$

où les $\ell(i_1,\ldots,i_{k+1})$ sont obtenues par les m^k codes de Huffman conditionnels. Comme ces codes sont optimaux isolément, il est évident que le schéma global d'ordre k est également optimal. Cela veut dire que quel que soit le code utilisé qui se sert d'une stratégie conditionnelle telle que le schéma d'ordre k de Huffman, la longueur moyenne obtenue ne peut pas être inférieure à celle obtenue par le codage de Huffman.

Si nous désignons, comme ci-dessus, par $\overline{\ell}_{(k)}$ la longueur moyenne par symbole de source obtenue par le codage de la source étendue S^{k+1}, on peut en particulier se demander si $\overline{\ell}^{(k)}$ est toujours inférieure ou égale à $\overline{\ell}_{(k)}$. (Remarquons que pour $k=0$ ces deux grandeurs sont évidemment identiques.) Il est assez surprenant de constater que la réponse est : pas toujours (cf exercices, pour des contre-exemples).

Une autre question intéressante : est-ce que la suite $\overline{\ell}^{(k)}$ est une suite décroissante ? On peut au moins le conjecturer, mais la preuve n'en a pas été faite, à notre connaissance.

Si nous considérons que nous avons affaire à une source stationnaire, désignons par $H(S^k)$ l'entropie de la source étendue d'ordre k, et par $H(S_{k+1}|S^k)$ l'entropie conditionnelle moyenne du $k+1$ symbole de S lorsque les k précédents sont connus, c'est-à-dire l'entropie moyenne des m^k sources de notre dispositif de code. Alors, on a évidemment $H(S_{k+1}|S^k) = H(S^{k+1}) - H(S^k)$. D'autre part, comme pour chacune des m^k sources conditionnelles l'inégalité de Shannon est vérifiée, il est immédiat que le meilleur code d'ordre k doit satisfaire l'inégalité

$$H(S_{k+1}|S^k) \leq \overline{\ell}^{(k)} < H(S_{k+1}|S^k) + 1. \tag{13.26}$$

Rappelons que la suite $H(S_{k+1}|S^k)$ est une suite décroissante pour k croissant. Par conséquent, la longueur moyenne du meilleur code d'ordre k atteignable est coincée entre deux limites qui décroissent. Mais cela ne nous permet pas d'affirmer que les longueurs moyennes réalisées par les codes de Huffman successifs sont elles-mêmes décroissantes.

Par ailleurs, en guise de comparaison des deux méthodes (codage d'ordre k et codage de l'extension d'ordre $k+1$) nous pouvons énoncer (démonstration immédiate) les relations suivantes entre les différentes entropies :

$$H(S_{k+2}|S^{k+1}) \leq H(S_{k+1}|S^k) \leq \frac{H(S^{k+1})}{k+1} \leq \frac{H(S^k)}{k} \leq \cdots \leq H(S). \tag{13.27}$$

Il est clair que, même si ces grandeurs ne fournissent qu'une approximation des longueurs moyennes des messages codés obtenues à l'aide des différents codes, elles seront néanmoins utiles pour décider (sans nécessiter la construction des codes) si cela vaut la peine de recourir, pour un texte donné, au codage d'ordre supérieur.

Suggestion : réfléchir à un moyen efficace en temps d'accès pour stocker le modèle probabiliste d'ordre supérieur utilisé par le codage de Huffman.

Exercice. Soit $S = \{s_1, s_2, s_3\}$ et soient les fréquences de digrammes de S données par la matrice

$$[f_{i,j}] = \begin{bmatrix} 0.70 & 0.05 & 0.05 \\ 0.02 & 0.04 & 0.04 \\ 0.08 & 0.01 & 0.01 \end{bmatrix}.$$

Construire les codes de Huffman pour S et S^2 et le code de Huffman d'ordre 1. Calculer et comparer les longueurs moyennes (par symbole de S) de ces trois codes.[6]

[6]Solution exercice page 227 : $\overline{\ell}^{(0)}(S) = 1.2$; $\overline{\ell}^{(1)}(S) = 1.18$; $\overline{\ell}_{(1)}(S) = 0.91$.

13.3.2 Codage arithmétique d'ordre supérieur

Comment tirer profit de la connaissance des multigrammes en codage arithmétique ?

Une des réponses possibles est la suivante : on utilise les multigrammes pour calculer les probabilités (ou fréquences) conditionnelles $p(s_p|s_{p-1},\ldots,s_{p-k})$ (effets de bords traités selon la méthode 2, citée auparavant). A partir de cette information, il est évidemment possible de calculer les probabilités associées à n'importe quelle suite de symboles source correspondant aux noeuds de l'arbre des symboles source. En effet, on peut utiliser la formule de récurrence suivante :

$$p(s_1,\ldots,s_n) = p(s_1,\ldots,s_{n-1})p(s_n|s_{n-1},\ldots,s_{n-k}). \tag{13.28}$$

Si on utilise ce type de schéma, on peut se convaincre que, pour coder des messages de longueur N, on obtient une borne supérieure (sur le nombre de bits par symbole de source) qui peut s'écrire sous la forme

$$\overline{\ell} \leq \frac{1 + H(S) + H(S_2|S^1) + \cdots + H(S_k|S^{k+1}) + (N-k)H(S_{k+1}|S^k)}{N}, \tag{13.29}$$

ce qui est proche de $H(S_{k+1}|S^k)$ pour $N \gg k$.

13.4 METHODES ADAPTATIVES

Nous restons toujours dans la cadre de la compression de données réversible. La nouveauté est que nous laissons tomber l'hypothèse qu'une analyse statistique du texte source soit intéressante, c'est-à-dire l'hypothèse que ce texte ressemble à ce qu'une source stationnaire pourrait produire. On peut citer de nombreux exemples de textes qui ne possèdent pas la propriété de stationnarité. Par exemple les fichiers postscript se caractérisent généralement par un contenu dont la nature varie fortement d'un point à l'autre : au début on trouve généralement un en-tête décrivant les macros postscript et les fontes utilisées, ensuite on trouve un entrelacement de parties qui correspondent à du texte en langue naturelle encodé en postscript par le logiciel de traitement de texte utilisé et des images (parfois bitmap) dont le code postscript est produit par d'autres logiciels. Intuitivement, il semble assez évident que pour ce type de textes il est intéressant de changer la stratégie de codage au fur et à mesure du parcours du texte. Reste à voir comment cela peut se faire de manière efficace et effective du point de vue taux de compression.

Cette section et la suivante s'intéressent aux méthodes dites de codage "universel" parce que leurs performances sont peu sensibles à la nature du texte compressé.

L'idée dans cette section est principalement d'exploiter les techniques déjà discutées ci-dessus (Huffman, arithmétique) en utilisant des comptages des fréquences relatives des symboles (ou multigrammes, dans le cas des méthodes adaptatives d'ordre supérieur) qui adaptent les fréquences aux propriétés locales du texte. Un des avantages supplémentaires de ces méthodes est qu'elles suppriment la nécessité de transmettre la table de code (celle-ci étant reconstituée adaptativement lors du décodage), et diminuent donc un certain type de coût caché. Comme dans la section précédente, nous allons nous focaliser sur la méthode de Huffman adaptative, nous ne ferons que quelques remarques en ce qui concerne les méthodes arithmétiques.

13.4.1 Codage de Huffman adaptatif

Dans cette approche, on maintient, au fur et à mesure du parcours du texte source, un comptage du nombre de symboles de chaque type recontré jusqu'alors, et c'est ce comptage qui est exploité pour coder le symbole suivant. Lorsque celui-ci a été encodé, le compteur correspondant est mis à jour et on passe au symbole suivant. Le décodeur procède exactement de la même manière, pour décoder un symbole il utilise les comptages des symboles précédents : il est donc synchrone avec l'encodeur et tout se passe bien, pour autant que les conventions de départ soient respectées.

Quelles sont les conventions de départ ? Il s'agit d'un détail d'implémentation qui peut être traité avec un surcoût négligeable, comme nous le verrons.

Par contre une question de principe reste pendante. En effet, dans les méthodes des sections précédentes, nous avons implicitement supposé que l'arbre de code était construit par l'encodeur puis transmis au décodeur comme préambule du message codé. Ici, nous supposons que les deux construisent leur arbre de code en parallèle : que

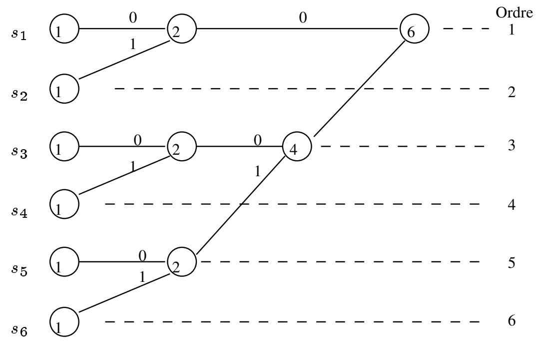


Figure 13.4. Arbre de Huffman (codage adaptatif) : initialisation

faire si certains choix sont ambigus (nous savons que la méthode de Huffman permet en principe de construire plusieurs arbres (cf. les deux feuilles les moins probables sont des siblings et il est indifférent de savoir lequel reçoit le bit 1). Pour ce problème on pourrait évidemment utiliser un ordre lexicographique, ce qui permet de trancher en associant le bit 0 au symbole le plus petit selon cet ordre (le préambule indiquerait alors au décodeur l'ordre lexicographique des symboles de source, si cela était nécessaire). Mais la question est plus tordue que cela, car le problème peut également se poser pour des noeuds internes de l'arbre de code. Il est clair, que d'une manière ou d'une autre il faut introduire des conventions sur la manière dont l'étiquetage des branches de l'arbre de Huffman doit être fait, afin de mettre encodeur et décodeur sur la même longueur d'onde.

Un autre problème est celui de la mise à jour de l'arbre de code. En effet, comme à chaque symbole les comptages changent, cet arbre est susceptible de changer également. Si nous voulons que l'algorithme de compression/décompression soit efficace du point de vue computationnel, il faut trouver une solution pour mettre à jour efficacement cet arbre.

Dans cette introduction nous allons un peu nous faciliter la tâche du point de vue explicatif, en utilisant une méthode de mise à jour peu subtile. Nous indiquons cependant qu'il existe un algorithme efficace, appelé méthode de "Knuth-Gallager", qui est celui qui est utilisé normalement en pratique. Nous en donnerons une description intuitive à la fin de cette section. Dans ce qui suit, nous allons traîter successivement les trois problèmes mentionnés : initialisation, ambiguïté et mise à jour de l'arbre.

Commençons par l'initialisation : tous les compteurs de symboles sont mis à 1.

Ensuite, construisons de façon non-ambiguë un arbre initial :

1. les symboles sont numérotés par ordre lexicographique (graphiquement de haut en bas, voir figure 13.4);

2. ces symboles sont ensuite associés aux noeuds terminaux de l'arbre de Huffman, et ces noeuds sont étiquetés au moyen des compteurs initialisés à 1 et nous leur associons également un numéro d'ordre correspondant à l'ordre lexicographique des symboles de départ;

3. l'arbre de Huffman est ensuite construit de manière non-ambiguë en fusionnant les deux noeuds ayant les compteurs les plus faibles, et en cas de possibilités multiples en choisissant les noeuds d'ordre (à ne pas confondre avec le niveau) le plus élevé : le noeud père hérite du numéro d'ordre le plus faible de ses fils, et son compteur est initialisé à la somme des compteurs des fils. Enfin, les branches sont étiquetées en associant un 0 au fils de numéro d'ordre le plus faible.

La figure 13.4 explicite l'arbre ainsi obtenu à la première étape du codage d'une source $S = \{s_1, s_2, s_3, s_4, s_5, s_6\}$.

Enfin, indiquons comment mettre à jour l'arbre en prenant en compte des symboles successifs du texte. Soit s_i le symbole suivant; on procède comme suit :

1. il est d'abord encodé à l'aide de l'arbre courant;

2. son compteur est incrémenté, ensuite la feuille qui lui correspond est déplacée verticalement (sans changer l'ordre relatif des autres noeuds), jusqu'à ce que les feuilles soient de nouveau classées par ordre décroissant de leurs compteurs : s'il se déplace, il se déplacera vers le haut, et le moins loin possible;

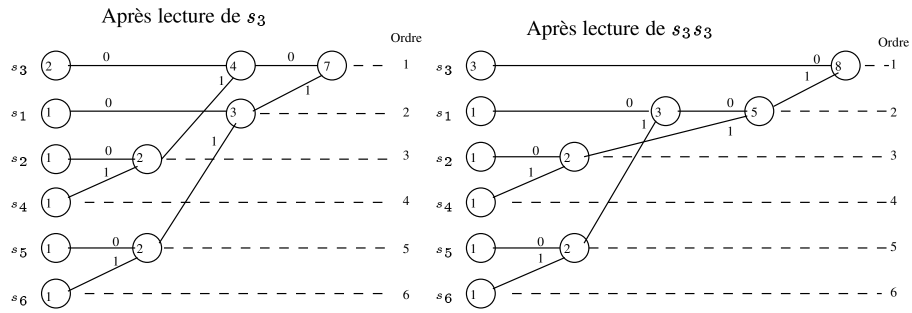


Figure 13.5. Arbre de Huffman (codage adaptatif) : deux étapes

3. l'arbre est reconstruit selon la méthode décrite pour l'arbre initial : en cas de possibilités multiples, priorité à la fusion des noeuds d'ordre le plus élevé.

La figure 13.5 indique les arbres obtenus après comptabilisation d'un premier symbole (s_3) et d'un second symbole (s_3, également).

Discussion. Dans la méthode qui précède, on comptabilise toutes les lettres du texte source. Le système tend donc progressivement vers le code qui serait obtenu si on avait dès le départ comptabilisé les fréquences relatives des symboles dans le texte source. On n'a donc pas vraiment affaire à une méthode adaptative.

Quel est alors son avantage, compte tenu des difficultés que nous venons de voir concernant la mise à jour (et tout n'a pas encore été dit à ce sujet) des comptages et arbres de code ? Un premier avantage assez évident est que cette méthode ne nécessite pas le stockage du texte source : elle fonctionne de manière purement on-line, ce qui permet de suivre (pour autant que les mises à jour se fassent de manière suffisamment rapide) exactement le rythme de la source. Or il existe de nombreuses applications (notamment dans le domaine du multimedia, ou dans le contexte de la téléphonie numérique) où on souhaite pouvoir transmettre les symboles au même rythme que le mécanisme qui les produit.

Ensuite, il existe un "truc" qui permet à la méthode de devenir potentiellement plus efficace, et de fait réellement adaptative : il suffit de lui permettre d'oublier une partie des symboles lus précédemment de façon à lui permettre de s'adapter de façon plus souple aux changements de comportement des messages de source. L'idée est très simple et consiste à diviser périodiquement les comptages par un nombre entier (disons division entière par une puissance entière de 2, ce qui facilite l'implémentation) : cela permet aux comptages de suivre de plus près le comportement non-stationnaire de la source. Si la source est stationnaire, cette opération n'aura pas un effet très marqué. Si, par contre, elle est effectivement non stationnaire l'effet peut être très spectaculaire. Bien sûr, le décodeur doit être en phase avec la politique adoptée par l'encodeur et il faut s'arranger pour éviter des comptes nuls. Mais il s'agit là de questions de détails que nous ne discuterons pas plus ici.

Enfin, si nécessaire, indiquons que les idées qui viennent d'être discutées sont également applicables dans le cadre du codage de Huffman d'ordre k.

Méthode de Knuth et Gallager. Nous venons de voir qu'il faut maintenir l'arbre du code à chaque symbole, ce qui peut parfois conduire à une modification complète de celui-ci. En l'absence d'autres possibilités il faut alors reconstruire complètement l'arbre à chaque étape, ce qui peut être lourd si l'alphabet de source est grand (voir également chapitre 8).

L'idée de Gallager est basée sur le résultat suivant.

Théorème. *Soit T un arbre binaire*[7] *avec $m \geq 2$ feuilles, à chaque noeud u duquel est associé un poids $w(u)$ positif. Supposons de plus que chaque noeud père reçoit comme poids la somme des poids de ses fils. Alors cet arbre est un arbre de Huffman (c'est-à-dire il peut être obtenu par application de l'algorithme de Huffman), si, et seulement si les 2m -2 noeuds (en excluant la racine) de l'arbre T peuvent être ordonnés selon un ordre, disons $u_1, \ldots, u_{2m-2}$, qui vérifie les deux propriétés suivantes :*

[7]Un arbre binaire comportant m feuilles possède exactement $2m - 1$ noeuds, au total.

1. $w(u_1) \leq w(u2) \leq \cdots \leq w(u_{2m-2})$ *et*

2. pour $k = 1, \ldots, m-1$, u_{2k-1} *et* u_{2k} *sont des siblings dans l'arbre.*

Suggestion : revoir à ce stade la partie du chapitre 8 concernant l'algorithme de Huffman et la pseudo-méthode d'optimisation que nous y avons discutée.

L'algorithme de Knuth-Gallager consiste alors à adapter l'arbre à chaque étape (correspondant à l'incrémentation d'un compteur de symbole) pour satisfaire cette propriété, ce qui est faisable en un nombre d'opérations au plus proportionnel à m, alors que la reconstruction complète de l'arbre prend de l'ordre de m^2 opérations. De plus, la méthode de Knuth-Gallager peut exploiter les situations où l'arbre avec compteurs mis à jour satisfait déjà la propriété (c'est fréquent, pourquoi ?) en ne faisant rien. Nous ne décrirons pas cet algorithme ici, mais nous demandons au lecteur de se convaincre que la propriété de Gallager est bien une condition nécessaire et suffisante d'arbre de Huffman.

13.4.2 Sur le codage arithmétique adaptatif

Ce qui vient d'être dit concernant la construction incrémentale des f_i, au fur et à mesure de la consommation d'un texte source n'est évidemment pas spécifique à la méthode de Huffman.

En réalité, il suffit de supposer que de part et d'autre de la chaîne de communication nous disposons d'une copie exacte du même programme qui fournit à chaque pas de l'algorithme le modèle de source à utiliser pour le traitement du symbole suivant. La mécanique du codage arithmétique est tout à fait indépendante de ce qui se trouve dans cette boîte que nous appelons "modèle probabiliste". C'est donc un plus très significatif par rapport à l'algorithme de Huffman, qui souffre de la nécessité de mettre à jour son arbre de code.

Il s'en suit que nous pouvons coupler le codage arithmétique avec n'importe quel modèle de source, qu'il soit purement adaptatif, complètement fixé a priori, ou un mélange de connaissances a priori et mises à jour adaptatives, comme le permettent par exemple les approches Bayesiennes de l'apprentissage automatique.

On voit que le codage arithmétique apparaît comme un dispositif extrêmement souple pour l'exploitation de modèles de source "intelligents". Cela montre également la proximité des problèmes de compression de données avec certains problèmes de l'intelligence artificielle, notamment de l'apprentissage automatique et du raisonnement à l'aide de modèles probabilistes. Il n'est donc pas étonnant de constater que nombre des concepts vus dans ce cours son utilisés par les méthodes de l'intelligence artificielle.

13.5 METHODES DE DICTIONNAIRE

Dans les sections qui précèdent, les schémas de compression réversibles ont été conçus sur base d'un modèle statistique (ou probabiliste) postulé pour la source, et dont les paramètres sont en général estimés à partir du texte source qu'il s'agit d'encoder (il est évidemment possible d'utiliser ces schémas en utilisant un modèle probabiliste dont les paramètres sont connus à l'avance). Les méthodes de dictionnaire fonctionnent sur base d'une logique différente : elles utilisent une liste de mots source (le dictionnaire) qui, on l'espère, est assez riche pour contenir un grand nombre de mots longs du texte source. Le texte est alors encodé sous la forme d'une suite de pointeurs vers les mots du dictionnaire (comparer cette idée à l'analyse grammaticale utilisée pour construire la suite de symboles source au chapitre précédent). Il est évident qu'il est nécessaire que le décodeur soit au courant du contenu du dictionnaire.

On peut imaginer plusieurs façons de faire. Au niveau le plus simple, on peut utiliser une collection de dictionnaires fixes (p.ex. un dictionnaire pour les textes en français, un dictionnaire pour les programmes écrits en langage C, un dictionnaire pour les fichiers postscript ...). Un inconvénient de cette approche réside dans le fait qu'il faut maintenir à jour les dictionnaires : les changements doivent donc être propagés à tous les sites qui utilisent ce schéma. Ensuite, l'utilisation de dictionnaires fixes ne permet pas de compresser des textes de nature inconnue.

Afin de permettre la compression de textes quelconques, il est clair qu'il faut adapter le contenu du dictionnaire au texte compressé. Nous allons donc nous intéresser ici uniquement aux méthodes de dictionnaire adaptatives. La conception de ces méthodes remonte aux articles de 1977 et de 1978 de Ziv et Lempel. Les schémas généraux sont connus sous le nom LZ77 et LZ78. Des applications qui utilisent des variantes du schéma LZ77 sont par exemple GNUzip et PKZIP. Les schémas de type LZ78 sont utilisés par les modems (V.42bis), l'utilitaire Unix compress, et le format graphique GIF.

mots source	λ	1	0	11	01	010	00	10
$c(n)$	0	1	2	3	4	5	6	7
$c(n)_{\text{adresse binaire}}$	000	001	010	011	100	101	110	111
(adresse, bit)	–	(000,1)	(000,0)	(001,1)	(010,1)	(100,0)	(010,0)	(001,0)

Figure 13.6. Exemple : Lempel-Ziv de base

L'idée de base de ces méthodes consiste à construire le dictionnaire au fur et à mesure du parcours du texte. L'approche est donc naturellement adaptative (le dictionnaire est seulement une fonction du texte compressé) et on-line (le dictionnaire ne contient que des mots précédemment lus).

La différence principale entre ces méthodes est liée à la gestion du dictionnaire. Dans LZ77, le dictionnaire est assez petit, car composé de fragments du texte appartenant à une "fenêtre de texte récemment lue". LZ78 utilise une approche différente, en gérant un dictionnaire de taille (lentement) croissante organisé de manière structurée de façon à minimiser l'encombrement et les temps d'accès.

Ci-dessous nous nous contentons de décrire les principes généraux sur lesquels ces deux méthodes, et toutes leurs variantes, sont basées, et nous allons indiquer comment les propriétés théoriques de ces méthodes leur permettent de porter le nom de méthode "universelle". Nous suggérons au lecteur intéressé par les aspects théoriques de consulter la référence [CT91], et à celui désireux d'en savoir plus sur les questions pratiques d'implémentation de consulter la référence [HHJ97].

13.5.1 Algorithme de Lempel et Ziv de base

La méthode consiste à remplacer des tronçons du texte, par des pointeurs vers des tronçons de texte déjà encodés/décodés. Cela veut dire que nous analysons les textes avec une grammaire qui croît incrémentalement de la manière suivante :

- on commence initialement avec une grammaire (ou un dictionnaire) vide;
- ensuite, à chaque pas de l'algorithme on parcout le texte à partir du symbole courant, jusqu'à obtenir un mot qui ne figure pas encore dans le dictionnaire, et on inclut ce mot dans le dictionnaire.

Par exemple, si le texte T est $1011010100010\ldots$, cela donne $1, 0, 11, 01, 010, 00, 10, \ldots$. Après chaque virgule nous avançons le long du texte T tant que le mot parcouru figure dans le dictionnaire. Un moment de réfexion nous dit que ce processus s'arrêtera sur un mot ne figurant pas dans le dictionnaire, disons de longueur $n_i + 1$, tel que le préfixe m_i de longueur n_i de ce mot figure déjà dans le dictionnaire. Ce préfixe est donc de longueur maximale. Ce mot peut donc être encodé en utilisant d'une part un pointeur vers l'addresse dans le dictionnaire où figure le mot m_i et un seul symbole source (dans notre exemple un bit supplémentaire) pour préciser le dernier symbole de ce mot.

Nous supposons que les adresses sont numérotées à partir de 0, et que le dictionnaire contient initialement une chaîne vide à l'adresse 0. Plaçons nous au moment de l'introduction d'un nouveau mot source dans le dictionnaire et supposons que le dernier symbole de de ce mot soit le n-ème symbole de source. Désignons par $c(n)$ l'adresse du dictionnaire où ce mot sera introduit. Donc, lors du passage du n-ème symbole du texte T (avant l'introduction éventuelle du mot courant), le dictionnaire contient déjà $c(n)$ mots et l'adresse maximale de ces mots est $c(n)-1$. Au fur et à mesure du parcours du texte, cette adresse croît (constante par morceaux) et vaudra $c(N) - 1$ lors du codage du dernier paquet de symboles d'un texte de longueur N. On peut représenter les pointeurs à l'aide de $\lceil \log_2 c(N) - 1 \rceil$ bits. Par exemple, en supposant dans le cas ci-dessus que le texte s'arrête après les symboles 1011010100010, on aboutit au moment du codage du dernier mot à un dictionnaire de taille 7 (adresse maximale 6), et on peut utiliser 3 bits pour les adresses de pointeurs.

La figure 13.6 décrit ce qui se passe pour notre exemple : le texte codé est 0001000000110101100001000010.

Evidemment, le travail de "compression" effectué dans le cas de notre exemple n'est pas très spectaculaire, puisque le texte codé est plus de deux fois plus long que le texte source. D'autre part, le codage nécessite deux passes : une première où on constitue le dictionnaire et où on calcule le nombre de bits nécessaires pour les adresses, et une seconde passe où on encode le texte de sortie. Nous verrons ci-dessous comment améliorer le système de façon à raccourcir le texte codé, et à rendre le codage on-line.

mots source	λ	1	0	11	01	010	00	10
$c(n)$	0	1	2	3	4	5	6	7
$c(n)_{\text{adresse binaire}}$	000	001	010	011	100	101	110	111
$\lceil \log_2 c(n) \rceil$	-	0	1	2	2	3	3	3
(adresse, bit)	–	(,1)	(0,0)	(01,1)	(10,1)	(100,0)	(010,0)	(001,0)

Figure 13.7. Exemple : Lempel-Ziv on-line

Cependant, la méthode très simple que nous venons de décrire possède des propriétés importantes que nous allons énoncer sans démonstration.

La méthode que nous venons de décrire encode $c(N)$ mots source, pour un texte de longueur N, en utilisant pour chaque mot $\lceil \log c(N) - 1 \rceil + 1$ bits, c'est-à-dire par symbole source un peu moins de

$$\frac{c(N)(\log c(N) + 1)}{N} \text{ bits.} \tag{13.30}$$

Taille maximale du dictionnaire. On montre que la taille maximale du dictionnaire vérifie la relation

$$c(n) \leq \frac{n}{(1 - \epsilon_n) \log n}, \tag{13.31}$$

où $\lim_{n\to\infty} \epsilon_n = 0$.

Optimalité asymptotique. On montre que pour une source stationnaire ergodique, on a, avec une probabilité de 1,

$$\lim_{n\to\infty} \frac{c(n)(\log c(n) + 1)}{n} = H(\mathcal{S}), \tag{13.32}$$

ce qui veut dire que l'algorithme de Lempel et Ziv est asymptotiquement optimal.

13.5.2 Caractère on-line

Nous avons vu que la méthode de base décrite ci-dessus ne préservait pas le caractère on-line de l'algorithme, suite à la nécessité de connaître la taille du dictionnaire avant le codage des adresses.

Il existe un moyen très simple qui permet de restaurer le caractère on-line, et qui, du même coup permet de raccourcir la taille du texte encodé. En effet, au moment du codage (et donc du décodage) du n-ème mot, on connaît l'adresse maximale du dictionnaire partiel dans lequel il faut aller chercher le préfixe ($c(n)$), et on peut donc se contenter d'indiquer l'adresse en utilisant $\lceil \log_2 c(n) \rceil$ bits. Dans le cas de notre exemple, cela donne (voir figure 13.7) le texte codé 100011101100001000010 (ce texte est malgré tout plus long que le texte de départ).

On voit que le schéma raccourcit la longueur moyenne des pointeurs, mais intuitivement on sent bien que si la taille du texte croît indéfiniment la taille moyenne continuera à croître de façon logarithmique. Rien ne change donc du point de vue des propriétés asymptotiques. Néanmoins, l'algorithme est meilleur sur les textes de taille finie rencontrés en pratique. En fait, pour une source de mémoire finie, disons m, les propriétés asympotiques ne seront approchées que lorsque la méthode aura stocké tous les messages typiques de longueur m, ce qui peut représenter un nombre astronomique en terme de longueur du texte source.

13.5.3 Adaptabilité

L'algorithme présenté conduit à un dictionnaire qui cumule l'information depuis le début du texte. Si la nature du texte change en cours de route (c'est-à-dire si le comportement est non-stationnaire) on paye dès lors un surcoût important en terme de tailles de pointeurs et temps de calcul associé au codage/décodage. Les versions pratiques de cette méthode, que nous avons mentionnées en début de cette section utilisent pour ces raisons des dictionnaires avec faculté d'oubli. En d'autres mots, on ne retient que les mots rencontrés dans une fenêtre de taille limitée autour de la position courante dans le texte source. D'autre part, il est possible de comptabiliser la "popularité" de certains mots du dictionnaire, de façon à organiser celui-ci de manière optimisée en utilisant les techniques de codage statistique vues antérieurement (p.ex. codage de Huffman sur les adresses).

13.5.4 Optimalité relative

Les méthodes dérivées de l'algorithme Lempel-Ziv décrit de façon simplifiée dans ce qui précède sont populaires car très efficaces du point de vue computationnnel et en même temps capables de donner de bons résultats sur de nombreux types de textes source. Néanmoins, le fait que ces méthodes ne soient pas capables d'exploiter une connaissance a priori sur la nature du texte source (i.e. un modèle probabiliste) les rend forcément sous optimales, fait qu'elles atteignent leurs performances asymptotiques seulement pour des textes source très longs.

13.6 PARSING CONSTRUCTIF

Ici : la méthode de Ziv pour les arbres de parsing; l'utilisation de la méthode de construction d'arbres de décision pour l'identification de multigrammes.

13.7 RESUME

Dans ce chapitre nous avons développé les concepts qui sont à la base de la compression de données : analyse (segmentation de textes), modèles probabilistes, principales méthodes de compression, adaptabilité et caractère on-line.

Notre objectif était de mettre en évidence l'ensemble des éléments qui déterminent quel type de code est approprié pour une application pratique donnée. Sans aller jusque dans les détails d'implémentation, nous avons donné les indications nécessaires pour la mise en oeuvre des méthodes ainsi que les principales propriétés théoriques qui permettent d'en délimiter le champs d'application.

Avec la matière théorique du chapitre 8, et celle du chapitre 14, cela constitue une bonne vue d'ensemble de ce domaine. Jusqu'ici nous avons rencontré trois types de codes pour la compression de données :

Les codes de longueur fixe : c'était l'outil exploité dans le cadre du théorème AEP, qui dit essentiellement que seulement une très faible partie des messages possibles doivent être codés de manière efficace. Ces codes, bien que simples du point de vue conceptuel, ne sont néanmoins pas d'une grande utilité pratique.

Les codes de symboles : ils associent aux symboles source des mots de code dont la longueur est déterminée par leurs probabilités. L'algorithme de Huffman construit un code optimal pour des probabilités données. Ces codes sont sans préfixes et n'importe quel texte est encodé de façon déchifffrable. Si les symboles source respectent les probabilités utilisées lors de la construction du code, la longueur moyenne de celui-ci atteint la limite de Shannon à 1 bit près. Les codes de symboles sont l'équivalent des systèmes statiques de la théorie des systèmes : une fois la loi de probabilité fixée, la sortie à un moment donné (c'est-à-dire correspondant à un symbole de source donné) ne dépend que du symbole courant. Nous avons cependant montré que cette technique se prêtait à l'adaptation et au codage on-line, par le biais d'un modèle probabiliste adaptatif.

Les codes de "flux" : la caractéristique qui les distingue des précédents est qu'ils ne sont pas contraints d'émettre au moins un bit par symbole source. Un texte de longueur donnée n peut parfois être encodé en un message binaire de longueur $m < n$. Indépendamment du modèle probabiliste lui-même, ces codes se comportent en fait comme un système dynamique, puisque la sortie du codeur à un moment donné dépend non seulement du symbole source courant, mais également de ceux qui le précèdent. Dans cette famille de codes, nous avons décrit les deux "maîtres achats" du moment :

- Les codes arithmétiques, qui combinent un modèle probabiliste avec un algorithme de codage qui associe un texte source à un sous-intervalle de $[0, 1[$ dont la taille est approximativement proportionnelle à la probabilité du texte, telle que calculée par le modèle. Ce type de code est virtuellement optimal, si la source respecte le modèle probabiliste utilisé. Les codes arithmétiques mettent bien en évidence le fait que de bonnes performances de compression requièrent de *l'intelligence*, sous la forme d'un modèle probabiliste éventuellement adaptatif.
- Les algorithmes de la famille Lempel-Ziv sont également adaptatifs, dans la mesure où ils mémorisent les chaînes qui ont déjà été rencontrées. Ils sont fondés sur une philosophie "déterministe" qui dit que nous ne savons rien a priori au sujet du modèle probabiliste de la source, et que nous souhaitons un algorithme de compression qui est raisonnablement performant quel que soit ce modèle.

Nous faisons remarquer que dans ces deux philosophies on retrouve deux tendances fortes de l'intelligence artificielle : (i) une approche dirigée par la modélisation; (ii) une approche entièrement dirigée par les données. Les codes arithmétiques constituent essentiellement une approche dirigée par le modèle probabiliste. Si des informations a priori sont disponibles sur celui-ci, alors ces méthodes peuvent l'exploiter. Par ailleurs, si aucune information a priori n'est disponible, ces méthodes sont néanmoins opérationnelles, à condition de leur associer un algorithme d'apprentissage adaptatif, qui construit le modèle probabiliste au fur et à mesure que des données sont observées (par exemple, au fur et à mesure que le texte source est consommé).

Nous verrons dans la suite de ce cours que les concepts que nous venons de discuter sont omniprésents dans le domaine du traitement de l'information. En particulier, lorsque nous considérerons le problème du codage/décodage de canal nous verrons à nouveau toute l'importance des modèles probabilistes.

14 COMPRESSION D'IMAGES

De nos jours, la transmission et le stockage d'informations sur forme d'images est omniprésente, et la compression de données joue un rôle très important dans ce contexte. En effet, le volume de données brutes associées à des images sont très importants ce qui en impose la compression à la fois lors du stockage et de la transmission. Par exemple, une image haute résolution affichée sur un écran d'ordinateur (ou de télévision) représente de l'ordre de quelques MB en mémoire. Une séquence vidéo représente de l'ordre de 20 images par seconde, soit pour une minute de temps réel de l'ordre d'un GB de données brutes. Il est clair que sans l'utilisation des techniques de compression d'images il serait impossible (ou extrêmement coûteux) de transmettre et de stocker de façon digitale de tels volumes de données. Or comme nous l'avons vu dans ce cours, les techniques digitales permettent de manipuler efficacement les données grâce à l'informatique, et surtout, de lutter efficacement contre le bruit et la dégradation de la qualité au cours du temps.

Nous verrons dans ce chapitre que les techniques actuelles permettent (au sacrifice de la réversibilité) de réduire ces volumes de données par des facteurs de l'ordre de 100. Cela veut dire que, grâce à ces techniques, il devient possible de transmettre des images 100 fois plus rapidement (quelques secondes au lieu de plusieurs minutes, pour transmettre une image sur une ligne téléphonique) et d'en stocker 100 fois plus sur un même support (quelques dizaines de minutes de film vidéo sur un disque compact, au lieu d'une demie seconde).

La compression d'images est donc devenue un domaine extrêmement important pour permettre la mise en place des services multimédia, qui représentent une véritable industrie à l'heure actuelle, et certainement encore pour de nombreuses années. Remarquons que ce que nous venons de dire est également vrai pour la transmission et le stockage de sons, mais ici les volumes de données sont typiquement de l'ordre de 100 fois plus faibles qu'en imagerie.

Il y a d'autres domaines, moins bien connus du grand public, où les techniques de compression d'images deviennent très importantes. Citons par exemple le domaine médical, qui repose de plus en plus sur l'imagerie (vision en temps réel, archivage et échanges d'images médicales). Un autre domaine intéressant concerne la géophysique et l'astronomie. Ces sciences reposent en grande partie sur l'acquisition et l'analyse de volumes très importants d'images. Par exemple, le système d'observation de la terre (projet NASA) est composé d'un certain nombre de satellites qui, lorsqu'ils seront opérationnels, enverront de l'ordre de 50GB d'images par heure vers les centres de recherche terrestres. Un autre exemple est fourni par les bases de données d'empreintes digitales (par exemple celle du FBI comporte plusieurs dizaines de millions d'images : plusieurs dizaines de TeraBytes).

Le traitement d'images comporte un grand nombre de volets : l'acquisition, le filtrage, la compression, l'analyse, la reconnaissance, et la synthèse. Ici nous allons nous focaliser sur les questions de représentation et de compression d'images qui sont intimement liées.

Le lecteur sceptique pourrait se demander pourquoi nous dédions un chapitre complet à ces questions, et surtout pourquoi nous donnons ainsi un statut particulier à la compression d'images par rapport aux méthodes de compression de textes. Il y a plusieurs bonnes raisons à cela.

Tout d'abord, bien que les techniques développées au chapitre précédent puissent s'appliquer au domaine de l'image, il est avantageux d'exploiter la structure physique (bi- ou tri-dimensionnelle) des images lors de leur compression. La compression d'images dispose donc de techniques propres bien adaptées à ce problème, et il est préférable de les décrire dans un cadre moins abstrait.

D'autre part, comme le traitement d'images devient de plus en plus important il nous a semblé utile d'introduire quelques notions utiles dans ce contexte, afin de permettre aux étudiants qui n'auront pas l'occasion de se spécialiser en traitement d'images, de disposer néanmoins d'une certaine culture générale dans ce domaine. En lui consacrant un chapitre séparé, nous nous donnons un peu plus de liberté pour sortir du cadre strict de la compression de données.

Ensuite, nous verrons que la compression d'images conduit à généraliser les méthodes uni-dimensionnelles (dimension = temps) vues auparavant pour traiter des problèmes multi-dimensionnels. C'est donc un pas supplémentaire vers une plus grande richesse des méthodes que nous nous proposons de faire dans ce chapitre.

Nous allons également parler ici pour la première fois de techniques de codage irréversibles. En réalité tout un pan de la théorie de l'information étudie cette question. Il s'agit de la théorie de la *distorsion* qui évalue comment on peut augmenter les débits sur des canaux en tolérant une certaine irréversibilité. Nous donnerons une très brève introduction à cette branche de la théorie de l'information dans la troisième partie de ce cours. Mais il nous semble que le domaine de la compression d'images est particulièrement bien adapté pour parler de méthodes irréversibles : la notion d'image est intuitivement facile à comprendre et donc l'approximation d'images est plus facile à discuter de façon intuitive que l'approximation de signaux plus abstraits.

En corollaire, nous pensons que la matière que nous allons voir dans ce chapitre se prête particulièrement bien à une présentation intuitive, avec un minimum de formalismes mathématiques. Nous supposons que cela permettra aux étudiants de "souffler" un peu après avoir digéré un certain nombre de concepts nouveaux.

Nous signalons au lecteur que nous nous sommes basés en partie sur les références [GW93, HHJ97] pour la rédaction de ce chapitre.

14.1 QU'EST-CE QU'UNE IMAGE ?

Dans cette section nous nous proposons de décrire le modèle mathématique utilisé pour les images et d'introduire un des outils fondamentaux dans ce domaine, à savoir les *transformées* d'images.

14.1.1 Coordonnées spatiales

Mathématiquement, le terme image (monochrome) fait référence à une fonction d'intensité lumineuse bidimensionnelle

$$f(x,y) \ : [0, x_{max}] \times [0, y_{max}] \longrightarrow [0, f_{max}], \tag{14.1}$$

où x désigne la coordonnée verticale (usuellement orientée de haut en bas), y la coordonnée horizontale (orientée de gauche à droite), et $f(x,y)$ représente l'intensité lumineuse au point (x,y). On utilise le terme de coordonnées spatiales pour désigner le couple (x,y), et on parle de représentation spatiale lorsque l'image est donnée sous la forme d'une fonction $f(x,y)$.

Ce type de structure de données peut représenter par exemple une image photographique noir et blanc, une radiographie, ou l'une des composantes d'images multispectrales (p.ex. RGB, intensité d'énergie dans une certaine bande de longueurs d'onde). Elle peut aussi être utilisée pour représenter une fonction plus abstraite de deux variables, telle qu'un champ scalaire.

Dans le domaine vidéo on est également amené à représenter une suite temporelle d'images ($f(x,y,t)$) où le temps est généralement discret. De même, en physique on est amené à représenter des champs scalaires qui évoluent au cours du temps.

Par ailleurs, dans la plupart des applications modernes de l'imagerie on considère des versions échantillonnées de la fonction $f(x, y)$ selon une matrice de coordonnées spatiales $N \times M$: x représente alors l'indice de ligne de cette matrice et y l'indice de colonne. On utilise alors le terme de *pixel* (de l'anglais "picture element") pour désigner un point de cet espace discret, et par extension la valeur prise par $f(\cdot, \cdot)$ en ce point.

La convention d'orientation de haut en bas et de gauche à droite que nous avons choisie est cohérente avec la manière dont on parcourt un texte dans nos langues et la manière dont nous lisons habituellement une matrice. Remarquons cependant que contrairement à la dimension temporelle des signaux dynamiques, dont l'orientation est justifiée par le principe de causalité, dans le domaine du traitement d'images il n'y pas de justification physique de cette orientation.

Dans ce qui suit, nous allons nous focaliser entièrement sur des images échantillonnées du point de vue spatial. D'autre part, dans un certain nombre de cas nous supposerons également que les valeurs possibles prises par $f(x, y)$ sont discrétisées.

Modèles probabilistes. Dans ce chapitre nous ferons l'impasse sur la modélisation probabiliste d'images. Nous signalons au lecteur interessé que de nombreux travaux, notamment dans le domaine du filtrage d'images, exploitent des modèles probabilistes similaires à ceux que nous avons développés au chapitre 8 pour les sources. En traitement d'images, les modèles de Markov deviennent des réseaux de Markov qui permettent de modéliser des dépendances locales entre pixels voisins. Disons simplement qu'un réseau de Markov exprime le fait que lorsque les valeurs des pixels qui appartiennent à un certain voisinage d'un pixel donné sont connues, la valeur de ce pixel devient indépendante des autres pixels de l'image (plus éloignés). Cette propriété permet la constitution de modèles probabilistes avec un nombre (très relativement) limité de paramètres et qui modélisent néanmoins une partie importante des redondances présentes dans la plupart des images.

Enfin, signalons que les méthodes de dictionnaire se généralisent également aux images.

14.1.2 Transformées d'images

Tout comme en traitement du signal, les *transformées* (de Fourier, et autres) jouent un rôle très important dans le contexte du traitement d'images. Par exemple, les transformées de Fourier permettent de projeter l'information contenue dans une image dans le domaine fréquentiel, puis d'effectuer des opérations de filtrage dans ce domaine, en tirant profit des caractéristiques fréquentielles différentes du bruit et du signal utile.

En ce qui concerne notre sujet d'intérêt dans le cadre de ce cours, nous verrons que les transformées permettent de représenter l'information contenue dans une image sous une forme qui en facilite la compression. En particulier, alors que dans le domaine spatial une grande partie de la redondance d'images se situe au niveau des corrélations entre pixels voisins, les transformées d'images visent essentiellement à décorréler, c'est-à-dire à concentrer la redondance dans les monogrammes. Nous verrons plus loin comment ces transformées permettent de construire des approximations compactes d'images sans en dégrader la qualité telle que nous la percevons. C'est cette possibilité qui, lorsqu'elle est exploitée correctement, permet d'obtenir les taux de compression les plus spectaculaires, au prix d'un certaine irréversibilité "mathématique".

Nous ne nous attarderons pas ici sur les aspects relatifs à l'implémentation des opérations de transformation. Nous allons nous contenter d'en énoncer le principe général et de citer quelques transformées couramment utilisées. Nous reviendrons, dans les sections suivantes, sur l'utilisation de ces méthodes à des fins de compression. Pour nous simplifier la vie, nous allons considérer des matrices d'images carrées dans le domaine spatial.

Notons que toutes les transformées que nous considérons ici sont linéaires : la transformée d'une combinaison linéaire d'images est la combinaison linéaire correspondante des transformées de ces images.

14.1.2.1 Formulation. La transformée $\mathcal{G}$ d'une image $f(x, y)$ (de dimensions $N \times N$) au moyen d'un *noyau* $g(\cdot, \cdot, \cdot, \cdot)$ est la nouvelle image $N \times N$

$$\mathcal{G}(f) = T(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) g(x, y, u, v). \tag{14.2}$$

La transformée est inversible, et admet la transformée inverse $\mathcal{H}$ ayant comme noyau $h(\cdot,\cdot,\cdot,\cdot)$, si $\forall f(x,y)$

$$f(x,y) = \mathcal{H}(\mathcal{G}(f)) = \mathcal{H}(T) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v)h(x,y,u,v). \tag{14.3}$$

Les fonctions $g(i,j,\cdot,\cdot)$ et $h(\cdot,\cdot,i,j)$ peuvent êtres vues comme N^2 "fonctions" de base d'un développement en série.

Le noyau $g(\cdot,\cdot,\cdot,\cdot)$ (resp. $h(\cdot,\cdot,\cdot,\cdot)$) est dit séparable s'il peut s'écrire sous la forme $g(x,y,u,v) = g_1(x,u)g_2(y,v)$ (resp. $h(x,y,u,v) = h_1(x,u)h_2(y,v)$). Il est dit symétrique s'il est séparable et si on peut prendre $g_1(\cdot,\cdot) = g_2(\cdot,\cdot)$.

Il est clair alors qu'une transformée séparable et symétrique est spécifiée par la fonction $g_1(\cdot,\cdot)$ c'est-à-dire une matrice $N \times N$. On peut se convaincre que la transformée d'image est alors équivalente au double produit matriciel suivant

$$\boldsymbol{T} = \boldsymbol{G}^T\boldsymbol{F}\boldsymbol{G} \tag{14.4}$$

où $\boldsymbol{T}$ représente la matrice d'image transformée, $\boldsymbol{F}$ la matrice d'image dans le domaine spatial, et où $\boldsymbol{G} = [g(i,j)]$ est la matrice qui représente le noyau. En effet, on a en effectuant d'abord la transformée ligne par ligne

$$\begin{aligned} T(u,v) &= \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)g(x,y,u,v) && (14.5)\\ &= \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)g_1(x,u)g_1(y,v) && (14.6)\\ &= \sum_{x=0}^{N-1} g_1(x,u)\left[\sum_{y=0}^{N-1} f(x,y)g_1(y,v)\right] && (14.7)\end{aligned}$$

ce qui peut aussi s'écrire sous la forme

$$\boldsymbol{T} = \boldsymbol{G}^T[\boldsymbol{F}\boldsymbol{G}] = \boldsymbol{G}^T\boldsymbol{F}\boldsymbol{G}, \tag{14.8}$$

où le produit $\boldsymbol{T}_c = \boldsymbol{F}\boldsymbol{G}$ représente le calcul des transformées unidimensionnelles des N lignes de l'image $\boldsymbol{F}$. Le produit $\boldsymbol{G}^T\boldsymbol{T}_c$ représente le calcul des transformées unidimensionnelles des colonnes de $\boldsymbol{T}_c$.

La transformée inverse, si elle existe, est de toute évidence définie par la matrice $\boldsymbol{H} = \boldsymbol{G}^{-1}$. Cela veut donc dire que nous aurons affaire à une transformée inversible si et seulement si la matrice $\boldsymbol{G}$ est non-singulière. Dans les exemples que nous allons mentionner ci-dessous nous utiliserons des noyaux dont la matrice $\boldsymbol{G}$ est symétrique et unitaire (orthogonale dans le cas réel).

On voit, à partir de la formule (14.4) que, lorsque le noyau est symétrique, la transformée commute avec l'opération de transposition : la transformée d'une image transposée est alors la transposée de la transformée de l'image de départ.

Partant des transformées unidimensionnelles on peut donc construire les transformées bi-dimensionnelles (et plus généralement multi-dimensionnelles) séparables et symétriques correspondantes.

14.1.2.2 Transformée de Fourier bi-dimensionelle. La transformée de Fourier utilise le noyau suivant

$$g^F(x,y,u,v) = \frac{1}{N}\exp\left(\frac{-j2\pi(xu+yv)}{N}\right). \tag{14.9}$$

Ce noyau est évidemment séparable, car

$$g^F(x,y,u,v) = g_1^F(x,u)g_1^F(y,v), \tag{14.10}$$

avec

$$g_1^F(x,u) = \frac{1}{\sqrt{N}}\exp\left(\frac{-j2\pi(xu)}{N}\right), \tag{14.11}$$

Table 14.1. Transformée de Walsh (noyau pour$N = 8$)

x \ u	0	1	2	3	4	5	6	7
0	+	+	+	+	+	+	+	+
1	+	+	+	+	-	-	-	-
2	+	+	-	-	+	+	-	-
3	+	+	-	-	-	-	+	+
4	+	-	+	-	+	-	+	-
5	+	-	+	-	-	+	-	+
6	+	-	-	+	+	-	-	+
7	+	-	-	+	-	+	+	-

et comme $xu = ux$ on a de plus $g_1^F(i,j) = g_1^F(j,i)$. Cela veut dire que la matrice complexe $\boldsymbol{G}$ est symétrique ($\boldsymbol{G}^T = \boldsymbol{G}$)[1] Cette matrice est de plus unitaire : son inverse est donc identique à son adjointe qui est identique à sa conjugée en vertu de la symétrie de $\boldsymbol{G}$

$$\boldsymbol{G}^{-1} = \boldsymbol{G}^* = \overline{\boldsymbol{G}}. \tag{14.12}$$

Le noyau de la transformée inverse de Fourier est donc

$$h^F(x,y,u,v) = \frac{1}{N} \exp\left(\frac{+j2\pi(xu+yv)}{N}\right). \tag{14.13}$$

Nous avons ici affaire à un cas particulier de base orthogonale à deux dimensions. Notons au passage que ce que nous venons de faire pourrait aisément s'étendre à un nombre de dimensions plus grand.

Comme la transformée de Fourier bi-dimensionelle est équivalente à 2N transformées de Fourier uni-dimensionnelles, on peut donc exploiter l'algorithme FFT (fast Fourier transform) qui ne nécessite pour chacun de ces calculs que de l'ordre de $N \log N$ opérations.

Par ailleurs, le théorème d'échantillonnage vu dans le cas de signaux temporels au chapitre 10, s'applique également aux images pour choisir des intervalles d'échantillonnage appropriés en fonction du contenu fréquentiel (d'une image).

Tout comme en traitement du signal, la transformée de Fourier joue évidemment un rôle de tout premier plan en traitement d'images. Cependant, nous allons voir qu'il y a d'autres transformées très utilisées en pratique.

14.1.2.3 Transformée de Walsh. Cette transformée est obtenue en utilisant des fonctions de base (orthogonales) qui ne prennent que deux valeurs possibles $\pm\frac{1}{\sqrt{N}}$. Elle n'est définie que si $N = 2^n$ avec n entier, ce qui est souvent le cas en pratique.

Par exemple, la table 14.1 donne les valeurs du noyau (unidimensionnel) $g_1^W(x,u)$ pour $N = 8$. De toute évidence la matrice $\boldsymbol{G}$ est réelle, symétrique, et orthogonale. Elle est donc identique à son inverse. Mathématiquement, le noyau peut s'écrire sous la forme compacte suivante :

$$g_1^W(x,u) = h_1^W(x,u) = \frac{1}{\sqrt{N}} \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)} \tag{14.14}$$

où $b_k(z)$ désigne le k-ème bit de la représentation binaire du nombre entier z.

Tout comme pour la transformée de Fourier il existe un algorithme rapide pour le calcul de la transformée de Walsh.

[1] Il faut faire attention à ne pas confondre la symétrie de la transformée avec la symétrie de la matrice de transformation.

Table 14.2. Transformée de Hadamart pour$N = 8$)

x \ u	0	1	2	3	4	5	6	7
0	+	+	+	+	+	+	+	+
1	+	-	+	-	+	-	+	-
2	+	+	-	-	+	+	-	-
3	+	-	-	+	+	-	-	+
4	+	+	+	+	-	-	-	-
5	+	-	+	-	-	+	-	+
6	+	+	-	-	-	-	+	+
7	+	-	-	+	-	+	+	-

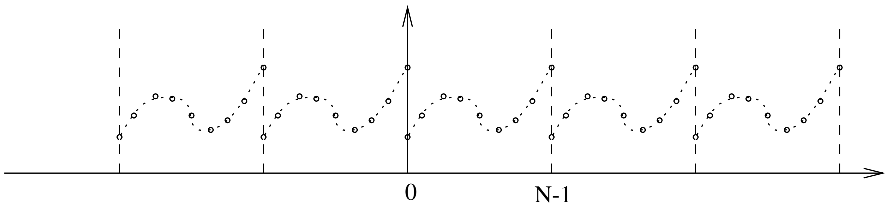


Figure 14.1. Extension périodique d'un signal : transformée de Fourier

14.1.2.4 Transformée de Hadamart. La transformée de Hadamart est identique à la transformée de Walsh, à une permutation près des lignes et colonnes de la matrice $\boldsymbol{G}$.

Par exemple, la table 14.2 indique les valeurs du noyau (unidimensionnel) $g_1^H(x, u)$ pour $N = 8$. A nouveau la matrice $\boldsymbol{G}$ est identique à son inverse. On montre que la matrice $\boldsymbol{G} = \frac{1}{\sqrt{N}}\boldsymbol{H}_N$ où la matrice $\boldsymbol{H}_N$ est obtenue récursivement par la formule suivante

$$\boldsymbol{H}_{2^0} = [1]; \boldsymbol{H}_{2^n} = \begin{bmatrix} \boldsymbol{H}_{2^{n-1}} & \boldsymbol{H}_{2^{n-1}} \\ \boldsymbol{H}_{2^{n-1}} & -\boldsymbol{H}_{2^{n-1}} \end{bmatrix}. \tag{14.15}$$

On peut voir les transformées de Walsh-Hadamart comme des versions simplifiées de la transformée de Fourier qui ont l'avantage de conduire à des calculs plus simples (additions/soustractions) et une image transformée dans le domaine réel.

Cependant, alors que la transformée de Fourier se généralise aisément au cas continu (remplacement des sommes par des intégrales), ce n'est pas le cas pour les transformées de Walsh-Hadamart, qui sont essentiellement discrètes.

14.1.2.5 Transformée en cosinus. La transformée de Fourier discrète que nous avons discutée ci-dessus n'est rien d'autre que le développement en série de Fourier d'une fonction périodique (période N), obtenue par extension périodique de la fonction de départ (voir 14.1). Un des inconvénients majeurs de la transformée de Fourier est d'introduire des composantes hautes fréquences liées aux effets de bords : si la valeur de $f(x, y)$ est différente en $x = 0$ et $x = N - 1$, l'extension périodique de l'image présente des discontinuités en ces points, ce qui se traduit par des composantes artificielles de haute fréquence. Par ailleurs, si l'extension périodique n'est ni paire, ni impaire, alors la série de Fourier comportera à la fois des termes en sinus et en cosinus (en d'autres termes, l'image transformée est complexe).

La transformée en cosinus est une version manipulée de la transformée de Fourier dans le but de faire en sorte que l'image transformée reste réelle lorsque l'image originale est à valeurs réelles. Elle consiste simplement à construire à partir du signal de départ un signal dont l'extension périodique sera continue et paire (voir figure

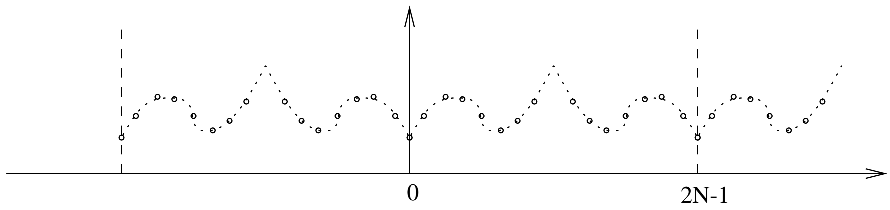


Figure 14.2. Extension périodique paire : transformée en cosinus

14.2). Le résultat est que la transformée de Fourier de ce nouveau signal sera purement réelle et ne présentera pas de termes haute fréquence liés aux effets de bords.

Il s'agit également d'une transformée symétrique et orthogonale. On montre (avec quelques manipulations algébriques) que cette transformée est définie par le noyau suivant[2]

$$g_1^C(x,u) = h_1^C(x,u) = \frac{1}{\sqrt{N}}\alpha(u)\cos\left(\frac{(2x+1)u\pi}{2N}\right), \tag{14.16}$$

avec

$$\alpha(0) = 1, \alpha(i) = \sqrt{2}, \forall i = 1,\ldots,N-1. \tag{14.17}$$

Cette transformée est devenue populaire dans les années récentes, suite au fait qu'elle est beaucoup utilisée dans le domaine de la compression d'images.

14.1.2.6 Opérations faciles sur les images transformées. L'intérêt pratique des transformées d'images est qu'elles permettent d'effectuer des opérations sur les images de manière efficace. En particulier, des opérations telles que filtrage des hautes fréquences se traduisent dans le domaine de Fourier par une simple multiplication par zéro des composantes fréquentielles filtrées. Par exemple, si on sait que l'image est composée de deux parties "signal" (basse fréquence) et "bruit" (haute fréquence) on peut améliorer la qualité de l'image en filtrant les hautes fréquences.

On sait également que les opérations de convolution (filtres convolutionnels) peuvent être réalisées par simple multiplication dans le domaine fréquentiel, c'est-à-dire avec un nombre d'opérations de l'ordre de $N^2 \log N$ au lieu de N^4.

De même, dans le domaine de la compression de données, l'entropie de l'image originale (distribuée sur un grand nombre de pixels) peut être concentrée sur un nombre beaucoup plus réduit de pixels de l'image transformée. En effet, pour la plupart des images on constate que les valeurs de la plupart des pixels de l'image transformée (p.ex. les composantes haute fréquence) sont très proches de zéro. Il s'en suit une réduction très marquée de l'entropie de l'histogramme de l'image transformée par rapport à l'image dans le domaine spatial. Cela permet aussi d'associer un code différent pour les pixels associés à des parties entières de l'image transformée. Nous reviendrons ultérieurement sur cette question.

On voit que les transformées de Walsh-Hadamart effectuent des opérations de moyennage de pixels voisins. Si on met à zéro les composantes haute fréquence cela aura donc pour effet de lisser l'image en effectuant des moyennes de pixels voisins (ce qui est équivalent à un changement de résolution).

Suggestion. La transformée de Walsh est définie pour des images $2^n \times 2^n$. Montrer que si nous reconstruisons une image avec les $2^{n-1} \times 2^{n-1}$ premiers termes de la transformée de Walsh, cela revient en fait à remplacer des blocs de pixels 2×2 par leur valeur moyenne. Plus généralement, si nous ne prenons que les $2^{n-i} \times 2^{n-i}$ permiers termes, cela revient à remplacer des blocs de pixels $2^i \times 2^i$ par leur valeur moyenne. En déduire la signification des $2^j \times 2^j$ premiers termes de la transformée de Walsh.

14.2 SOURCES DE REDONDANCE

Avant de discuter des techniques de compression d'image, voyons quelles sont les principales sources de redondance présentes dans une image. Nous utiliserons la terminologie de [GW93], qui distingue trois principales sources de redondance.

[2]Remarquer qu'il n'y a que n termes, alors que nous partons d'un transformée de Fourier de $2N$ valeurs.

14.2.1 Redondance de codage

Lorsqu'on considère l'histogramme des niveaux de gris d'une image monochrome, on se rend compte que certains niveaux de gris sont nettement plus fréquents que d'autres. En supposant que les valeurs de $f(x,y)$ soient discrètes (par exemple des nombres entiers représentés avec un certain nombre de bits), cela veut dire qu'il est possible de compresser l'image de façon réversible en utilisant les techniques du chapitre précédent, soit en associant aux différents niveaux de gris des codes de longueur variable, soit en associant un nombre réel à l'ensemble de l'image (codage arithmétique).

Les taux de compression qu'on peut obtenir à ce niveau sont typiquement de l'ordre de 2 à 3 au maximum.

14.2.2 Redondance inter-pixel

Il est clair qu'une bonne partie de la redondance d'images est plutôt liée au fait que les pixels voisins sont corrélés, c'est-à-dire que leurs niveaux de gris sont en général proches. Il est donc intéressant d'utiliser des modèles d'ordre supérieur.

Au moins trois stratégies sont possibles à ce niveau : (i) utilisation des techniques d'ordre supérieur adaptatives décrites au chapitre précédent, en parcourant l'image ligne par ligne ou colonne par colonne; (ii) utilisation de transformées d'images qui ont pour effet de modifier la structure de dépendance des pixels, et en particulier qui conduisent à une concentration de l'information dans un nombre réduit des pixels de l'image transformée; (iii) approches physiques et algorithmes de codage différentiels (voir ci-dessous).

Il est à remarquer que compte tenu de la grande taille des images (typiquement de l'ordre 1MB), les techniques dérivées de la méthode Lempel-Ziv peuvent être relativement performantes. Comme elles ne font pas d'hypothèse forte sur le modèle de source, elles permettent de compresser relativement bien la plupart des images. Par exemple, le format GIF utilise ce type de technique.

Lorsqu'on tient compte des redondances inter-pixel, les taux de compression en mode réversible peuvent typiquement monter à 8-10, ce qui représente une amélioration non-négligeable par rapport aux méthodes d'ordre zéro.

14.2.3 Redondance psychovisuelle

Enfin, un troisième niveau de redondance est lié non pas à l'information contenue dans l'image mais aux limitations du système qui va interpréter l'image (notre système psychovisuel en l'occurrence). En effet, notre système visuel est parfaitement capable de percevoir le message contenu dans une image même si celle-ci est légèrement bruitée. Inversement, ce système n'est pas capable de détecter certaines différences fines, et il n'est donc pas nécessaire non plus de les représenter.

Il est très intéressant de tenir compte des caractéristiques de ce système pour décider quelles "erreurs" de décodage peuvent être tolérées, et en déduire des techniques irréversibles qui permettent d'obtenir des taux de compression de l'ordre de 100 ou plus.

Cela est particulièrement vrai dans le contexte d'images vidéo, où la contrainte de temps est telle que nous ne pouvons percevoir en réalité qu'une très faible fraction de l'information contenue dans une image. Cette réalité est exploitée par exemple dans le domaine de la télévision, où certains formats de codage tiennent compte du fait que la résolution aux bords de l'image peut être réduite sensiblement par rapport à celle utilisée au centre, sans impact sur la qualité de l'image perçue.

14.3 SYSTEME DE COMPRESSION D'IMAGES

Après cette description intuitive des sources de redondance et des transformées d'image, essayons de mettre de l'ordre dans nos idées. La figure 14.3 représente les différents modules qui composent la plupart des systèmes pratiques de compression d'images.

L'encodeur est composé des trois blocs successifs :

- le premier bloc effectue un changement de représentation au moyen d'un transformée d'image (ou de toute autre technique), qui vise à réduire les redondances inter-pixel;

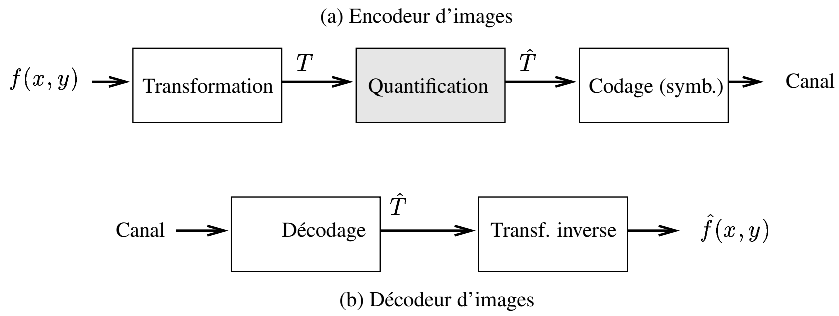


Figure 14.3. Système de compression d'image

- le second bloc n'est pas présent dans le cas de méthodes réversibles; il effectue une opération irréversible qui consiste généralement à réduire la précision de la sortie du premier bloc (par exemple en groupant certaines valeurs proches); ce bloc a pour effet de réduire considérablement l'entropie des pixels de l'image transformée;

- le troisième bloc code les groupes de pixels en sortie du quantificateur en exploitant le fait que certains groupes sont nettement plus fréquents que d'autres; à ce stade on fait généralement appel à l'une des méthodes générales décrites au chapitre précédent.

Le décodeur, quant à lui, ne dispose que de deux blocs (l'opération de quantification étant irréversible, il n'y pas de bloc de "déquantification") : le premier bloc restitue l'image transformée (ou une approximation de celle-ci) et le second l'image originale (ou une approximation de celle-ci).

Notons que dans un schéma de communication réel (cf paradigme de Shannon), la sortie de l'encodeur est encodée une deuxième fois pour lutter contre les bruits du canal (techniques similaires à celles discutées au chapitre 15), et il y a également un bloc "décodeur de canal" à la sortie de celui-ci.

Dans les applications (elles sont nombreuses) où on ne peut pas tolérer d'irréversibilité le bloc central de l'encodeur est supprimé. Il y a également des techniques de compression où la séparation entre transformation et codage est moins explicite. Nous allons expliquer ci-dessous intuitivement quelques techniques utilisées en pratique, afin de montrer la très grande diversité des solutions apportées au problème de compression d'images.

14.3.1 Dans le domaine réversible

14.3.1.1 Méthode d'ordre zéro. Ici on code l'image $f(x,y)$ directement, en utilisant un des algorithmes du chapitre précédent. En pratique, l'image est parcourrue ligne par ligne, et on code généralement les différences entre valeurs successives de $f(x,y)$, technique désignée sous le nom de *codage différentiel*.

Il existe des versions particulières de ces méthodes, applicables aux images binaires (deux niveaux de gris) : dans ce cas il est souvent plus efficace de segmenter l'image en groupes de symboles successifs identiques (suites de 0 ou de 1) dont on transmet la longueur.

Ces techniques de segmentation peuvent aussi se présenter sous-forme bi-dimensionelle. Elle consistent alors à représenter des plages de niveau constant à l'aide de diverses techniques de codage [GW93].

14.3.1.2 Plans de bits. Lorsque l'image est composée de plusieurs niveaux de gris on peut se ramener à plusieurs images binaires qui peuvent être codées comme ci-dessus. Par exemple, si les niveaux de gris sont donnés sur 8 bits, on peut décomposer l'image en 8 plans binaires dont chacun représente en chaque point (x,y) la valeur de l'un de ces 8 bits.

Il est possible de s'arranger (en utilisant un code spécial pour les nombres entiers, décrit ci-dessous) pour que les plans de bits ainsi obtenus soient composés de grandes plages constantes, sous l'hypothèse où les niveaux de gris varient lentement d'un point à l'autre de l'image. Par exemple, le code de Gray est un code binaire qui est tel que les mots associés à deux nombres entiers qui diffèrent de 1 ne diffèrent que d'un seul bit. En codant d'abord les niveaux de gris à l'aide du code de Gray, puis en utilisant la technique des plans de bits on obtient alors une décomposition de l'image originale en plans qui peuvent se coder efficacement à l'aide des techniques de segmentation mentionnées ci-dessus.

Code de Gray. Soit un nombre entier $k < 2^m$ représenté sous forme binaire classique $a_{m-1} \cdots a_0$:

$$k = \sum_{i=0}^{m-1} a_i 2^i.$$

Le code de Gray associe alors au mot $a_{m-1} \cdots a_0$ le code $g_{m-1} \cdots g_0$ défini récursivement par

$$\begin{aligned} g_i &= a_i \oplus a_{i+1}, \forall i = 0, \ldots, m-2 \\ g_{m-1} &= a_{m-1}. \end{aligned}$$

On peut se convaincre que dans ce code, deux nombres entiers k_1 et k_2 tels que $k_1 - k_2 = \pm 1$ recoivent des mots de code qui ne diffèrent que d'un seul bit.

14.3.1.3 Codage prédictif. Cette technique est utile de manière générale pour le codage de signaux numériques. Il s'agit d'une version adaptée aux signaux numériques des techniques d'ordre supérieur décrites au chapitre suivant.

Supposons que nous disposions d'un modèle qui nous permette de deviner avec une bonne précision la valeur d'un pixel, à partir des pixels déjà encodés (décodés). Appelons $\hat{f}_n$ la valeur ainsi prédite et soit e_n l'erreur de prédiction, c'est-à-dire la différence entre la valeur f_n, effectivement présente dans notre image, et cette prédiction :

$$f_n = \hat{f}_n + e_n. \tag{14.18}$$

Si notre modèle de prédiction était parfait, on aurait $e_n = 0, \forall n$. S'il est seulement bon, e_n sera petit comparé à la valeur de f_n. Pour une précision donnée de représentation de f_n, on pourra donc tabler sur le fait que le nombre de bits nécessaires pour encoder de façon compacte e_n est plus faible que celui nécessaire pour encoder f_n.

Les techniques de codage prédictif exploitent cette idée. On peut en imaginer autant de variantes qu'il est possible d'imaginer de variantes des modèles de prédiction. Une technique simple repose sur la prédiction linéaire : $\hat{f}_n$ est fourni par une combinaison linéaire des valeurs des pixels voisins déjà encodés (par exemple ceux qui sont à gauche ou au dessus du pixel courant, si l'image est parcourue dans le sens habituel). Notons que le codage différentiel est un exemple de codage prédictif simpliste, où la prédiction est obtenue simplement par la valeur du pixel précédent.

Signalons que le codage prédictif est particulièrement intéressant dans le cas d'images vidéo. Dans ce cas, le modèle prédictif peut se baser sur les valeurs des pixels correspondants dans les images qui précèdent l'image courante dans la séquence vidéo.

Notons également que le modèle prédictif joue ici le rôle du premier bloc de notre système d'encodage. Le troisième bloc consistant à coder les erreurs d'approximation.

Enfin, notons que dans le présent contexte les techniques usuelles consistent à *apprendre* le modèle prédictif à partir de l'image elle-même. Il faut alors transmettre ce modèle au décodeur et se pose alors le problème du compromis entre la complexité du modèle prédictif (dont la transmission est nécessaire) et l'entropie des erreurs de prédiction. Sans entrer dans les détails, signalons que ce compromis joue un rôle très important en apprentissage automatique, et qu'il existe un principe en apprentissage automatique qui vise à minimiser la *longueur totale de représentation*, c'est-à-dire le nombre total de bits qu'il faut transmettre au décodeur (modèle prédictif + erreurs résiduelles).

14.3.2 Dans le domaine irréversible

14.3.2.1 Codage prédictif irréversible. On ajoute au système discuté ci-dessus un bloc de quantification des erreurs e_n. Il existe alors tout un pan du traitement de signal qui vise à construire des systèmes prédicteurs + quantificateurs optimaux. Nous n'en dirons pas plus ici.

14.3.2.2 Utilisation des transformées d'images. Typiquement, les transformées ne sont pas appliquées sur la totalité de l'image (voir également la discussion ci-dessous concernant les ondelettes). En effet, l'application de transformées au niveau local permet de décorréler plus efficacement les pixels voisins.

La quantification opère alors sur les morceaux d'images transformées en se permettant une approximation plus grossière des composantes (genéralement haute fréquence) dont on sait qu'elles sont en général de faible amplitude et ne sont pas perçues finement par le système psychovisuel.

Enfin, le résultat est compressé à l'aide des techniques réversibles du chapitre précédent.

14.3.3 Standards de compression d'image

14.3.3.1 Images binaires. La plupart de ces standards furent intialement conçus pour la transmission de FAX. On y trouve essentiellement des techniques uni-dimensionnelles basées sur les idées décrites ci-dessus et bi-dimensionnelles (voir [GW93]).

14.3.3.2 Images à niveaux continus. On retrouve ici les standards JPEG (établis conjointement par l'ISO et le CCITT). Ce standard comprend un schéma irréversible basé sur l'utilisation de la transformée en cosinus, où la précision est limitée à 8 bits. L'image est décomposée en blocs 8×8 qui sont transformés en cosinus dont les coefficients sont encodés à l'aide de la méthode Huffman. Nous renvoyons également le lecteur aux références [GW93, HHJ97] pour les détails.

Le standard JPEG comprend également deux autres spécifications qui correspondent à des exigences de qualité plus strictes (la dernière est réversible).

14.3.3.3 Images séquentielles et couleur. Nous mentionnons les standards MPEG (Motion pictures expert group) mis au point par la CCITT et l'ISO. Ces standards étendent la technique basée sur la transformée en cosinus pour permettre l'exploitation de redondances inter-images, en utilisant des modèles prédictifs tels ceux que nous avons déjà discutés ci-dessus.

14.4 UNE BREVE INTRODUCTION AUX ONDELETTES

Ce dont nous allons parler dans cette dernière section est réellement le problème de représentation du signal. Nous voudrions trouver un moyen de le représenter de façon compacte et précise à la fois, sous la forme d'une combinaison linéaire de fonctions de base en nombre aussi réduit que possible.

Dans cette optique, c'est un peu comme si nous jouions au Légo avec comme briques élementaires nos fonctions de base. Evidemment, tout qui a déjà joué au Légo sait que l'ensemble idéal de briques est très dépendant de ce qu'on veut construire. Il nous faut donc un ensemble de briques suffisamment riche pour pouvoir construire toutes sortes d'images avec un nombre minimal de briques. Comme nous l'avons déjà illustré à la section §14.1.2, l'invention du Légo dans le domaine du traitement d'images n'est pas récente. Mais, les transformées discutées précédemment ont été essentiellement conçues dans l'optique de faciliter certaines opérations sur les images, et beaucoup moins dans le but explicite de faciliter la représentation.

Or la représentation d'une image est très liée à l'interprétation que nous en faisons lorsque nous la regardons. Nous "voyons" mieux certaines caractéristiques que d'autres, et le Légo idéal de ce point de vue devrait comporter des briques toutes faites pour représenter les morceaux d'une image que nous voyons. Or une des caractéristiques importantes de nombreuses images (et signaux temporels) est l'existence de phénomènes transitoires à courte portée spatiale : par exemple les discontinuités (bords) ou les changements locaux de texture sont assez fréquents mais se représentent difficilement à l'aide d'un nombre réduit de briques "périodiques" telles que celles utilisées dans les transformées discutées ci-dessus.

Afin de remédier à cette situation, différentes approches ont été proposées dans le passé, telles que l'utilisation de transformées périodiques de façon locale (cf. JPEG discuté ci-dessus) ou la combinaison des fonctions de base de Fourier avec des fenêtres spatiales.

Plus récemment, de nombreux développements en traitement du signal se sont fondés sur une famille de fonctions de base appelées ondelettes. Nous allons ici en donner une introduction très brève et intuitive, en recommandant les références [Mey93, Dau92] au lecteur intéressé par ce domaine.

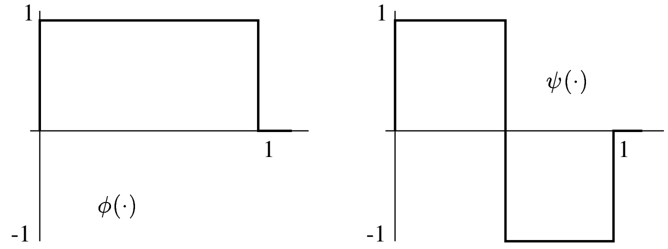


Figure 14.4. Briques élementaires pour la construction des ondelettes de Haar

Dans le cas de signaux temporels, on appelle *ondelette* un signal oscillant de durée limitée. De même, dans le cas des images, on appelle ondelette une image de portée spatiale limitée. Une base d'ondelettes est obtenue par translation et mise à l'échelle d'une seule fonction de base. Pour le besoin de notre explication, nous allons nous focaliser ici sur l'exemple très simple des ondelettes de *Haar*. La figure 14.4 représente les deux composantes qui permettent de construire cette famille d'ondelettes, à savoir une fonction de mise à l'échelle

$$\phi(x) = \begin{cases} 1, & \text{si } 0 \leq x < 1, \\ 0, & \text{sinon} \end{cases} \tag{14.19}$$

et une "ondelette" mère

$$\psi(x) = \begin{cases} 1, & \text{si } 0 \leq x < \frac{1}{2}, \\ -1, & \text{si } \frac{1}{2} \leq x < 1, \\ 0, & \text{sinon.} \end{cases} \tag{14.20}$$

Notons qu'en discrétisant l'intervalle $[0, 1]$ en deux parties de même longueur, ces deux fonctions peuvent être vues comme deux vecteurs de $\mathbb{R}^2$:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}. \tag{14.21}$$

Elles définissent donc un noyau pour une transformée d'images 2×2. De toute évidence ces vecteurs sont orthogonaux (non-normés). Pour obtenir une base d'images 4×4 on peut compléter ces fonctions par deux fonctions obtenues par mise à l'échelle et translation de ψ

$$\psi_0^1(x) = \psi(2x) \text{ et } \psi_1^1(x) = \psi(2x - 1). \tag{14.22}$$

En divisant par deux l'intervalle de "discrétisation" de x, on obtient alors les quatre vecteurs suivants:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}. \tag{14.23}$$

Ces vecteurs peuvent évidemment être normalisés pour produire une base orthonormée de $\mathbb{R}^4$. Par ailleurs, rien ne nous empêche de continuer notre processus de discrétisation en divisant encore nos intervalles par deux. Au total, si nous répétons k fois cette opération, nous aboutissons à 2^{k+1} fonctions de base orthogonales qui peuvent être orthonormées pour former une base de $\mathbb{R}^{2^{k+1}}$. Ces fonctions peuvent être désignées en utilisant deux indices i et j

$$\psi_j^i = \psi(2^i x - j), \text{ avec } j = 0, 1, \ldots, 2^i - 1, \tag{14.24}$$

laquelle notation implique que $\psi(x) = \psi_0^0(x)$.

Ayant construit les fonctions (vecteurs) de base, nous pouvons en déduire la transformée bi-dimensionnelle de Haar. Nous attirons l'attention du lecteur sur le fait que la matrice $\boldsymbol{G}_{\text{Haar}}$ n'est pas symétrique, contrairement aux autres transformées mentionnées plus haut.

Essayons de mettre en évidence comment cette base opère à la fois dans le domaine spatial et fréquentiel. On voit que les fonctions de base dont la fréquence est élevée, sont nulles en dehors d'une plage limitée dans le domaine spatial. Les coefficients correspondants dans l'image transformée ne seront donc fonction que des

Figure 14.5. Images compressées par ondelettes (facteurs de compression)

valeurs des pixels dans cette région limitée. En d'autres termes, ces ondelettes extrayent de l'information sur les changements locaux. D'autre part, pour une fréquence donnée, on dispose exactement du nombre de fonctions de base nécessaire pour couvrir complètement l'image : seules les parties de l'image où se trouvent les “discontuités” donneront lieu à des termes haute fréquence. On peut donc s'attendre à ce que ces briques soient bien adaptées pour représenter des informations de haute fréquence localisées dans une partie spatiale limitée de l'image, telles que des bords ou des changements de texture.

Un schéma de compression irréversible peut être obtenu de la manière suivante:

1. soit l'image $f(x, y)$ et une tolérance $\epsilon < 0$.

2. calculer la transformée $T_{\text{Haar}} = \boldsymbol{G}_{\text{Haar}}^T \boldsymbol{F} \boldsymbol{G}_{\text{Haar}}$ de l'image et arrondir à zéro tous les coefficients inférieurs à ϵ.

3. encoder l'image transformée au moyen d'une des techniques réversibles discutées plus haut.

Du point de vue des calculs, il faut remarquer que la grande majorité des éléments de l'image transformée ne fait intervenir qu'un nombre très limité de pixels de l'image de départ. Les calculs, tenant compte de ce caractère creux peuvent donc se faire en principe efficacement sans trop de difficultés. Cela étant, le caractère récursif des vecteurs de base permet en fait l'utilisation d'algorithmes récursifs rapides du type de ceux utilisés pour la transformée de Fourier rapide. La figure 14.5 montre trois images obtenues par cette technique.

14.5 RESUME

Dans ce chapitre nous avons voulu soulever le voile sur le domaine passionnant du traitement d'images. Notre approche a été plus physique que mathématique, ceci étant justifié par le fait que l'image est un domaine ou l'intuition physique est à la fois facile (parce que nous sommes tous habitués à regarder des images) et d'une grande utilité.

Cependant, nous avons pu nous rendre compte que les techniques introduites dans les chapitres précédents sont également utilisées en compression d'images. De nouveau nous avons vu apparaître la dualité construction/exploitation de modèles, qui est omniprésente dans le contexte des méthodes stochastiques, et plus généralement dans le domaine de l'ingénieur.

Notre objectif dans ce chapitre était d'indiquer aux étudiants les concepts et méthodes fondamentales qui sont à la base du traitement d'images tout en leur montrant les techniques utilisées à l'heure actuelle.

Une différence importante entre la structure des images et celle des messages de sources discrètes est que l'alphabet est ici composé de valeurs numériques continues. C'est également le cas dans un grand nombre d'autres domaines (son, instrumentation physique. . .). Nous en avons profité pour introduire quelques-unes des techniques de traitement de signal utilisées pour manipuler des signaux de cette nature : transformées, modèles prédictifs, techniques de quantification. Nous en avons également profité pour discuter de techniques irréversibles, car c'est dans le domaine du signal continu que ces techniques peuvent sans doute avoir le plus d'intérêt. Ceci est lié au fait qu'un signal continu ne peut de toutes façons ni être représenté ni être transmis avec une précision infinie, à moins de disposer des resources physiques illimitées. Par ailleurs, et heureusement, dans bon nombre d'applications les exigences de précision sont suffisamment lâches pour que les techniques de compression irréversible puissent être exploitées avec un gain économique considérable.

15 LUTTE CONTRE LE BRUIT DE CANAL

Note. Malgré de nombreux efforts, ce chapitre reste à l'heure actuelle dans une version provisoire. Nous avons cherché à motiver de manière pratique les différentes approches au codage de canal, sans pour autant sacrifier la pureté du raisonnement. Le domaine étant vaste et souvent ardu du point de vue théorique, le résultat peut paraître indigeste, dans l'état actuel des notes.

Ces notes seront remises à jour pour l'année académique suivante (2000-2001) et nous invitons les étudiants intéressés à se procurer une nouvelle version en temps voulu (p.ex. une version électronique à partir du site WEB du service de Méthodes Stochastiques).

Louis WEHENKEL
Décembre 1999.

15.1 INTRODUCTION

Le second théorème de Shannon, publié en 1948, montre comment en principe le codage aléatoire permettrait d'atteindre la capacité d'un canal bruité, avec un taux d'erreurs arbitrairement faible. Cependant, bien que presque tous les codes aléatoires générés de cette façon sont en principe bons, il est nécessaire d'utiliser des longueurs de blocs assez importantes, ce qui conduit à des tables de code de taille gigantesque (la taille croît exponentiellement avec la longueur des blocs, et cela d'autant plus que le débit souhaité est grand). En conséquence, le décodage devient pratiquement irréalisable.

C'est la raison pour laquelle, depuis cette publication, de nombreux travaux ont porté sur la mise au point de codes de canal structurés de manière à en rendre le décodage faisable. Mais, contrairement au domaine de la compression de données où les limites promises par Shannon sont pratiquement atteintes dans de nombreux cas, le codage de canal n'a pas encore atteint ce niveau de maturité à l'heure actuelle. Par exemple, il n'existe pas à l'heure actuelle de méthode universelle de codage de canal, qui pourrait approcher asymptotiquement les limites de Shannon.

Cependant, les recherches ont donné lieu à de nombreux codes qui permettent de réduire significativement les taux d'erreurs tout en ne sacrifiant pas complètement les débits.

Dans ce cours, nous ne pourrons donner qu'un aperçu très limité des approches actuellement utilisées pour le codage de canal. Plutôt que de discuter en détails l'état de l'art dans ce domaine, nous allons essayer d'introduire les concepts principaux qui interviennent dans ce contexte. Après avoir posé le problème, nous commencerons par une discussion plus détaillée de la raison pour laquelle il est si difficile de trouver de bons codes de canal qui soient efficacement décodables. Ensuite, nous décrirons dans les grandes lignes les méthodes de codage algébriques, qui reposent essentiellement sur l'exploitation des structures mathématiques de corps fini et d'espace linéaire défini sur un corps. Nous discuterons ensuite des différents modes d'utilisation possibles du canal Gaussien en fonction du rapport signal bruit. Cela nous permettra de motiver d'une part l'utilisation d'alphabets de grande taille en cas de rapport signal bruit élevé (p.ex. modems pour lignes téléphoniques), d'autre part l'utilisation de codes binaires et du décodage par décisions souples si le rapport signal bruit est faible (communications spatiales, canaux à large bande). Nous introduisons ensuite les codes convolutionnels et leurs techniques de représentation et de décodage graphiques.

Nous terminerons ce chapitre en donnant un aperçu de deux techniques de codage plus récentes, les codes dans l'espace Euclidien et les turbo-codes. Toutes deux ont déjà donné lieu à de nombreuses applications et s'adressent aux deux conditions extrêmes à savoir les systèmes à haut rapport signal bruit (les codes sur les espaces Euclidiens) et aux canaux à faible rapport signal bruit (ou à large bande passante par rapport au rapport P/N_0), à savoir les turbo-codes.

Afin d'alléger le texte, nous avons collationné dans l'appendice E les quelques notions de structures algébriques (groupes, corps, espaces vectoriels) nécessaires pour la bonne compréhension de ce qui suit. Nous fournissons également à la fin de ce chapitre un supplément relatif aux polynômes définis sur un corps fini et aux corps de Gallois.

15.2 TERMINOLOGIE ET NOTATIONS

Dans les chapitres 9 et 10 nous avons démontré le second théorèmes de Shannon en faisant appel à la notion de débit de code R. Le débit d'un code construit sur un alphabet de canal de taille quelconque (éventuellement infini) relie le nombre de mots du code et la longueur de ceux-ci par la formule

$$M = \lfloor 2^{nR} \rfloor.$$

Dans la suite de ce chapitre nous supposerons la plupart du temps que les mots de code correspondent à des messages source codés de façon binaire sur k bits (c'est sous cette forme que la plupart des sources émettent leurs messages). On a donc $M = 2^k$. Par conséquent le *débit du code* sera donné par la formule

$$R_{\mathcal{C}} = \frac{k}{n}, \tag{15.1}$$

où n désigne toujours le nombre de symboles de canal utilisés pour construire les mots de code. En pratique l'alphabet du canal n'est pas nécessairement binaire (cf par exemple le canal Gaussien qui utilise un alphabet de taille non-dénombrable). Si l'alphabet du canal est binaire alors la capacité est au maximum égale à 1. En général, si l'alphabet de canal est de taille q, on a

$$C \leq \log q. \tag{15.2}$$

Le second théorème de Shannon dit alors qu'il est possible de communiquer sur un canal de manière aussi fiable que l'on veut, à condition que

$$R_{\mathcal{C}} < C_{\text{canal}},$$

où C_{canal} désigne la capacité en information du canal par symbole transmis sur celui-ci. On peut interpréter cette limite en disant qu'il faut transmettre au moins $\frac{1}{C_{\text{canal}}}$ symboles de canal par bit source pour communiquer de manière fiable. On suppose ici que la source est déjà compressée de manière idéale, et que donc les symboles source sont a priori équiprobables et indépendants.

En pratique, les canaux sont caractérisés, d'une part, par leur capacité en information C_{canal} (en Shannon/symbole), d'autre part, par leur débit D_{canal} (ou vitesse, en symboles/seconde). Le produit des deux donne alors le nombre de bits source par seconde qu'on peut transmettre de manière arbitrairement fiable sur ce canal.

Dans la mesure où aucun code réel n'est parfait, et qu'en pratique il n'est pas non plus nécessaire d'atteindre la perfection, il est certainement intéressant de chercher à quantifier les limites du possible pour des niveaux de probabilité non nuls. D'un point de vue théorique, c'est la théorie de la distorsion (voir chapitre 11) qui permet de formaliser le compromis entre le débit de code atteignable et la probabilité d'erreur de décodage.

Plus précisément, cette théorie nous apprend (entre autres) le compromis qui existe entre le nombre de bits nécessaires au codage *irréversible* d'une source binaire et la probabilité d'erreur moyenne par bit source. Le rapport entre le nombre de bits n d'un code irréversible nécessaire pour coder un message binaire de longueur k, sous la contrainte que la probabilité d'erreur par symbole source ne dépasse pas une valeur spécifiée P_e, est appelé débit de distorsion et noté $R_{\text{dist}}(P_e)$.

Le troisième théorème de Shannon (chapitre 11) dit que pour des messages longs ($n \to \infty$) la borne supérieure de ce débit tend vers

$$R_{\text{dist}}(P_e) = H(S) - H_2(P_e), \tag{15.3}$$

où P_e désigne la probabilité d'erreur moyenne par bit source (nous utiliserons aussi le terme de "bit d'information" dans la suite).

Dans le contexte du codage de canal, nous nous permettons de particulariser cette équation en supposant que la source émet des symboles binaires indépendants et équiprobables (il s'agit d'une source binaire "blanche"). On obtient

$$R_{\text{dist}}(P_e) = 1 - H_2(P_e). \tag{15.4}$$

La mise en commun du second et du troisième théorème de Shannon permet alors de calculer le débit de canal maximum pour une probabilité d'erreur par bit source spécifiée. On a, par utilisation du canal

$$D_{\mathcal{S}}^{\max}(P_e) = \frac{C_{\text{canal}}}{R_{\text{dist}}(P_e)} = \frac{C_{\text{canal}}}{1 - H_2(P_e)} \text{bits source/utilisation du canal}. \tag{15.5}$$

En tenant compte du débit du canal D_{canal} on obtient aussi

$$D_{\mathcal{S}}^{\max}(P_e) = D_{\text{canal}} \frac{C_{\text{canal}}}{R_{\text{dist}}(P_e)} = \frac{D_{\text{canal}} C_{\text{canal}}}{1 - H_2(P_e)} \text{bits source/seconde}. \tag{15.6}$$

On voit que si on tolère une probabilité d'erreur nulle le débit maximum devient égal au produit $D_{\text{canal}} \times C_{\text{canal}}$, et que si on tolère une probabilité d'erreur égale à $\frac{1}{2}$ le débit source maximal devient infini.

En pratique on est souvent amené à caractériser des codes de canal dont on connaît le débit R (par exemple parce qu'on donne k et n) et dont on peut évaluer (ou calculer) la probabilité d'erreur par bit. On peut alors comparer ces codes avec la limite de Shannon en utilisant la formule

$$R_{\mathcal{C}}^{\max} = \frac{C_{\text{canal}}}{1 - H_2(P_e)}, \tag{15.7}$$

pour déterminer le débit maximum possible pour le niveau de fiabilité du code, et voir dans quelle mesure le code est sous-optimal en terme de débit. De même on peut calculer la probabilité d'erreur minimale atteignable au débit spécifié par le code en résolvant l'équation

$$H_2(P_e^{\min}) = \max\left\{0, 1 - \frac{C_{\text{canal}}}{R_{\mathcal{C}}}\right\} \tag{15.8}$$

par rapport à $P_e^{\min}$ dans l'intervalle $[0; 0.5]$, et voir dans quelle mesure le code est sous-optimal en terme de probabilité d'erreur.

Exemple. La figure 15.1 montre graphiquement la limite de Shannon pour un canal binaire symétrique avec une probabilité d'erreur de transmission $p = 0.1$. On voit que la région atteignable est délimitée pour les hauts niveaux de fiabilité par une asymptote verticale passant par la capacité du canal ($C = 1 - H_2(p) = 0.53$). Pour les faibles niveaux de fiabilité elle est délimitée par une asymptote horizontale passant par la probabilité d'erreur $P_e = 0.5$: en supposant que les bits source sont équiprobables le récepteur peut deviner le message émis par la source avec une probabilité d'erreur qui vaut 0.5, sans recevoir aucune information en sortie du canal.

On a également représenté sur cette figure les points correspondant à trois façons différentes de coder l'information avant de la transmettre sur le canal

1. Sans code : on obtient évidemment un débit $R = 1$ et une probabilité d'erreur $P_e = p = 0.1$.
2. Un code de répétition triple : cela consiste à envoyer chaque symbole source recopié trois fois et à décoder les symboles selon la règle de majorité; le débit est $R = 1/3$ et la probabilité d'erreur (voir cours oral) vaut $P_e = 0.028$.

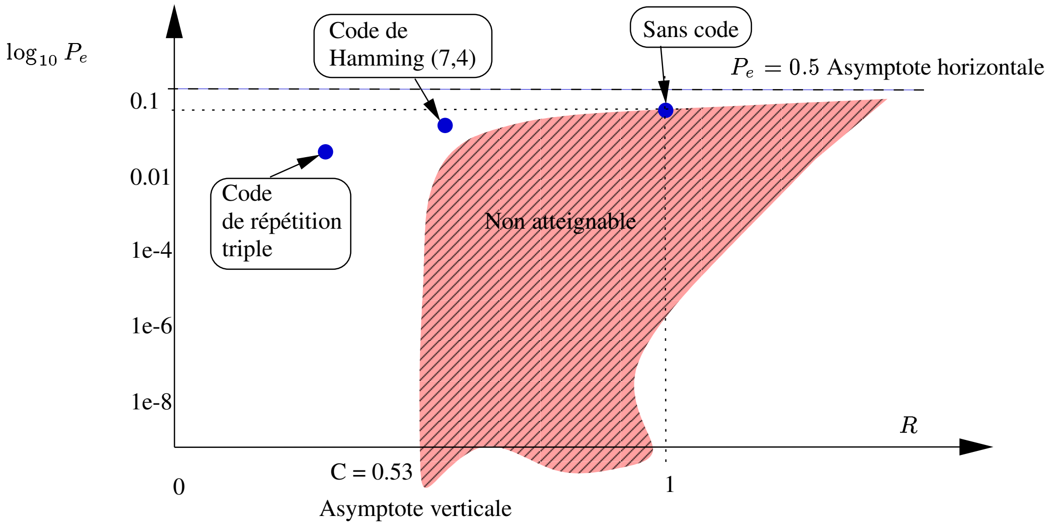


Figure 15.1. Compromis débit vs probabilité d'erreur pour le canal symétrique binaire (avec $p = 0.1$)

3. Le code de Hamming $(7, 4)$ (expliqué dans la suite) : son débit est $R = 4/7$ et sa probabilité d'erreur par bit source est $P_e = 0.07$.

Nous verrons à la fin de ce chapitre un graphique similaire qui montrera comment les différentes techniques de codage de canal que nous allons discuter dans la suite se placent par rapport à la limite de Shannon.

15.3 DE LA DIFFICULTE DU DECODAGE OPTIMAL

Nous avons vu aux chapitres 9 et 10 que pour approcher la limite de Shannon (i.e. $R_{\mathcal{C}} \approx C_{\text{canal}}$) il est nécessaire de faire appel à des espaces de signaux de grande dimension (pour que la loi des grands nombres agisse efficacement).

Comme nous venons de le souligner, cela veut dire que n doit être grand et donc a fortiori M (qui croît exponentiellement avec n pour les bons codes).

Le code étant donné, le décodage optimal nécessite donc la recherche d'un mot parmi $\approx 2^{nC}$ candidats, que ce soit par la règle MAP (maximum de probabilité a posteriori) ML (maximum de vraisemblance) ou, le cas échéant MHD (distance de Hamming minimale, décrit dans la suite).

En toute généralité, c'est-à-dire sans faire aucune hypothèse sur la structure de l'ensemble de mots de code, seule la méthode de décodage énumérative permet de résoudre ce problème de façon exacte[1]. Par conséquent, comme la complexité de calcul de cette méthode est exponentielle, nous avons seulement trois choix possibles pour rendre le décodage faisable :

1. Utilisation de codes courts : en pratique choix de $n \leq 30$.

2. Décodage heuristique : utilisation d'un algorithme efficace, mais sous-optimal.

3. Utilisation de codes structurés : pour lesquels des algorithmes efficaces existent.

En pratique, ces trois voies ont donné lieu à des nombreuses recherches et d'aussi nombreuses publications. Il semble aujourd'hui que c'est la combinaison de ces trois approches qui offre les perspectives les plus intéressantes [Bat97, BPJ$^+$98].

Dans la suite nous allons discuter principalement trois voies qui ont été explorées intensivement dans la recherche sur le codage de canal.

[1] Si n n'est pas trop grand, il est cependant possible de remplacer la recherche énumérative du plus proche voisin par un adressage direct dans une table de taille q^n qui contient pour chaque mot Y^n possible le mot de code le plus proche.

- L'approche algébrique : cette approche vise à construire des codes fortement structurés du point de vue mathématique, ce qui en facilite d'une part le décodage, d'autre part la démonstration d'un certain nombre de garanties en termes de capacité de détection et de correction d'erreurs. Cette approche repose fortement sur l'utilisation de notions de mathématiques (structures algébriques discrètes) assez complexes, et sans doute nouvelles pour les étudiants. Nous ne pourrons par conséquent qu'entrevoir une partie relativement faible de l'iceberg dans le cadre limité de ce cours. Notons que l'approche algébrique est intimement liée au modèle de canal discret.

  L'approche algébrique a donné lieu aux codes de Reed-Solomon, qui constituaient encore récemment le couronnement du codage pour canaux bruités. A titre d'exemple on peut citer les systèmes d'enregistrement numériques sur disques compact qui utilisent des codes de Reed-Solomon concaténés qui permettent de corriger des rafales de 4000 erreurs de lecture successives.

- L'approche probabiliste : dans sa démarche, cette approche est plus intimement liée à la théorie de l'information. L'idée est ici de construire des codes ayant des propriétés similaires au codage aléatoire utilisé par Shannon pour démontrer le second théorème. Sachant que ce genre de code risque a priori d'être difficile à décoder, une partie de la recherche dans ce domaine a eu pour objectif de construire des algorithmes de décodage sous-optimaux (mais pas trop) et efficaces. L'approche probabiliste est bien adaptée au modèle de canal continu et donc susceptible de mieux exploiter la plupart des canaux réels que l'approche algébrique.

  Cette approche a été réinvestiguée à plusieurs reprises dans le passé, mais on peut dire que l'invention des turbo-codes au début des années 1990 en constitue actuellement le couronnement, à tel point que ces derniers se retrouvent à l'heure actuelle dans un bon nombre de spécifications de nos futures normes de télécommunications.

- L'approche géométrique : elle est en rapport avec les interprétations géométriques utilisées dans le cadre du canal Gaussien. Il s'agit ici de développer des codes qui exploitent bien le canal Gaussien à faible niveau de bruit, qui requièrent l'utilisation d'un alphabet de code de grande taille, comme nous le verrons.

  Ici les progrès sont également assez récents et reposent sur l'utilisation de codes constitués par des vecteurs dans un espace Euclidien de grande dimension, organisés selon une structure de réseau cristallin. On peut retrouver ce genre de code dans le cadre de l'utilisation de modems à hautes performances.

Notre discussion commence par le codage/décodage algébrique, ensuite nous discuterons les implications des données relatives au canal Gaussien sur le type de système de codage. Cela nous permettra de trouver une bonne motivation pour aller au-delà du codage algébrique. Nous discuterons ensuite les deux approches adaptées aux canaux continus, à savoir les codes convolutionnels et le codage en espace Euclidien. Nous ferons ensuite quelques remarques sur différentes manières de combiner des codes, et terminerons le chapitre par une introduction aux turbo-codes qui mettent en oeuvre la plupart des outils introduits dans ce chapitre.

Le lecteur attentif remarquera que l'ensemble des codes étudiés dans ce chapitre sont linéaires, et se demandera à juste titre pourquoi il en est ainsi. A ce sujet il faut faire deux remarques : (i) tous les codes ne sont pas linéaires (mais ceux que nous étudions ici le sont tous); (ii) il existe une version du second théorème de Shannon qui dit qu'il est possible d'atteindre le débit maximal à erreur arbitrairement faible, même si on se restreint aux codes linéaires.

15.4 CANAL DISCRET - CORRECTION ET DETECTION D'ERREURS

La figure 15.2 décrit le schéma de communication dans le cas où le canal est modélisé de façon discrète.

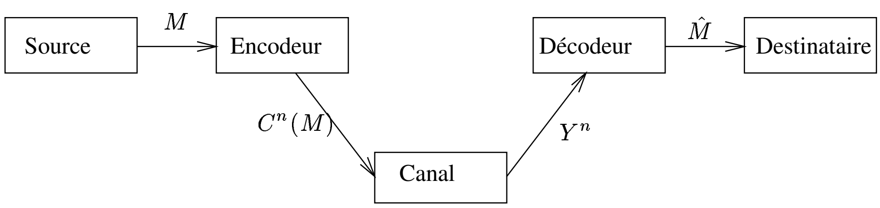


Figure 15.2. Codage de canal

Une source produit un message constitué d'une suite de symboles de source qui doit être transmis au récepteur au moyen d'un canal bruité. Sans perte de généralité, nous allons supposer que les alphabets d'entrée et de sortie du canal sont identiques (nous désignons cet alphabet par l'ensemble de symboles $\mathcal{X}$, supposé de taille q). Un code $\mathcal{C}$ sur $\mathcal{X}$ est alors un ensemble de mots formés à l'aide des symboles de $\mathcal{X}$. Comme usuellement, nous désignerons par $\mathcal{X}^n$ l'ensemble de tous les mots de longueur n qui peuvent être construits à l'aide de l'alphabet de canal (nous désignerons aussi cet ensemble par le terme *espace de signaux*) et nous supposerons que tous les mots de notre code ont même longueur (n). L'utilisation de tels codes (appelés codes en blocs) rend le décodage nettement plus facile. Le code $\mathcal{C}$ est alors constitué par un ensemble de M mots C^n de longueur n, et le problème sera de choisir l'ensemble $\mathcal{C}$ de façon à minimiser la probabilité d'erreur de décodage et d'autre part permettre un décodage efficace.

Pour ce qui concerne notre discussion ci-dessous, nous allons considérer que le bruit de canal agit en restituant avec une probabilité assez grande les symboles d'entrée à sa sortie. Nous supposerons également que ce processus de choix d'un symbole de sortie en fonction du symbole d'entrée est stationnaire et sans mémoire. Il est donc caractérisé par une matrice de probabilités conditionnelles $P(Y_j|X_i)$, $(X_i, Y_j \in \mathcal{X})$, où X_i désigne un symbole écrit à l'entrée du canal et Y_j le symbole correspondant lu en sortie.

Notons que dans la réalité un canal est souvent de nature continue et on obtient un canal discret en y ajoutant des dispositifs de modulation et de démodulation numériques. Dans ce cas, il est possible de placer le décodeur soit en sortie du canal continu ou bien en sortie du démodulateur. Comme le démodulateur consiste à prendre une décision symbole par symbole sur base du nombre réel correspondant reçu, il conduit normalement à une perte d'information, qui sera d'autant plus grande que le rapport signal bruit est faible. Donc, en principe le décodage sur base des nombres réels reçus peut donner lieu à une probabilité d'erreur de décodage plus faible (voir aussi chapitre 10).

Dans ce qui suit, nous allons cependant d'abord nous focaliser sur le modèle de canal discret de la figure 15.2. Nous analyserons le décodage à partir de sorties continues à la fin de ce chapitre.

15.4.1 Décodage à distance de Hamming minimale

On définit la distance de Hamming $\mathrm{d}_H(X^n, Y^n)$ entre deux mots X^n et Y^n de longueur n par le nombre total d'indices $i \leq n$ tels que $X_i \neq Y_i$.

Le décodage à distance de Hamming minimale consiste alors à choisir pour un vecteur reçu Y^n un mot de code $C^n \in \mathcal{C}$ tel que $\mathrm{d}_H(C^n, Y^n)$ est minimale.

Sous certaines conditions, le décodage à distance de Hamming minimale conduit à une probabilité d'erreur de décodage minimale (voir ci-dessous). Cependant, nous allons ici nous intéresser à la capacité de correction d'un nombre limité d'erreurs, l'idée étant que si le bruit est suffisamment faible, la probabilité d'avoir un grand nombre de symboles corrompus sera faible.

La *distance minimale* du code est par définition la distance minimale entre paires de mots différents appartenant à $\mathcal{C}$:

$$\mathrm{d}_H(\mathcal{C}) = \min\{\mathrm{d}_H(C^n, C'^n)\ : C^n, C'^n \in \mathcal{C}, C^n \neq C'^n\}. \tag{15.9}$$

Théorème 1. *Si le code $\mathcal{C}$ a une distance minimale $\mathrm{d}_H(\mathcal{C}) = d$, alors le décodage à distance de Hamming minimale est capable de corriger $\lfloor \frac{1}{2}(d-1) \rfloor$ erreurs.*

Si un code comporte M mots de longueur n et a une distance minimale de d on parle de code (n, M, d). Il est clair que pour une valeur donnée de n, les paramètres M et d varient en sens opposé.

Désignons par $A_q(n, d)$ la valeur maximale de M telle qu'il existe un code (n, M, d). Evidemment, on a $A_q(n, 1) = q^n$ (en choisissant $\mathcal{C} = \mathcal{X}^n$). Mais, on ne dispose pas d'une méthode directe de calcul de $A_q(n, d)$, et même pour des valeurs relativement faibles de q, n et d, la valeur de $A_q(n, d)$ est inconnue. Par exemple, dans le cas binaire, la valeur de $A_2(10, 3)$ est inconnue. Seules les bornes suivantes sont connues à l'heure actuelle

$$72 \leq A_2(10, 3) \leq 79.$$

Définition : t-sphère. La t-sphère d'un mot de code C^n est l'ensemble de mots de longueur n noté $S_t(C^n)$ défini par

$$S_t(C^n) = \{X^n\ : \mathrm{d}_H(C^n, X^n) \leq t\}.$$

Nous verrons que la caractérisation des t-sphères d'un code permet d'en caractériser les capacités de correction d'erreurs ainsi que sa relative efficacité.

15.4.2 Codes équivalents

Il est clair qu'un code peut être représenté au moyen d'un tableau $M \times n$, dont les lignes correspondent aux M mots du code de longueur n. Nous appellerons ce tableau *la matrice du code*.

Il est assez évident également que les propriétés d'un code doivent rester inchangées si nous permutons les lignes ou bien les colonnes de sa matrice, ou bien encore si nous permutons les symboles de l'alphabet de canal. Nous dirons donc que deux codes $\mathcal{C}$ et $\mathcal{C}'$ sont *équivalents* si l'on peut passer de l'un à l'autre au moyen d'une combinaison de permutations des lignes ou de colonnes de leur matrice, et/ou de permutations de l'alphabet de canal.

On en déduit que si deux codes sont équivalents, alors leurs ensembles de distances intermots (l'ensemble d'entiers positifs ou nuls composé des distances entre paires de mots d'un code) sont nécessairement identiques.

D'autre part, si X^n est un mot quelconque et $\mathcal{C}$ un code de longueur n alors il existe un code équivalent à $\mathcal{C}$ qui contient le mot X^n (on peut transformer par exemple le premier mot du code en X^n au moyen d'au plus n permutations de symboles).

15.4.3 Codes parfaits

La situation idéale du point de vue économie se produit lorsqu'il existe une valeur de t telle que d'une part les t-sphères qui entourent les mots de code soient toutes disjointes, et que d'autre part leur union contienne tous les mots de $\mathcal{X}^n$. Un code qui vérifie cette propriété est dit *parfait*, parce qu'il exploite au mieux l'idée d'empilement de sphères.

Voici deux propriétés des codes parfaits.

Correction d'erreurs. Un code parfait permet de corriger t erreurs mais ne permet pas de corriger $t+1$ erreurs. (*Pourquoi ?*)

Restriction sur d. Si un code (n, M, d) est parfait, alors d est impair. (*Pourquoi ?*)

Suggestion.

1. Montrer que tout code trivial (c'est-à-dire qui ne contient qu'un seul mot de code) est parfait.

2. Montrer qu'un code binaire de longueur n, avec n impair, et qui ne contient que les deux mots $0000\cdots00$ *et* $1111\cdots11$ *est parfait. Ce type de code porte le nom de code de répétition.*

15.4.4 Décodage au maximum de vraisemblance

Nous savons, depuis le chapitre 9, que la stratégie de décodage optimale (qui minimise la probabilité d'erreur) consiste à décoder la séquence reçue Y^n en choisissant le mot de code C^n qui maximise la probabilité a posteriori

$$P(C^n|Y^n).$$

Puisque

$$P(C^n|Y^n) = \frac{P(Y^n|C^n)P(C^n)}{P(Y^n)},$$

lorsque les mots de codes sont distribués de façon équiprobale cette règle revient à choisir C^n qui maximise $P(Y^n|C^n)$, c'est-à-dire la règle de décodage *au maximum de vraisemblance*.

Dans l'hypothèse d'un canal sans mémoire on a

$$P(Y^n|C^n) = \prod_{i=1}^{n} P(Y_i|C_i).$$

En désignant par q la taille de l'alphabet du code, si nous supposons que $P(Y_i|C_i)$ vaut $p = 1 - (q-1)\epsilon$ lorsque $Y_i = C_i$ (symbole de sortie non corrompu) et vaut ϵ lorsque $Y_i \neq C_i$, on en déduit que

$$\log P(Y^n|C^n) = (n - \mathrm{d}_H(C^n, Y^n)) \log p + \mathrm{d}_H(C^n, Y^n)) \log \epsilon.$$

Par conséquent

$$\frac{\partial \log P(Y^n|C^n)}{\partial \mathrm{d}_H(C^n, Y^n)} = \log \frac{\epsilon}{p} = \log \frac{\epsilon}{1-(q-1)\epsilon},$$

et, pour autant que $\epsilon < 1/q$, la règle du maximum de vraisemblance est équivalente à choisir le mot de code C^n le plus proche de Y^n au sens de la distance de Hamming.

Il est clair que les conditions introduites ci-dessus pour montrer l'équivalence du décodage à distance de Hamming minimale et celui qui minimise la probabilité d'erreur sont assez restrictives. En particulier, il est facile d'imaginer des circonstances (i.e. des matrices de confusion de canal) telles que la probabilité d'erreur de décodage ne soit plus nécessairement croissante lorsque la distance de Hamming croît.

Cette remarque suggère une limitation intrinsèque des méthodes purement algébriques qui ont longtemps dominé dans le domaine du codage de canal. Le décodage à distance de Hamming minimale est évidemment justifié dans le cas d'un canal binaire symétrique. Mais ce type de canal constitue souvent une abstraction de la réalité qui serait mieux modélisée à l'aide d'un canal continu en sortie. Nous avons en effet vu au début du chapitre 10 que l'utilisation d'un canal binaire symétrique comme modèle conduit à une perte d'information en sortie (quantification) qui se traduit par une réduction sensible de la capacité si le rapport/signal bruit est élevé. Nous verrons plus loin dans ce chapitre que l'algorithme de Viterbi permet d'effectuer le décodage au maximum de vraisemblance sans faire appel nécessairement à un alphabet de sortie discret.

Nous suggérons au lecteur intéressé de consulter la référence [Bat97] pour une critique approfondie des méthodes de codage purement algébriques.

15.4.5 Discussion du codage aléatoire

La démonstration du second théorème de Shannon a fait appel au codage aléatoire comme technique magique permettant de générer avec une grande probabilité de bons codes. S'il est tellement facile de construire de bons codes, on peut alors se demander comment cela se fait qu'en pratique cette méthode ne fonctionne pas.

La difficulté du codage aléatoire est liée au problème de non faisabilité computationnelle. En effet, la bonne qualité des codes est "garantie" seulement pour les grandes valeurs de n. Or, on sait que le nombre M de mots de codes nécessaires pour approcher la limite de Shannon croît exponentiellement avec n (le taux étant donné par la capacité du canal).

Dès lors, comme la recherche d'un bon code nécessite le calcul de la probabilité d'erreur de décodage d'un certain nombre de codes candidats (au moins un), l'effort de calcul lié à la sélection d'un bon code aléatoire croît également de façon exponentielle.

On pourrait objecter que pour un canal donné ce travail ne doit être fait qu'une seule fois, et qu'une méthode "off-line" permettrait de trouver un bon code, quitte à consacrer suffisamment de puissance de calcul. Cependant, quelle que soit la puissance de calcul disponible, la croissance exponentielle limitera très fortement la longueur du code pouvant être ainsi produit.

Par ailleurs, même si par chance nous pouvions produire un bon code aléatoire en un temps raisonnable, le décodage de celui-ci resterait a priori difficile, car de complexité exponentielle (cf. section suivante).

15.5 CODAGE ALGEBRIQUE - DETECTION ET CORRECTION D'ERREURS

Dans cette section nous allons décrire les deux principales approches algébriques à la construction de codes "facilement" décodables. Les codes sont construits à partir d'une structure d'espace vectoriel linéaire induite à partir d'un corps fini. Les codes construits à l'aide de blocs de symboles de longueur n sont des sous-espaces linéaires de dimension k d'un espace de dimension n. Ces codes peuvent être générés soit à l'aide d'une matrice génératrice de dimension $k \times n$, soit à l'aide d'un polynôme générateur de degré $(n-k)$.

Ci-dessous nous raisonnons exclusivement sur les canaux discrets, en particulier le canal symétrique binaire. Nous supposons que le modèle du canal est donné, par exemple en déduisant ses caractéristiques à partir du canal physique et du système de modulation (et de mise en forme) utilisé.

15.5.1 Espaces vectoriels finis

Nous considérons le cas d'un alphabet de canal fini $\mathcal{X}$ de taille q, et nous supposerons que nous voulons coder un ensemble de q^k messages de source à l'aide de mots de code de longueur n. Il y a donc lieu de choisir q^k mots de code parmi les q^n possibles.

Notre objectif ici est de munir l'ensemble des séquences possibles $\mathcal{X}^n$ (d'entrée et de sortie du canal) d'une structure d'espace vectoriel linéaire. Rappelons que nous nous focalisons ici sur des ensembles $\mathcal{X}$ finis.

Pour munir l'ensemble $\mathcal{X}^n$ d'une structure d'espace vectoriel linéaire, on commence par munir $\mathcal{X}$ d'une structure de corps fini (deux opérations : addition et multiplication qui satisfont à certains axiomes). La structure d'espace vectoriel linéaire est ensuite induite sur $\mathcal{X}^n$ à partir de cette structure de corps, tout comme la structure d'espace vectoriel euclidien est induite sur $\mathbb{R}^n$ à partir de la structure de corps de $\mathbb{R}$. Cependant, le fait que $\mathcal{X}$ soit fini conduit à un certain nombre de particularités qui peuvent nous compliquer la vie. Nous renvoyons le lecteur à l'appendice E, qui fournit les éléments mathématiques de base relatifs aux corps finis et discute un certain nombre de leurs particularités.

Retenons ici que pour pouvoir parler de corps fini, il faut que le nombre q d'éléments de $\mathcal{X}$ soit une puissance entière d'un nombre premier. C'est notamment le cas si q est premier, et en particulier si $q = 2$. Dans ce cas particulier, le corps fini est défini par les opérations d'addition et de multiplication *modulo* 2 (il n'y a pas d'autre choix possible). Dans les exemples utilisés au cours oral nous nous sommes limités à ce cas particulier afin de faciliter la compréhension; ci-dessous nous adopterons un point de vue un peu plus général.

15.5.2 Codes linéaires en blocs (n, k)

En supposant que n soit fixé, que $\mathcal{X}$ (l'alphabet utilisé par le canal discret) soit muni d'une structure de corps, et que $\mathcal{X}^n$ soit muni de la structure d'espace vectoriel linéaire correspondante, un code linéaire (n, k) est par définition un sous-espace vectoriel linéaire de $\mathcal{X}^n$ de dimension k. Le nombre de vecteurs différents d'un tel sous-espace vaut q^k, et donc le nombre M de mots de code vaut également q^k.

Remarquons tout d'abord que tout code linéaire contient nécessairement le mot de code nul (dont les composantes sont toutes égales à l'élément neutre de l'addition sur le corps $\mathcal{X}$).

Dans un espace linéaire, le *poids de Hamming* d'un vecteur est par définition égal à la distance de ce vecteur au vecteur nul, c'est-à-dire au nombre de composantes non nulles du vecteur. Donc, la distance de Hamming entre deux vecteurs est aussi égale au poids de Hamming du vecteur formé par leur différence.

Nous allons mettre en évidence les principaux avantages liés à ce choix de structure.

15.5.2.1 Avantage de représentation. *Note. Nous représentons ici les mots de code par des vecteurs ligne.*

Alors qu'un code quelconque de taille $M = q^k$ nécessite une table de taille $M \times n$ pour être représenté, un code linéaire est entièrement déterminé au moyen d'une base du sous-espace linéaire qui lui correspond, c'est-à-dire un tableau $\boldsymbol{G}$ de taille $k \times n$ (qu'on désigne sous le nom de matrice génératrice).

Tout élément du code peut alors s'écrire comme une combinaison linéaire des lignes de la matrice $\boldsymbol{G}$, c'est-à-dire sous la forme d'un produit

$$\boldsymbol{x} = \boldsymbol{s}\boldsymbol{G},$$

où $\boldsymbol{s}$ est un vecteur quelconque de longueur k. Les q^k mots de codes sont ainsi générés à partir des q^k vecteurs (ligne) $\boldsymbol{s}$ (signaux) de longueur k : $\boldsymbol{s}$ représente un des messages d'entrée possibles de l'encodeur, et ce dernier construit le mot de code à l'aide d'une opération purement algébrique de multiplication par la matrice génératrice selon l'arithmétique du corps construit sur l'alphabet du code (l'appendice à ce chapitre montre comment on peut construire ces arithmétiques pour des valeurs de q qui sont une puissance entière d'un nombre premier).

15.5.2.2 Standardisation : mise sous forme systématique. En réalité, il est possible d'encore simplifier considérablement les opérations algébriques en remarquant qu'on peut se limiter à des familles de matrices génératrices encore plus simples.

En effet, il est assez immédiat que si $\boldsymbol{G}$ est la matrice génératrice d'un code, alors toute matrice $\boldsymbol{G}'$ obtenue à partir de $\boldsymbol{G}$ au moyen de combinaisons des opérations suivantes donne un code équivalent

1. permutation de lignes ou de colonnes
2. multiplication d'une ligne ou d'une colonne par un scalaire non nul
3. substitution d'une ligne par la somme de celle-ci et d'une ligne parallèle

Par conséquent, comme il est possible de standardiser au moyen des opérations précédentes la matrice $\boldsymbol{G}$ sous la forme

$$\boldsymbol{G} = [\boldsymbol{I}_k, \boldsymbol{P}],$$

(ou $\boldsymbol{I}_k$ représente la matrice identité $k \times k$ dans notre algèbre) nous pouvons aussi bien limiter notre attention à l'étude des codes linéaires engendrés par de telles matrices. La donnée de la matrice $\boldsymbol{P}$ de dimension $k \times (n-k)$ suffit donc pour caractériser entièrement ce genre de code.

Pour une matrice génératice mise sous la forme standard $[\boldsymbol{I}_k, \boldsymbol{P}]$, la construction du mot de code associé à un message $\boldsymbol{s}$ consiste à lui concaténer le mot de longueur $n-k$ obtenu par

$$\boldsymbol{p}(\boldsymbol{s}) = \boldsymbol{s}\boldsymbol{P}$$

.

On dit que $\boldsymbol{p}(\boldsymbol{s})$ sont les bits (symboles) de contrôle de parité et que $\boldsymbol{P}$ est la matrice de parité.

Parce que les k premiers symboles transmettent les symbole source on dit que le code est *systématique*. On a donc

$$\boldsymbol{x}(\boldsymbol{s}) = \boldsymbol{s}\boldsymbol{G} = [\boldsymbol{s}\boldsymbol{I}_k, \boldsymbol{s}\boldsymbol{P}] = [\boldsymbol{s}, \boldsymbol{s}\boldsymbol{P}].$$

15.5.2.3 Distance minimale (d). *La distance minimale (d) d'un code linéaire est égale au poids de Hamming minimal des vecteurs non nuls de ce code.*

En effet, le poids minimal est certainement une borne supérieure de la distance minimale, puisque le vecteur nul fait partie de tout code linéaire. D'autre part, si la distance minimale était strictement inférieure, cela voudrait dire qu'il existerait deux mots de code tels que leur différence serait de poids égal à d, ce qui est en contradiction avec le fait que ce mot doit faire partie du code, celui-ci étant linéaire par hypothèse.

15.5.2.4 Encodage. En supposant que les $M = q^k$ messages de source sont codés à l'aide de k symboles q-aires (changement d'alphabet déjà effectué si nécessaire), il suffit de calculer le mot de code à l'aide de la formule

$$\boldsymbol{x} = \boldsymbol{s}\boldsymbol{G}.$$

Puisque la matrice $\boldsymbol{G}$ est normalisée, cela revient en fait à prendre $x_i = s_i, \forall i = 1, \ldots, k$ et

$$[x_{k+1} \cdots x_n] = \boldsymbol{s}\boldsymbol{P}.$$

On voit que le codage d'un bloc de k symboles source s'effectue en leur associant un suffixe de $n-k$ symboles calculés au moyen d'une opération linéaire ($\mathcal{O}(n(n-k))$ opérations élémentaires), ce qui justifie le nom de *code linéaire en blocs*.

Des équations précédentes on déduit également que tout mot de code doit satisfaire l'équation (dite *de contrôle*) suivante

$$[-\boldsymbol{P}^T, I_{n-k}]\boldsymbol{x}^T = \boldsymbol{0},$$

qui consiste simplement à recalculer les symboles de parité à partir des k premiers symboles et à les retrancher des $n-k$ symboles de parité du mot $\boldsymbol{x}$.

La matrice $\boldsymbol{H} = [-\boldsymbol{P}^T, I_{n-k}]$ est appelée matrice de contrôle du code. Il est à noter que dans le cas d'un alphabet binaire (arithmétique modulo 2) on a

$$-\boldsymbol{P}^T = \boldsymbol{P}^T$$

.

La matrice $\boldsymbol{H}$ est évidemment de rang $n-k$ et ses $(n-k)$ lignes définissent donc un sous-espace linéaire de dimension $n-k$. De ce point de vue, le code apparaît comme le "complément orthogonal" du sous-espace linéaire engendré par $\boldsymbol{H}$. Tout comme $\boldsymbol{G}$, la donnée de $\boldsymbol{H}$ caractérise bien sûr entièrement le code.

15.5.2.5 Décodage à distance minimale. Le problème est de trouver pour un vecteur reçu le (ou un) vecteur du code le plus proche au sens de la distance de Hamming. Plaçons nous dans le cas binaire pour décrire l'algorithme utilisé en pratique, basé sur l'idée suivante :

Remarquons tout d'abord que $\mathcal{C}$ étant un sous-espace linéaire de $\mathcal{X}^n$, il en est aussi un sous-groupe. On peut donc factoriser $\mathcal{X}^n$ par la technique des cosets (voir appendice E).

Soit alors $\boldsymbol{y}$ le vecteur reçu et considérons le code $\mathcal{C}$. Pourtout $\boldsymbol{x} \in \mathcal{C}$ on peut toujours écrire que $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$. Nous disons que $\boldsymbol{e}$ est un vecteur d'erreur *possible*, s'il existe $\boldsymbol{x} \in \mathcal{C}$ tel que cette équation soit vérifiée. Le décodage à distance minimale revient donc à trouver un vecteur d'erreur possible de poids de Hamming minimal.

Notons par $\boldsymbol{y} + \mathcal{C}$ l'ensemble des vecteurs qui peuvent s'écrire sous la forme $\boldsymbol{y} + \boldsymbol{x}$ où $\boldsymbol{x}$ est un vecteur quelconque de $\mathcal{C}$. Il s'agit du coset de $\boldsymbol{y}$ *modulo* $\mathcal{C}$ (voir appendice E). Notons que cet ensemble est identique (en toute généralité, à cause de la linéarité du code $\mathcal{C}$) à l'ensemble des vecteurs qui s'écrivent $\boldsymbol{y} - \boldsymbol{x}$, c'est-à-dire à l'ensemble des vecteurs d'erreurs $\boldsymbol{e}$ possibles pour un $\boldsymbol{y}$ reçu.

Il suffit donc, pour le décodage, de produire le (ou un) vecteur de poids de Hamming minimal appartenant au coset $\boldsymbol{y} + \mathcal{C}$. Parmi les vecteurs de poids minimal de $\boldsymbol{y} + \mathcal{C}$ nous pouvons en choisir un arbitrairement et lui donner le statut de *tête* du coset (voir également appendice E).

Or, nous savons par ailleurs que le nombre de cosets modulo $\mathcal{C}$ différents est exactement q^{n-k}. Cela veut dire qu'en faisant varier $\boldsymbol{y}$ il y a exactement q^{n-k} vecteurs d'erreurs minimaux possibles. De plus, deux vecteurs $\boldsymbol{y}_1$ et $\boldsymbol{y}_2$ donneront lieu au même résultat (même vecteur d'erreur) si et seulement si ces deux vecteurs appartiennent au même coset, c'est-à-dire si et seulement si il existe un vecteur $\boldsymbol{x} \in \mathcal{C}$ tel que $\boldsymbol{y}_1 = \boldsymbol{y}_2 + \boldsymbol{x}$, c'est-à-dire si et seulement si

$$\boldsymbol{H}\boldsymbol{y}_1^T = \boldsymbol{H}(\boldsymbol{y}_2^T + \boldsymbol{x}^T) = \boldsymbol{H}\boldsymbol{y}_2^T,$$

car $\boldsymbol{x} \in \mathcal{C} \Leftrightarrow \boldsymbol{H}\boldsymbol{x}^T = 0$.

En d'autres termes, si nous définissons par $\boldsymbol{a} = \boldsymbol{H}\boldsymbol{y}^T$ le syndrôme du vecteur $\boldsymbol{y}$, tous les vecteurs d'un même coset ont aussi même syndrôme (on peut donc parler du syndrôme du coset). Réciproquement, des vecteurs ayant des syndrômes identiques appartiennent forcément à des cosets identiques.

Il y a donc d'une part une correspondance bijective entre vecteur d'erreur possible minimal et coset, d'autre part entre coset et syndrôme. On conclut qu'il y a une correspondance bijective entre syndrôme et vecteur d'erreur possible minimal.

La propriété générale précédente donne lieu à l'algorithme de décodage semi-efficace suivant (basé sur une table de correspondance entre syndrômes et vecteurs d'erreurs minimaux, construite préalablement) :

1. à la réception de $\boldsymbol{y}$ calculer le syndrôme $\boldsymbol{H}\boldsymbol{y}^T$;
2. en fonction du résultat aller chercher la tête de coset dans la table, soit $\boldsymbol{e}_0$;
3. décoder $\boldsymbol{y}$ par $\boldsymbol{y} - \boldsymbol{e}_0$.

Nous qualifions cet algorithme de semi-efficace car il fait appel à une table de taille $q^{(n-k)}$. Il peut donc conduire à une utilisation mémoire importante en pratique, mais à un temps d'accès relativement rapide, dans la mesure où le syndrôme peut encore être utilisé comme adresse directe. En tout état de cause, la recherche énumérative est évidemment plus efficace dans une table de taille $q^{(n-k)}$ que dans une table de taille q^k (en pratique on a souvent $n - k \ll k$), et les calculs sont plus simples dans le cas présent (test d'égalité de syndrômes de longueur $n-k$) que lors de la recherche du plus proche voisin (calcul de distances de Hamming de vecteurs de longueur n).

Construction de codes linéaires en bloc. Historiquement, une bonne partie (sinon la majeure) des travaux sur le codage algébrique ont consisté à définir des familles de codes linéaires vérifiant des propriétés de plus en plus structurantes. Nous allons en illustrer les plus élémentaires dans ce qui suit, mais nous suggérons au lecteur qui souhaite en savoir plus de consulter des ouvrages spécialisés (p.ex. [Adá91] et les références qui sont données dans cet ouvrage).

15.5.3 Codes de Hamming binaires

Nous allons illustrer les notions qui viennent d'être introduites au moyen d'une famille de codes utilisés pour la correction et la détection des erreurs dans le cas où les canaux sont relativement fiables (p.ex. réseaux locaux). Nous allons nous placer dans le cas d'un canal binaire symétrique et supposer que la probabilité de corruption d'un bit (p) est suffisamment faible pour qu'on puisse considérer que la probabilité d'avoir plus d'une erreur de transmission sur un bloc de n bits est pratiquement négligeable. On s'intéresse alors à la possibilité de correction d'une seule erreur et éventuellement à la possibilité de détection de deux erreurs. On cherche donc des codes dont la distance minimale d vaut au moins 3, et qui idéalement soient parfaits.

Nous avons vu que le code de répétition $(3, 1)$ est un code parfait avec $d = 3$. Nous allons maintenant étudier un moyen systématique permettant de construire une plus large famille de tels codes : ceux-ci portent le nom de codes de Hamming.

15.5.3.1 Définition. On travaille ici avec des alphabets binaires, et donc les opérations algébriques sont les opérations dites "modulo 2". Soit r un nombre entier (en pratique supérieur à 1) et soit Z_2^r l'espace linéaire binaire de dimension r (l'ensemble des mots binaires de longueur r) et soit $\boldsymbol{H}$ une matrice de dimension $r \times (2^r - 1)$ dont les colonnes sont les $(2^r - 1)$ vecteurs non-nuls de Z_2^r (pris dans un ordre quelconque).

Alors $\boldsymbol{H}$ est une matrice de contrôle d'un code binaire (n, k) où

$$n = 2^r - 1 \qquad \text{et} \qquad k = n - r. \tag{15.10}$$

En effet, la matrice contient forcément exactement r colonnes linéairement indépendantes parmi les n et le complément orthogonal de l'espace engendré par cette matrice est donc bien un espace linéaire de dimension $k = n - r$. Les codes ainsi obtenus peuvent évidemment être standardisés ce qui permet d'en déterminer la matrice de parité $\boldsymbol{P}$ et donc $\boldsymbol{G}$.

Les valeurs possibles de (n, k) sont déterminées à partir de (15.10) en utilisant $r = 1, 2, 3, 4, \ldots 10, \ldots$ On obtient

$$(1, 0); (3, 1); (7, 4); (15, 11); \ldots; (1023, 1013), \tag{15.11}$$

dont le premier est quelque peu trivial, et le second constitue le code de répétition dont nous avons déjà parlé.

Nous allons voir que les r bits de parité permettent de retrouver le bit corrompu en cas d'erreur simple.

15.5.3.2 Correction et détection d'erreurs. La propriété principale des codes de Hamming est qu'il s'agit de codes parfaits de capacité de correction d'une seule erreur et de détection de deux erreurs.

Montrons d'abord que la distance minimale des codes de Hamming pour $r \geq 3$ vaut exactement 3.

Elle ne peut pas valoir 1, car sinon il existerait un mot de code de poids 1 (disons $\boldsymbol{x}_i$ qui aurait un 1 en position i et des 0 ailleurs) tel que $\boldsymbol{H}\boldsymbol{x}_i = \boldsymbol{0}$, ce qui voudrait dire que la i-ème colonne de $\boldsymbol{H}$ est nulle. Or nous savons que toutes les colonnes de $\boldsymbol{H}$ sont non-nulles par construction.

De même, si la distance minimale valait 2, on en déduirait que $\boldsymbol{H}$ posséderait deux colonnes dont la somme serait nulle, ce qui est impossible puisque toutes les colonnes de cette matrice sont différentes.

Par contre, il est facile de trouver trois colonnes de $\boldsymbol{H}$ dont la somme est nulle, et on en déduit qu'il existe des mots de code de poids 3 et donc la distance minimale vaut 3.

On en déduit que le décodage à distance de Hamming minimale d'un tel code de Hamming est capable de détecter toutes les erreurs doubles, mais pas les erreurs triples, et qu'il est capable de corriger toutes les erreurs simples.

Pour montrer qu'il s'agit d'un code parfait, il suffit de se convaincre que les t-sphères de taille $t = 1$ autour des mots du code couvrent exactement l'espace de tous les mots binaires de longueur n. Or, chacune de ces sphères contient exactement $1 + n$ vecteurs. Comme il y a 2^k mots de codes on obtient au total

$$2^k + n2^k = 2^k(1 + 2^r - 1) = 2^{k+r} = 2^n$$

mots, soit ce qu'il faut.$\square$

15.5.3.3 Exemple illustratif : le code de Hamming (7,4).

Matrices et tables du code. Ce code correspond à $r = 3$. Partons donc d'une matrice de contrôle 3×7 mise sous forme systématique de la manière suivante

$$\boldsymbol{H} = \left[\begin{array}{cccc|ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right] = \left[\boldsymbol{P}^T \boldsymbol{I}_3\right],$$

dont on déduit la matrice génératrice suivante

$$\boldsymbol{G} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right] = [\boldsymbol{I}_4 \boldsymbol{P}],$$

dont on peut déduire à son tour la table du code qui est obtenue en multipliant la matrice par tous les mots binaires $\boldsymbol{s}$ de longueur $k = 4$, soit

$\mathcal{C} =$

$\boldsymbol{s}$	$\boldsymbol{x}$	$\boldsymbol{s}$	$\boldsymbol{x}$	$\boldsymbol{s}$	$\boldsymbol{x}$	$\boldsymbol{s}$	$\boldsymbol{x}$
0000	0000000	0100	0100110	1000	1000101	1100	1100011
0001	0001011	0101	0101101	1001	1001110	1101	1101000
0010	0010111	0110	0110001	1010	1010010	1110	1110100
0011	0011100	0111	0111010	1011	1011001	1111	1111111

Décodage. En supposant que le canal binaire est symétrique de probabilité de corruption d'un symbole $p < 0.5$, nous savons que le décodage à distance de Hamming minimale est optimal (équivalent au décodage au maximum de vraisemblance).

Voyons comment fonctionnent l'encodage et le décodage ($n = 7, k = 4, r = 3$)

1. La source émet un mot sur les $M = 2^k = 16$ possibles, ce qui revient à choisir un signal binaire de longueur k, disons $\boldsymbol{s}$.
2. L'encodeur émet le mot $\boldsymbol{x}$: les k premiers bits correspondent au mot $\boldsymbol{s}$. Ensuite il calcule et transmet les r (=3) bits de parité $\boldsymbol{x}_p = \boldsymbol{sP}$ (le premier bit de parité vaut $s_1 + s_2 + s_3$, le second vaut $s_2 + s_3 + s_4$, et le troisième vaut $s_1 + s_3 + s_4$).
3. Le mot $\boldsymbol{x}$ est transmis sur le canal et le mot $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{b}$ est reçu, où $\boldsymbol{b}$ désigne le vecteur de bruit.
4. Le récepteur calcule le syndrôme $\boldsymbol{a}$, c'est-à-dire le mot de trois bits défini par $\boldsymbol{a} = \boldsymbol{H}\boldsymbol{y}^T$, et détermine le vecteur de bruit $\boldsymbol{b}'$ de poids 0 ou 1 tel que $\boldsymbol{H}\boldsymbol{y}^T = \boldsymbol{H}\boldsymbol{b}'^T$.
5. Le mot $\boldsymbol{y}$ est décodé par la règle $\boldsymbol{x}' = \boldsymbol{y} + \boldsymbol{b}'$ et $\boldsymbol{s}'$ est obtenu à partir des 4 premiers bits de $\boldsymbol{x}'$.

La table de décodage qui relie les syndrômes aux vecteurs de bruit de poids 1 est la suivante :

$\boldsymbol{a}^T$	$\boldsymbol{b}$'	$\boldsymbol{a}^T$	$\boldsymbol{b}$'
000	0000000	100	0000100
001	0000001	101	1000000
010	0000010	110	0100000
011	0001000	111	0010000

(15.12)

Analyse des erreurs. En utilisant le décodage par syndrôme (ou de manière équivalente, le décodage au maximum de vraisemblance) une erreur se produira sur le mot $\boldsymbol{x}'$ dès lors que deux bits en sont corrompus lors de la transmission. Mais, est-ce que ces erreurs se répercutent nécessairement sur le préfixe $\boldsymbol{s}'$?

On pourrait en effet imaginer qu'on aurait deux ou trois erreurs de transmission concentrées sur les bits de parité, et que le décodage retrouve correctement les bits de signal. Il n'en est rien, car comme nous venons de le voir, si plusieurs des trois derniers bits du mot de code ont été corrompus, cela se traduira nécessairement par un syndrôme de poids supérieur ou égal à 1, ce qui, comme le montre la table 15.12 donnera lieu lors du décodage à l'hypothèse "1 des bits de signal est corrompu".

La règle de décodage au maximum de vraisemblance entraîne en réalité qu'il y aura erreur de décodage dès lors que deux ou plus de bits ont été corrompus. La probabilité de décodage correct est donc égale à la probabilité d'avoir aucun ou un seul bit corrompu lors de la transmission d'un mot de 7 bits, ou plus généralement de n bits.

On a donc la formule suivante (où p désigne la probabilité de corruption d'un bit transmis sur le canal)

$$P_e^{\text{mot}} = 1 - (1-p)^n - np(1-p)^{n-1} \tag{15.13}$$

où P_e^{mot} désigne la probabilité d'erreur de décodage par mot source de k bits codés sur n bits.

Dans le cas du code Hamming $(7, 4)$ avec $p = 0.1$ on obtient $P_e^{\text{mot}} = 0.149694$.

Sans code correcteur d'erreur on aurait une probabilité d'erreur de transmission de mots de k bits égale à

$$P_e^{\text{mot}} = 1 - (1-p)^k \tag{15.14}$$

soit quand $k = 4$ et $p = 0.1$, $P_e^{\text{mot}} = 0.3439$.

D'autre part, on est sûr que deux erreurs seront détectées. Si nous supposons que dans ce cas on peut retransmettre le message (avec une probabilité d'erreur P_e^{mot}) on aura

$$\begin{aligned} P_e^{\text{mot}} &= 1 - (1-p)^n - np(1-p)^{n-1} - C_2^n p^2 (1-p)^{n-2}(1 - P_e^{\text{mot}}) && (15.15) \\ &\approx 1 - (1-p)^n - np(1-p)^{n-1} + C_2^n p^2 (1-p)^{n-2}. && (15.16) \end{aligned}$$

15.5.4 Codes cycliques

Un code $\mathcal{C}(n, k)$ est dit cyclique s'il est linéaire et s'il contient toutes les permutations circulaires de ses mots de code. Il suffit donc qu'il soit linéaire et que si $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{C}$ alors $\boldsymbol{x}' = (x_n, x_1, \ldots, x_{n-1}) \in \mathcal{C}$.

Note. Ici nous ferons la discussion pour une valeur de q quelconque (compatible avec la structure de corps fini, i.e. $q = p^m$, avec p premier et m entier positif). Nous désignerons ci-dessous par K la structure de corps fini définie sur l'alphabet du code.

Les codes cycliques disposent de propriétés encore plus attractives du point de vue algébrique et du point de vue simplicité de représentation. Nous verrons que pour un code cyclique (n, k) la donnée d'un seul mot de code de longueur $n - k + 1$ suffit à décrire complètement le code. Nous verrons également qu'il existe un algorithme de décodage efficace qui tire profit de la structure cyclique du code.

15.5.4.1 Représentation des mots de code par des polynômes. Nous allons introduire un nouveau type de modèle, à savoir les polynômes à coefficients dans K, dont nous nous servirons également dans la suite. Nous renvoyons le lecteur à l'appendice 15.A pour une discussion plus détaillée de ces objets.

Identifions un mot de code $(x_1, \ldots, x_n)$ (où $x_i \in K$) de longueur n avec le polynôme de degré $n - 1$ en la variable D

$$x_1 + x_2 D + x_3 D^2 + \cdots + x^n D^{n-1}. \tag{15.17}$$

Définissons ensuite sur l'ensemble des polynômes de degré $n - 1$ les opérations d'addition et de multiplication comme suit :

- Addition : l'addition de deux polynômes de degré $n - 1$ est obtenue en additionnant leurs coefficients selon la règle d'addition définie sur K.
- Multiplication : la multiplication de deux polynômes de degré $n - 1$ est obtenue modulo $D^n - 1$, c'est-à-dire de la manière suivante :

1. On multiplie les deux polynômes de la manière usuelle, en utilisant l'arithmétique définie sur K pour le calcul des coefficients du polynôme résultant : cela donne un polynôme au plus de degré $2n-2$;
2. on calcule ensuite le reste de la division de ce polynôme par le polynôme $D^n - 1$, ce qui donne un polynôme au plus de degré $n-1$, à coefficients dans K.

Par exemple (en supposant que $n = 3$ et que K est le corps Z_2), si on multiplie les polynômes $a(D) = 1 + D + D^2$ et $b(D) = 1 + D^2$ on obtient successivement

1. $(1 + D + D^2)(1 + D^2) = 1 + D + D^2 + D^2 + D^3 + D^4 = 1 + D + D^3 + D^4$
   (On a $D^2 + D^2 = 0$ en arithmétique modulo 2.)
2. en divisant par $D^3 - 1$ on obtient comme reste le polynôme nul (on a $1 + D + D^3 + D^4 = (D^3 + 1)(D + 1)$)

L'observation fondamentale ici consiste à remarquer que le décalage cyclique vers la droite de k bits d'un mot de code revient alors à effectuer la multiplication du polynôme correspondant par le polynôme D^k (selon les règles établies ci-dessus). Si

$$a(D) = a_n D^{n-1} + a_{n-1} D^{n-2} + \cdots + a_1$$

on a en effet

$$Da(D) = a_n D^n + a_{n-1} D^{n-1} + \cdots + a_1 D$$

et aussi

$$Da(D) = a_n(D^n - 1) + a_{n-1} D^{n-1} + \cdots + a_1 D + a_n,$$

et donc le reste de la division de $Da(D)$ par $(D^n - 1)$ est bien le polynôme

$$a_{n-1} D^{n-1} + \cdots + a_1 D + a_n,$$

c'est-à-dire le polynôme correspondant au mot décalé de manière cyclique vers la droite de 1 pas.

15.5.4.2 Propriétés des codes cycliques.

1. Comme le code cyclique est linéaire on en déduit la propriété suivante : *si $x(D)$ représente un mot de code et $a(D)$ un polynôme de degré $n-1$ au plus, alors le polynôme $x(D)a(D)$ (modulo $D^n - 1$) représente encore un mot de code.* (Cela résulte directement de la propriété précédente et du fait que le code est linéaire).

2. *Soit un code cyclique de type (n, k), et soit $g(D)$ un mot de ce code non-nul et de degré minimal. Alors : (i) le degré de $g(D)$ vaut $n-k$; (ii) tous les mots du code peuvent être générés en multipliant $g(D)$ par un polynôme de degré inférieur à k; (iii) tous les mots de code de degré $n-k$ sont égaux à $g(D)$ à une constante multiplicative près; (iv) $g_1 \neq 0$.*

   Démonstration.

   Soit $g(D)$ un mot de code de degré minimal et s ce degré.

   Alors les propriétés (iii) et (iv) doivent être vraies car si ce n'était pas le cas, il serait facile de construire des mots de code de degré strictement inférieur à s.

   En effet, pour démontrer (iii) supposons qu'il existe un autre mot de code $g'(D)$ de degré s différent de $g(D)$ et soit g_{s+1} et g'_{s+1} les coefficients de degré s : on en déduit que le mot $g''(D) = g_{s+1}g'(D) - g'_{s+1}g(D)$ est de degré inférieur à s et fait partie du code (par linéarité). Par conséquent on doit avoir $g''(D) = 0$.

   Pour démontrer (iv), il suffit de constater que si $g_1 = 0$ alors il serait possible de construire un mot de code à partir de $g(D)$ par décalage cyclique et dont le degré serait strictement inférieur à s.

   En ce qui concerne la propriété (ii), on peut raisonner come suit. En effet, soit s le degré minimal des mots de code non-nuls du code cyclique (n, k) et soit $g(D)$ un mot de code ayant ce degré. Bien sûr tous les multiples de $g(D)$ par des polynômes de degré inférieur à $n-s$ sont des mots du code. Réciproquement, si $x(D)$ est un mot du code, on peut toujours écrire

$$x(D) = q(D)g(D) + r(D)$$

où $q(D)$ désigne le polynôme quotient et $r(D)$ le reste. Puisque $q(D)g(D)$ et $x(D)$ sont des mots du code, leur différence $r(D)$ l'est aussi. Or celui-ci est de degré strictement inférieur à s, ce qui n'est possible que si $r(D) = 0$.

Enfin, montrons (i), c'est-à-dire que $s = n - k$.

Vu ce qui précède, les $n - s$ mots $g(D), Dg(D), D^2g(D), \ldots, D^{n-s-1}g(D)$ forment une base du code (ils sont linéairement indépendants et engendrent tous les mots du code par combinaison linéaire). Donc on doit avoir $n - s = k$ puisque le code est (n, k), et réciproquement.

3. *Si $\mathcal{C}$ est un code cyclique de (n, k) de polynôme générateur*
$$g(D) = g_1 + g_2D + \cdots + g_{n-k+1}D^{n-k}$$
alors une matrice génératrice du code est donnée par la matrice $k \times n$ suivante
$$\boldsymbol{G} = \begin{bmatrix} g_1 & g_2 & g_3 & \cdots & g_{n-k+1} & 0 & 0 & \cdots & 0 \\ 0 & g_1 & g_2 & \cdots & g_{n-k} & g_{n-k+1} & 0 & \cdots & 0 \\ 0 & 0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} & g_{n-k+1} & \cdots & 0 \\ \vdots & \vdots & \cdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & g_{n-k+1} \end{bmatrix}$$

4. *Le polynôme générateur $g(D)$ est nécessairement un facteur de $D^n - 1$, c'est-à-dire qu'il existe un polynôme $h(D)$ de degré k tel que*
$$D^n + 1 = g(D)h(D)$$
ou encore
$$0 = g(D)h(D) \quad (\textit{modulo } D^n - 1).$$
Ce polynôme porte aussi le nom de polynôme de contrôle car la condition nécessaire et suffisante pour qu'un mot $x(D)$ fasse partie du code est
$$0 = x(D)h(D) \quad (\text{modulo } D^n - 1).$$
Réciproquement, si $g(D)$ est un facteur de $D^n - 1$ alors il engendre un code cyclique.

Ce qui précède confirme qu'un code cyclique (n, k) est entièrement défini par les $n - k + 1$ coefficients de son polynôme générateur. De plus, pour une valeur fixée de n, seuls les facteurs du polynôme $D^n - 1$ peuvent donner lieu à des codes cycliques, et chacun de ces facteurs donne effectivement lieu à un code cyclique.

15.5.4.3 Encodage. On peut imaginer deux façons d'encoder. La première résulte directement de la structure de la matrice génératrice
$$\boldsymbol{x}(\boldsymbol{s}) = \boldsymbol{s}\boldsymbol{G} \Leftrightarrow x(D) = s(D)g(D),$$
où $s(D)$ est un polynôme de degré $k - 1$ qui représente un message source à encoder. Cette première façon de faire n'est pas systématique.

On peut aussi s'arranger pour produire un mot de code de manière systématique en faisant uniquement appel à l'algèbre polynomiale. En effet, on peut associer à $s(D)$ un mot de code au moyen de la règle d'encodage suivante :
$$s(D) \to x(D)$$
où
$$x(D) = D^{n-k}s(D) - r(D)$$
où $r(D)$ est le reste de la division de $D^{n-k}s(D)$ par $g(D)$, c'est-à-dire qu'on a
$$D^{n-k}s(D) = q(D)g(D) + r(D),$$
ce qui implique que $x(D) = q(D)g(D)$ et donc que $x(D)$ est bien un mot du code. D'autre part, le degré de $r(D)$ est forcément inférieur à $n - k$. Donc, le mot de code contient le message source aux k positions de poids le plus élevé.

Notons qu'il est possible d'implémenter le calcul du reste de la division polynomiale au moyen de systèmes de registres à décalage.

15.5.4.4 Décodage. On parle ici de polynôme syndrôme. Il s'agit du polynôme obtenu en calculant le reste de la division du mot de code par le polynôme générateur. Il s'agit d'un polynôme de degré inférieur à $n - k$.

Evidemment, si le mot reçu est un mot de code, le syndrôme est le polynôme nul. D'autre part, si le mot reçu $y(D)$ est le fruit de la corruption d'un mot de code $x(D)$ par un pattern d'erreurs $e(D)$, alors $y(D)$ et $e(D)$ auront le même polynôme syndrôme. En effet,

$$x(D) = g(D)q(D)$$

et si

$$y(D) = x(D) + e(D)$$

et que

$$y(D) = g(D)q'(D) + r(D),$$

alors

$$e(D) = g(D)(q'(D) - q(D)) + r(D).$$

Plus généralement, les syndrômes d'erreurs d'un polynôme reçu sont invariants si on ajoute (ou retranche) à ce polynôme un polynôme qui représente un mot du code.

On pourrait alors associer à chaque polynôme syndrôme possible (c'est-à-dire à chaque polynôme de degré inférieur ou égal à $n - k - 1$) un pattern d'erreur de poids minimal, selon le même schéma que celui que nous avons discuté plus haut.

Mais, grâce à la structure cyclique du code, nous pouvons en fait faire ce décodage de manière nettement plus efficace. En effet les codes cycliques présentent la propriété suivante : *si $s(D)$ est le syndrôme d'un mot $y(D)$, alors le mot $y'(D)$ obtenu à partir de $y(D)$ par un décalage circulaire (multiplication par D, modulo $D^n - 1$) a comme syndrôme le reste de la division de $Ds(D)$ par le polynôme générateur $g(D)$.*

Cette propriété permet de corriger les bits du mot $y(D)$ de manière séquentielle en partant du bit de poids le plus fort. Cela donne l'algorithme de décodage suivant (MEGGIT) dans le cas binaire :

Etape 1. On construit une liste qui contient tous les syndrômes qui correspondent aux patterns d'erreur (polynôme de correction) de degré $n - 1$ (tels que $e_n = 1$, dans le cas binaire).

Etape 2. Le mot reçu $y(D)$ est utilisé pour alimenter un circuit capable de calculer le syndrôme (reste de la division par $g(D)$).

Etape 3. Si le syndrôme fait partie de la liste construite à l'étape 1, le bit de poids le plus fort de $y(D)$ doit être modifié (inversé) pour obtenir le bit de poids le plus fort du mot décodé $x'(D)$.

Etape 4. On effectue un décalage circulaire vers la droite de $y(D)$ et on recommence à l'étape 2. Après n itérations le décodage est terminé.

Le gain essentiel de cet algorithme est de diminuer considérablement la taille de la table qui associe les vecteurs d'erreur au syndrômes, puisque non seulement on ne stocke que les syndrômes dont le vecteur d'erreur contient un 1 à la position n, mais en plus il n'est pas nécessaire de stocker les vecteurs d'erreurs (on ne se sert que de la présence du vecteur dans la table).

Par exemple, dans le cas des codes de Hamming (voir ci-dessous), il n'est nécessaire de stocker qu'un seul syndrôme au lieu de 7 (puisque ce code ne corrige que les erreurs simples). Dans cas du code BCH (15,7) qui corrige des erreurs doubles on ne doit stocker que 15 syndrômes (vecteurs de dimension $n - k = 8$) au lieu d'une table comportant 121 ($1 + 15 + C_{15}^2$) lignes de longueur 23 (chaque ligne associe un syndrôme de longueur 8 et un vecteur de 0, 1 et 2 erreurs de longueur 15).

15.5.4.5 Exemple 1 : codes de Hamming binaires. Fixons $n = 7$, et voyons comment en déduire les codes cycliques de longueur 7. Pour cela il suffit de factoriser $D^7 - 1$ en facteurs irréductibles : on a (en arithmétique modulo 2)

$$D^7 - 1 = (1 + D + D^3)(1 + D^2 + D^3)(1 + D).$$

Nous en déduisons qu'il existe deux codes cycliques $(7, 4)$ et un code cyclique $(7, 6)$.

Voyons le code correspondant au polynôme

$$g(D) = (1 + D + D^3).$$

La matrice génératrice de ce code vaut

$$\boldsymbol{G} = \left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right].$$

Nous pouvons la mettre sous forme systématique en effectuant quelques opérations sur les lignes de $\boldsymbol{G}$. Par exemple en ajoutant à la première ligne les lignes 2 et 3 on obtient

$$\boldsymbol{G}' = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right],$$

ensuite en ajoutant à la seconde ligne les lignes 3 et 4 on obtient

$$\boldsymbol{G}'' = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right],$$

et en ajoutant à la troisième ligne la ligne 4 on a finalement

$$\boldsymbol{G}'' = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right].$$

Cette dernière matrice est essentiellement celle du code de Hamming, à une permutation près des trois dernières colonnes correspondant aux bits de parité.

15.5.4.6 Exemple 2 : un code BCH binaire $(15, 7)$. Soit $g(D) = 1+D^4+D^6+D^7+D^8$. Ce polynôme est un facteur de $D^{15} - 1$, car on a

$$D^{15} - 1 = (1 + D^4 + D^6 + D^7 + D^8)(1 + D^4 + D^6 + D^7).$$

Il définit donc bien un code cyclique de type $(15, 7)$.

On peut montrer que ce code à une distance minimale de 5, et peut donc corriger des erreurs doubles.

15.5.4.7 Exemple 3 : code de Golay. Il s'agit d'un code cyclique binaire $(23, 12)$ parfait et capable de corriger trois erreurs. Le polynôme générateur de ce code est

$$g(D) = 1 + D^2 + D^4 + D^5 + D^6 + D^{10} + D^{11}.$$

On montre que la distance minimale du code vaut 7.

Montrons que le code est parfait. Pour s'en convaincre il suffit de faire la somme des tailles des $t-$sphères de rayon 3 qui entourent chacun des 2^k mots du code. Or, pour un mot du code $x(D)$ cette sphère contient 1 vecteur à distance nulle $(x(D))$, C_{23}^1 mots à distance 1, C_{23}^2 mots à distance 2, et C_{23}^3 mots à distance 3. Au total on obtient que chaque $3-$sphère contient

$$1 + C_{23}^1 + C_{23}^2 + C_{23}^3 = 2048 = 2^{11}$$

mots de code.

Ces $3-$sphères ne se recouvrent pas et l'union des 2^{12} sphères correspondant aux 2^{12} mots de code contient donc

$$2^{12}2^{11} = 2^{23}$$

vecteurs différents, c'est-à-dire qu'elle couvre tous les mots binaires de longueur 23. Le code est donc bien parfait.

15.5.4.8 Existence de codes parfaits. Un résultat important de la théorie des codes algébriques concerne l'existence des codes binaires parfaits. Ce résultat dit que les seuls codes parfaits binaires sont les suivants (ou des codes équivalents) :

1. Les codes de répétition $(n, 1)$ de longueur n impaire.
2. Les codes de Hamming.
3. Le code de Golay $(23, 12)$.

Tous ces codes sont cycliques.

15.5.4.9 Détection des erreurs en rafale. Pour un code (n, k), un pattern d'erreur de type rafale est un mot de longueur n ayant exactement l bits consécutifs non-nuls, modulo décalage cyclique. On retrouve souvent ce type d'erreurs en pratique (p.ex. poussières sur un CD-ROM, bruit ponctuel dans un canal de télécommunications).

Un code linéaire détecte alors des erreurs en rafales de longueur l, si aucun pattern d'erreur de type rafale de longueur l ne fait partie du code. En pratique on s'intéresse plutôt à des codes qui sont capables de détecter des erreurs en rafale de longueur quelconque $l \leq l_{\text{max}}$.

On a le résultat fondamental suivant : *tous les codes cycliques* (n, k) *détectent les erreurs en rafale de longueur* l*, pour autant que* $l \leq n - k$.

La démonstration est quasi immédiate. En effet, si un mot de type rafale de longueur $l \leq n - k$ faisait partie du code, alors on en déduirait (par décalage cyclique) l'existence d'un polynôme de degré $l - 1$ faisant partie du code, ce qui est impossible puisque le polynôme générateur est de degré $n - k$ et que ce polynôme est le polynôme non-nul de degré minimal.

Donc, le code de Hamming $(7, 4)$ détecte les erreurs en rafale de longueur 3, le code BCH $(15, 7)$ détecte les rafales de longueur au plus 8, et le code de Golay $(23, 12)$ détecte les rafales de longueur 11.

Notons qu'on peut également étudier la capacité de *correction* des erreurs en rafale. Nous renvoyons le lecteur intéressé à la littérature plus spécialisée (p.ex. [Adá91]) pour une discussion de divers types de codes qui corrigent les erreurs multiples en rafale. Nous nous contentons ici de discuter une technique dite *d'entrelacement* qui permet de construire un code qui corrige jl erreurs en rafale à partir d'un code qui en corrige l.

15.5.4.10 Entrelacement de codes cycliques. Soit $\mathcal{C}$ un code cyclique (n, k). Alors l'entrelacement d'ordre j de ce code consiste en un code (jn, jk) dont les mots de code sont obtenus en choisissant j mots du code $\mathcal{C}$ et en alternant leurs symboles. Cela revient donc à décomposer un bloc de jk symboles source en j blocs de longueur k, à encoder chacun de ces blocs à l'aide du code $\mathcal{C}$ et au lieu de concatener les j blocs de longueur n ainsi obtenus on les entrelace pour la transmission et on les désentrelace avant décodage, bloc par bloc.

L'entrelacement permet de construire des codes qui ont de bonnes propriétés de détection et/ou de correction d'erreurs en rafale, sans augmenter le débit de code. En effet, si $\mathcal{C}$ corrige (resp. détecte) les rafales de longueur l, $\mathcal{C}'$ obtenu par entrelacement d'ordre j de $\mathcal{C}$ corrigera (resp. détectera) des rafales de longueur jl, alors que les deux codes auront le même débit.

Il est intéressant de considérer le cas particulier des codes cycliques. Notons que l'entrelacement d'ordre quelconque d'un même code cyclique produit encore un code cyclique. De plus, si

$$g(D) = g_1 + g_2 D + g_3 D^2 + \cdots + g_{n-k+1} D^{n-k}$$

est le polynôme générateur de $\mathcal{C}$ alors le polynôme générateur de $\mathcal{C}'$ est donné par

$$g'(D) = g_1 + g_2 D^j + g_3 D^{2j} + \cdots + g_{n-k+1} D^{j(n-k+1)},$$

en d'autres termes on a

$$g'(D) = g(D^j).$$

Notons que ce type de technique est utilisée dans le contexte des systèmes d'enregistrement sur disque compact. A titre d'information, nous signalons que les techniques utilisées en pratique font appel aux codes de Reed-Solomon et à l'entrelacement, et permettent de corriger des rafales jusqu'à une longueur de 4000, ce qui est nécessaire pour offrir une robustesse suffisante étant donné la taille des grains de poussière et des rayures par rapport à la densité d'écriture sur les disques compacts.

15.5.5 Codes BCH et Reed-Solomon

Au-delà de l'intérêt intellectuel et pédagogique des codes cycliques liés aux propriétés intéressantes de l'algèbre des polynômes modulo $D^n - 1$, un des intérêts pratiques de cette famille de codes est qu'elle permet de construire des codes qui répondent à une spécification en terme de distance minimale d. Il s'agit des codes BCH (Bose, Chaudhury et Hocquenghem), dont les codes de Hamming constituent le cas particulier pour $d = 3$, et dont nous avons par ailleurs vu un exemple ci-dessus.

Le codes algébriques que nous venons de décrire dans le cas binaire, pour la facilité de l'exposé, peuvent être construits sur un corps fini de taille quelconque. D'ailleurs, les codes algébriques les plus puissants font appel à des corps finis non binaires, et la plupart du temps à des corps d'extension construits à partir du corps binaire, comprenant 2^q symboles différents, en pratique représentés par des vecteurs binaires de longueur q.

L'appendice à ce chapitre fournit les éléments de base nécessaires à la compréhension de ce procédé de construction des corps finis (corps de Gallois). La difficulté tient essentiellement dans la construction d'une opération de multiplication qui possède les propriétés requises par un corps. (Lorsque k est un nombre non premier, l'opération de multiplcation “modulo k” ne convient pas. Nous suggérons au lecteur intéressé de consulter l'appendice au chapitre pour avoir une idée de la manière de procéder dans ce cas.)

15.6 UTILISATION DU CANAL GAUSSIEN

Avant de procéder avec une famille très importante de codes, nous allons discuter de la manière dont on peut utiliser un canal Gaussien, avec ou sans schéma de modulation numérique. Cette discussion est importante, car le modèle canal Gaussien est un modèle de référence, sinon universel, tout de même souvent réaliste en pratique. D'autre part, comme nous allons le voir, le schéma de modulation peut limiter la capacité du canal, s'il est mal adapté aux propriétés de celui-ci.

Nous avons vu au chapitre 10 que la capacité (en bits par seconde) du canal Gaussien avec limitation de puissance vaut

$$C = f_0 \log\left(1 + \frac{P}{N_0 f_0}\right),$$

où N_0 est la densité spectrale de puissance du bruit (ceci est une donnée pour l'ingénieur) et où P et f_0 constituent des données du système de modulation utilisé. L'ingénieur peut souvent jouer sur ces deux paramètres afin d'atteindre la capacité souhaitée. En particulier, l'augmentation de la puissance du signal à la réception P permet évidemment d'augmenter la capacité avec comme seule contrainte la faisabilité physique et économique. D'autre part, à puissance fixée, l'augmentation de la bande passante f_0 du système de modulation permet aussi d'augmenter la capacité, mais cette fois avec une limite supérieure qui vaut

$$\frac{P}{N_0} \log e.$$

Une fois que les deux paramètres constructifs f_0 et P sont fixés, se pose alors la question de savoir comment utiliser les ressources de manière à maximiser le débit sur le canal tout en rencontrant les spécifications de fiabilité. Le premier choix est celui d'un schéma de communication où on identifie le rôle joué par l'encodeur, le modulateur, le décodeur et éventuellement le démodulateur. Par modulation, nous entendons ici passage d'un alphabet discret (p.ex. suite de bits) à un alphabet de signaux réels (série temporelle), et par démodulation nous entendons l'opération inverse.

On peut essentiellement distinguer trois schémas

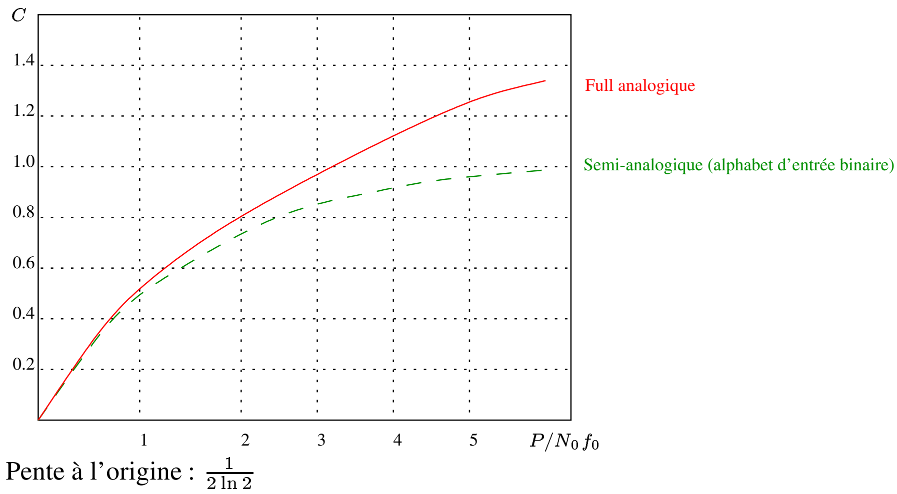


Figure 15.3. Capacités relatives en mode "full analogique" et "semi-analogique" ($q = 2$)

1. Full analogique : l'encodeur associe directement à un message source parmi $M = 2^k$, une suite de nombres réels de longueur n, et le décodeur choisit sur base de la suite de nombres réels reçue un des M messages.

2. Semi analogique : l'encodeur associe au message source un mot de code de longueur n construit sur un alphabet discret (disons de taille q), le modulateur convertit chacun des symboles du code en un nombre réel, le décodeur choisit sur base de la séquence de nombres réels reçus un mot de code.

3. Full numérique : l'encodeur associe au message source un mot de code de longueur n construit sur un alphabet discret (disons de taille q), le modulateur convertit chacun des symboles du code en un nombre réel, le démodulateur convertit chaque nombre réel reçu en un symbole du code, et enfin le décodeur choisit un mot de code sur base de la séquence de symboles de longueur n que lui présente le démodulateur. On appelle ce type de décodage le décodage à décisions fermes (nous reviendrons là-dessus un peu plus loin).

La démonstration du second théorème de Shannon fait appel au modèle "full analogique", et montre qu'avec ce type de procédé il est possible d'atteindre la capacité. Le théorème de non-création d'information nous dit que le modèle semi analogique risque de conduire à une capacité plus faible, et que le modèle full numérique risque de conduire à une réduction de capacité supplémentaire.

Dans le modèle full analogique on a une capacité (en bits par utilisation du canal)

$$C_{\text{F-a}} = \frac{1}{2} \log \left(1 + \frac{P}{N_0 f_0} \right).$$

Dans le modèle semi analogique la quantité d'information qui peut être transmise par utilisation du canal est nécessairement limitée par la taille de l'alphabet :

$$C_{\text{S-a}} \leq \log q.$$

Par exemple la figure 15.3 montre graphiquement la relation entre la capacité du canal en mode "full analogique" et en mode "semi-analogique" (ici à alphabet d'entrée binaire) en fonction du rapport signal bruit $P/N_0 f_0$. (Nous ne reproduisons pas les calculs qui conduisent à ces courbes.) On constate une croissance asymptotique de la courbe "full analogique" alors que la courbe "semi-analogique" a une asymptote horizontale à la valeur maximale $C_{\text{S-a}}^{\max} = \log q = 1$, conséquence de la restriction à l'alphabet binaire.

Donc, pour que la contrainte introduite par un alphabet de canal discret ne soit pas trop limitative, il faut que

$$q > \sqrt{1 + \frac{P}{N_0 f_0}} \quad \text{ou bien} \quad \frac{P}{N_0 f_0} < q^2 - 1.$$

Le tableau suivant reprend pour les valeurs de q égales aux puissances entières de 2 (celles qui donnent lieu à des corps d'extension binaires) la relation entre la taille de l'alphabet minimal et le rapport signal bruit du canal :

q	$\frac{P}{N_0 f_0}$	idem (dB)
2	3	4,77
4	15	11,76
8	63	17,99
16	255	24,07
32	1023	30,10
64	4095	36,12
128	16383	42,14

En particulier, on voit que pour l'exemple de la ligne téléphonique du chapitre 10, on doit prendre une taille d'alphabet d'entrée au moins égal à 16, afin de bien exploiter le (relativement) faible niveau de bruit dans ce type de canal. Effectivement les modems travaillent avec des tailles d'alphabet de canal relativement élevées.

Le troisième mode d'utilisation du canal conduit en principe à une réduction supplémentaire de la capacité. Si le rapport signal bruit est très élevé, on peut se convaincre que l'effet sera négligeable par rapport à l'effet de quantification du signal à l'entrée. La capacité restera limitée à $\log q$. Par contre, lorsque le rapport signal bruit est faible, la limitation de taille de l'alphabet d'entrée ne joue plus (cf figure 15.3), mais dans ce cas on peut montrer (dans le cas binaire) que la capacité est réduite (à cause des décisions fermes) par un facteur $\frac{2}{\pi} \approx 0.64$, ce qui est non-négligeable.

On peut synthétiser ce qui vient d'être dit de la manière suivante : le fait d'utiliser un alphabet discret à l'entrée du canal conduit à une perte irréversible de capacité, d'autant plus grande que le rapport signal bruit est élevé. Donc, si le rapport signal bruit est élévé il faut utiliser un codage de canal à alphabet de taille assez élevée pour qu'on puisse négliger l'effet de quantification. Si le rapport signal bruit est faible (de l'ordre de zéro dB) alors on peut se contenter d'utiliser une modulation binaire, mais il faut éviter d'utiliser les décisions fermes (décisions symbole par symbole) à la sortie du canal, et au contraire il faut utiliser lors du décodage de canal toute l'information disponible à la sortie du canal sous forme de nombres réels.

Nous allons maintenant discuter de techniques de codage et de décodage qui permettent de s'adapter à ces conditions.

15.7 CODES CONVOLUTIONNELS - APPROCHE PROBABILISTE

Une autre façon de coder des messages de source fait appel à la notion de convolution. Elle repose également sur une structure de corps fini et est donc sujette aux mêmes restrictions que les codes algébriques.

Par ailleurs, l'opération de convolution étant linéaire, pour une longueur donnée des messages d'entrée, les séquences de sortie d'un codeur convolutionnel forment également un espace linéaire. Cependant, la manière dont ces séquences de sortie sont produites ne nécessite pas le choix préalable d'une longueur de blocs.

La grande popularité des codes convolutionnels provient sans doute essentiellement de l'existence d'un algorithme de décodage efficace (algorithme de *Viterbi*) au sens du maximum de vraisemblance. Comme expliqué au cours oral, cet algorithme peut être "dégradé" pour effectuer le décodage à distance de Hamming minimale, mais peut également exploiter directement la matrice de transition du canal dans le cas général.

De plus, cet algorithme permet d'exploiter les *décisions souples* en sortie d'un canal continu. On entend par là qu'il est capable d'exploiter directement les signaux continus reçus à la sortie du canal : les signaux de sortie du canal sont vus comme le résultat de la superposition des signaux transmis (choisis en fonction de la sortie du codeur convolutionnel par le système de modulation) et d'un bruit additif dont les effets aux différents instants d'échantillonnage sont indépendants. La discussion de la section précédente nous a appris que cette approche permet en principe d'éviter une réduction de capacité (facteur 0.64).

Nous allons tout d'abord montrer comment on peut générer un code convolutionnel à l'aide d'un ensemble de de registres à décalage et des opérateurs simples de type addition modulo q. A nouveau nous allons restreindre notre attention au cas d'un alphabet binaire.

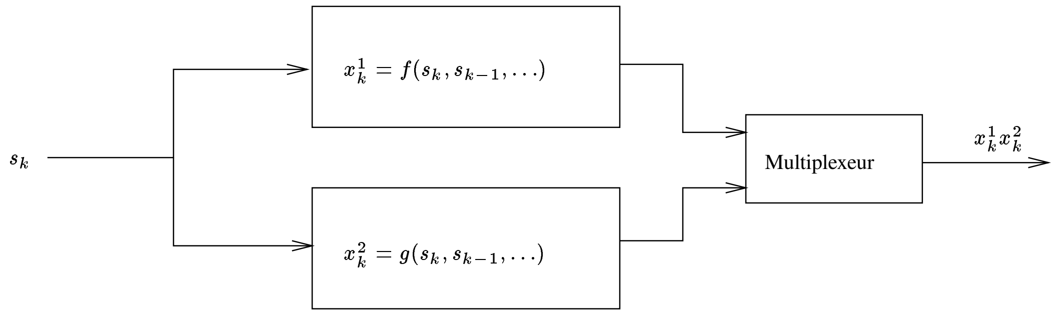


Figure 15.4. Représentation graphique d'un encodeur convolutionnel de débit $\frac{1}{2}$

15.7.1 Encodeurs convolutionnels

Les encodeurs convolutionnels utilisent des alphabets d'entrée et de sortie identiques tout comme les codes algébriques. Ils peuvent fonctionner sur un alphabet de taille quelconque pour autant qu'on puisse y définir une structure de corps (donc la taille doit être une puissance entière d'un nombre premier).

La figure 15.4 montre de façon générique le fonctionnement d'un code convolutionnel (ici de débit 0.5). Le message d'entrée est utilisé pour alimenter séquentiellement un certain nombre de modules (ici 2) qui calculent chacun un symbole de sortie à chaque pas de temps à partir des symboles d'entrée obtenus à cet instant et aux instants précédents. Les différentes sorties sont ensuite multiplexées et envoyées séquentiellement sur le canal. Le débit du code est égal à l'inverse du nombre de modules : pour chaque symbole source lu à l'entrée on envoie $\frac{1}{R}$ symboles sur le canal.

Pour que le code puisse réellement mériter le terme de convolutionnel, il faut que les relations entre les séquences d'entrée et de sortie de chaque module soient linéaires et invariantes dans le temps. La linéarité signifie que les mots de code associés à des combinaisons linéaires de séquences d'entrée correspondront à la combinaison linéaire des mots de code de chacune de ces séquences; en particulier une séquence d'entrée comportant N zéros produira une séquence de sortie comportant $\frac{N}{R}$ ($2N$ dans le cas de la figure 15.4) symboles nuls. L'invariance dans le temps signifie que si on envoie un message source décalé dans le temps (préfixé par j symboles nuls), on doit retrouver à la sortie de chacun des modules le mot de code correspondant décalé de la même manière dans le temps (c'est-à-dire préfixé par j symboles nuls). Si l'encodeur comprend i modules, le mot de code envoyé sur le canal sera alors préfixé de ij symboles nuls.

Un encodeur convolutionnel peut être représenté par un circuit comportant des registres à décalage et des opérateurs d'addition et de multiplication par une constante. Dans le cas binaire, la multiplication par une constante est triviale et il n'est pas nécessaire de la représenter explicitement, et de plus les opérations d'addition sont équivalentes à une opération logique de "ou exclusif".

La figure 15.5 représente un exemple simple d'encodeur convolutionnel construit sur un alphabet binaire. Il s'agit d'un encodeur de mémoire 2 (car il dispose de deux régistres internes qui mémorisent les deux symboles d'entrée précédents) : la mémoire est donc ici caractérisée par deux symboles binaires (e_k^1, e_k^2) qui représentent le contenu des registres à décalage à l'instant k. On doit supposer qu'avant de recevoir le premier bit source, les deux registres sont dans l'état nul (sinon le code ne pourrait pas être linéaire). Le fonctionnement du système est le suivant : à chaque pas de temps k on combine d'abord les valeurs de l'entrée s_k et de la mémoire e_k^i pour calculer les sorties, ensuite chaque registre à décalage est mis à jour par la valeur qui figure à son entrée.

Notons que l'encodeur ne définit pas explicitement les longueurs k des messages de source et des mots code (n) qui seront envoyés sur le canal, mais seulement la relation $n = 2k$.

15.7.2 Transformée en D

Note. Pour être cohérent avec les notations usuellement utilisées en théorie des systèmes, nous allons dans cette section numéroter les symboles source en commmençant par 0.

Le fonctionnement de l'encodeur de la figure 15.5 est analogue à celui d'un système en temps discret, où la séquence d'entrée représente l'entrée, la mémoire l'état, et la séquence de sortie la sortie. Puisque l'état initial est

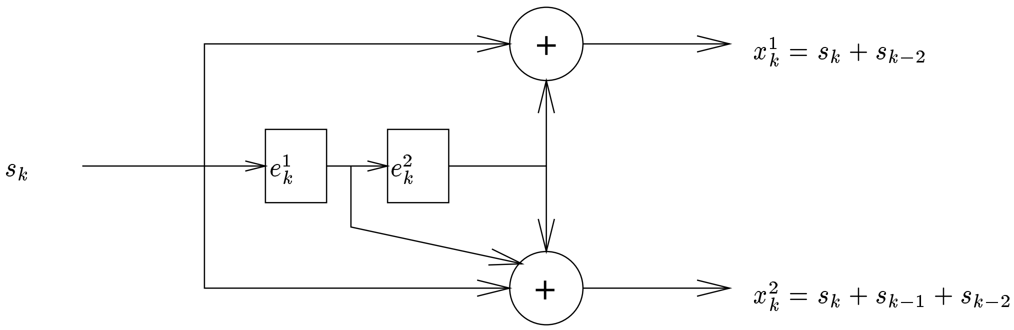


Figure 15.5. Encodeur convolutionnel non-récursif et non systématique

relaxé, on peut représenter la relation entre l'entrée et les sorties au moyen de la transformée en D (l'équivalent de la tranformée en z pour les systèmes en temps discret).

Dans ce formalisme, une séquence de symboles (d'entrée, de sortie, ou d'états d'un registre) est représentée par un série formelle en la variable D (cette variable représente l'opérateur de retard unitaire, c'est-à-dire de décalage dans le temps vers la droite) :

$$\boldsymbol{s}(D) = s_0 + s_1 D + \cdots + s_k D^k + \cdots \tag{15.18}$$
$$\boldsymbol{x}^i(D) = x_0^i + x_1^i D + \cdots + x_k^i D^k + \cdots , \tag{15.19}$$

Alors, si on définit par $\boldsymbol{h}^i(D)$ la réponse impulsionnelle du i-ième module (la séquence de sortie produite lorsque le message d'entrée est une suite qui commence par le symbole 1, puis se termine par une suite de zéros de longueur indéfinie) on a

$$\boldsymbol{x}^i(D) = \boldsymbol{h}^i(D)\boldsymbol{s}(D). \tag{15.20}$$

Par exemple, pour l'encodeur de la figure 15.5 on pourra vérifier que

$$\boldsymbol{h}^1(D) = 1 + D^2 \tag{15.21}$$
$$\boldsymbol{h}^2(D) = 1 + D + D^2. \tag{15.22}$$

On a en effet

$$\boldsymbol{e}^1(D) = D\boldsymbol{s}(D) \tag{15.23}$$
$$\boldsymbol{e}^2(D) = D\boldsymbol{e}^1(D) = D^2\boldsymbol{s}(D) \tag{15.24}$$
$$\boldsymbol{x}^1(D) = \boldsymbol{s}(D) + \boldsymbol{e}^2(D) = (1 + D^2)\boldsymbol{s}(D) \tag{15.25}$$
$$\boldsymbol{x}^2(D) = \boldsymbol{s}(D) + \boldsymbol{e}^1(D) + \boldsymbol{e}^2(D) = (1 + D + D^2)\boldsymbol{s}(D). \tag{15.26}$$

Les deux seules différences sont que nous travaillons ici avec des alphabets discrets alors qu'en théorie des systèmes on considère des signaux à valeurs réelles, et que nous considérons des durées finies alors qu'en théorie des systèmes l'axe du temps est ouvert à droite. Le nombre d'états possibles de l'encodeur est également fini et vaut au maximum q^m où q est la taille de l'alphabet et m la taille de la mémoire.

Nous allons voir que les algorithmes de décodage font appel à une modélisation explicite des états possibles à chaque instant, et on peut donc se douter déjà à ce stade (et cela sera confirmé dans la suite) que les performances de décodage souffrent si la mémoire est trop grande, ce qui constitue une limitation pour les codes convolutionnels.

15.7.3 Utilisation de diagrammes en treillis

On peut réprésenter l'encodeur par une machine d'état (similaire à un diagramme d'état d'une chaîne de Markov), où les transitions sont choisies par les symboles (aléatoires) émis par la source. De plus, avant chaque transition

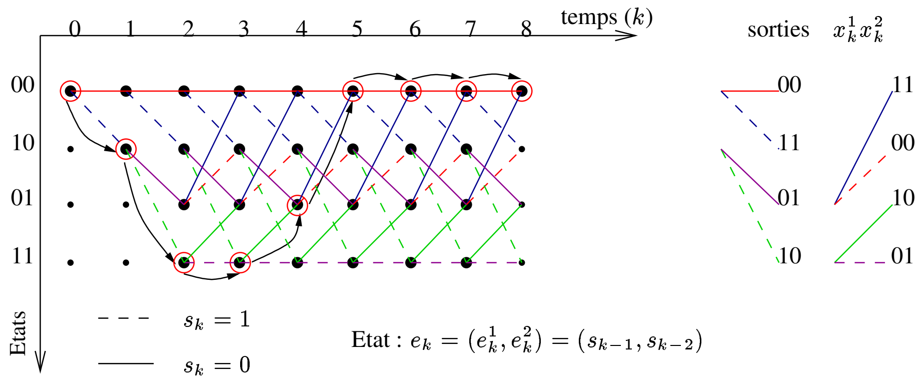


Figure 15.6. Treillis des trajectoires correspondant à l'encodeur de la figure 15.5

d'état l'encodeur émet un certain nombre de symboles. Les symboles reçus par le décodeur forment alors ce qu'on appelle habituellement une chaîne de Markov cachée, parce que les états ne sont pas directement observables.

Si nous représentons les trajectoires d'états possibles de l'encodeur au fil du temps, nous aurons une représentation des mots de code possibles générés par celui-ci. La figure 15.6 résume sous la forme d'un treillis les parcours possibles de l'encodeur de la figure 15.5.

Sur cette figure les 4 états possibles dans lesquels l'encodeur peut se trouver à tout moment sont représentés sur l'axe vertical. Les arcs qui lient des états à l'instant k à l'état à l'instant $k-1$ représentent les transitions possibles : on voit que pour chaque état de départ il n'y a que deux transitions possibles, correspondant aux deux valeurs possibles du symbole d'entrée. Ces transitions sont représentées sur la partie droite de la figure avec les valeurs des symboles de sorties correspondants.

La figure représente également le trajet dans le treillis suivi par l'encodeur lorsqu'on lui présente à l'entrée la séquence 11100000. La suite d'états parcourus est (les symboles de sortie émis avant chaque transition sont indiqués au dessus des flèches)

$$00 \xrightarrow{11} 10 \xrightarrow{10} 11 \xrightarrow{01} 11 \xrightarrow{10} 01 \xrightarrow{11} 00 \xrightarrow{00} 00 \xrightarrow{00} 00 \xrightarrow{00} 00.$$

Le mot de code correspondant est donc

$$11, 10, 01, 10, 11, 00, 00, 00.$$

Le treillis comporte 2^N chemins de longueur N qui représentent les 2^N mots de code de longueur $2N$ qui peuvent être produits par l'encodeur. On peut se convaincre que tous les mots de code sont bien différents par le raisonnement suivant : $\boldsymbol{s}(D) \neq 0 \Rightarrow \boldsymbol{x}^i(D) = \boldsymbol{h}^i(D)\boldsymbol{s}(D) \neq 0$ (pour autant que $\boldsymbol{h}^i(D) \neq 0$ modulo D); or comme le code est linéaire on a $h^i(D)(\boldsymbol{s}(D) - \boldsymbol{s}'(D)) = h^i(D)\boldsymbol{s}(D) - h^i(D)\boldsymbol{s}'(D)$; donc $h^i(D)\boldsymbol{s}(D) = h^i(D)\boldsymbol{s}'(D) \Rightarrow \boldsymbol{s}(D) = \boldsymbol{s}'(D)$ où les polynômes d'entrée et de sortie sont de degré $N-1$ au plus et donc les comparaisons s'entendent "modulo D^N". Sous ces conditions, il est donc impossible que deux mots de codes identiques soient produits par deux messages de source différents, et il y a donc exactement autant de messages source différents que de mots de code différents.

On voit également qu'à partir d'un certain nombre de transitions le diagramme devient périodique.

Notons que la propriété de déchiffrabilité n'est pas nécessairement vérifiée par tous les encodeurs convolutionnels. Pour qu'un tel code soit déchiffrable, il faut et il suffit que la réponse impulsionnelle d'au moins une des sorties soit non-nulle à l'instant initial (il ne faut donc pas que la réponse impulsionnelle soit multiple de D).

15.7.4 Décodage : recherche du plus court chemin dans le treillis

Le décodage peut se faire de la manière suivante (nous supposons que nous utilisons un canal sans mémoire, et dont l'alphabet est binaire).

1. Il est clair qu'à chaque suite d'entrée correspond un chemin dans le treillis. Nous savons aussi que le code est déchiffrable : deux messages d'entrée différents donnent lieu à des chemins différents, et il suffit de trouver le chemin dans le treillis pour identifier le message envoyé à l'entrée.

2. Soit $\boldsymbol{x}(D)$ le message envoyé (dans notre exemple de longueur $2N$, résultant du codage d'un message source $\boldsymbol{s}(D)$ de longueur N). On peut aussi voir le message envoyé comme une séquence de longueur N construite sur un alphabet de taille $q' = q^m$ (où m désigne la taille de la mémoire de l'encodeur; ici $m = 2$ et $q' = 4$). Désignons par X^N cette séquence de longueur N et par Y^N la séquence correspondante obtenue en sortie (suite aux erreurs de transmission ayant lieu dans le canal). Tout se passe comme si nous utilisions maintenant un canal avec un alphabet d'ordre q^2. Ce canal reste un canal sans mémoire et sa matrice de transition est obtenue à partir de la matrice de transition du canal binaire que nous utilisons en réalité.

3. Le décodage optimal (qui minimise la probabilité de se tromper sur le message $\boldsymbol{s}(D)$) consistera donc à trouver $\hat{X}^N$ tel que $P(X^N \neq \hat{X}^N)$ soit minimale. Il faut donc choisir $\hat{X}^N$ tel que $P(\hat{X}^N|Y^N) \geq P(X^N|Y^N), \forall X^N$, ce qui est encore équivalent à trouver la séquence $\hat{X}^N$ qui maximise
$$P(Y^N|\hat{X}^N)P(\hat{X}^N).$$

4. Il faut tenir compte des contraintes du code : seules les suites X^N qui correspondent à un chemin dans le treillis sont possibles. Pour le calcul des $P(\hat{X}^N)$ possibles il faut aussi tenir compte de la loi de probabilités relative aux symboles source et du fait que l'état initial est connu.

5. Mais, en supposant que la source émet des symboles indépendants et équiprobables, tous les messages source de longueur N sont équiprobables et donc aussi tous les chemins du graphe : $P(X^N) = 2^{-N}$. Pour le décodage optimal il suffit donc de maximiser $P(Y^N|X^N)$.

6. Comme le canal est sans mémoire on a $P(Y^N|X^N) = \prod_{i=1}^N P(Y_i|X_i)$. Le problème du décodage revient donc à minimiser
$$-\log P(Y^N|X^N) = \sum_{i=1}^N -\log P(Y_i|X_i).$$

7. En d'autres termes le problème du décodage revient à trouver le chemin le moins cher dans le treillis où les $-\log P(Y_i|X_i)$ mesurent les “coûts” des arcs.

15.7.5 Décisions souples

Nous venons de poser le problème du décodage de canal lorsque la sortie du canal est discrète. Nous avons cependant vu à la section 15.6 que pour le canal Gaussien (le modèle habituel en télécommunications) il est souhaitable de l'utiliser en mode semi-analogique (dans les conditions de bruit élevé on évite ainsi une perte irrévocable de capacité) et donc d'utiliser pour le décodage en sortie un alphabet continu sur les Y^N. Lorsqu'on fait cela on parle de *décodage à décisions souples*.

Le décodage que nous venons de décrire peut très bien s'accomoder de ce type de modèle. Il suffit en effet de considérer dans le calcul de
$$P(Y_i|X_i)$$
que les deux composantes de Y_i sont distribuées de manière indépendante (conditionnellement à X_i) selon une loi Gaussienne dont la variance découle directement du rapport signal bruit. Il s'en suit que l'utilisation d'un alphabet continu en sortie a pour seul effet de modifier la *valeur* des arcs du treillis et ne remet pas en question les algorithmes que nous discuterons ci-dessous. Tout se passe alors comme si nous avions affaire à un canal discret dont la matrice de transition est dépendante de l'instant k (et peut être déduite des valeurs continues observées en sortie pour les symboles correspondants).

Nous donnerons, dans le cadre des Turbo-codes une description plus détaillée du décodage à décisions souples. Ci-dessous nous montrerons sur un exemple simple la différence entre le décodage à décisions souples et à décisions dures.

15.7.6 Algorithme de Viterbi

Nous venons de voir que le décodage des codes convolutionnels revient à trouver le chemin le moins cher dans un graphe. La solution par énumération n'est pas possible, car pour des longueurs utilisées en pratique (par exemple $N \approx 1000$) le nombre de messages source est tellement grand qu'il n'est même pas possible de les énumérer.

Pour ceux qui n'en sont pas convaincus faisons donc le calcul explicite : si $N = 1000$, le code comprend $2^{1000} \approx 10^{301}$ mots. Si chaque électron de l'univers (il y en a à peu près 10^{80}) était un processeur 1000 GHz capable d'évaluer la probabilité d'un mot de code en une seule instruction, on pourrait traiter ainsi $10^{12} \times 10^{80} \approx 10^{92}$ mots de code par seconde, et si on les laisse travailler pendant une durée égale à l'âge de l'univers (3×10^{17} secondes), ces ordinateurs traiteraient 3×10^{109} mots de code au total. Il faudrait encore attendre environ 10^{190} fois l'âge de l'univers pour avoir la réponse. Il est donc hors de question de stocker explicitement les mots du code et aussi de les parcourir en les construisant temporairement.

L'algorithme de Viterbi permet cependant de résoudre le problème du décodage optimal de manière efficace en tirant profit de la structure en treillis du code. On voit que pour stocker le treillis du code il suffit d'un volume de mémoire qui est proportionnel à q^m (le nombre d'états possibles) multiplié par N. L'algorithme de Viterbi exploite cette structure et construit le chemin optimal de façon séquentielle avec un nombre d'opérations linéaire en fonction de la taille du treillis.

Il est basé sur la propriété générale suivante (nous désignons par e^{K+1} une suite d'états $e_0, \ldots, e_K$ correspondant à un chemin autorisé (chaque transition $e_i \to e_{i+1}$ correspond effectivement à un arc) :

Si e^{K+1} est un chemin optimal menant vers l'état e_{K+1}, e^K est un aussi un chemin optimal vers l'état e_K .

Cette propriété dit que les préfixes d'un chemin optimal sont aussi des chemins optimaux. Elle est évidente (que se passerait-il s'il elle n'était pas vraie ?) mais a des conséquences foudroyantes du point de vue algorithmique.

En effet, si nous connaissons les q^m chemins optimaux de longueur K menant vers chacun des q^m états (noeuds en position K) et les coûts des transitions, on peut en déduire directement les chemins optimaux de longueur $K + 1$ menant vers chacun des q^m états (noeuds en position $K + 1$).

Il suffit maintenant de parcourir le graphe de gauche à droite en partant de l'état initial, et en construisant les chemins les plus courts de longueur $K = 1, 2, \ldots, N$ passant par chacun des états du treillis en position $K = 1, 2 \ldots N$.

En fin de parcours nous connaissons les chemins les plus courts qui mènent à chacun des q^m états en position N, et le mot de code correspond à celui-qui est de coût optimal. Il suffit alors de retourner en arrière et de collecter les symboles source qui correspondent à ce chemin.

15.7.6.1 Discussion. L'algorithme de Viterbi est applicable pour le décodage de codes convolutionnels ainsi que pour le décodage des codes linéaires algébriques que nous avons vus dans la section précédente. Son efficacité dépend essentiellement de la structure du treillis, dont la hauteur croît rapidement avec la mémoire de l'encodeur et la taille de l'alphabet dans le cas des codes convolutionnels. C'est pourquoi en pratique on utilise l'alphabet binaire et des encodeurs de mémoire relativement faible. Cet algorithme est également utilisé pour l'inférence des suites d'états les plus probables dans les chaînes de Markov cachées.

Le problème général du décodage au maximum de la probabilité a posteriori des codes linéaires est NP-complet, ce qui veut dire que si quelqu'un trouve un algorithme efficace pour le résoudre il provoquera par là une révolution dans le domaine de l'informatique théorique et pratique. C'est donc la structure convolutionnelle du code qui rend le décodage efficace possible.

Du point de vue du décodage, l'algorithme que nous avons décrit dans les sections précédentes fait l'hypothèse de canal et de source sans mémoire. Si l'un ou l'autre avaient de la mémoire, il ne serait pas possible de décomposer la probabilité a posteriori d'un mot de code en un produit de termes qui correspondent aux arcs du treillis, et l'algorithme ne serait plus applicable[2]. Ce n'est pas trop grave en pratique puisque la spécialisation d'un décodeur pour les canaux et/ou sources avec mémoire ne se justifie que rarement en pratique.

L'algorithme de Viterbi est un cas particulier d'une classe plus générale de méthodes d'inférence probabiliste à partir de modèles graphiques, qui constitue un domaine de recherche extrêmement actif depuis quelques années. On voit ici apparaître la connexion entre théorie de l'information et du codage et intelligence artificielle qui est en train de se construire dans le monde de la recherche [Fre99, Mac99].

[2]Il est cependant possible d'étendre la notion de treillis dans un certain nombre de cas particuliers que nous ne discuterons pas ici.

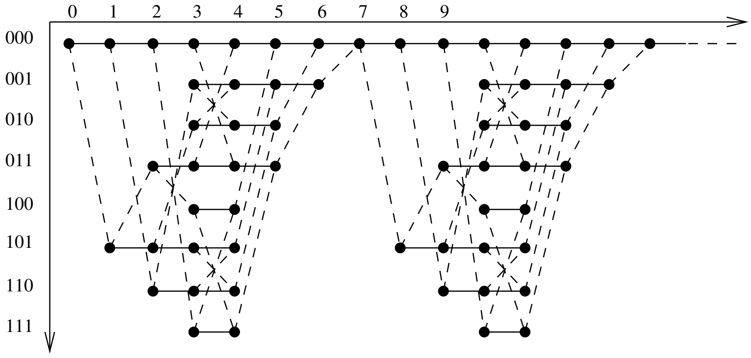


Figure 15.7. Treillis pour le code de Hamming $(7,4)$

15.7.7 Décodage à décisions souples de codes linéaires algébriques

Lors de notre discussion des codes algébriques nous avons mis au point des techniques de décodage qui se basent sur le canal discret (essentiellement le canal binaire symétrique). Nous venons de voir que l'algorithme de Viterbi permet quant à lui d'exploiter les décisions souples en sortie du canal et de ce fait de mieux exploiter l'information disponible. Il serait donc intéressant de pouvoir appliquer cette technique aussi dans le cas des codes algébriques, et il suffit pour cela d'être capable de représenter ces codes sous la forme d'un treillis.

En fait, il est possible de représenter n'importe quel code linéaire en blocs sous la forme d'un treillis de code. Mais ces treillis ont des structures différentes des treillis de codes convolutionnels. Par exemple la figure 15.7 représente le code de Hamming $(7,4)$.

Le treillis est obtenu selon la procédure suivante :

1. On définit les états à partir de la matrice de contrôle.

   P.ex. pour le code $(7,4)$ on a

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} = [\boldsymbol{P}^T \boldsymbol{I}_3]$$

2. On sait que

$$\boldsymbol{H}\boldsymbol{x}^T = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

   En d'autres mots

$$\sum_{i=1}^{7} x_i \begin{bmatrix} h_{1,i} \\ h_{2,i} \\ h_{3,i} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

3. L'état représente alors le vecteur de contrôle de parité partiel :

$$e_k = \sum_{i=1}^{k} x_i \begin{bmatrix} h_{1,i} \\ h_{2,i} \\ h_{3,i} \end{bmatrix}$$

   et

$$e_0 = e_7 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

On voit que ce treillis a une structure très différente du treillis d'un code convolutionnel : dans la première partie du treillis chaque noeud a exactement un seul arc entrant, alors que dans la seconde partie du treillis chaque noeud a exactement un seul arc sortant. D'autre part, les parties du treillis correspondant à des blocs successifs de longueur n (ici $n = 7$) sont découplés, puisque tous les chemins passant d'un bloc au suivant doivent passer par le même état (00) en position in. Cela implique que le chemin le moins cher dans un treillis correspondant au codage de l blocs successifs est obtenu en calculant les chemins les moins chers pour chacun des blocs. Cette propriété traduit le décodage par blocs de longueur fixée a priori des codes algébriques.

On constate que le nombre d'états (la hauteur du treillis) est donné par 2^{n-k}. Ce nombre peut devenir très important pour des codes de débit faible. Il serait donc intéressant de savoir s'il n'existe pas plusieurs treillis possibles pour un code donné et de disposer d'une méthode permettant de minimiser le nombre d'états de façon à augmenter la vitesse de décodage. L'investigation de ces questions ainsi que la mise au point d'algorithmes efficaces basés sur les treillis pour les codes linéaires est également un domaine de recherche très actif à l'heure actuelle [LKFF98].

15.8 MINIMISER L'ERREUR PAR BIT VS MINIMISER L'ERREUR PAR MESSAGE

Dans les développements qui précèdent nous avons défini le décodage optimal comme le décodage qui minimise en moyenne la probabilité de se tromper sur le mot de code émis, c'est-à-dire sur le message source émis. Lorsque le décodeur se trompe, on peut cependant se demander de combien il se trompe en moyenne, c'est-à-dire s'intéresser au nombre moyen de bits erronés dans le message source après décodage. D'ailleurs pour le décodage optimal on peut utiliser comme critère la minimisation du nombre moyen de bits source erronés. Cela pourrait par exemple être intéressant si le message source est encodé à l'aide d'un code permettant la détection d'erreurs multiples, de façon à pouvoir demander le renvoi d'un bloc de symboles en cas de nécessité.

Il est possible de définir des algorithmes de décodage sur base de ce critère et il faut savoir qu'en général le décodage à probabilité d'erreur minimale par mot de code donne des résultats différents du décodage à probabilité d'erreur minimale par bit source. Nous allons voir, à la section sur les turbo-codes l'algorithme BCJR qui mimimise la probabilité d'erreur par bit en se basant sur le treillis du code.

Ici nous allons illustrer trois façons différentes de poser le problème du décodage, en nous servant d'un exemple simple (relatif au code de Hamming $(7,4)$), et nous verrons que selon la manière dont on pose le problème on peut obtenir des résultats différents. Avant cela, nous allons rediscuter le problème du décodage de canal dans un cadre concret, en supposant que le canal est Gaussien et à modulation antipodale.

15.8.1 Position du problème

La source choisit un message $\boldsymbol{s}$ (que nous supposerons binaire) de longueur k qui est ensuite encodé pour donner un mot de code $\boldsymbol{x}$ binaire de longueur n. Celui-ci est transformé en une série d'impulsions d'amplitude $\pm x$, où x est fonction de la puissance d'émission utilisée. Dans la suite, nous supposerons que les sorties du canal sont normalisées de telle manière qu'on puisse considérer que $x = 1$. A la sortie du canal, un système de démodulation récupère donc un nombre réel $y_i \in]-\infty; +\infty[$ pour chacun des symboles x_i du mot de code.

La solution complète du problème de décodage consisterait à calculer la loi de probabilité conditionnelle sur l'ensemble des mots de code possibles à partir des informations reçues en sortie $\boldsymbol{y} = y_1, \ldots, y_n$:

$$P(\boldsymbol{x}|\boldsymbol{y}), \forall \boldsymbol{x} \in \mathcal{C}. \tag{15.27}$$

A partir de cette information complète, un utilisateur pourrait alors prendre des décisions optimales en fonction de ses propres objectifs : choisir le mot de code le plus probable (s'il veut minimiser la probabilité d'erreur de décodage), choisir pour chaque bit source la valeur la plus probable (s'il veut minimiser le nombre moyen de bits source érronés), ou encore calculer pour chaque bit source la loi de probabilité marginale (par exemple si nous avons affaire à des codes concaténés et que nous voulons fournir une information sur la vraisemblance des bits au décodeur de niveau supérieur).

Cependant, en pratique, la table de code est en général tellement grande qu'il n'est pas possible de représenter avec les ressources disponibles la loi $P(\boldsymbol{x}|\boldsymbol{y})$, ni a fortiori de la calculer. Par conséquent, il faut composer en réduisant nos ambitions, et on peut alors se poser les deux problèmes plus simples suivants :

1. Trouver le mot source le plus probable, i.e. $\boldsymbol{s}^*$ qui maximise $P(\boldsymbol{s}|\boldsymbol{y})$.

2. Trouver pour chaque symbole s_i, la loi de probabilité $P(s_i|\boldsymbol{y})$, et la valeur s_i^* qui la maximise.

Avant d'illustrer ces deux façons de "décoder", voyons quelle information est nécessaire à cette fin. Evidemment, puisque les mots de code sont en bijection avec les mots source, la recherche du meilleur mot source revient à la recherche du meilleur mot de code. D'autre part, si le code est systématique (ce que nous supposerons dans la suite), on a $x_i = s_i, \forall i = 1, \ldots, k$. et donc le calcul des $P(s_i|\boldsymbol{y})$ (i=1, ...,k) revient aussi au calcul des $P(x_i|\boldsymbol{y})$.

15.8.2 Rapports de vraisemblance

On a

$$P(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})P(\boldsymbol{x})}{p(\boldsymbol{y})}, \tag{15.28}$$

où $p(\boldsymbol{y}|\boldsymbol{x})$ et $p(\boldsymbol{y})$ sont des densités de probabilités puisque les variables y_i en sortie du canal sont continues.

Souvent on supposera que les différents mots de code sont équiprobables a priori. Mais, étant donné une loi de probabilité sur les mots de source (pas nécessairement uniforme) on peut toujours en déduire a priori la loi $P(\boldsymbol{x})$. Dans la suite nous verrons que généralement le type d'information dont on dispose sur les symboles source fait cependant l'hypothèse que ceux-ci sont indépendants :

$$P(\boldsymbol{s}) = \prod_{i=1}^{k} P(s_i).$$

Partant de cette information, on peut en déduire la probabilité des mots de code de façon directe par

$$P(\boldsymbol{x}(\boldsymbol{s})) = P(\boldsymbol{s}) = \prod_{i=1}^{k} P(s_i),$$

où $\boldsymbol{x}(\boldsymbol{s})$ désigne le mot de code associé au message source $\boldsymbol{s}$.

Le terme $p(\boldsymbol{y})$ est un terme de normalisation qui assure que les lois de probabilités calculées somment à 1. Pour le problème 1, nous n'aurons pas besoin de calculer ce terme explicitement, car comme nous l'avons déjà dit la solution du problème 1 est indépendante de cette normalisation.

Pour le problème 2, on pourra toujours renormaliser les $P(x_i|\boldsymbol{y})$ a posteriori, pour chaque valeur de i. D'ailleurs, comme nous travaillons avec des alphabets binaires, il est en fait suffisant de connaître les rapports *de vraisemblance*

$$l_i(\boldsymbol{y}) \quad = \quad \frac{P(x_i = 1|\boldsymbol{y})}{P(x_i = 0|\boldsymbol{y})} \tag{15.29}$$

ou encore

$$L_i(\boldsymbol{y}) \quad = \quad \log l_i(\boldsymbol{y}) \tag{15.30}$$

dont on pourra déduire les probabilités par les formules

$$P(x_i = 1|\boldsymbol{y}) \quad = \quad \frac{\exp\{+L_i(\boldsymbol{y})\}}{1 + \exp\{+L_i(\boldsymbol{y})\}} \tag{15.31}$$

$$P(x_i = 0|\boldsymbol{y}) \quad = \quad \frac{\exp\{-L_i(\boldsymbol{y})\}}{1 + \exp\{-L_i(\boldsymbol{y})\}}. \tag{15.32}$$

Nous verrons plus loin que cette représentation facilite l'écriture d'un certain nombre de formules dans le cas où les alphabets sont binaires.

On définit aussi le rapport de vraisemblance a priori sur un symbole source par

$$L_i^e = \log \frac{P(x_i = 1)}{P(x_i = 0)}. \tag{15.33}$$

Si les symboles sont équiprobables a priori on a évidemment $L_i^e = 0$. On voit que le décodage symbole par symbole revient en fait à utiliser l'information apportée par le vecteur $\boldsymbol{y}$ pour mettre à jour les rapports de vraisemblance L_i. Nous verrons que cette vision est exploitée systématiquement dans le contexte des Turbo-codes.

Avant de passer à notre exemple illustratif, il nous reste à discuter le terme $p(\boldsymbol{y}|\boldsymbol{x})$ relatif au modèle du canal. Si le canal est sans mémoire, nous savons que ce terme peut être factorisé de la manière suivante

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{n} p(y_i|x_i). \tag{15.34}$$

D'autre part, notre canal est Gaussien, ce qui implique que

$$p(y_i|x_i = 1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i-1)^2}{2\sigma^2}\right) \tag{15.35}$$

$$p(y_i|x_i = 0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i+1)^2}{2\sigma^2}\right), \tag{15.36}$$

et on constate à nouveau que toute l'information apportée par un y_i peut être compactée sous la forme d'un rapport de vraisemblance

$$L(y_i) = \log \frac{p(y_i|x_i = 1)}{p(y_i|x_i = 0)} = \frac{2y_i}{\sigma^2}, \tag{15.37}$$

car ce rapport de vraisemblance permet de trouver $p(y_i|x_i = 0)$ et $p(y_i|x_i = 1)$ à une constante multiplicative près (à l'aide d'équations similaires aux équations (15.31-15.32)), et l'équation (15.28) peut être multipliée par une constante arbitraire de normalisation γ pour autant que celle-ci soit indépendante de $\boldsymbol{x}$.

Par exemple, nous pouvons renormaliser en multipliant pour chaque valeur de i les $p(y_i|x_i = 1)$ et $p(y_i|x_i = 0)$ par un facteur γ_i de telle manière que leur somme vaille 1. Ces grandeurs ressemblent alors étrangement à une loi de probabilité discrète sur x_i : en fait il s'agit de la loi qui ne tient compte que de l'information apportée par la sortie y_i et qui suppose que les x_i sont équiprobables. C'est cette loi qui serait utilisée pour trancher de manière dure sur la valeur du i-ième symbole sans tenir compte des contraintes du code : le symbole x_i choisi est celui qui est le plus probable selon cette loi, et la probabilité locale de se tromper vaut $1 - p(y_i|x_i)$. C'est cette dernière probabilité qui caractérise la probabilité d'erreur du canal discretisé à l'instant i, à laquelle nous avons déjà fait référence plus haut.

Pour obtenir cette normalisation il suffit de prendre

$$\gamma_i = \frac{1}{p(y_i|x_i = 1) + p(y_i|x_i = 0)}, \tag{15.38}$$

et en définissant $P(y_i|x_i = 1) \triangleq \gamma_i p(y_i|x_i = 1)$ et $P(y_i|x_i = 0) \triangleq \gamma_i p(y_i|x_i = 0)$ on obtient en substituant dans (15.28)

$$P(\boldsymbol{x}|\boldsymbol{y}) = \frac{P(\boldsymbol{x}) \prod_{i=1}^{n} P(y_i|x_i)}{\gamma p(\boldsymbol{y})}, \tag{15.39}$$

où $\gamma = \prod_{i=1}^{n} \gamma_i$.

En pratique, il suffit de calculer le terme $P(\boldsymbol{x}) \prod_{i=1}^{n} P(y_i|x_i)$ puis de normaliser pour en déduire une loi de probabilités conditionnelle sur $\boldsymbol{x}$.

Si on est intéressé par une loi de probabilités conditionnelles sur l'un des x_i, il suffit en principe de calculer les termes

$$a_i = \sum_{\boldsymbol{x} \in \mathcal{C}} P(\boldsymbol{x}) \prod_{j=1}^{n} P(y_j|x_j) \delta(x_i = 1) \tag{15.40}$$

et

$$b_i = \sum_{\boldsymbol{x} \in \mathcal{C}} P(\boldsymbol{x}) \prod_{j=1}^{n} P(y_j|x_j) \delta(x_i = 0) \tag{15.41}$$

puis de normaliser en divisant par $a_i + b_i$ pour obtenir $P(x_i = 1|\boldsymbol{y})$ et $P(x_i = 0|\boldsymbol{y})$. Nous verrons, dans le cadre des turbo-codes que ce type de calcul peut être obtenu de manière récursive et très efficace en faisant appel aux rapports de vraisemblance.

Table 15.1. Vraisemblances de mots du code de Hamming lorsque qu'on a reçu en sortie un vecteur $\boldsymbol{y}$ dont les composantes ont les rapports de vraisemblance $(\frac{0.1}{0.9}, \frac{0.4}{0.6}, \frac{0.9}{0.1}, \frac{0.1}{0.9}, \frac{0.1}{0.9}, \frac{0.1}{0.9}, \frac{0.3}{0.7})$

Mot de code $\boldsymbol{x}$	$\ell(\boldsymbol{x}\|\boldsymbol{y})$	$P(\boldsymbol{x}\|\boldsymbol{y})$
0000000	0.0275562	0.2456
0001011	0.0001458	0.0013
0010111	0.0013122	0.0117
0011100	0.0030618	0.0273
0100110	0.0002268	0.0020
0101101	0.0000972	0.0009
0110001	0.0708588	0.6315
0111010	0.0020412	0.0182
1000101	0.0001458	0.0013
1001110	0.0000042	0.0000
1010010	0.0030618	0.0273
1011001	0.0013122	0.0117
1100011	0.0000972	0.0009
1101000	0.0002268	0.0020
1110100	0.0020412	0.0182
1111111	0.0000108	0.0001
Total	0.1122000	1.0000

15.8.3 Exemple illustratif

Considérons le code de Hamming $(7, 4)$ (page 263). Nous supposons que les mots source et donc de code sont équiprobables (on peut donc se passer de faire intervenir le facteur $P(\boldsymbol{x})$ dans les calculs de vraisemblance), et que nous avons reçu un vecteur de 7 nombres réels en sortie du canal dont les vraisemblances normalisées sont données par

$$\boldsymbol{a} = (P(y_1|x_1 = 1), (P(y_2|x_2 = 1), \ldots, (P(y_7|x_7 = 1)) \quad = \quad (0.1, 0.4, 0.9, 0.1, 0.1, 0.1, 0.3), \tag{15.42}$$

et donc

$$\boldsymbol{b} = (P(y_1|x_1 = 0), (P(y_2|x_2 = 0), \ldots, (P(y_7|x_7 = 0)) \quad = \quad (0.9, 0.6, 0.1, 0.9, 0.9, 0.9, 0.7). \tag{15.43}$$

Comme la table de code est de taille réduite, nous pouvons ici, pour le besoin de l'illustration, calculer explicitement la vraisemblance de chaque mot de code et donc sa probabilité (en normalisant). On obtient la table 15.1. La seconde colonne de cette table est définie par

$$\ell(\boldsymbol{x}|\boldsymbol{y}) = \prod_{i=1}^{7} a_i^{x_i} b_i^{(1-x_i)}, \tag{15.44}$$

la troisième est obtenue en normalisant la seconde colonne et la quatrième indique simplement de façon graphique l'information de la troisième colonne. On voit que le mot de code le plus probable a posteriori est le mot 0110001.

Notons ici au passage que le décodage à décisions dures aurait donné en sortie du canal le mot 0010000 dont le plus proche voisin dans la table de code au sens de la métrique de Hamming est évidemment le mot 0000000. On voit donc sur cet exemple que le décodage à décisions souples choisit un mot différent du mot choisi par l'approche algébrique.

La table 15.2 fournit les probabilités $P(x_i|\boldsymbol{y})$ des 7 bits du mot de code. Elle est obtenue à partir de la table 15.1 en marginalisant. En d'autres termes la probabilité $P(x_i = 1|\boldsymbol{y})$ est obtenue à partir de la somme des lignes de la table 15.1 correspondant aux mots de code tels que $x_i = 1$. On montre également dans cette table la valeur des rapports de vraisemblance des valeurs reçues. On voit à partir de la table que les valeurs les plus probables pour les 7 bits correspondent au mot 0110001, c'est-à-dire au mot également obtenu en sortie du décodage optimal par mot de code. Nous allons voir immédiatement à partir d'un autre exemple que ceci n'est

Table 15.2. Vraisemblances et probabilités a posteriori des bits du code de Hamming lorsque qu'on a reçu en sortie un vecteur $\boldsymbol{y}$ dont les composantes ont les rapports de vraisemblance $(\frac{0.1}{0.9}, \frac{0.4}{0.6}, \frac{0.9}{0.1}, \frac{0.1}{0.9}, \frac{0.1}{0.9}, \frac{0.1}{0.9}, \frac{0.3}{0.7})$

i	Vraisemblance normalisée		Probabilités a posteriori	
	$P(y_i\|x_i=1)$	$P(y_i\|x_i=0)$	$P(x_i=1\|\boldsymbol{y})$	$P(x_i=0\|\boldsymbol{y})$
1	0.1	0.9	0.061	0.939
2	0.4	0.6	0.674	0.326
3	0.9	0.1	0.746	0.254
4	0.1	0.9	0.061	0.939
5	0.1	0.9	0.061	0.939
6	0.1	0.9	0.061	0.939
7	0.3	0.7	0.659	0.341

Table 15.3. Vraisemblances de mots du code de Hamming lorsque qu'on a reçu en sortie un vecteur $\boldsymbol{y}$ dont les composantes ont les rapports de vraisemblance $(\frac{0.2}{0.8}, \frac{0.2}{0.8}, \frac{0.9}{0.1}, \frac{0.2}{0.8}, \frac{0.2}{0.8}, \frac{0.2}{0.8}, \frac{0.2}{0.8})$

Mot de code $\boldsymbol{x}$	$\ell(\boldsymbol{x}\|\boldsymbol{y})$	$P(\boldsymbol{x}\|\boldsymbol{y})$
0000000	0.0262144	0.3006
0001011	0.0004096	0.0047
0010111	0.0036864	0.0423
0011100	0.0147456	0.1691
0100110	0.0004096	0.0047
0101101	0.0001024	0.0012
0110001	0.0147456	0.1691
0111010	0.0036864	0.0423
1000101	0.0004096	0.0047
1001110	0.0001024	0.0012
1010010	0.0147456	0.1691
1011001	0.0036864	0.0423
1100011	0.0001024	0.0012
1101000	0.0004096	0.0047
1110100	0.0036864	0.0423
1111111	0.0000576	0.0007
Total	0.0872000	1.0000

pas vrai en général. Mais ici, nous voudrions insister sur le fait que le décodage par bit donne une information sur la fiabilité des bits source obtenus après décodage. Par exemple, à partir de la table 15.2 on voit que les bits 1 et 4 sont plus fiables que les bits 2 et 3. Ce type d'information pourrait être exploitée à un niveau supérieur par exemple, si $\boldsymbol{s}$ est lui-même un message codé.

Voyons ce que nous obtenons pour une autre réception. Les tables 15.3 et 15.4 montrent par exemple ce qui se passe si nous recevons un vecteur $\boldsymbol{y}$ correspondant aux vraisemblances normalisées

$$\boldsymbol{a} = (0.2, 0.2, 0.9, 0.2, 0.2, 0.2, 0.2).$$

Dans ce deuxième exemple on voit que le décodage par mots donne le mot 0000000. Cette fois-ci il s'agit du même mot que celui que nous aurions obtenu par le décodage à décisions dures et la métrique de Hamming. Par contre, le décodage par bit donnerait ici le mot 0010000 qui ne figure même pas dans la table de code.

Cependant, on voit, en considérant les probabilités $P(x_i = 1|\boldsymbol{y})$ de la table 15.4 que le calcul (c'est-à-dire la prise en compte des contraintes du code) a conduit à une modification des valeurs de $P(x_i|\boldsymbol{y})$ par rapport aux rapports de vraisemblance a priori. En particulier, on voit que la loi de probabilité se rapproche légèrement de la loi uniforme.

Table 15.4. Vraisemblances et probabilités a posteriori des bits du code de Hamming lorsque qu'on a reçu en sortie un vecteur $\boldsymbol{y}$ dont les composantes ont les rapports de vraisemblance $(\frac{0.2}{0.8}, \frac{0.2}{0.8}, \frac{0.9}{0.1}, \frac{0.2}{0.8}, \frac{0.2}{0.8}, \frac{0.2}{0.8}, \frac{0.2}{0.8})$

i	Vraisemblance normalisée		Probabilités a posteriori	
	$P(y_i\|x_i = 1)$	$P(y_i\|x_i = 0)$	$P(x_i = 1\|\boldsymbol{y})$	$P(x_i = 0\|\boldsymbol{y})$
1	0.2	0.8	0.266	0.734
2	0.2	0.8	0.266	0.734
3	0.9	0.1	0.677	0.323
4	0.2	0.8	0.266	0.734
5	0.2	0.8	0.266	0.734
6	0.2	0.8	0.266	0.734
7	0.2	0.8	0.266	0.734

15.8.3.1 Intérêt du décodage par symboles source. Vu ce qui précède, on peut légitimement s'intéresser de l'intérêt du décodage par bit source.

Le décodage par bits (ou par symboles) que nous venons d'illustrer est aussi appelé décodage à sortie pondérée, parce que ce type de décodage permet de calculer une probabilité (c'est à dire un niveau de confiance) associé à chacun des symboles reçus, et dans le cas de codes systématiques, cela permet directement de caractériser les symboles source.

Bien qu'à la sortie d'un décodeur (c'est-à-dire conditionnellement au message $\boldsymbol{y}$ reçu) les symboles source ne sont en général pas indépendants, on peut néanmoins considérer que la distribution de probabilités conjointe induite à partir des marginales en faisant l'hypothèse d'indépendance entre symboles

$$P(\boldsymbol{s}|\boldsymbol{y}) = \prod_{i=1}^{n} P(s_i|\boldsymbol{y}), \tag{15.45}$$

est une approximation "calculable et utile" en sortie d'un décodeur.

15.8.4 Illustration de l'algorithme de Viterbi

La figure 15.8 représente le treillis du code de Hamming dont les arcs on été décorés par les vraisemblances normalisées relatives aux deux suites reçues, que nous venons de discuter. On suppose que ces deux suites sont transmises l'une après l'autre. Pour des raisons de lisibilité tous les arcs n'ont pas été étiquetés.

Le décodage au maximum de vraisemblance (par mots) revient à trouver dans ce treillis le chemin dont le coût est minimal, ou encore tel que le produit des valeurs associées aux arcs soit maximal. Nous avons représenté sur la partie inférieure de la figure pour le treillis de gauche, les chemins optimaux déterminés par l'algorithme. Les nombres associés à chaque état représentent le produit des vraisemblances normalisées associées aux arcs le long du plus court chemin qui mène vers cet état. L'état intitial est étiqueté avec la valeur 1. Nous n'avons établi que les trois premières étapes de l'agorithme, calculant les chemins les plus vraisemblables vers chaque état en position 3 (il est à noter que pour le code de Hamming, il n'y a qu'un seul chemin possible pour arriver à chacun des noeuds à cette étape). A titre d'exercice le lecteur pourra poursuivre l'algorithme de Viterbi et déterminer ainsi le meilleur chemin de longueur 7 et en déduire le mot de code le plus probable. On voit que l'algorithme détermine en fait un sous-graphe (un arbre) qui représente de façon compacte tous les chemins optimaux. Cela est possible à cause de la propriété de préfixe de ces chemins optimaux, exploitée par l'algorithme de Viterbi.

15.9 CODAGE DANS UN ESPACE EUCLIDIEN

Nous avons vu que lorsque les canaux sont de bonne qualité (rapport signal bruit élevé) leur capacité peut être largement supérieure à un bit par utilisation du canal. Il est alors nécessaire, si on souhaite atteindre des débits de codes proches de la limite de Shannon d'utiliser un alphabet de code de taille suffisamment élevée.

Avec les outils que nous venons d'introduire nous savons en principe construire des codes algébriques et convolutionnels utilisant des alphabets discrets de taille arbitrairement grande (pour autant qu'on prenne une puis-

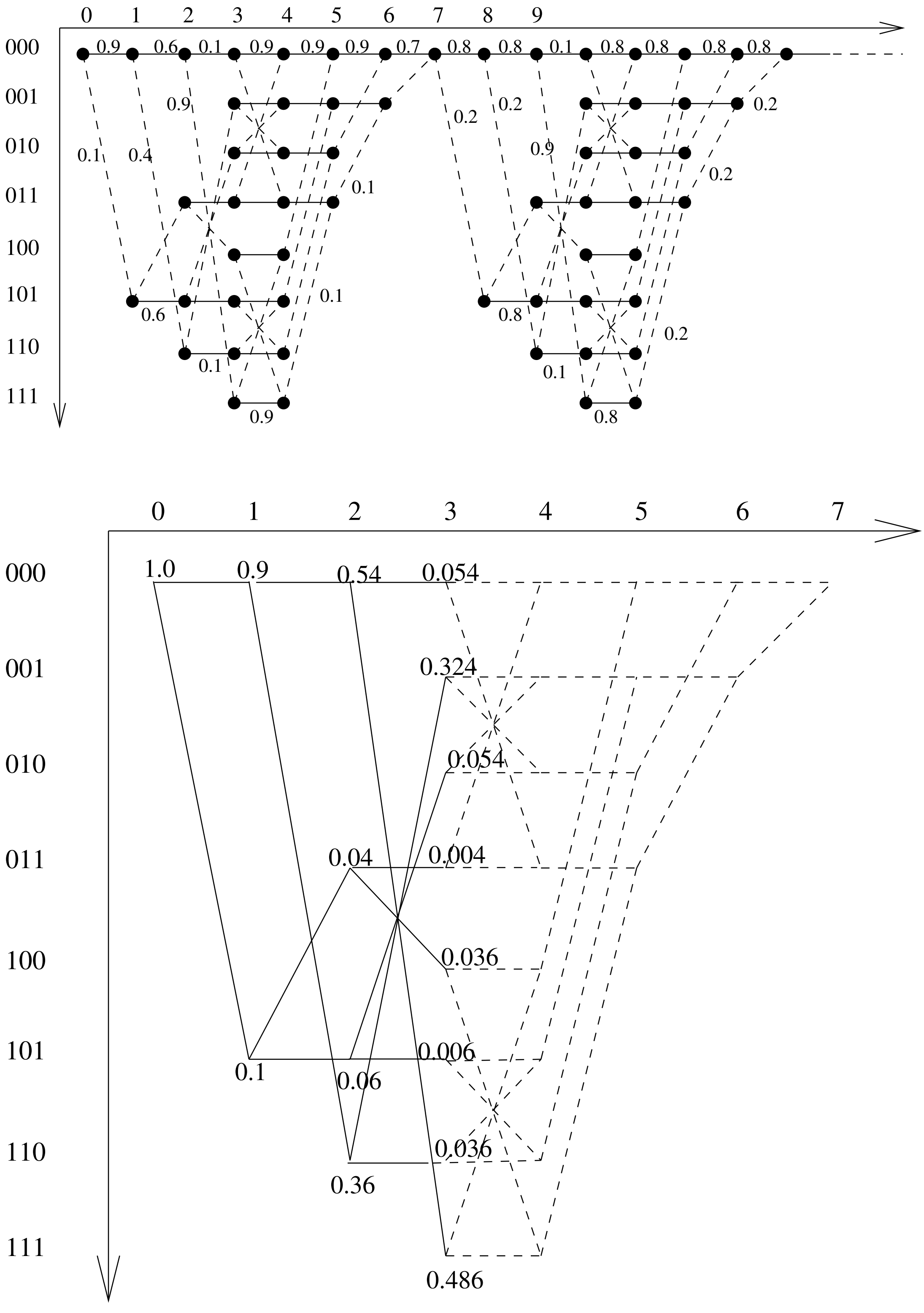


Figure 15.8. Treillis pour le code de Hamming $(7, 4)$

sance entière d'un nombre premier). Cependant, utiliser ce genre d'alphabet nécessite également l'utilisation de systèmes de modulation avec un grand nombre d'états et un des problèmes théoriques principaux de ces systèmes est qu'ils donnent lieu à des modèles de canal non-symétriques. C'est gênant pour le décodage algébrique qui fait une hypothèse forte sur la nature des erreurs de transmission (cf. sens de la métrique de Hamming). On pourrait alors se dire qu'il serait possible de faire appel à des codes convolutionnels plutôt que des codes algébriques, ou du moins d'utiliser le décodage par treillis sur décisions souples pour les codes algébriques. Le décodage par décisions souples n'utilise pas la métrique de Hamming et est donc en principe capable de traiter correctement les canaux du type que nous venons de décrire. Cependant, nous butons ici sur le problème de la complexité du décodage, lié à la croissance rapide du nombre d'états en fonction de la taille de l'alphabet.

Une autre solution, que nous avions suggérée au chapitre 10, consisterait à encoder un mot de source binaire de longueur k à l'aide d'un vecteur de nombres réels (en pratique de longueur $n \ll k$) représentant les coordonnées d'un signal dans un espace de signaux bien choisi (base orthonormée de taille n). Nous savons que le bruit Gaussien se traduira dans cet espace par un vecteur de bruit additif de longueur n dont les composantes sont des variables aléatoires Gaussiennes indépendantes, de moyenne nulle et de variance $\frac{N_0}{2}$. Le décodage peut alors s'opérer dans l'espace Euclidien de dimension n, en cherchant le vecteur de code le plus proche (selon la distance euclidienne) du vecteur bruité reçu.

Evidemment, le problème du décodage de codes aléatoires reste un obstacle. D'autre part, nous ne pouvons plus ici considérer des tables de codes qui seraient des sous-espaces linéaires de l'espace Euclidien, car cela donnerait lieu à un nombre de mots de codes infini. Il nous faut donc trouver un moyen différent de structurer l'ensemble de mots de code de manière à ce qu'il remplisse bien l'espace de signaux tout en facilitant le décodage.

Il y a principalement deux approches qui ont été investiguées dans ce domaine.

15.9.1 Codage par permutations

Cette approche consiste à générer un ensemble de mots de code à partir de toutes les permutations possibles d'un mot unique relativement long. Ces codes sont extrêmement faciles à décoder dans le cadre d'un canal à bruit additif Gaussien. En effet, supposons que le code soit obtenu par permutation du vecteur

$$(\underbrace{a_1, \ldots, a_1}_{n_1}, \underbrace{a_2, \ldots, a_2}_{n_2}, \ldots, \underbrace{a_k, \ldots, a_k}_{n_k}),$$

où les a_i sont ordonnés par ordre croissant. Pour le décodage, il suffit alors de trier les coordonnées reçues par ordre croissant des valeurs y_i et de décider que les n_1 premières correspondent à la valeur a_1, les n_2 suivantes à la valeur a_2 ..., puis de restaurer l'ordre dans lequel les symboles ont été reçus pour trouver le mot de code correspondant. La théorie permet d'étudier la distance Euclidienne minimale entre ce type de mots de code et d'en caractériser la puissance requise en fonction du choix des a_i et des n_i. La principale difficulté est alors liée à l'association entre les messages source et les vecteurs du code.

15.9.2 Codage par réseaux de points

Ici on s'inspire des structures de réseaux cristallins étudiées par les physiciens et aussi les mathématiciens.

L'idée consiste à organiser les mots de code (vecteurs de $\mathbb{R}^n$) sous la forme d'un réseau de points. On peut construire un tel réseau de points en choisissant un certain nombre de vecteurs de base en nombre inférieur à la dimension n de l'espace dans lequel on travaille. Soit $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ une telle base ($k \leq n$) alors le réseau est défini par l'ensemble des combinaisons linéaires à coefficients entiers positifs ou négatifs. Par exemple la figure 15.9 illustre un réseau de points dans $\mathbb{R}^2$. On a également représenté la base utilisée composée des deux vecteurs $(1, 0)$ et $(1/2, \sqrt{3}/2)$ ainsi que les régions du plan correspondant au décodage à distance Euclidienne minimale équivalente (pour un canal Gaussien à bruit additif) au décodage au maximum de vraisemblance.

La région hexagonale qui entoure un mot du code a la propriété du plus proche voisin : si un vecteur tombe dans cette région alors le mot de code associé à la région est le plus proche voisin du vecteur reçu. Cette région contient une sphère de bruit dite de taille maximale qui définit un niveau de bruit en dessous duquel on peut garantir le décodage sans erreur. L'ensemble de ces sphères de bruit ne couvre pas complètement l'espace de code disponible. Il y a aussi, pour une dimension finie de l'espace de codage, un risque de confusion entre symboles.

Nous ne ferons pas d'incursion dans la théorie de ces réseaux. Nous renvoyons le lecteur intéressé à la littérature plus spécialisée (voir par exemple [Bat97] et les références qui y sont fournies). Signalons simplement ici que la construction d'un bon réseau, qui paraît simple à deux dimensions, devient un problème extrêmement

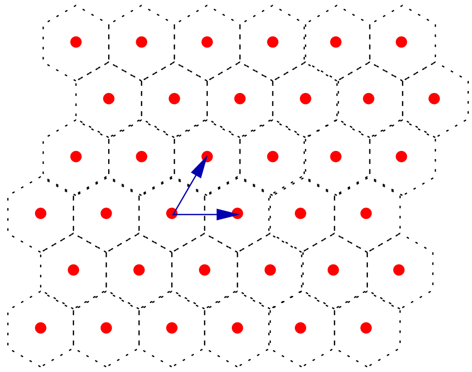


Figure 15.9. Exemple de de réseau de points dans $\mathbb{R}^2$

difficile lorsque le nombre de dimensions devient grand. D'autre part, le décodage de ce type de code reste difficile.

15.10 COMBINAISON DE CODES - POURQUOI ET COMMENT ?

Il y a principalement deux raisons de combiner des codes :

- On dispose d'un système de communication complet (canal physique + codage de canal) qui ne répond pas aux spécifications de l'application pratique considérée : on peut alors souhaiter utiliser un deuxième code pour modifier le compromis fiabilité-débit. Le premier système se présente alors comme un nouveau canal "virtuel", dont on peut en principe déduire les propriétés à partir de celle du canal physique et de son code. Une fois déterminées les caractéristiques, on peut alors appliquer les techniques de ce chapitre à ce canal virtuel.

  Il faut remarquer que si le code utilisé intialement est sous-optimal (selon la figure 15.1 de la page 254 cela veut dire qu'il est placé loin de la frontière de la région inadmissible), ce qui est fort probable, il conduira à une perte de capacité irrévocable par rapport au canal physique sous-jacent. Ceci est à rapprocher du théorème de non création d'information et aussi du second principe de la thermodynamique.

- On espère, en combinant plusieurs codes faciles à encoder/décoder, construire de nouveaux codes ayant certaines propriétés souhaitées, et qui restent faciles à décoder. Nous avons vu un premier exemple de ce type de combinaison dans le cadre de l'entrelacement de codes convolutionnels dans le but d'augmenter la robustesse vis-à-vis d'erreurs en rafales. Nous en verrons, à la section suivante, un autre type dans le contexte des turbo-codes qui ont pour but de s'approcher de la limite Shannon.

Parmi les différentes manières de combiner deux (ou plusieurs) codes on peut citer le produit, la concaténation, et la mise en parallèle. Nous les décrivons brièvement ci-dessous.

15.10.1 Codes produits de codes linéaires

Un code produit de deux codes $\mathcal{C}_1(n_1, k_1)$ et $\mathcal{C}_2(n_2, k_2)$ est un code noté $\mathcal{C}_2 \otimes \mathcal{C}_1(n_1 n_2, k_1 k_2)$ obtenu de la manière suivante :

- on considère un bloc de $k_1 k_2$ symboles source qu'on code selon le code $\mathcal{C}_1$ donnant lieu à k_2 vecteurs de code de longueur n_1;
- on range ces vecteurs dans un tableau $k_2 \times n_1$, une ligne par vecteur;
- on code ensuite les colonnes de ce tableau au moyen du second code, ce qui donne lieu à n_1 blocs de longueur n_2.

Le débit du code produit est donc égal au produit des débits des deux codes. On montre également que si d_1 et d_2 sont les distances minimales des deux codes, la distance minimale du code produit vaut $d_1 d_2$.

Le décodage des codes produits se fait au moyen d'une suite de deux décodeurs qui fonctionnent de manière symétrique à l'encodeur. Cela veut dire qu'on doit d'abord décoder $\mathcal{C}_2$ puis $\mathcal{C}_1$. On peut ici se dire qu'il serait intéressant d'utiliser le décodage à décision pondérée en sortie du premier décodeur de manière à préserver un maximum d'information pour le second décodeur.

L'opération d'entrelacement que nous avons vue auparavant est un cas particulier du produit de deux codes, lorsque le premier code est trivial. Par exemple, si le premier code est le code trivial (l, l) (de débit $R = 1$) et que le second code est de type (n, k) alors le produit des deux codes revient exactement à effectuer l entrelacements du second code. Notons que la matrice génératrice du premier code est alors la matrice identité $\boldsymbol{I}_l$, et si

$$\boldsymbol{G} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & & & \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix}$$

est la matrice génératrice du second, la matrice génératrice du code produit sera la "mega-matrice"

$$\boldsymbol{G} \otimes \boldsymbol{I}_l = \begin{bmatrix} g_{11}\boldsymbol{I}_l & g_{12}\boldsymbol{I}_l & \cdots & g_{1n}\boldsymbol{I}_l \\ g_{21}\boldsymbol{I}_l & g_{22}\boldsymbol{I}_l & \cdots & g_{2n}\boldsymbol{I}_l \\ \vdots & & & \\ g_{k1}\boldsymbol{I}_l & g_{k2}\boldsymbol{I}_l & \cdots & g_{kn}\boldsymbol{I}_l \end{bmatrix}.$$

Cela reste vrai en général, et si on suppose que $\boldsymbol{G}^1$ est la matrice génératrice du premier code, et $\boldsymbol{G}^2$ celle du second on aura la matrice

$$\boldsymbol{G}^2 \otimes \boldsymbol{G}^1 = \begin{bmatrix} g^2_{11}\boldsymbol{G}^1 & g^2_{12}\boldsymbol{G}^1 & \cdots & g^2_{1n}\boldsymbol{G}^1 \\ g^2_{21}\boldsymbol{G}^1 & g^2_{22}\boldsymbol{G}^1 & \cdots & g^2_{2n}\boldsymbol{G}^1 \\ \vdots & & & \\ g^2_{k1}\boldsymbol{G}^1 & g^2_{k2}\boldsymbol{G}^1 & \cdots & g^2_{kn}\boldsymbol{G}^1 \end{bmatrix},$$

comme matrice génératrice du code produit.

15.10.2 Concatenation de codes

L'opération de concaténation est une version plus générale de la précédente. Elle consiste simplement à recoder au moyen du second code le flux de symboles engendré à la sortie du premier.

15.10.3 Mise en parallèle de deux ou plusieurs codes

Cette opération consiste à coder plusieurs fois le message source au moyen de plusieurs codes et à transmettre en parallèle sur le canal (en utilisant le multiplexage) les mots de code résultants. Le décodage du flux de symbole résultant à la sortie du canal devrait alors idéalement se faire en tenant compte du fait que le mot source trouvé doit être le même à la sortie de tous les décodeurs. Un manière "heuristique" de faire cela consiste à utiliser des algorithmes itératifs qui partagent à chaque itération de l'information de manière à satisfaire simultanément les contraintes des différents mots de code mis en parallèle.

Contrairement aux deux techniques précédentes, il est possible d'obtenir de bons codes en mettant en parallèle plusieurs codes relativement mauvais. En effet, comme nous allons le voir dans la section suivante, les turbo-codes peuvent être vus comme résultant de la mise en parallèle de deux codes dont le premier est un code convolutionnel, et le second résulte de la concaténation d'un code de permutation (de débit $R = 1$) et du même code convolutionnel.

15.11 APERÇU SUR LES TURBO-CODES

Dans cette section nous allons décrire le principe général des turbo-codes. Ces codes exploitent les trois principes suivants

1. Codage convolutionnel (récursif systématique)

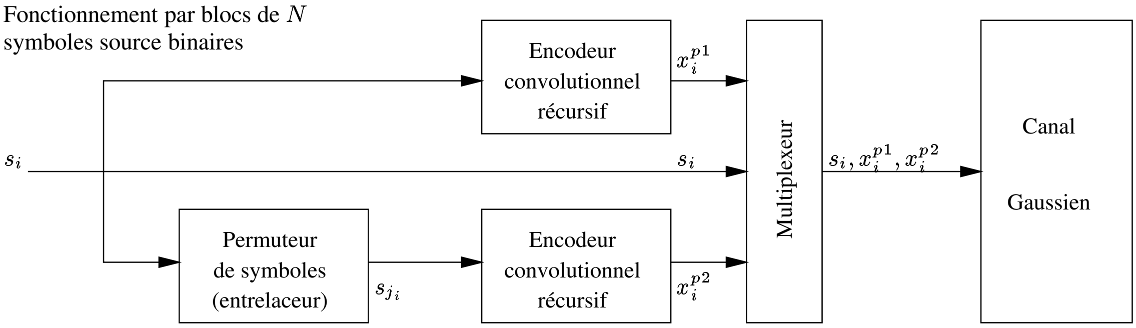


Figure 15.10. Schéma de principe d'un encodeur de turbo-code

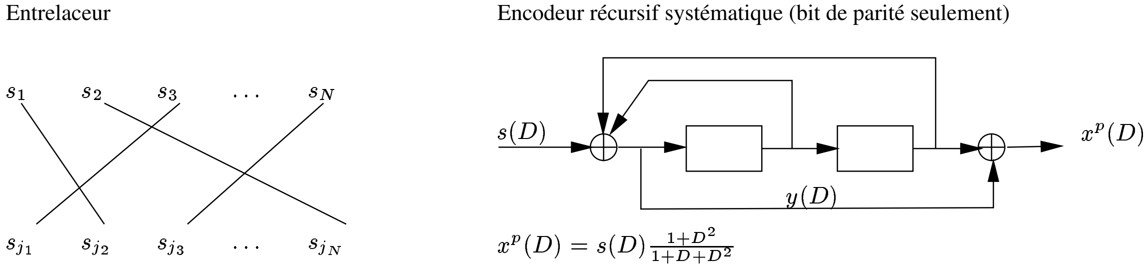


Figure 15.11. Schéma de principe d'un entrelaceur et d'un encodeur récursif

2. Combinaison de codes

3. Décodage itératif en mode bit par bit

Nous commençons par décrire le principe général des encodeurs pour turbo-codes, ensuite nous discuterons de l'algorithme BCJR qui permet d'effectuer le décodage MAP bit par bit de codes convolutionnels systématiques, enfin nous montrerons comment le décodeur des turbo-codes exploite cet algorithme.

15.11.1 Encodeur

La figure 15.10 décrit le principe général d'un encodeur de turbo-code.

L'encodeur consomme un message source composé de N bits (en pratique on a $N \approx 1000$) et génère un mot de code comportant $3N$ bits. Le code est donc de débit $R = 1/3$ puisque pour chaque bit source trois bits sont envoyés sur le canal (nous verrons plus loin comment on peut augmenter le débit d'un turbo-code). Le code est systématique puisque les bits source se retrouvent tels quels dans le mot de code. Les deux autres ensembles de bits sont des bits de parité construits au moyen d'un encodeur convolutionnel récursif (les deux encodeurs récursifs sont identiques). La seule différence (mais elle est de taille) entre les deux messages de parité $\boldsymbol{x}^{p1}$ et $\boldsymbol{x}^{p2}$ est que l'on a permuté l'ordre des bits source pour le second au moyen d'un dispositif appelé permuteur ou entrelaceur. La figure 15.11 montre de façon schématique comment fonctionnent l'entrelaceur et l'encodeur récursif (celui-ci est initialement au repos).

15.11.1.1 Encodeur convolutionnel récursif. La partie droite de la figure 15.11 montre un exemple simple d'encodeur récursif. Cet encodeur est dit récursif car il dispose de boucles de retour qui reinjectent le contenu de la mémoire à l'entrée de l'encodeur. Ce type de dispositif a la particularité de disposer d'une réponse impulsionnelle $h(D)$ de durée infinie. En effet, supposons que la suite à l'entrée $s(D)$ soit une impulsion en $t = 0$, i.e. $s(D) = 1 + 0D + 0D^2 + \ldots + 0D^k + \ldots$; on aura en sortie la suite $h(D)$ qu'on peut déterminer de la manière suivante :

- On a $y(D) = s(D) + Dy(D) + D^2y(D)$ et donc $s(D) = (1 + D + D^2)y(D)$.

- Par ailleurs, on a $x^p(D) = y(D) + D^2y(D) = (1 + D^2)y(D)$.

- En éliminant $y(D)$ on trouve

$$h(D) \triangleq \frac{x^p(D)}{s(D)} = \frac{(1+D^2)}{(1+D+D^2)} = \frac{(1+D)^3}{1+D^3}.$$

- On a

$$h(D) = \frac{(1+D)^3}{1+D^3} = \frac{1+D+D^2+D^3}{1+D^3} = (1+D+D^2+D^3)(1+D^3+D^6+D^9+\ldots+D^{3k}+\ldots)$$

  où nous avons mis en évidence le caractère périodique de $h(D)$ (le second facteur est le développement en série de $(1+D^3)^{-1}$). Notons la similitude de ce type de machine avec un générateur de nombres aléatoires.

On peut déduire de la forme de $h(D)$ que seules les entrées qui sont des multiples de $(1+D+D^2)$ donneront lieu à une réponse de durée finie. Cela est gênant, dans la mesure ou cela signifie qu'un nombre non négligeable de messages source auront un vecteur de parité à poids faible et risquent de ce fait d'être confondus avec le message nul. (Nous savons que le poids minimal des mots de code non-nuls d'un code est égal à la distance minimale de ce code). La seule manière d'éviter ce problème serait d'augmenter le degré du polynôme au dénominateur, mais cela ne peut se faire qu'en augmentant la taille de la mémoire de l'encodeur et nous savons que le prix à payer en termes de complexité de décodage est prohibitif. Cela explique la raison pour laquelle les encodeurs récursifs n'ont pas intéressé beaucoup les chercheurs dans le domaine, jusqu'à la découverte par les auteurs des turbo-codes d'un moyen subtil de restaurer une distance minimale importante et d'un algorithme de décodage efficace correspondant.

15.11.1.2 Entrelaceur. L'entrelaceur effectue une permutation de l'ordre des bits d'entrée. En pratique, cette permutation est choisie de manière aléatoire, afin de bien perturber l'ordre des symboles avant d'alimenter le second encodeur récursif. Le but de l'ensemble est de produire des mots de code qui soient éloignés dans l'espace des signaux, et qui ressemblent à ce que pourrait donner un codage aléatoire.

Dans cette optique, le fait d'utiliser deux fois le même encodeur mais avec des entrées permutées augmente les chances d'avoir des mots de code qui sont éloignés dans l'espace des signaux. En fait on assure une distance importante de tous les mots de code non-nuls par rapport au vecteur nul, parce que si une entrée donne une sortie de poids faible sur un des encodeurs (elle serait alors probablement multiple de $(1 + D + D^2)$), il y a très peu de chances que cette entrée une fois permutée soit encore multiple de $(1 + D + D^2)$ et donc que le deuxième encodeur donne lieu à une sortie de poids faible.

Ceci "justifie" de manière intuitive le choix de l'encodeur des turbo-codes.

15.11.2 Décodage des turbo-codes

Les turbo-codes sont évidemment des codes linéaires. Par conséquent, on pourrait pour leur décodage se servir de la technique des treillis de codes et de l'algorithme de Viterbi. Cependant, une des conséquences néfastes de l'opération d'entrelacement est qu'elle conduit de fait à un treillis de code dont la profondeur peut être très grande, et donc rend caduc le décodage par l'algorithme de Viterbi. Par contre, le décodage partiel qui ne fait appel qu'à la partie systématique et à l'une seule des suites de contrôle de parité (p.ex. s_i, x_i^{p1}) peut se faire de manière efficace (en supposant que le décodeur connaît la règle de permutation) pour autant que la mémoire de l'encodeur récursif ne soit pas trop grande. Toute l'astuce du décodage itératif des turbo-codes consiste alors à décoder le message reçu au moyen d'une succession de décodages simples, qui échangent de l'information à chaque itération.

La figure 15.12 indique le schéma de principe de ce décodeur itératif. On voit que le système est composé de deux décodeurs qui fonctionnent de manière itérative en échangeant les informations L_{ij}^e. Chacun de ces décodeurs calcule une distribution de probabilité a posteriori pour chaque bit source a partir des trois informations suivantes

1. La version bruitée y^s correspondant à la partie systématique de l'encodeur.
2. La version bruitée y^{pi} correspondant à une des deux parties de contrôle de parité de l'encodeur.

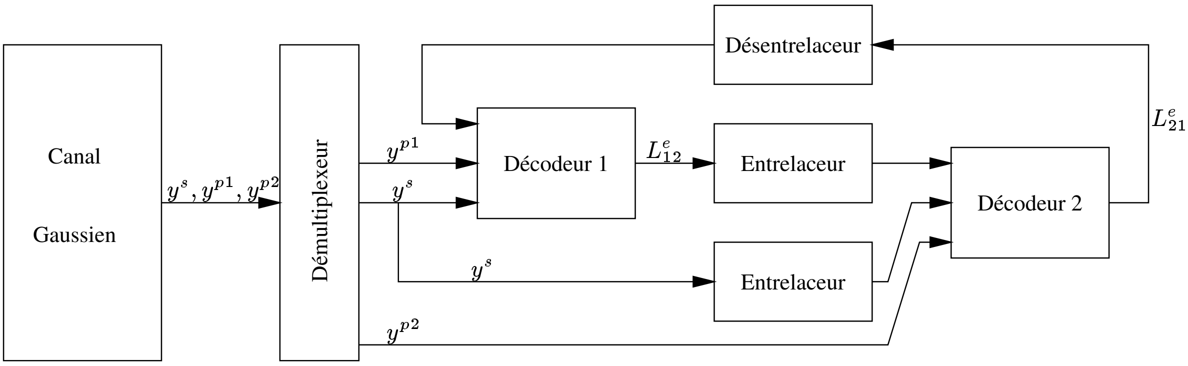


Figure 15.12. Schéma de principe d'un décodeur itératif pour turbo-codes

3. Une information a priori sur les probabilités des bits source (symbolisée par L_{ij}^e sur la figure). A la première itération, cette information correspondra pour le décodeur 1, à l'information a priori sur les bits source $P(s_i)$; ensuite cette information est rafraîchie en fonction du résultat du décodage de l'autre décodeur.

Nous allons détailler ci-dessous la manière dont les deux décodeurs échangent de l'information à chaque cycle, mais avant cela nous devons décrire l'algorithme BCJR qui permet de calculer efficacement les $P(s_i|\boldsymbol{y})$ pour un encodeur systématique.

15.11.2.1 Décodage d'une suite $(s_i, x_i^{p1})_{i=1,\ldots,N}$: algorithme BCJR. L'algorithme BCJR (Bahl, Cocke, Jelinek, Raviv) calcule de manière (relativement) efficace et exacte les probabilités a posteriori des symboles de source (supposés indépendants a priori) étant donné l'observation des sorties du canal Gaussien (sans mémoire) provenant du bruitage d'une suite de symboles d'un code linéaire.

Nous en présentons ici la version spécialisée correspondant au cas où le code est un code convolutionnel systématique de débit $R = 1/2$, c'est-à-dire qui pour chaque symbole source construit deux symboles de code dont le premier est simplement la recopie du symbole source, et le second est obtenu à l'aide d'un dispositif linéaire (récursif ou non). Comme dans le reste de cette section, nous supposons ici que l'alphabet du code est binaire, mais cela ne constitue pas en soi une restriction. (D'ailleurs des variantes de cet algorithme sont également utilisées dans le cadre de la prédiction de séries temporelles générées par des modèles de Markov cachés ayant un nombre quelconque d'états.)

Nous travaillons avec des blocs de symboles source binaires de longueur N et nous disposons donc en sortie du canal de $2N$ nombres réels résultant du processus "modulation-transmission avec bruit-démodulation". Nous supposons, que chaque symbole binaire se traduit par une variable aléatoire Gaussienne en sortie du canal dont la variance est σ^2 et la moyenne ± 1 (en fonction du symbole envoyé 1 ou 0). On peut toujours se ramener à ce type de situation, quitte à renormaliser les sorties du canal si nécessaire. Nous supposerons que la mémoire de l'encodeur vaut m et que donc le nombre d'états de celui-ci vaut 2^m.

Note. Nous attirons l'attention du lecteur sur le fait que nous retournons ici à notre convention habituelle en ce qui concerne la numérotation des symboles, c'est-à-dire en commençant avec l'indice 1.

Désignons par $\boldsymbol{y}$ le vecteur $(y_1^s, y_1^p, \ldots, y_k^s, y_k^p, \ldots, y_N^s, y_N^p)$ reçu en sortie du canal, et soit

$$L(s_k) \triangleq \log \frac{P(s_k = 1|\boldsymbol{y})}{P(s_k = 0|\boldsymbol{y})} \tag{15.46}$$

le rapport de vraisemblance relatif au k-ième symbole source à la sortie du canal. Bien entendu, la connaissance de ce rapport est équivalente à la connaissance de $P(s_k = 1|\boldsymbol{y})$ et $P(s_k = 0|\boldsymbol{y})$ et permet donc notamment de décoder par symbole. Mais ici, nous nous intéressons explicement au calcul de ce rapport de vraisemblance.

Pour rendre l'écriture de certaines formules symétriques nous allons utiliser parfois la variable u_k équivalente à s_k mais dont les deux valeurs sont symétriques par rapport à l'origine. On a $u_k = (-1)^{1+s_k}$. On a donc

$$L(u_k) \triangleq \log \frac{P(u_k = 1|\boldsymbol{y})}{P(u_k = -1|\boldsymbol{y})} = L(s_k) \tag{15.47}$$

et aussi

$$L_e(s_k) = L_e(u_k) = \log \frac{P(u_k = 1)}{P(u_k = -1)}. \tag{15.48}$$

On déduit de ces relations que

$$P(s_k) = P(u_k) = A_k \exp\left[u_k \frac{1}{2} L_e(u_k)\right], \tag{15.49}$$

où A_k est une constante de normalisation.

On a évidemment aussi

$$L(s_k) = \log \frac{P(s_k = 1|\boldsymbol{y})p(\boldsymbol{y})}{P(s_k = 0|\boldsymbol{y})p(\boldsymbol{y})}. \tag{15.50}$$

Par exemple, $P(s_k = 1|\boldsymbol{y})p(\boldsymbol{y}) = P(s_k = 1, \boldsymbol{y})$ pourrait être calculé en effectuant la somme sur tous les mots de code possibles sous la contrainte $s_k = 1$. Mais, en termes de treillis de code, cela revient aussi à calculer la somme des probabilités des transitions $t_k = (e_{k-1}, e_k)$ possibles à l'étape $k-1$, sous la contrainte que cette transition corresponde à un bit d'information $s_k = 1$. Désignons par S_1 l'ensemble des transitions de notre encodeur qui correspondent à un symbole d'information valant 1 et S_0 l'ensemble des transitions qui correspondent à un symbole d'information valant 0. Si on suppose que l'encodeur commence dans un état quelconque (p.ex. choisi de manière aléatoire), ces ensembles ne dépendent pas de l'instant k. Evidemment, on peut toujours a posteriori tenir compte du fait qu'on connaît a priori l'état initial (et éventuellement final, voir remarque sur la terminaison).

On peut donc écrire $L(s_k)$ sous la forme

$$L(s_k) = \log\left(\frac{\sum_{t\in S_1} P(t_k = t, \boldsymbol{y})}{\sum_{t\in S_0} P(t_k = t, \boldsymbol{y})}\right) = \log\left(\frac{\sum_{(e,\overline{e})\in S_1} P(e_{k-1} = e, e_k = \overline{e}, \boldsymbol{y})}{\sum_{(e,\overline{e})\in S_0} P(e_{k-1} = e, e_k = \overline{e}, \boldsymbol{y})}\right). \tag{15.51}$$

Montrons comment un calcul récursif permet de calculer les termes du type $P(e_{k-1} = e, e_k = \overline{e}, \boldsymbol{y})$ relatifs à toutes les transitions du treillis du code.

On commence par séparer $P(e_{k-1}, e_k, \boldsymbol{y})$ (en simplifiant la notation) en trois facteurs de la manière suivante (les coordonnées de $\boldsymbol{y}$ sont comptées par paires pour alléger la notation)

$$\begin{aligned}
P(e_{k-1}, e_k, \boldsymbol{y}) &= P(e_{k-1}, e_k, \boldsymbol{y}_1^{k-1}, \boldsymbol{y}_k^k, \boldsymbol{y}_{k+1}^N) & (15.52)\\
&= P(e_{k-1}, \boldsymbol{y}_1^{k-1}) P(e_k, \boldsymbol{y}_k^k | e_{k-1}, \boldsymbol{y}_1^{k-1}) P(\boldsymbol{y}_{k+1}^N | e_{k-1}, e_k, \boldsymbol{y}_1^{k-1}, \boldsymbol{y}_k) & (15.53)\\
&= P(e_{k-1}, \boldsymbol{y}_1^{k-1}) P(e_k, \boldsymbol{y}_k^k | e_{k-1}) P(\boldsymbol{y}_{k+1}^N | e_k), & (15.54)
\end{aligned}$$

où les simplifications sont justifiées par le fait que le canal et la source sont ici supposées sans mémoire ($\boldsymbol{y}_i^j$ désigne les sorties du canal correspondant aux instants $2i-1, 2i, \ldots, 2j-1, 2j$).

On peut donc écrire aussi

$$P(e_{k-1}, e_k, \boldsymbol{y}) = \alpha_{k-1}(e_{k-1}) \gamma_k(e_{k-1}, e_k) \beta_k(e_k). \tag{15.55}$$

où on ne fait intervenir dans les trois facteurs que les parties qui sont effectivement variables (la sortie du canal est connue au moment du décodage). On peut interpéter les trois termes de la manière suivante

- $\gamma_k(e_{k-1}, e_k) = P(e_k, \boldsymbol{y}_k^k | e_{k-1})$ est une donnée purement locale relative à une transition : elle fait intervenir le choix du symbole source s_k (équivalent au choix de la transition lorsque l'origine de celle-ci e_{k-1} est donnée) et le modèle du canal. On peut en effet écrire cette probabilité sous la forme

  $$\gamma_k(e_{k-1}, e_k) = P(s_k, x_k^p, y_k^s, y_k^p | e_{k-1}) = P(s_k, x_k^p | e_{k-1}) P(y_k^s | s_k) P(y_k^p | x_k^p) = P(s_k) P(y_k^s | s_k) P(y_k^p | x_k^p).$$

  Dans le cas du modèle de canal Gaussien, et en fonction du rapport de vraisemblance a priori sur les symboles source on obtient finalement

  $$\gamma_k(e_{k-1}, e_k) = A_k B_k \exp\left[\frac{u_k L_e(u_k)}{2}\right] \exp\left[\frac{u_k y_k^s}{\sigma^2}\right] \exp\left[\frac{v_k y_k^p}{\sigma^2}\right], \tag{15.56}$$

où u_k et de v_k désignent les versions “antipodales” des symboles de source et de parité, et où A_k et B_k sont des constantes de normalisation indépendantes des valeurs de u_k et de v_k, c'est-à-dire de la transition pour laquelle on calcule γ_k.

Les valeurs de γ_k caractérisent l'ensemble des transitions du treillis du code et peuvent être déterminées au fur et à mesure de la réception des sorties. Ce sont ces valeurs qui déterminent la qualité des arcs du treillis de code, et qui seraient utilisées par l'algorithme de Viterbi pour trouver le chemin le plus probable (ici ce chemin correspondrait à la qualité maximale, vue la définition des γ_k).

Notons qu'il n'est pas nécessaire en pratique de déterminer les constantes de normalisation car celles-ci ne sont pas dépendantes des transitions (elles sont cependant dépendantes de la valeur de k mais ce n'est pas grave).

- $\alpha_k(e_k) = P(e_k, \boldsymbol{y}_1^k)$ qui est relative aux chemins à gauche. Il s'agit de la probabilité conjointe d'aboutir à l'état e_k et d'observer les sorties $\boldsymbol{y}_1^k$.

  Cette grandeur caractérise donc les états du réseau en fonction des observations passées et présentes. L'astuce consiste ici à la calculer de manière récursive, en partant de la donnée relative aux états initiaux $P(e_0)$ (en pratique on suppose que $\alpha_0(e_0) = P(e_0 = 0) = 1$ et $P(e_0 \neq 0) = 0$ mais cela n'est pas fondamental pour notre récursion).

  En effet, on a ($\forall e_k \in S, k \geq 1$)

$$\begin{aligned}
\alpha_k(e_k) = P(e_k, \boldsymbol{y}_1^k) &= \sum_{e_{k-1} \in S} P(e_{k-1}, e_k, \boldsymbol{y}_1^k) && (15.57)\\
&= \sum_{e_{k-1} \in S} P(e_{k-1}, e_k, \boldsymbol{y}_k^k, \boldsymbol{y}_1^{k-1}) && (15.58)\\
&= \sum_{e_{k-1} \in S} P(e_{k-1}, \boldsymbol{y}_1^{k-1}) P(e_k, \boldsymbol{y}_k^k | e_{k-1}, \boldsymbol{y}_1^{k-1}) && (15.59)\\
&= \sum_{e_{k-1} \in S} P(e_{k-1}, \boldsymbol{y}_1^{k-1}) P(e_k, \boldsymbol{y}_k^k | e_{k-1}) && (15.60)\\
&= \sum_{e_{k-1} \in S} \alpha_{k-1}(e_{k-1}) \gamma_k(e_{k-1}, e_k). && (15.61)
\end{aligned}$$

  Evidemment, dans cette formule seuls les états parents de e_k interviennent dans la sommation. Le calcul de tous les α_k (pour tous les états e_k) se fait de gauche à droite en propageant les valeurs déjà calculées le long des transitions, en multipliant par la probabilité de ces transitions, et en cumulant les valeurs aux extrémités de celle-ci. Au total, le nombre de calculs est exactement égal au nombre de transitions possibles à l'instant $k-1$, qui vaut exactement $2 \times |S|$ (puisqu'il y a exactement deux arcs sortants par noeud du treillis de code).

  On constate que le nombre d'opérations est proportionnel au produit du nombre d'états (S) de l'encodeur et à la longueur N de la séquence qui est décodée.

- $\beta_k(e_k) = P(\boldsymbol{y}_{k+1}^N | e_k)$ est un terme relatif aux états qui fait intervenir les observations futures.

  Un raisonnement analogue au précédent permet de calculer les β_k en partant de la droite. On obtient la relation de “rétro-propagation” suivante

$$\beta_{k-1}(e_{k-1}) = \sum_{e_k \in S} \gamma_k(e_{k-1}, e_k) \beta_k(e_k). \tag{15.62}$$

  Cette relation est initialisée avec $\beta_N(e_N) = 1$ pour chacun des états terminaux possibles du code, et $\beta_N(e_N) = 0$ pour les autres états. Si on s'arrange pour que tous les mots de code se terminent dans l'état nul, seul reste le terme relatif à cet état, ce qui est préférable pour des raisons que nous n'expliciterons pas ici. On peut s'arranger pour que cela soit le cas en terminant correctement les mots de code, c'est-à-dire en les finissant avec un suffixe qui assure le retour de l'encodeur dans l'état 0. Ce processus est appelé *terminaison du code*.

Remarque. Les formules de récurrence que nous venons de présenter peuvent être sensibles aux erreurs d'arrondi (grand nombre de mutiplications/additions par des nombres petits), et en pratique on utilise des versions normalisées de ces formules. Nous renvoyons le lecteur intéressé à la littérature spécialisée pour plus de détails sur les aspects d'implémentation.

15.11.2.2 Décodage itératif pour turbo-codes. Analysons la formule de décodage itératif pour un code systématique que nous venons de voir :

$$L(s_k) = L^k = \log\left(\frac{\sum_{t\in S_1} P(t_k = t, \boldsymbol{y})}{\sum_{t\in S_0} P(t_k = t, \boldsymbol{y})}\right) \quad (15.63)$$

$$= \log\left(\frac{\sum_{(e_{k-1},e_k)\in S_1} \alpha_{k-1}(e_{k-1})\gamma_k(e_{k-1},e_k)\beta_k(e_k)}{\sum_{(e_{k-1},e_k)\in S_0} \alpha_{k-1}(e_{k-1})\gamma_k(e_{k-1},e_k)\beta_k(e_k)}\right). \quad (15.64)$$

L'idée est de décomposer le terme relatif à $\gamma_k(e_{k-1}, e_k)$ en ses trois contributions selon la formule (15.56). En utilisant la notation $\gamma_k^e(e_{k-1}, e_k) = \exp\left[\frac{v_k y_k^p}{\sigma^2}\right]$ et en simplifiant les facteurs $A_k B_k$, on obtient pour chacun des décodeurs ($i = 1, 2$) et en désignant par j $(= 2, 1)$ l'autre décodeur

$$L_i^k = \log\left(\frac{\sum_{(e_{k-1},e_k)\in S_1} \alpha_{k-1}(e_{k-1})\gamma_k^e(e_{k-1},e_k)\beta_k(e_k)\exp\left[\frac{u_k L_e(u_k)}{2}\right]\exp\left[\frac{u_k y_k^s}{\sigma^2}\right]}{\sum_{(e_{k-1},e_k)\in S_0} \alpha_{k-1}(e_{k-1})\gamma_k^e(e_{k-1},e_k)\beta_k(e_k)\exp\left[\frac{u_k L_e(u_k)}{2}\right]\exp\left[\frac{u_k y_k^s}{\sigma^2}\right]}\right) \quad (15.65)$$

$$= \log\left(\frac{\sum_{(e_{k-1},e_k)\in S_1} \alpha_{k-1}(e_{k-1})\gamma_k^e(e_{k-1},e_k)\beta_k(e_k)\exp\left[\frac{+1 L_e(u_k)}{2}\right]\exp\left[\frac{+1 y_k^s}{\sigma^2}\right]}{\sum_{(e_{k-1},e_k)\in S_0} \alpha_{k-1}(e_{k-1})\gamma_k^e(e_{k-1},e_k)\beta_k(e_k)\exp\left[\frac{-1 L_e(u_k)}{2}\right]\exp\left[\frac{-1 y_k^s}{\sigma^2}\right]}\right) \quad (15.66)$$

$$= \log\left(\frac{\sum_{(e_{k-1},e_k)\in S_1} \alpha_{k-1}(e_{k-1})\gamma_k^e(e_{k-1},e_k)\beta_k(e_k)}{\sum_{(e_{k-1},e_k)\in S_0} \alpha_{k-1}(e_{k-1})\gamma_k^e(e_{k-1},e_k)\beta_k(e_k)}\right) + L_e^k + \frac{2y_k^s}{\sigma^2} \quad (15.67)$$

où nous avons décomposé la vraisemblance en trois termes

- $\log\left(\frac{\cdots}{\cdots}\right) \triangleq L_{i,j}^k$ qui désigne l'information apportée par les bits de parité relatifs au décodeur i; cette information sera fournie à l'autre décodeur comme information a priori.
- $L_e^k \triangleq L_{j,i}^k$ constitue l'information a priori sur le symbole source u_k; à la première itération elle est constituée par le modèle de la source; aux itérations suivantes elle est obtenue de la part de l'autre décodeur (d'où la notation).
- $\frac{2y_k^s}{\sigma^2} \triangleq L_{s,i}^k(y_k^s)$ constitue l'information apportée par les sorties du canal relatifs aux bits systématiques; elle est la même pour les deux décodeurs et ne change pas d'une itération à la suivante.

Le principe du décodeur itératif est alors le suivant (voir figure 15.12) :

1. Les deux décodeurs opèrent de façon itérative.
2. A chaque étape, un des décodeurs (disons i) reçoit en entrée l'information a priori sur les symboles source sous la forme de rapports de vraisemblance partiels $L_{j,i}^k$ calculés par le décodeur précédent (ou à partir du modèle de source, s'il n'y a pas encore d'information disponible de ce type). Ensuite il procède de la manière suivante :
   (a) Il décore les arcs du treillis au moyen des γ_k en utilisant les trois termes relatifs à (i) l'information a priori, (ii) les sorties du canal relatives aux bits systématiques, (iii) les sorties relatives aux aux bits de contrôle de parité. (Comme ces deux derniers ne changent pas, il suffit en fait de tenir compte de la modification des informations a priori).
   (b) Il calcule les α_k et les β_k à partir des γ_k et ensuite les termes $L_{i,j}^e(u_k)$ pour chaque bit source. Il fournit ce terme en entrée du décodeur suivant.

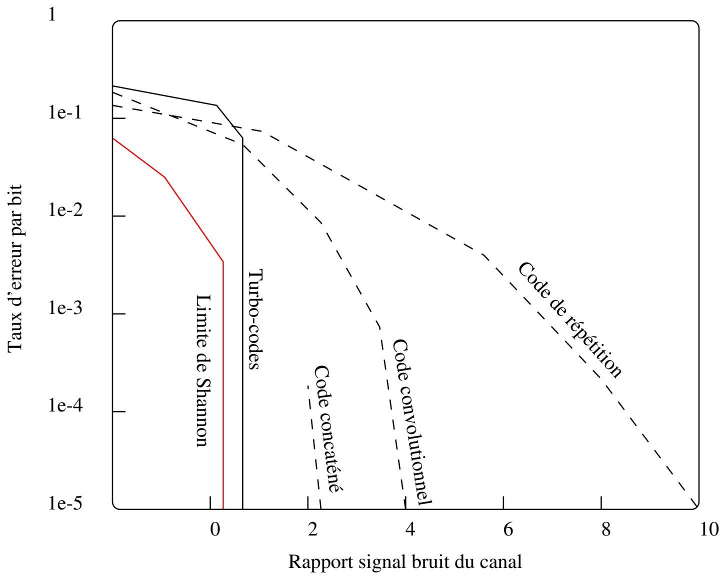


Figure 15.13. Performances relatives de différents codes de canal

3. Le processus s'arrête après un nombre (par exemple fixé a priori) d'itérations et c'est le dernier décodeur qui calcule les valeurs de $L_i(u_k)$ et décide pour chaque symbole source s'il s'agit d'un 1 ou d'un zéro en fonction du signe de cette grandeur (positif ou négatif).

15.11.3 Discussion

Les turbo-codes combinent de manière astucieuse un certain nombre de techniques connues avant leur invention. Les choix ont été faits de manière pragmatique et comme cette découverte est récente on n'en cerne pas encore à l'heure actuelle tous les tenants et aboutissants. Mais les performances des turbo-codes sont nettement meilleures que celles de la plupart des méthodes de codage connues à l'heure actuelle, en particulier les méthodes de codage algébriques.

Dans ce qui précède nous avons surtout cherché à mettre en évidence les grands principes : pourquoi un turbo-code est a priori susceptible d'avoir de bonnes performances; comment leur algorithme de décodage peut fonctionner de manière efficace sans gaspiller de l'information.

Nous n'avons décrit que des turbo-codes de débit $R = 1/3$. On peut augmenter le débit des turbo-codes en ne transmettant qu'une partie des symboles de parité (p.ex. pour obtenir un débit $R = 1/2$, un symbole de parité sur deux, pour chaque encodeur); le décodeur est alors amené à deviner les valeurs correspondantes à la sortie du canal, ce qui consisterait dans le cas du canal Gaussien utilisé en modulation antipodale à utiliser la valeur $y = 0$.

15.12 SYNTHÈSE

La figure 15.13 illustre graphiquement les performances de différents codes entrevus dans ce chapitre, tous ayant un débit $R = 1/2$, dans le cas du canal Gaussien utilisé avec un alphabet binaire en entrée. L'axe horizontal représente le rapport signal/bruit dans le canal, et l'axe vertical les performance en termes de taux d'erreurs par bit source. La courbe la plus à gauche représente la limite de Shannon, qui indique qu'avec un code de débit $1/2$ il est possible de fonctionner à taux d'erreurs arbitrairement faible pour autant que le rapport signal bruit soit supérieur à 0.2 dB. On voit que les turbo-codes approchent cette limite avec une sous-optimalité qui en terme de rapport signal bruit vaut 0.5dB. On y voit clairement la supériorité des turbo-codes.

Il a été estimé à la fin des années 1960 que chaque dB gagné dans le contexte de communications spatiales, réduisait le budget de lancement d'un montant d'un million de dollars (réduction du poids des engins spatiaux). Aujourd'hui, pour les missions lointaines l'économie représenterait de l'ordre de 80 millions de dollars par dB. Certaines missions spatiales ont échoué parce que la liaison avec la terre ont été perturbées par des niveaux de bruit très élevés. Par exemple, l'échec de la mission Gallileo a couté de l'ordre d'un milliard de dollars au total. Peut-être que l'utilisation de codes plus puissants (turbo-codes ou autres) aurait permis de l'éviter. On peut donc dire que les gains potentiels apportés par les techniques de codage de canal efficaces sont loin d'être négligeables.

15.13 RESUME

Nous n'avons malheureusement pu qu'effleurer très superficiellement le domaine passionnant que constitue le codage de canal et ses ramifications diverses.

Cependant nous espérons avoir pu mettre en évidence certaines des raisons qui font que ce domaine reste à l'heure actuelle un domaine de recherche très actif.

En effet, les progrès possibles dans le domaine du codage de canal offrent la possibilité de réduire sensiblement les coûts de transmission et/ou de stockage de l'information, ou bien d'en augmenter la vitesse et/ou la densité. Nous savons que les progrès récents et futurs de l'informatique et des télécommunications reposent notamment sur les progrès dans ces domaines. Une part peut être achevée par une approche physique consistant à utiliser de nouveaux matériaux par exemple, une autre - non moins importante - par une approche système reposant sur la théorie de l'information, visant à utiliser le plus intelligemment possible les dispositifs physiques existants et à venir.

Même si notre présentation du codage de canal a été très limitée, et sans doute un peu opaque, nous espérons avoir mis en évidence que la solution de ce problème passe par une approche pluridisciplinaire faisant appel à la modélisation physique des canaux, la théorie de l'information pour l'exploitation théorique de ces modèles, à l'algorithmique pour la mise en oeuvre des solutions efficaces, et au bon sens de l'ingénieur pour la combinaison fructueuse de ces divers outils.

Appendice 15.A: Compléments sur les polynômes

Cet appendice fournit quelques précisions sur les polynômes définis sur un corps fini.

Désignons par K le corps fini dans lequel nous travaillons.

Définition des polynômes sur K

Un polynôme sur K est une expression (formelle) du type

$$p(D) = p_0 + p_1 D + \cdots + p_m D^m,$$

où on peut omettre les termes dont le coefficient est nul. Le degré du polynôme est par définition la puissance associée au coefficient non-nul de degré le plus élevé (ici m si $p_m \neq 0$). Deux polynômes sont identiques si leurs coefficients de même degré sont tous égaux deux à deux. Le polynôme nul est le polynôme dont tous les coefficients sont nuls. On le note 0. Par extension, on dit que le degré de ce polynôme vaut -1.

Un polynôme définit aussi une fonction sur K, puisqu'on peut faire varier $D \in K$ et calculer la valeur $p(D)$ selon l'arithmétique de K. Cependant, nous ne pouvons pas identifier les polynômes avec des fonctions car deux polynômes peuvent être différents et néanmoins définir une même fonction. Par exemple si $K = Z_2$, les polynômes

$$D + 1$$

et

$$D^3 + D^2 + D + 1$$

sont différents mais définissent cependant la même fonction. En fait, il n'y a que 4 fonctions différentes qu'on peut définir sur Z_2 alors qu'on peut y définir une infinité de polynômes différents.

Un polynôme est dit *monique*, si son coefficient de degré le plus élevé vaut 1.

Opérations sur les polynômes

On définit l'addition et la multiplication de polynômes de la manière usuelle (en utilisant l'arithmétique de K pour les calculs). L'ensemble des polynômes muni de ces deux opérations forme un anneau (inexistence des inverses).

Par exemple, l'addition de deux polynômes $p(D)$ et $q(D)$ consiste en un polynôme $r(D)$ dont les coefficient sont obtenus en sommant les coefficients de même degré dans K. Par exemple, dans Z_2 on a

$$(D + 1) + (D^3 + D^2 + D + 1) = (D^3 + D^2).$$

Si on se donne deux polynômes quelconques non nuls $p(D)$ et $p'(D)$ alors on peut toujours écrire

$$p(D) = p'(D)q(D) + r(D)$$

où $q(D)$ désigne le quotient de p par p' et $r(D)$ le reste (l'opération est analogue à la division entière dite "longue").

Si $m < m'$ alors $q(D) = 0$ et $r(D) = p(D)$. Sinon, le degré de $q(D)$ est $m - m'$ et le degré de $r(D)$ est au plus $m' - 1$. Si $r(D)$ est nul on dit que $p'(D)$ est un facteur de $p(D)$, ou que $p'(D)$ factorise $p(D)$, ou encore que $p(D)$ est divisible par $p'(D)$.

Zéros d'un polynôme. Un élément $a \in K$ est un zéro du polynôme $p(D)$ si $p(a) = 0$, c'est-à-dire si la fonction associée au polynôme s'annule en ce point.

On montre que $a \in K$ est un zéro du polynôme $p(D)$ si et seulement si $(D - a)$ factorise $p(D)$.Plus généralement, si $a_1, a_2, \ldots, a_n$ sont des zéros distincts de $p(D)$ alors on a $p(D) = (D - a_1)(D - a_2) \cdots (D - a_n)q(D)$. En d'autre mots $p(D)$ est divisible par le polynôme $(D - a_1)(D - a_2) \cdots (D - a_n)$.

Par exemple, dans Z_2 le polynôme D^3+1 s'annule en $D = 1$ et est donc divisible par $D-1$ (ici $D-1 = D+1$). On a

$$D^3 + 1 = (D + 1)(D^2 + D + 1).$$

Par contre, toujours dans Z_2, le polynôme
$$D^2 + D + 1$$
n'a pas de zéro.

Le polynôme $D^3 + 1$ se factorise dans Z_3 en $(D + 1)^3$. Donc, la factorisation de polynômes dépend du corps K avec lequel on travaille.

Polynômes irréductibles et factorisation

Un polynôme $p(D)$ défini sur K est dit irréductible s'il ne peut pas être factorisé sous la forme d'un produit de deux polynômes de degré strictement inférieur à celui de $p(D)$.

Par exemple, dans Z_2, $D^2 + D + 1$ est irréductible. Tous les polynômes de degré 0 ou 1 sont irréductibles. Un polynôme de degré 2 ou 3 est irréductible si et seulement si il n'a pas de zéros. Un polynôme de degré supérieur à 3 peut ne pas avoir de zéros tout en étant réductible (p.ex. on peut prendre le produit de deux polynômes irréductibles de degré 2 comme contre-exemple).

La propriété d'irréductibilité est similaire à la notion de nombre premier : un nombre premier est un nombre entier qui ne peut pas être factorisé sous la forme de deux nombres entiers strictement plus petits.

Pour les nombres premiers on a le théorème de factorisation qui dit que tout nombre entier peut être factorisé de manière unique en un produit de nombres premiers. Pour les polynômes on a : *tout polynôme peut être factorisé de façon unique sous la forme*
$$p(D) = ap_1(D)p_2(D)\cdots p_s(D)$$
où les $p_i(D)$ sont des polynômes irréductibles moniques.

L'analogie va en fait beaucoup plus loin. Nous avons vu que l'ensemble des nombre entiers positifs qui ne constitue pas un corps (il s'agit cependant d'un anneau), pouvait être transformé en corps, à condition d'y définir l'égalité comme une égalité "modulo" un nombre premier p (factorisation de Z par le sous-groupe des multiples de p.)

Nous verrons à l'appendice 15.B qu'on peut réduire l'ensemble des polynômes définis sur K en définissant l'égalité (et donc aussi les opérations d'addition et de multiplication) modulo $p(D)$, pour autant que $p(D)$ soit un polynôme irréductible.

Appendice 15.B: Corps de Gallois

Dans cette appendice nous allons élucider la procédure de construction de corps par extension d'un corps de base. Cette opération permet de construire tous les corps finis qui existent à partir des seuls corps de type "modulo p" ou p est un nombre entier premier. Nous verrons qu'on obtient ainsi une famille de corps dits d'extension de taille p^m où m est un nombre entier quelconque.

Extension algébrique d'un corps

L'opération d'extension permet de construire de nouveaux corps plus grands à partir d'un corps pré-existant. Nous allons d'abord illustrer la procédure en montrant comment on peut ainsi construire le corps des nombres complexes à partir du corps des nombres réels. Ensuite nous énoncerons quelques résultats généraux relatifs aux corps de Gallois, utiles dans le domaine de la théorie des codes. Nous renvoyons le lecteur à la référence [Adá91] pour un traitement plus complet et plus approfondi de ces questions.

Soit $p(D)$ un polynôme de degré $m \geq 1$ défini sur K. Alors, on entend par extension algébrique de K l'ensemble des polynômes sur K de degré inférieur à m en une variable α munis des opérations d'addition et de multiplication définies comme suit :

Addition. On a
$$f(\alpha) + g(\alpha) = h(\alpha) \Leftrightarrow f(D) + g(D) = h(D),$$
autrement dit l'addition correspond à l'addition classique de polynômes sur K.

Multiplication. On a
$$f(\alpha)g(\alpha) = h(\alpha) \Leftrightarrow f(D)g(D) = p(D)q(D) + h(D),$$

autrement dit $h(D)$ est en fait le reste de la division de $f(D)g(D)$ par $p(D)$ dans K.

On pourra se convaincre que l'extension produit une structure d'anneau. Cette structure est finie si K l'est et comporte alors exactement $|K|^m$ éléments distincts.

D'autre part si $p(D)$ n'est pas irréductible alors cette structure n'est pas un corps. En effet, dans ce cas il existera au moins deux polynômes non-nuls de degré inférieur à m, $f(D)$ et $g(D)$ tels que $p(D) = f(D)g(D)$. Cela implique alors que $f(\alpha)g(\alpha) = 0$, ce qui est impossible dans un corps.

Réciproquement, si $p(D)$ est un polynôme irréductible alors l'extension de K par $p(D)$ est un corps.

Le corps d'extension des nombres réels. Dans $\mathbb{R}$, le polynôme $x^2 + 1$ est irréductible. On en déduit que l'extension de $\mathbb{R}$ par $x^2 + 1$ est un corps.

Les éléments de ce corps sont donc les polynômes de degré 1, du type $a + \alpha b$, où $a, b \in \mathbb{R}$. L'addition de deux pareils polynômes donne

$$(a + \alpha b) + (a' + \alpha b') = (a + a') + \alpha(b + b'),$$

et leur multiplication

$$(a + \alpha b)(a' + \alpha b') = [aa' + (ab' + a'b)\alpha + bb'\alpha^2]\text{mod}(\alpha^2 + 1) = (aa' - bb') + (ab' + a'b)\alpha.$$

Ces règles correspondent exactement aux règles d'addition et de multiplication des nombres complexes. D'autre part, en prenant le cas particulier $a = a' = 0$, $b = b' = 1$ la règle de multiplication donne

$$\alpha^2 = -1.$$

Tout se passe comme si on avait créé un nouveau nombre imaginaire α qui est solution de $x^2 + 1 = 0$, et qu'on avait étendu les opérations d'addition et de multiplication classiques en tenant compte de la contrainte que $\alpha^2 + 1 = 0$.

Corps finis et corps de Gallois. L'extension algébrique d'un corps de type Z_p (p étant un nombre premier) par un polynôme irréductible de degré m est appelé corps de Gallois, et désigné par $CG(p^m)$.

On montre que tous ces corps existent, et on montre aussi qu'il n'existe essentiellement pas d'autres corps finis que les corps de Gallois.

Exemple $CG(4)$. Il s'agit de l'extension de Z_2 par le polynôme irréductible $D^2 + D + 1$.

Les quatre éléments de ce corps sont les polynômes 0, 1, α, $1 + \alpha$ (on va appeler cet élément β).

On a $\alpha\beta = [\alpha^2 + \alpha]\text{mod}\alpha^2 + \alpha + 1 = 1$. On a également $\alpha^2 = \beta$...

La figure 15.B.1 reprend les règles d'addition et de multiplication dans $CG(4)$.

$+$	0	1	α	β
0	0	1	α	β
1	1	0	β	α
α	α	β	0	1
β	β	α	1	0

$\times$	0	1	α	β
0	0	0	0	0
1	0	1	α	β
α	0	α	β	1
β	0	β	1	α

Figure 15.B.1. Règles d'addition et de multiplication dans $CG(4)$

Comme $\alpha^2 = \beta$, on peut aussi dire que le corps est composé de l'élément nul 0 et des trois puissances successives de α une racine imaginaire du polynôme $D^2 + D + 1$ dans Z_2.

D'autre part, on a également que $D^3 + 1 = (D + 1)(D^2 + D + 1)$. Donc 1 et α sont des racines cubiques de l'unité. De plus, on voit que α^2 est également une racine de $p(D) = (D^2 + D + 1)$. En effet, on a

$$(p(D))^2 = (D^2 + D + 1)^2 = (D^4 + D^2 + 1) = p(D^2)$$

et donc $p(\alpha^2) = p(\alpha)^2 = 0$.

La moralité de notre discussion est que le corps de Gallois $CG(4)$ est composé de l'élément neutre 0 et des trois racines imaginaires de l'équation $D^3 + 1$: $\alpha^0, \alpha^1, \alpha^2$. La multiplication d'élements non-nuls dans ce corps de Gallois est alors équivalente à l'addition des exposants modulo 3.

Eléments primitifs. Un élément d'un corps de Gallois tel que α dans notre exemple précédent qui permet de générer tous les éléments non-nuls par élévation à la puissance est dit "élément primitif". Nous allons montrer qu'un tel élément existe dans tout corps fini. La caractéristique de ce type d'élément α est que l'équation

$$\alpha^n + 1 = 0$$

n'est pas vérifiée pour des "petites" valeurs de n.

Un élément $a \in K$ est dit d'ordre $n = 1, 2, 3, \ldots$ si $a^n = 1$ alors que $a^k \neq 1, \forall k = 1, 2, \ldots, n-1$.

Par exemple dans Z_5, l'élément 2 a l'ordre 4, car dans Z_5 on a $2^2 = 4, 2^3 = 3, 2^4 = 1$.

Dans $\mathbb{R}$, 1 est d'ordre 1 et -1 est d'ordre 2. L'ordre des autres nombres réels est indéfini.

Un élément a d'un corps fini est primitif si tout élément non nul du corps est égal à une certaine puissance de a.

Par exemple le nombre entier 2 est un élément primitif de Z_5.

On a les propriétés suivantes en ce qui concerne les corps finis.

1. *Contrairement au cas de $\mathbb{R}$ où certains éléments (presque tous en fait) n'ont pas d'ordre, dans un corps fini tout élément possède un ordre. De plus, si $a \in K$ est d'ordre n, alors les éléments $a, a^2, a^3, \ldots, a^n$ sont tous distincts et on a $a^k = 1$ si et seulement si k est un multiple de n. (Démonstration : voir [Adá91])*

2. *Si le corps est fini et comporte r éléments, alors un élément est primitif si son ordre vaut $r - 1$.*

   Cela est évident.

3. *Tout corps fini contient un élément primitif.* (Celui-ci n'est pas nécessairement unique).

   En effet, soit a l'élément du corps K (de taille r) d'ordre maximal, disons n. On a forcément $n \leq r - 1$, car la première propriété appliquée à a implique que K possède au moins n élements non-nuls.

   Montrons alors que $n \geq r - 1$ ce qui suffira pour prouver la thèse. On peut montrer que chaque élément non-nul b de K doit avoir un ordre k qui divise n. On aura alors $b^n = 1$ en vertu de la première propriété ci-dessus. Donc, nous aurons prouvé que $D^n - 1$ a au moins $r - 1$ zéros distincts (les $r - 1$ élements non-nuls de K), il doit donc être au moins de degré $r - 1$.

   Soit alors s l'ordre de b et soit la factorisation en nombre premiers de s :

   $$s = p^i q^j \cdots .$$

   Nous allons montrer que p^i doit diviser n, et donc aussi (par symétrie) $q^j \ldots$ et finalement aussi s.

   On peut toujours écrire n sous la forme $n = p^t n'$ où n' n'est pas divisible par p et où éventuellement $t = 0$ (si n n'était pas divisible par p). Il suffit alors de montrer que $i \leq t$. Nous renvoyons le lecteur à la référence [Adá91] plus les détails un peu fastidieux de cette démonstration.

Nous venons de montrer que dans un corps fini de taille r, il existe un élement α primitif d'ordre $r - 1$, tel que tous les éléments non-nuls du corps s'écrivent sous la forme des $r - 1$ puissances successives de α. Tous ces éléments sont des racines du polynôme $D^{r-1} - 1$.

Construction du corps de Gallois $CG(16)$. Nous allons illustrer comment on construit ce corps selon une approche analogue à la façon dont on introduit le corps de nombres complexes par extension du corps des nombre réels aux zéros du polynôme $x^2 + 1$.

Considérons le cas particulier où $p = 2$ et montrons comment on peut construire par exemple le corps d'extension de taille finie $q = p^m$ (ici nous prenons $m = 4, p = 2$, et on a $q = 16$). Pour cela on considère le polynôme $D^{2^m - 1} + 1$ et on procède de la manière suivante :

1. On factorise $D^{2^4-1}+1 = D^{15}+1$ en facteurs irréductibles. Cela donne

$$(1+D+D^4)(1+D^3+D^4)(1+D+D^2+D^3+D^4)(1+D+D^2)(1+D)$$

.

2. On considère un des facteurs qui est de degré m (4 ici), par exemple $p(D) = (1+D+D^4)$.
3. On définit ensuite formellement les racines de ce polynome (il n'en a pas dans le corps binaire, mais c'est comme pour x^2+1 qui n'a pas non plus de racine dans $\mathbb{R}$). Soit α cette racine : on a donc

$$1+\alpha+\alpha^4 = 0.$$

4. Comme α est une racine d'un facteur de $D^{15}+1$ elle est aussi une racine 15-ième de 1. De même, toutes les puissances successives de α sont aussi de racines 15-ièmes de 1.
5. Comme on a $\alpha^{15} = 1$, cela entraine que $\alpha^a\alpha^b = \alpha^{a+b(mod 15)}$. ce qui veut dire que les puissances de α sont calculées modulo 15.
6. On constitue ensuite l'ensemble $0, 1, \alpha, \alpha^2, \ldots, \alpha^{14}$ et on le munit de deux opérations (qui sont des extensions de ces opérations sur le corps binaire)
   (a) Multiplication : addition des exposants modulo 15;
   (b) Addition : on associe à chaque élément α^i un polynôme à coefficients binaires, de degré au plus 3 obtenu en calculant

$$D^i(\text{modulo } p(D)),$$

   et l'addition est obtenue par l'addition de ces polynômes.
   On obtient la représentation suivante.
   0 est représenté par le polynôme nul.
   1 (= α^0) est représenté par le polynôme 1.
   α est représenté par par le polynôme D.
   α^2 est représenté par le polynôme D^2.
   α^3 est représenté par le polynôme D^3.
   α^4 est représenté par le polynôme $1+D$.
   α^5 est représenté par le polynôme $D+D^2$.
   α^6 est représenté par le polynôme D^2+D^3.
   α^7 est représenté par le polynôme $1+D+D^3$.
   α^8 est représenté par le polynôme $1+D^2$.
   α^9 est représenté par le polynôme $D+D^3$.
   α^{10} est représenté par le polynôme $1+D+D^2$.
   α^{11} est représenté par le polynôme $D+D^2+D^3$.
   α^{12} est représenté par le polynôme $1+D+D^2+D^3$.
   α^{13} est représenté par le polynôme $1+D^2+D^3$.
   α^{14} est représenté par le polynôme $1+D^3$.

On peut vérifier que les propriétés d'un corps sont bien assurées par les règles que nous venons de définir : nous savons que les polynômes forment un groupe commutatif et que l'addition modulo 15 définit également un groupe commutatif. Donc les deux opérations jouissent bien des propriétés souhaitées. 0 est un élément absorbant pour la multiplication par définition et on pourra vérifier explicitement la propriété de distributivité. Nous noterons également qu'au lieu de représenter les éléments par des polynômes de degré 4, nous pourrions leur associer les vecteurs de coefficients de ces polynômes et utiliser l'addition vectorielle commme outil de travail.

Nous arrêtons ici notre excursion qui visait à illustrer le type d'outils mathématiques mis en oeuvre durant les 50 dernières années dans la cadre de la poursuite de la limite de Shannon. Nous suggérons au lecteur intéressé de consulter les références sur le codage algébrique pour en savoir plus sur ces codes.

IV Synthèse

16 AU DELÀ DE CE COURS

Ces notes ainsi que le cours oral ont eu pour but d'introduire les notions fondamentales de la théorie de l'information. Les trois parties du cours couvrent respectivement (i) la modélisation et le raisonnement probabiliste; (ii) les trois grands théorèmes de Shannon relatifs au codage fiable et efficace de l'information; (iii) une introduction aux techniques de compression de données et au codage de canal. Du point de vue de l'ingénieur informaticien ou électronicien, cette dernière partie justifie à elle seule l'intérêt de la théorie de l'information et, plus généralement, des méthodes probabilistes. En particulier de nombreux développements récents dans ce domaine ainsi que des recherches en cours, visent à améliorer les performances des techniques de codage en essayant de se rapprocher toujours plus des limites ultimes, énoncées par Shannon voici plus d'un demi siècle.

Dans ce qui suit nous allons brièvement évoquer quelques questions qui touchent au domaine de la théorie de l'information et du codage et qui n'ont pas pu être couvertes dans ce cours.

16.1 THÉORIE DE L'INFORMATION ALGORITHMIQUE

La théorie de l'information algorithmique vise, comme son nom l'indique, à donner une définition *algorithmique* à la notion d'information.

Cette théorie est basée sur la notion de procédure de calcul formalisée par les machines de Turing. En gros, elle dit que le contenu en information apporté par une séquence de bits (et plus généralement par une donnée informatique, qu'on peut toujours supposée représentée sous la forme d'une chaîne de bits) est d'autant plus faible qu'il est facile de produire cette chaîne à l'aide d'un programme.

Plus précisément, on définit la mesure de complexité de Kolmogorov d'une chaîne de bits comme suit. On se donne une machine de Turing universelle (programme et sorties sur un alphabet binaire), et on recherche le programme le plus court dont l'exécution sur cette machine produit en sortie la chaîne de bits. Un résultat fondamental montre que pour des chaînes complexes cette notion est essentiellement indépendante de la machine de Turing particulière par rapport à laquelle on la définit. Plus précisément, si nous désignons par $\mathcal{U}_1$ et $\mathcal{U}_2$ deux machines de Turing universelles quelconques, et par $K_{\mathcal{U}_1}(x)$ et $K_{\mathcal{U}_2}(x)$ les complexités respectives d'une chaîne quelconque x sur ces deux machines, alors il existe une constante c indépendante de x (mais a priori dépendante des deux machines considérées) telle que

$$|K_{\mathcal{U}_1}(x) - K_{\mathcal{U}_2}(x)| \leq c. \tag{16.1}$$

La constante "disparaît" lorsqu'on s'intéresse à des chaînes complexes. Autrement dit la complexité de Kolmogorov est alors indépendante de la machine de Turing universelle considérée.

Notons que cette mesure de complexité ne fait aucunement référence à une modélisation probabiliste d'une source d'information. Cependant, si on considère la complexité de Kolmogorov de messages longs produits par une source, on montre que l'espérance mathématique de la complexité de Kolmorogov par symbole source de ces messages converge vers le débit d'entropie de la source. Autrement dit, les enseignements de la théorie de l'information algorithmique rejoignent ceux de la théorie de l'information de Shannon.

La théorie de l'information algorithmique étudie également différentes questions de type "calculabilité". En particulier, on démontre qu'il n'existe pas d'algorithme capable de calculer la complexité de Kolmorogov d'une chaîne quelconque. Par voie de conséquence, il n'existe pas non plus d'algorithme universellement optimal de compression de données.[1]

Enfin, la théorie de l'information algorithmique fournit une approche systématique au problème générique de modélisation de systèmes à partir d'observations. Dans ce domaine, on applique souvent l'idée que la "meilleure" explication d'une série d'observations est aussi celle qui est la plus simple. En effet, le principe d'Occam[2] dit que lorsqu'on doit expliquer à l'aide d'un modèle mathématique une série d'observations empiriques, il faut choisir un modèle qui n'introduit que les détails strictement nécessaires au vu des observations. Traduit en langage informatique, cela donne le principe de "mininimisation de la longueur de description" qui dit que parmi un ensemble de modèles candidats il faut choisir celui dont la représentation, conjointement avec la représentation des erreurs de prédiction sur les données observées conduit à une complexité de Kolmogorov minimale. De nombreuses méthodes d'apprentissage automatique (et de compression de données) sont fondées sur ce principe en faisant appel à diverses approximations calculables de la complexité de Kolmogorov.

16.2 PRINCIPES ENTROPIQUES

Nous avons vu dans le cadre de ce cours différentes distributions de probabilités qui avaient pour propriété de maximiser l'entropie (ou l'entropie différentielle, dans le cas continu). Par exemple, la loi uniforme maximise l'entropie lorsque seule l'ensemble de variation d'une variable aléatoire est donné. Si, par contre, nous imposons la donnée d'une valeur moyenne et d'une variance, la loi qui maximise l'entropie différentielle est la loi Gaussienne.

La généralisation de cette idée donne lieu au théorème suivant

Distributions d'entropie maximale
Soit $\mathcal{F}$ l'ensemble des densités de probabilité $f(\cdot)$ qui satisfont aux contraintes

$$\int_S f(x) r_i(x) dx = \alpha_i, \; \forall i = 1, \ldots, m, \tag{16.2}$$

et telles que

$$\int_S f(x) dx = 1. \tag{16.3}$$

Et soit $f^*(x) = e^{(\lambda_0 + \sum_{i=1}^m \lambda_i r_i(x))}$ où les λ_i sont choisis pour satisfaire (16.2) et (16.3). Alors, $f^*(x)$ est celle (dans $\mathcal{F}$) qui maximise aussi l'entropie différentielle

$$H_d(x) = -\int_s f(x) \log f(x) dx. \tag{16.4}$$

D'un point de vue pratique, lorsqu'on est dans une situation de modélisation probabiliste, on peut appliquer ce résultat pour choisir une distribution de probabilités compatible avec les hypothèses ou données disponibles. Les équations (16.2) représentent ces hypothèses, et la loi qui maximise (16.4) est celle qui en quelque sorte fait le moins d'hypothèses gratuites supplémentaires, puisqu'elle maximise l'incertitude sur x tout en satisfaisant les contraintes données.

Par exemple, en utilisant $r_1(x) = x$ et $r_2(x) = x^2$ les équations (16.2) reviennent à fixer deux premiers moments de la distribution de x. Le théorème nous dit dans ce cas que la densité de probabilité qui maximise l'entropie prend l'allure $f(x) = e^{(\lambda_0 + \lambda_1 x + \lambda_2 x^2)}$, c'est-à-dire une densité Gaussienne.

[1]Ce qui ne contredit pas le premier théorème de Shannon qui affirme qu'il existe des algorithmes qui en termes de performances moyennes sont arbitrairement proches de ce type d'algorithme.
[2]Guilleaume d'Occam, philosophe théologien anglais du 14ème siècle.

Le principe de maximisation de l'entropie est à mettre en parallèle avec le principe de minimisation de la longueur de représentation totale de la section précédente. Il est notamment utilisé en traitement du signal pour modéliser des processus aléatoires à partir de la connaissance d'une partie de leur fonction d'autocorrélation.

16.3 THÉORIE DES PORTEFEUILLES

La théorie des portefeuilles vise notamment à définir et à étudier des stratégies d'investissement en bourse, en tenant compte d'une part qu'on dispose de ressources limitées et d'autre part que les cours de la bourse ne sont pas parfaitement prévisibles.

Si on représent le cours boursier par un processus aléatoire stationnaire, on démontre que la stratégie optimale est d'autant meilleure que le débit d'entropie est faible. Quantitativement, la somme du taux de doublement pour une stratégie optimale et du débit d'entropie est une constante. L'utilisation d'informations supplémentaires se traduit par un débit d'entropie diminué et donc un taux de doublement qui croit proportionnellement.

16.4 THÉORIE DE L'INFORMATION DES RÉSEAUX

Dans ce cours nous avons étudié le codage de source et de canal dans le cadre d'un protocole de communication simple (appelé paradigme de Shannon) où un émetteur et un récepteur disposent d'un canal dont les caractéristiques sont fixées a priori et qui leur est réservé. Dans ce contexte, nous avons vu que le codage de source et de canal peuvent être découplés, chacun donnant lieu à son propre théorème de Shannon. ce paradigme a conduit au développement séparé des techniques de compression de données d'une part et de codage de canal par ailleurs. Nous avons également montré que l'utilisation d'un canal de feedback entre le destinataire et la source n'augmentait pas la capacité de communication sur un canal sans mémoire.

Que se passe-t-il lorsque plusieurs sources et plusieurs récepteurs communiquent au travers d'un système de communication en se partageant les ressources ? Par exemple, lorsque plusieurs couples de personnes discutent au cours d'un cocktail, ils se partagent un seul canal sonore; ce qui est signal pour les uns devient bruit pour les autres. Comment gérer de manière optimale ce genre de situation multi-utilisateurs, par exemple en maximisant le débit moyen d'utilisation d'un réseau de transmission de données, ou d'un réseau d'antennes GSM ?

En réalité lorsqu'on considère un système multi-utilisateurs il faut considérer de nouveaux éléments tels qu'interférence, coopération et feedback. Le problème est nettement plus complexe que dans le cadre du paradigme de Shannon, et pour communiquer efficacement dans un environnement de réseau il faut en théorie faire appel à un codage simultané source-canal et permettre la coopérations des sources au travers du système de communication. Dans certaines configurations simplifiées (p.ex. les situations de type “broadcast” où une seule source envoie de l'information vers plusieurs récepteurs) des résultats théoriques et pratiques existent. Mais une théorie générale reste encore un objectif lointain à l'heure actuelle.

Bibliographie

[Adá91] J. Adámek, *Foundations of coding*, Wiley Interscience, 1991. 262, 269, 298, 300

[Bat97] G. Battail, *Théorie de l'information. Application aux techniques de communication*, Masson, 1997. 254, 258, 286

[BPJ+98] C. Berrou, R. Pyndiah, M. Jézéquel, A. Glavieux, and P. Adde, *La double correction des turbo codes*, La Recherche (1998), no. 315, 34–37. 254

[Bil79] P. Billingsley, *Probability and measure*, John Wiley and Sons, 1979. 12, 33

[CT91] T. M. Cover and J. A. Thomas, *Elements of information theory*, Wiley, 1991. 2, 45, 205, 207, 221, 232

[Dau92] I. Daubechies, *Ten lectures on wavelets*, SIAM, 1992. 247

[Fre99] R. Frey, *Graphical models for machine learning and information theory*, MIT Press, 1999. 76, 95, 277

[GW93] R. C. Gonzalez and R. E. Woods, *Digital image processing*, Addison Wesley, 1993. 238, 243, 245, 247

[HHJ97] D. Hankerson, G. A. Harris, and P. D. Johnson Jr, *Introduction to information theory and data compression*, CRC Press, 1997. 221, 224, 232, 238, 247

[LKFF98] S. Lin, T. Kasami, T. Fujiwara, and M. Fossorier, *Trellises and trellis-based algorithms for linear block codes*, Kluwer Academic Publishers, 1998. 279

[Mac99] D. J.C. MacKay, *Information theory, inference, and learning algorithms*, Unpublished manuscript, 1999. 277

[Mey93] Y. Meyer, *Wavelets. algorithms and applications.*, SIAM, 1993. 247

[Pea88] J. Pearl, *Probabilistic reasoning in intelligent systems : networks of plausible inference*, Morgan Kaufmann, 1988. 76, 95

[Rab89] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition,", *Proceedings of the IEEE*, Vol. 77, No. 2, 1989, pp. 257–286. 106

[Sap90] G. Saporta, *Probabilités, analyse des données et statistique*, Technip, 1990. 12, 33

[Wel98] D. Welsh, *Codes and cryptography*, Oxford Science Publications, 1998.