Introduction à l'Analyse Numérique : Travail MatLab

Prof. Q. Louveaux - MATH0006-1

Université de Liège, année académique 2008-2009

Consignes

Le travail s'effectue par groupes de deux. Un rapport écrit est à remettre pour le 17 novembre. Il comprendra au maximum 7 pages, plus éventuellement le code MatLab en annexe. L'évaluation orale aura lieu la semaine du 24 novembre. On vous y demandera d'effectuer une démonstration du code réalisé.

Problème

On propose d'étudier l'algorithme d'analyse de liens "PageRank", employé par Google comme mesure de l'importance relative d'un document dans un réseau d'hyperliens (notamment, l'importance d'une page sur le web).

On peut représenter le réseau analysé par un graphe. Un exemple en est donné à la figure 1. Les noeuds y représentent les pages web, et les arcs orientés y figurent les hyperliens pointant d'une page à l'autre. Nous y avons également indiqué les résultats du calcul du PageRank pour chaque noeud du graphe. On peut associer à un tel graphe une matrice d'*adjacence* A. Chaque ligne de celle-ci donne le nombre d'arcs *sortant* d'un noeud. Par exemple, l'élément A_{ij} donne le nombre d'arcs pointant du noeud i au noeud j. Voici la matrice d'adjacence A du graphe de la figure 1 et les PageRank $\mathbf{r}$ associés :

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} 0.0350 \\ 0.0273 \\ 0.2930 \\ 0.2763 \\ 0.0273 \\ 0.0273 \\ 0.0273 \\ 0.0813 \\ 0.0964 \\ 0.1091 \end{bmatrix}$$

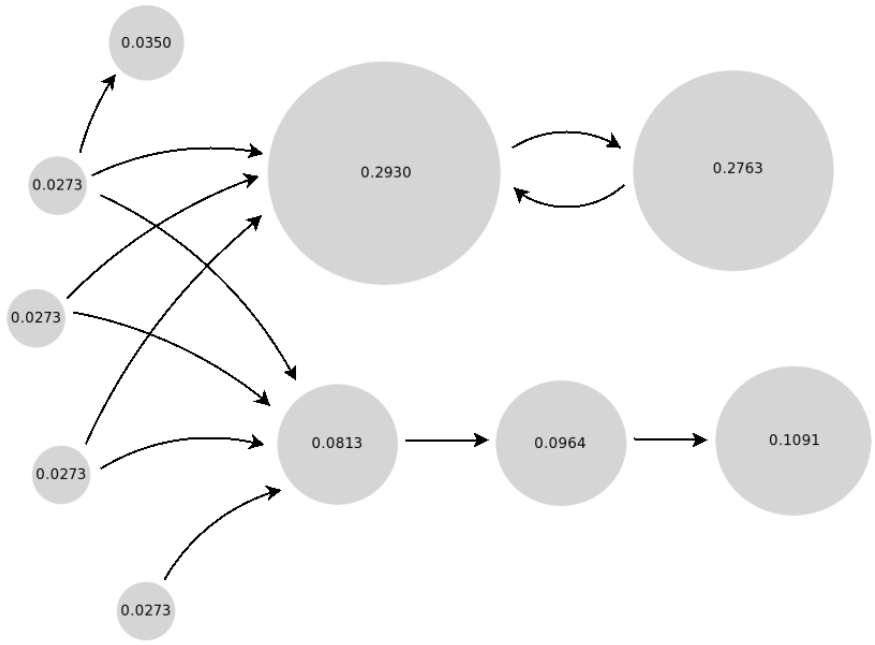

FIG. 1 – Exemple de PageRank d'un graphe

Algorithme

PageRank non amorti

En première approximation, le PageRank de google est calculé comme suit :

– 1. La PageRank d'un document j est une somme pondérée des PageRank des documents i qui pointent vers j.

– 2. Le facteur de pondération est

$$\tilde{A}_{ij} = \frac{\text{nombre de liens de } i \text{ vers } j}{\text{nombre de liens sortant de } i} = \frac{A_{ij}}{\sum_{k=0}^{N-1} A_{ik}}$$

– 3. Les liens d'un document vers lui-même (boucles) ne sont pas comptés :

$$\tilde{A}_{ii} = 0,\ \forall i$$

– 4. Si un document i ne comprend aucun lien sortant, on considère qu'il contient exactement un lien vers tous les autres. Dans ce cas

$$\tilde{A}_{ij} = \frac{1}{N},\ \forall j$$

N étant le nombre de noeuds du graphe.

On peut donc construire la matrice $\tilde{A}$ des coefficients de pondération. Notons que les points 2 à 4 donnent

la propriété suivante :

$$\sum_{j=0}^{N-1} \tilde{A}_{ij} = 1 \ , \ \ \forall i$$

Et du point 1, on peut déduire la relation

$$\mathbf{r} = \tilde{A}^T \mathbf{r} \tag{1}$$

où on recherche le vecteur $\mathbf{r}$.

Ceci revient à trouver le vecteur propre de $\tilde{A}^T$ associé à la valeur propre 1. On peut également démontrer que cette valeur propre existe toujours pour la matrice $\tilde{A}^T$, et qu'il s'agit de la plus grande valeur propre (en module).

PageRank amorti

La matrice $\tilde{A}$ peut être vue comme une matrice stochastique : l'élément $\tilde{A}_{ij}$ indique la probabilité de transition d'un état i à un état j. De ce point de vue, le vecteur $\mathbf{r}$ indique les probabilités des états, à l'état stationnaire (i.e. après un grand nombre de transitions).

Cependant, on peut remarquer que l'équation 1 ne modélise pas la probabilité qu'un utilisateur arrête de surfer et reprenne à une page aléatoire du graphe. Pour remédier à ce problème, on peut introduire un facteur d appelé facteur d'amortissement :

$$\mathbf{r} = \frac{1-d}{N} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + d\tilde{A}^T \mathbf{r} \tag{2}$$

$(1-d)$ représente cette probabilité que l'utilisateur se lasse et interrompe son parcours. On considère alors qu'il choisit une page quelconque du graphe (avec une probabilité uniforme), puis reprend suivant le schéma conventionnel. Ici, nous prendrons toujours $d = 0.85$.

Données

Votre code doit fonctionner pour n'importe quel graphe, donc indépendamment de la matrice d'adjacence utilisée.

Vous pouvez donc choisir cette matrice à votre convenance. Voici par exemple comment générer une matrice carrée de valeurs aléatoires :

```
floor(rand(10) * 8)
```

Ceci donnera une matrice 10×10 de nombres entiers compris entre 0 et 7.

Cependant, il peut être intéressant de tester l'algorithme en situation réelle. Pour celà, nous mettons à votre disposition une application de calcul de graphe à partir de pages web existantes. Sur la page

du travail (`http://www.montefiore.ulg.ac.be/~poirrier`), vous pourrez entrer l'adresse d'un site de votre choix (e.g. `www.wikipedia.org`, ...) et récupérer la matrice d'adjacence d'un graphe contenant ce site et quelques-uns de ses "voisins".

Question 1

Ecrire une fonction qui calcule la matrice d'adjacence modifiée $\tilde{A}$ à partir d'une matrice d'adjacence A donnée. Les fonctions écrites pour les questions suivantes pourront faire appel à cette fonction.

Question 2

Ecrire une fonction qui calcule le PageRank (non amorti) à partir d'une matrice d'adjacence A donnée. Le calcul de la valeur propre dominante doit être effectué par la méthode de la puissance. Vous devez implémenter celle-ci vous-même, éventuellement dans une fonction séparée.
Appliquez votre code à un ensemble de pages donné. Interprétez brièvement les résultats.

Note :
- Vous pouvez vérifier vos résultats à l'aide de la fonction `eig` de MatLab, qui fournit les vecteurs propres et valeurs propres d'une matrice.
- Le vecteur $\mathbf{r}$ devra être normalisé de sorte que $\sum_{i=0}^{N-1} \mathbf{r}_i = 1$.

Question 3

Ecrire une fonction qui, à partir d'une matrice d'adjacence A donnée, calcule le PageRank amorti ($d = 0.85$) de manière directe (sans processus itératif).

Question 4

Exprimer l'équation 2 :

$$\mathbf{r} = \frac{1-d}{N} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + d\tilde{A}^T\mathbf{r}$$

sous forme de problème aux valeurs propres de la forme $\mathbf{r} = M^T\mathbf{r}$, tel que la matrice M soit stochastique (la somme des éléments des lignes vaut 1). De la sorte, il existera une valeur propre 1 qui sera dominante. On pourra donc la calculer par la méthode de la puissance.

Remarque : Puisque le vecteur $\mathbf{r}$ peut être vu comme un vecteur de probabilités, on sait que la somme de ses éléments vaut 1, donc

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \mathbf{r} = 1$$

Question 5

Implémenter le calcul du PageRank amorti par la méthode de la puissance, à l'aide de la formule trouvée à la question précédente.

Question 6

Il est normal que la fonction que vous avez créée pour le calcul du PageRank en tant que problème aux valeurs propres soit moins performante (i.e. plus lente) que la fonction écrite pour la Question 3. En effet, MatLab utilise du code natif (directement compris par l'ordinateur) pour le type d'implémentation utilisé à la Question 3. A l'inverse, vous avez implémenté la méthode de la puissance à l'aide d'un script MatLab (interprété par MatLab avant d'être exécuté sur la machine).

Cependant, MatLab dispose d'une fonction `eigs` utilisant un algorithme dérivé de la méthode de la puissance (l'itération d'Arnoldi).

a. Implémenter le calcul du PageRank amorti à l'aide de la fonction `eigs`.

b. Ecrire une fonction qui compare les temps de calcul des fonctions employées aux questions 3 et 6a. Pour calculer les temps de calcul, vous pouvez utiliser les fonctions `tic` et `toc` ou la fonction `cputime`.

Note : Pour comparer les performances, il est préférable de prendre des matrices de grande taille. On mesure ainsi des temps plus longs, et on obtient des résultats plus précis.

Question 7

a. Ecrire une fonction qui reçoit en entrée une page i et le PageRank $\mathbf{r}_i$ objectif que l'on voudrait atteindre pour cette page et qui calcule en sortie le facteur d d'amortissement qui permet d'atteindre cette valeur $\mathbf{r}_i$.

b. Ecrire une fonction générique qui utilise l'algorithme de la descente du gradient pour optimiser le PageRank de n'importe quelle page. Cette fonction doit prendre en argument
– la matrice d'adjacence A du graphe initial
– un vecteur donnant les indices des pages dont on veut augmenter le PageRank
– un vecteur donnant les indices des pages dont on peut modifier les liens sortants
et doit retourner la matrice d'adjacence *modifiée* $\tilde{A}$ du graphe optimisé. Si on désire augmenter le PageRank de plusieurs pages, la variable à optimiser est la somme de leurs PageRank.

Note : Il sera probablement nécessaire de choisir des matrices plus petites pour pouvoir effectuer le calcul dans des temps raisonnables.
Si vous le désirez, vous pouvez réduire l'ambition en considérant la minimisation d'une fonction à une variable.

Critères d'évaluation

- Le code doit être correct et écrit par vous (ce que nous vérifierons à la présentation orale).
- Le code doit utiliser au maximum les possibilités **vectorielles** de matlab.
  Par exemple, il est préférable d'utiliser la fonction `sum` que de créer une boucle qui calcule la somme des éléments d'un vecteur.
- La **justification** des choix numériques est très importante. Pensez à expliquer les choix qui vous ont semblé cruciaux.
- La clarté du rapport sera aussi évaluée. Le nombre de pages étant limité, vous ne devez par exemple pas répéter l'énoncé. Allez donc à l'essentiel. Veillez également à soigner l'orthographe. Nous donnerons un point (sur 20) supplémentaire aux rapports écrits sans faute. Un point (sur 20) peut être retiré aux rapports dont l'orthographe est indigne d'un rapport professionnel (ceci ne vaut pas pour les groupes composés de deux personnes dont le français n'est pas la langue maternelle).
- Pour obtenir une note supérieure ou égale à 15/20, il faut avoir abordé un des deux points de la Question 7, le point **b.** étant considéré comme plus compliqué.