

Compression

Guillaume TOCHON

17 mars 2018

Table des matières

1	Compression de données	1
1.1	Sur la compressabilité	2
1.1.1	Entropie	2
1.1.2	Probabilités	3
1.1.3	Application à l'informatique	3
1.1.4	Cas de l'entropie nulle (À corriger avec le prof)	4
1.1.5	Calcul de l'entropie	4
1.1.6	Exercice	5
2	Algorithmes de compression conservatifs	6
2.1	Run-length encoding	6
2.1.1	Cas du fax	6
2.2	Algorithme d'Huffman	6
2.3	L'algorithme bzip2	7
2.4	Algorithme LZW	7
3	Conversion analogique-digitale	7
3.1	Un petit exemple	7
3.2	Le problème de l'échantillonnage	7
3.3	À quelle vitesse un signal varie-t-il ?	8
3.3.1	La transformée en séries de Fourier. [Compléter]	8
3.3.2	Le produit scalaire	8
3.3.3	Illustration géométrique de la décomposition en série de Fourier	8
3.3.4	La décomposition en série de Fourier	8
3.3.5	La transformée de Fourier	9
3.3.6	Propriétés pratiques du Dirac	9
3.3.7	Après le peigne de Dirac	9

1 Compression de données

Site des slides : www.lrde.epita.fr/~gtochon/CODO/

Ces notes sont à travailler en parallèle avec les slides.

Compression + décompression

Sans compression de données, un film 720p d'une heure ferait presque 400Go.

La naissance naïve de la compression de données remonte aux environs du code morse. Shannon est responsable des fondements mathématiques.

Autres acteurs : - Abraham Lempel - David Huffman - Terry Welch

Le but de la compression est que lors de la décompression, ce soit le moins visible possible par l'utilisateur.

1.1 Sur la compressabilité

1.1.1 Entropie

On va chercher à éliminer les redondances, et les gaspillages. L'outil que l'on va utiliser est l'entropie. Du point de vue de Shannon, plus un message est probable, moins il contient d'informations.

Prenons un alphabet Σ de N symboles :

$$\Sigma = \{s_1, s_2, \dots, s_N\}$$

Avec leurs probabilités d'occurrence respectives :

$$p_i = p(s_i)$$

Évidemment, on a

$$\sum_{i=1}^N p_i = 1$$

Soit F un fichier composé de N_F éléments de Σ .

Statistiquement, s_i est présent $p_i \times N_F$ fois.

On définit q_i la quantité d'information totale d'un symbole :

$$q(s_i) = -\log_2(p_i)$$

On a donc Q_{Tot} la quantité d'information propre totale de s_i contenue dans F , d'unité Sh/symbole, abrégée Sh/symb :

$$Q_{Tot}(s_i) = -N_F \cdot p_i \cdot \log_2(p_i) \text{ Sh/symb}$$

On définit donc $Q_{Tot}(F)$ la quantité d'information contenue dans un fichier F , d'unité Sh :

$$\begin{aligned} Q_{Tot}(F) &= \sum_{i=1}^{N_F} (Q_{Tot}(s_i)) \\ &= \sum_{i=1}^{N_F} -N_F \cdot p_i \cdot \log_2(p_i) \\ &= -N_F \sum_{i=1}^{N_F} p_i \cdot \log_2(p_i) \text{ Sh} \end{aligned}$$

Or ce $-N_F$ devant la somme pose problème : **la quantité d'information dépend de la taille du fichier**. Cela implique qu'un gros fichier "probable" contient moins d'information qu'un petit fichier improbable.

On définit donc l'entropie H ainsi :

$$H = \frac{Q_{Tot}(F)}{N_F} = - \sum_{i=1}^{N_F} p_i \cdot \log_2(p_i)$$

On notera en outre que :

$$H \equiv Sh$$

On remarquera que l'entropie H est **indépendante** de la taille du fichier.

H ne dépend donc que de l'alphabet considéré Σ et de la distribution des symboles qui compose le fichier F .

1.1.2 Probabilités

Soit X une variable aléatoire telle que :

$$X = \{x_1, x_2, \dots, x_n\}$$

Et

$$p_i = P(X = x_i)$$

On a donc l'espérance de X :

$$E(X) = \sum_{i=1}^n x_i \cdot p_i = \sum_{i=1}^n x_i \cdot P(X = x_i)$$

Et on a donc :

$$E(\varphi(X)) = \sum_{i=1}^n \varphi(x_i) P(X = x_i)$$

On a donc, pour l'entropie :

$$\begin{aligned} H &= - \sum_{i=1}^{N_F} p_i \cdot \log_2(p_i) \\ &= \sum_{i=1}^{N_F} \underbrace{(-\log_2(p_i))}_{q_i} \cdot p_i \\ &= \sum_{i=1}^{N_F} q_i \cdot p_i \\ &= E(q(s_i)) \end{aligned}$$

1.1.3 Application à l'informatique

On considère donc :

$$\begin{aligned} N_\Sigma &= \{0, 1\} \\ p(0) &= p_0 = p \\ p(1) &= p_1 = 1 - p \end{aligned}$$

On a alors :

$$\begin{aligned} H &= -p \cdot \log_2(p) - (1-p) \log_2(1-p) \\ &= H(p) \end{aligned}$$

Par convention, on pose :

$$\begin{aligned} H &= - \sum_{i=1}^{N_F} p_i \cdot \log_2(p_i) \\ p_i \cdot \log_2(p_i) &= 0 \quad \text{si} \quad p_i = 0 \end{aligned}$$

1.1.4 Cas de l'entropie nulle (À corriger avec le prof)

$$H(0) = H(1) = 0$$

On a des fichiers complètement désordonnés :

$$p = 0 \implies F = \{1111...1\}$$

$$p = 1 \implies F = \{0000...0\}$$

Donc l'entropie est nulle.

Quand $p = \frac{1}{2}$, les symboles sont équiprobables, le fichier est complètement désordonné, donc complètement aléatoire. Dans ce cas, l'entropie est donc **maximale**.

Voir la figure 1.

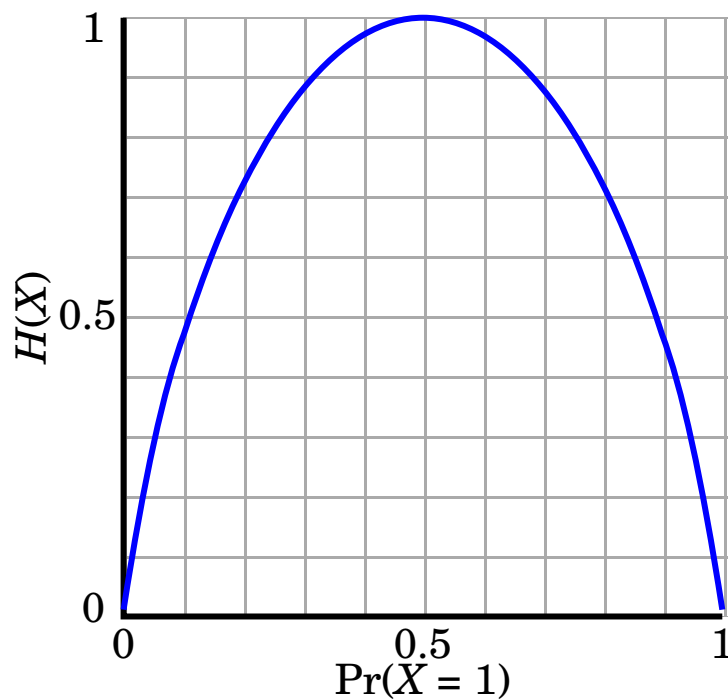


FIGURE 1 – Entropie en fonction de $P(X = 1)$

Attention ! L'entropie ne voit pas les méta-symboles ! Par exemple, ce fichier est considéré comme parfaitement aléatoire :

$$F = \{01010101\}$$

1.1.5 Calcul de l'entropie

Soit l'alphabet

$$\Sigma = \{s_1, s_2, \dots, s_N\}$$

De longueur :

$$\forall n \in \mathbb{N}, \quad N_{\Sigma} = 2^n$$

Avec :

$$\forall s_i \in \llbracket 1, N_\Sigma \rrbracket, \quad p(s_i) = p_i = \frac{1}{N_\Sigma}$$

On a donc :

$$\begin{aligned} H &= - \sum_{i=1}^{N_\Sigma} p_i \cdot \log_2(p_i) \\ &= - \sum_{i=1}^{2^n} \frac{1}{2^n} \cdot \underbrace{\log_2\left(\frac{1}{2^n}\right)}_{-n} \\ &= \frac{1}{2^n} (-n) \underbrace{\sum_{i=1}^{2^n} 1}_{2^n} \\ &= n \end{aligned}$$

1.1.6 Exercice

VÉRIFIER AVEC LE PROF !

De combien de bits a-t-on besoin pour encoder le fichier suivant ?

$$F = \{ACABBDDDBAAABCAAA\}$$

Dans ce cas, on utilisera l'alphabet suivant :

$$\Sigma = \{A, B, C, D\}$$

Comptons le nombre d'occurrences de chacune des lettres de Σ :

$$\left. \begin{array}{l} A : 8 \\ B : 4 \\ C : 2 \\ D : 2 \end{array} \right\} \text{ Longueur : 16}$$

1. Non compressé ?

C'est tout simplement, en notant Σ l'alphabet utilisé pour encoder le fichier,

$$\begin{aligned} \text{Résultat} &= \text{Longueur} \times \text{Nombre de bits nécessaire pour encoder } \Sigma \\ &= 16 \cdot \log_2(n_\Sigma) \\ &= 16 \cdot \log_2(4) \\ &= 32 \text{ bits} \end{aligned}$$

2. Compressé ?

Calculons l'entropie du fichier :

$$\begin{aligned} H &= p_A \cdot \log_2(p_A) + p_B \cdot \log_2(p_B) + p_C \cdot \log_2(p_C) + p_D \cdot \log_2(p_D) \\ &= \frac{1}{2} \cdot \log_2\left(\frac{1}{2}\right) + \frac{1}{4} \cdot \log_2\left(\frac{1}{4}\right) + \frac{1}{8} \cdot \log_2\left(\frac{1}{8}\right) + \frac{1}{8} \cdot \log_2\left(\frac{1}{8}\right) \\ &= \frac{7}{4} \end{aligned}$$

Donc la longueur totale minimale du message est :

$$\begin{aligned}
\text{Résultat} &= \text{Longueur} \times H \\
&= 16 \cdot \frac{7}{4} \\
&= 28 \text{ bits}
\end{aligned}$$

2 Algorithmes de compression conservatifs

2.1 Run-length encoding

Pas génial pour le texte.

Mieux pour les images de synthèse. C'est utilisé dans bitmap.

Le principe c'est de coder combien de fois on voit chaque caractère.

Pour les nombres, caractère spécial.

2.1.1 Cas du fax

C'est utilisé pour le fax et ça marchait bien car il y a beaucoup de blanc.

On peut même compresser la différence entre deux lignes consécutives pour gagner de la place.

C'est une technique très largement utilisée, par exemple dans JPEG et MPEG.

2.2 Algorithme d'Huffman

Un des algorithmes les plus célèbres.

Code entropique : il est basé sur la distribution statistique des symboles.

Deux étapes :

1. Construction d'un arbre binaire donc les feuilles sont les symboles.
2. Encodage du message avec cet arbre.

Première étape :

1. Trier la table
2. Fusionner les deux symboles

Seconde étape :

1. Parcourir l'arbre de la racine vers les veuiulles (les symboles)
2. Toujours assigner 1 au fils gauche et 1 au fils droit (ou l'inverse)
3. Lire l'encodage sur la branche entière.

C'est un encodage à préfixe donc on peut le décoder à la volée. Il est optimal à la longueur d'encodage moyenne parmi les encodages à préfixe. Il nous donne exactement l'entropie tant que que les symboles sont une puissance de 2.

Problème : c'est de l'encodage symbole par symbole.

De nos jours, on ne s'en sert pas tel quel, mais en conjonction avec des algorithmes plus complexes.

2.3 L'algorithme bzip2

Libre et open-source, créé entre 1996 et 2000. Grosse usine à gaz.

L'idée :

1. Appliquer la transformée de Burrows-Wheeler. Cela change l'ordre des caractères pour former des suites de caractères identiques, tout en restant réversible à la décompression.

Encodage :

- Contruire une matrice contenant toutes les rotations de la string à transformer.
- Trier les colonnes selon l'ordre alphabétique.
- Stocker la rotation de la matrice et ces colonnes.

Décodage :

- Concaténer les chaîne avec la dernière colonne de la table.
 - À céompléter
2. La transformée "Move to front"
 3. Algorithme d'Huffman.

Problème : lent à la compression, mais très économe en place. Très efficace pour langage naturel, pas trop pour du code.

2.4 Algorithme LZW

On a intérêt à le refaire sur papier pour bien le comprendre.

C'est le premier algorithme, avec LZ78, qui fasse de l'apprentissage de dictionnaire non-supervisé.

Voir le processus sur les diapos.

C'est utilisé dans GIF.

3 Conversion analogique-digitale

3.1 Un petit exemple

On prend un fichier audio que l'on ouvre avec Audacity pour voir que le fichier ressemble toujours à un entité continue de loin mais discrète de près.

C'est une pression qui dépend de l'espace et du temps.

On considère notre tension avec la variable $f_a : \mathbb{R} \rightarrow \mathbb{R}$.

Que l'on va discrétiser ainsi : $f_d : \mathbb{Z} \rightarrow \mathbb{R}$.

3.2 Le problème de l'échantillonnage

Voir slides.

On prend des points $(t_n)_{n \in \mathbb{Z}}$ pour faire la séquence

$$(f_d(n) = f_a(t_n))_{n \in \mathbb{Z}}$$

Pour faire simple, on prend des points à un intervalle régulier. On a donc

$$t_n = nT_e$$

T_e : Période d'échantillonnage

$f_e = \frac{1}{T_e}$: Fréquence d'échantillonnage

Il faut la choisir avec soin pour avoir un bon compromis taille/qualité.

3.3 À quelle vitesse un signal varie-t-il ?

On va faire une représentation fréquentielle puisque la représentation temporelle ne convient pas. Cette représentation est une **transformée de Fourier**.

Elle prend beaucoup moins de place et représente la même information.

Comment y arrive-t-on ?

3.3.1 La transformée en séries de Fourier. [Compléter]

Elle s'écrit :

$$c_n = \frac{1}{T} \int_0^T x(t) e^{-2i\pi \frac{n}{T}t} dt$$

Elle s'applique pour des raisons physiques.

Prenons un exemple simple :

Soit une fonction carrée de période T :

$$f(t) = \begin{cases} 1 & \text{si } t \in [0, \frac{T}{2}] \\ 0 & \text{si } t \in [\frac{T}{2}, T] \end{cases}$$

On a

$$\tilde{f} = \frac{1}{2} + \sum_{n=1}^N$$

3.3.2 Le produit scalaire

Voir le poly.

Le produit scalaire $\langle x, y \rangle$ entre deux éléments $(x, y) \in E \times E$

Dans $E = \mathbb{R}^2$:

$$\langle x, y \rangle = \|x\| \|y\| \cos \theta$$

Dans $E = \mathbb{R}^n$:

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \sum_{i=1}^n x_i \cdot u_i = \sum_{i=1}^n$$

Dans $E = L$:

3.3.3 Illustration géométrique de la décomposition en série de Fourier

3.3.4 La décomposition en série de Fourier

$$\begin{aligned} f(t) &= \sum_{-\infty}^{+\infty} c_n e^{i2\pi \frac{n}{T}t} \\ &= a_0 + \sum_{n=1}^{+\infty} a_n \cos(2\pi \frac{n}{T}t) + b_n \cos(2\pi \frac{n}{T}t) \end{aligned}$$

3.3.5 La transformée de Fourier

$$c_n = \frac{1}{T} \int_0^T f(t) e^{-j 2\pi n t / T} dt$$

On a

$$\mathcal{F}(f \times g) = \mathcal{F}(f) * \mathcal{F}(g)$$

3.3.6 Propriétés pratiques du Dirac

$$f(t) \times \delta(t - t_0) = f(t_0) \times \delta(t - t_0)$$

C'est l'élément neutre du produit de convolution.

$$f(t) = \delta(t - t_0) = f(t - t_0)$$

Voir le reste sur les diapos.

3.3.7 Après le peigne de Dirac

$$\tilde{f}_d = f_a \times W_{T_e}(t)$$

$$\begin{aligned} \mathcal{F}(\tilde{f}_d) &= \mathcal{F}(f_a \times W_{T_e}) \\ &= \underbrace{\mathcal{F}(f_a)}_{\tilde{f}_a(u)} \times \mathcal{F}(W_{T_e}) \\ &= \end{aligned}$$

3.3.8 Effet stroboscopique