Huffman

Auteurs : Quentin Januel, Loïc Mohin et Anthony Villeneuve
Mentors : Olivier Gipouloux et Stéphane Gaussent

Fait à : Université Jean Monnet, Saint Etienne

24 février 2019

Résumé

Dans ce rapport, on se propose d'étudier le codage d'un texte en binaire par les arbres de Huffman. Nous verrons pourquoi la rentabilité de l'algorithme est indépendante de tout choix ainsi que pourquoi ce codage est optimal.

Table des matières

1 Prérequis

Dans cette section, nous allons tâcher de définir les outils dont nous aurons besoin pour l'analyse des arbres de Huffman.

1.1 Alphabet

On appelle Σ un alphabet dont les éléments sont appelés des lettres.
Un mot sur Σ est un n-uplet de lettres : $m = (a_1,\ a_2,\ ...,\ a_n)$.
L'ensemble des mots sur Σ est noté $\Sigma^* := \{m \in \Sigma^n,\ \forall n \in \mathbb{N}\}$.
Soit $m = (a_1,\ a_2,\ ...,\ a_n)$ un mot, on appelle longueur du mot m notée $|m|$ l'entier n.
Enfin, on note ε le mot vide (unique mot de longueur 0).

On peut munir Σ^* d'une loi de composition interne, la concaténation $+$:
$(a_1,\ ...,\ a_n) + (b_1,\ ...,\ b_n) = (a_1,\ ...,\ a_n,\ b_1,\ ...,\ b_n)$.
On observe alors que $(\Sigma^*,\ +)$ est un monoïde.

1.2 Mot pondéré

On dit qu'un élement $(a,\ n) \in \Sigma^* \times \mathbb{N}$ est un mot pondéré de Σ et on notera :

1. $l((a,\ n)) = a$ (l pour "lettre"),
2. $p((a,\ n)) = n$ (p pour "poids")

On définit alors la somme de mots pondérés $x+y = (l(x)+l(y),\ p(x)+p(y))$ et on se retrouve avec un nouveau monoïde : $(\Sigma^* \times \mathbb{N},\ +)$.

1.3 Arbre binaire

Soit E un ensemble, on dit que $A := (Q,\ T)$ est un arbre binaire sur E avec $Q \subset E$ et $T \subset E \times \mathbb{F}_2 \times E$ s'il respecte les 3 propriétés suivantes :

1. $\exists\ !\ r \in Q,\ \forall (x,\ b) \in Q \times \mathbb{F}_2,\ (x, b, r) \notin T$ (r est appelée racine de A notée $r(A)$),
2. $\forall x_2 \in Q \backslash \{r\},\ \exists\ !\ (x_1,\ b) \in Q \times \mathbb{F}_2,\ (x_1,\ b,\ x_2) \in T$, (on dit que x_1 est un parent de x_2 que l'on note $\pi(x_2)$),
3. $\forall (x_1,\ b) \in Q \times \mathbb{F}_2,\ \text{card}(\{x_2 \in Q,\ (x_1,\ b,\ x_2) \in T\}) \leq 1$.

Les éléments de Q (notés $q(A)$) sont appelés les états et les éléments de T (notés $t(A)$) transitions.
Enfin, l'ensemble des arbres binaires sur E est noté $\mathcal{A}_E$.

2 Arbre de Huffman

Si l'on prend un texte quelconque et que l'on compte le nombre d'occurrences de chaque lettre afin de se retrouver avec une liste de lettres pondérées, considérées comme des arbres binaires à un état, on peut alors construire l'arbre de Huffman de ce texte en fusionnant à chaque étape les 2 arbres dont les poids des racines sont minimum jusqu'à ne se retrouver qu'avec un seul arbre.
Une fusion consiste à rajouter une racine tel que son fils gauche soit l'un des deux arbres et son fils gauche l'autre arbre.

On peut ensuite encoder le texte en suivant le chemin depuis la racine jusqu'à chaque lettre en ajoutant un 0 par transition à gauche et un 1 pour la droite.
Un tel arbre n'est pas unique car on peut choisir les arbres à poids minima s'il y en a plusieurs, de plus il n'y a pas de restriction sur quel arbre mettre à gauche ou à droite au moment de fusionner.
L'objectif sera donc de montrer que ces choix n'affectent pas le nombre de bits nécessaires pour l'encodage.

2.1 Définition

Tâchons d'abord de définir quels arbres parmi les arbres binaires sont des arbres de Huffman.
Prenons un alphabet Σ quelconque. Tout arbre de la forme

$$(\{x\},\ \emptyset),\ x \in \Sigma^* \times \mathbb{N},\ |l(x)| = 1$$

sera appelé arbre de Huffman sur Σ.
De plus, soient A et B deux arbres de Huffman et $r := r(A) + r(B)$ de telle sorte que

$$q(A) \cap q(B) \cap \{r\} = \emptyset$$

Alors on pose

$$M_{A,\ B} := (q(A) \cup q(B) \cup \{r\},\ t(A) \cup t(B) \cup \{(r,\ 0,\ r(A)),\ (r,\ 1,\ r(B))\})$$

qui est également un arbre de Huffman et on dit que M est la fusion de A et de B.
Notons $\mathcal{H}_\Sigma$ l'ensemble des arbres de Huffman sur Σ.
On pose aussi

$$\begin{array}{rccc} m: & \mathcal{H}_\Sigma \times \mathcal{H}_\Sigma & \to & \mathcal{H}_\Sigma \\ & (A,\ B) & \mapsto & M_{A,\ B} \end{array}$$

2.2 Proposition 1

Pour tout alphabet Σ, on a $\mathcal{H}_\Sigma \subset \mathcal{A}_{\Sigma^* \times \mathbb{N}}$.

Preuve :

Pour les arbres de Huffman de la forme $(\{x\},\ \emptyset)$:

1. La racine est x et est bien unique (aucune transition de la forme $(.,.,x)$ vu que l'ensemble des transitions est vide).
2. Aucun état n'est pas la racine donc la propriété est trivialement vérifiée.
3. L'ensemble des transitions étant vide, le cardinal sera toujours inférieur à 1.

Pour les arbres de Huffman de la forme $M_{A,\ B}$:

1. $r(A)$, $r(B)$ et r sont les seuls candidats pour être des racines (car A et B sont des arbres binaires et n'ont qu'une seule racine).
   Or on a $(r,\ 0,\ r(A))$ et $(r,\ 1,\ r(B))$ des transitions donc $r(A)$ et $r(B)$ ne sont pas racines. La racine r est bien unique.
2. Soit x un état n'étant pas la racine, il appartient donc soit à $q(A)$, soit à $q(B)$. S'il n'est pas racine d'un de ces sous arbres, alors la propriété est vérifiée, sinon les transitions $(r,\ 0,\ r(A))$ et $(r,\ 1,\ r(B))$ remplissent la propriété.
3. La propriété est déjà vérifiée pour tout état qui n'est pas r, or seulement deux transitions sont rajoutées et on a

   $$\text{card}(\{x \in Q,\ (r,\ 0,\ x) \in T\}) = \text{card}(\{x \in Q,\ (r,\ 1,\ x) \in T\}) = 1 \leq 1$$

□

2.3 Classification des arbres de Huffman

L'objectif est de construire des classes d'équivalence d'arbres de Huffman selon le nombre de bits qu'ils encodent.

L'ensemble des feuilles d'un arbre binaire A est

$$\mathcal{F}_A := \{x_1 \in q(A),\ \forall (x_1,\ b) \in q(A) \times \mathbb{F}_2,\ (x_1,\ b,\ x_2) \notin t(A)\}$$

Soit ω une fonction qui à un arbre lui associe cette longueur d'encodage, on a par définition

$$\begin{array}{rccc} \omega : & \mathcal{H}_\Sigma & \to & \mathbb{N} \\ & A & \mapsto & \displaystyle\sum_{x \in \mathcal{F}_A} n_A(x) \times p(x) \end{array}$$

où $n_A(x)$ est la profondeur de x dans l'arbre A, c'est-à-dire l'entier n tel que $\pi^n(x) = r(A)$.
On définit la relation d'équivalence

$$A\mathcal{R}B \Longleftrightarrow \omega(A) = \omega(B),\ \forall A,\ B \in \mathcal{H}_\Sigma$$

On dénote également $\overline{\mathcal{H}_\Sigma} := \mathcal{H}_\Sigma/\mathcal{R}$ l'ensemble quotient de $\mathcal{H}_\Sigma$ par $\mathcal{R}$.

2.4 Lemme

$\forall A,\ B \in \mathcal{H}_\Sigma,\ \omega \circ m(A,\ B) = \omega(A) + p \circ r(A) + \omega(B) + p \circ r(B)$

Preuve :
Soient A et B deux arbres de Huffman,

$$\begin{aligned}
\omega \circ m(A,\ B) &= \sum_{x \in \mathcal{F}_{m(A,\ B)}} n_{m(A,\ B)}(x) \times p(x) \\
&= \sum_{x \in \mathcal{F}_A} (n_A(x) + 1) \times p(x) + \sum_{x \in \mathcal{F}_B} (n_B(x) + 1) \times p(x) \\
&= \sum_{x \in \mathcal{F}_A} n_A(x) \times p(x) + \sum_{x \in \mathcal{F}_A} p(x) + \sum_{x \in \mathcal{F}_B} n_B(x) \times p(x) + \sum_{x \in \mathcal{F}_B} p(x) \\
&= \omega(A) + p \circ r(A) + \omega(B) + p \circ r(B)
\end{aligned}$$

□

2.5 Proposition 2

$$\forall A \in \mathcal{H}_\Sigma,\ \omega(A) = \sum_{(x_1,\ b,\ x_2) \in t(A)} p(x_2)$$

Preuve :
Comme chaque arbre de Huffman peut s'exprimer de la forme $m(m(..., ...), m(..., ...))$ jusqu'à se retrouver avec des arbres tels $(\{x\},\ \emptyset)$, procédons par récurrence sur la fonction fusion.
Soit $A := (\{x\},\ \emptyset)$, alors

$$\omega(A) = 0 = \sum_{(x_1,\ b,\ x_2) \in t(A)} p(x_2)$$

La propriété est bien vérifiée.
Prenons maintenant deux arbres A et B tels que la propriété soit vraie,

vérifions la pour $m(A,\ B)$:

$$\begin{aligned}\omega \circ m(A,\ B) &= \omega(A) + \omega(B) + p \circ r(A) + p \circ r(B) \\ &= \sum_{(x_1,\ b,\ x_2) \in t(A)} p(x_2) + \sum_{(x_1,\ b,\ x_2) \in t(B)} p(x_2) + \sum_{(x_1,\ b,\ x_2) \in \{(r,\ 0,\ r(A)),\ (r,\ 1,\ r(B))\}} p(x_2) \\ &= \sum_{(x_1,\ b,\ x_2) \in t \circ m(A,\ B)} p(x_2)\end{aligned}$$

La propriété est donc vraie pour tout arbre de Huffman. □

2.6 Théorème

Pour tout mot de Σ, l'arbre de Huffman associé est unique dans $\overline{\mathcal{H}_\Sigma}$.

Preuve :

Prenons un mot de Σ^*, l'algorithme de Huffman nous dit de considérer l'ensemble des arbres de Huffman de la forme$(\{(a,\ n)\},\ \emptyset)$ avec a les lettres du mot et n leur nombre d'occurrences.

Ensuite, étant donnée un ensemble de n arbres, il faut considérer l'ensemble de $n-1$ arbres où l'on a fusionné les deux arbres dont les poids des racines sont minimum, et ce jusqu'à ne se retrouver qu'avec un seul arbre.

Or ces arbres ne sont pas forcément uniques et de plus $m(A,\ B) \neq m(B,\ A)$.

Observons ce que devient la somme des longueurs d'encodage ω d'un ensemble d'arbres X après une itération de l'algorithme.

Soient A et B deux arbres de X dont les poids des racines sont minimum. La nouvelle liste sera donc $Y := X \backslash \{A,\ B\} \cup \{m(A,\ B)\}$.

$$\begin{aligned}\sum_{C \in Y} \omega(C) &= \omega \circ m(A,\ B) + \sum_{C \in X \backslash \{A,\ B\}} \omega(C) \\ &= \omega(A) + \omega(B) + p \circ r(A) + p \circ r(B) + \sum_{C \in X \backslash \{A,\ B\}} \omega(C) \\ &= p \circ r(A) + p \circ r(B) + \sum_{C \in X} \omega(C)\end{aligned}$$

On constate que la somme des poids a augmenté de $p \circ r(A) + p \circ r(B)$ ce qui est indépendant de l'ordre de la fusion entre A et B ainsi que le choix des arbres aux poids minimum parmi tous les arbres de l'ensemble.

Ainsi si l'algorithme a donné pour un mot m deux arbres A et B différents, on aura quand même $\omega(A) = \omega(B) \iff A \mathcal{R} B$ ce qui montre qu'ils

appartiennent à la même classe d'équivalence dans $\overline{\mathcal{H}_\Sigma}$. $\square$

Profitons en pour poser $h : \Sigma^* \to \overline{\mathcal{H}_\Sigma}$ la fonction qui à un mot de Σ y associe la classe d'équivalence dans $\overline{\mathcal{H}_\Sigma}$ des arbres de Huffman construits par l'algorithme.

3 Utilisation concrète

L'intérêt même des arbres de Huffman est de compresser un message. Considérons le message $m := "caba" \in \Sigma^* := a,\ b,\ c$ ainsi que son arbre de Huffman associé $A := h(m) \in \overline{\mathcal{H}_\Sigma}$. Cette section a alors pour but d'expliquer comment utiliser A pour encoder m.

3.1 Encodage de l'arbre lui-même

Afin de pouvoir reconstituer le message, il faut tout d'abord trouver un moyen d'encoder l'arbre de Huffman du message en binaire. Cette tâche est simplifiée de par la définition d'arbre binaire donnée plus haut : supposons que l'arbre A se note

$$
\begin{aligned}
A = (&\\
&\{(a,\ 2),\ (b,\ 1),\ (c,\ 1),\ (bc,\ 2),\ (abc,\ 4)\},\\
&\{((abc,\ 4),\ 0,\ (a,\ 2)),\ ((abc,\ 4),\ 1,\ (bc,\ 2)),\ ((bc,\ 2),\ 0,\ (b,\ 1)),\ (bc,\ 2),\ 1,\ (c,\ 1))\}\\
&)
\end{aligned}
$$

On voit alors que l'arbre n'est défini que par une succession de caractères, mieux encore il peut être simplifié pour une écriture plus courte mais néanmoins toujours suffisante : il est bon de noter que seules les transitions donnent toute l'information nécessaire de l'arbre, que les poids sont inutiles ainsi que les mots ayant une longueur supérieure à 1. De plus, la justesse de la syntaxe mathématique n'est ici pas importante.
Remplaçons donc chaque mot de longueur non 1 par une lettre quelconque qui n'est pas encore utilisée (ici x pour abc et y pour ab). Mettons toutes les transitions gauches du côté gauche et idem pour la droite. Enfin, séparons chaque symbole par une virgule, chaque transition par 2 virgules et les deux côtés par 3 virgules. Pour l'exemple, le même arbre donnerait ceci :

$$x, a, , y, b, , , x, y, , y, c$$

On peut alors utiliser la table ASCII afin d'encoder un tel arbre. Pour séparer l'arbre du début du message il est possible de rajouter à nouveau 3 virgules à la fin de l'arbre.

Note Il serait aussi possible de créer l'arbre de Huffman basé sur les fréquences d'appartition des lettres de l'alphabet dans la langue française, l'avantage étant qu'il n'y aurait alors plus besoin d'encoder l'arbre pour chaque message. Seulement la compression ne sera pas optimale et suppose la langue du message.

3.2 Encodage du message en utilisant l'arbre

Remplaçons chaque lettre par la succession des transitions partant de la racine jusqu'à la feuille correspondante à la lettre. Une transition sera encodée par un 0 si elle va à gauche, et par un 1 si elle va à droite. L'algorithme de décodage ayant connaissance de l'arbre, il sait quand se termine la définition d'un caractère lorsqu'il arrive sur une feuille.
En utilisant l'arbre A tel décrit dans ci-dessus, le message serait alors encodé par

11 0 10 0

4 Performance

4.1 Performance d'un encodage ASCII

WIP

4.2 Performance d'un encodage par les arbres de Huffman

WIP

5 Optimalité

5.1 Complexité de Kolmogorov

WIP

5.2 Entropie

WIP