TP 5

Chaînes de Markov

Compétences attendues.

- ✓ Savoir simuler n termes d'une chaîne de Markov.
- ✓ Savoir déterminer les lois d'une chaîne de Markov et les tracer.
- ✓ Savoir déterminer la loi stationnaire d'une chaîne de Markov.

Liste des commandes `Python` exigibles aux concours.

- Dans la librairie `numpy` : `np.arange, np.array, np.zeros, np.dot, np.transpose`
- Dans la librairie `numpy.linalg` : `al.inv, al.matrix_power, al.eig`
- Dans la librairie `numpy.random` : `rd.random()`
- Dans la librairie `matplotlib.pyplot` : `plt.plot, plt.show`

Anthony Mansuy
Professeur de Mathématiques en deuxième année de CPGE filière ECG au Lycée Clemenceau (Reims)
Page personnelle : `http://anthony-mansuy.fr`
E-mail : `mansuy.anthony@hotmail.fr`

1 Système Bonus-Malus des assurances

Le système des bonus-malus chez les assureurs des conducteurs a été introduit par la loi. Il poursuit essentiellement deux buts :

- responsabiliser les assurés et les inciter à plus de prudence au volant en pénalisant par une augmentation de prime le responsable d'un ou de plusieurs accidents tandis que dans la situation inverse, l'assuré bénéficiera d'une réduction de prime.
- ajuster le montant de la prime au cours du temps afin que celui-ci reflète le risque réel que représente l'assuré.

On s'intéresse ici à un modèle où l'assureur propose trois taux à ses assurés :

- taux 1 (bonus 50%) ;
- taux 2 (tarif de base) ;
- taux 3 (malus 100%).

Les nouveaux assurés font leur entrée dans le système au taux 2. Chaque année sans sinistre est récompensée par un passage du taux 3 vers le taux 2 ou du taux 2 vers le taux 1. Chaque année avec sinistre est pénalisée par un passage du taux 1 vers le taux 2 ou du taux 2 vers le taux 3.
Un bon conducteur a chaque année 5% de chance d'avoir un sinistre et un mauvais conducteur 15%. On considère qu'il y a autant de bons que de mauvais conducteurs.

Étude de la trajectoire d'un bon conducteur au cours du temps

Pour tout $n \in \mathbb{N}$, on note X_n la variable aléatoire qui à un bon conducteur associe son taux d'assurance lors de la n-ème année.

1. Justifier que $(X_n)_{n\in\mathbb{N}}$ est une chaîne de Markov et représenter son graphe probabiliste.
2. Déterminer sa matrice de transition A.
3. Représenter avec `Python` une trajectoire d'un bon conducteur sur 10 années. Pour cela, on pourra :
   - Créer une matrice E de taille 10×3 dont les lignes contiennent les lois de $X_0, X_1, \ldots, X_9$.
   - Tracer les courbes des suites $(P(X_n = 1)), (P(X_n = 2))$ et $(P(X_n = 3))$.
4. Pour tout $n \in \mathbb{N}$, on pose U_n le n-ième état probabiliste :

$$U_n = \begin{pmatrix} P(X_n = 1) & P(X_n = 2) & P(X_n = 3) \end{pmatrix}.$$

   L'objectif de cette question est de démontrer que la suite $(U_n)_{n\in\mathbb{N}}$ tend vers une loi stationnaire U.
   (a) Diagonaliser A avec `Python`. En déduire la limite de la suite de matrices $(A^n)_{n\in\mathbb{N}}$.
   (b) En déduire la loi stationnaire U de la chaîne de Markov.
   (c) Retrouver ce résultat avec la commande `al.matrix_power`.

Étude de la trajectoire d'un mauvais conducteur au cours du temps

5. (a) Déterminer la matrice de transition B dans ce cas.
   (b) Calculer la loi stationnaire V de cette chaîne de Markov. On utilisera cette fois-ci que V est l'unique vecteur-ligne stochastique solution de ${}^tB \times {}^tV = {}^tV$.

Qualité du système Bonus-Malus mis en place par l'assureur

6. (a) Calculer la probabilité, pour un conducteur au taux 3 après un nombre d'années assez important, d'être un mauvais conducteur. Le taux 3 est-il bien discriminant ?
   (b) Calculer la probabilité, pour un conducteur au taux 1 après un nombre d'années assez important, d'être un bon conducteur.
   Que pensez-vous du système de Bonus-Malus mis en place par l'assureur ?

2 Ruine du joueur

Un joueur joue à un jeu au casino. On suppose qu'il dispose d'une fortune de a euros et que celle du casino est de b euros. A chaque tour, il gagne un euro avec la probabilité $p \in]0,1[$ ou perd un euro (donné au casino) avec la probabilité $q = 1 - p$. Le jeu s'arrête lorsque le joueur ou le casino se retrouve ruiné.
On peut alors modéliser la situation par une chaîne de Markov homogène $(X_n)_{n\in\mathbb{N}}$ où X_n représente la fortune du joueur à l'issue du n-ième tour.

1. Déterminer la matrice de transition A associée à cette chaîne de Markov.

   Donner sans calcul deux états stables.

2. Compléter et exécuter la fonction suivante qui prend en entrée p, a et b, simule l'expérience décrite dans l'énoncé et renvoie en sortie une liste contenant l'état de la fortune du joueur à chaque tour :

```
def ruinejoueur(p,a,b):
    L = [a]
    while ..... and ..... :
        if ..... :
            L.append(L[-1]+1)
        else:
            L.append( ..... )
    return(L)
```

3. On ajoute les instructions suivantes à la suite du programme précédent :

```
p = 1/3
a = 20
b = 300
L = ruinejoueur(p,a,b)
N = list(range(len(L))
plt.plot(N, L)
plt.show()
```

   (a) Tester ce programme plusieurs fois. Comment interpréter les résultats graphiques obtenus ? Quel destin attend le joueur ?

   (b) Tester de nouveau plusieurs fois ce programme en changeant les valeurs de p, a et b. Que dire de l'issue de l'expérience ?

4. Compléter et exécuter la fonction suivante qui simule 1000 chaînes de Markov indépendantes et calcule à partir de ces 1000 simulations l'effectif N puis la fréquence f de l'événement "le joueur est ruiné à la fin du jeu" :

```
def ruinejoueurfrequence(p,a,b)
      N = 0
      for k in range(1000):
          L = ruinejoueur(p,a,b)
          if ..... :
              N = N+1
      f = .....
      return(f)
```

   Tester ce programme pour $a = 20$, $b = 300$ et différentes valeurs de p ($p < 1/2$, $p > 1/2$, p égal ou très proche de $1/2$) et estimer la probabilité de l'événement "le joueur est ruiné" dans chacun des cas.

   La valeur obtenue est-elle la valeur exacte de la probabilité que le joueur soit ruiné ? Pourquoi ? Quel est le lien entre fréquence et probabilité ?

3 Google 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 trier par ordre d'importance ?
- Comment choisir, pour un mot clé donné les pages les plus susceptibles d'intéresser l'internaute ?

L'idée des fondateurs de Google est d'analyser pour cela les liens existants entre les différentes pages : un lien d'une page i vers une page j est ainsi vu comme un "vote de confiance" pour la page j de la part de la page i. Pour mesurer le nombre de liens vers une page, et donc sa popularité, ils définissent un algorithme appelé PageRank dont voici le fonctionnement : on numérote l'ensemble des pages d'Internet de 1 à N. On modélise les déplacements d'un internaute par une chaîne de Markov $(X_n)_{n\in\mathbb{N}}$, où X_n est le numéro de la page où se situe l'internaute après n déplacements. On considère que :

- à l'instant 0, l'internaute choisit une page internet au hasard parmi les N pages ;
- lorsque l'internaute est sur une page i,
  - avec probabilité $c \in]0,1[$, il quitte la page et se dirige vers une page quelconque du web choisie aléatoirement ;
  - avec probabilité $1-c$, il se déplace aléatoirement vers une des ℓ_i pages accessibles depuis la page i (on considère qu'une page sans lien pointe sur elle-même).

1. Justifier que la matrice de transition $A = (a_{i,j})$ associée est donnée par :

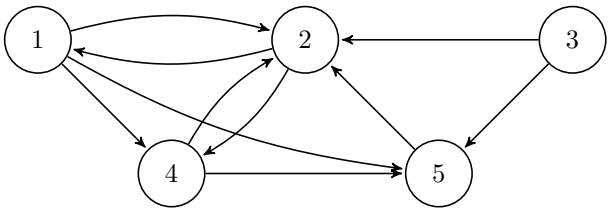
$$\forall (i,j) \in [\![1,N]\!]^2, \quad a_{i,j} = \begin{cases} \dfrac{c}{N} + \dfrac{1-c}{\ell_i} & \text{si la page } i \text{ pointe vers la page } j \\ \dfrac{c}{N} & \text{sinon.} \end{cases}$$

2. Le paramètre c est confidentiel chez Google. Une étude démontre cependant qu'un internaute relance sa navigation au hasard au bout de 6 pages en moyenne. Justifier qu'il est pertinent de choisir $c = 0,15$, ce qu'on fera dans la suite.

L'algorithme PageRank de Google consiste alors à déterminer la loi stationnaire U associée à cette chaîne de Markov, ce qui correspond à chercher un vecteur propre associé à la valeur propre 1 de ${}^tA$. Cette recherche est difficile de part la taille gigantesque de A qui rend impossible un calcul direct. On utilise pour cela des algorithmes d'approximation, exploitant notamment le fait que $A - \frac{c}{N} \times \text{ones}(N,N)$ possède beaucoup de zéros.

On définit alors le PageRank de la page i par le nombre $U(i) = \lim\limits_{n\to+\infty} P(X_n = i)$. Il s'agit de la probabilité, en surfant au hasard, d'arriver sur cette page. On remarquera 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 vue par Google comme plus pertinente étant donné qu'elle possède davantage de liens qui pointent vers elle.

3. On considère un réseau de $N = 5$ pages différentes, les liens entre elles étant représentés dans le graphe probabiliste ci dessous :



   (a) Écrire la matrice de transition associée.
   (b) Vérifier que la chaîne de Markov admet une unique loi stationnaire que l'on déterminera (à l'aide de la commande `al.eig` de la bibliothèque `numpy.linalg`).
   (c) Calculer les lois des variables $X_0, X_1, \ldots, X_{10}$ et les ranger dans une matrice E dont la i-ième ligne est la loi de X_i.
   Tracer les courbes des suites $(P(X_n = 1)), (P(X_n = 2)), \ldots, (P(X_n = 5))$.
   Y-a-t-il convergence en loi de la chaîne de Markov (X_n) vers la loi stationnaire ?
   (d) Déterminer le PageRank de chaque page internet, et les ranger par indice de popularité. Commenter.