Processus de Markov

Un processus de Markov est une suite d'expériences aléatoires où la prédiction des états futurs ne dépend que de l'état présent et non des états passés. On parle de processus sans mémoire. En d'autres termes, pour de tels processus, si on connaît le présent, la connaissance du passé n'apporte pas d'information supplémentaire utile pour la prédiction du futur.

1 Introduction : calcul matriciel en Python

Pour manipuler des matrices en Python, on a besoin de deux bibliothèques :

1. la bibliothèque `numpy` importée avec la commande `import numpy as np`
2. la bibliothèque `numpy.linalg` importée avec la commande `import numpy.linalg as al`

▷ La commande pour créer une matrice est `np.array([[ligne1],[ligne2],[ligne3],etc])`. En Python, une matrice se définit donc ligne par ligne. Définir en Python les matrices suivantes :

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & -1 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} -2 \\ 0 \\ 5 \end{pmatrix}, \quad E = \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \end{pmatrix}, \quad F = \begin{pmatrix} -2 & 0 & 1 \\ 1 & -2 & 0 \\ 0 & 1 & -2 \end{pmatrix}$$

▷ La commande `n,p=np.shape(M)` renvoie un couple (n, p) avec n égal au nombre de lignes de M et p égal au nombre de colonnes de M. Vérifier avec les matrices A, B, C, D et E que vous obtenez bien les tailles attendues.

▷ Quelle différence y a-t-il entre les commandes `np.array([1,0,0])` et `np.array([[1],[0],[0]])` ?

▷ Tester les opérations `A+A, 2*A, A*A, A**A, A/A`. Qu'obtenez-vous ?

▷ Pour effectuer le produit matriciel de deux matrices M et N, on utilise la commande `np.dot(M,N)`.
Calculer AF, AD, FD, FB. Qu'obtenez-vous ?

▷ Ecrire une fonction Python nommée `puissance` prenant en entrée un entier n et une matrice A et qui calcule A^n. La recopier ci-dessous.

▷ Pour calculer les puissances d'une matrice A, on peut aussi utiliser la commande `al.matrix_power(A,n)`. Comparer les résultats obtenus à l'aide de votre fonction `puissance` et de la commande `al.matrix_power(A,n)`.

2 Un premier exemple

Gustave, un étudiant d'ECG1, ne connaît que trois endroits dans sa vie : son lit où il dort, la cantine où il mange et la salle 02 où il travaille. Ses journées sont assez semblables les unes aux autres. Toutes les minutes, il peut soit changer d'activité, soit continuer celle qu'il était en train de faire.

- A l'instant 0, il dort.
- Quand il dort, il a 8 chances sur 10 de ne pas se réveiller la minute suivante.
- Quand il se réveille, il y a une chance sur 2 qu'il aille manger et une chance sur 2 qu'il se mette au travail.
- Le repas ne dure qu'une minute. Après avoir mangé, il y a 9 chances sur 10 qu'il se remette au travail et une chance sur 10 qu'il retourne dormir.
- Travailler est fatiguant mais Gustave est persévérant et quand il travaille il y a 19 chances sur 20 qu'il continue la minute suivant. Dans le pire des scénarios, il s'effondre dans son lit la minute suivante.

Pour tout $n \in \mathbb{N}$, on note D_n l'événement « Gustave dort à l'instant n » et $d_n = P(D_n)$, M_n l'événement « Gustave mange à l'instant n » et $m_n = P(M_n)$, et T_n l'événement « Gustave travaille » et $t_n = P(T_n)$.

1. (a) Exprimer en notations mathématiques les probabilités conditionnelles données par l'énoncé.

   (b) Démontrer les relations suivantes :

$$\begin{cases} d_{n+1} &= \frac{8}{10}d_n &+ \frac{1}{10}m_n &+ \frac{1}{20}t_n \\ m_{n+1} &= \frac{1}{10}d_n & & \\ t_{n+1} &= \frac{1}{10}d_n &+ \frac{9}{10}m_n &+ \frac{19}{20}t_n \end{cases}$$

   (c) On définit le vecteur colonne $U_n = \begin{pmatrix} d_n \\ m_n \\ t_n \end{pmatrix}$. Expliciter une matrice A telle que $U_{n+1} = AU_n$.

   On dit que A est la matrice de transition associée au processus de Markov.

   (d) Montrer que, pour tout entier naturel n, $U_n = A^n U_0$. Préciser U_0.

2. (a) Ecrire une fonction Python qui prend en entrée un entier naturel n et calcule le vecteur U_n.

   (b) Tester pour différentes valeurs de n. Que remarque-t-on ?

3. (a) Nous allons représenter sur Python l'état D par 1, l'état M par 2 et l'état T par 3.
   Compléter la fonction suivante pour qu'elle simule la suite des états $X_0, \ldots, X_n$ et les mémorise dans un vecteur X :

```
1  def etats_gustave(n):
2      X=np.zeros(n+1)
3      X[0]=.....
4      for k in range(n):
5          if X[k]==1:
6              x=rd.random()
7              if x<8/10:
8                  X[k+1]=.....
9              elif ..................... :
10                 X[k+1]=.......
11             else:
12                 X[k+1]=........
13         elif ............ :
14             y=.............
15             if ............. :
16                 X[k+1]=.............
17             else:
18                  X[k+1]=.............
19         elif ............... :
20             z=.............
21             if ............. :
22                 X[k+1]=...............
23             else :
24                 X[k+1]=...............
25     return X
```

(b) Tester la fonction pour différentes valeurs de n.

4. A l'aide de la fonction `etats_gustave(n)`, nous allons étudier les fréquences d'apparition des différents état de Gustave après 1 heure.

(a) Pour cela, compléter la fonction suivante qui prend en entrée un entier nb correspondant au nombre de simulations voulues et renvoie D, M et T les fréquences d'apparition des différents états.

```
def frequence_gustave(nb):
    D=0
    M=0
    T=0
    for k in range(nb):
        X=etats_gustave(60)
        if ..................:
            D=.................
        elif ..................:
            M=.................
        else:
            T=.....................
    return .....................
```

(b) Tester la fonction pour $nb = 1000$. Comparer ce résultat avec le résultat de la fonction `calcul_U(60)`. Qu'observez-vous ? Expliquez.

3 Un deuxième exemple

Gustave décide de réviser ses cours en vue d'un concours, et dispose pour cela de trois choix : réviser les mathématiques, l'économie ou l'anglais. On observe pendant un certain temps les efforts de Gustave et on constate les phénomènes suivants :

- A l'heure numéro 0, Gustave travaille l'économie.
- S'il révise les mathématiques à l'heure n, alors à l'heure $n+1$ il révisera encore des mathématiques avec une probabilité $\frac{1}{2}$ et sinon il révisera l'économie (mais pas l'anglais).
- S'il révise l'économie à l'heure n, il aura trois chances sur quatre de réviser les mathématiques l'heure suivante et une chance sur quatre de réviser l'anglais.
- S'il révise l'anglais à l'heure n, il se consacrera l'heure suivante à l'économie ou à nouveau à l'anglais, avec une probabilité de $\frac{1}{2}$ pour chaque.

On note A_n l'événement « Gustave révise les mathématiques à l'heure n", B_n l'événement « Gustave révise l'économie à l'heure n » et C_n l'événement « Gustave révise l'anglais à l'heure n ».

Par ailleurs, on pose $a_n = P(A_n), b_n = P(B_n), c_n = P(C_n)$ et $U_n = \begin{pmatrix} a_n \\ b_n \\ c_n \end{pmatrix}$.

1. (a) Exprimer, pour tout entier naturel $n, a_{n+1}, b_{n+1}, c_{n+1}$ en fonction de a_n, b_n, c_n.

(b) Expliciter une matrice A telle que $U_{n+1} = AU_n$

(c) De la même façon que dans la partie précédente, on a pour tout $n \in \mathbb{N}$, $U_n = A^n U_0$. Préciser U_0.

2. **Loi empirique de X_{10}.**
On représente sur Python l'état A par 1, l'état B par 2 et l'état C par 3.

(a) De la même manière que dans la partie précédente, on souhaite écrire une fonction permettant de simuler la suite des états X_0, X_1, X_2 etc
En vous inspirant de la fonction `etats_gustave(n)`, écrire une telle fonction qu'on nommera `etats_gustave_matiere(n)` où n désigne le nombre d'heures pendant lesquelles on simule les états.

(b) A l'aide de votre fonction `etats_gustave_matiere(n)`, écrire une fonction, s'inspirant de `frequence_gustave(nb)`, permettant de calculer la fréquence d'apparition des différents états après 10 heures de travail. Elle prendra en entrée nb égal au nombre de simulations voulues.

3. **Loi théorique de X_{10} :**

(a) Ecrire une fonction permettant de calculer U_n pour un n donné en entrée.

(b) Tester votre fonction pour $n = 10$. Est-ce cohérent avec les résultats obtenus pour la loi empirique de X_{10} ?

4. (a) Considérons la matrice $P = \begin{pmatrix} 3 & 3 & -1 \\ 2 & -4 & 0 \\ 1 & 1 & 1 \end{pmatrix}$. Calculer l'inverse de P. Vérifier le résultat avec `Python`.
La commande pour inverser une matrice A en Python est `al.inv(A)`.

(b) Déterminer la matrice D telle que $D = P^{-1}AP$. Vérifier le résultat avec `Python`.

(c) Montrer que, pour tout entier naturel n, $D^n = P^{-1}A^nP$.

(d) En déduire A^n, puis a_n, b_n et c_n en fonction de n.

(e) Calculer les limites lorsque n tend vers $+\infty$ de a_n, b_n et c_n. Comparer les résultats obtenus avec ceux des questions 2 et 3.