Algorithmique avancée: Compression de données

Serge Haddad

LMF, ENS Paris-Saclay & CNRS & INRIA

L3

Plan

L'entropie

Soit un événement Ev de probabilité p de réalisation.

Quelle est la valeur $v(p)$ de l'information de sa réalisation (dans le futur) ?

Soit Ev' de probabilité p' indépendant de Ev. Alors on vend $Ev \wedge Ev'$ pour $v(pp')$ ou Ev puis Ev' pour $v(p) + v(p')$. D'où $v(pp') = v(p) + v(p')$.

Choisissons $v(\frac{1}{2}) = 1$. Alors : $\boxed{v(x) = -\log_2(x)}$

Preuve.

- soit $f(x) = v(2^{-x})$. Alors $f(1) = 1$, $f(x+y) = f(x) + f(y)$;
- pour tout n, $f(n) = nf(1) = n$;
- pour tout $\frac{p}{q}$, $qf(\frac{p}{q}) = f(p)$ d'où $f(\frac{p}{q}) = \frac{p}{q}$;
- f croissante et $\mathbb{Q}$ dense dans $\mathbb{R}$ implique pour tout x, $f(x) = x$.

- Soit X la variable aléatoire associée à la réalisation de Ev.
  Alors son *entropie* $H(X) = -p\log_2(p) - (1-p)\log_2(1-p)$.
  Soit X à valeurs dans $\{x_i\}_{i \leq n}$ de distribution p : $p(x_i) = p_i$.
  Alors : $\boxed{H(p) = H(X) = -\sum_{i \leq n} p_i \log_2(p_i) = -\mathbf{E}(\log_2(p(X)))}$
- Soit $D \in \mathbb{N}$ avec $D \geq 2$, alors $H_D(p) = H_D(X) = -\mathbf{E}(\log_D(p(X)))$.

D'autres entropies

Soit un couple de v.a. (X, Y) avec $p_{i,j} = \mathbf{Pr}(X = x_i \wedge Y = y_i)$ et $p_{i,*} = \sum_j p_{i,j}$.
L'*entropie conditionnelle* est définie par $H(Y|X) = \sum_i p_{i,*} H(Y|X = x_i)$.

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

Preuve.

$$\begin{aligned} H(X,Y) &= -\sum_{i,j} p_{i,j} \log(p_{i,j}) \\ &= -\sum_{i,j} p_{i,j} \log\left(\frac{p_{i,j}}{p_{i,*}}\right) - \sum_{i,j} p_{i,j} \log(p_{i,*}) \\ &= -\sum_i p_{i,*} \sum_j \frac{p_{i,j}}{p_{i,*}} \log\left(\frac{p_{i,j}}{p_{i,*}}\right) - \sum_i p_{i,*} \log(p_{i,*}) = H(Y|X) + H(X) \end{aligned}$$

Soit $p = \{p_i\}_{i \leq n}$ et $p' = \{p'_i\}_{i \leq n}$ des distributions.
L'*entropie relative* (divergence de Kullback-Leibler) est définie par :

$$D(p||p') = \sum_{i \leq n} p_i \log(\tfrac{p_i}{p'_i})$$

$$\boxed{D(p||p') \geq 0 \text{ et } D(p||p') = 0 \text{ ssi } p = p'}$$

Preuve. Par convexité $-\sum_i p_i \log(\frac{p'_i}{p_i}) \geq -\log(\sum_i p_i \frac{p'_i}{p_i}) = 0$
Par stricte convexité, l'égalité n'a lieu que si pour tout i, $p_i = p'_i$.

L'information mutuelle

Soit (X, Y) un couple de v.a. de distributions $p = \{p_{i,j}\}_{i,j}$
avec $p^X = \{\sum_j p_{i,j}\}_i$ et $p^Y = \{\sum_i p_{i,j}\}_j$.

p^{XY} est la distribution définie par $p^{XY}_{i,j} = p^X_i p^Y_j$.

L'*information mutuelle* de X et Y est définie par :

$$I(X;Y) = D(p||p^{XY})$$

$$\boxed{I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)}$$

Preuve.

$$\begin{aligned}
I(X;Y) &= \sum_{i,j} p_{i,j} \log(\frac{p_{i,j}}{p^X_i p^Y_j}) \\
&= -\sum_{i,j} p_{i,j} \log(p^X_i) + \sum_{i,j} p_{i,j} \log(\frac{p_{i,j}}{p^Y_j}) \\
&= -\sum_i p^X_i \log(p^X_i) + \sum_j p^Y_j \sum_i \frac{p_{i,j}}{p^Y_j} \log(\frac{p_{i,j}}{p^Y_j}) \\
&= H(X) - H(X|Y)
\end{aligned}$$

Par conséquent, $H(Y) \geq H(Y|X)$ avec égalité ssi X et Y sont indépendantes.

Maximisation de l'entropie

Soit X une v.a. de distribution $\{p_i\}_{i\leq n}$.

$$\boxed{H(X) \text{ est maximal ssi } X \text{ est équirépartie.}}$$

Preuve. Soit Y une v.a. équirépartie de distribution u. Alors :

$0 \leq D(p||u) = \sum_{i\leq n} p_i \log(p_i n) = \sum_{i\leq n} p_i \log(p_i) + \log(n) = H(Y) - H(X)$

Soit X une v.a. de distribution $\{p_i\}_{i\in\mathbb{N}}$ et de moyenne $\mu = \frac{p}{1-p}$ avec $0 \leq p < 1$.

$$\boxed{H(X) \text{ est maximal ssi } X \text{ a une distribution géométrique } p_i = p^i(1-p).}$$

Preuve. Soit Y de distribution géométrique $g_i = p^i(1-p)$.

$$\begin{aligned}
H(X) = & -\sum p_i \log(p_i) = & -\sum p_i \log(\frac{p_i}{g_i} g_i) \\
= & -D(p||g) - \sum p_i \log(g_i) \leq & -\sum p_i \log(p^i(1-p)) \\
= & -\sum i p_i \log(p) - \log(1-p) = & -\sum i g_i \log(p) - \log(1-p) \\
= & -\sum g_i \log(p^i(1-p)) = & H(Y)
\end{aligned}$$

Dans ce cas $H(X) = (\mu+1)\log(\mu+1) - \mu\log(\mu)$.

Code

- Soit $\mathcal{X}$ un ensemble fini, Σ un alphabet et C une fonction de $\mathcal{X}$ dans Σ^*.
  On note $\ell_x = |C(x)|$. C s'étend à $\mathcal{X}^*$ par : $C(x_1 \dots x_k) = C(x_1) \dots C(x_k)$.
  Soit $\sigma \in \mathcal{X}^*$, on note $\ell_\sigma = |C(\sigma)|$.
- C est un *code* si C est injective.
  C est préfixe si pour tout $x \neq y \in \mathcal{X}$, $C(x)$ n'est pas un préfixe de $C(y)$.

Si C est préfixe alors C est un code.

Preuve. *(par l'absurde)*
Soit $u = x_1 \dots x_m$ et $v = y_1 \dots y_n$ tels que :
$u \neq v$, $C(u) = C(v)$ et $|u| + |v|$ soit minimal.
Puisque C est préfixe, $0 < m, n$ et $x_1 = y_1$ et donc $C(x_2 \dots x_m) = C(y_2 \dots y_n)$.

- Soit $p = \{p_x\}_{x \in \mathcal{X}}$ une distribution. Alors la *longueur* (moyenne vis à vis de p) de C est définie par :

$$\ell_p(C) = \sum_{x \in \mathcal{X}} p_x \ell_x$$

Théorème de McMillan (1)

Soit $D = |\Sigma|$. Tout code C vérifie l'inégalité de Kraft $\sum_{x\in\mathcal{X}} D^{-\ell_x} \leq 1$
Soit $\{\ell_x\}_{x\in\mathcal{X}}$ vérifiant l'inégalité de Kraft.
Alors il existe C code préfixe tel que pour tout x, $|C(x)| = \ell_x$.

Preuve. Soit $k \in \mathbb{N}$.

$\left(\sum_{x\in\mathcal{X}} D^{-\ell_x}\right)^k = \sum_{x_1\in\mathcal{X}} \cdots \sum_{x_k\in\mathcal{X}} D^{-\ell_{x_1}} \ldots D^{-\ell_{x_k}} = \sum_{\sigma\in\mathcal{X}^k} D^{-l_\sigma}$

Soit $a(m) = |\{\sigma \in \mathcal{X}^k \mid l_\sigma = m\}|$ et $\ell_{\max} = \max(\ell_x \mid x \in \mathcal{X})$.

$$\sum_{\sigma\in\mathcal{X}^k} D^{-\ell(\sigma)} = \sum_{m=1}^{k\ell_{\max}} a(m)D^{-m} \leq \sum_{m=1}^{k\ell_{\max}} D^m D^{-m} = k\ell_{\max}$$

Par conséquent : $\forall k > 0 \ \sum_{x\in\mathcal{X}} D^{-\ell(x)} \leq (k\ell_{\max})^{\frac{1}{k}}$

En passant à la limite lorsque k tend vers l'infini, on obtient :

$$\sum_{x\in\mathcal{X}} D^{-\ell(x)} \leq 1$$

Théorème de McMillan (2)

Preuve (suite). Soit $\mathcal{X} = \{x_1, \ldots, x_n\}$ avec pour tout $i < n$, $\ell_{x_i} \leq \ell_{x_{i+1}}$.
Soit $\Sigma = \{0, 1, \ldots, D-1\}$. On définit les mots du code par induction :

- $C(x_1) = 0^{\ell_{x_1}}$;
- Soit $C(x_i) = \alpha_1 \ldots \alpha_{\ell(x_i)}$ et soit $j \leq \ell_{x_i}$ le plus grand indice tel que $\alpha_j < D-1$ alors $C(x_{i+1}) = \alpha_1 \ldots (\alpha_j + 1)0^{\ell_{x_{i+1}} - j}$.

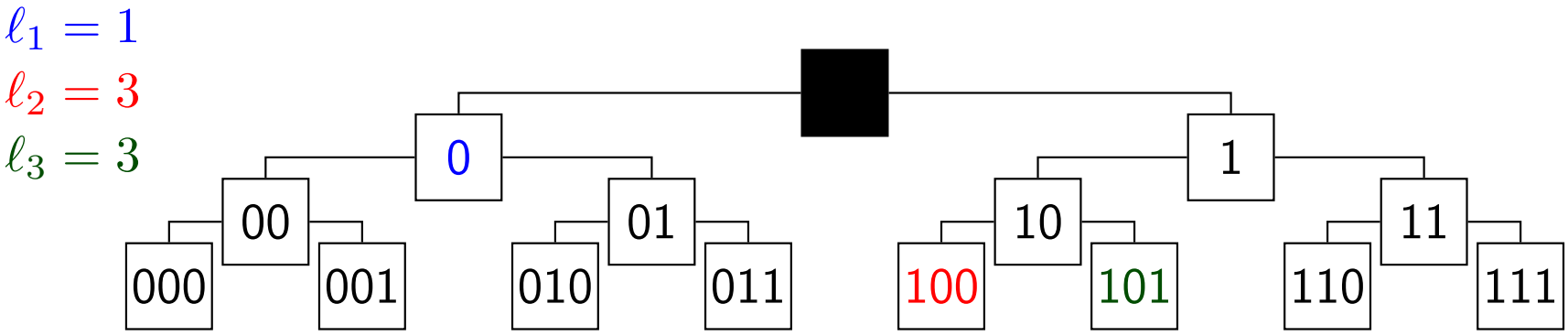
Si cette construction s'achève, montrons par induction que C est un code préfixe.

L'hypothèse d'induction affirme qu'après la $i$ème itération, pour tout $a < b \leq i$:

1. $C(x_a) \leq_{lex} C(x_b)$ où $\leq_{lex}$ désigne l'ordre lexicographique ;
2. $\exists u \leq \ell(x_a)\ C(x_a)[u] < C(x_b)[u]$.

- Considérons l'itération $i+1$ et soit $a \leq i$. Par construction, $C(x_i) \leq_{lex} C(x_{i+1})$ ce qui établit 1. : $C(x_a) \leq_{lex} C(x_{i+1})$ puisque $\leq_{lex}$ est un ordre.
- $C(x_i)[j] < C(x_{i+1})[j]$ où j est l'indice utilisé dans la construction.
  Soit $a < i$, $\exists u \leq l(x_a)\ C(x_a)[u] < C(x_i)[u]$ et soit le plus petit u.
  Puisque $C(x_a) \leq_{lex} C(x_i)$, pour tout $s < u$, $C(x_a)[s] = C(x_i)[s]$.
  Si $u \leq j$, alors $C(x_a)[u] < C(x_i)[u] \leq C(x_{i+1})[u]$.
  Si $u > j$, alors $C(x_a)[j] = C(x_i)[j] < C(x_{i+1})[j]$. Ce qui établit 2.

Théorème de McMillan (3)



Preuve (fin). Il reste à vérifier qu'on a pu trouver un tel j à chaque étape.

On identifie $\bigcup_{i=0}^{\ell_{max}} \Sigma^i$ à l'arbre de hauteur $\ell_{\max}$ où chaque sommet interne a D fils.

Un mot $\alpha_1 \dots \alpha_k$ correspond au sommet atteint en suivant depuis la racine le fils α_1 puis le fils α_2, etc. Le nombre de feuilles de cet arbre est $D^{\ell_{\max}}$.

Un sommet à hauteur h a $D^{\ell_{max}-h}$ feuilles dans son sous-arbre.

Les sommets associés au code préfixe ne partagent pas leurs feuilles mais les feuilles couvertes par les codes construits sont contiguës.

Supposons que la construction se bloque avec $C(x_i) = (D-1)^{\ell_{x_i}}$ et $i < n$.

Toutes les feuilles sont couvertes. Donc : $\sum_{j \leq i} D^{\ell_{max}-\ell_{x_j}} = D^{\ell_{max}}$
Puisque cette somme est partielle, cela contredit l'inégalité de Kraft.

Généralisation du théorème

Soit $\mathcal{X} = \{x_n\}_{n\in\mathbb{N}}$ et $D = |\Sigma|$.
Tout code C vérifie l'inégalité de Kraft $\sum_{x\in\mathcal{X}} D^{-\ell_x} \leq 1$.
Soit $\{\ell_x\}_{x\in\mathcal{X}}$ vérifiant l'inégalité de Kraft.
Alors il existe C code préfixe tel que pour tout x, $|C(x)| = \ell_x$.

Preuve.

Pour tout n, on a $\sum_{m\leq n} D^{-\ell_{x_m}} \leq 1$. Le résultat s'obtient par passage à la limite.

La « construction » du code préfixe est aussi valide pour le cas infini
avec une preuve presque identique : $\ell_{\max}$ devient ℓ_{x_n}
où n est le dernier code construit avant blocage.

Entropie et longueur du code

Soit $D = |\Sigma|$, une distribution $p = \{p_x\}_{x \in \mathcal{X}}$ et C un code. Alors :

$$L_p(C) \geq H_D(p)$$

De plus, il y a égalité ssi $D^{-\ell_x} = p_x$ pour tout $x \in \mathcal{X}$.

Preuve. Soit la distribution $r = \{\frac{D^{-\ell_x}}{\sum_{x \in \mathcal{X}} D^{-\ell_x}}\}_{x \in \mathcal{X}}$ et $c = \sum_{x \in \mathcal{X}} D^{-\ell_x} \leq 1$.

$$\begin{aligned}
L_p(C) - H_D(p) = & \sum_{x \in \mathcal{X}} p_x \ell_x + \sum_{x \in \mathcal{X}} p_x \log_D(p_x) \\
= & -\sum_{x \in \mathcal{X}} p_x \log_D(D^{-\ell_x}) + \sum_{x \in \mathcal{X}} p_x \log_D(p_x) \\
= & \sum_{x \in \mathcal{X}} p_x \log_D(\frac{p_x}{D^{-\ell_x}}) = \sum_{x \in \mathcal{X}} p_x \log_D(\frac{p_x}{r_x}) + \log_D(\frac{1}{c}) \\
= & D(p \parallel r) + \log_D(\frac{1}{c}) \geq 0
\end{aligned}$$

Pour l'égalité il faut que $p = r$ et $c = 1$, ce qui établit pour tout x, $D^{-\ell_x} = p_x$.

Plan

Le codage de Shannon-Fano

Posons $\ell_x = \lceil -\log_D(p_x) \rceil$. $\quad \sum_{x\in\mathcal{X}} D^{-\ell_x} = \sum_{x\in\mathcal{X}} D^{-\lceil -\log_D(p_x)\rceil} \leq \sum_{x\in\mathcal{X}} D^{\log_D(p_x)} = 1$

Le codage de Shannon-Fano est obtenu par la procédure du théorème de McMillan.

- Encadrement de la longueur du code de Shannon-Fano.

$$H_D(p) \leq L_p(C) = \sum_{x\in\mathcal{X}} p_x \ell_x = \sum_{x\in\mathcal{X}} p_x \lceil -\log_D(p_x)\rceil < \sum_{x\in\mathcal{X}} p_x(-\log_D(p_x)+1) = H_D(p)+1$$

- Soit C_k le codage de Shannon-Fano appliqué à des blocs de k événements :

$$H_D(X_1,\ldots,X_k) \leq L_{p^k}(C_k) < H_D(X_1,\ldots,X_k)+1$$

où les $X_1,\ldots,X_k$ sont des v.a. indépendantes de distribution p.

Par conséquent $H_D(X_1,\ldots,X_k) = H_D(X_1)+\ldots+H_D(X_k) = kH_D(p)$

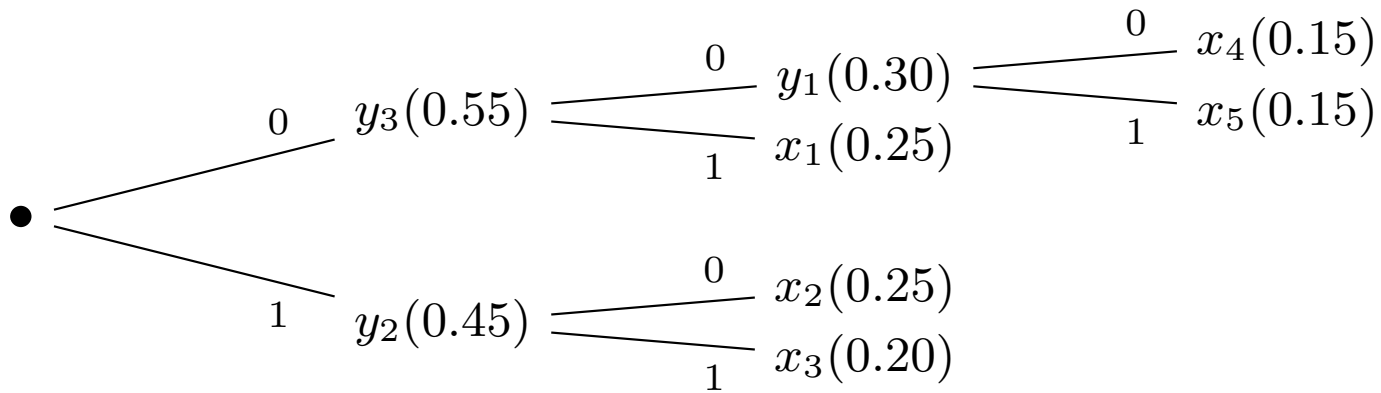
D'où $H_D(p) \leq \frac{L_{p^k}(C_k)}{k} < H_D(p)+\frac{1}{k}$ et $\lim_{k\to\infty} \frac{L_{p^k}(C_k)}{k} = H_D(p)$.

Cette suite de codes est donc asymptotiquement optimale ...

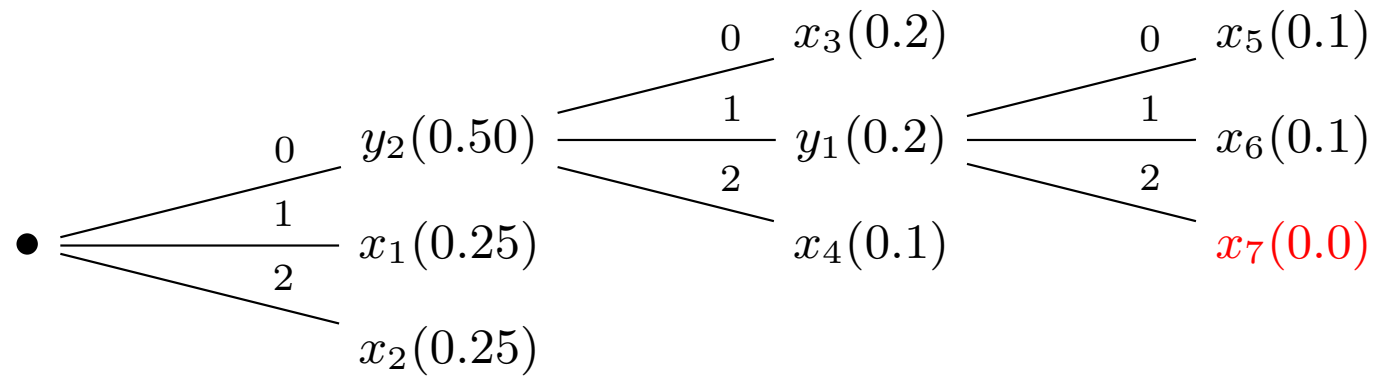
mais la taille de la table de codage croît exponentiellement en fonction de k.

Le codage de Huffman : illustrations

- Soit $D = 2$ et $\mathcal{X} = \{x_1, \ldots, x_5\}$
  avec $p_{x_1} = p_{x_2} = 0.25$, $p_{x_3} = 0.2$ et $p_{x_4} = p_{x_5} = 0.15$.



- Soit $D = 3$ et $\mathcal{X} = \{x_1, \ldots, x_6\}$
  avec $p_{x_1} = p_{x_2} = 0.25$, $p_{x_3} = 0.2$ et $p_{x_4} = p_{x_5} = p_{x_6} = 0.1$.



L'algorithme de Huffman

H est un tas binaire (pour minimum).

$Fils[i, j]$ est le $j$ème fils de i $(0 \leq j < D)$.

For i **from** 1 **to** n **do** $\mathbf{AjoutTas}(H, i, p_i)$
While $n \mod (D-1) \neq 1$ **do** $n \leftarrow n+1$; $\mathbf{AjoutTas}(H, (n, 0))$
$ind \leftarrow n$
While $\mathbf{Taille}(H) > 1$
  $ind \leftarrow ind + 1$
  $sp \leftarrow 0$
  For i **from** 0 **to** $D-1$ **do** $(j, p) \leftarrow \mathbf{Extraire}(H)$; $Fils[ind, i] \leftarrow j$; $sp \leftarrow sp + p$
  $\mathbf{AjoutTas}(H, (ind, sp))$
$C[ind] \leftarrow \varepsilon$
For i **from** ind **downto** $n+1$ **do**
  For j **from** 0 **to** $D-1$ **do** $C[Fils[i, j]] \leftarrow C[i] \cdot j$

L'algorithme opère en $\Theta(n \log(n))$.

Optimalité du code (1)

Soit un arbre $\mathcal{A}$ dont les sommets internes ont $D > 1$ fils.
Alors n_f, le nombre de feuilles de $\mathcal{A}$, vérifie : $n_f \mod D - 1 = 1$

Preuve par récurrence sur le nombre de sommets. Cas de base immédiat.

La racine de l'arbre $\mathcal{A}$ a D fils et notons $n_1, \ldots, n_d$ le nombre de feuilles des sous-arbres issus de ces fils.

$n_i \mod D - 1 = 1$. D'où $n_f \mod D - 1 = \sum_i n_i \mod D - 1 = D \mod D - 1 = 1$.

Soient $p_1 \geq p_2 \geq \ldots \geq p_n$ avec $n \mod (D-1) = 1$.
Alors il existe un code préfixe optimal C qui vérifie :

1. Si $p_i > p_j$ alors $\ell_i \leq \ell_j$ (avec $\ell_i = |C(x_i)|$) ;
2. Les D mots les plus longs ont la même longueur ;
3. D mots associés aux probabilités les plus faibles partagent le même père.

Preuve. Supposons 1. ne soit pas vérifiée.

Soit C' obtenu en échangeant les codes de x_i et x_j.

$$L(C) - L(C') = (p_i - p_j)\ell_i + (p_j - p_i)\ell_j = (p_i - p_j)(\ell_i - \ell_j) > 0$$

Ce qui contredit l'optimalité.

Optimalité du code (2)

Preuve (suite). Soit k le nombre de mots les plus longs. Supposons que $k < D$.
On peut regrouper ces mots sous un même père sans changer $\ell(C)$
et supprimer les pères sans fils.

- Si $k = 1$ on « remonte » l'unique mot le plus long.
- Si $1 < k < D$.

Considérons l'arbre dans lequel les mots les plus longs ont été supprimés.
Soit n' son nombre de feuilles.
$n' \mod (D-1) = n - k + 1 \mod (D-1) = 2 - k \mod (D-1) \neq 1$.
Par conséquent, l'un des autres sommets internes a moins de D fils.
On peut donc « remonter » l'un des mots les plus longs.

Ces remontées ne pouvant être effectuées indéfiniment,
on obtient un code optimal avec $k \geq D$ ce qui établit 2.

Puisque D mots associés aux probabilités les plus faibles ont même longueur,
on peut par échange ou replacement garantir
que ces D mots ont le même père sans changer $\ell(C)$.

Optimalité du code (3)

Le code de Huffman est optimal.

Considérons un code optimal préfixe C qui vérifie les hypothèses précédentes.

On peut construire un code préfixe C' pour la distribution $p_1, \ldots, p_{n-D}, \sum_{n-D<i\leq n} p_i$

- en supprimant les D événements associés à $\{p_i\}_{n-D<i\leq n}$ partageant le même père ;
- en associant ce père l'événement de probabilité $\sum_{n-D<i\leq n} p_i$.

Soit un code préfixe C'_1 pour $p_1, \ldots, p_{n-D}, \sum_{n-D<i\leq n} p_i$, on peut fabriquer un code préfixe C_1 pour $p_1, \ldots, p_n$

- en remplaçant la feuille associée à l'événement de probabilité $\sum_{n-D<i\leq n} p_i$ par un noeud interne ;
- en lui associant comme fils les événements associés à $\{p_i\}_{n-D<i\leq n}$.

$$\ell(C) = \ell(C') + \sum_{n-D<i\leq n} p_i \text{ et } \ell(C_1) = \ell(C'_1) + \sum_{n-D<i\leq n} p_i$$

Par conséquent C' est un code optimal.

On peut alors transformer C' pour qu'il vérifie les hypothèses précédentes.

L'algorithme de Huffman consiste exactement en ce procédé itératif.

Le codage de Shannon-Fano-Elias

Soit $\Sigma = \{1, 2, \ldots, n\}$ avec pour tout i, $p_i > 0$ et $D = \{0, 1\}$.

$\lfloor z \rfloor_\ell$ désigne la troncature de $z < 1$ à ses ℓ premiers bits après la virgule.
Soit $F_i = \sum_{j \leq i} p_j$ et $\bar{F}_i = \sum_{j<i} p_j + \frac{1}{2} p_i$. Soit $\ell_i = \lceil -\log(p_i) \rceil + 1$.

$$0 \leq \bar{F}_i - \lfloor \bar{F}_i \rfloor_{l_i} < 2^{-\ell_i} \leq 2^{\log(p_i)-1} \leq \frac{p_i}{2}$$

Par conséquent, $\lfloor \bar{F}_i \rfloor_{\ell_i} \in]F_{i-1}, \bar{F}_i]$ (avec $F_0 = 0$).

$C(i)$ correspond aux ℓ_i premiers bits de la partie fractionnaire de $\bar{F}_i$
et peut être identifié à $\lfloor \bar{F}_i \rfloor_{\ell_i}$.

Soit $z_1 z_2 \ldots z_\ell$ un mot du code.
On lui associe l'intervalle de $[0, 1]$: $[0.z_1 z_2 \ldots z_\ell, 0.z_1 z_2 \ldots z_\ell + 2^{-\ell}[$.

Le code est préfixe ssi les intervalles associés sont disjoints. Or :

$$\lfloor \bar{F}_i \rfloor_{\ell_i} + 2^{-\ell_i} \leq \bar{F}_i + 2^{-\ell_i} \leq F_i < \lfloor \bar{F}_{i+1} \rfloor_{\ell_{i+1}}$$

La majoration de la longueur du code est immédiate.

$$\ell(C) = \sum_i p_i (\lceil -\log(p_i) \rceil + 1) < H(p) + 2$$

Illustration

i	p_i	F_i	$\bar{F}_i$	$\bar{F}_i$ en binaire	ℓ_i	$C(i)$
1	0.25	0.25	0.125	0.001	3	001
2	0.25	0.50	0.375	0.011	3	011
3	0.20	0.70	0.600	$0.1(0011)^\omega$	4	1001
4	0.15	0.85	0.775	$0.110(0011)^\omega$	4	1100
5	0.15	1.00	0.925	$0.111(0110)^\omega$	4	1110

$(0011)^\omega = \frac{3}{16} \sum_{i \in \mathbb{N}} \left(\frac{1}{16}\right)^i = \frac{3}{16} \frac{16}{15} = \frac{1}{5}$.

Par conséquent $(0110)^\omega = \frac{2}{5}$.

Le codage arithmétique : encodage

Le codage arithmétique est le codage de Shanon-Fano-Elias appliqué à un bloc de taille b.

Il présente l'avantage de ne pas stocker la table de codage.

Observation.

Soit $F^-_{\alpha_1 \dots \alpha_i} \stackrel{\text{def}}{=} \sum_{\alpha'_1 \dots, \alpha'_i < \alpha_1 \dots, \alpha_i} p_{\alpha'_1 \dots, \alpha'_i}$ pour l'ordre lexicographique où $p_{\alpha'_1 \dots, \alpha'_i}$ est la probabilité d'occurrence du bloc $\alpha'_1 \dots \alpha'_i$.

$$F^-_{\alpha_1 \dots \alpha_i \alpha_{i+1}} = F^-_{\alpha_1 \dots \alpha_i} + p_{\alpha_1 \dots \alpha_i} F^-_{\alpha_{i+1}} \qquad \text{avec } F^-_{n+1} = 1$$

$$p_{\alpha_1 \dots \alpha_i \alpha_{i+1}} = p_{\alpha_1 \dots \alpha_i} p_{\alpha_{i+1}}$$

Encodage de $\alpha_1 \dots \alpha_b$.

$sp \leftarrow 0$; $p \leftarrow 1$

For i **from** 1 **to** b **do**

  $sp \leftarrow sp + pF^-_{\alpha_i}$

  $p \leftarrow pp_{\alpha_i}$

$sp \leftarrow sp + \frac{p}{2}$; $\ell \leftarrow \lceil -\log(p) \rceil + 1$; $r \leftarrow \lfloor sp \rfloor_\ell$

Le codage arithmétique : décodage

Observation.

L'algorithme de décodage repose sur les inéquations suivantes :

$$F^-_{\alpha_1} \leq F^-_{\alpha_1\alpha_2} \leq \ldots \leq F^-_{\alpha_1\ldots\alpha_b} < \lfloor \bar{F}_{\alpha_1\ldots\alpha_b} \rfloor_{\ell_{\alpha_1\ldots\alpha_b}}$$

$$\lfloor \bar{F}_{\alpha_1\ldots\alpha_b} \rfloor_{\ell_{\alpha_1\ldots\alpha_b}} < F^-_{\alpha_1\ldots(\alpha_b+1)} \leq \ldots \leq F^-_{\alpha_1(\alpha_2+1)} \leq F^-_{\alpha_1+1}$$

Décodage de r en un bloc B.

$sp \leftarrow 0\,;\, p \leftarrow 1$

For i **from** 1 **to** b **do**

  $j \leftarrow 1$

  While $sp + pF^-_{j+1} \leq r$ **do** $j \leftarrow j + 1$

  $B[i] \leftarrow j$

  $sp \leftarrow sp + pF^-_j$

  $p \leftarrow pp_j$

Plan

Codage statique versus codage adaptatif

Compression d'un fichier.

- Calcul des fréquences de caractères $p = \{p_x\}_{x \in \mathcal{X}}$
- Compression optimisée en fonction de p
- Stockage du fichier compressé avec la table de codage

Flux de caractères sur un canal.

- Calcul des fréquences de caractères à la volée
- Codage adapté aux fréquences courantes
- Table de codage maintenue par l'émetteur et le récepteur

Pour l'analyse asymptotique du codage adaptatif,
on suppose que l'apparition des caractères suit une distribution inconnue de l'émetteur et que les tirages sont indépendants.

($\Sigma = \{0, 1\}$ pour les codages adaptatifs étudiés)

Huffman adaptatif

L'émetteur et le récepteur maintiennent un arbre de Huffman correspondant aux occurrences des caractères déjà apparus.

Le caractère courant est codé et décodé avec cet arbre et l'arbre est mis à jour *incrémentalement* en fonction de la nouvelle occurrence.

Soit l'arbre d'un code préfixe de n caractères avec $w(u)$ le poids du sommet u. S'il existe une numérotation des sommets $(u_i)_{1 \leq i < 2n}$ vérifiant :

- pour tout $i < j$, $w(u_i) \leq w(u_j)$;
- pour tout $1 \leq j < n$, u_{2j-1} et u_{2j} ont même père.

Alors ce code est un code de Huffman.

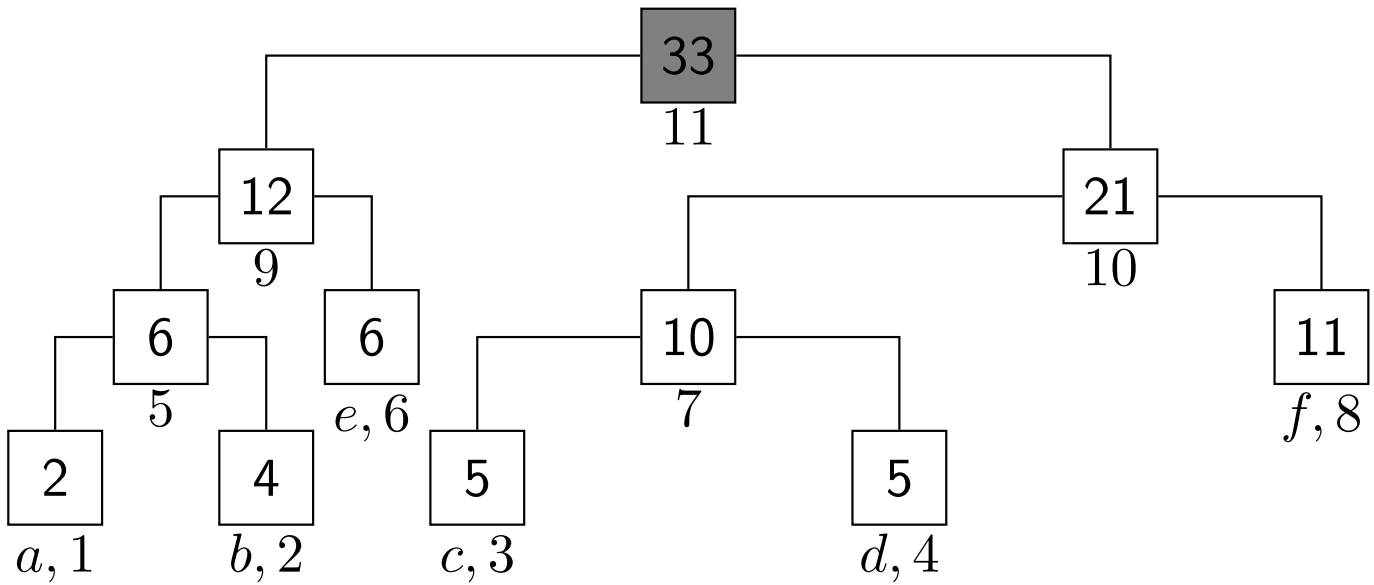
Preuve.

Si on dispose de cette numérotation alors on applique l'algorithme de Huffman en choisissant les sommets $\{1, 2\}$, puis $\{3, 4\}$, etc. et on obtient l'arbre original.

Mise à jour efficace de l'arbre

Lorsque $w(u_i)$ est incrémenté :

- Si $E := \{j \mid j > i \wedge w(u_j) < w(u_i)\} \neq \emptyset$ alors
  on échange les indices et les sous-arbres de u_i et de $u_{\max(E)}$;
- On incrémente le poids du père de u_i (devenu $u_{\max(E)}$)
  et on itère le procédé jusqu'à la racine.



Initialisation de l'arbre

Chaque événement a un code de première apparition.

A l'arrivée du premier caractère x, un arbre de deux feuilles est créé :

- l'une associée à x ;
- l'autre associée au joker $?$ sans occurrence.

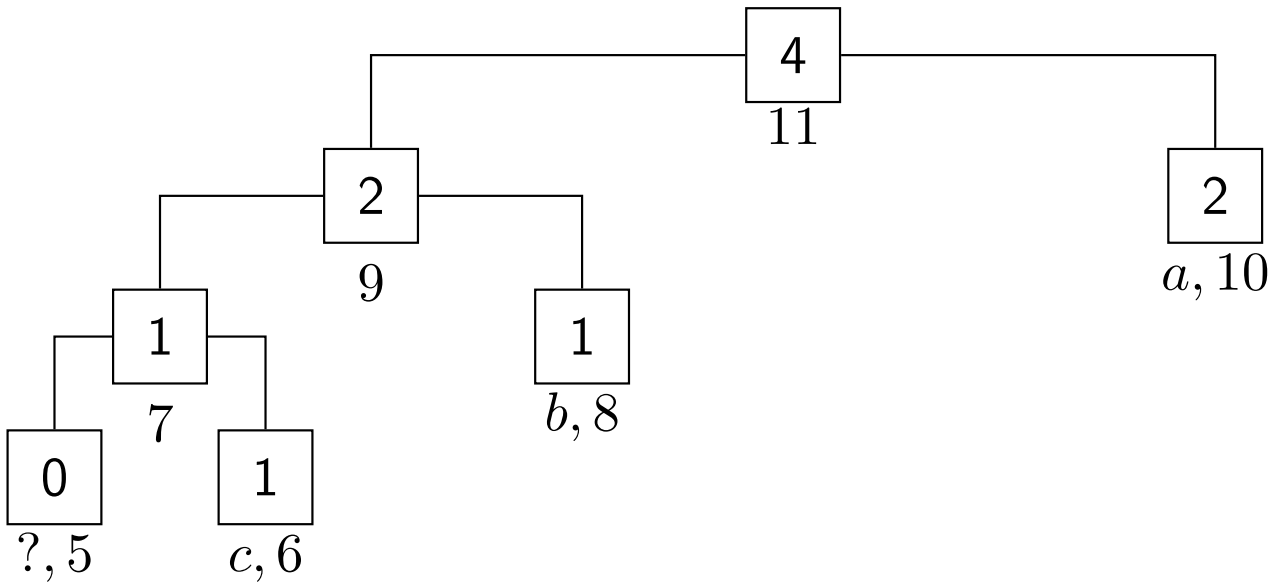
Chaque fois qu'un nouveau caractère y transmis, le code envoyé est la concaténation du code du $?$ et du code de première apparition de y.

- La feuille du caractère joker devient un sommet interne avec deux fils ;
- Une feuille pour le caractère joker et une feuille pour y.

Lorsque le dernier caractère apparaît,
on remplace le caractère joker par ce caractère.

Lorsqu'on met à jour le frère du caractère joker,
on met à jour son père et on démarre du père la mise à jour de l'arbre.

Illustration de l'initialisation



Analyse asymptotique

- Soit P_i^k la fréquence aléatoire de x_i dans le flux des k premiers caractères.

  D'après la loi forte des grands nombres, P_i^k converge presque sûrement vers p_i.

  Donc presque sûrement, $(P_i^k)_{i \leq n}$ converge vers $(p_i)_{i \leq n}$.

- Considérons un flux ω dont la suite des fréquences converge vers $(p_i)_{i \leq n}$.

  Pour tout ε il existe un k_ε tel que pour tout i et $k \geq k_\varepsilon$, $P_i^k(\omega) \leq p_i(1+\varepsilon)$.

  Soit ℓ_k la longueur du code à la $k$ième étape pour $k \geq k_\varepsilon$.

  Soit la longueur d'un code de Huffman ℓ_H : $\ell_k \leq \ell_H(1+\varepsilon)$.

- Soit $0 < \delta = \min(\frac{\ell(C)-\ell_H}{\ell_H} \mid \ell(C) > \ell_H)$.

  Si $\varepsilon < \delta$ alors pour $k \geq k_\varepsilon$, $\ell_k = \ell_H$.

Attention : le code peut ne jamais se stabiliser ($p_1 = p_2 = p_3 = \frac{1}{3}$)
mais il est assuré de se stabiliser si les p_i sont distincts.

Le découpage de Lempel-Ziv

Découpage du flux $w = (x_i)_{i\in\mathbb{N}}$**.** *(ici* $\mathcal{X} = \{0,1\}$*)*

- Le flux transmis $x_1 \dots x_n$ a été découpé en mots $y_1 \dots y_k = x_1 \dots x_n$ tels que pour tout $i \neq j$, $y_i \neq y_j$;
- Le nouveau mot à transmettre y_{k+1} est le plus petit mot $x_{n+1} \dots x_m$ tel que $y_{k+1} \notin \{y_i\}_{i\leq k}$.

Illustration. *(les points indiquent le découpage)*

- $0 \cdot 0^2 \cdot \dots \cdot 0^i$: $\frac{i(i+1)}{2}$ caractères découpés en i mots ;
- $0 \cdot 1 \cdot 00 \cdot \dots \cdot 1^i$: $(i-1)2^{i+1}+2$ caractères découpés en $2^{i+1}-2$ mots.

Le ratio nombre de mots du découpage / taille du mot à coder.

- $n \in \mathbb{N}_w$ ssi il existe $k(n)$ tel que $w_n \overset{\text{def}}{=} x_1 \dots x_n = y_1 y_2 \dots y_{k(n)}$.
- Soit $n \in \mathbb{N}_w$ il existe un unique i_n tel que $(i_n-1)2^{i_n+1}+2 < n \leq i_n 2^{i_n+2}+2$.
- $k(n) \leq 2^{i_n+2}-2$. D'où $\frac{k(n)}{n} \leq \frac{2^{i_n+2}-2}{(i_n-1)2^{i_n+1}+2} \leq \frac{2}{i_n-1}$.
- Puisque $\lim_{n\to\infty} i_n = \infty$, $\lim_{n\to\infty} \frac{k(n)}{n} = 0$.

Le codage de Lempel-Ziv

Codage

- $y_0 = \varepsilon$, $C(y_0) = 0$ et si $y_i = y_j x$ avec $x \in \{0, 1\}$ alors $C(y_i) = j \cdot x$;
- où j est écrit en binaire avec $\lfloor \log_2(\max(i-1, 1)) \rfloor + 1$ bits.

Illustration. La table de codage pour $w = 0 \cdot 01 \cdot 1 \cdot 10 \cdot 011$.

indice	0	1	2	3	4	5
nbre de bits de j		1	1	2	2	3
mot	ε	0	01	1	10	011
code	0	$[0]_b0$=00	$[1]_b1$=11	$[0]_b1$=001	$[3]_b0$=110	$[2]_b1$=0101

Observations.

Pour tout $i \leq k(n)$, $|C(y_i)| \leq \log_2(k(n)) + 2$.

D'où $\frac{|C(w_n)|}{n} \leq \frac{k(n)}{n} \log_2(k(n)) + \frac{2k(n)}{n}$.

Donc la longueur moyenne du code vérifie :

$$\limsup_{n\to\infty} \frac{|C(w_n)|}{n} \leq \limsup_{n\to\infty} \frac{k(n)}{n} \log_2(k(n))$$

Notation. $c_\ell(n) = |\{j \leq k(n) \mid |y_j| = \ell\}|$

le nombre de mots de longueur ℓ du découpage de w_n.

Une seconde majoration

$$\limsup_{n\to\infty} \frac{k(n)}{n} \log_2(k(n)) \leq \limsup_{n\to\infty} \frac{1}{n} \sum_{\ell\in\mathbb{N}} c_\ell(n) \log_2(c_\ell(n))$$

Preuve.

- Soit une v.a. U_n telle que $\mathbf{Pr}(U_n = \ell) = \frac{c_\ell(n)}{k(n)}$. $\mathbf{E}(U_n) = \sum_{\ell\in\mathbb{N}} \ell \frac{c_\ell(n)}{k(n)} = \frac{n}{k(n)}$.
- Appliquons la majoration de l'entropie par la loi géométrique :

$$\begin{aligned} H(U_n) \leq & (\frac{n}{k(n)} + 1) \log_2(\frac{n}{k(n)} + 1) - (\frac{n}{k(n)}) \log_2(\frac{n}{k(n)}) \\ = & (\frac{n}{k(n)} + 1)(\log_2(\frac{n}{k(n)} + 1) - \log_2(\frac{n}{k(n)})) + \log_2(\frac{n}{k(n)}) \\ = & (\frac{n}{k(n)} + 1) \log_2(\frac{k(n)}{n} + 1) - \log_2(\frac{k(n)}{n}) \end{aligned}$$

- Par conséquent :

$$\begin{aligned} \frac{k(n)}{n} \log_2(k(n)) = & \frac{1}{n} \sum_{\ell\in\mathbb{N}} c_\ell(n) \log_2(k(n)) = \frac{1}{n} (\sum_{\ell\in\mathbb{N}} c_\ell(n) \log_2(\frac{k(n)}{c_\ell(n)}) + \sum_{\ell\in\mathbb{N}} c_\ell(n) \log_2(c_\ell(n)) \\ = & \frac{k(n)}{n} H(U_n) + \frac{1}{n} \sum_{\ell\in\mathbb{N}} c_\ell(n) \log_2(c_\ell(n)) \\ \leq & (\frac{k(n)}{n} + 1) \log_2(\frac{k(n)}{n} + 1) - \frac{k(n)}{n} \log_2(\frac{k(n)}{n}) + \frac{1}{n} \sum_{\ell\in\mathbb{N}} c_\ell(n) \log_2(c_\ell(n)) \end{aligned}$$

Le résultat s'obtient par passage à la limite.

Inégalité de Ziv

Soit $(p(x))_{x\in\{0,1\}}$ une distribution quelconque.

Soit $n \in \mathbb{N}_w$:

$$\sum_{\ell\in\mathbb{N}} c_\ell(n)\log_2(c_\ell(n)) \leq -\sum_{i\leq n}\log_2(p(x_i))$$

Preuve.

Soit $y_j = x_u \dots x_v$. Alors $p(y_j) = \prod_{u\leq i\leq v} p(x_i)$. Donc :

$$-\sum_{i\leq n}\log_2(p(x_i)) = -\sum_{j\leq k(n)}\log_2(p(y_j))$$

$$= -\sum_{\ell\in\mathbb{N}} c_\ell(n)\sum_{|y_j|=\ell}\frac{1}{c_\ell(n)}\log_2(p(y_j)) \geq -\sum_{\ell\in\mathbb{N}} c_\ell(n)\log_2\Big(\sum_{|y_j|=\ell}\frac{1}{c_\ell(n)}p(y_j)\Big)$$

Puisque tous les y_j sont distincts $\sum_{|y_j|=\ell} p(y_j) \leq 1$.

D'où $-\sum_{i\leq n}\log_2(p(x_i)) \geq -\sum_{\ell\in\mathbb{N}} c_\ell(n)\log_2(\frac{1}{c_\ell(n)}) = \sum_{l\in\mathbb{N}} c_\ell(n)\log_2(c_\ell(n))$

Conséquence. $\limsup_{n\to\infty}\frac{|C(w_n)|}{n} \leq \limsup_{n\to\infty} -\frac{1}{n}\sum_{i\leq n}\log_2(p(x_i))$.

Optimalité asymptotique de Lempel-Ziv

Soit $(X_i)_{i\in\mathbb{N}}$ une famille de v.a. indépendantes de distribution commune p.

Alors d'après la loi forte des grands nombres, presque sûrement :

$$\lim_{n\to\infty} -\frac{1}{n}\sum_{i\leq n}\log_2(p(X_i)) = -\mathbf{E}(\log_2(p(X_1))$$

.

Or :

$$-\mathbf{E}(\log_2(p(X_1)) = H(X_1)$$

Soit $W_n = X_1 \dots X_n$. On a donc presque sûrement :

$$\limsup_{n\to\infty} \frac{|C(W_n)|}{n} \leq H(X_1)$$