TP A4 - Chaînes de Markov 2

Dans tout le TP, on importe les modules suivants :

```
1 import numpy as np
  import numpy.random as rd
  import numpy.linalg as al
  import matplotlib.pyplot as plt
```

1 Loi stationnaire d'une chaîne de Markov

Nous avons constaté sur l'exemple de Doudou le hamster que lorsque n devient grand, la loi de X_n ne dépend plus de l'état initial.
Plus précisément, (sous quelques hypothèses techniques dont nous reparlerons plus tard) si M est la matrice de transition de la chaîne de Markov, alors quel que soit le vecteur probabiliste μ_0 décrivant l'état initial, on a, pour n suffisamment grand,

$$\mu_n = \mu_0 M^n \simeq \mu.$$

où μ ne dépend pas de n.

Exercice 1 ⋆

1. Rappeler pourquoi 1 est valeur propre de M.
2. En considérant une limite coordonnée coordonnée, montrer que μ vérifie $\mu = \mu M$.
3. En déduire que ${}^tM{}^\mu = {}^t\mu$, puis que 1 est valeur propre de ${}^tM$ et qu'un vecteur propre associé est ${}^t\mu$.

En `Python`, la commande `al.eig(M)` permet d'obtenir les valeurs propres et des vecteurs propres associés d'une matrice M (représentée sous forme de tableau `numpy`).

Exercice 2 ⋆

1. Exécuter la commande `al.eig` sur la matrice de transition de Doudou. Commenter le résultat obtenu : que signifie chacun des termes obtenus ?
2. 1 est-il bien valeur propre ? Donner un vecteur propre associé. Retrouve-t-on l'état stationnaire observé jusqu'à présent ?

2 Une chaîne de Markov célèbre : le PageRank

En 1997, deux étudiants de Stanford, Larry Page et Sergueï Brin ont l'idée d'utiliser les chaînes de Markov pour créer un moteur de recherche. Le problème des moteurs de recherche de l'époque est le suivant : une fois qu'on a associé une liste de mots-clés à chaque page du Web, comment les « ordonner » par ordre d'importance ? Comment choisir, pour un mot clé donné (par exemple « matrice ») les pages les plus susceptibles d'intéresser l'internaute ?
À part demander à un humain de trier les pages par ordre d'intérêt (notion bien évidemment subjective), il semble difficile de proposer un classement pertinent de ces pages.
L'idée des fondateurs de Google est la suivante : numérotons l'ensemble des pages d'Internet de 1 à N (N est bien évidemment très très grand et difficile à estimer mais il est fini - pour information en 1997, on estimait à 10^6 le nombre de sites web et il est plus plus de 2×10^9 en 2025).
On considère qu'il existe un nombre $c \in]0,1[$ tel que, à chaque étape de sa navigation, un internaute :

- suit un des liens de la page sur laquelle il se trouve avec probabilité $1-c$. Si la page ne contient pas de lien, alors on repart, de manière équiprobable vers n'importe laquelle des N pages.

- ou bien, avec probabilité c, repart vers n'importe laquelle des N pages d'Internet, toutes les pages étant équiprobables.

Donc si X_n est la variable aléatoire égale au numéro de la page sur laquelle se trouve l'internaute à la $n^{\text{ème}}$ étape de sa navigation, on a :

$$P_{[X_i=i]}(X_{n+1}=j) = \begin{cases} \frac{1}{N} & \text{si la page } i \text{ ne contient aucun lien} \\ \frac{c}{N} & \text{si la page } i \text{ contient des liens mais aucun vers la page } j \\ \frac{c}{N} + \frac{1-c}{\ell_i} & \text{sinon où } \ell_i \text{ est le nombre de liens que contient la page } i \end{cases}$$

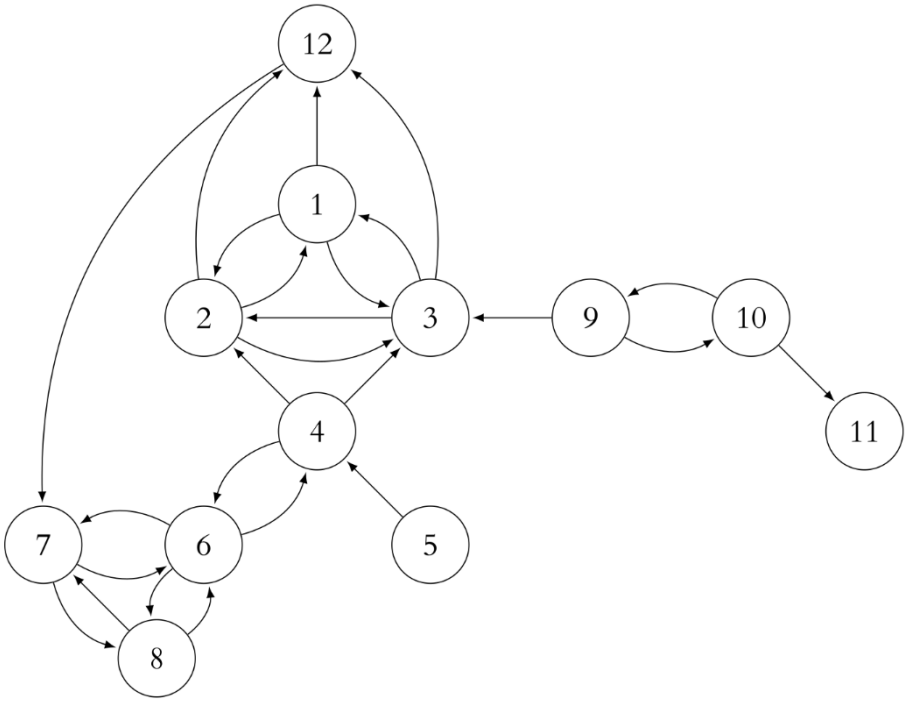
La suite (X_n) est alors une chaîne de Markov et on appelle PageRank (du nom de Larry Page) de la page i le nombre $\lim\limits_{n\to+\infty} P(X_n = i)$. Le PageRank d'une page est la probabilité, en surfant « au hasard » d'arriver sur cette page. En particulier, d'après de qu'on a dit précédemment, cela ne dépend pas de la page où commence la navigation.
On remarque notamment que plus une page possède de liens pointant vers elle, plus son PageRank est élevé, même si ce n'est pas le seul facteur à prendre en compte. Ainsi, une page avec un PageRank élevé est censée être plus pertinente étant donné qu'elle possède davantage de liens qui pointent vers elle.
Afin de calculer le PageRank de chacune des pages référencées, les serveurs de doivent alors déterminer la loi stationnaire de cette chaîne de Markov, c'est-à-dire calculer un vecteur propre de la matrice de transition (qui, rappelons-le, est une très grande matrice). Nous ne nous attarderons pas sur cet aspect, mais il existe des algorithmes permettant de calculer rapidement un vecteur propre d'une matrice très grande.

Exercice 3 ⋆

Afin d'augmenter le PageRank de mon site, je vais avoir besoin de davantage de liens pointant vers lui. Vaut-il mieux pour moi avoir un lien sur chacun des blogs de mes étudiants, ou un seul lien sur la page d'accueil du site du Monde ?

Exercice 4 ⋆⋆⋆

On considère un modèle simple de Web comportant 12 pages, le schéma ci-dessous représentant les liens entre les pages.



1. Essayer de déterminer quelles pages sont susceptibles d'avoir un PageRank plus élevé que les autres, ou au contraire bien plus faible.

2. On construit la matrice suivante :

```
B = np.array([
    [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
    [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
    [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
    [0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
])
```

Cette matrice contient un 1 à la $i^{\text{ème}}$ ligne et $j^{\text{ème}}$ colonne s'il y a un lien de la page i à la page j et 0 sinon.
En théorie des graphes, comment s'appelle une telle matrice ?
Copier ce code afin de pouvoir utiliser la matrice B par la suite.

3. Compléter le code suivant pour que, en prenant $0,15$ comme valeur de c, il crée, à partir de B, la matrice de transition C de la chaîne de Markov. Puis déterminer le PageRank, c'est-à-dire la loi stationnaire de la chaîne de Markov.

```
C = np.zeros((12,12))
c = 0.15
for i in range(12):
    l = np.sum(B[i,:]) # nombre de liens de la page i
    for j in range(12):
        if l == 0:
            C[i,j] = ...
        else:
            C[i,j] = ...

vals, vecs = al.eig(np.transpose(C))
# on suppose que la première valeur propre sera 1
PageRank = vecs[:,0]/np.sum(vecs[:,0])
print(PageRank)
```

Que constatez-vous ?

4. Modifier la matrice B pour supprimer le lien de la page 12 vers la page 7, et recalculer le PageRank. Même question, mais en ajoutant cette fois un lien de la page 12 vers la page 9.

Exercice 5 - La constante c ⋆

Une étude démontre qu'en pratique, un internaute relance sa navigation au hasard au bout de 6 pages en moyenne. Expliquer pourquoi il est alors pertinent de choisir $c = 0,15$.