Codage

Concepts : codage, compression
Méthodes : algorithme glouton, algorithme de Huffman

Présentation

À la fin de la guerre Shannon découvre les premiers concepts de la théorie du codage. Dans un article fondateur [4], il associe à une séquence de lettres (ou de mots) son entropie (définition analogue à la définition en physique) et en a donné une interprétation qui a servi de base à la théorie des codes et de la compression. Sette entropie représente une quantité d'information contenue dans la séquence de lettres. Pour une introduction simple à ce domaine, l'ouvrage de Guiasu et Theodorecu [3] apporte les éléments essentiels. Les conséquences de cette théories sont nombreuses en informatique. On trouvera quelques exemples élémentaires dans le livre de Brémaud [1] chap. V.

L'objectif de cette note de cours est de donner la preuve de l'inégalité de Shannon pour les codages ayant la propriété du préfixe. La preuve est donnée dans le cas d'un codage binaire. La généralisation à un codage par un alphabet de plus de 2 symboles est facile, voir [1].

Parmi les algorithmes de codage, l'algorithme de Huffman joue un rôle central, d'une part par sa simplicité et d'autre part par la mise en œuvre de méthode gloutonne produisant une solution optimale. On trouvera des détails sur l'algorithme et sa preuve dans l'ouvrage de Cormen et al. [2] chapitre 16.

Résultat principal

On suppose que l'on cherche à coder, par un codage ayant la propriété du préfixe [1], un ensemble de k symboles $\mathcal{S} = \{S_1, \cdots, S_k\}$ à l'aide des symboles 0 et 1. Un code h est donc une fonction qui associe à chaque symbole S_i un mot code s_i (mot de longueur l_i composé de 0 et de 1). La seule information dont on dispose est la proportion dans le texte des k symboles à coder. On note ces proportions $\{p_1, \cdots, p_k\}$. Bien entendu

$$\sum_{i=1}^{k} p_i = 1.$$

La longueur moyenne du code h est alors définie par

$$L(h) = \sum_{i=1}^{k} Proportion\_symbole(S_i).longueur(h(S_i)) = \sum_{i=1}^{k} p_i l_i.$$

Théorème 1 (*Shannon(1948)*)

Si on définit

$$L_{inf} = \inf_h L(h),$$

la longueur moyenne minimale des codages h ayant la propriété du préfixe, alors

$$\mathcal{H}(p) \leqslant L_{inf} \leqslant \mathcal{H}(p) + 1,$$

où $\mathcal{H}(p)$ appelée entropie *du système est définie par*

$$\mathcal{H}(p) = -\sum_{i=1}^{k} p_i \log_2 p_i.$$

1. un codage a la propriété du préfixe si tout mot-code n'est le préfixe d'aucun autre mot-code. Un code ayant la propriété du préfixe est non ambigu, c'est à dire uniquement déchiffrable

Codage compressif

Ce résultat est fondamental au sens où il lie la longueur moyenne du code aux proportions, ceci indépendamment du codage h choisi. D'une certaine manière, cela signifie que l'on ne pourra pas compresser l'information au delà d'un certain facteur $\mathcal{H}(p)$. Il y a une part irréductible de "hasard" que l'on ne peut pas coder !

La preuve de ce théorème se décompose en quatre étapes indépendantes. Les deux premières étapes décrivent des propriétés structurelles des codes ayant la propriété du préfixe (inégalité de Kraft). La troisième étape montre que la longueur moyenne minimale d'un codage est plus petite que $\mathcal{H}(p)$ et dans la quatrième étape on construit un codage de longueur moyenne plus petite que $\mathcal{H}(p) + 1$.

Inégalité de Kraft

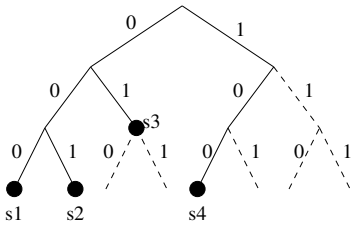
On considère un codage h ayant la propriété du préfixe. Soit $s_1, \cdots, s_k$ et $l_1, \cdots, l_k$ les mots codes et leur longueur.

Proposition 1 (*Inégalité de Kraft*)

Pour tout codage h ayant la propriété du préfixe on a l'inégalité

$$\sum_{i=1}^{k} 2^{-l_i} \leqslant 1.$$

On note $m = \max_i l_i$ la plus grande longueur des mots codes. Comme le code a la propriété du préfixe, on peut placer, de manière unique, ces mots codes sur un arbre $\mathcal{T}$ binaire équilibré de profondeur m (voir l'exemple de la figure 1), chaque mot code étant une feuille de cet arbre.



On considère les 4 mots codes associés aux symboles $\{S_1, \cdots, S_4\}$

Symbole	Code	longueur
S_1	000	3
S_2	001	3
S_3	01	2
S_4	100	3

Dans ce cas,

$$\sum_{i=1}^{4} 2^{-l_i} = \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^2} + \frac{1}{2^3} = \frac{5}{8} \leqslant 1.$$

L'inégalité de Kraft est bien vérifiée.

FIGURE 1 – Un exemple de positionnement des mots-codes dans l'arbre binaire $\mathcal{T}$

La preuve de l'inégalité, se fait en comptant le nombre de feuilles issues du mot code s_i dans l'arbre binaire complet $\overline{\mathcal{T}}$ de profondeur m . Comme le mot code est à la profondeur l_i le nœud correspondant est le père de 2^{m-l_i} feuilles à la profondeur m (voir figure 2).

Comme le codage a la propriété du préfixe les sous arbres issus des mots codes sont disjoints et donc le nombre total de feuilles issues des mots codes à la profondeur m est égal à

$$\sum_{i=1}^{k} 2^{m-l_i}.$$

Ce nombre de feuilles est inférieur ou égal au nombre total de feuilles à la profondeur m, c'est à dire à 2^m. En écrivant cette inégalité on obtient

$$\sum_{i=1}^{k} 2^{m-l_i} \leqslant 2^m.$$

En divisant les 2 termes de l'inégalité par 2^m on obtient l'inégalité de Kraft.

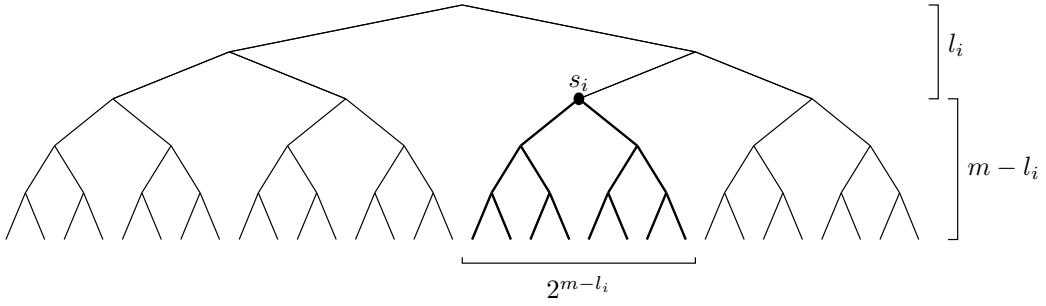
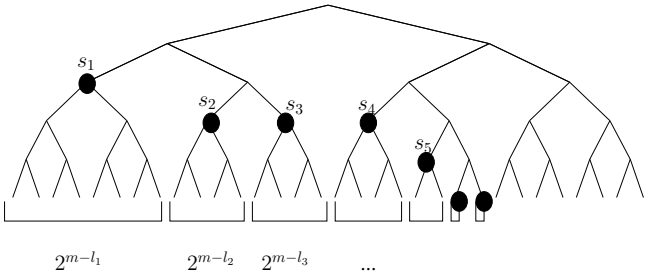


FIGURE 2 – Nombre de feuilles de profondeur m issues de s_i dans l'arbre binaire complet $\overline{\mathcal{T}}$

Codage vérifiant l'inégalité de Kraft

On suppose maintenant donné k longueurs $l_1, \cdots, l_k$ vérifiant l'inégalité de Kraft. Existe-t'il k mots-codes de longueurs respectives l_i possédant la propriété du préfixe ?

La réponse est oui. Pour le prouver on s'inspire du dessin précédent. On suppose les l_i classés par ordre croissant et on regroupe les feuilles du niveau m par paquets de taille 2^{m-l_i}. D'après l'inégalité de Kraft, les paquets peuvent tous être placés sur le dernier niveau de l'arbre. On trouve alors le codage associé en identifiant le père de chaque paquet (voir figure 3).



On considère les 7 longueurs de mots codes, $(2, 3, 3, 3, 4, 5, 5)$. On vérifie aisément l'inégalité de Kraft.

Longueur	Code	symbole
2	00	s_1
3	010	s_2
3	011	s_3
3	100	s_4
4	1010	s_5
5	10110	s_6
5	10111	s_7

FIGURE 3 – Un exemple de positionnement des mots-codes dans l'arbre binaire

La longueur moyenne d'un code est minorée par $\mathcal{H}(p)$

Considérons le problème de minimisation de la fonction f définie par

$$f(x_1, \cdots, x_k) = \sum_{i=1}^{k} p_i x_i,$$

sous la contrainte (inégalité de Kraft)

$$\sum_{i=1}^{k} 2^{-x_i} \leqslant 1.$$

On peut montrer que la fonction f est minimale sous la contrainte de l'inégalité de Kraft au point l^* de coordonnées $l_i^* = -\log_2 p_i$. Ce résultat est obtenu en utilisant les outils de minimisation de fonction de plusieurs variables. Au point l^* on vérifie que

$$\sum_{i=1}^{k} 2^{-l_i^*} = \sum_{i=1}^{k} 2^{-(-\log_2 p_i)} = \sum_{i=1}^{k} p_i = 1.$$

De plus

$$\sum_{i=1}^{k} p_i l_i^* = \sum_{i=1}^{k} p_i(-\log_2 p_i) = \mathcal{H}(p).$$

On a donc pour tout codage h de longueurs $l_1, \cdots, l_k$

$$\mathcal{H}(p) = f(l_1^*, \cdots, l_k^*) \leqslant f(l_1, \cdots, l_k) = L(h).$$

Code de longueur moyenne majorée par $\mathcal{H}(p) + 1$

Il reste maintenant à construire un code de longueur moyenne proche de $\mathcal{H}(p)$. Pour cela, considérons les valeurs

$$l_i^{sup} = \lceil l_i^* \rceil,$$

partie entière supérieur de l_i^* (seul nombre entier vérifiant $l_i^* \leqslant l_i^{sup} < l_i^* + 1$).

Comme $l_i^* \leqslant l_i^{sup}$, on a $2^{-l_i^*} \geqslant 2^{-l_i^{sup}}$ et

$$1 = \sum_{i=1}^{k} 2^{-l_i^*} \geqslant \sum_{i=1}^{k} 2^{-l_i^{sup}}.$$

D'après la réciproque de l'inégalité de Kraft il existe un codage ayant la propriété du préfixe et de longueurs $l_1^{sup}, \cdots, l_k^{sup}$. Ce codage a une longueur moyenne

$$\sum_{i=1}^{k} p_i l_i^{sup} \leqslant \sum_{i=1}^{k} p_i(l_i^* + 1) = \mathcal{H}(p) + 1.$$

On vient de montrer qu'il existe un codage ayant la propriété du préfixe de longueur moyenne plus petite que $\mathcal{H}(p) + 1$. On en conclut que le codage optimal a une longueur plus petite que $\mathcal{H}(p) + 1$.

Algorithme de Huffman

Le problème à résoudre est de trouver un code binaire h (c'est à dire un arbre), ayant la propriété du préfixe (facile à décoder) et minimisant la longueur moyenne du code. Pour réaliser cet arbre on construit une solution en partant des feuilles les plus profondes (probabilité les plus faibles) puis en les combinant deux à deux on construit l'arbre.

```
ALGORITHME_HUFFMAN
  Données : Un ensemble S de k symboles avec leurs pondérations p
  Résultat : Un arbre optimal de codage (nœud racine)
  Indice i
  Nœud x,y,z
  FAP F
  for s ∈ S
  |  z=nouveau_nœud() z.symbole=s z.poid=p(s)
  |  Insérer (F,z)
  for i = 1 to K − 1
  |  x=Extraire (F) y=Extraire (F) ;
  |  z=nouveau_nœud() z.gauche=x z.droit=y z.poids=x.poids+y.poids
  |  Insérer (F,z)
  Retourne Extraire (F)
```

Algorithme 1: Algorithme de Huffman (1951)

Dans cet algorithme on utilise une structure de donnée auxiliaire *file à priorité* F dans laquelle on insère des couples (nœud, priorité) et on extrait les couples selon leur priorité. Pour cet algorithme, l'extraction se fera par poids de nœud croissant.

Codage compressif

L'algorithme de Huffman produit un code ayant la propriété du préfixe de longueur moyenne optimale. C'est un exemple d'algorithme **glouton**, c'est à dire qu'à chaque étape l'algorithme effectue une opération qui le rapproche de la solution optimale [2]. L'algorithme nécessite donc de l'ordre de $\mathcal{O}(k)$ opérations d'insertion et d'extraction de F et, si la file à priorité est implantée dans une structure de donnée adéquate (par exemple un tas), le coût d'insertion et d'extraction est majoré par $\log K$. Ce qui donne à cet algorithme une complexité de l'ordre de $\mathcal{O}(K \log K)$.

La preuve de cet algorithme repose sur 2 lemmes préliminaires dont la preuve basée sur un argument d'échange est laissée en exercice.

Lemme 1 (*Probabilités les plus faibles*)

> *Soit S un alphabet de k lettres. Soit s et s' deux symboles de plus petite probabilité. Alors il existe un codage préfixé optimal pour S tel que les mots codes de s et de s' ne diffèrent que par le dernier bit.*

Indication : prendre un arbre optimal et le transformer de manière à vérifier la propriété.

Lemme 2 (*Propagation de l'optimalité*)

> *Soit T un arbre de codage optimal (complet) de S. Alors la fusion z de 2 feuilles sœurs s et y affectée de la somme des poids des feuilles $z.poids = x.poids + y.poids$ produit un arbre optimal pour l'alphabet S' dans lequel tous les caractères $x.symbole$ et $y.symbole$ ont été remplacés par un même nouveau symbole.*

Indication : raisonner par l'absurde

Pour prouver l'algorithme on va montrer que l'assertion suivante est vraie à chaque fin de l'itération principale.

Assertion : *La file à priorité contient une forêt incluse dans un arbre de codage optimal de l'alphabet S.*

D'abord la question de l'existence d'un arbre optimal ne se pose pas. En effet, l'ensemble des d'arbres binaires complets avec k feuilles étiquetées (ensemble des codages complets ayant la propriété du préfixe) est fin donc admet au moins un élément de longueur moyenne minimale.

Initialisation L'assertion est donc vraie au début de l'itération principale (une feuille est un sous-arbre de tout sous-arbre optimal).

Preuve partielle : Si l'assertion est vraie à l'entrée de l'itération, il existe un arbre optimal contenant la forêt incluse dans la file à priorité. Soit x et y les nœuds extraits de la FAP, d'après le lemme 1 il existe un arbre optimal tel que x et y soient 2 feuilles sœurs. Par le lemme 2 l'arbre obtenu lorsque l'on effectue la fusion de x et y reste optimal.

Terminaison : L'algorithme, faisant un nombre fini d'itérations, se termine. Chaque itération diminue de 1 le nombre de nœuds dans la FAP.

Conclusion À la fin des itérations il ne reste qu'un nœud qui est la racine d'un arbre optimal.

La construction de la table de codage se fait par simple parcours préfixé de l'arbre de codage.

Références

[1] Pierre Brémaud. *Introductions aux Probabilités*. Springer-Verlag, Berlin, 1984.

[2] Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Algorithmique*. Dunod, 2011.

[3] Silviu Guiasu and Radu Theodorecu. *Incertitude et Information*. Les Presses de l'Université LAVAL, Québec, 1971.

[4] Claude Elwood Shannon. A mathematical theory of communications. *Bells Systems Technical Journal*, 27, 1948.

2. à chaque étape on choisit la meilleure solution en espérant obtenir une bonne approximation de la solution optimale