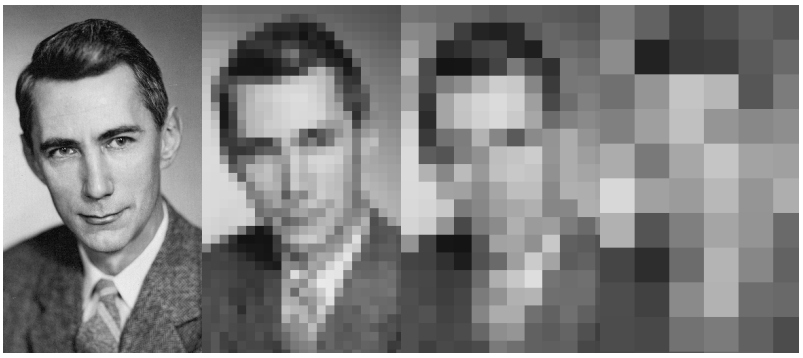
Claude Shannon et la compression des données

Gabriel Peyré

Claude Shannon et la compression des données



Gabriel Peyré
CNRS et Université Paris-Dauphine
gabriel.peyre@ceremade.dauphine.fr

5 août 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, le père fondateur de la théorie de l'information. L'impact de cette théorie sur notre société est comparable à celui des théories de Darwin ou d'Einstein. 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.

La théorie de Shannon décrit les trois étapes clés de la conversion depuis le monde analogique vers le monde numérique :

(i) l'échantillonnage, qui permet de passer de fonctions continues à une succession de nombres ;

(ii) la compression, qui permet de passer à une succession la plus compacte possible de chiffres binaires ;

(iii) le codage correcteur d'erreurs, qui rend le message binaire 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. 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 présente les bases de la compression des données (partie (ii)) telles que définies par Claude Shannon. Les deux autres parties (i) et (iii) ne seront que brièvement mentionnées, et nécessitent des connaissances mathématiques plus avancées.

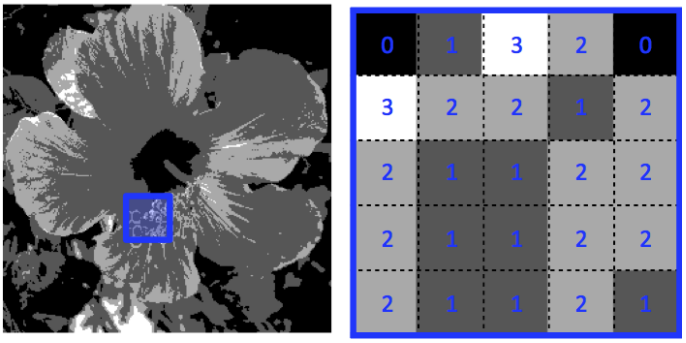


FIGURE 1 – Une image en niveaux de gris et un zoom sur un carré de 5×5 pixels.

1 Codage uniforme

Les données manipulées par un ordinateur doivent être représentées à l'aide de l'écriture binaire, c'est-dire une succession de 0 et de 1. Cependant, les données (par exemple du texte, des images ou des sons) sont initialement représentées sous la forme d'une sucession 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.

Dans la suite de cet article, plutôt que d'utiliser du texte, 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).

La figure 1 montre un exemple d'une image avec 4 niveaux de gris, avec un zoom sur un sous-ensemble de $5x5$ pixels.

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

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

L'étape de codage procède en associant à chaqu'un 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

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

Ainsi, la suite de symbole $(0, 1, 3)$ sera 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 donnera le code

$$0001111000111010011010010110101001011010011011001.$$

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

2 Logarithme et code binaire

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

$$N = 2^\ell \quad \Longleftrightarrow \quad \log_2(N) := \ell.$$

Il est posible de définir $\log_2(N)$ même lorsque N n'est plus une puissance de 2, mais dans ce cas, $\log_2(N)$ n'est plus en général un nombre entier.

3 Codage à longueur variables

La longueur (c'est-à-dire le nombre de zeros et de uns) du code binaire utilisé 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* est l'unité fondamentale de l'information, et a été introduite principalement par Claude Shannon.

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 symbole $(0, 1, 3)$ sera codée comme

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

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

$$0010100010010001101110101111010111010101.$$

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.

4 Codage prefixe et décodage

Pour l'instant, on ne s'est occupé que du codage, mais il faut s'assurer que le message 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

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

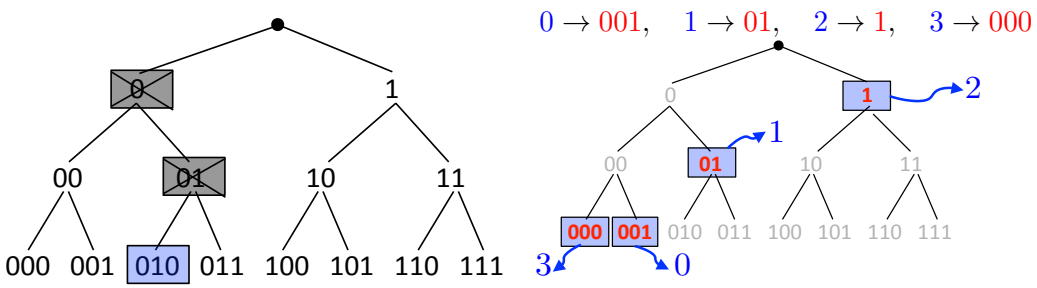


FIGURE 2 – Gauche : arbre complet de tous les codes de longueur 3 ; droite : exemple de codage préfixe.

c'est facile, 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 sera 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-ambigüe, il faut qu'aucun mot du codage ne soit le début d'un autre mot. Si ce n'est pas le cas, on parle de codage *préfixe*, et l'on peut donc effectuer progressivement le décodage. Le décodage progressif de ce message est effectué ainsi :

$$001010001001000110111010111101011101011101 \longrightarrow \text{décode } 0$$
$$0\ 01000100100011011101011110101110101101 \longrightarrow \text{décode } 1$$
$$0\ 1\ 000100100011011101011110101110101101 \longrightarrow \text{décode } 3$$
$$0\ 1\ \ 3\ 1001000110111010111101011101011101 \longrightarrow \text{décode } 2\ldots$$

5 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 noeuds intérieurs.

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 oeuvre. 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.

6 Fréquence empiriques

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 apparait 9 fois et que l'on souhaite coder une suite de 25 symboles. La liste des fréquences pour cette suite de symboles est ainsi

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

On peut utiliser ces fréquences empirique pour calculer la longueur $\bar{\mathcal{L}}$ de la suite de bits obtenue avec un codage. Si l'on prend l'exemple du codage variable

$$0 \mapsto c_0 := 001, \quad 1 \mapsto c_1 := 01, \quad 2 \mapsto c_2 := 1, \quad 3 \mapsto c_3 := 000,$$

le nombre de bits est alors 25 fois le nombre de bits moyen par symbole $\mathcal{L}$

$$\bar{\mathcal{L}} := 25 \times \mathcal{L}$$

et ce nombre de bits moyen est obtenu en sommant les fréquences multipliées par les longueurs des mots de codage

$$\mathcal{L} := p_0 \times L(c_0) + p_1 \times L(c_1) + p_2 \times L(c_2) + p_3 \times L(c_3) = \tfrac{42}{25} \approx 1.68 \text{ bits},$$

où l'on a noté $L(c_0) = L(001) = 3$ la longueur (nombre de bits) du mot de code $c_0 := 001$. 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. Le code à longueur variable a permis de gagner de la place de stockage.

7 Entropie et modélisation aléatoire

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 des données.

Le codage préfixe à longueur variable montre que l'on peut ainsi obtenir un nombre de bits moyen $\mathcal{L}$ plus faible que le nombre $\log_2(N)$ de bits obtenu par un code uniforme. On peut se demander quel est le nombre minimal de bits nécessaire pour coder une suite

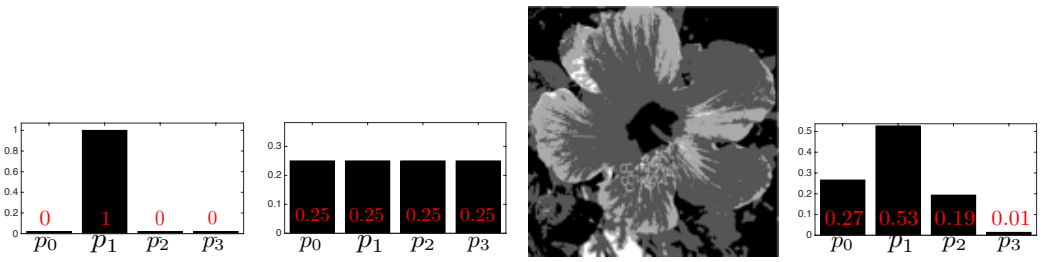


FIGURE 3 – Trois exemples de distributions de probabilité avec les entropies correspondantes. L'histogramme de droite est celui des fréquences d'apparition des niveaux de gris quantifiés de l'image.

de symboles. Ce nombre de bits constitue en quelque sorte la complexité intrinsèque de cette suite. Afin de répondre mathématiquement à cette question, Claude Shannon a proposé d'étudier un modèle aléatoire des symboles à coder. Il suppose que les symboles qui composent la suite sont tirés *indépendamment* selon une variable aléatoire V (la source du message). Ceci signifie que les symboles composants le message sont modélisés comme des variables aléatoires indépendantes ayant la même distribution que V.

Dans le cas de $N = 4$ valeurs possibles pour les symboles, cette variable V a pour distribution de probabilité (p_0, p_1, p_2, p_3), c'est-à-dire que la probabilité que chaque symbole de la suite (supposé généré par la source V) soit égal à $v \in \{0, 1, 2, 3\}$ est égale à $\mathbb{P}(V = v) = p_v$. Ceci constitue un exemple important de modélisation, qui n'est bien sûr pas toujours vraie (on verra plus bas que l'hypothèse d'indépendance peut être remise en cause), mais permet d'analyser le problème très finement. En pratique, on souhaite coder une suite de symboles, et l'on va définir (p_0, p_1, p_2, p_3) à l'aide des fréquences empiriques définies plus haut. Le modèle sera donc d'autant plus précis que le nombre P (ici $P = 25$) de symboles à coder est grand.

L'entropie de la distribution de la source V est définie par la formule

$$\mathcal{H}_V := -\sum_{v=0}^{N-1} p_v \log(p_v),$$

où l'on utilise la convention $0 \log(0) = 0$. Dans notre cas, on a $N = 4$ valeurs pour les symboles. Cette convention signifie que l'on ne prend pas en compte les probabilités nulles dans cette formule.

On peut montrer que l'entropie vérifie

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

L'entropie $\mathcal{H}_V = 0$ est minimum lorsque les fréquences p_v sont toutes nulles sauf une (figure suivante, gauche), ce qui correspond à la modélisation d'une suite constante de symboles. Au contraire, $\mathcal{H}_V = \log_2(N)$ est maximale lorsque toutes les fréquences sont égales, $p_v = 1/N$, ce qui correspond intuitivement à la modélisation d'une suite maximalement « incertaine ». Les situations intermédiaires entre ces deux extrêmes (comme par exemple la distribution des pixels d'une image montrée sur la figure suivante) correspondent à des entropies intermédiaires. Intuitivement, l'entropie correspond donc à une mesure du degré d'incertitude d'une distribution.

La figure 3 montre trois exemples de distributions de probabilité avec les entropies correspondantes. L'histogramme de droite est celui des fréquences d'apparition des niveaux de gris quantifiés de l'image en niveau de gris initiale.

Si l'on considère un codage $(c_v)_v$ des différents symboles, et l'on note $L(c_v)$ la longueur de chaque mot de code, alors le nombre de bits moyen qu'il faudra pour coder des symboles générés par la source V est

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

où l'on rappelle que $L(c_v)$ est la longueur (i.e. le nombre de bits) du mot de code c_v utilisé pour coder le symbole v. Il faut bien distinguer le nombre de bits moyen $\mathcal{L}$ pour coder une suite finie donnée de symboles, et le nombre $\mathcal{L}_V$ qui est associé à une modélisation de données à l'aide d'une source. Si la modélisation est exacte, c'est-à-dire que les symbole qui composent le message sont bien tirés indépendamment selon la même distribution que la source V, alors, selon la loi des grands nombres, on a

$$\mathcal{L} \overset{P\to+\infty}{\longrightarrow} \mathcal{L}_V,$$

(la convergence étant vraie « presque surement »).

8 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. On peut bien sûr se demander si cette borne est précise, et s'il est possible de construire des codes atteignant la borne de Shannon. 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 simple et élégant de programmation dynamique. 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 loin de la borne de Shannon) 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 palier ce problème, il faut coder les symboles par groupes, ce qui peut être effectué de façon efficace à l'aide des codages arithmétiques [5], qui atteignent la borne de Shannon lorsque l'on code une suite infinie de symboles.

9 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

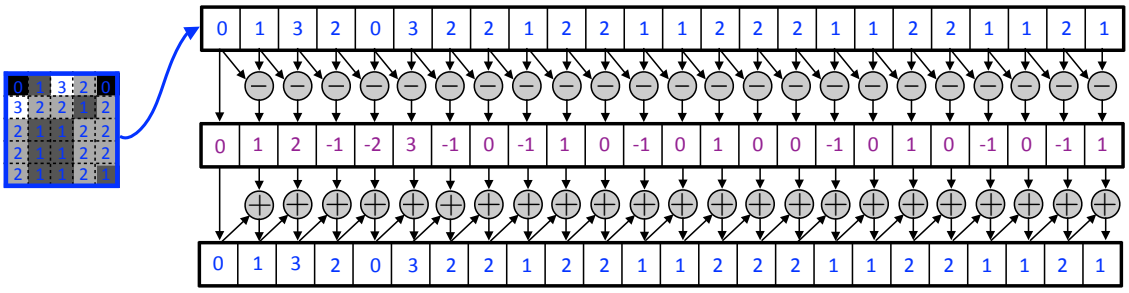


FIGURE 4 – Représentation par différences

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 re-transformer 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 $(d_i := v_i - v_{i-1})_{i=1}^{P-1}$. 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

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

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

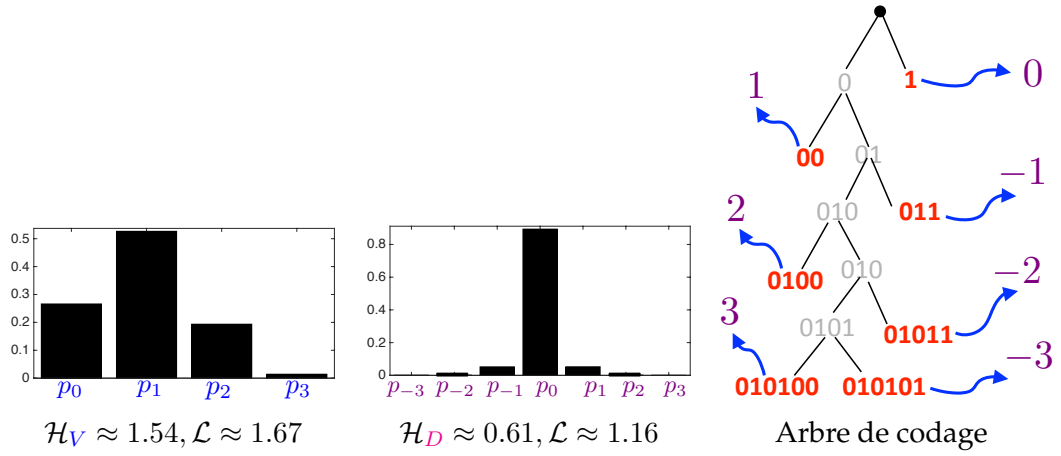
Comme les pixels peuvent prendre les valeurs $\{0, 1, 2, 3\}$, les différences peuvent prendre quant à elles les valeurs $\{-3, \dots, 3\}$. Elles peuvent en particulier être négatives (ce qui ne pose pas de problème particulier pour définir un codage). La figure suivante compare les histogrammes des 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. Elle montre également l'arbre d'un codage préfixe optimal (calculé par l'algorithme de Huffman [2]) associé à l'histogramme des 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



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

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

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'images 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, qui est considérée comme la plus efficace à l'heure actuelle, et qui utilise la théorie des ondelettes, voir le livre [3] pour plus de détails.

10 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 ont révolutionné les communications et l'informatique durant la deuxième moitié du 20^e siècle, et ont permis l'émergence d'internet au début du 21^e siècle. Cette théorie utilise des modèles aléatoires pertinents pour les données et les méthodes de transmission, et énonce des bornes indépassables, qui permettent de quantifier l'efficacité des méthodes proposées par la suite. Pour les modèles les plus simples considérés par Shannon, ces bornes ont été atteintes pour les trois parties (i), (ii) et (iii) de sa théorie qui ont été mentionnées dans l'introduction. En particulier le codage de Hufmann [2] et le codage arithmétique [5] apportent une solution pour la compression de symboles générés indépendamment par une source. Un des enjeux des disciplines gravitant autour de la théorie de l'information, du traitement du signal et de l'image est de proposer des modèles plus riches et de rechercher des bornes et des méthodes optimales.

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 sur le site `www.numerical-tours.com` [4].

Glossaire

- **Pixels** : 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 symbole générés par V.

Remerciements

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.

[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.