Codage et Entropie

Ch. Dossal

Septembre 2010

1 Introduction

Nous proposons dans ce TD de réaliser un codeur de Huffmann ainsi qu'un codeur arithmétique et de tester leur propritiés théoriques sur des cas pratiques. Les programmes proposés seront exploités ultérieurement en compression.
Dans la suite, nous allons considérer l'alphabet $\mathcal{A}$ constitué des N entiers compris entre 1 et N ainsi qu'un vecteur fréquence F de taille N contenant les fréquences d'apparition des différents symboles.

2 Rappels Théoriques

Soit $\mathcal{A}$ un alphabet et S une suite d'éléments aléatoire de cet alphabet dont chaque élément est tiré indépendamment et selon une loi $\mathcal{P}$. Pour tout code C, l'espérance en bits du code d'un mot de n symboles est minorée par $n \times H$ où $H = \sum_{a \in \mathcal{A}} \log_2(p(a))p(a)$ est l'entropie associée à la loi de probabilité $\mathcal{P}$.
Une forte entropie nécessite donc un codage plus long, une faible entropie permet des codages compacts.
Dans tout algorithme de compression, la notion d'entropie est fondamentale. Un bon algorithme de compression doit fournir une représentation des données avec une faible entropie. Le codage de Huffmann et le codage arithmétique sont deux codages qui permettent d'approcher la borne théorique de l'entropie.
Dans les deux sections suivantes les tests seront effectués en priorité avec l'alphabet $\mathcal{A} = \{1, 2, 3, 4, 5\}$ et les fréquences suivantes $F = [0.1, 0.2, 0.3, 0.25, 0.15]$.

3 Codage de Huffmann

1. Ecrire un programme Huffmann

   ```
   function H=Huffmann(F)
   ```

   qui prend en entrée un vecteur de fréquence F de taille N et qui renvoie une liste de N mots binaires correspondant aux codages de Huffmann des symboles dont les fréquences sont données par F. Ce programme doit également afficher l'entropie théorique et l'espérance du nombre de bits par symbole du codage de Huffman.
   Il est possible de tout programmer en moins de 35 lignes.
   Quelques conseils :

- Faites un programme récursif qui traite d'abord le cas trivial où F est de longeur 2.
- Ensuite en utilisant la commande *s*ort vous pouvez extraire les deux plus petites valeurs de F et reconstruire un vecteur $F2$ contenant les valeurs de F en ayant remplacé les deux plus petites valeurs par la somme des deux.
- Vous pouvez ensuite relancer la fonction Huffmann sur $F2$
- A partir de la liste $H2$=Huffmann(F2) vous pouvez déterminer celle de F par concaténation.
- Attention, la permutation donnée par la commande *s*ort peut être utile.

2. Ecrire un programme de codage

   `function [C,H]=BitSeq(S,F)`

   qui prend en entrée un vecteur S d'entiers entre 1 et N, un vecteur de fréquence F de taille N et qui renvoie une séquence de bits C et la liste H des codes des éléments du dictionnaire.

3. Ecrire la fonction inverse qui à partir d'une suite binaire C et d'une liste de codes H reconstruit le vecteur S.

   `function S=DecHuff(C,H)`

4. Ecrire un programme qui génère une liste de n variables aléatoires indépendantes à valeur dans $[1, N]$ avec des probabilités données par le vecteur F de taille N.

   `function S=ListeAlea(n,F)`

5. Tester la longueur moyenne du code fournit par l'algorithme de Huffman.

4 Codage Arithmétique

Le codage arithmétique permet de coder une suite de mots dans un nombre réel. Chaque mot est associé à un intervalle et tout point de l'intervalle code le mot.

6. Ecrire un programme prenant en entrée un vecteur S à valeur entière dans $[1, N]$, un vecteur de fréquence F et qui renvoie un réel X codant le vecteur S.

   `function X=CodeA(S,F)`

7. Ecrire un programme prenant en entrée un réel X, un vecteur de fréquence F et un entier n et qui envoie un vecteur S de n valeurs.

   `function S=DeCodeA(X,F,n)`

   Ces deux codes peuvent être rédigés chacun en moins de 15 lignes.

8. Peut on coder une suite aussi grande que l'on veut dans X ?

9. Quelle est la limite théorique ?

10. Avec le F proposé dans la section 1, estimez pratiquement le nombre de symboles que l'on peut coder dans un réel.

11. Comment estimer le nombre de bits moyens utilisés pour coder un symbole par cette méthode.

12. Comment coder un mot plus grand ?

5 Conclusions

13. Que pensez vous de ces deux méthodes ?

14. Dans quel type de situations le codage de Huffmann peut être vraiment moins bon que le codage arithmétique ?

15. Comment généraliser ces deux méthodes dans des situations où les fréquences d'apparition de symboles sont inconnues ?