Python **(à rendre au plus tard le dimanche 18/12/2022)**

Devoir

Enseignant: Mohammed Sedki *pageweb:* masedki.github.io

Instructions :

Le devoir est à rendre sous format d'un fichier pdf exporté à partir d'un jupyter notebook et nommé `prenom_nom.pdf` ou `prenom1_nom1_prenom2_nom2.pdf`^ pour un binôme. **Le fichier pdf doit contenir les résultats et sorties des cellules de code faute de quoi la réponse sera incomplète.**

Problème I. Chaîne de Markov : (10 points)

Supposons que le processus aléatoire $(X_t)_{t\in\mathbb{N}}$ modélise la situation économique d'un pays par deux états possibles (par exemple, la croissance et la récession de l'économie) et définissons

$$p_{1,t} = \Pr\{\text{état 1 au temps } t\}, \quad p_{2,t} = \Pr\{\text{état 2 au temps } t\},$$

et

$$\pi_{ij} = \Pr\{\text{état } i \text{ au temps } t+1 | \text{ état } j \text{ au temps } t\}.$$

La chaîne de Markov est définie par

$$\begin{bmatrix} p_{1,t+1} \\ p_{2,t+1} \end{bmatrix} = \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix} \begin{bmatrix} p_{1,t} \\ p_{2,t} \end{bmatrix}.$$

En général, la chaîne de Markov à k états est donnée par

$$\underbrace{\begin{bmatrix} p_{1,t+1} \\ \vdots \\ p_{k,t+1} \end{bmatrix}}_{p_{t+1}} = \underbrace{\begin{bmatrix} \pi_{11} & \cdots & \pi_{1k} \\ \vdots & \ddots & \vdots \\ \pi_{k1} & \cdots & \pi_{kk} \end{bmatrix}}_{\Pi} \underbrace{\begin{bmatrix} p_{1,t} \\ \vdots \\ p_{k,t} \end{bmatrix}}_{p_t}.$$

où la Π est appelée, la matrice de transition de la chaîne de Markov.

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

Listing 1 – bloc d'importation

Exemple numérique 1.

$$\Pi = \begin{bmatrix} 0.9 & .0.25 \\ 0.1 & 0.75 \end{bmatrix}, \quad p_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}.$$

```
transition_matrix = np.array([[0.9, 0.25], [0.1, 0.75]])
print(transition_matrix)
initial_probability = np.array([0.5, 0.5])
print(initial_probability)
```

Listing 2 – exemple de loi initiale et matrice de transition

La chaîne de Markov évolue (évolution de la loi de X_t) comme suit

$$p_{t+1} = \Pi p_t, \quad t = 0, 1, \ldots,$$

1. Écrire une fonction `markov_chains` qui prend en entrée, une matrice de transition, une loi initiale et un nombre d'itérations et qui permet d'afficher l'évolution de p_t pour t allant de 0 à nombre d'itérations. On s'attend à une sortie comme suit

```
markov_chain = markov_chains(transition_matrix, initial_probability, 10)
print(markov_chain)

## sortie
[[0.5         0.5       ]
 [0.575       0.425     ]
 [0.62375     0.37625   ]
 [0.6554375   0.3445625 ]
 [0.67603437  0.32396563]
 [0.68942234  0.31057766]
 [0.69812452  0.30187548]
 [0.70378094  0.29621906]
 [0.70745761  0.29254239]
 [0.70984745  0.29015255]
 [0.71140084  0.28859916]]
```

Listing 3 – exemple d'exécution de la fonction

2. À l'aide des fonctions et fonctionnalités offertes par le module `matplotlib.pyplot`, produire une figure du style de la figure 1.

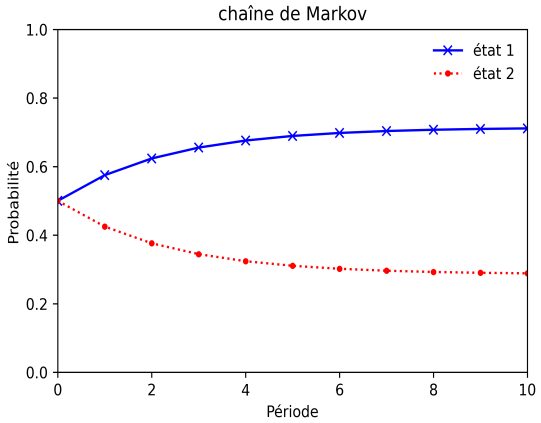

FIGURE 1 – Évolution de la loi p_t pour t allant de 0 à 10.

3. Généraliser le code précédent pour différentes lois initiales p_0 pour produire la figure 2.

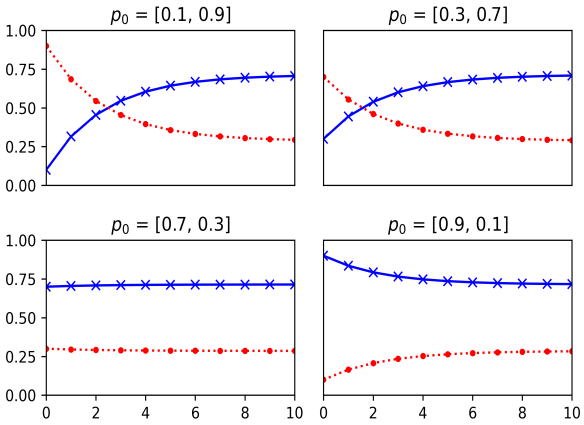

FIGURE 2 – Évolution de la loi p_t pour différentes lois initiales p_0.

Loi stationnaire d'une chaîne de Markov :
On dit qu'une chaîne de Markov a atteint la stationnarité si sa loi courante vérifie l'égalité suivante

$$p^* = \Pi p^*.$$

Dans le cas d'une chaîne de Markov à deux états $k = 2$, nous avons la formule explicite de p^* :

$$p_1^* = \frac{1 - \pi_{22}}{2 - \pi_{11} - \pi_{22}}, \quad p_2^* = \frac{1 - \pi_{11}}{2 - \pi_{11} - \pi_{22}}.$$

4. La loi stationnaire d'une chaîne de Markov dépend-t-elle de sa loi initiale ? Implémenter et exécuter cette formule sous *python* pour calculer la loi stationnaire sur notre exemple précédent à deux états.

Exemple numérique 2.
Soient une matrice de transition et trois vecteurs de lois initiales donnés par

$$\Pi = \begin{bmatrix} 0.9 & 0.1 & 0 & 0 \\ 0.1 & 0.9 & 0 & 0 \\ 0 & 0 & 0.2 & 0.8 \\ 0 & 0 & 0.8 & 0.2 \end{bmatrix}, \quad p_0^1 = \begin{bmatrix} 0.2 \\ 0.8 \\ 0 \\ 0 \end{bmatrix}, \quad p_0^2 = \begin{bmatrix} 0.1 \\ 0.4 \\ 0.4 \\ 0.1 \end{bmatrix}, \quad p_0^3 = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.2 \\ 0.8 \end{bmatrix}.$$

5. Tracer les équivalents de la figure 1 pour chacune des lois initiales p_0^1, p_0^2 et p_0^3. Commenter.

Problème II. Intervalles de confiance : (10 points)

Durant le cours de statistique, nous avons abordé l'estimation de la probabilité de succès dans un modèle de Bernoulli (probabilité d'obtenir un pile ou face). Nous avons alors obtenu l'estimateur par maximum de vraisemblance suivant

$$\hat{p}_n = \frac{1}{n} \sum_{i=1}^{n} X_i$$

Les intervalles de confiance nous permettent d'estimer à quel point nous pouvons nous rapprocher de la valeur réelle que nous estimons. Logiquement, cela semble étrange, n'est-ce pas ? Nous ne connaissons pas vraiment la valeur exacte de ce que nous estimons (sinon, pourquoi l'estimer ?), et pourtant, d'une manière ou d'une autre, nous savons à quel point nous pouvons nous approcher de quelque chose que nous admettons ne pas connaître ? En fin de compte, nous voulons faire des déclarations comme *la probabilité que la valeur estimée soit dans notre intervalle est de 90%*. Malheureusement, c'est quelque chose que nous ne pourrons pas dire avec nos méthodes. le mieux que nous puissions faire est de dire à peu près ce qui suit : *si nous réalisons l'expérience plusieurs fois, alors l'intervalle de confiance piègera la vraie valeur du paramètre dans 90% des cas.*

Revenons à notre exemple de pile ou face (variable aléatoire de Bernoulli). Une façon d'obtenir un intervalle de confiance est d'utiliser l'inégalité de Hoeffding qui s'énonce dans le de nos variables de Bernoulli comme suit :

$$\mathbb{P}(|\hat{p}_n - p| > \epsilon) \leq 2 \exp(-2n\epsilon^2)$$

Nous pouvons obtenir un intervalle de confiance $\mathbb{I} = [\hat{p}_n - \epsilon_n, \hat{p}_n + \epsilon_n]$, où ϵ_n est donné par

$$\epsilon_n = \sqrt{\frac{1}{2n} \log \frac{2}{\alpha}}$$

de sorte à ce que la partie droite de l'inégalité de Hoeffding soit égale à α. Nous avons donc

$$\mathbb{P}(p \notin \mathbb{I}) = \mathbb{P}(|\hat{p}_n - p| > \epsilon_n) \leq \alpha$$

.

Nous avons aussi, $\mathbb{P}(p \in \mathbb{I}) \geq 1 - \alpha$. Comme exemple numérique, prenons $n = 100$ et $\alpha = 0.05$, nous obtenons $\epsilon_n = 0.136$. On déduit l'intervalle de confiance à 95%

$$\mathbb{I} = [\hat{p}_n - \epsilon_n, \hat{p}_n + \epsilon_n] = [\hat{p}_n - 0.136, \hat{p}_n + 0.136]$$

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import  bernoulli
b=bernoulli(.5) # declaration d'une Bernoulli de parametre 1/2
n=100
```

Listing 4 – import

1. Écrire un code sans boucle `for` pour calculer l'intervalle de confiance obtenu par l'inégalité de Hoeffding 200 fois pour le paramètre d'une loi de Bernoulli de $p = 0.5$ et $n = 100$. Afficher le pourcentage de cas où le IC capte la vraie valeur du paramètre.

Le résultat de la question précédente montre que l'estimateur et l'intervalle correspondant ont pu piéger la vraie valeur au moins 95% du temps. C'est ainsi qu'il faut interpréter l'action des intervalles de confiance. Cependant, la pratique habituelle consiste à ne pas utiliser l'inégalité de Hoeffding et à utiliser à la place des résultats de la normalité asymptotique.

Dans le cas du modèle de Bernoulli, le théorème central limite ou la normalité asymptotique de l'estimateur par maximum de vraisemblance nous permet d'utiliser la normalité suivante :

$$\frac{\hat{p}_n - p}{\mathtt{se}} \rightsquigarrow \mathcal{N}(0,1) \quad \text{où} \quad \mathtt{se} = \sqrt{\mathbb{V}(\hat{p}_n)}.$$

Par simple de calcul de variance de moyenne empirique, nous savons que $\mathbb{V}(\hat{p}_n) = \frac{p(1-p)}{n}$ et en remplaçant p par son estimateur, nous obtenons l'approximation de l'écart type

$$\hat{\mathtt{se}} = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}.$$

À partir du résultat de normalité précédent sur le paramètre d'une loi de Bernoulli, on sait que$\hat{p}_n \sim \mathcal{N}(p, \widehat{\mathtt{se}}^2)$. Ainsi, si nous souhaitons obtenir un intervalle de confiance à $1-\alpha$, nous pouvons calculer

$$\mathbb{P}(|\hat{p}_n - p| < \xi) > 1 - \alpha$$

à l'aide de la loi asymptotique $\mathcal{N}(0, \widehat{\mathtt{se}}^2)$. Il suffit de calculer

$$\int_{-\xi}^{\xi} \mathcal{N}(0, \widehat{\mathtt{se}}^2) dx > 1 - \alpha$$

Le calcul de ξ semble compliqué dans la formule précédente mais `scipy.stats` nous fournit ce dont nous avons besoin.

2. À l'aide de la méthode `stats.norm.interval`, écrire un code sans boucle `for` pour calculer l'intervalle de confiance obtenu par normalité asymptotique 200 fois pour le paramètre d'une loi de Bernoulli de $p = 0.5$ et $n = 100$. Tracer sur la même figure, les 200 IC obtenus par l'inégalité de Hoeffding, les 200 IC obtenus par normalité asymptotique et les 200 valeur de l'estimateur comme dans la figure 3

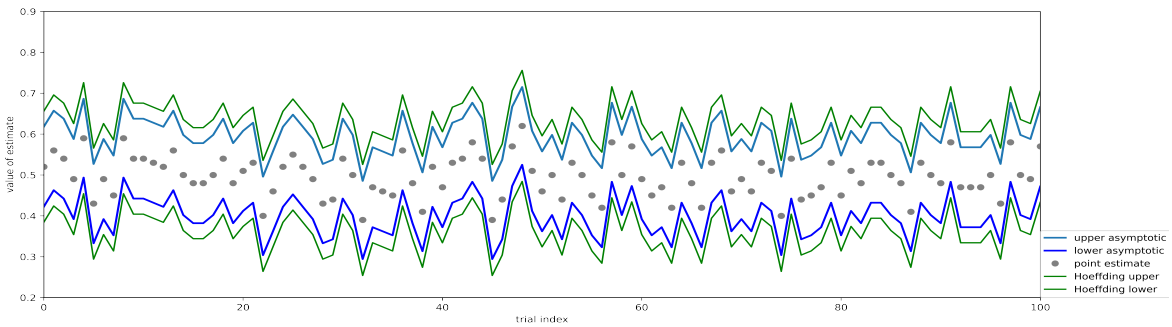

FIGURE 3 – Bandes de confiance Hoeffding versus normalité asymptotique.