Projet MAP 311
Entropie et codage de source

Alice Andrès, Quentin Soubeyran

3 juillet 2017

1 Entropie d'une distribution de probabilité

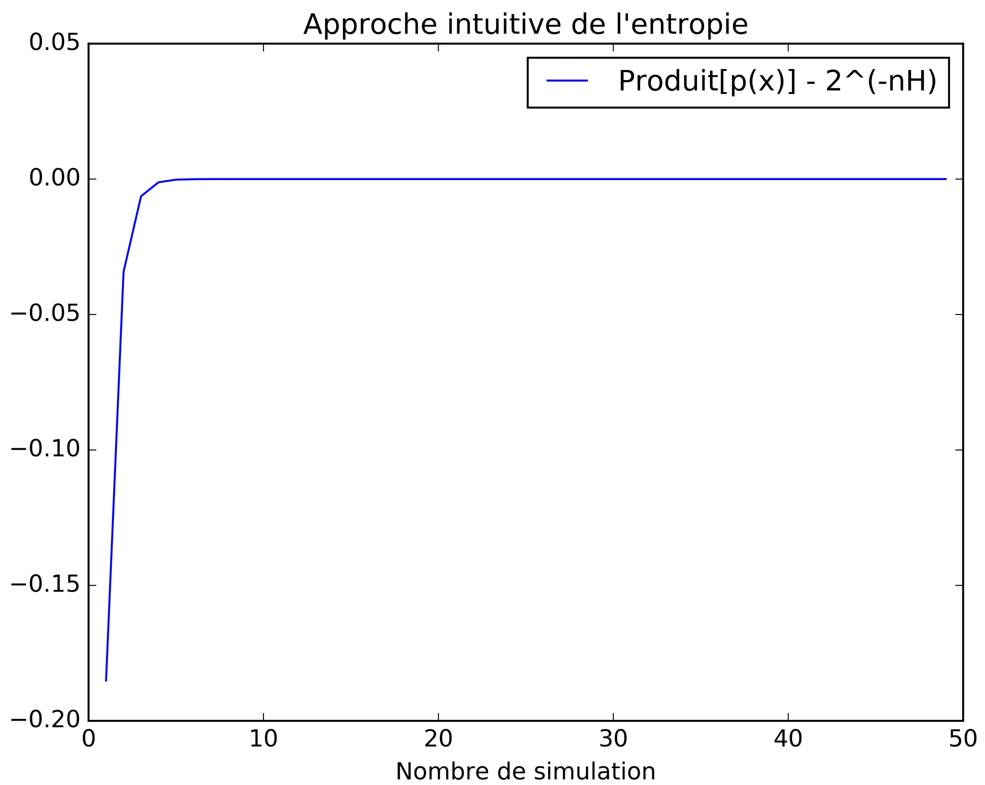
1.1 Cadre de travail et idée intuitive

Question 1

Comme $X \sim \mathcal{B}(N,p)$, on a :

$$p(x) = \binom{N}{k} p^k (1-p)^{(N-k)}$$

On peut ainsi calculer l'entropie $\mathcal{H}$ de X numériquement. Simulons n variables aléatoires et calculons la différence entre $2^{-n\mathcal{H}}$ et $p_X(x_1)...p_X(x_n)$.



On remarque que la différence converge rapidement vers 0 (cette observation est confirmée pour un grand nombre de simulation, 10000 par exemple).

1.2 Entropie relative et information mutuelle

Question 2

$$\mathcal{D}(\mathcal{B}(a)||\mathcal{B}(b)) = a\log_2(\frac{a}{b}) + (1-a)\log_2(\frac{1-a}{1-b})$$

D'où

$$\mathcal{D}(\mathcal{B}(a)||\mathcal{B}(b)) - \mathcal{D}(\mathcal{B}(b)||\mathcal{B}(a)) = (a+b)\log_2\frac{a}{b} + (2-a-b)\log_2\frac{1-a}{1-b}$$

Or pour $a = \frac{1}{4}$ et $b = \frac{1}{2}$,

$$\mathcal{D}(\mathcal{B}(a)||\mathcal{B}(b)) - \mathcal{D}(\mathcal{B}(b)||\mathcal{B}(a)) = \frac{3}{4}\log_2{(2)} + \frac{5}{4}\log_2{(\frac{3}{2})} \neq 0$$

Ainsi, dans le cas général, $\mathcal{D}(p||q) \neq \mathcal{D}(q||p)$

Question 3a

La fonction $-\log_2$ est strictement convexe. Alors, d'après l'inégalité de Jensen,

$$\begin{aligned}\sum_{x\in E} \mathrm{p}(x)\left(-\log_2\frac{q(x)}{p(x)}\right) &\geq -\log_2\sum_{x\in E} p(x)\frac{q(x)}{p(x)} \\ &\geq -\log_2\sum_{x\in E} q(x) \\ &\geq 0\end{aligned}$$

Ainsi $D(p||q) \geq 0$. La stricte convexité de $-\log_2$ permet de conclure qu'il y a égalité si et seulement si $\forall x \in E$, $p(x) = q(x)$, soit $p = q$.

Question 3b

D'après Q3a, $\mathcal{I}(X, Y) = \mathcal{D}(p_{(X,Y)}||p_X \otimes p_Y) \geq 0$

Avec égalité si et seulement si $p_{(X,Y)} = p_X \otimes p_Y$ soit X et Y indépendantes.

Question 4a

$$\begin{aligned}\mathcal{H}(X, Y) &= -\sum_{x,y\in E^2} p_{X,Y}(x, y)\log_2{(p_{X,Y}(x, y))} \\ &= -\sum_{x\in E}\sum_{y\in E} p_X(x)p_{Y|X=x}(y)\log_2 p_X(x) + \log_2 p_{Y|X=x}(y) \\ &= \mathcal{H}(X) + \sum_{x\in E} p_X(x)\left(-\sum_{Y\in E} p_{Y|X=x}(y)\log_2 p_{Y|X=x}(y)\right)\end{aligned}$$

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

Et l'on a montré l'égalité.

Question 4b

$$
\begin{aligned}
\mathcal{I}(X,Y) &= \sum_{(X,Y)\in E} p_{X,Y}(x,y)\log_2 \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)} \\
&= \sum_{(X,Y)\in E} \mathrm{p_X}(x)p_{Y|X=x}(y)\log_2\left(p_{Y|X=x}(y)\right) - \sum_{(X,Y)\in E} p_Y(y)p_{X|Y=y}(x)\log_2\left(p_Y(y)\right) \\
&= \mathcal{H}(Y) - \mathcal{H}(Y|X) \\
&= \mathcal{H}(X) - \mathcal{H}(X|Y) \quad \text{par symétrie des rôles de X et Y} \\
&= \mathcal{H}(Y) - (\mathcal{H}(X,Y) - \mathcal{H}(X)) \quad \text{cf. (Q4a)} \\
&= \mathcal{H}(X) + \mathcal{H}(Y) - \mathcal{H}(X,Y)
\end{aligned}
$$

Question 4c

D'après Q4b, $\mathcal{H}(X,Y) = \mathcal{H}(X) - \mathcal{I}(X;Y)$ Or $\mathcal{I}(X;Y) \geq 0$
Ainsi, $\mathcal{H}(X,Y) \leq \mathcal{H}(X)$

Question 5a

On utilise l'algorithme d'inversion de la fonction de répartition pour une loi discrète.

On utilise python pour déterminer un nombre a aléatoirement suivant la loi uniforme, entre 0 et 1, et on pose Y tel que :

$$Y = x_i \iff \sum_{j=1}^{i-1} p_j < a \leq \sum_{j=1}^{i} p_{j+1}$$

On peut appliquer ce principe pour $X \rightsquigarrow \mathcal{B}(\frac{1}{3})$
Soit $a \sim \mathcal{U}([0;1])$. Notons aussi $x_0 = 1$ et $x_1 = 0$
Alors $\mathbb{P}(X = x_0) = \frac{2}{3} = \mathbb{P}(a < \frac{2}{3})$ et $\mathbb{P}(X = x_1) = \frac{1}{3} = \mathbb{P}(a > \frac{2}{3})$.

Question 5b

Voir le code dans le fichier Q5.py.

Question 6a

Voir le code dans le fichier Q6.py.

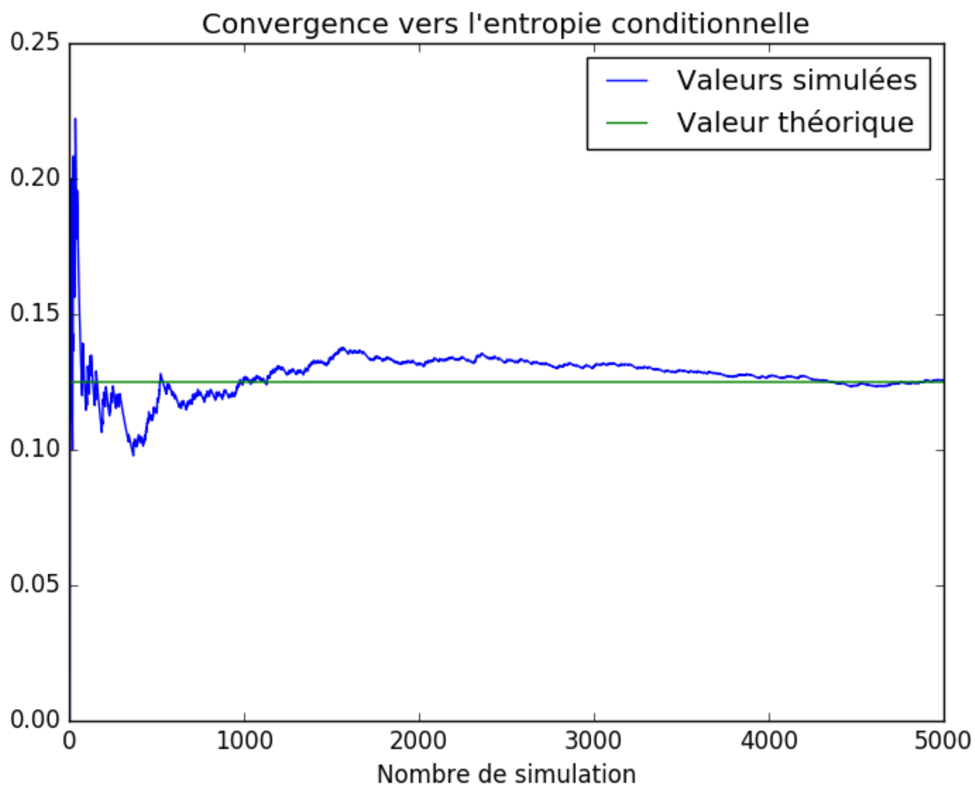
On obtient

$$H(X) \approx 0.6685644431995964 \text{ et } \mathcal{H}(X|Y) = 0,125$$

Question 6b

Voir le code dans le fichier Q6.py.

On observe, encore une fois, une convergence de $\frac{1}{k}\sum_{i=1}^{k} \mathcal{H}(X|Y = y_i)$ vers $\mathcal{H}(X|Y) = 0,125$.



Question 6c

On obtient

$$\mathcal{H}(X|Y=0) = 0 \text{ et } \mathcal{H}(X|Y=1) = 1 > \mathcal{H}(X) \approx 0.6685644432$$

Cela ne va pas dans le sens de la Q4b, mais il est normal qu'on ait plus d'informations lorsque l'on connaît Y dans sa totalité plutôt que seulement certaines de ses issues.

2 Application au codage de source

2.1 Théorème du codage de source

Question 7a

$$\begin{aligned}\mathcal{D}(p_X||q) &= \sum_{x \in E} p_X(x) log_2(\frac{p_X(x)}{\frac{1}{c}d^{-l(x)}}) \\ &\geq 0\end{aligned}$$

Alors

$$\sum_{x\in E} p_X(x)log_2(p_X(x)) \geq -\sum_{x\in E} p_X(x)l(x)log_2(d) + \sum_{x\in E} p_X(x)log_2(\frac{1}{c})$$

$$\iff$$

$$\begin{aligned} -\mathcal{H}(X) &\geq -log_2(d)\mathbb{E}(X) + log_2(\frac{1}{c}) \\ &\geq -log_2(d)\mathbb{E}(X) \quad (\text{car} \leq 1) \end{aligned}$$

D'où $\frac{\mathcal{H}(x)}{log_2(d)} \leq \mathbb{E}[l(X)]$

Le cas d'égalité se déduit de celui de $\mathcal{D}$, et a lieu pour $p_X = q$, soit les $p_X(x)$ sont des puissances négatives de d.

Question 7b

Soit p une loi de probabilité telle que qui s'écrit $p_X(x) = \frac{1}{c}d^{-n_x}$ avec $c = \sum_{x\in E} d^{-n_x}$.

Cas 1 : $c \leq 1$ Prenons $\forall x \in E, l_0(x) = n_x$

Cas 2 : $c > 1$ Alors soit k tel que $\frac{c}{d^k} \leq 1$

$p_X(x) = \frac{d^k}{c}d^{-n_x-k}$, avec $\sum_{x\in E} d^{-n_x-k} \leq \frac{c}{d^k} \leq 1$

Posons alors $\forall x \in E, l_0(x) = n_x + k$

Cette application vérifie l'inégalité de Kraft-McMillan, et vérifie le cas d'égalité de la question Q7a d'après les calculs précédents pour q définie à partir de la fonction l_0.

Question 7c

La fonction puissance étant bijective sur $\mathbb{R}^+$, on a :

$$\forall x \in E, \exists \alpha_x, \quad p_X(x) = d^{\alpha_x}$$

Posons c et β tels que :

$$c = \sum_{x\in E} d^{\alpha_x} = d^{\beta}$$

Alors

$$\forall x \in E, \quad p_X(x) = \frac{1}{c}d^{-(\beta-\alpha_x)}$$

On pose donc

$$l_0(x) = \beta - \alpha_x$$

D'où

$$\begin{aligned} \mathbb{E}[\overline{l_0}(X)] &= \sum_{x\in E} \overline{l_0}(X)\mathbb{P}(X = x) \\ &< \sum_{x\in E} l_0(X)\mathbb{P}(X = x) + \sum_{x\in E} \mathbb{P}(X = x) \end{aligned}$$

Or d'après la question Q7a, la forme de $p_X(x) = \frac{1}{c}d^{-(\beta-\alpha_x)}$ assure :

$$\frac{\mathcal{H}(X)}{\log_2(d)} = \mathbb{E}[l(X)] \quad \text{puisque } \mathcal{D}(p_X||p_X) = 0$$

On en conclut :

$$\mathbb{E}[\overline{l_0}(X)] < \frac{\mathcal{H}(X)}{\log_2(d)} + 1$$

2.2 Mise en oeuvre de l'algorithme - L'algorithme de Huffman

Question 9a

Voici le tableau des occurrences.

a	b	c	d	e	f
2	3	1	2	2	1

On choisit c et f

a	b	d	e	cf
2	3	2	2	2

On choisit e et cf

a	b	d	ecf
2	3	2	4

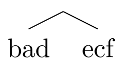
On choisit a et d

b	ad	ecf
3	4	4

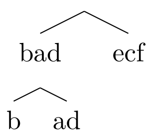
On choisit b et ad

bad	ecf
7	4

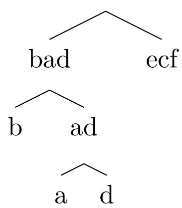
On n'a plus que deux éléments, construisons donc l'arbre en remontant les étapes précédentes.



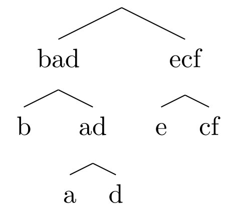
On décompose bad en b et ad



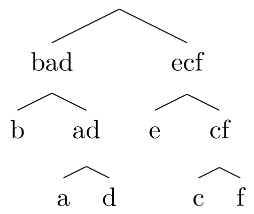
On décompose ad en a et d



On décompose ecf en e et cf



On décompose cf en c et f



On en déduit le codage de Huffman :

a	b	c	d	e	f
010	00	110	011	10	111

Question 9b

Voir le fichier Python Q9.py

Question 9c

Voir code dans le fichier Q9.py.

L'idée est d'utiliser une fonction récursive pour construire l'arbre selon l'algorithme présenté par le sujet. On utilise pour cela :

- le module `heapq` qui permet de gérer des tas, facilitant ainsi la recherche les deux minimums (point (i) de l'algorithme).
- Une classe pour représenter l'arbre de Huffman pendant sa consruction par l'algorithme, `HuffTree`, implémentée dans `CustomObject.py`. Celle-ci sert à représenter la concaténation de deux caractère ou arbres des étapes précédentes. Ainsi, une fois l'arbre complet construit, celui-ci est exactement de la forme du dernier arbre de la question précédente.
  Cette classe implémente également `toDict()`, une fonction donnant une représentation sous forme de dictionnaire, plus adaptée.
- Une classe permettant de donner une valeur de comparaison à n'importe quel objet (classe `Token` de `CustomObject.py`). Cela permet de mettre les instances de `HuffTree` directement dans le tas géré par `heapq`.

A la différence de l'algorithme présenté par le sujet, l'étape (iii) est effectuée au fur et à mesure, en même temps que l'étape (ii), par la construction d'une instance de `HuffTree` à chaque appel récursif de la fonction auxiliaire interne.

Question 9d

Le code est disponible dans le fichier Q9.py. On constate que le langage de Huffman correspond bien à un langage décrit par le Théorème de Schannon, puisqu'il vérifie la double inégalité (iii)

En effet, on obtient pour une probabilité uniforme :

$$3.4585110748 \leq 3.54475447545 < 4.4585110748$$

Et pour la répartition des lettres de la langue française :

$$2.77115542449 \leq 2.83174404962 < 3.77115542449$$

Question 9e

Les moyennes obtenues sont de l'ordre de 3, soit très intéressantes par rapport à 8 bit ; surtout lorsque la répartition des caractères n'est pas uniforme.

Mais l'alphabet considéré est restreint par rapport à ce que les 8 bits peuvent coder. Il est donc plus pertinent de comparer au nombre minimal de bits pour coder 11 caractères, qui est de $\log_2(11) \approx 3,45$.

On a donc besoin de 4 bits au minimum pour coder ces 11 caractères ; cela est aussi nécessaire avec le codage de Huffman lorsque la répartition est uniforme. On gagne cependant un bit lorsques les fréquences sont disparates : on passe à 3 bits. ($\lceil \mathbb{E}(l(X)) \rceil$).