

Claude Shannon et la compression des données

Gabriel Peyré

▶ **To cite this version:**

Gabriel Peyré. Claude Shannon et la compression des données. 2016. hal-01343890v3

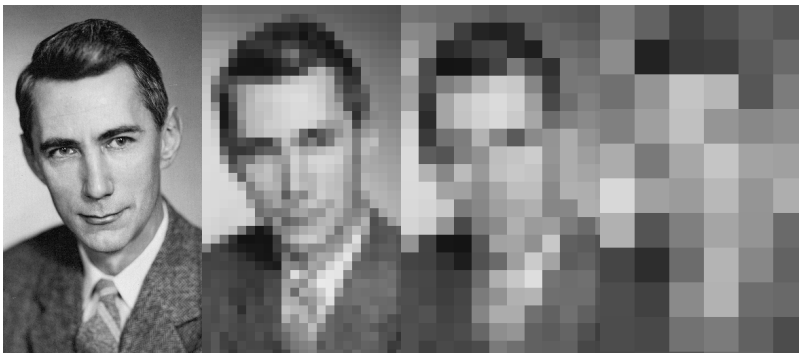
HAL Id: hal-01343890
https://hal.science/hal-01343890v3

Preprint submitted on 16 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Claude Shannon et la compression des données



Gabriel Peyré
CNRS et DMA, École Normale Supérieure
gabriel.peyre@ens.fr

16 septembre 2016

Résumé

L'immense majorité des données (texte, son, image, vidéo, etc.) sont stockées et manipulées sous forme numérique, c'est-à-dire à l'aide de nombres entiers qui sont convertis en une succession de bits (des 0 et des 1). La conversion depuis le monde analogique continu vers ces représentations numériques discrètes est décrite par la théorie élaborée par Claude Shannon (30 avril 1916–24 février 2001), le père fondateur de la théorie de l'information. L'impact de cette théorie sur notre société est absolument colossal. Pourtant son nom est quasi inconnu du grand public. Le centenaire de la naissance de Claude Shannon est donc une bonne excuse pour présenter l'œuvre d'un très grand scientifique.

1 Données numériques et codage

Dans le monde numérique qui nous entoure, toutes les données (images, films, sons, textes, etc.) sont codées informatiquement sous la forme d'une succession de 0 et des 1. Ce codage n'est pas limité au stockage sur des ordinateurs, il est aussi central pour les communications sur internet (envois de courriels, « streaming » vidéo [1], etc.) ainsi que pour des applications aussi diverses que les lecteurs de musique, les liseuses électroniques ou les téléphones portables.

Cependant, les données (par exemple du texte, des sons, des images ou des vidéos) sont initialement représentées sous la forme d'une succession de *symboles*, qui ne sont pas forcément des 0 ou des 1. Par exemple, pour le cas d'un texte, les symboles sont les lettres de l'alphabet. Pour les cas des images, il s'agit des valeurs des pixels. Il faut donc pouvoir convertir cette suite de symboles en une suite de 0 et de 1. Il faut également

1. https://fr.wikipedia.org/wiki/Streaming

pouvoir le faire de façon économe, c'est-à-dire en utilisant la suite la plus courte possible. Ceci est crucial pour pouvoir stocker efficacement ces données sur un disque dur, où bien les transmettre rapidement sur le réseau internet. Cette problématique de *compression* est devenue un enjeu majeur car les données stockées et transmises croissent de façon exponentielle.

La théorie élaborée par Claude Shannon décrit les bases théoriques et algorithmiques de ce codage. Il a formalisé mathématiquement les trois étapes clés de la conversion depuis le monde analogique vers le monde numérique :

(i) l'*échantillonnage* [2], qui permet de passer de données continues à une succession de nombres ;

(ii) le *codage* [3] (on parle aussi de compression), qui permet de passer à une succession la plus compacte possible de 0 et de 1 (on parle de code binaire) ;

(iii) le *codage correcteur d'erreurs* [4], qui rend le code robuste aux erreurs et aux attaques.

Pour chacune de ces étapes, Claude Shannon a établi dans [6, 7], sous des hypothèses précises sur les données et le canal de transmission, des « bornes d'optimalité ». Ces bornes énoncent des limites de performance indépassables, quelle que soit la méthode utilisée. Par exemple, pour la phase de codage (ii), cette borne correspond à la taille théorique minimale des messages binaires permettant de coder l'information voulue. Dans la deuxième moitié du 20^e siècle, des méthodes et des algorithmes de calculs efficaces ont été élaborés permettant d'atteindre les bornes de Shannon, débouchant au 21^e siècle sur l'explosion de l'ère numérique. Cet article se concentre sur la partie (ii) et présente les bases de la compression des données telles que définies par Claude Shannon. Pour la partie (iii), on pourra par exemple consulter cet article d'images des mathématiques [5].

Vous pourrez trouver à la fin de cet article un glossaire récapitulant les termes les plus importants.

2 Codage et décodage

Nous allons maintenant décrire et étudier la transformation (le codage) depuis la suite de symboles $\{0, 1, 2, 3\}$ vers un code binaire, c'est-à-dire une suite de 0 et de 1.

2.1 Exemple d'une image

Dans la suite de cet article, je vais illustrer mes propos à l'aide d'images en niveaux de gris. Une telle image est composée de pixels. Pour simplifier, nous allons considérer seulement des pixels avec 4 niveaux de gris :

— 0 : noir
— 1 : gris foncé,
— 2 : gris clair,
— 3 : blanc.

Cependant, tout ce qui va être décrit par la suite se généralise à un nombre arbitraire de niveaux de gris (en général, les images que l'on trouve sur internet ont 256 niveaux) et même aux images couleurs (que l'on peut décomposer en 3 images monochromes, les composantes rouge, vert et bleue).

2. https://fr.wikipedia.org/wiki/échantillonnage_(signal)
3. https://fr.wikipedia.org/wiki/Compression_de_données
4. https://fr.wikipedia.org/wiki/Code_correcteur
5. http://images.math.cnrs.fr/Qui-est-ce

La figure 1 montre un exemple d'une image avec 4 niveaux de gris, avec un zoom sur un sous-ensemble de 5×5 pixels.

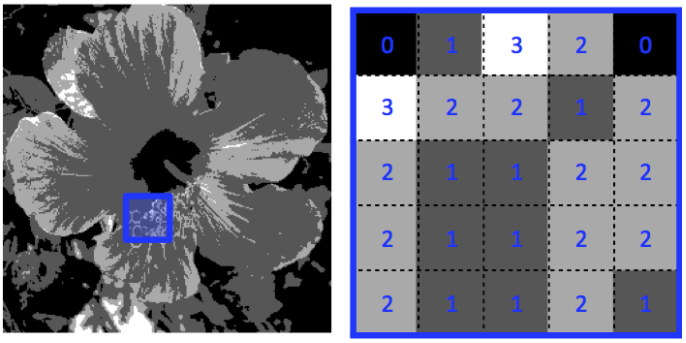


FIGURE 1 – Une image en niveaux de gris et un zoom sur un carré de 5×5 pixels.

Nous allons nous concentrer sur cet ensemble de 25 pixels (le reste de l'image se traite de la même façon). Si on met les unes à la suite des autres les 25 valeurs correspondantes, on obtient la suite suivante de symboles, qui sont des nombres entre 0 et 3

$$(0, 1, 3, 2, 0, 3, 2, 2, 1, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 1, 1, 2, 1).$$

2.2 Codage uniforme

L'étape de codage procède donc en associant à chacun des symboles $\{0, 1, 2, 3\}$ un mot de code, qui est une suite de 0 et de 1.

Une stratégie possible est d'utiliser le codage

$$0 \mapsto 00, \quad 1 \mapsto 01, \quad 2 \mapsto 10, \quad 3 \mapsto 11.$$

Il s'agit d'un cas particulier de codage *uniforme*, qui associe à chaque symbole un mot de code de longueur fixe (ici de longueur constante 2).

Ainsi, la suite de symboles $(0, 1, 3)$ est codée comme

$$(0, 1, 3) \overset{\text{codage}}{\longmapsto} (00, 01, 11) \overset{\text{regroupement}}{\longmapsto} 000111.$$

La suite complète des symboles correspondant à l'image de 5×5 pixels montrée plus haut donnera le code

$$00011110001110100110100101101010010110100101011001.$$

La longueur (c'est-à-dire le nombre de 0 et de 1) de la suite de 0 et de 1 utilisée pour coder un message se mesure en nombre de *bits*. En utilisant le codage uniforme précédent, qui utilise 2 bits par symboles, comme l'on doit coder 25 symboles, on obtient une longueur

$$\bar{\mathcal{L}} = 25 \times 2 = 50 \text{ bits}$$

Le *bit* (« binary digit ») est l'unité fondamentale de l'information, et a été introduite par John Tukey [6], qui était un collaborateur de Claude Shannon.

6. https://fr.wikipedia.org/wiki/John_Tukey

2.3 Logarithme et codage uniforme

Si le nombre N de symboles possibles (ici $N = 4$) est une puissance de 2, c'est à dire que $N = 2^\ell$ (ici $N = 4 = 2^2$ donc $\ell = 2$), on peut toujours construire un tel code *uniforme* où l'on associe à chaque symbole son écriture binaire. On a donné plus haut l'exemple du codage uniforme de $N = 4$ symboles, et le cas de $N = 8$ (donc $\ell = 3$) symboles correspond au codage

$$0 \mapsto 000, \quad 1 \mapsto 001, \quad 2 \mapsto 010, \quad 3 \mapsto 011,$$

$$4 \mapsto 100, \quad 5 \mapsto 101, \quad 6 \mapsto 110, \quad 7 \mapsto 111.$$

Cette écriture binaire a une longueur ℓ, que l'on appelle le logarithme en base de 2[7] de N, ce que l'on note

$$N = 2^\ell \quad \Longleftrightarrow \quad \log_2(N) \overset{\text{def.}}{=} \ell.$$

La définition de $\log_2(x)$ s'étend aussi au cas où x n'est pas une puissance de 2, mais dans ce cas, $\log_2(x)$ n'est pas un nombre entier. Pour un nombre réel strictement positif x, le logarithme vérifie $\log_2(1/x) = -\log_2(x)$, donc par exemple, on a $\log_2(1/4) = -\log_2(4) = 2$.

2.4 Codage à longueur variable

Une question importante est de savoir si l'on peut faire mieux (c'est-à-dire utiliser moins de bits pour coder la même suite de symboles). On peut par exemple utiliser à la place d'un code uniforme, le codage suivant

$$0 \mapsto 001, \quad 1 \mapsto 01, \quad 2 \mapsto 1, \quad 3 \mapsto 000.$$

Avec un tel codage, la suite de symboles $(0, 1, 3)$ est codée comme

$$(0, 1, 3) \overset{\text{codage}}{\longmapsto} (001, 01, 000) \overset{\text{regroupement}}{\longmapsto} 00101000.$$

La suite complète des symboles correspondant à l'image de 5×5 pixels donnera le code

$$0010100010010001101110101111010111010101101.$$

La longueur du code binaire obtenue est donc maintenant

$$\bar{\mathcal{L}} = 42 \text{ bits}$$

Ceci montre qu'on peut donc faire mieux qu'avec un codage *uniforme* en utilisant un codage *variable*, qui associe à chaque symbole un code de longueur variable.

On peut également définir le nombre de bits moyen par symbole $\mathcal{L}$, qui se calcule, ici pour une suite de 25 symboles, comme

$$\mathcal{L} \overset{\text{def.}}{=} \frac{\bar{\mathcal{L}}}{25} = \frac{42}{25} = 1.68 \text{ bits}.$$

Par rapport à un codage uniforme, on voit que le nombre de bits moyen par symbole est passé de $\log_2(N) = 2$ bits à 1.68 bits.

7. https://fr.wikipedia.org/wiki/Logarithme_binaire

2.5 Codage préfixe et décodage

Pour l'instant, on ne s'est occupé que du codage, mais il faut s'assurer que le code obtenu est *décodable*, c'est-à-dire que l'on puisse retrouver la suite de symboles provenant d'un code binaire. Tous les codages ne permettent pas de faire ce chemin inverse.

Pour les codages uniformes, comme le codage

$$0 \mapsto 00, \quad 1 \mapsto 01, \quad 2 \mapsto 10, \quad 3 \mapsto 11.$$

il suffit de séparer la suite de bits en paquets de longueur $\log_2(N)$ (ici $N = 4$ et $\log_2(N) = 2$) et d'utiliser la table de codage en sens inverse. Ainsi, le code binaire 000111 est décodé comme

$$000111 \overset{\text{séparation}}{\longmapsto} (00, 01, 11) \overset{\text{décodage}}{\longmapsto} (0, 1, 3).$$

Par contre, si l'on considère le codage

$$0 \mapsto 0, \quad 1 \mapsto 10, \quad 2 \mapsto 110, \quad 3 \mapsto 101,$$

alors la suite de bits 1010 peut être décodée de deux façons :

$$1010 \overset{\text{séparation}}{\longmapsto} (10, 10) \overset{\text{décodage}}{\longmapsto} (1, 1),$$

ou bien

$$1010 \overset{\text{séparation}}{\longmapsto} (101, 0) \overset{\text{décodage}}{\longmapsto} (3, 0).$$

Ceci signifie que cette suite peut être décodée soit comme la suite de symboles $(1, 1)$, soit comme la suite $(3, 0)$. Le problème est que le mot de codage 10 utilisé pour coder 1 est le début du mot 101 utilisé pour coder 3.

Pour être capable de faire le décodage de façon non ambiguë, il faut qu'aucun mot du codage ne soit le début d'un autre mot. Si c'est le cas, on parle de codage *préfixe*[8], et l'on peut donc effectuer progressivement le décodage. On vérifie facilement que c'est bien le cas du codage non uniforme déjà considéré précédemment

$$0 \mapsto 001, \quad 1 \mapsto 01, \quad 2 \mapsto 1, \quad 3 \mapsto 000.$$

Le décodage progressif du message de 25 symboles des pixels de l'image est effectué ainsi :

$$001010001001000110111010111101011101011 01 \longrightarrow \text{décode } 0$$
$$0\ 01000100100011011101011110101110101101 \longrightarrow \text{décode } 1$$
$$0\ 1\ 000100100011011101011110101110101101 \longrightarrow \text{décode } 3$$
$$0\ 1\ \ 3\ 10010001101110101111010111010 1101 \longrightarrow \text{décode } 2 \ldots$$

2.6 Codes et arbres

Comme le montre la figure 2, en haut à gauche, il est possible de placer l'ensemble des codes binaires de moins de ℓ bits dans un arbre de profondeur $\ell + 1$. Les 2^ℓ mots de longueur exactement ℓ occupent les feuilles, et les mots plus courts sont les nœuds intérieurs.

8. https://fr.wikipedia.org/wiki/Code_préfixe

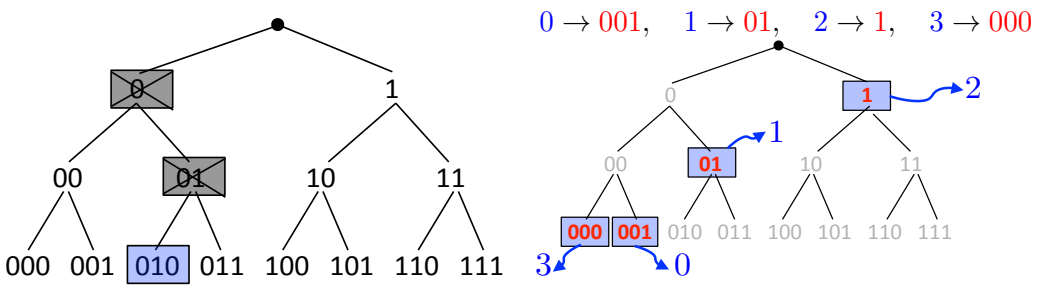


FIGURE 2 – Gauche : arbre complet de tous les codes de longueur 3 ; droite : exemple de codage préfixe.

Les codages préfixes sont alors représentés comme les feuilles des sous-arbres de cet arbre complet. La figure 2, en haut à droite, montre à quel sous-arbre correspond le code à longueur variable

$$0 \mapsto 001, \quad 1 \mapsto 01, \quad 2 \mapsto 1, \quad 3 \mapsto 000.$$

Une fois que l'on a représenté un codage préfixe comme un sous-arbre binaire, l'algorithme de décodage est particulièrement simple à mettre en œuvre. Lorsque l'on commence le décodage, on se place à la racine, et on descend à chaque nouveau bit lu soit à gauche (pour un 0) soit à droite (pour un 1). Lorsque l'on atteint une feuille du sous-arbre, on émet alors le mot du code correspondant à cette feuille, et l'on redémarre à la racine. La figure précédente illustre le processus de décodage.

3 La borne de Shannon

Après avoir décrit les techniques de codage, nous allons maintenant expliquer la théorie de Shannon, qui analyse la performance de ces techniques (c'est-à-dire le nombre de bits nécessaire au codage) en effectuant une modélisation aléatoire du message à coder (qui est composé d'une suite de symboles).

3.1 Code de longueur minimale et modélisation aléatoire

L'utilisation d'un codage préfixe à longueur variable montre que l'on peut obtenir un nombre de bits moyen $\mathcal{L}$ plus faible que le nombre $\log_2(N)$ de bits obtenu par un code uniforme. La question fondamentale, à la fois sur un plan pratique et théorique, est donc de savoir si l'on peut *trouver un codage préfixe donnant lieu à un nombre de bits moyen par symbole minimal*.

Cette question est mal posée, car sa réponse dépend du message qu'il faudra coder, et ce message est en général inconnu a priori. Il faut donc un modèle pour décrire les messages possibles. L'idée fondamentale introduite par Claude Shannon est d'utiliser un modèle probabiliste : on ne sait pas quels messages on aura à coder, mais on suppose qu'on connaît la probabilité d'apparition des symboles composant ce message.

Shannon suppose ainsi que les symboles qui composent le message modélisé sont tirés *indépendamment* [9] selon une variable aléatoire V (la source du message). Ceci signifie que les symboles composants le message modélisé sont des variables aléatoires indépendantes ayant la même distribution que V.

9. https://fr.wikipedia.org/wiki/Indépendances_(probabilité)

3.2 Fréquences empiriques

Afin d'appliquer ce modèle probabiliste à un message donné, on va faire comme si l'on tirait au sort chaque symbole l'un à la suite de l'autre selon des probabilités identiques aux fréquences que l'on observe (en moyenne) dans le cas étudié.

Ceci signifie que l'on impose à la distribution de la source V d'être égale aux fréquences empiriques observées dans le message. Les fréquences empiriques (p_0, p_1, p_2, p_3) sont les fréquences d'apparition des différents symboles $(0, 1, 2, 3)$. Pour la suite des 25 pixels de l'image en niveaux de gris

$$(0, 1, 3, 2, 0, 3, 2, 2, 1, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 1, 1, 2, 1),$$

la fréquence p_1 est égale à $9/25$ car le symbole 1 apparaît 9 fois et que l'on souhaite coder une suite de 25 symboles. La liste des fréquences empiriques pour cette suite de symboles est ainsi

$$p_0 = \tfrac{2}{25}, \quad p_1 = \tfrac{9}{25}, \quad p_2 = \tfrac{12}{25}, \quad p_3 = \tfrac{2}{25}.$$

La modélisation aléatoire impose donc à la variable V d'avoir pour distribution de probabilité (p_0, p_1, p_2, p_3), c'est-à-dire que la probabilité qu'un symbole du message modélisé (supposé généré par la source V) soit égal à $v \in \{0, 1, 2, 3\}$ vaut $\mathbb{P}(V = v) = p_v$.

Ceci constitue un exemple important de modélisation, qui n'est bien sûr pas toujours pertinente mais permet d'analyser finement le problème. Par exemple, dans le cas d'une image, si un pixel est noir, le suivant a de fortes chances de l'être aussi, même si la fréquence globale du noir est faible. Ceci met en défaut l'hypothèse d'indépendance (la section « Transformation de l'information » détaille cet exemple).

3.3 Entropie

Afin de répondre au problème de codage avec un nombre de bits moyen minimum, Shannon a introduit un objet mathématique fondamental : *l'entropie* [10]. L'entropie a été inventée par Ludwig Boltzmann [11] dans le cadre de la thermodynamique [12] et ce concept a été repris par Claude Shannon pour développer sa théorie de l'information. L'entropie de la distribution de la source V est définie par la formule

$$\mathcal{H}_V \overset{\text{def.}}{=} -\sum_{v=0}^{N-1} p_v \times \log_2(p_v).$$

Cette formule signifie que l'on fait la somme, pour tous les symboles v possibles, de la fréquence d'apparition p_v du symbole v multipliée par le logarithme $\log_2(p_v)$ de cette fréquence, puis que l'on prend l'opposé (signe moins) du nombre obtenu.

Comme le logarithme est une fonction croissante, et comme $\log_2(1) = 0$, on a $\log_2(p_v) \leqslant 0$ car $p_v \leqslant 1$ (une probabilité est toujours plus petite que 1). Le signe moins devant la formule définissant l'entropie assure que cette quantité est toujours positive.

Dans notre cas, on a $N = 4$ valeurs pour les symboles, et on utilise donc la formule

$$\mathcal{H}_V \overset{\text{def.}}{=} -p_0 \times \log_2(p_0) - p_1 \times \log_2(p_1) - p_2 \times \log_2(p_2) - p_3 \times \log_2(p_3).$$

10. https://fr.wikipedia.org/wiki/Entropie
11. https://fr.wikipedia.org/wiki/Ludwig_Boltzmann
12. https://fr.wikipedia.org/wiki/Entropie_(thermodynamique)

Il est à noter que si jamais $p_v = 0$, alors il faut utiliser la convention $p_v \times \log_2(p_v) = 0 \times \log_2(0) = 0$. Cette convention signifie que l'on ne prend pas en compte les probabilités nulles dans cette formule.

Le but de l'entropie est de quantifier l'incertitude sur les suites de symboles possibles générées par la source V. On peut montrer que l'entropie vérifie

$$0 \leqslant \mathcal{H}_V \leqslant \log_2(N).$$

Les deux valeurs extrêmes correspondent ainsi à des incertitudes respectivement minimale et maximale.

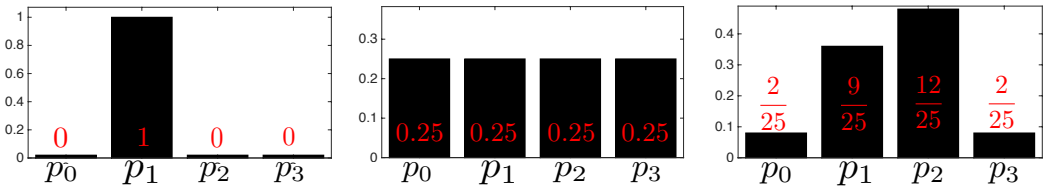


FIGURE 3 – Trois exemples de distributions de probabilité avec les entropies correspondantes.

— **Entropie minimale.** L'entropie $\mathcal{H}_V = 0$ est minimale lorsque les fréquences p_v sont toutes nulles sauf une. La figure 3, gauche, montre le cas où $p_1 = 1$ et toutes les autres probabilités sont nulles.
Dans ce cas, on a

$$\mathcal{H}_V = -0 \times \log_2(0) - 1 \times \log_2(1) - 0 \times \log_2(0) - 0 \times \log_2(0) = 0.$$

où l'on rappelle que $\log_2(1) = 0$ et que, par convention, on a $0 \times \log_2(0) = 0$. Ceci correspond à la modélisation d'une suite constante de symboles, et la source génèrera par exemple avec probabilité 1 la suite suivante de 25 symboles

$$(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0).$$

— **Entropie maximale.** Au contraire, $\mathcal{H}_V = \log_2(N)$ est maximale lorsque toutes les fréquences sont égales, $p_v = 1/N$. Dans notre cas où $N = 4$, on a en effet

$$\mathcal{H}_V = -\tfrac{1}{4} \times \log_2(\tfrac{1}{4}) - \tfrac{1}{4} \times \log_2(\tfrac{1}{4}) - \tfrac{1}{4} \times \log_2(\tfrac{1}{4}) - \tfrac{1}{4} \times \log_2(\tfrac{1}{4}) = \log_2(4) = 2,$$

où l'on a utilisé le fait que $\log_2(1/x) = -\log(x)$ et donc en particulier $\log_2(\frac{1}{4}) = -\log_2(4)$. La figure 3, centre, suivante montre l'histogramme correspondant à ce cas.
Cette situation correspond intuitivement à la modélisation d'une suite maximalement *incertaine*. Voici par exemple deux suites de 25 symboles générés par une telle source V

$$(2,2,1,1,3,0,3,3,3,0,1,1,2,0,2,0,2,1,3,2,0,2,2,1,3),$$

$$(3,3,1,2,0,0,2,2,1,3,2,2,3,3,2,0,0,3,0,1,3,0,1,1,2).$$

— **Entropie intermédiaire.** Les situations intermédiaires entre ces deux extrêmes correspondent à des entropies intermédiaires. Par exemple, on peut considérer la distribution des 25 pixels considérés au début de cet article, qui correspondent au message

$$(0,1,3,2,0,3,2,2,1,2,2,1,1,2,2,2,1,1,2,2,1,1,2,1).$$

Pour cette distribution, on rappelle que l'on a les probabilités

$$p_0 = \tfrac{2}{25}, \quad p_1 = \tfrac{9}{25}, \quad p_2 = \tfrac{12}{25}, \quad p_3 = \tfrac{2}{25},$$

la figure 3, droite, montre l'histogramme correspondant à ces valeurs.
L'entropie vaut alors

$$\mathcal{H}_V = -\tfrac{2}{25} \times \log_2(\tfrac{2}{25}) - \tfrac{9}{25} \times \log_2(\tfrac{9}{25}) - \tfrac{12}{25} \times \log_2(\tfrac{12}{25}) - \tfrac{2}{25} \times \log_2(\tfrac{2}{25}) \approx 1.62,$$

ce qui correspond bien une valeur « intermédiaire » de l'entropie.

3.4 Nombre de bits moyen d'une source

Dans la suite, on note c_v le code associé à un symbole v. On note $L(c_v)$ la longueur (i.e. le nombre de bits) de chaque mot c_v de code. Pour un codage uniforme, alors la longueur est constante $L(c_v) = \log_2(N)$. Par contre, si l'on prend l'exemple du codage variable

$$0 \mapsto c_0 \overset{\text{def.}}{=} 001, \quad 1 \mapsto c_1 \overset{\text{def.}}{=} 01, \quad 2 \mapsto c_2 \overset{\text{def.}}{=} 1, \quad 3 \mapsto c_3 \overset{\text{def.}}{=} 000,$$

alors $L(c_0) = L(001) = 3$.

On remarque que l'on peut calculer le nombre de bit moyen $\mathcal{L}$ du codage d'un message à l'aide des fréquences empiriques comme suit :

$$\mathcal{L} = \sum_{v=0}^{N-1} p_v \times L(c_v).$$

Cette formule signifie que l'on fait la somme, pour tous les symboles v possibles, de la fréquence d'apparition p_v du symbole multipliée par la longueur $L(c_v)$ du mot de code c_v. Par exemple, dans notre cas, pour $N = 4$, on a la formule

$$\mathcal{L} = p_0 \times L(c_0) + p_1 \times L(c_1) + p_2 \times L(c_2) + p_3 \times L(c_3).$$

Dans le cadre de la modélisation aléatoire à l'aide d'une source V, on va noter $\mathcal{L}_V$ ce nombre de bit moyen, qui est associé à la source V ayant la distribution $(p_v)_v$.

3.5 Borne de Shannon pour le codage

Claude Shannon a montré dans son article [6] que l'entropie permettait de borner le nombre de bits moyen $\mathcal{L}_V$ dans le cadre de ce modèle aléatoire. Il a en effet montré que pour tout codage préfixe, on a

$$\mathcal{H}_V \leqslant \mathcal{L}_V.$$

Il s'agit d'une borne inférieure, qui dit qu'aucun codage préfixe ne peut faire mieux que cette borne.

Ce résultat est fondamental, car il décrit une limite indépassable, quelle que soit la technique de codage préfixe utilisée. Sa preuve est trop difficile pour être exposée ici, elle utilise la représentation sous forme d'arbre détaillée plus haut à la section 2.6, on pourra

regarder par exemple [3] pour obtenir tous les détails. Cette preuve montre qu'il faut dépenser en moyenne au moins $-\log_2(p_v)$ bits (qui est, comme on l'a déjà vu, toujours un nombre positif) pour coder un symbole v si l'on veut avoir un codage efficace. Les symboles les plus fréquents doivent nécessiter moins de bits, car p_v est plus petit, donc la longueur optimale $-\log_2(p_v)$ l'est également. Ceci qui est très naturel, comme on peut en particulier le voir pour les deux cas extrêmes :

— **Entropie minimale.** Si $\mathcal{H}_V = 0$, alors avec probabilité 1, la suite de symboles est composée d'un unique symbole. Dans ce cas de figure, l'utilisation d'un codage préfixe est très inefficace, car celui-ci doit utiliser au moins un bit par symbole i.e. $\mathcal{L}_V \geqslant 1$, et donc un tel codage est loin d'atteindre la borne de Shannon.
L'entropie étant nulle, la borne dit que l'on souhaiterait ne rien dépenser pour le codage. Ceci est logique, car il n'y a pas besoin de coder une telle suite (puisque c'est toujours la même). Des techniques de codage plus avancées (par exemple le codage arithmétiques [13] [5]) permettent de contourner ce problème et atteignent la borne de Shannon quand le nombre de symboles à coder tend vers l'infini.

— **Entropie maximale.** Si $\mathcal{H}_V = \log_2(N)$, alors tous les symboles sont équiprobables, donc on doit utiliser des mots de code de même longueur pour tous les symboles, ce qui est obtenu par un code uniforme. Comme on l'a vu plus haut, un tel code nécessite $\mathcal{L}_V = \log_2(N) = \mathcal{H}_V$ bits par symbole, et donc la borne inférieure de Shannon est atteinte dans ce cas.

— **Entropie intermédiaire.** Pour le cas de la distribution des 25 pixels considérés au début de cet article, qui correspondent au message

$$(0, 1, 3, 2, 0, 3, 2, 2, 1, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 1, 1, 2, 1),$$

on rappelle que l'entropie et le nombre moyen de bits, qui ont déjà été calculés, valent respectivement

$$\mathcal{H}_V \approx 1.62 \text{ bits} \quad \text{et} \quad \mathcal{L}_V = 1.68 \text{ bits}.$$

Ces valeurs sont bien en accord avec la borne de Shannon, et montrent que le codage préfixe utilisé permet d'être assez proche de cette borne.

On peut se demander si cette borne est précise, et s'il est possible de construire des codes atteignant la borne de Shannon dans tous les cas (et pas juste les deux cas extrêmes). Huffmann a proposé dans [2] une construction d'un codage « optimal » (i.e. ayant la longueur moyenne $\mathcal{L}_V$ minimale pour une source V donnée) à l'aide d'un algorithme élégant. La longueur moyenne obtenue par ce codage vérifie

$$\mathcal{H}_V \leqslant \mathcal{L}_V \leqslant \mathcal{H}_V + 1.$$

Le fait que cette longueur moyenne puisse être potentiellement aussi grande que $\mathcal{H}_V + 1$ (et donc assez différente de la borne inférieure de Shannon $\mathcal{H}_V$) provient du fait que la longueur $L(c_v)$ d'un mot c_v du code est un nombre entier, alors que la longueur optimale devrait être $-\log_2(p_v)$, qui n'est pas en général un nombre entier. Pour pallier ce problème, il faut coder les symboles par groupes, ce qui peut être effectué de façon efficace à l'aide des codages arithmétiques [14] [5], qui atteignent la borne de Shannon lorsque l'on code une suite infinie de symboles.

La théorie de Shannon permet donc de borner la longueur moyenne, ce qui donne une information importante sur la performance d'une méthode de codage pour une source

13. https://fr.wikipedia.org/wiki/Codage_arithmétique
14. https://fr.wikipedia.org/wiki/Codage_arithmétique

donnée. Cette borne ne donne cependant pas d'information sur d'autres quantités statistiques potentiellement intéressantes, telles que la longueur maximale ou la longueur médiane.

3.6 Transformation de l'information

La borne de l'entropie précédente fait l'hypothèse que les symboles qui composent le message à coder sont générés de façon *indépendante* par la source V. Cette hypothèse permet une analyse mathématique simple du problème, mais elle est en général fausse pour des données complexes, comme par exemple pour l'image montrée à la figure suivante. En effet, on voit bien que la valeur d'un pixel n'est pas du tout indépendante de celles de ses voisins. Par exemple, il y a de grandes zones homogènes où la valeur des pixels est quasi-constante.

Afin d'améliorer les performances de codage, et obtenir des méthodes de compression d'image efficaces, il est crucial de retransformer la suite de symboles, afin de réduire son entropie en exploitant les dépendances entre les pixels. Une transformation très simple permettant de le faire consiste à remplacer les valeurs des P pixels $(v_i)_{i=1}^{P}$ par celles de leurs différences $(d_i \overset{\text{def.}}{=} v_i - v_{i-1})_{i=1}^{P-1}$. En effet, dans une zone uniforme, les différences successives vont être nulles car les pixels ont la même valeur. La figure 4 montre comment effectuer un tel calcul. Elle montre aussi que cette transformation est bijective, c'est-à-dire que l'on peut revenir aux valeurs d'origine $(v_i)_i$ en effectuant une sommation progressive des différences, c'est-à-dire en calculant

$$v_i = v_0 + \sum_{j=1}^{i} d_j.$$

Afin de pouvoir faire cette inversion, il faut bien sûr avoir conservé la valeur v_0 du premier pixel. La bijectivité de la transformation

$$(v_0, \ldots, v_{P-1}) \longmapsto (v_0, d_1, \ldots, d_{P-1})$$

est cruciale pour pouvoir faire le décodage et afficher l'image décodée.

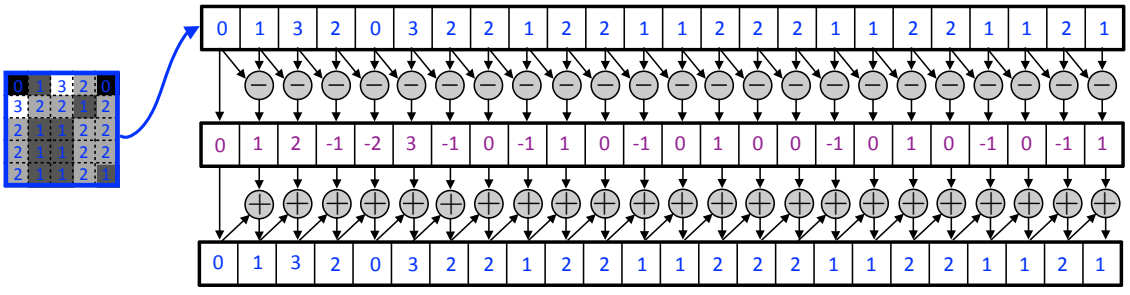
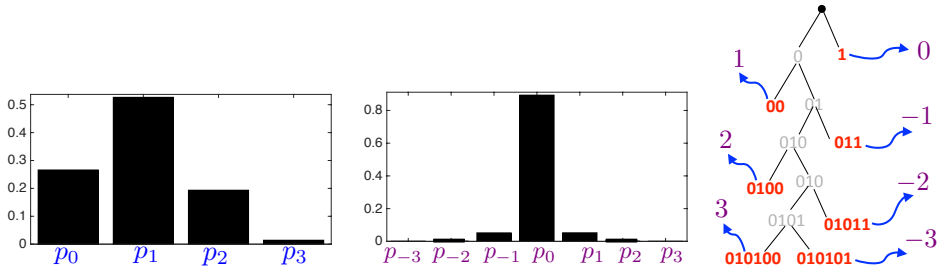


FIGURE 4 – Représentation par différences

Comme les pixels peuvent prendre les valeurs $\{0, 1, 2, 3\}$, les différences peuvent prendre quant à elles les valeurs $\{-3, \ldots, 3\}$. Elles peuvent en particulier être négatives (ce qui ne pose pas de problème particulier pour définir un codage). La figure suivante compare les histogrammes des pixels et des différences. On constate que l'histogramme des différences est beaucoup plus « piqué » au voisinage de 0, ce qui est logique, car

de nombreuses différences (correspondant aux zones homogènes) sont nulles ou petites. L'entropie $\mathcal{H}_D$ de l'histogramme des différences (que l'on peut modéliser avec une source D) est donc nettement plus faible que l'entropie $\mathcal{H}_V$ des pixels.

La figure 5 montre une comparaison des histogrammes des valeurs des pixels et des différences, calculés sur l'ensemble de l'image (et pas uniquement sur le sous-ensemble de 25 pixels initial). Elle montre également l'arbre d'un codage préfixe optimal (calculé par l'algorithme de Huffman [2]) associé à l'histogramme des différences.



$\mathcal{H}_V \approx 1.54, \mathcal{L} \approx 1.67$ $\mathcal{H}_D \approx 0.61, \mathcal{L} \approx 1.16$ Arbre de codage

FIGURE 5 – Comparaison des histogrammes des valeurs des pixels et des différences, et un arbre de codage de ces différences.

Cet arbre correspond au codage

$$-3 \mapsto 010101, -2 \mapsto 01011, -1 \mapsto 011, 0 \mapsto 1, 1 \mapsto 00, 2 \mapsto 0100, 3 \mapsto 010100.$$

Ce codage a une longueur moyenne $\mathcal{L} \approx 1.16$ bits. Ce nombre moyen est bien conforme à la borne de l'entropie, et il est significativement plus petit que la longueur moyenne associée à l'histogramme des pixels (1.67 bits), qui est elle-même plus petite que la longueur moyenne associée à un codage uniforme ($\log_2(4) = 2$ bits). Si l'on répercute ces longueurs au codage de la totalité de l'image de 256×256 en niveau de gris, on obtient ainsi les gains suivants, où 1 ko = 8×1024 bits est un *kilo octet*.

Codage uniforme		Codage des pixels		Codage des différences
16.3 ko	$\longrightarrow$	13.7 ko	$\longrightarrow$	9.5 ko

Les méthodes les plus performantes de compression d'image utilisent des transformations plus complexes, et exploitent de façon plus fine la régularité locale des images. C'est le cas de la méthode de compression JPEG-2000 [15], qui est considérée comme la plus efficace à l'heure actuelle, et qui utilise la théorie des ondelettes [16], voir le livre [3] pour plus de détails. Il existe bien d'autres cas où la non-indépendance des symboles peut être utilisée pour améliorer les performances de codage. Un exemple important est celui de la suite des lettres composant un texte.

4 Conclusion

La théorie mathématique initiée par Claude Shannon définit un cadre de pensée nécessaire à l'élaboration de techniques efficaces pour l'acquisition, le traitement, le stockage et la transmission des données sous forme numérique. Ce sont ces techniques qui

15. https://fr.wikipedia.org/wiki/JPEG_2000
16. https://fr.wikipedia.org/wiki/Ondelette

ont révolutionné les communications et l'informatique durant la deuxième moitié du 20^e siècle, et ont permis la croissance d'internet au début du 21^e siècle. Sans les apports révolutionnaires de Shannon, vous ne pourriez pas partir en vacances avec votre bibliothèque entière dans votre liseuse électronique, et tous les épisodes de *Game of Thrones* sur votre tablette !

Pour obtenir plus de détails sur la théorie de l'information, on pourra consulter [1], pour son utilisation en traitement du signal et de l'image, on pourra regarder [3]. Les codes informatiques permettant de reproduire les figures de cet article sont disponibles en ligne [17], et d'autres codes sont accessibles sur le site `www.numerical-tours.com` [4].

Glossaire

- **Pixel** : emplacement sur la grille carrée d'une image, parfois utilisé pour faire référence à la valeur associée.
- **Symbole** : élément v d'un ensemble fini, par exemple $\{0, \ldots, N-1\}$.
- **Code** : succession de 0 et de 1 utilisé pour coder un symbole v.
- **Codage** : ensemble des correspondances entre un symbole v et un code binaire, par exemple $2 \mapsto 10$. Fait aussi référence à l'action de remplacer une suite de symboles par un ensemble de bits.
- **Distribution empirique** : fréquence p_v d'apparition du symbole v dans la suite de symboles à coder.
- **Histogramme** : synonyme de distribution empirique, fait aussi parfois référence à la représentation graphique de ces valeurs.
- **Source** : variable aléatoire V modélisant les symboles, avec la distribution $\mathbb{P}(V = v) = p_v$.
- **Entropie** : $\mathcal{H}_V$ est un nombre positif associé à la source V, et qui dépend de sa distribution de probabilité $(p_v)_v$.
- **Nombre de bits moyen d'une suite :** $\mathcal{L}$ est associé au codage d'une suite de symboles.
- **Nombre de bits moyen de la source :** $\mathcal{L}_V$ est associé au codage des symboles générés par V.

Remerciements

Je remercie Marie-Noëlle Peyré, Gwenn Guichaoua, François Béguin, Gérard Grancher, Aurélien Djament et François Sauvageot pour leurs relectures attentives.

L'image de la fleur est due à Maitine Bergounioux. L'image de Shannon utilisée pour le logo de l'article est due à l'utilisateur telehistoriska du site flickr (sous license CC BY-NC 2.0).

Références

[1] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2006.

[2] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9) :1098–1101, 1952.

17. `https://github.com/gpeyre/2016-shannon-theory`

[3] S. G. Mallat. *A wavelet tour of signal processing*. Elsevier/Academic Press, Amsterdam, third edition, 2009.

[4] G. Peyré. The numerical tours of signal processing - advanced computational signal and image processing, `www.numerical-tours.com`. *IEEE Computing in Science and Engineering*, 13(4) :94–97, 2011.

[5] J. Rissanen and G. Langdon. Arithmetic coding. *IBM Journal of Research and Development*, 23(2) :149–162, 1979.

[6] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3) :379–423, 1948.

[7] C. E. Shannon. Communication in the presence of noise. *Proc. Institute of Radio Engineers*, 37(1) :10–21, 1949.