Codage statistique

Codage arithmétique

Codage arithmétique

- Principes de base :

  - le code est associé à la séquence des symboles m_i (messages), et non pas à chaque symbole de la séquence.

  - on code des intervalles $[c_i, d_i[$ pour chaque symbole.

  - itération jusqu'à avoir codé la séquence entière

  - on code une séquence par un unique nombre réel appartenant à l'intervalle [0, 1 [.

Le codage arithmétique permet, à partir de la probabilité d'apparition des symboles d'une source de créer un seul mot-code qui soit associé à une séquence de longueur arbitraire de symboles. Ceci diffère du code de Huffman qui attribue des mots-codes de longueurs variables à chaque symbole de la source.
Le code associé à une séquence est un nombre réel de l'intervalle [0, 1[. Ce code est construit par subdivisions récursives d'intervalles. Un intervalle est subdivisé pour chaque nouveau symbole qui appartient à la séquence. On obtient, en définitive, un sous-intervalle de l'intervalle [0, 1[tel que tout nombre réel appartenant à cet intervalle représente la séquence à coder.

Codage arithmétique

➢ ***Définitions:***

- Soit une source $S = \{s_1, s_2, \ldots, s_N\}$ avec $p_k = P(s_k)$

- $[Ls_k, Hs_k[$ est l'intervalle « ***assigné*** » au symbole s_k

avec: $Hs_k - Ls_k = p_k$

➢ ***Algorithme de codage:***

1) Initialisation: $L_c = 0$; $H_c = 1$
2) Déterminer les longueurs des sous intervalles liés aux symboles
3) Passer au codage sur le prochain symbole s_k
4) Déterminer, pour le sous-intervalle considéré, le raffinement en sous intervalles
5) Répéter les étapes 2, 3, et 4 jusqu'à obtenir le code de la séquence entière

Soit $S = \{s_1, \ldots, s_N\}$ une source pouvant délivrer N symboles et munie des probabilités suivantes : pour tout entier k de [1, N], $P\{s_k\} = p_k$.

Pour coder une séquence $s_M = \{s_{\alpha 1}, s_{\alpha 2}, \ldots, s_{\alpha M}\}$ de M symboles, on utilise l'algorithme suivant :

- Étape 1 :

On initialise un premier intervalle avec deux bornes : la borne inférieure $L_c = 0$ et la borne supérieure $H_c = 1$ (correspondant à la probabilité de choisir un premier symbole $s_{\alpha 1}$ parmi tous les symboles s_k de la source). Cet intervalle [0, 1[est partitionné en N sous-intervalles $[Ls_k, Hs_k[$ en fonction des probabilités de chaque symbole s_k de la source. Ce sont les partitions initiales. La longueur ($Ls_k - Hs_k$) de ces sous intervalles est donnée par : $Ls_k - Hs_k = p_k$, on a donc : $Ls_k = \sum_{i=1}^{k-1} p_i$ et $Hs_k = \sum_{i=1}^{k} p_i$.

- Étape 2 :

On mesure la longueur du sous-intervalle : longueur = $H_c - L_c$. Un nombre réel appartenant à cet intervalle représente le symbole.

- Étape 3 :

On effectue le codage du prochain symbole $s_{\alpha k}$ sachant le symbole précédent.

- Étape 4 :

Le sous-intervalle est subdivisé à nouveau. Les nouvelles valeurs de ses bornes H_c et L_c sont alors données par : $\begin{cases} L_c = L_c + \text{longueur} \times Ls_k \\ H_c = L_c + \text{longueur} \times Hs_k \end{cases}$

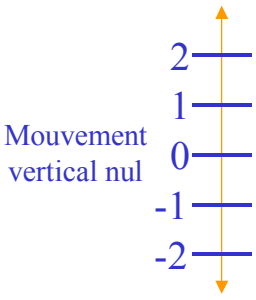
- Étape 5 :

Les étapes 2, 3, et 4 sont répétées jusqu'à obtenir le mot-code représentant la séquence complète des symboles source.

Codage arithmétique

Exemple

On considère la source S= {-2, -1, 0, 1, 2} qui définit une série de 5 vecteurs de mouvements verticaux possibles (vecteurs utilisés en codage vidéo)



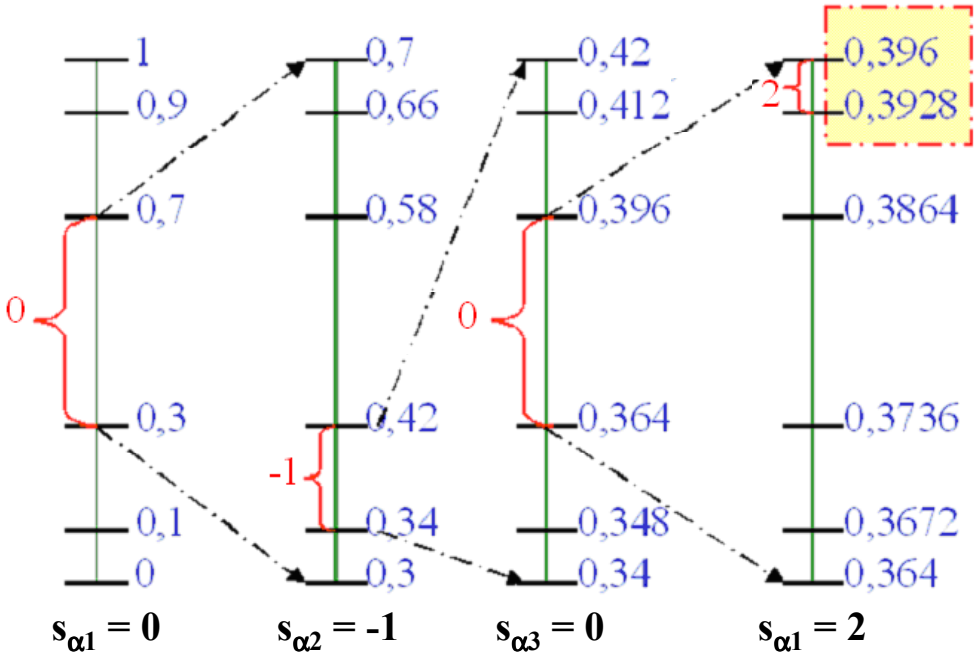
- 'Y' est la variable aléatoire qui désigne la valeur d'un vecteur de mouvement.
- On a les probabilités suivantes:
  - $P\{Y = -2\} = p_1 = 0,1$
  - $P\{Y = -1\} = p_2 = 0,2$
  - $P\{Y = 0\} = p_3 = 0,4$
  - $P\{Y = 1\} = p_4 = 0,2$
  - $P\{Y = 2\} = p_5 = 0,1$

⇒ L'objectif est de coder la séquence de vecteurs de mouvement (0, -1, 0, 2)

Voici un exemple d'application du codage arithmétique dans le cadre de codage des vecteurs de mouvement d'un signal vidéo.

Codage arithmétique

Subdivisions de la partition initiale



Pour coder la séquence $\{s_{\alpha 1}, s_{\alpha 2}, s_{\alpha 3}, s_{\alpha 4}\} = \{s_3, s_2, s_3, s_5\} = \{0, -1, 0, 2\}$, on divise l'intervalle [0, 1[en 5 partitions initiales, puis on se place sur le sous-intervalle correspondant au premier vecteur de mouvement de la séquence (de valeur 0). Pour le prochain symbole de la séquence, le vecteur de mouvement –1, on subdivise la partition initiale du premier vecteur '0' en autant d'intervalles qu'il y'a de vecteurs possibles.

On procède récursivement pour tous les symboles (ici les vecteurs de mouvement) de la séquence. Dès lors, on peut coder la séquence (0, -1, 0, 2) des vecteurs de mouvements verticaux par tout nombre réel appartenant à l'intervalle [0,3928 ; 0,936[. Par exemple, le nombre 0,394 code cette séquence. Il requiert 9 bits pour son codage (car $2^8 \leq 394 \leq 2^9$), soit : 9/5 = 1,8 bits/symbole.

Codage arithmétique

Algorithme de décodage de la séquence

1) Initialisation: $L_c = 0$; $H_c = 1$

2) Calculer la longueur du sous-intervalle du code : $H_c - L_c$

3) Trouver le sous-intervalle $[Ls_k, Hs_k[$ du symbole avec $1 \leq k \leq N$

tel que: $Ls_k \leq (\text{mot-code} - L_c) / \text{longueur} < Hs_k$

4) Obtention du symbole: s_k

5) Mettre a jour le sous-intervalle:
$$\begin{cases} L_c = L_c + \text{longueur} \times Ls_k \\ H_c = L_c + \text{longueur} \times Hs_k \end{cases}$$

6) Répéter les étapes 2, 3, 4 et 5 jusqu'à obtenir le décodage de tous les symboles de la séquence

La figure ci-dessus présente l'algorithme de décodage pour un mot-code obtenu après un codage arithmétique. On se propose d'appliquer cet algorithme à l'exemple précédent. On considère le mot-code '$M_c = 0{,}394$' et on souhaite savoir à quelle séquence il correspond :

[Début Algorithme]

Étape 1 :

On initialise les bornes des sous-intervalles $[L_c, H_c]$: $L_c = 0$ et $H_c = 1$.

Étape 2 :

On calcule la longueur L du sous-intervalle : $L = H_c - L_c = 1$.

Étape 3 :

On calcule le nombre $(M_c - L_c) / L = 0{,}394$, et on cherche k tel que ce nombre soit compris dans la partition initiale $[Ls_k, Hs_k[$.
Dans le cas présent, il s'agit de l'intervalle [0,3 ; 0,7[qui est aussi la partition initiale $[Ls_3, Hs_3[$.

Étape 4 :

Le ***premier symbole*** de la séquence est le ***symbole s_3*** (***vecteur de mouvement 0***).

Étape 5 :

On met à jour le sous-intervalle $[L_c, H_c[$ comme au moment du codage, de manière à décoder le second symbole :

$$L_c = L_c + L \times Ls_3 = 0 + 1 \times 0,3 = 0,3$$
$$H_c = L_c + L \times Hs_3 = 0 + 1 \times 0,7 = 0,7$$

Étape 2 :

On calcule la longueur L du sous-intervalle : $L = H_c - L_c = 0,7 – 0,3 = 0,4$.

Étape 3 :

$(M_c - L_c) / L = (0,394 – 0,3) / 0,4 = 0,235$.
Ce nombre est compris dans la partition initiale $[Ls_2, Hs_2[= [0,1 ; 0,3[$.

Étape 4 :

Le ***second symbole*** de la séquence est le ***symbole s_2*** (***valeur -1***).

Étape 5 :

$$L_c = L_c + L \times Ls_2 = 0,3 + 0,4 \times 0,1 = 0,34$$
$$H_c = L_c + L \times Hs_2 = 0,3 + 0,4 \times 0,3 = 0,42$$

Étape 2 :

On calcule la longueur L du sous-intervalle : $L = H_c - L_c = 0,42 – 0,34 = 0,08$.

Étape 3 :

$(M_c - L_c) / L = (0,394 – 0,34) / 0,08 = 0,675$.
Ce nombre est compris dans la partition initiale $[Ls_3, Hs_3[= [0,3 ; 0,7[$.

Étape 4 :

Le ***troisième symbole*** de la séquence est le ***symbole s_3*** (***valeur 0***).

Étape 5 :

$$L_c = L_c + L \times Ls_3 = 0,34 + 0,08 \times 0,3 = 0,364$$
$$H_c = L_c + L \times Hs_3 = 0,34 + 0,08 \times 0,7 = 0,396$$

Étape 2 :

On calcule la longueur L du sous-intervalle : $L = H_c - L_c = 0,396 – 0,364 = 0,032$.

Étape 3 :

(M_c - L_c) / L = (0,394 – 0,364) / 0,032 = 0,93.
Ce nombre est compris dans la partition initiale [Ls_5, Hs_5[= [0,9 ; 1[.

Étape 4 :

Le ***quatrième symbole*** de la séquence est le ***symbole s_5*** (***valeur 2***).

[Fin Algorithme]

Le décodage du nombre 0,394 aboutit à la séquence originale $\{s_{\alpha 1}, s_{\alpha 2}, s_{\alpha 3}, s_{\alpha 4}\} = \{s_3, s_2, s_3, s_5\} = \{ 0, -1, 0, 2 \}$.

Contrairement au codage de Huffman, le codage arithmétique permet de coder sur un nombre de bits non nécessairement entier. Cet avantage permet de réaliser une compression plus performante.
Ces mêmes performances peuvent être améliorées dès lors que les tables de probabilités ne sont pas fixées (comme c'est le cas dans l'exemple présenté), mais adaptées à la séquence courante et fonction des séquences précédentes. On parle alors de codage arithmétique adaptatif.