Algorithmes de Markov

Joran Bigalet

Décembre 2010

Résumé

Un *Algorithme de Markov* est un système à file d'attente de réécriture de chaine de caractères. Cela consiste à transformer une chaine de caractères, dans un certain alphabet, en une autre, et cela par une série de règles définies au préalable.

Table des matières

1 Définitions

Dans cette section, nous allons introduire la définition formelle des Algorithmes de Markov, pour ensuite se concentrer sur quelques exemples.

1.1 Définition formelle

Définition 1. Un *algorithme de Markov* est défini par (Σ, P, F), où
- Σ est un alphabet ;
- P est un ensemble fini et ordonné de règles de transformation :
$$\alpha_1 \to \beta_1, \ldots, \alpha_k \to \beta_k,$$
avec $\alpha_i, \beta_i \in \Sigma^*$;
- F est un sous-ensemble de P représentant les règles terminales, que l'on notera par convention $\alpha \to. \beta$.

Avec pour entrée un mot u de Σ^*, on parcours dans l'ordre l'ensemble P, à la recherche d'une règle r_i ayant un α_i appartenant à u. L'algorithme s'arrete si aucune règle satisfaisante n'est trouvée. Sinon, on remplace α_i par β_i dans u. Si r_i était terminale, l'algorithme s'arrete, sinon, on recommence depuis le départ.

1.2 Quelques exemples

Exemple 2. Algorithme de transformation binaire en unaire : $\mathbf{A} = (\Sigma, P, F)$, avec

- Σ ={'0', '1', '|'}
- Règles de P :
  - p_1 : '|0' → '0||' ;
  - p_2 : '1' → '0|' ;
  - p_3 : '0' → ε ;
- $F = \phi$.

(On peut remarquer qu'il n'y a ici aucune règle terminale, mais l'on pourrait rajouter une règle p_4 : '|' → '|' terminale, avec ainsi $F = \{p_4\}$).
L'application de cet algorithme à un nombre binaire résulte en sa réécriture en base unaire.
Par exemple, pour u = '101', on obtient :

'101'→'0|01'→'00||1'→'00||0|'→'00|0|||'→'000|||||'→'00|||||'→'0|||||'→'|||||'.

Exemple 3. Effet miroir sur un mot $\mathbf{B} = (\Sigma, P, F)$, avec

- Σ ={'a', 'b', 'X', 'Y'}
- Règles de P :
  - p_1 : 'XX' → 'Y' ;
  - p_2 : 'Ya' → 'aY' ;
  - p_3 : 'Yb' → 'bY' ;
  - p_4 : 'YX' → 'Y' ;
  - p_5 : 'Y' →. ε ;
  - p_6 : 'Xaa' → 'aXa' ;
  - p_7 : 'Xab' → 'bXa' ;
  - p_8 : 'Xba' → 'aXb' ;
  - p_9 : 'Xbb' → 'bXb' ;
  - p_{10} : ε → 'X' ;
- $F = \{p_5\}$.

L'application de cet algorithme à un mot $u \in \{a, b\}^*$ renvoie son miroir.
Par exemple, pour u = 'abbab', on obtient :

'abbab'→'Xabbab'→'bXabab'→'bbXaab'→'bbaXab'→'bbabXa'→'XbbabXa'
→'bXbabXa'→'baXbbXa'→'babXbXa'→'XbabXbXa'→'aXbbXbXa'
→'abXbXbXa'→'XabXbXbXa'→'bXaXbXbXa'→'XbXaXbXbXa'
→'XXbXaXbXbXa'→'YbXaXbXbXa'→'bYXaXbXbXa'→'bYaXbXbXa'
→'baYXbXbXa'→'baYbXbXa'→'babYXbXa'→'babYbXa'→'babbYXa'
→'babbYa'→'babbaY'→'babba'.

Exemple 4. Calcul du PGCD de deux nombres unaires $\mathbf{B} = (\Sigma, P, F)$, avec

- Σ ={'|', 'X', 'Y', 'Z', '-'}
- Règles de P :
  - p_1 : '|X' → 'X|' ;
  - p_2 : '|-|' → 'X-' ;
  - p_3 : '|-' → '-Y' ;
  - p_4 : 'Y' → '|' ;
  - p_5 : 'X' → 'Z' ;
  - p_6 : 'Z' → '|' ;
  - p_7 : '-' → ε ;
- $F = \phi$.

L'application de cet algorithme à deux nombre a et b encodé en unaire sous la forme "$\underbrace{||\ldots|}_{a}\text{-}\underbrace{||\ldots|}_{b}$" renvoie le pgcd de a et de b sous forme unaire : " $\underbrace{||\ldots|}_{pgcd(a,b)}$ ".

Par exemple, pour $a = 4$ et $b = 6$, on a u = '||||-||||||' et on obtient :

'||||-||||||'→'|||X-|||||'→'||X|-|||||'→'|X||-|||||'→'X|||-|||||'→'X||X-||||'→'X|X|-||||'
→'XX||-||||'→'XX|X-|||'→'XXX|-|||'→'XXXX-||'→'ZXXX-||'→'ZZXX-||'
→'ZZZX-||'→'ZZZZ-||'→'|ZZZ-||'→'||ZZ-||'→'|||Z-||'→'||||-||' →'|||X-|'
→'||X|-|'→'|X||-|'→'X|||-|'→'X||X-'→'X|X|-'→'XX||-'→'XX|-Y'
→'XX-YY'→'XX-|Y'→'XX-||'→'ZX-||'→'ZZ-||'→'|Z-||'→'||-||'→'|X-|'
→'X|-|'→'XX-'→'ZX-'→'ZZ-'→'|Z-'→'||-'→'|-Y'→'-YY'→'-|Y'→'-||'→'||'.

2 Equivalence avec les machines de Turing

Intuitivement, une machine de Turing est plus agile qu'un algorithme de Markov, dans le sens que le déplacement de la "tête d'écriture et de lecture" peut se faire aussi bien à droite qu'à gauche, tandis-que les algorithmes de Markov ne permettent qu'un accès de gauche à droite. Il est facile de comprendre qu'une machine de Turing puisse simuler un algorithme de Markov.

2.1 Simulation d'un algorithme de Markov par une machine de Turing

Proposition 5. *Tout algorithme de Markov peut être simulé par une machine de Turing.*

Preuve. Pour un algorithme de Markov donné, il suffit simplement de simuler chaque règle $[A \to B]$ par un ensemble d'états representant 4 étapes de traitement :

- La reconnaissance de A dans le mot ;
- Le décalage de certaines parties du mot en vue du remplacement de A par B ;
- Le remplacement de A par B ;
- Le passage à l'état suivant (et donc à la règle suivante), ou simplement la mise en place d'un état final si la règle était finale.

Ce sens de l'équivalence étant très intuitif, il est plus interessant de se pencher sur la réciproque.

2.2 Simulation d'une machine de Turing par un algorithme de Markov

Proposition 6. *Toute machine de Turing peut être simulée par un algorithme de Markov.*

Preuve. Pour une machine de Turing déterministe $MT = (Q, \Sigma, E, q_0, F)$ donnée, on construit un algorithme de Markov $AM = (\Sigma', P, F')$:

- On choisit a, b, f, des caractères vérifiants $a, b, f \notin \Sigma$, qui nous permetteront d'annoter les états de MT dans AM.
  Pour chaque état $q_i \in Q$, on peut par exemple définir $n_i \in \{a, b\}*$ comme l'encodage de i en binaire dont les '0' ont été remplacés par a, et les '1' par des b.
- $\Sigma' = \Sigma \cup \{a, b, f\}$
- On défini les règles de P (dans l'ordre de définition) :
  - $f \to n_0$;
  - Pour chaque transition $t_i = (q_a, x, q_b, y, d) \in E$:
    Si $d = \square$, $xn_a \to yn_b$,
    Si $d = \leftarrow$, $xn_a \to n_b y$,
    Si $d = \rightarrow$, $\forall s \in \Sigma$, $xn_a s \to ysn_b$.
- Enfin, pour chaque état $q_i \in F$, on ajoute à la fin de P et de F' : $n_i \to . \varepsilon$.

Pour un mot $M = cM'$ d'entrée quelquoncque de MT, avec c le premier caractère de M, on obtient $M'' = cfM'$ le mot d'entrée de AM.
La chaine de caractères de type n_i apparait une et une seule fois dans M'' après la première étape. Sa position indique la position de la "tête de lecture/écriture", et son type indique l'état courant.
La machine de Turing MT étant déterministe, il n'existe au plus qu'une transition par couple (q, x), $q \in Q, x \in \Sigma$, ce qui implique qu'il n'y a aucun conflit dans l'ordre des règles de AM. A chaque étape de l'algorithme de Markov AM, on a alors une règle applicable (terminale ou non), correspondant à la position de la "tête de lecture/écriture" et à l'état courant, qui reproduit bien les mêmes calculs que la machine de Turing MT.
De plus, toute machine de Turing étant équivalente à une machine de Turing déterministe, on a bien montré que toute machine de Turing pouvait être simulée par un algorithme de Markov.