Algorithme du PageRank

14 Octobre 2021

Pierre-Jean Charpin & José Lagès

*Vous devez rendre votre rapport de TP avant le **9 décembre 2021**. Envoyez le rapport et votre code Python **commentés** à l'adresse mail suivante : pj.charpin@femto-st.fr. L'utilisation de LaTeX est obligatoire. Vos résultats doivent être commentés et illustrés (si possible) par des figures soigneusement réalisées. Votre rapport doit être rédigé de sorte à ce que son poids reste le plus faible possible. Pour ce faire, prévilégiez l'utilisation d'images vectorielles pour vos illustrations (format .pdf, .svg, .eps) qui seront, en principe, beaucoup plus légères que celles "habituelles" aux formats .jpg, .png, .. Si vous avez des questions ou des remarques, n'hésitez pas à m'envoyer un email à l'adresse ci-dessus.*

1 Remise à niveau

Le but de cette section est de faire un rappel, ou bien de découvrir le language Python à travers des exemples simples qui feront utiliser la plupart des outils nécessaires à ce TP.

1.1 Vecteurs et matrices

Le module *numpy* est un module très utile, voir même indispensable de python. Il dispose de nombreuses fonctions très utiles et permet notamment de créer des vecteurs et des matrices (tableaux de nombres).

Exercice : Définir une matrice et un vecteur numpy de la manière suivante :

```
1: import numpy as np
2: A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
3: b = np.array([-3, -2, -1])
```

Exercice : Faire le produit matrice-vecteur Ab à l'aide de la fonction *dot* de *numpy* et affichez le résultat.

1.2 Fonctions

Le language Python propose la possibilité à l'utilisateur de créer des fonctions ayant pour but d'effectuer des opérations sur des variables données en entrée de ces fonctions. La manière de définir une fonction est la suivante (dans cet exemple, la fonction nomDeLaFonction prend en argument 2 variables et en fait la somme) :

```
1: def nomDeLaFonction(var1,var2)
2:     res = var1 + var2
3:     return(res)
```

Exercice : Définir la fonction qui à x fait correspondre $\sin(x)$. L'appliquer au calcul du sinus de π puis au sinus du vecteur $(0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, 2\pi)$.

Un triangle quelconque peut être caractérisé par ses trois sommets $(x_1; y_1)$, $(x_2; y_2)$ et $(x_3; y_3)$. L'aire de ce triangle peut être calculé à l'aide de la formule suivante :

$$\mathrm{Aire} = \frac{1}{2}|(x_1 - x_3)(y_2 - y_1) - (x_1 - x_2)(y_3 - y_1)| \tag{1}$$

Exercice : Ecrire une fonction qui renvoie l'aire d'un triangle dont on donne les coordonnées des trois sommets en argument (par exemple [[0, 0], [0, 1], [1, 0]])

1.3 Tracé de fonctions

Exercice : Importez le module *matplotlib.pyplot* et utiliser la fonction *plot* de ce module pour tracer les graphes des fonctions suivantes. Pour représenter le graphe d'une fonction f sur un intervalle [a,b], il faut tout d'abord la «discrétiser» ; c'est-à-dire définir une liste de points $(x_i, f(x_i))$ qui va permettre d'approcher le graphe de la fonction par une ligne brisée. Bien sur, plus on augmente le nombre de points, plus la ligne brisée est «proche» du graphe (en un certain sens). Plus concrètement, on commence par construire un vecteur X qui discrétise l'intervalle [a,b], en considérant un ensemble de points équidistants dans [a,b]. Pour ce faire on peut utiliser la fonction *linspace* du module *numpy*.

— $f(x) = \cos^2(x)$ avec $x \in [0, 5]$
— Sur la même figure, les fonctions : $f(x) = \cos^2(x)$, $g(x) = \cos(2x)$ et $h(x) = \cos(x^2)$.

Penser à utiliser des symboles différents pour les différentes représentations graphiques et à mettre une légende.

1.4 Diagonalisation de matrice

Le module *linalg* de *numpy* dispose de la fonction *eig* permettant de calculer les valeurs propres et vecteurs propres d'une matrice carrée.

Exercice : Diagonalisez (trouvez les valeurs propres et les vecteurs propres) la matrice M suivante et vérifiez votre résultat à l'aide du site internet Wims.

$$\mathrm{M} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

La fonction *argsort* du module *numpy* permet de déterminer les indices d'une liste de sorte à trier cette liste dans l'ordre croissant. Par exemple, la fonction *argsort* appliquée à la liste L=[3,5,1,8,4] donnera le résultat suivant : [2,0,4,1,3] car l'élément le plus petit de L est 1 qui est situé en troisième position, puis 3 qui est en première position etc..

Exercice : Triez par ordre croissant la liste des valeurs propres de M en utilisant la fonction *argsort*.

Exercice : Que font les commandes *print(M[2, :])* et *print(M[:,1])* ?

1.5 Lecture/Ecriture de fichiers

La fonction *loadtxt* du module *numpy* permet d'extraire les informations présentes dans un fichier texte et de les ranger dans une matrice. A l'inverse, la fonction *savetxt* permet d'écrire dans un fichier texte une variable présente dans votre environnement python.

Exercice : Chargez le fichier *"matrix.txt"* dans une matrice Mat à l'aide de la fonction *loadtxt*, calculez $\mathrm{Mat}^2 + 2\mathrm{Mat} - 3\mathrm{Id}$ et écrivez votre résultat dans un fichier texte *"res.txt"* (les nombres doivent être écrits au format entier).

2 PageRank : Introduction et cas général

2.1 Le vecteur PageRank

Les réseaux sont omniprésents : les interactions sociales, c'est-à-dire les personnes liées par des connaissances, peuvent être modélisées par des réseaux sociaux ; le World Wide Web (WWW) est une collection de myriades de pages web hyperliées ; des milliards de neurones communiquent par des signaux électriques ; dix mille gènes interagissent entre eux ; les secteurs économiques transforment les biens provenant d'autres secteurs et fournissent des biens à d'autres secteurs...

Dans de nombreux cas, les (grandes quantités de) données peuvent être considérées comme un réseau.[a] Par exemple, considérons "Les Misérables" de V. Hugo, et considérons les personnages comme les nœuds d'un réseau. Nous pouvons imaginer de nombreuses façons de relier chaque personnage aux autres. Une façon simple est la suivante : considérons que si un personnage B est cité dans le livre juste après un personnage A alors un lien $A-B$ est créé entre les deux personnages. En suivant cette procédure pour tous les personnages, on construit un réseau non orienté. Si l'on considère que A est cité avant B, on peut ajouter cette information supplémentaire en attribuant une direction au lien qui est maintenant $A \to B$. En suivant cette procédure pour tous les caractères, on construit un réseau dirigé. Si l'on compte le nombre de fois dans le livre où le personnage B est cité juste après A, cela peut servir à donner un poids au lien $A \to B$. En suivant cette procédure pour tous les personnages, on construit un réseau dirigé et pondéré.

Une fois que les données sont modélisées par un réseau, une mesure de centralité permet de déterminer le degré de centralité ou d'importance d'un nœud. Pour un réseau social, une telle mesure nous indique quelle personne est la plus influente. Pour le WWW, elle permet de classer les pages web par pertinence. Pour "Les Misérables", elle indique quel personnage joue le rôle le plus central dans l'histoire. Il existe de nombreuses mesures de la centralité. La plus simple est la centralité de degré. Le degré d'un nœud est le nombre de liens allant vers ou partant de ce nœud. Pour la centralité de degré, les nœuds du réseau sont classés en fonction de leur degré. Selon cette mesure simple, le nœud le plus (moins) important est le nœud le plus (moins) connecté. Notez que la notion d'importance est relative à la mesure de centralité que nous utilisons. Pour deux mesures de centralité différentes, le classement des nœuds, et par conséquent leur importance dans le réseau, peut être différent.

Nous nous concentrerons ici sur la centralité PageRank qui était au cœur du moteur de recherche Google à ses débuts en 1998 [1]. L'algorithme associé est très puissant car il est capable de classer des milliards de nœuds en un nombre raisonnable d'étapes de calcul. Considérons un réseau avec N nœuds qui sont reliés entre eux par ℓ liens. Attribuons un numéro ou une étiquette à chaque nœud, par exemple, 1,2,...,N. La structure du réseau est codée dans la matrice d'adjacence A dont les éléments sont

$$A_{ij} = \begin{cases} 1 & \text{si } j \text{ pointe vers } i \\ 0 & \text{sinon.} \end{cases} \tag{3}$$

Pour un réseau non orienté, la matrice A est symétrique, c'est-à-dire que $A_{ij} = A_{ji}$, $\forall (i,j)$. Dans leur article fondateur de 1998 [1], Brin et Page, cofondateurs de Google, ont décrit l'algorithme PageRank sans utiliser le formalisme matriciel. Ils ont également omis de parler des chaînes de Markov [2], une classe de modèles stochastiques, introduite en 1906, à laquelle appartient l'algorithme PageRank. La matrice d'adjacence A est convertie en une matrice stochastique S dont les éléments sont

$$S_{ij} = \begin{cases} A_{ij}/k_j^{\text{out}} & \text{si } k_j^{\text{out}} \neq 0 \\ 1/N & \text{sinon.} \end{cases} \tag{4}$$

Ici, $k_j^{\text{out}} = \sum_{i=1}^{N} A_{ij}$ est le degré de sortie du $j$ème nœud. Autrement dit, k_j^{out} est le nombre de façons de quitter le nœud j. Par extension, nous pouvons également définir le degré d'entrée du $i$ème nœud comme $k_i^{\text{in}} = \sum_{j=1}^{N} A_{ij}$. La valeur k_i^{in} donne le nombre de façons de sauter au $i$ème nœud. On observe facilement que pour toute colonne j, $\sum_{i=1}^{N} S_{ij} = 1$, c'est-à-dire que chaque colonne de la matrice stochastique S est normalisée à l'unité. L'élément de la matrice stochastique S_{ij} est la probabilité de transition du nœud j au nœud i. Un internaute aléatoire, condamné à errer éternellement de nœud en nœud sur le réseau, a une probabilité S_{ij} de sauter au nœud i s'il est initialement localisé sur le nœud j. Si le nombre de sorties du nœud j est de k_j^{out}, alors le surfeur aléatoire peut sauter du nœud j à n'importe quel nœud adjacent avec la même probabilité $S_{ij} = 1/k_j^{\text{out}}$. Si le noeud j est un noeud ballant, c'est-à-dire qu'il n'y a pas de sortie du noeud j, alors on permet au surfeur aléatoire de sauter, avec une probabilité de $1/N$, à n'importe quel noeud du réseau. Supposons que le surfeur aléatoire commence son voyage à partir du nœud j_0. Le vecteur $\mathbf{p}^{(1)} = S\mathbf{p}^{(0)}$, avec $p_i^{(0)} = \delta_{ij_0}$, donne la distribution de probabilité décrivant la localisation du surfeur aléatoire après un saut, c'est-à-dire une itération du processus stochastique.[b] La $i$ième composante du vecteur $\mathbf{p}^{(n)} = S^n\mathbf{p}^{(0)}$, c'est-à-dire $p_i^{(n)}$, est la probabilité que le surfeur aléatoire aboutisse au noeud i après n itérations.[c] Le vecteur $\mathbf{p}^{(\infty)}$ donne la distribution de probabilité après un parcours infini du

a. En mathématiques, un réseau est appelé un graphe. Un nœud d'un réseau est un sommet d'un graphe, et un lien entre deux nœuds d'un réseau est une arête entre deux sommets d'un graphe. Il s'agit simplement d'un changement de nom.

b. Le symbole de Kronecker δ_{ij} est égal à 1 si $i = j$ et est égal à 0 sinon.

c. Puisque les composantes vectorielles de $\mathbf{p}^{(n)}$ sont des probabilités, nous avons $\|\mathbf{p}^{(n)}\|_1 = \sum_{i=1}^{N} p_i^{(n)} = 1$.

surfeur aléatoire. Cette distribution de probabilité n'est pas unique car elle dépend de la distribution de probabilité initiale $\mathbf{p}^{(0)}$, c'est-à-dire de l'endroit où le surfeur aléatoire commence son voyage. Afin d'obtenir une solution unique de la marche aléatoire sur le réseau, on définit l'opérateur de Perron-Frobenius G dont les éléments sont

$$G_{ij} = \alpha S_{ij} + (1-\alpha)\, v_i. \tag{5}$$

Ici, le paramètre $\alpha \in [0.5, 1[$ est le facteur d'amortissement et $\mathbf{v}$ est un vecteur préférentiel. L'équation (5) nous indique qu'à chaque itération, le surfeur aléatoire suit, avec la probabilité α, la structure du réseau, ou saute à n'importe quel nœud, avec la probabilité $(1-\alpha)$, selon le vecteur préférentiel $\mathbf{v}$. Dans l'article authentique [1], le facteur d'amortissement est pris égal à $\alpha = 0.85$ et les composantes du vecteur préférentiel sont $v_i = 1/N$, $\forall i$. L'opérateur de Perron-Frobenius G est appelé la matrice de Google [3]. Ce facteur d'amortissement permet à l'internaute aléatoire de ne jamais être bloqué dans un sous-réseau et, par conséquent, il n'existe qu'une seule solution $\mathbf{P}$ pour la distribution de probabilité du voyage infini, $G^{\infty}\mathbf{p}^{(0)} = \mathbf{P}$ pour toute distribution de probabilité initiale $\mathbf{p}^{(0)}$. Le vecteur $\mathbf{P}$ donne la distribution de probabilité en régime permanent du processus stochastique codé par la matrice stochastique G puisque

$$G\mathbf{P} = \mathbf{P}. \tag{6}$$

La $i$ième composante du vecteur PageRank $\mathbf{P}$ est proportionnelle au nombre de fois où le surfeur aléatoire atteint le noeud i. Il est possible de définir l'indice PageRank K. On attribue le rang $K = 1$ au noeud i_1 tel que

$$P_{i_1} = \max_{i \in [1,\ldots,N]} \{P_i\},$$

le rang $K = 2$ au noeud i_2 tel que

$$P_{i_2} = \max_{i \in [1,\ldots,N]-\{i_1\}} \{P_i\},$$

le rang K au noeud i_K tel que

$$P_{i_K} = \max_{i \in [1,\ldots,N]-\{i_1,\ldots,i_K\}} \{P_i\},$$

et le rang $K = N$ au noeud i_N tel que

$$P_{i_N} = \min_{i \in [1,\ldots,N]} \{P_i\}.$$

Les nœuds du réseau sont ensuite classés par ordre décroissant de leurs valeurs des probabilités PageRank. Le nœud auquel est attribué $K = 1$ ($K = N$) est le nœud le plus (moins) important ou central du réseau selon la mesure de centralité PageRank.

Un algorithme possible permettant de calculer le vecteur $\mathbf{P}$ est celui par itération de puissance dont le pseudo-code est présenté ci-après 2.1. Cet algorithme se base sur le résultat précédemment énoncé : "il n'existe qu'une seule solution $\mathbf{P}$ pour la distribution de probabilité du voyage infini, $G^{\infty}\mathbf{p}^{(0)} = \mathbf{P}$ pour toute distribution de probabilité initiale $\mathbf{p}^{(0)}$. "

Algorithm 1 Méthode d'itération de puissance

Data : $\mathrm{P}_{\mathrm{init}}$, ε, G^*
Result : CheiRank vector $\mathbf{P} = G^{\infty}\mathbf{p}^{(0)}$

```
while G*P − P ≥ ε do
    P = G*P
end
```

Cet algorithme se traduit de la manière suivante : tant que le produit $\mathrm{G^*P}$ est plus grand que P à ε près (ε étant un nombre réel aussi petit que l'on souhaite), on continue d'itérer le produit matriciel.

Calcul du PageRank : diagonalisation directe

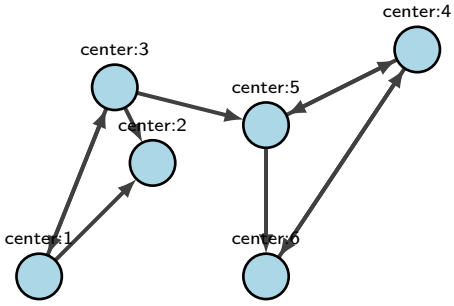

FIGURE 1 – Network 1.

Pour le réseau affiché ci-dessus,

- classez les noeuds selon la centralité PageRank en résolvant le problème aux valeurs propres (6). On prendra la valeur originale $\alpha = 0.85$ et on partira du noeud 3,
- que se passe-t-il pour les probabilités de PageRank et le classement PageRank lorsque α varie dans l'intervalle $[0.5, 1[$,
- comparez la centralité PageRank avec la centralité de degré.,
- réalisez des figures pertinentes mettant en évidence vos résultats.

2.2 Le vecteur CheiRank

Un autre vecteur est intéressant à prendre en compte, c'est le vecteur CheiRank $\mathbf{P}^*$ construit de la même manière que le vecteur PageRank $\mathbf{P}$ mais en considérant la matrice G^* construite à partir de A^t. A contrario du vecteur PageRank qui nous informe sur la probabilité d'atteindre tel ou tel noeud au bout d'une infinité de déplacements, le vecteur CheiRank nous informe du niveau de **communication** d'un noeud. La valeur la plus élevée (faible) du CheiRank caractérise donc le noeud le plus (moins) communiquant.

Toujours en considérant le réseau précédent :

- classez les noeuds selon la centralité CheiRank, du plus communiquant au moins communiquant, en résolvant le problème d'itération de puissance (2.1), considérant la matrice G^*.
- commentez ce classement.
- le classement évolue-t-il en fonction du vecteur d'arrivée $\mathbf{P}^*$?
- tracez, à chaque itération, un histogramme du vecteur CheiRank en fonction des noeuds (en executant votre code, on doit donc voir une petite animation de l'évolution de la valeur du CheiRank en fonction de l'itération).

---

3 PageRank : Pokémon, liez les tous !

Le système de combat dans le jeu video Pokémon peut être vu comme un réseau dirigé. Dans cet univers, les pokémons sont des animaux guerriers classables en types bien définis. Ces types sont généralement liés aux élements tels que la terre, l'eau le feu, l'air.. Un pokémon peut avoir des attaques de types différents. Si une attaque de type t_1 touche un pokémon de type t_2 , cette attaque peut être boostée voir totalement annulée en fonction des types mis en jeu.

Voici un tableau récapitulatif des règles concernant les faiblesses et les forces pour les différents types.

On y voit alors un réseau dirigé dont les noeuds sont les types. Un lien $\mathrm{t}_1 \to \mathrm{t}_2$ signifie qu'une attaque de type t_1 est lancée sur un pokémon de type t_2 . Le poids de ce lien définit le bonus ou le malus que l'attaque subira.

FIGURE 2 – Résumé de l'impact des attaques en fonction de leur type. 0 signifie que l'attaque n'a aucun effet, 1/2 que le type ciblé présente une résistance moyenne, 2 que l'attaque est super efficace contre le type ciblé et les cases blanches représentent une résistance normale de 1.

En considérant le réseau des types de pokémons,

- que représente la matrice d'adjacence A dans un réseau ?
- même question pour la matrice stochastique S.
- Ecrivez un script python exécutable et soigneusement commenté, prenant en argument les fichiers 'nodes.txt' et 'links.dat' afin de déterminer le PageRank et le CheiRank du réseau associé aux types depokémons.
- synthétisez vos résultats dans deux fichiers texte générés par votre script (un pour le PageRank et l'autre pour le CheiRank) ayant la forme suivante :
  - Première ligne : ε utilisé et nombre de noeuds
  - Deuxième ligne : ligne vide
  - Puis un tableau à cinq colonnes (avec séparateur) : Rang – ID – Type – PageRank (itération de puissance) – PageRank (diagonalisation)
- Pour chaque classement, donnez une signification du premier et du dernier noeud (question ouverte).
- (Bonus) A l'aide du module *networkx* de python, tracez le réseau correspondant au problème en tenant compte des poids de chacun des noeuds (chaque lien entre noeuds peut être d'une couleur différente en fonction de la valeur de ce lien).

Références

[1] S. Brin and L. Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. In: Seventh International World-Wide Web Conference (WWW 1998), April 14-18, 1998, Brisbane, Australia

[2] A.A. Markov, Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga, Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete, 2-ya seriya, 15, 135 (1906) (Extension of the limit theorems of probability theory to a sum of variables connected in a chain)

[3] A.N. Langville and C.D. Meyer, *Google's PageRank and Beyond : The Science of Search Engine Rankings*, Princeton University Press (2006)