YNOV - Master 2 DATAEN

Maths DataScience

Probabilités et statistiques, Algèbre linéaire avec Python

1. Vecteurs et matrices

Un élément unique, un vecteur ligne ou colonne sont des cas particuliers de matrice (tableau 2D).

Une des bibliothèques de Python les plus utilisées en algèbre linéaire est la bibliothèque numpy. On commence par l'importer pour pouvoir l'utiliser pour la suite.

```
import numpy as np
```

Exercice 1.

Création de tableaux, accès à un élément d'un tableau

Création d'un vecteur ligne :

```
L1 = np.array([10, 20, 30, 40])
L1
```

Création d'un vecteur colonne :

```
C1 = np.array([[10], [20], [30], [40]])
C1
```

Question : Comment obtenir C1 à partir de L1 ?

```
C2 = L1.reshape((4, 1))
C2
```

Accès à un élément d'un tableau. Les indices commencent à 0.

```
L1[2]

C1[2]

C1[2][0]
```

Création d'une matrice :

```
M1 = np.array([[10, 20, 30], [40, 50, 60], [70, 80, 90]])
M1
```

La matrice est 3 x 3 :

```
M1.shape
```

Taille d'une matrice, d'un vecteur, d'un tableau :

```
np.size(L1)

np.size(C1)

np.size(M1)

np.size(M1[1])
```

Accéder aux éléments d'un tableau 2D (indiceLigne, indiceColonne) :

```
iL = 1
iC = 2
M1[iL, iC]
```

Extraire une ligne, une colonne :

```
print(M1[1, :])
print(M1[:, 2])
```

Extraire une sous-matrice :

```
M1[1:2, 0:1]
```

Fonctions prédéfinies de création de tableaux :

```
M2 = np.zeros(3)
M2

M3 = np.zeros((1,3))
M3

M4 = np.zeros((2, 3))
M4

M5 = np.ones((2,4))
M5

Id = np.eye(3)
Id
```

Créer un tableau à partir de ses lignes :

```
L1 = np.array([5, 1, 3])
L2 = np.array([10, 4, 2])
L3 = np.array([3, 5, 0])
M6 = np.concatenate(([L1], [L2], [L3]))
M6
```

Redimensionner une matrice :

```
- le nombre d'élements est le même avant et après
- le parcours ligne par ligne donne la même suite d'éléments

L1 = np.array([5, 1, 3])
L2 = np.array([10, 4, 2])
M7 = np.concatenate(([L1], [L2]))
M7

M8 = np.reshape(M7, (3, 2))
M8
```

Opérations sur les matrices

Produit scalaire de deux vecteurs :

```
L1 = np.array([1, 2, 3])
L2 = np.array([1, 2, 3])
prod = np.dot(L1, L2)
prod
```

Soit M la matrice définie comme suit :

```
L = np.linspace(0, 10, 9)
M = np.array([L, L, L])
M
```

Somme de deux matrices :

```
M + M
```

Produit élément par élément de deux matrices :

```
M * M
```

Produit de deux matrices (compatibles) :

```
np.matmul(M, M) #erreur : pourquoi ?

np.matmul(M, M.T)
```

A présent, nous allons illustrer la puissance de le bibliothèque numpy. Pour cela nous allons d'abord définir notre propre fonction prod_matrices calculant le produit en utilisant des boucles imbriquées :

```
def prod_matrices(M1, M2):
    M = np.zeros((M1.shape[0], M2.shape[1]))
    for i in range(M1.shape[0]):
        for j in range(M2.shape[1]):
            for k in range(M1.shape[1]):
                M[i, j] += M1[i, k] * M2[k,j]
    return M
```

Test de la fonction :

```
M1 = np.array([[2, 1, 1], [4, 5, 7]])
M2 = np.array([[3, 2], [2, 1], [5, 4]])
prod_matrices(M1, M2)
```

Avec un peu plus d'éléments :

```
L = np.linspace(0, 1000, 1000)
M = np.array([L, L, L])

P = prod_matrices(M, M.T)
P
```

La fonction time permet d'afficher des statistiques sur le temps d'exécution d'une instruction Python. Exécutez les instructions suivantes et observez les résultats :

```
%time P = prod_matrices(M, M.T)

%time P= np.matmul(M, M.T)
```

La différence est encore plus flagrante si on utilise des matrices creuses :

```
L1 = np.linspace(0, 1000, 1000)
L2 = np.zeros(1000)
M = np.array([L2, L2, L1])

%time P = prod_matrices(M, M.T)

%time P= np.matmul(M, M.T)
```

Exercice 2.

Chercher des éléments dans un tableau

1. A tester :

```
from random import sample, choices

V1 = np.reshape(choices(range(1, 25), k=200), (200, ))
V1

V2 = V1[V1>20]
V2
```

1. Créer deux vecteurs ligne d'entiers Va et Vb, de même taille, contenant des valeurs entières aléatoires.

```

```

1. Tester le code suivant. Que contient V?

```
V = np.array([Va[i] if Va[i] > Vb[i] else Vb[i] for i in
range(Va.shape[0])])
V
```

1. Quelle est la fonction de numpy qui permet d'obtenir le même résultat? Tester.

```

```

2. Probabilités et statistiques, lois usuelles

Dans cette section, nous allons passer en revue quelques éléments de probabilités et statistiques. Nous allons essentiellement faire de l'inférence statistique et manipuler la loi normale.

Loi des grands nombres

On lance une pièce biaiséé. On obtient pile avec probabilité $p \in [0,1)$ et face avec probabilité $q=1-p$.

Ecrire une fonction lancer(n, p) qui simule le lancer de ce dé n fois avec la probabilité p d'obtenir pile et qui retourne le nombre de fois où pile est obtenu.

Indication : uniform(0,1) de la bibliothèque random retourne un nombre réel aléatoire tiré uniformément dans l'intervalle $[0,1)$.

```

```

Essayer avec différentes valeurs p et des valeurs de n croissantes. Qu'observez-vous ?

```

```

Loi normale

La loi normale est très utilisée en statistiques. Elle est, entre autre, la loi limite pour d'autres lois.

Commençons par générer des nombres suivant une loi normale mais avec différentes valeurs pour les paramètres μ et σ.

Exécutez la cellule suivante et observez le résultat :

```
from scipy.stats import norm
import matplotlib.pyplot as plt
%matplotlib inline

mu = [0, 1, 2, 3]
sigma = [1, 2, 3, 4]

x_min = -16
x_max = 16
x_nb = 100

x = np.linspace(x_min, x_max, x_nb)

for i in range(len(mu)):
    y = norm.pdf(x, mu[i], sigma[i])
    plt.plot(x, y, label='mu='+str(mu[i])+', sigma='+str(sigma[i]))
plt.legend(loc='upper right')
plt.show()
```

Commentez les courbes et observez l'impact des valeurs de σ.

Inférence des paramètres d'une loi normale

On insère une bannière publicitaire dans une page web. On trace les connexions à cette page auprès de 50 utilisateurs et ce pendant 1000 jours. Pour chaque visite on note si l'utilisateur a cliqué sur la bannière ou non.

Les chiffres des clics (par jour) sont donnés dans le fichier baniere.csv disponible à l'adresse https://www.labri.fr/perso/zemmari/datasets/baniere.csv.

1. Charger les données dans une variable de nom data en utilisant la fonction read_csv de la bibliothèque pandas (elle accepte les urls).

```

```

1. Donner une estimation de la probabilité qu'un visiteur du site (choisi au hasard) clique sur la bannière.

```

```

1. Le code suivant permet de tracer l'histogramme des données comme une courbe. Qu'observez-vous pour la forme de la courbe ?

```
from collections import Counter

compteur = Counter(list(data[0]))
```

```
#print(compteur)

x = list(compteur.keys())
print(x)
y = [compteur[x[i]] for i in range(len(x))  ]
print(y)
plt.scatter(x, y, c='red', marker='+')
```

1. Calculez l'estimateur $\hat{\mu}$ de la moyenne et $\hat{\sigma}$ de la variance.

Indication : des estimateurs non biaisés de la moyenne et de la variance d'une série $(x_i)_{1\leq i\leq n}$ sont donnés par :

$$\hat{\mu}=\frac{1}{n}\sum_{i=1}^{n} x_i, \hat{\sigma}^2=\frac{1}{n}\sum_{i=1}^{n}(x_i-\hat{\mu})^2.$$

On peut néanmoins utiliser directement les fonctions average et std de la bibliothèque numpy.

```

```

1. Dessinez la courbe de la fonction de répartition de la loi normale de paramètres $\hat{\mu}$ et $\hat{\sigma}$ sur la même figure que celle de la question 4. Qu'observez vous ?

Exercice 3

Le nombre de véhicules passant sur une petite route de campagne en une journée est une variable aléatoire X qui suit une loi de Poisson $P(5)$. On va simuler une année de circulation sur cette route. Pour cela, importons la bibliothèque numpy.random à l'aide de l'instruction import numpy.random as rd, et utilisons la commande A = rd.poisson(5,[52,7]). La i-eme ligne de A correspond à la i-eme semaine de circulation. On obtient ainsi une matrice A qui contient 52×7 nombres tirés suivant la loi $P(5)$.

1 Déterminer le nombre maximal de véhicules circulant sur la route en une journée durant l'année. Et le nombre maximal de véhicules un mercredi durant l'année ?

2 Déterminer les numéros des semaines où la route a connu son maximum de fréquentation.

3 Calculer le nombre moyen de véhicules par jour durant ces 364 jours. Recommencez plusieurs fois en recréant la matrice A. Que constatez-vous ? Était-ce prévisible ?

5 Toujours en une seule ligne de commande, déterminer le nombre de jours où la route a connu son minimum de fréquentation.

4 En une seule ligne de commande, évaluer la probabilité qu'une journée voit passer plus de 8 véhicules. Sauriez-vous calculer la valeur exacte de cette probabilité ?

Exercice 4:

On lance indéfiniment un dé équilibré à 6 faces numérotées de 1 à 6. Écrire un programme permettant de simuler ces lancers de dé et qui renvoie le rang du lancer où l'on obtient pour la première fois un numéro déjà obtenu

Exercice 5

À un guichet, des clients peuvent venir expédier ou retirer un colis. Au cours d'une journée, le nombre de clients N qui s'y présentent suit la loi de Poisson de paramètre λ (où $\lambda \in R_{+¿¿}$). Chaque client a une probabilité p où $p \in ¿0,1¿$ de venir pour expédier un colis et $1-p$ pour en retirer un. On note C le nombre de colis expédiés dans la journée.

1 Pour tout $k \in N$, déterminer l'espérance de C conditionnellement à l'événement $[N=k)$.

2 En déduire l'espérance de C.

3 On suppose dans cette question que $p=0.3$ et $\lambda=2.5$.

(a) Écrire un programme renvoyant un vecteur C contenant $n=10000$ réalisations de la variable aléatoire C.

(b) Comparer alors l'espérance empirique obtenue avec l'espérance théorique.

(c) On souhaite représenter le diagramme en bâtons des fréquences de l'échantillon.

i. Exécuter la commande np.mean(C<=4). Que dire des valeurs prises par C ?

ii. Représenter le diagramme en bâtons des fréquences de l'échantillon

Exercice 6

1 Simuler avec la fonction Exponentielle $N=10000$ valeurs de la loi $E(0.5)$.

2 Tracer la courbe représentative de la densité f de la loi $E(0.5)$.

3 Tracer l'histogramme des fréquences de l'échantillon obtenu (on prendra pour cela une subdivision c de l'intervalle $[0,10)$ en $p=100$ intervalles de même longueur).

Comparer l'histogramme des fréquences de l'échantillon à la courbe représentative de f. Qu'en pensez vous ?

4 Procéder de même pour la loi $\Gamma(3)$.

BON COURAGE