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

11 juillet 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. Cette théorie décrit les trois étapes clés de cette 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.

1 Données numériques et information

Le problème de la compression consiste à stocker de manière économe (c'est-à-dire dans un fichier informatique prenant le moins de place possible) des données (par exemple un texte, un son, une image ou une vidéo). Ces données sont initialement représentées par une suite de symboles, qui peuvent être par exemple des lettres (pour un texte) ou des nombres entiers (pour un son, une image ou une vidéo). Nous allons d'abord décrire comment cette succession de symboles est obtenue. Ceci correspond à l'échantillonnage et à la quantification de l'information (la partie (i) évoquée dans le résumé de l'article).

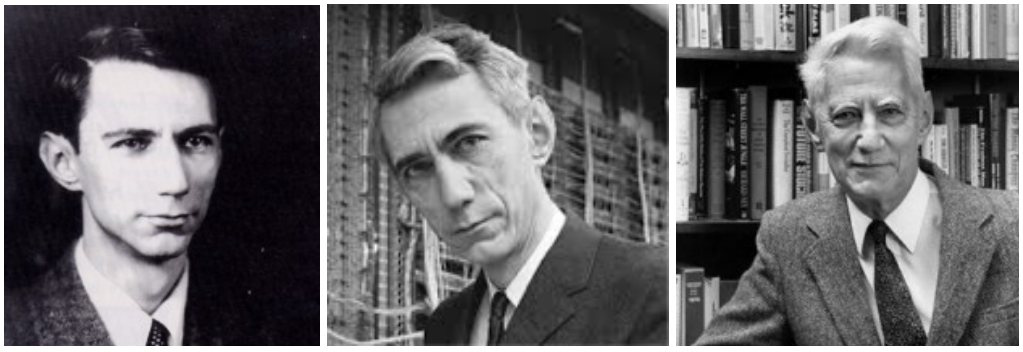


FIGURE 1 – Claude Shannon

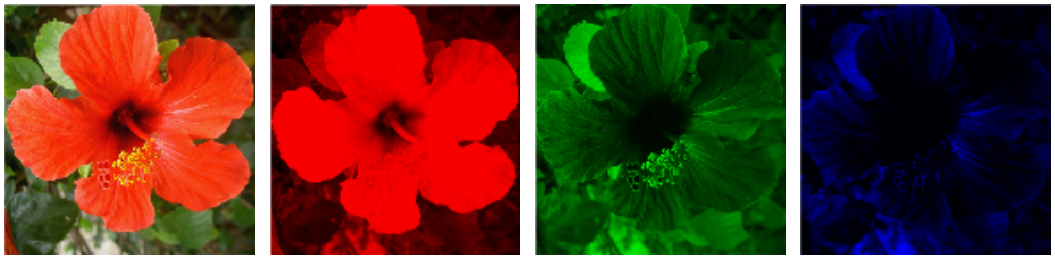


FIGURE 2 – Image couleur

1.1 De l'analogique au numérique

Pour le cas d'un texte, celui-ci est généralement directement transcrit sous forme d'une succession de lettres lorsqu'il est tapé par un être humain. Mais pour les autres sources de données, la situation est moins simple, et ceci fait l'objet de la théorie de l'échantillonnage, qui est l'un des éléments de la théorie de Shannon. Dans la suite de cet exposé, je vais prendre l'exemple des images, mais la plupart des choses que je vais dire s'appliquent à bien d'autres types de données. Par souci de simplicité, je vais considérer uniquement des images en niveaux de gris (images en « noir et blanc »). En général, on décompose en effet une image couleur en trois composantes (rouge, verte et bleu, voir la Figure 2).

La figure 3 montre un exemple d'image avec un zoom sur un contour. Cette image est composée de valeurs $f(x_i)$ qui sont disposées sur une grille carrée régulière de « pixels » $(x_i)_i$. En général, on normalise les valeurs de f de telle sorte que $f(x) \in [0, 1]$. On affiche en noir (respectivement en blanc) un pixel x_i tel que $f(x_i) = 0$ (respectivement $f(x_i) = 1$), et on donne des niveaux de gris intermédiaires pour les autres valeurs (voir aussi la ligne du haut de la figure 4). Je vais maintenant expliquer comment les valeurs de ces pixels sont calculées par un appareil photo.

1.2 L'échantillonnage

La conversion depuis le monde analogique (la lumière émise ou réfléchie par la scène que l'on prend en photo) vers les données numériques s'effectue à l'aide des capteurs CCD de l'appareil photo. Ces capteurs transforment l'énergie lumineuse en un courant électrique. Plus il y a de lumière, plus ce courant est fort. On peut modéliser mathématiquement ce procédé comme *l'échantillonnage* d'une fonction continue $f : x \to f(x) \in \mathbb{R}$ qui représente la quantité de lumière arrivant en un point $x \in \mathbb{R}^2$ du plan focal (qui est bi-

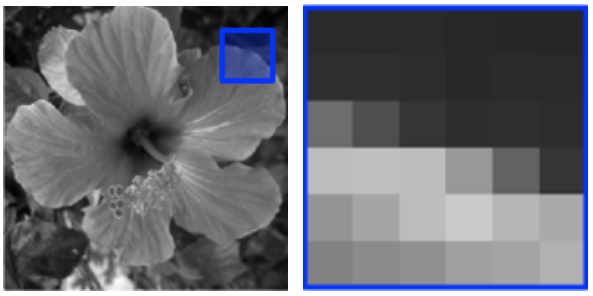


FIGURE 3 – Une image en niveaux de gris et un zoom sur un carré de 6×6 pixels.

dimensionnel) afin d'obtenir un ensemble de N valeurs $(f(x_i))_{i=1}^P$, où P est le nombre de capteurs (donc le nombre de pixels de l'image obtenue) et $(x_i)_{i=1}^P \subset \mathbb{R}^2$ leurs positions. On notera que l'on a choisi un ordre arbitraire pour ces positions $(x_i)_{i=1}^P$, de sorte que $(f(x_i))_{i=1}^P$ est en fait une liste de P valeurs. Le théorème d'échantillonnage de Shannon-Whittaker [7, 8] donne une condition sous laquelle on ne perd pas d'information lors de cet échantillonnage. Ce théorème nécessite la connaissance de l'analyse de Fourier pour être énoncé, mais on peut l'interpréter comme suit : plus la fonction f est *irrégulière*, c'est-à-dire plus ses valeurs varient rapidement, plus le nombre d'échantillons doit être grand. Le théorème d'échantillonnage quantifie de façon précise la régularité de f et relie cette quantité au nombre P minimal que l'on doit choisir. Le résultat énoncé par ce théorème est intuitif : par exemple si l'on veut prendre en photo un objet très net, avec des contours bien définis, alors il faut beaucoup de pixels. En effet, la fonction f correspondant à une telle scène a des variations brusques au niveau de ses contours, et l'on doit donc choisir P très grand, ce qui nécessite une image avec énormément de pixels. Au contraire, si l'on prend en photo un objet très flou, sans contours, alors on peut se limiter à l'utilisation d'une image avec beaucoup moins de pixels.

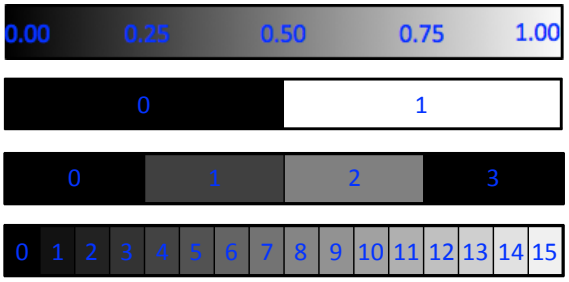
1.3 Quantification

Afin de pouvoir manipuler informatiquement (pour les stocker, les modifier, les transmettre) les P valeurs réelles $f(x_i) \in [0, 1]$ ainsi obtenues après échantillonnage, il faut les approcher à l'aide de nombres entiers, ce qui correspond au processus de *quantification*. On se fixe pour ce faire un nombre entier maximum N, et l'on partage l'intervalle $[0, 1]$ en N sous-intervalle de longueur $1/N$, et l'on choisit de représenter $f(x_i)$ par le symbole $v_i \in \{0, \dots, N-1\}$ si $f(x_i)$ tombe dans le $(v_i)^e$ intervalle, c'est-à-dire que v_i est défini par la condition

$$\frac{v_i}{N} \leqslant f(x_i) < \frac{v_i + 1}{N},$$

avec le cas particulier que $f(x_i) = 1$ est codé par $v_i = N - 1$. Les tableaux de la figure 4 montrent graphiquement cette conversion. Cette étape de quantification produit donc une suite $(v_i)_{i=1}^P$ de P *symboles*, qui sont donc des nombres entiers entre 0 et $N - 1$.

Plus N augmente, plus la représentation numérique à l'aide des nombres entiers est fidèle à l'image analogique de départ. On peut en effet afficher une image en niveaux de gris en donnant la valeur $\frac{v_i}{N-1} \in [0, 1]$ au i^e pixel. La figure (5) montre comment la qualité s'améliore lorsque N augmente, et déjà avec $N = 16$ valeurs on ne discerne que très peu de différences avec l'image originale. Le choix le plus usuel pour le stockage des images est l'utilisation de $N = 256$ niveaux de gris, ce qui donne une qualité visuelle quasi-parfaite.



Réels dans $[0, 1]$

Entiers dans $\{0, 1\}$

Entiers dans $\{0, 1, 2, 3\}$

Entiers dans $\{0, 1, \ldots, 15\}$

FIGURE 4 – Tableaux de quantification des nombres réels dans $[0, 1]$.

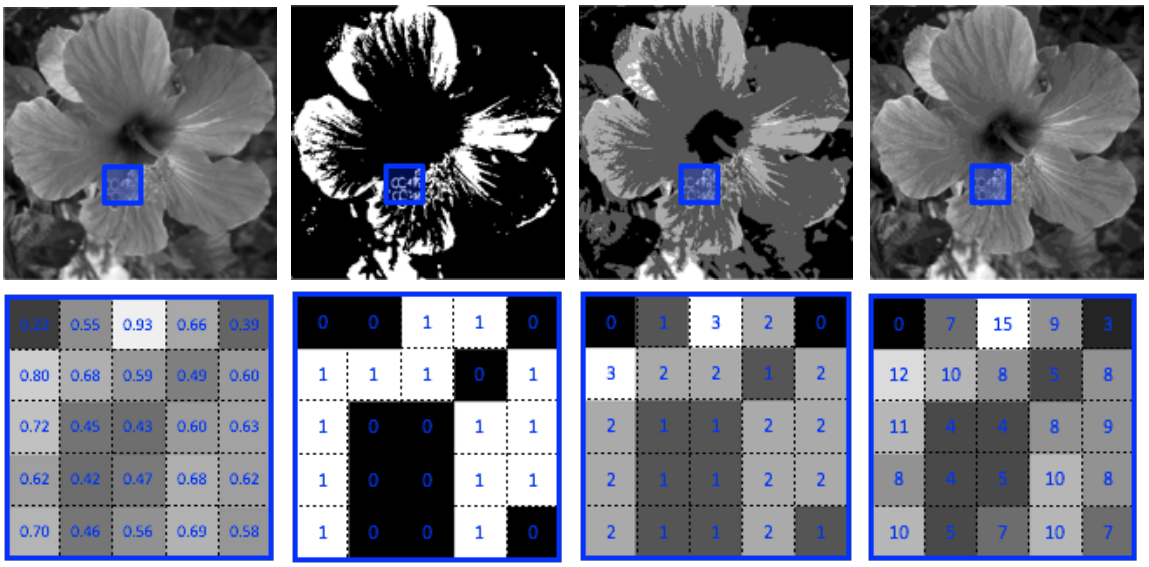


FIGURE 5 – Image non quantifiée (gauche) et quantifiée avec un nombre de bits croissants.

2 Codage et décodage

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. Nous allons maintenant décrire et étudier la transformation (le codage) depuis la suite de symboles vers la suite de 0 et de 1.

2.1 Codage binaire.

L'étape de codage consiste donc à associer à chaque symbole $v \in \{0, \ldots, N-1\}$ une suite notée c_v de 0 et de 1, ce que l'on nomme le code du symbole v. Le codage de la suite de P symboles pour les stocker informatiquement correspond à la transformation

$$(v_1, v_2, \ldots, v_P) \longmapsto (c_{v_1}, c_{v_2}, \ldots, c_{v_P}). \tag{1}$$

La figure 6, gauche, montre le code binaire produit lors du codage d'un ensemble de 5×5 pixels de $N = 4$ symboles, en utilisant le codage

v	c_v
0	00
1	01

v	c_v
2	10
3	11

(2)

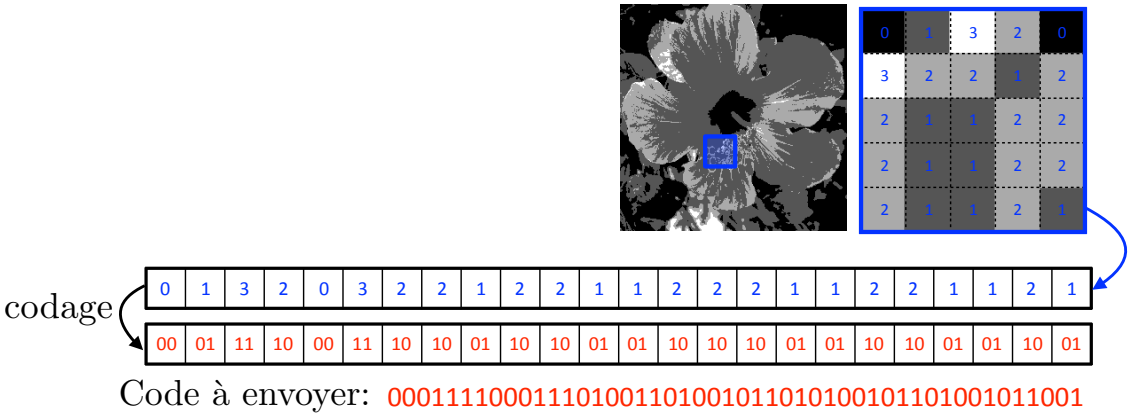


FIGURE 6 – Codage uniforme d'une sous-image de 5×5 pixels.

Il s'agit d'un cas particulier de codage *uniforme*, qui associe à chaque symbole v un code c_v de longueur $L(c_v)$ fixe (ici de longueur constante $L(c_v) = 2$ pour tout v).

2.2 Codage uniforme.

La façon la plus simple de construire un tel code est d'utiliser l'écriture en base 2 de v. Pour ce faire, nous supposons que $N = 2^\ell$ est une puissance de 2, ce que l'on note aussi à l'aide du *logarithme* en base 2 de la façon

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

Dans la suite, on note $[\cdot]_2$ pour indiquer une écriture en base 2. L'écriture en base 2 d'un nombre correspond à la décomposition

$$[a_{\ell-1} \dots a_3 a_2 a_1 a_0]_2 = a_{\ell-1} 2^{\ell-1} + \dots + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0 \in \{0, \dots, 2^\ell - 1\}.$$

Par exemple, pour $N = 8 = 2^3$, on a l'écriture binaire de 6 suivante

$$[110]_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 6.$$

A l'aide de cette écriture binaire, on définit le codage binaire

$$v = [a_{\ell-1} \dots a_3 a_2 a_1 a_0]_2 \longmapsto c_v = a_{\ell-1} \dots a_3 a_2 a_1 a_0.$$

Par exemple, le codage binaire de 6 est 110.

La table (2) donne les codes binaires pour $N = 2$, et les deux tableaux qui suivent donnent les codes pour $N = 4$ et $N = 8$.

v	c_v	v	c_v
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

v	c_v	v	c_v
0	0000	4	0100
1	0001	5	0101
2	0010	6	0110
3	0011	7	0111

v	c_v	v	c_v
8	1000	12	1100
9	1001	13	1101
10	1010	14	1110
11	1011	15	1111

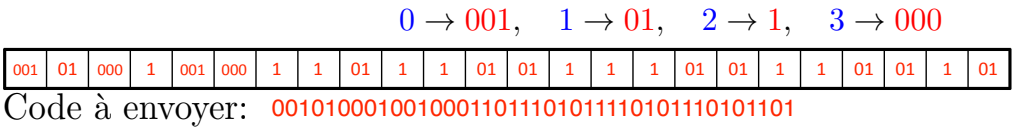


FIGURE 7 – Codage la même suite de symbole qu'à la figure 6, mais avec un code à longueur variable

2.3 Code à longueur variables

Le codage sur $N = 4$ bits des valeurs des $P = 25$ symboles correspondant aux 25 pixels de la figure 6 à l'aide du code uniforme

$$(0,1,2,3) \longmapsto (c_0 = 00, c_1 = 01, c_2 = 10, c_3 = 11)$$

nécessite donc

$$\bar{\mathcal{L}} = P \times \log_2(N) = 50 \text{ bits} \tag{3}$$

où le *bit* est l'unité fondamentale de l'information, et a été introduit principalement par Claude Shannon.

Il s'agit d'un code uniforme, car chaque symbole est codé à l'aide du même nombre $\log_2(N)$ de bits. 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). La figure 7 propose d'utiliser le codage

$$(0,1,2,3) \longmapsto (c_0 = 001, c_1 = 01, c_2 = 1, c_3 = 000).$$

Elle montre qu'on peut donc faire mieux qu'avec un codage uniforme en utilisant un codage variable, qui associe à chaque symbole v un code c_v de longueur $L(c_v)$ variable.

Pour obtenir la longueur $\bar{\mathcal{L}}$ de la suite de bits obtenue, on peut utiliser les fréquences d'apparition des différents symboles, définies par

$$\forall\, v \in \{0,\ldots,N-1\}, \quad p_v \overset{\text{def.}}{=} \frac{1}{P} \sharp \{i \in \{1,\ldots,P\} \;;\; v_i = v\}\,. \tag{4}$$

Ces fréquences vérifient $\sum_{v=0}^{N-1} p_v = 1$, et p_v est d'autant plus grand que le mot v est fréquent dans le message à coder.

Le nombre de bits obtenus avec un code $v \mapsto c_v$ pour coder le message est alors

$$\bar{\mathcal{L}} \overset{\text{def.}}{=} P \times \mathcal{L} \quad \text{où} \quad \mathcal{L} \overset{\text{def.}}{=} \sum_{v=0}^{N-1} p_v \times L(c_v). \tag{5}$$

Par rapport à la formule (3) du code uniforme, on voit que le nombre de bits moyen par symbole est passé de $\log_2(N)$ à $\mathcal{L}$.

Dans le cas de la suite de symboles de la figure 7, on a

v	p_v	$L(c_v)$
0	$\frac{2}{25}$	3
1	$\frac{9}{25}$	2
2	$\frac{12}{25}$	1
3	$\frac{2}{25}$	3

$$\bar{\mathcal{L}} = 2 \times 3 + 9 \times 2 + 12 \times 1 + 2 \times 3 = 42 \text{ bits},$$
$$\mathcal{L} = 42/25 = 1.68 \text{ bits}.$$

On constate donc que le code à longueur variable a permis de gagner de la place de stockage.

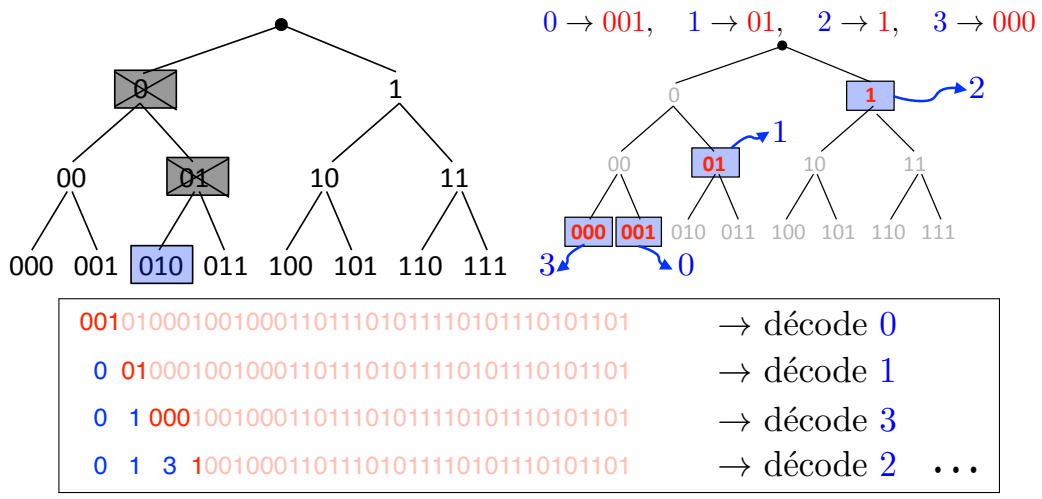


FIGURE 8 – Arbre complet de tous les codes de longueur 3, exemple de codage préfixe et exemple de décodage du code de la figure 7.

2.4 Codage prefixe.

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 est capable de faire le chemin inverse de (1)

$$(c_{v_1}, c_{v_2}, \ldots, c_{v_P}) \longmapsto (v_1, v_2, \ldots, v_P). \tag{6}$$

Tous les codages ne permettent pas de faire ce chemin inverse. Pour les codages uniformes, comme celui utilisé à la figure 6, c'est facile, il suffit de séparer la suite de bits en paquets de longueur $\log_2(N)$ et d'utiliser la table de codage (ici (2)) en sens inverse. Par contre, si l'on considère le codage pour 4 symboles

$$(0, 1, 2, 3) \longmapsto (c_0 = 0, c_1 = 10, c_2 = 110, c_3 = 101)$$

alors la suite de bits 1010 peut correspondre soit à c_1c_1 soit à c_3c_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 code $c_1 = 10$ est le début du mot $c_3 = 101$.

Pour être capable de faire le décodage de façon non-ambigüe, il faut qu'aucun mot c_v du codage ne soit le début d'un autre mot $c_{v'}$. Si ce n'est pas le cas, on parle de codage *préfixe*, et l'on peut donc effectuer progressivement le décodage. Le bas de la figure 8 montre le décodage progressif de ce message.

Comme le montre la figure 8 en bas 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. Les codages préfixes sont alors représentés comme les feuilles des sous-arbres de cet arbre complet. La figure 8 en bas à droite montre à quel sous-arbre correspond le code utilisé à la figure 7.

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 8 en haut illustre le processus de décodage.

3 Borne de l'entropie

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.

3.1 Modélisation aléatoire

Le codage préfixe à longueur variable effectué à la figure 7 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 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 $(v_i)_i$ qui composent la suite sont tirés *indépendamment* selon une variable aléatoire V (la source du message). Ceci signifie que les v_i sont modélisés comme des variables aléatoires indépendantes ayant la même distribution que V.

Cette variable V a pour distribution de probabilité $(p_v)_{v=0}^{N-1}$, c'est à dire que la probabilité que v_i (supposé généré par la source V) soit égal à v 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 $(v_1, \dots, v_P)$, et l'on va définir p_v à l'aide des fréquences empiriques (4). Le modèle sera donc d'autant plus précis que P est grand.

3.2 Entropie

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

$$\mathcal{H}_V \overset{\text{def.}}{=} -\sum_{v=0}^{N-1} p_v \log(p_v), \tag{7}$$

où l'on utilise la convention $0 \log(0) = 0$. Cette convention signifie que l'on ne prend pas en compte les probabilités nulles dans cette formule.

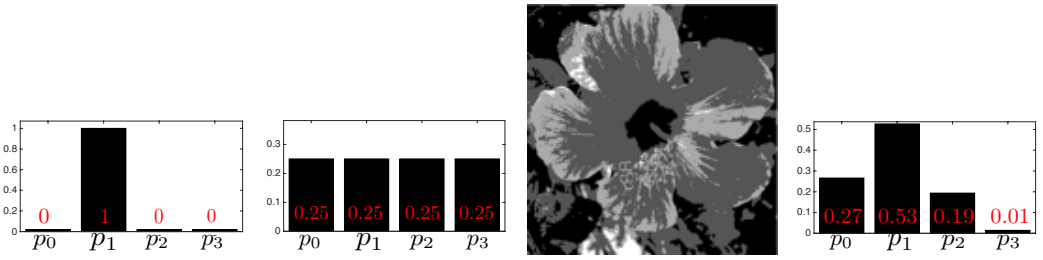
On peut montrer que l'entropie vérifie

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

L'entropie $\mathcal{H}_V = 0$ est minimum lorsque les fréquences p_v sont toutes nulles sauf 1 (figure 9, gauche), ce qui correspond à 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 à la droite de la figure 9) correspondent à des entropies intermédiaires. Intuitivement, l'entropie correspond donc à une mesure du degré d'incertitude d'une distribution.

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

$$\mathcal{L}_V \overset{\text{def.}}{=} \sum_{v=0}^{N-1} p_v \times L(c_V). \tag{8}$$



$\mathcal{H}_V = 0$ $\mathcal{H}_V = \log_2(2) = 1$ Image $\mathcal{H}_V \approx 1.54$

FIGURE 9 – 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.

Il faut bien distinguer le nombre de bits moyen $\mathcal{L}$ défini en (5) 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 v_i sont bien tirés indépendamment selon la même distribution que la source V, alors, selon la loi des grands nombres, on a

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

3.3 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

$$\mathcal{H}_V \leqslant \mathcal{L}_V. \tag{9}$$

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

$$\mathcal{H}_V \leqslant \mathcal{L}_V \leqslant \mathcal{H}_V + 1. \tag{10}$$

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 $L(c_v)$ des mots 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.

3.4 Transformation de l'information

La borne (9) fait l'hypothèse que les symboles $(v_1, v_2, \ldots)$ 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 5. En effet, on voit bien que la valeur

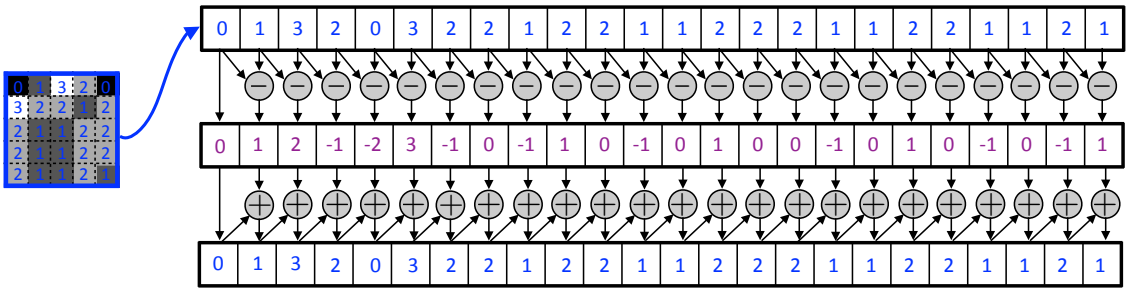


FIGURE 10 – Représentation par différences

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 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 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 10 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

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

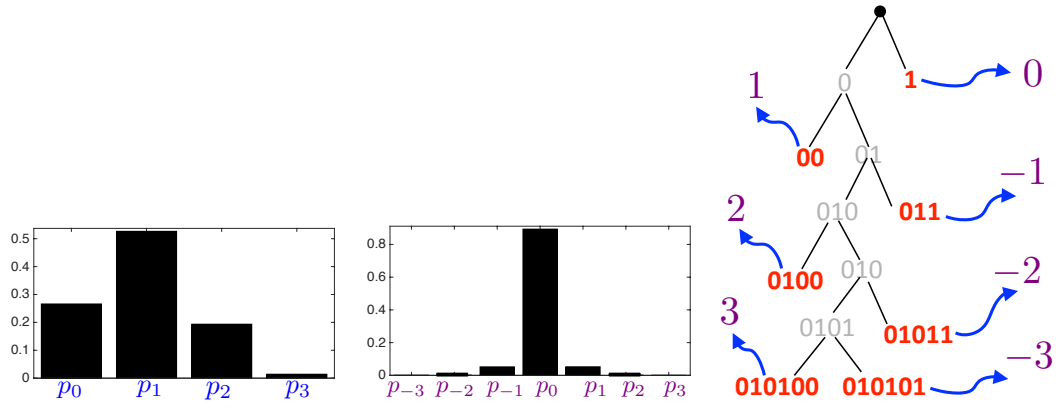
est cruciale pour pouvoir faire le décodage et afficher l'image décodée.

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 11 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 partie droite de la figure 11 montre 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

$$c_{-3} = 010101, c_{-2} = 01011, c_{-1} = 011, c_0 = 1, c_1 = 00, c_2 = 0100, c_3 = 010100.$$

Ce codage a une longueur moyenne $\mathcal{L} \approx 1.16$ bits. Ce nombre moyen est bien conforme à (10), 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



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

FIGURE 11 – Comparaison des histogrammes des valeurs des pixels et des différences, et un arbre de codage de ces différences.

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 montrée à la figure 5, on obtient ainsi les gains suivants

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

où 1 ko = 8×1024 bits est un *kilo octet*.

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.

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 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 x_i sur la grille carrée d'une image, parfois utilisé pour faire référence à la valeur $f(x_i)$ associée.
- **Symbole** : élément v d'un ensemble fini, par exemple $\{0, \ldots, N-1\}$.
- **Code** : mot c_v (succession de 0 et de 1) utilisé pour coder un symbole v.
- **Codage** : l'ensemble des correspondances $(v \mapsto c_v)_v$, fait aussi référence à l'action de remplacer une suite de symboles $(v_i)_i$ par les codes $(c_{v_i})_i$.
- **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** : $\mathcal{H}_V$ définie en (7) 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, voir (5).
- **Nombre de bits moyen de la source :** $\mathcal{L}_V$ est associé au codage des symbole générés par V, voir (8).

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.

[8] J.M. Whittaker. *Interpolatory Function Theory*. Cambridge University Press, London, 1935.