3IF TP PageRank

2018

1 Introduction

Une page web est importante si des pages web importantes pointent vers elle. C'est sur cette idée que Brin et Page ont construit l'algorithme PageRank à la fin des années 90 — un brevet a été déposé en 1998 et validé en 2001.

Soit $\mathcal{P}$ un ensemble de pages dont on souhaite évaluer l'importance sous la forme d'un score attribué à chaque page. On considère le graphe orienté $G = (\mathcal{P}, A)$ contenant un sommet pour chaque page $P_i \in \mathcal{P}$, et un arc $(P_i, P_j) \in A$ pour chaque hyperlien de la page P_i pointant sur la page P_j. L'ensemble $pred(P_i)$ des prédécesseurs de P_i dans G correspond à l'ensemble des pages pointant (*backlink*) sur P_i (et le demi-degré intérieur de P_i, $d^{\circ -}(P_i)$, est la cardinalité de cet ensemble), tandis que l'ensemble $succ(P_i)$ des successeurs de P_i dans G correspond à l'ensemble des pages sur lesquelles P_i pointe (et le demi-degré extérieur de P_i, $d^{\circ +}(P_i)$, est la cardinalité de cet ensemble).

L'objectif de ce TP est de calculer pour chaque page $P_i \in \mathcal{P}$ un score $r(P_i)$ traduisant l'importance de P_i. Une première possibilité consiste à définir ce score par la somme des importances des pages pointant sur P_i normalisées par le nombre de liens dans chacune de ces pages :

$$r(P_i) = \sum_{P_j \in pred(P_i)} \frac{r(P_j)}{d^{\circ +}(P_j)}$$

Pour passer de cette définition à un algorithme, on propose de la transformer en une relation de récurrence : initialement, toutes les pages ont le même score $\frac{1}{|\mathcal{P}|}$; à chaque étape, on applique la définition précédente pour modifier le score de chaque page en fonction des scores des pages pointant sur elle.

$$r_0(P_i) = \frac{1}{|\mathcal{P}|}$$
$$r_{k+1}(P_i) = \sum_{P_j \in pred(P_i)} \frac{r_k(P_j)}{d^{\circ +}(P_j)}$$

Sous cette forme, on voit que l'idée de normaliser par $d^{\circ +}(P_j)$ est guidée par l'intention de conserver l'importance au fil des itérations.

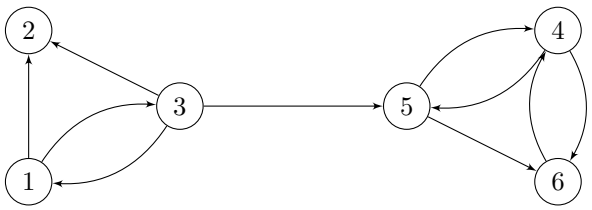
Deux remarques :

— *A priori*, la convergence de ce processus n'est pas assurée.

— S'il converge, les scores des pages dans la configuration stable seront-ils fidèles à la notion subjective de l'importance au principe de cette dérivation ?

2 Exemple

Voici un exemple de graphe avec sa matrice d'adjacence (M). On introduit également la matrice de marche aléatoire correspondante (H) : cette matrice est obtenue à partir de M en divisant chaque coefficient $M[i][j]$ par $d^{\circ+}(i)$.



$$M = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$H = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

3 Expression matricielle de l'algorithme itératif

Exercice : Exprimer sous la forme du produit d'un vecteur par une matrice l'algorithme itératif décrit par la relation de récurrence introduite plus haut.

Exercice : Voir le rapport entre l'itération n de l'algorithme et H^n.

4 Implémentation

Pour l'implémentation, vous partirez d'un code source fourni [1]. Vous sont données : des structures pour représenter vecteurs et matrices, des fonctions pour allouer et libérer la mémoire pour ces structures, des fonctions pour lire et écrire la matrice d'un graphe dans un format textuel. Le graphe exemple introduit plus haut est stocké dans le fichier `g.dat`.

Remarque : le code source fourni comprend également des structures de données pour gérer des matrices creuses (i.e., remplies majoritairement avec des zéros) : dans ce cas, il est plus efficace de considérer une représentation du graphe sous forme de listes d'adjacence.

Exercice : Implémenter l'algorithme.

1. `http://peportier.me/teaching_2017_2018/`

5 Nœuds absorbants

Pour l'instant, l'algorithme ne converge pas de manière satisfaisante. On observe l'effet du nœud 2, dit absorbant car il ne redistribue pas son importance. Autrement dit, H n'est pas stochastique car certaines de ses lignes ne correspondent pas à un vecteur de probabilité (i.e., la somme des valeurs de la ligne est différente de 1).

Exercice : Transformer la matrice H en une matrice S stochastique en changeant le moins possible le sens de H (i.e., une matrice représentant une marche aléatoire en suivant les hyperliens sur le graphe des pages web).

6 Ergodicité

La convergence n'est toujours pas satisfaisante. Pour mieux analyser le phénomène, on s'intéresse aux questions suivantes.

Exercice : À quoi correspondent les puissances de la matrice d'adjacence M du graphe G ? Par exemple, que représentent les valeurs sur la diagonale principale de M^3 ? Comment interpréter les puissances de la matrice stochastique S ?

On dit d'une matrice stochastique représentant un graphe qu'elle est *ergodique* si, pour deux nœuds quelconques, la probabilité de rejoindre le second en partant du premier en un nombre fini d'étapes n'est pas nulle.

Pour rendre la matrice S ergodique, Brin et Page ont introduit la métaphore du *surfeur aléatoire*. Le surfeur suit la structure des hyperliens du web mais il peut parfois saisir une URL dans la barre de navigation, se téléportant ainsi vers une page quelconque non nécessairement connectée à la page courante. Pour la téléportation, chaque page web cible est équiprobable.

Exercice : Pour la rendre ergodique, faire une combinaison convexe de S avec une matrice de rang 1 qui représente la téléportation :

$$E = \alpha S + (1 - \alpha)\frac{1}{n}ee^T$$

Avec $0 \leq \alpha \leq 1$ et e le vecteur unité de dimension n.

Exercice : L'algorithme semble-t-il converger de manière satisfaisante ?

7 Bénéficier du fait que H soit une matrice creuse

La matrice H est extrêmement creuse, car chaque page ne contient généralement que peu d'hyperliens. Autrement dit, le demi-degré intérieur de chaque sommet du graphe est très petit par rapport au nombre de sommets. La matrice E, ergodique, est dense. Pour que le calcul de la distribution stationnaire sur E puisse bénéficier du fait que H soit creuse, on exprime E en fonction de H. Dans le calcul ci-dessous, a est un vecteur indicateur des nœuds absorbants (il possède des zéros partout sauf aux indices des nœuds absorbants).

$$
\begin{aligned}
E &= \alpha S + (1-\alpha)\frac{1}{n}ee^T \\
&= \alpha\left(H + \frac{1}{n}ae^T)\right) + (1-\alpha)\frac{1}{n}ee^T \\
&= \alpha H + (\alpha a + (1-\alpha)e)\frac{1}{n}e^T
\end{aligned}
$$

Ainsi, à chaque itération de l'algorithme, le vecteur des scores π peut être mis à jour de la manière suivante :

$$
r_{k+1}^T = \alpha r_k^T H + (\alpha r_k^T a + 1 - \alpha)\frac{1}{n}e^T
$$

Exercice : Écrire une version de l'algorithme prenant en compte le caractère creux de H, en utilisant une représentation par listes d'adjacence au lieu d'une représentation par matrice d'adjacence. La tester sur `genetic.dat`.