TD2 centrale 2 algèbre probas

I Simulations de lois de probabiltés

Exercice 1 *(centrale extrait) On considère N étudiants souhaitant réussir à un examen. A chaque fois qu'un étudiant passe l'examen, il a une probabilité $p \in]0,1[$ de réussite.*
Tous les passages de l'examen sont indépendants. Soit X_k le nombre de passage nécessaires au candidat n° k pour réussir l'examen. On pose $X = X_1 + \cdots + X_N$ et $Y = \max(X_1, \ldots, X_N)$.

1. *Ecrire des fonctions `X(N,p)`et `Y(N,p)` qui simulent X et Y.*
2. *Donner une valeur approché de $E(X)$ et $E(Y)$ avec, par exemple, $N = 3$ et $p = \frac{1}{3}$.*

Solution de l'exercice:

```
import numpy.random as rd

#q1
def X(N,p):
    t=rd.geometric(p, N)
    k=0
    for i in range(N):
        k+=t[i]
    return k

def Y(N,p):
    t=rd.geometric(p, N)
    k=-float('inf')
    for i in range(N):
        if k<t[i]:
            k=t[i]
    return k
```

Le mieux est de grouper les deux fonctions précédentes en une seule fonction:

```
def XY(N,p):
    t=rd.geometric(p, N)
    k=0
    b=-float('inf')
    for i in range(N):
        k+=t[i]
        if b<t[i]:
            b=t[i]
    return (k,b)

#q2
```

L'idée fonctions suivantes (calculer la moyenne empirique pour obtenir une approximation de l'espérance), repose sur la loi faible des grands nombres.

```
def EX(k,N,p):
    n=0
    for i in range(k):
       n+= X(N,p)
    return (n/k)
print(EX(1000,3,0.3))

def EY(k,N,p):
    n=0
```

```
    for i in range(k):
       n+= Y(N,p)
    return (n/k)
print(EY(1000,3,0.3))
```

II Algèbre linéaire

Exercice 2 *Soit* $A = \begin{pmatrix} 0 & \dots & 0 & 1 \\ 1 & 0 & & 0 \\ & \ddots & \ddots & \vdots \\ O & & 1 & 0 \end{pmatrix}$ *et* $B = \begin{pmatrix} A & I_n \\ 0_n & A \end{pmatrix}$.

1. *Définir une fonction* `A(n)` *qui renvoie la matrice* A.
2. *Définir une fonction* `B(n)` *qui renvoie la matrice* B *définie par blocs*
3. *Pour* $n = 3$ *et* $i \in [[2, 5]]$ *calculer* B^i.

Solution de l'exercice:

```
import numpy as np
import numpy.linalg as alg
#q1
def A(n):
    A0=np.zeros((n,n))
    for i in range (n):
        A0[i][i-1]=1
    return A0
#print(A(3))
#q2
def B(n):
    A0 = A(n)
    I=np.eye(n)
    O=np.zeros((n,n))

    C=np.concatenate((A0,I),axis=1)

    D=np.concatenate((O,A0),axis=1)
    B=np.concatenate((C,D),axis=0)
    return B
#print(B(3))
#q3
B=B(3)
Bi=B
for i in range(4):
    Bi=np.dot(B,Bi)
    print(Bi)
#on remarque une certaine regularite on trouverait que B**i est une matrice par bloc avec des A**i et i*A**
```

Exercice 3 *Soit* $A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & -1 & 1 \\ 1 & 2 & 1 \end{pmatrix}$ *et* $B = \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix}$

Avec python numpy:

1. *Résoudre le système linéaire* $\begin{cases} x_1 + 2x_2 + 3x_3 = 1 \\ x_1 - x_2 + x_3 = 2 \\ x_1 - 2x_2 + x_3 = 5 \end{cases}$
2. *Calculer le polynôme caractéristique de* A.

3. *Donner une base de vecteurs propres de A.*

Solution de l'exercice:

```
import numpy as np
import numpy.linalg as alg
A=np.array([[1,2,3],[1,-1,1],[1,2,1]])
B=np.array([1,2,5])

#q1
k=alg.solve(A,B)
print(k)

#q2

d=np.poly(A)
print(d)#coef du poly cara
#ou utiliser alg.eigvals(A) -> donne liste valeurs propres pour retrouver poly cara

#q3
c,f= alg.eig(A)
print(f)#coordonnees des vect propres
```

Exercice 4 *(Centrale 2) On note H_n l'ensemble des matrices de $\mathcal{M}_n(\mathbb{R})$ à coefficients dans $\{-1,1\}$. dont les colonnes sont orthogonales pour le produit scalaire canonique et SH_n, l'ensemble des matrices symétriques de H_n.*

1. *Soit A une matrice à coefficients dans $\{-1,1\}$. Montrer que $A \in H_n \Leftrightarrow A \times A^T = nI_n$.*
2. *Que renvoie l'instruction:`[[x,y] for x in [0,1]for y in [0,1]]`?*
   Ecrire une suite d'instructions qui donne l'ensemble des matrices de H_4. Quel est le cardinal de H_4. Même question avec SH_4.
   Trouver $\{tr(A),\ A \in SH_4\}$. Le démontrer sans python.

Solution de l'exercice:

```
#q2
#print([[x,y] for x in [0,1] for y in [0,1]])
#renvoit les differents tuples possibles composes des valeurs 0 et ou 1


#on commence par determiner l'ensemble des matrice de dim 4,4 \U{e0} coefficients dans {-1,1}
e=[[a,b,c,d] for a in [-1,1] for b in [-1,1] for c in [-1,1]for d in [-1,1]]

M=[np.array([a,b,c,d]) for a in e for b in e for c in e for d in e]

print (M[1])

#on recupere ensuite les matrices verifiant la condition sur la transpose
H=[]
n=len(M)
for i in range (n):
    l=M[i]
    lt=np.transpose(M[i])
    #if np.dot(l,lt)==np.diag([4,4,4,4]): pose probleme
    if np.array_equal(np.dot(l,lt),np.diag([4,4,4,4])): # test egalite matricielle
        H.append(l)

print(len(H))#cardinal H
```

```
#on cherche ensuite les matrices symetriques dans H:

Liste_S4=[]
for m in Liste_H:
    if np.array_equal(m,np.transpose(m)):
        Liste_S4.append(m)
t=[]
for m in Liste_S4:
    t.append(np.trace(m))
t=set(t) # conversion en ensemble, ce qui supprime les doublons
```

On trouve que $tr(A) \in \{-4, 0, 4\}$.

Si $A \in S_4$, alors $A \times A^T = nI_n$ et $A = A^T$ donc $A^2 = nI_n$. On en déduit que $X^2 - n = (X - \sqrt{n})(X + \sqrt{n})$ est un polynôme annulateur de A scindé à racine simples A est diagonalisable et $sp(A) \subset \{-\sqrt{n}, \sqrt{n}\}$. On a donc $tr(A) = p\sqrt{n} + (n-p)(-\sqrt{n}) = (2p-n)\sqrt{n}$ avec $p \in [[0, n]]$. Comme $A \neq \lambda I_n$, on ne peut pas avoir $p = 0$ ou $p = n$ donc $p \in [[1, n-1]]$. Si $n = 4$; si $p = 1$, $tr(A) = -4$, si $p = 2$, $tr(A) = 0$, si $p = 3$, $tr(A) = 4$.

III Polynômes

Exercice 5 *On pose $T_0 = 1$ et $T_1 = X$ et $\forall n \in \mathbb{N}$, $T_{n+2} = 2XT_{n+1} - T_n$.*

1. *Avec python polynomial, définir une fonction qui renvoie le polynôme T_n (sous forme d'objet Polynomial).*

2. *Vérifier la valeur du coefficient dominant obtenu.*

3. *En examinant T_i pour $i \in [[1, 10]]$, que peut on conjecturer sur la suite (T_n)? (parité, nombre de racines)*

Solution de l'exercice:

```
from numpy.polynomial import Polynomial
x=Polynomial([0,1])#attention ordre coef on commence par le coef constant
def T(n):
    t=[Polynomial([1]),Polynomial([0,1])]
    k=2
    while k!=(n+1):
        a=2*x*t[k-1]-t[k-2]
        t.append(a)
        k+=1
    return t
l=T(10)
for i in range (10):
    print(l[i].coef,len(l[i].roots()))

#coef dominant =2**n et Tn est pair si n pair
```

Exercice 6 *En urilisant python polynomial, Donner l'expression de la dérivée $10^{ème}$ de $f : x \mapsto e^{-x^2}$.*

Solution de l'exercice:

On commence par montrer l'existence d'une suite de polynômes (P_n) vérifiant:
$\forall n \in \mathbb{N}$, $\forall x \in \mathbb{R}$, $f^{(n)}(x) = P_n(x)\, e^{-x^2}$ et on établit une relation de récurrence. Pour la suite, on peut s'inspirer de l'exercice précédent.

Exercice 7 *Déterminer le reste dans la division euclidienne de $X^n + 1$ par $X^4 + 1$ pour $n \in [1, 40]$. Que constat-on?*

Solution de l'exercice:

```
from numpy.polynomial import Polynomial
P=Polynomial([1,0,0,0,1])
for i in range(1,40):
    L=[1]+[0 for i in range(n-1)]+[1]
    print (L%P.coef)
# on remarque qu'un reste nul apparait avec une certaine regularite
```