FACULTÉ DES SCIENCES ET TECHNOLOGIES
Nancy

M1-Algèbre linèaire appliquèe

Google PageRank

December, 2023

Lidia Nievas
Malang Badiane

UNIVERSITÉ
DE LORRAINE

Projet d'algebre lineaire

MALANG BADIANE   LIDIA NIEVAS DUENAS

December 26, 2023

1 Introduction

Le moteur de recherche Google a été mis au point à la fin des années 90 par S.Brin et L.Page. À l'issue d'une requête, il affiche les réponses dans un ordre reposant sur un indice de popularité des pages web : le PageRank. Ce procédé a permis a Google de devance la concurrence.Le but du projet sera d'implémenter le calcul de l'indice :il s'agit d'un calcul de vecteur propre par la méthode de puissance itérée On supposera que le web est constitué des quatres pages données ci-dessous, les caractères en gras indiquent les hyperliens :

1.1 P1 : Matrice des hyperliens

La matrice des hyperliens code les liens entre les pages du web. la **matrice de Google** est un employé parle célèbre moteur de recherche repose sur cette matrice.

1.2 P2 : Matrice stochastique

Une matrice stochastique est une matrice carrée dont les coefficients sont positifs et dont la somme des coefficients vaut 1 sur chaque ligne. La **matrice de Google** est un exemple célèbre de matrice stochastique, dont le **vecteur propre** de la valeur propre dominante fournit le PageRank.

1.3 P3 : Matrice de Google

La matrice de Google est une **matrice stochastique** déduite de la matrice des hyperliens de l'ensemble des pages Web. Le vecteur PageRank de classement des pages par popularité est un **vecteur propre** associé à la plus grande valeur propre de cette matrice.

1.4 P4 : Vecteur propre

Si **A** est un matrice rélle n × n , on dit que le vecteur $v \in R^n$ est un vecteur propre de **A** si $v \neq 0$ et s'il existe $\lambda \in C$ tel que $\mathbf{A}v = \lambda v$.
Question1
Sur cet exemple, donnons les matrices d'hyperlien H, stochastique S, puis celle de G en prenant $\alpha = 0 : 85$, et le vecteur v uniforme :

$$v = \frac{1e}{n}$$

On définie La matrice d'hyperlien H par : $H_{i,j} = \begin{cases} \frac{1}{p_{i,j}} & \text{si} \mathrm{P}_i \text{ possede un lien qui point vers la page } P_j \\ 0 & sinon \end{cases}$

Ainsi on aura dans notre cas :

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

On déduit la matrice stochastique S :

S = H + a$\frac{1}{n} \cdot e^T$

$$S = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix}$$

On définie la matrice de google par :
$G = \alpha S + (1 - \alpha)\, ev^T$
Ce qui nous donne pour $\alpha = 0 : 85$ et $v = \frac{1}{4}e$

$$G = \begin{pmatrix} 0.0375 & 0.0375 & 0.0375 & 0.0375 \\ 0.0375 & 0.0375 & 0.4625 & 0.4625 \\ 0.0375 & 0.4625 & 0.0375 & 0.4625 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix}$$

Code sur Matlab:

untitled | TP_alg.m

```
%EXercice 1
disp('Exercice 1')
n=4;
    %Matrice H
H=[0,0,1,0;0,0,1/2,1/2;0,1/2,0,1/2;0,0,0,0];
disp('H=')
disp(H)
    %Matrice S
S=[0,0,1,0;0,0,1/2,1/2;0,1/2,0,1/2;1/4,1/4,1/4,1/4];
disp('S=')
disp(S)
    %Matrice G
alpha=0.85;
e=[1;1;1;1];
v=(1/n)*e;
G=alpha*S+(1-alpha)*e*v';
disp('G=')
disp(G)
```

Sortie d'écran:

```
Exercice 1
H=
         0         0    1.0000         0
         0         0    0.5000    0.5000
         0    0.5000         0    0.5000
         0         0         0         0

S=
         0         0    1.0000         0
         0         0    0.5000    0.5000
         0    0.5000         0    0.5000
    0.2500    0.2500    0.2500    0.2500

G=
    0.0375    0.0375    0.8875    0.0375
    0.0375    0.0375    0.4625    0.4625
    0.0375    0.4625    0.0375    0.4625
    0.2500    0.2500    0.2500    0.2500
fx
```

2 L'algorithme de la puissance itérée :

2.1 Question 2

- Le pseudo-code :
Entrés : matrice stochastique $A < 0$ de taille n $\times$ n, vecteur ligne stochastique $x_0 \geq 0$ d'initialisation, tolérence $\epsilon > 0$ le résidu, borne $nmax$ sur le nombre d'itération.
Poser $z = x_0 A$
Calculer le résidu : $eta = ||z - x||_1$
mettre à jour :$x = z, k = 1$
Tant Que: $> \epsilon$ et $k < nmax$
Poser $z = xA$,
cacluler le 'residu' : $= ||z - x||_1$
Mettre à jour :$x = z, k = k + 1$
Fin TantQue
Sorties : vecteur ligne stochastique $x > 0$ tel que $xA = x$, et nombre k d'itèrations.

Le code sur Matlab :

```
%Fonction exercice 2
function[x,k]=puissanceiteree(A,x0,epsilon,nmax)
    %Entrée: matrice stochatisque A
            %vecteur ligne stochastique x0 d'initialisation
            %entier epsilon, la tolérance
            %entier nmax, nombre max d'itérations
    %Sortie: le PageRank pi
            %entier k, le nombre d'itérations
    z=x0*A;
    eta=norm(z-x0,1);
    x=z;
    k=1;
    while (eta>epsilon && k<nmax)
        z=x*A;
        eta=norm(z-x,1);
        x=z;
        k=k+1;
    end
end
```

2.2 Question 3

On calcule le pagerank des pages :

```
%Exercice 3
disp('Exercice 3')
    %Définition des variables
pi0=[0.25,0.25,0.25,0.25];
epsilon=0.01;
nmax=100;
    %Calcul du PageRank avec puissanceiteree
[pi,k]=puissanceiteree(G,pi0,epsilon,nmax);
disp('pi=')
disp(pi)
disp('k=')
disp(k)
```

On obtient le résultat suivant :

```
Exercice 3
pi=
    0.1104    0.2413    0.3054    0.3428

k=
     6
```

On constate que la méthode converge en six itérations et le vecteur du pagerank est : $\pi = (0.1104, 0.2413, 0.3054, 0.3428)$
On constate que la page P4 est la plus populaire suivie de P3, P2 et P1

2.3 Question 4

On teste maintenant differents vecteurs de personnalisation :

```
%Exercice 4
disp('Exercice 4')
    %Définition des variables
alpha=0.85;
e=[1;1;1;1];
v1=[0.1,0.4,0.1,0.4];
v2=[0.02,0.48,0.02,0.48];
G1=alpha*S+(1-alpha)*e*v1;
G2=alpha*S+(1-alpha)*e*v2;
pi0=[0.25,0.25,0.25,0.25];
epsilon=0.01;
nmax=100;
    %Calcul du PageRank avec puissanceiteree
[pi1,k1]=puissanceiteree(G1,pi0,epsilon,nmax);
disp('pi1=')
disp(pi1)
disp('k1=')
disp(k1)
[pi2,k2]=puissanceiteree(G2,pi0,epsilon,nmax);
disp('pi2=')
disp(pi2)
disp('k2=')
disp(k2)
```

Sortie d'ècran:

```
Exercice 4
pi1=
    0.0932    0.2586    0.2808    0.3674

k1=
     6

pi2=
    0.0839    0.2678    0.2677    0.3806

k2=
     6
```

On constate que l'indice de populatité de la page 4 est le plus grand dans les trois cas. Donc la page la plus populaire est la page 4. le vecteur de personnalisation v est choisi pour gonfler le l'indice de popularité des pages. Lorsqu'on gonfle une composante du vecteur v on augmente son indice de popularité :par exemple en modifiant v4 au lieu de 0.4 on met 0.48 ,l'indice de popularité de la page 4 est passée de 0.3674 à 0.3806.

3 Le PageRank

3.1 Question 5

On donne le code du pageRank :

```
%Exercice 5
disp('Exercice 5')
    %Définition des variables
v1=[0.1;0.4;0.1;0.4];
v2=[0.02;0.48;0.02;0.48];
    %Calcul du PageRank avec PageRank
[pi1,k1]=PageRank(H,v1,alpha,pi0,epsilon,nmax);
disp('pi1=')
disp(pi1)
disp('k1=')
disp(k1)
[pi2,k2]=PageRank(H,v2,alpha,pi0,epsilon,nmax);
disp('pi2=')
disp(pi2)
disp('k2=')
disp(k2)
```

```
%Fonction exercice 5
function[pi,k]=PageRank(H,v,alpha,pi0,epsilon,nmax)
    %Entrée: matrice des hyperliens H
            %vecteur stochastique de personnalisation v
            %entier alpha, le parametre
            %vecteur ligne stochastique pi0 d'initialisation
            %entier epsilon, la tolérance
            %entier nmax, nombre max d'itérations
    %Sortie: le PageRank pi
            %entier k, le nombre d'itérations
    n=size(v,1);
    e=ones(1,n)';
    a=zeros(1,n)';
    for i=1:n
        if(H(i,:)==zeros(1,n))
            a(i)=1;
        end
    end
    z=alpha*(pi0*sparse(H)+(pi0*a)*e'/n)+(1-alpha)*v';
    eta=norm(z-pi0,1);
    pi=z;
    k=1;
    while (eta>epsilon && k<nmax)
        z=alpha*(pi*sparse(H)+(pi*a)*e'/n)+(1-alpha)*v';
        eta=norm(z-pi,1);
        pi=z;
        k=k+1;
    end
end
```

Sortie d'ècren:

```
Exercice 5
pi1=
    0.0932    0.2586    0.2808    0.3674

k1=
     6

pi2=
    0.0839    0.2678    0.2677    0.3806

k2=
     6
```

3.2 Question 6

La fonction genererH :

```
%Exercice 6
disp('Exercice 6')
[H]=genererH(5,3);
disp('H=')
disp(full(H))
```

```
%Fonction exercice 6
function[H]=genererH(n,m)
     %Entrée: entier n, nombre des pages
            %entier m, nombre des hyperliens
    %Sortie: matrice des hyperliens H
    if n<=m
        disp("Erreur")
    end
    H=sparse(n,n);
    for i=1:n
        nbre_hyperliens=(m+1)*rand();
        nombre_hyperliens=floor(nbre_hyperliens);
        if nombre_hyperliens~=0
            count=1;
            while count<=nombre_hyperliens
                index=1+n*rand();
                indice=floor(index);
                if (H(i,indice)==sparse(1,1))
                    H(i,indice)=1/nombre_hyperliens;
                    count=count+1;
                end
            end
        end
    end
end
```

Sortie d'èren:

```
Exercice 6
H=
         0         0    0.5000         0    0.5000
    0.3333    0.3333         0    0.3333         0
    1.0000         0         0         0         0
         0    1.0000         0         0         0
         0         0         0         0         0
```

3.3 Question 7

Comparaison du temps des calculs du pageRank et la puissance iterée:
Pour cette question nous executons les deux fonctions(pageRank,puissanceiterre) et à chaque fois nous conservons leurs temps d'execution des deux fonctions.

```
untitled   TP_alg.m
%Exercice 7
disp('Exercice 7')
n=10000;
[H]=genererH(n,50);

tic
    %Définition des variables pour écrire G
a=zeros(1,n)';
%Remplissez le vecteur 'a' avec des 1 là où la matrice 'H' a une ligne pleine de 0
    for i=1:n
        if(H(i,:)==zeros(1,n))
           a(i)=1;
        end
    end
nmax=100;
alpha=0;5;
pi0=ones(1,n)/n;
S=H+(a/n)*ones(1,n);
G=alpha*S+(1-alpha)*ones(1,n)*pi0';
    %Calcul du PageRank avec puissanceiteree
[pi,k]=puissanceiteree(G,pi0,0.001,nmax);
toc

tic
    %Définition des variables
nmax=100;
alpha=0.5;
pi0=ones(1,n)/n;
w=ones(1,n)/n;
v=w';
    %Calcul du PageRank avec PageRank
[pi,k]=PageRank(H,v,alpha,pi0,0.001,nmax);
toc
```

Sortie d'èren:

```
Exercice 7
Elapsed time is 12.182010 seconds.
Elapsed time is 6.966459 seconds.
```

Pour une matrice de taille 1000×1000, On constate que la méthode du PageRank est moins couteuse au niveau de temps par rapport à la méthode de la puissance iterée.

3.4 Question 8

Nos ordinateurs n'arrivent pas executer les codes, par soucis d'insufisance de mémoire pour stocké les variables de type 100000×100000.
Mais le code est:

```
%Exercice 8
disp('Exercice 8')
n=100000;
[H]=genererH(n,50);

tic
    %Définition des variables pour calculer G
a=zeros(1,n)';
%Remplissez le vecteur 'a' avec des 1 là où la matrice 'H' a une ligne pleine de 0
    for i=1:n
        if(H(i,:)==zeros(1,n))
            a(i)=1;
        end
    end
nmax=100;
a=sparse(a);
alpha=0;5;
pi0=ones(1,n)/n;
pi0=sparse(pi0);
S=H+(a/n)*ones(1,n);
S=sparse(S);
G=alpha*S+(1-alpha)*ones(1,n)*pi0';
G=sparse(G);
    %Calcul du PageRank avec puissanceiteree
[pi,k]=puissanceiteree(G,pi0,0.001,nmax);
toc

tic
    %Définition des variables
nmax=100;
alpha=0.5;
pi0=ones(1,n)/n;
pi0=sparse(pi0);
w=ones(1,n)/n;
v=w';
v=sparse(v);
    %Calcul du PageRank avec PageRank
[pi,k]=PageRank(H,v,alpha,pi0,0.001,nmax);
toc
```

3.5 Quesion 9

Honorer le contrat : Pour honorer le contrat, il suffira de manipuler le vecteur v de personnalisation qui permettra d'élever le rang des pages concernèes dans le vecteur colonne des indices de popularité. Pour le marché qui nous est proposé, on va augmenter l'indice de popularité des 5 pages P6, . . ., P10. Pour ce faire, on va gonfler les composante du vecteur v entre la 6eme et la 10eme et diminuer les autres, tout en veillant à ce que v soit toujours unitaire.

```
%Exercice 9
disp('Exercice 9')
    %Définition des variables
n=100000;
[H]=genererH(n,50);
nmax=100;
alpha=0.5;
pi0=ones(1,n)/n;
w=ones(1,n)/n;
v=w';
    %Calcul du PageRank
[pi,k]=PageRank(H,v,alpha,pi0,0.001,nmax);
disp(pi(1,6:10))
    %Calcul du PageRank avec les pages 6 à 10 avec plus de poids
v0=per_v(n)';
[pi,k]=PageRank(H,v0,alpha,pi0,0.001,nmax);
disp(pi(1,6:10))

%Fonction pour implementer les poids sur les pages 6 à 10
function[v0]=per_v(n)
    %Entrée: un entier n, la taille du matrice
    %Sortie: un vecteur v0
    v0=(1/n)*ones(1,n);
    for i=6:10
        v0(i)=1.5*(1/n);
    end
    for i=1:5
        v0(i)=(1-sum(v0(6:10)))/(n-5);
    end
    for i=11:n
        v0(i)=(1-sum(v0(6:10)))/(n-5);
    end
end
```

Appliqué à une matrice d'une taille 10000x10000, l'ordinateur n'a pas assez de mémoire pour exécuter le programme, mais un exemple de ce qui se passerait avec une matrice plus petite (de taille 15x15 dans ce cas) est le suivant :

```
Exercice 9
PageRank normal
    0.0676    0.0892    0.0770    0.0512    0.0579

PageRank avec les pages 6 à 10 avec plus de poids
    0.0811    0.1065    0.0902    0.0660    0.0741
```

4 Conclusion

Les moteurs de recherche sont d'énormes facteurs de pouvoir sur le Web, guidant les gens vers l'information et prestations de service.
Google est le moteur de recherche le plus performant de ces dernières années, ses résultats de recherche très complets et précis. Quand Google n'était qu'un projet de cherche précoce à Stanford, plusieurs articles ont été écrits décrivant les algorithmes sousjacents.
L'algorithme dominant a été appelé PageRank et est toujours la clé pour fournir classements précis pour les résultats de recherche. Une fonctionnalité primordiale des moteurs de recherche de pages web consiste en un tri des résultats associés à une requête par ordre d'importance ou de pertinence.
Nous présentons un modèle permettant de définir une quantification de cette notion (Pagerank) a priori floue et des éléments de formalisation pour la résolution numérique du problème. On commence par une première approche naturelle non satisfaisante dans certains cas. Un raffinement de l'algorithme est donc introduit pour améliorer les résultats.