R5.A.12/R5.B.10 Modélisations mathématiques

Séquence 2 : Valeurs propres

Thibault Godin, Lucie Naert

IUT de Vannes

15 septembre 2024

Avancement

- ~~Semaine 1 : Mariages stables avec Gale-Shapley~~
- ~~Semaine 2 : Mariages stables équitables avec Selkow~~
- ~~Semaine 3 : Flots et Affectations avec Edmonds-Karp~~
- **Semaine 4 : Initiations aux valeurs propres**
- **Semaine 5 : Classement des pages Web avec PageRank**
- Semaine 6 : Clustering spectral et découpes de vaches
- Semaine 7 : Compression et débruitage d'images avec SVD
- Semaine 8 : Évaluation avec sujet surprise

Plan

Matrices : quelques rappels

Une matrice réelle est un tableau dont les éléments réels sont repérés par un indice ligne suivi d'un indice colonne :

$$\begin{pmatrix} a_{11} & \dots & a_{1p} \\ a_{21} & \dots & a_{2p} \\ \vdots & \vdots & \vdots \\ a_{n1} & \dots & a_{np} \end{pmatrix}$$

L'ensemble des matrices à n lignes et p colonnes (à coefficients réels) est noté $\mathcal{M}_{n,p}(\mathbb{R})$.

Matrices carrés

Une matrice est carré si elle possède autant de lignes que de colonnes :

$$\begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & \vdots & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$$

L'ensemble des matrices carrées d'ordre n lignes (à coefficients réels) est noté $\mathcal{M}_n(\mathbb{R})$.

Dans ce cours, nous nous intéresserons exclusivement aux matrices carrées.

Multiplication d'une matrice par un vecteur

A vous de jouer :

1. Calculez le produit $M\mathbf{v}$ avec $M = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{pmatrix}$ et $\mathbf{v} = \begin{pmatrix} 3 \\ 2 \\ 5 \end{pmatrix}$
2. Que remarquez-vous ?

Plan

Valeurs et vecteurs propres : définition

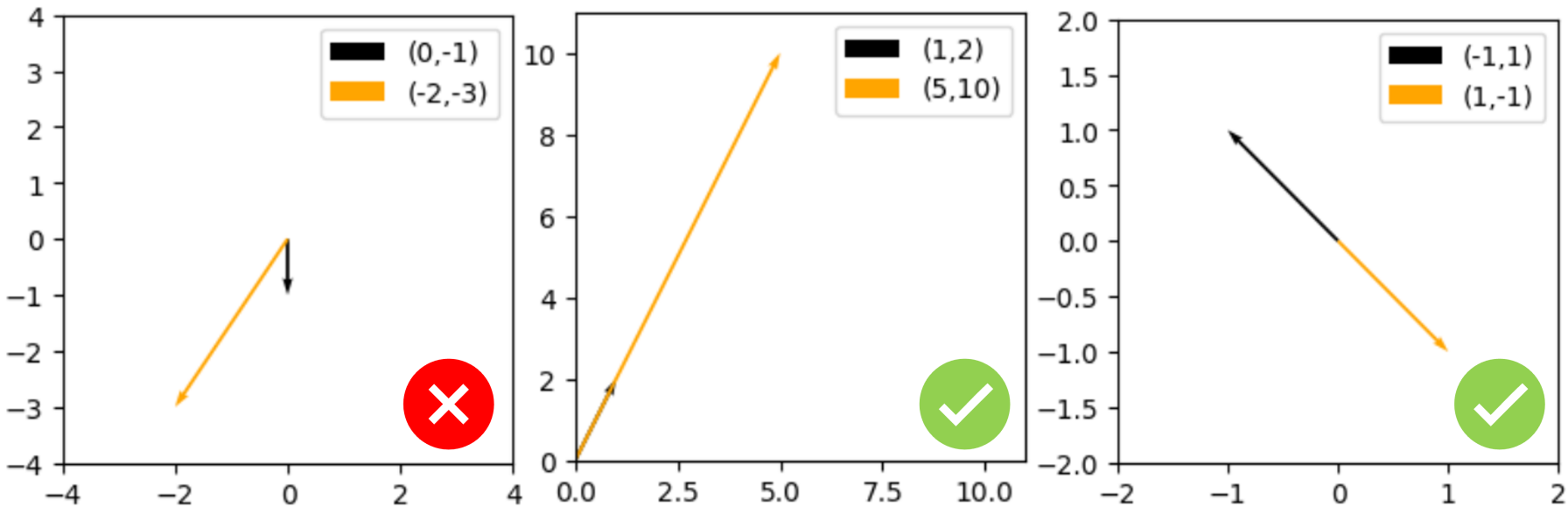
Soient $M \in \mathcal{M}_n(\mathbb{R})$, $\mathbf{v} \in \mathbb{R}^n$ et $\lambda \in \mathbb{R}$ tels que $M\mathbf{v} = \lambda\mathbf{v}$ alors :

- $\mathbf{v}$ est un **vecteur propre** de M.
- λ est la **valeur propre** de M associée au vecteur propre $\mathbf{v}$.

Quelles est la valeur propre associées au vecteur propre du diapositive précédent ?

Interprétation graphique

Pour la matrice $M \in \mathcal{M}_2(\mathbb{R})$ avec $M = \begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}$



Applications

Soit $A \in \mathcal{M}_3(\mathbb{R})$, la matrice $A = \begin{pmatrix} 1 & 3 & 3 \\ -2 & 11 & -2 \\ 8 & -7 & 6 \end{pmatrix}$

1. Vérifiez que $X_1 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$ est vecteur propre de A et donnez sa valeur propre associée.
2. Vérifiez que $X_2 = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$ est vecteur propre de A et donnez sa valeur propre associée.

Applications (suite)

Soit $A \in \mathcal{M}_3(\mathbb{R})$, la matrice $A = \begin{pmatrix} 1 & 3 & 3 \\ -2 & 11 & -2 \\ 8 & -7 & 6 \end{pmatrix}$

3. Vérifiez que $\lambda_3 = 7$ est valeur propre de A et donnez son vecteur propre associé.

Source : Exemple tiré de exo7

Propriétés

- Soit $A \in \mathcal{M}_n(\mathbb{R})$ alors A admet au plus n valeurs propres.
- Soit $A \in \mathcal{M}_n(\mathbb{R})$ et λ une de ses valeurs propres. On appelle multiplicité de λ le nombre de vecteurs deux à deux linéairement indépendants qui ont λ pour valeur propre associée.
- Les valeurs propres d'une matrice diagonale sont ses coefficients diagonaux.

Normalisation

En pratique, on fixe les vecteurs propres en les **normalisant** car il y a sinon une infinité de vecteur propres par valeur propre (vecteurs colinéaires).

Normaliser un vecteur u revient à diviser chacune de ses composantes par la norme $||u||$ du vecteur de façon à ce que le vecteur normalisé résultant ait une norme de 1.

Pour rappel : Soit $u = (x_1, x_2, ..., x_n)$, $||u|| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$

Ou, en désignant le produit scalaire par un point . : $||u|| = \sqrt{u.u}$

Plan

Calcul algébrique

Soit $M \in \mathcal{M}_n(\mathbb{R})$

1. Pour calculer les valeurs propres λ_i de M. On commence par calculer le polynome caractéristique de M : $P_M(X) = Det(M - XI_n)$.

   $\rightarrow$ *Algorithme exact de calcul de déterminant en* $\mathcal{O}(n^3)$.

2. Les valeurs propres sont les racines de $P_M(X)$, c'est-à-dire les X tels que $P_M(X) = 0$.

   $\rightarrow$ *Algorithme approximé de calcul des racines en* $\mathcal{O}(n^2)$.

3. A partir du calcul des valeurs propres λ_i, il est possible de trouver les vecteurs propres v_i associés en résolvant le système $Mv_i = \lambda_i v_i$.

   $\rightarrow$ *Algorithme exact (pivot de Gauss) en* $\mathcal{O}(n^3)$.

Méthode itérative

Algorithme des puissances itérées :

Cet algorithme permet de calculer efficacement **la plus grande valeur propre d'une matrice**.

Entrées :

- Matrice M carrée de taille n dont on cherche les valeurs propres
- nombre d'itération maximum et/ou tolérance t (pour l'arrêt de l'algorithme)

Sortie : une approximation de la plus grande valeur propre de M

Déroulé de l'algorithme

1. Initialiser un vecteur v_0 de taille n avec des valeurs aléatoires.

   Normaliser v_0.

2. Initialiser un compteur d'itération à 0.

3. Calculer $v_1 = Mv_0$ et normaliser v_1

4. Tant que $||v_k - v_{k+1}|| > t$ et/ou que le nombre d'itérations max n'est pas atteint :

   a. $v_{k+1} = Mv_k$ et normaliser v_{k+1}

   b. Incrémenter le nombre d'itérations

5. Retourner la valeur propre correspondant au dernier vecteur v_k

L'algorithme des puissances itérées donne la plus grande valeur propre d'une matrice. Pour le calculs des autres valeurs propres, il est possible d'adapter cet algorithme (puissances inverse / déflation) ou d'utiliser des algorithmes plus puissants (méthode QR, cf. TP).

Plan

En Python (1/3)

```
#imports necessaires :

import numpy as np

from numpy import linalg as la


# Exemple de matrice

A = np.array([[1,2],[4,3]])


# Exemple de vecteur

x=np.array([1,2])


# Produit matriciel

print(A@x) #[ 5 10]
```

En Python (2/3)

En anglais, "valeur propre" se dit *eigenvalue* et "vecteur propre" *eigenvector*. Python reprend cette dénomination. Ainsi, en Python, la fonction pour obtenir les valeurs et vecteurs propres d'une matrice est la fonction `numpy.linalg.eig()`

```
# Calcul des valeurs et vecteurs propres
eigenvalues_A, eigenvectors_A = la.eig(A)


#Affichage
for i in range(len(eigenvalues_A)) :
        print("valeur propre n°",i,"=", eigenvalues_A[i])
        print("valeur propre correspondant : ", eigenvectors_A[:,i])
```

En Python (3/3)

Résultat :

```
valeur propre n° 0 = -1.0
valeur propre correspondant :  [-0.9486833   0.31622777]
valeur propre n° 1 = 4.0
valeur propre correspondant :  [-0.89442719 -0.4472136 ]
```

Plan

Plan

PageRank

PageRank est un algorithme qui permet de classer des pages web par importance relative.

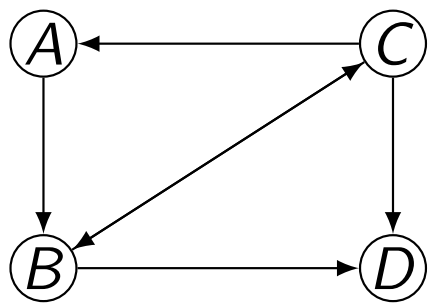
Pour mesurer cette importance, PageRank utilise les liens entre les pages et se basent sur les hypothèses suivantes :

- Si le site 1 renvoie vers le site 2, le site 1 considère le site 2 comme intéressant.
- Plus il y a de sites qui renvoient vers le site 2, plus celui-ci doit être important.
- Si un site important renvoie vers un site 2, le site 2 est sûrement important aussi
- Si le site 1 renvoie vers beaucoup d'autres sites, il faut répartir l'importance entre ces sites.

Graphes orientés et matrices d'adjacence

On peut représenter les liens entre des pages web sous forme d'un graphe orienté.

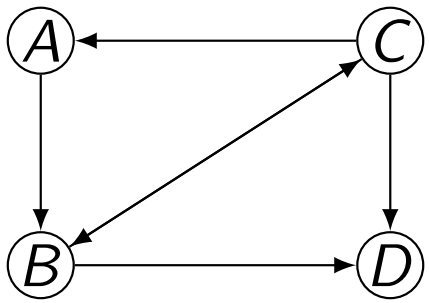
Par exemple : Soient 4 pages web A, B, C, D dont les liens sont représentés par le graphe orienté ci-dessous :



Lecture : Le site A a un lien vers le site B, B possède des liens vers C et D, etc.

Graphes orientés et matrices d'adjacence

Un tel graphe est représenté par la matrice d'adjacence suivante :



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

Plan

Algorithme de PageRank (version simplifiée)

Entrées :

- Graphe orienté G (représentant le réseau de sites web)
- Nombre d'itérations souhaitées *iter*

Sortie : Mesure d'importance pour chaque nœud du graphe

Déroulé de l'algorithme (version simplifiée)

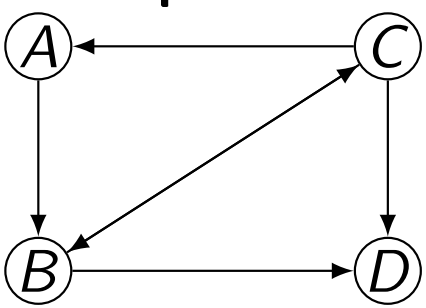
PageRank fonctionne sur le principe d'une marche aléatoire dans le graphe :

1. Nombre d'itération $i = 0$
2. Choix d'un nœud n_0 au hasard dans G
3. Tant que $i < iter$:
   - choix d'un nouveau nœud n_i parmi les voisins de n_{i-1} via l'un de ses liens sortants tiré uniformément.
   - on conserve le nœud visité dans une liste
   - $i = i + 1$
4. Calcul de la fréquence d'occupation de chaque nœud.

Fréquence d'occupation

On appelle **fréquence d'occupation** d'un nœud k, le nombre de fois où k a été visité divisé par le nombre d'itérations. Cette fréquence donne l'importance du nœud.

Exemple :



Sur 10 itérations, imaginons que la marche aléatoire donne : $[ABCBCABDDD]$

Les scores d'importance seront les suivants :

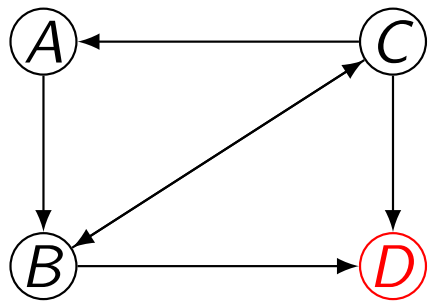
$f_A = \frac{2}{10} = 0.2,\ f_B = \frac{3}{10} = 0.3,$

$f_C = \frac{2}{10} = 0.2,\ f_D = \frac{3}{10} = 0.3$

Que remarquez-vous ?

Le "hic"

Problème : si un nœud ne contient pas de liens sortants, ou si un ensemble de nœuds forme une boucle sans liens sortants, le marcheur aléatoire se retrouve piégé indéfiniment et un site sans grande importance peut se retrouver avec un score énorme.



Solution

Pour éviter cela, la version native de PageRank modifie légèrement la marche aléatoire présentée plus haut afin qu'à chaque étape, il y ait une certaine probabilité que le marcheur saute vers un nœud aléatoire plutôt que de suivre un lien. On appelle cela une **téléportation**.

Cette probabilité est déterminée par un paramètre, α, qui est la probabilité de suivre un lien. Ainsi $1 - \alpha$ est la probabilité de faire un saut aléatoire.

Algorithme de PageRank (version avec téléportation)

Entrées :

- Graphe orienté G (représentant le réseau de sites web)
- Nombre d'itérations souhaitées *iter*
- **Probabilité α de ne pas se téléporter ($0 \leq \alpha \leq 1$)**

Sortie : Mesure d'importance pour chaque nœud du graphe

Déroulé de l'algorithme (version avec téléportation)

1. Nombre d'itération $i = 0$
2. Choix d'un nœud n_0 au hasard dans G
3. Tant que $i < iter$:
   - **Tir d'un nombre x au hasard entre 0 et 1**
     - si $x < \alpha$: choix d'un nouveau nœud n_i parmi les voisins de n_{i-1} via l'un de ses liens sortants tiré uniformément.
     - **sinon, téléportation : choix d'un nouveau nœud n_i parmi tous les noeuds du graphe**
   - on conserve le nœud visité dans une liste
   - $i = i + 1$
4. Calcul de la fréquence d'occupation de chaque nœud.

Lien avec les valeurs propres

A voir en TP !