TP3

CHAÎNES DE MARKOV.

Table des matières

Simuler une chaîne de Markov à l'aide de sa matrice de transition 2
Évolution d'une maladie. 2
Algorithme du Page Rank de Google. 4
Un exemple avec 3 sites 4
Le cas général 5

ECG 2 Maths appliquées, http://louismerlin.fr.

Simuler une chaîne de Markov à l'aide de sa matrice de transition

Regardons comment on peut simuler une trajectoire de chaîne de Markov avec sa matrice de transition. On suppose donc qu'on dispose de la matrice de transition

$$M = (m_{i,j}) = \left(\mathbb{P}_{(X_n=i)}(X_{n+1} = j)\right)_{i,j}.$$

On écrit alors un programme qui simule la trajectoire des n premiers pas de la chaîne (n est à choisir par l'utilisateur) avec la position initiale x_0 (à choisir aussi par l'utilisateur).

```
import numpy as np

def markov_avec_matrice(n,M,x0):
  X=np.zeros(n)
  X[0]=x0
  for k in range (n-1):
    X[k+1]=np.random.choice(a=list(range(len(M))), p= M[int(X[k
    ]),:])
  return X
```

L'utilisation et la justification de la bonne marche de ce programme demandent quelques explications. La fonction np.random.choice(a,p) simule une variable aléatoire avec support dans a et dont les probabilités sont contenues dans p. Les objets a et p doivent être des listes de même taille.

Ici on choisit $a = [\![0, n-1]\!]$ (n est la taille de la matrice de transition). On identifie donc l'ensemble des états de la chaîne à $[\![0, n-1]\!]$. On rappelle que si M est une matrice carrée de taille $n \times n$, alors la commande len(P) renvoie l'entier n.

Ensuite, pour choisir les probabilités avec lesquelles on obtient p, il s'agit de savoir en quel état se trouve la chaîne, et utiliser les coefficients de la matrice de transition adaptés. Dans le $k$ième tour de boucle, la chaîne se trouve donc dans l'état X[k] et la loi de X[k+1] s'obtient donc en conditionnant par rapport à $X_k =$X[k]. Les valeurs de la loi de X_{k+1} conditionnée à $X_k =$X[k] sont sur la X[k]ième ligne de la matrice, que l'on extrait donc avec la commande M[X[k], :].

Nous verrons une autre méthode de simulation d'une chaîne de Markov (très similaire) dans le paragraphe suivant. Les deux sont à connaître absolument.

Évolution d'une maladie.

On modélise l'évolution d'une maladie au sein d'une population au cours du temps en classant les individus en trois groupes :

- Le groupe S des individus sains et non immunisés ;
- Le groupe M des individus malades ;
- Le groupe I des individus immunisés.

On appelle $E = \{S, M, I\}$ l'ensemble des **états** possibles. On donne la **loi d'évolution** suivante :

- À l'instant 0, tous les individus sont dans l'état S.
- La moitié des individus dans l'état S à l'instant n restent dans le même état à l'instant $n+1$ et l'autre moitié tombe malade ; un dixième des individus immunisés à l'instant n passent dans le groupe S à l'instant $n+1$, les autres restent immunisés. Un malade sur cinq le reste, les autres guérissent et deviennent immunisés.

1. Représenter le modèle précédemment décrit par un graphe ;

2. On s'intéresse à la suite (X_n) de variables aléatoires telle que X_n désigne l'état du système l'instant n, c'est à dire que $X_n(\Omega) = \{1, 2, 3\}$, où $(X_n = 1)$ (resp. 2, 3) correspond à l'état S (resp. M, N) à l'instant n.) La suite (X_n) est donc une chaîne de Markov.

a. Recopier et compléter le programme ci-dessous pour qu'il simule la suite des états X_0, X_1, ..., X_n (c'est-à-dire une trajectoire de la chaîne).

```
import numpy as np

def Exemple(n):
X=np.zeros(n)
X[0]=1
for k in range (n-1) :
    if X[k] == 1 :
        X[k+1] = np.random.choice(np.array([1,2,3]),.......)
    elif X[k] == 2 :
        X[k+1] = ......................................
    else :
        X[k+1] = np.random.choice(np.array([1,2,3]),.......)
    return X
```

b. Recopier, exécuter et interpréter les commandes suivantes

```
import matplotlib.pyplot as plt

y = Exemple(1000)
plt.plot(y)
plt.show()
```

c. Même question avec les commandes ci-dessous

```
U=np.zeros(100)
for k in range (100):
    U[k]=Exemple(1000)[999];


frequ = np.zeros(3)
for j in range (3):
    for k in range len(U):
        if U[k] == j+1 :
            frequ[j]=frequ[j]+1

plt.bar(frequ)
plt.show()
```

3. Pour $n \in \mathbb{N}$, on note

$$U_n = (P(X_n = 1), P(X_n = 2), P(X_n = 3)).$$

a. Déterminer la matrice de transition de la chaîne de Markov (elle vérifie donc

$$U_{n+1} = U_n A).$$

b. Montrer que, pour tout $n \in \mathbb{N}$, on a $U_n = U_0 A^n$. Préciser U_0, (c'est la loi initiale).

c. Définir, dans la console, la matrice de transition `A` et le vecteur de la loi initiale `u0`, et calculer $U_0 A^{1000}$. Comparer la cohérence des résultats avec ceux de la Question 2c.

4. Utiliser le programme du paragraphe précédent pour réécrire une fonction Exemple_bis(n) qui simule la trajectoire de la même chaîne de Markov.

Algorithme du Page Rank de Google.

Nous allons illustrer la méthode utilisée par Google pour classer les pages web par "indice de popularité". L'idée est d'étudier l'évolution des déplacements d'un individu sur internet, et de regarder la probabilité p_i d'être sur un site i au bout d'une très grand nombre de déplacements (on s'intéresse donc à la probabilité stationnaire de la chaîne). On dit alors que p_i est alors appelée "indice de popularité de la page i". Nous commencerons par regarder un cas très simple (et très édulcoré !) où il n'existerait sur le Web que 3 pages internet (!) puis on étudiera le cas général.

Un exemple avec 3 sites. À l'issue de sa requête, un internaute est susceptible d'aller sur trois sites A, B, et C. On suppose que l'internaute se déplace forcément vers un lien de la page Web où il se trouve.

- le site A contient un lien vers lui-même, un lien vers B, et un lien vers C ;
- le site B contient 5 liens vers lui-même, un vers A et un vers C ;
- le site C comporte un lien vers A, 7 liens vers B et 4 vers C.

À l'instant 0, l'internaute choisit au hasard l'un des trois sites. Par la suite, la probabilité de passer d'un site (à l'instant n) vers un autre (à l'instant n + 1) est proportionnelle au nombre de liens du premier site vers le deuxième.

Pour $n \in \mathbb{N}$, on désigne par a_n, b_n et c_n les probabilités des évènements A_n (resp. B_n, C_n) définis respectivement par le fait de se trouver sur le site A (resp. B, C) à l'instant n.

Enfin, on note X_n la variable aléatoire définie par

$$X_n(\omega) = \begin{cases} 1, & \text{si } \omega \in A_n \\ 2, & \text{si } \omega \in B_n \\ 3, & \text{si } \omega \in C_n \end{cases}$$

1. En justifiant rigoureusement au moins l'une des égalités, exprimer a_{n+1}, (resp. b_{n+1}, c_{n+1}) en fonction des trois réels a_n, b_n et c_n.

2. Écrire la matrice de transition S.

3. Mémoriser la matrice S dans votre éditeur Python puis compléter et exécuter le programme suivant qui permet d'afficher la trajectoire $(X_0, X_1, ..., X_n)$ de la chaîne de Markov (X_n).

```
import numpy as np
import matplotlib.pyplot as plt

def navigation(n,M,x0):
  X=np.zeros(n)
  X[0]=.........................
  for k in range(n-1):
    X[k+1]=np.random.choice(a=list(range(3)), p= ............)
  return X

plt.plot(navigation(100))
plt.show()
```

4. Est-ce qu'un site à l'air plus visité que les autres ? Moins visité que les autres ? Comment l'expliquer ?

5. Calculer avec Python les vecteurs de lois $U_0, U_1, ..., U_{10}$. (On recopiera et complétera le programme suivant pour définir une matrice E de taille 3×11 dont la $i-$ème colonne contiendra le vecteur U_{i-1}.

```
E=np.zeros((3, 11))
E(:, 1)=(1/3)*np.ones((3,1))
for k in range (9)
    E(:, k+2)=.....................
```

6. Tracer sur un même graphique les courbes des suites $(a_n), (b_n)$ et (c_n). Que remarque-t-on ?

7. On appelle "indice de notoriété" d'une page A la valeur $a = \lim a_n$. Donner une valeur approchée.

8. Regarder la valeur exacte des indices de notoriété en calculant la valeur exacte de l'état stable. On admettra que la probabilité invariante est unique.

On rappelle que la mesure invariante est un 1*-vecteur propre de la matrice transposée de la matrice de transition. On rappelle que dans la bibliothèque* numpy.linealg*, la fonction* eig(P) *sert à trouver le spectre et les espaces propres de* P*.*

Le cas général. On modélise l'ensemble des pages du Web par un ensemble fini d'états $E = \{1, 2, ..., N\}$ (le nombre N est bien très grand, de l'ordre de 10^{13}, mais reste fini).

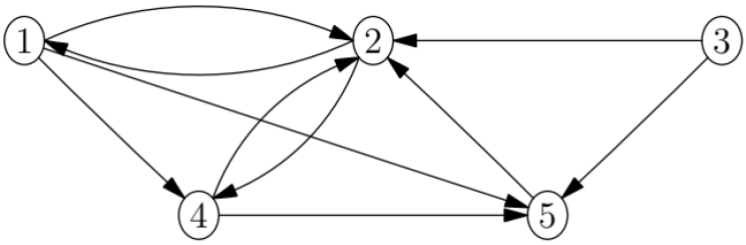
On modélise les déplacements de l'internaute par une chaînes de Markov (X_n) où la variable X_n est le numéro de la page où se situe l'internaute après n déplacements. Après avoir fait des études statistiques des comportements sur le Web, Google part du principe que :

- À l'instant 0, l'internaute choisit une page internet au hasard parmi les N pages ;
- Lorsqu'un internaute est sur une page j :
  - dans 85% des cas, il se déplacera au hasard vers l'une des ℓ_j pages pointées par j (ici, une page a au plus un lien vers une autre).
  - dans 15% des cas, il se déplacera au hasard vers n'importe laquelle des pages du Web.

La matrice de transition $M = (P(i, j))$ est alors définie par

$$P(i,j) = \begin{cases} 0,85 \times \frac{1}{\ell_j} + 0,15 \times \frac{1}{N}, & \text{si } j \text{ pointe vers } i \\ 0,15 \times \frac{1}{N}, & \text{sinon} \end{cases}$$

Nous allons illustrer l'algorithme de PageRank sur un groupe autonome de $N = 5$ pages différentes, ayant des liens pointant les pages les unes vers les autres, selon le diagramme suivant :



1. Taper et expliquer la commande

```
import numpy as np

A=0.15*(1/5)*np.ones((5,5))+0.85*np.array([[0, 1/2, 0, 0, 0], [1/3, 0, 1/2, 1/2, 0],
    [0, 0, 0, 0, 0], [1/3, 1/2, 0, 0, 0], [1/3, 0, 1/2, 1/2, 1]])
```

2. Déterminer la valeur exacte de la loi stationnaire (à l'aide de la commande eig()).

3. Recopier et exécuter les commandes suivantes. La chaîne converge-t-elle vers l'état stationnaire ?

```
import matplotlib.pyplot as plt

E=np.zeros((5, 11))
E(:, 0)=(1/5)*np.ones((5,1))
for k in range (10):
    E(:, k+1)=A np.dot E(:, k)

plt.bar([E(1,:),E(2,:),E(3,:),E(4,:),E(5,:)])
plt.show()
```

4. Classer les pages par indice de popularité. Les résultats vous semblent-ils cohérents ? Commenter.