Agro-Veto – Planche 6

Question de cours

Espérance d'une variable aléatoire réelle admettant une densité f continue.

Exercice avec préparation

On rappelle les commandes et fonctions Python pouvant être utiles dans le cadre de cet exercice.

— On importe le module `numpy` sous l'alias `np` en écrivant en tête de script : `import numpy as np`.

— L'expression `np.random.rand(n,p)` donne une matrice numpy de forme (`shape`) `(n,p)` dont les entrées sont des réels tirés indépendamment et uniformément dans l'intervalle $]0,1[$.

— L'expression `np.linalg.matrix_rank(A)` donne le rang de la matrice numpy `A`.

— L'expression `np.dot(A,B)` donne le produit AB des matrices numpy $A =$`A` et $B =$`B`.

Soit $n \geq 2$, $A \in \mathcal{M}_n(\mathbb{R})$, $X \in \mathcal{M}_{n,1}(\mathbb{R})$ et $Y \in \mathcal{M}_{1,n}(\mathbb{R})$ tel que $\sigma = YAX$ est un nombre réel non nul. On note $r = \text{rang}(A)$.

On pose alors $B = \dfrac{1}{\sigma} AXYA$ et $C = A - B$.

1. Étude d'un exemple. Soit $A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$, $X = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ et $Y = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$.
   (a) Déterminer le rang de A
   (b) Calculer σ, B et déterminer le rang de C.
2. Étude informatique.
   (a) Écrire une fonction Python qui prend en entrée des matrices numpy `A`$= A$, `X`$= X$ et `Y`$= Y$ de formes appropriées et renvoie le couple (B, C) si $\sigma \neq 0$, rien sinon.
   (b) Engendrer aléatoirement des matrices X et Y et après essais, émettre une conjecture sur le lien entre le rang de A et le rang de C.
3. (a) Montrer que $\ker(A) \subset \ker(C)$.
   (b) Montrer que $X \in \ker(C)$ et en déduire que $\text{rang}(C) \leq r - 1$.
   (c) Déterminer $\text{rang}(B)$. En déduire que $\text{rang}(C) = r - 1$.
   (d) Montrer que tout matrice $A \in \mathcal{M}_n$ de rang r peut s'écrire comme somme de r matrices de rang 1.
4. Écrire une fonction Python d'argument une matrice numpy `A`$= A$ carrée et renvoyant, en liste, une décomposition de A telle que celle exhibée

Éléments de correction–Planche 6

Exercice avec préparation

1. Étude d'un exemple. Soit $A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$, $X = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ et $Y = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$.

   (a) On peut déterminer ce rang avec Python

```
A = np.array([[1,1,0,0],
              [1,0,1,0],
              [0,1,0,1],
              [0,0,1,1]])
X = np.array([[1],[1],[0],[0]]); Y = np.array([[1,0,0,0]])

r = np.linalg.matrix_rank(A)
```

   Manuellement, on détermine le rang de A par la méthode du pivot de Gauss. Les matrices suivantes ont toutes même rang :

$$rg(A) = rg \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} (L_2 \leftarrow L_2 - L_1) = rg \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} (L_3 \leftarrow L_3 + L_2)$$
$$= rg \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} (L_4 \leftarrow L_4 - L_3)$$

   Le rang de cette dernière matrice échelonnée, et donc celui de A, est 3.

   (b) On a (calcul machine)

$$\sigma = 2,\ B = \frac{1}{2} \begin{pmatrix} 2 & 2 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},\ C = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & -1 & 2 & 0 \\ -1 & 1 & 0 & 2 \\ 0 & 0 & 2 & 2 \end{pmatrix}$$

   et le rang de C est 2 (ça se fait en machine ou à la main en voyant que la somme des deux lignes centrales vaut la dernière).

2. Étude informatique.

   (a)

```
def BC(A,X,Y,tol=1e-7):
    """
    retourne B et C du texte si sigma \not= 0
    tol pour le test à zero d'un float
    """
    r = np.linalg.matrix_rank(A)
    sigma = np.dot(Y,np.dot(A,X))[0,0]
```

```
        if np.abs(sigma)> tol:
            B = 1/sigma*np.dot(A,np.dot(X,np.dot(Y,A)))
            C = A - B
            s = np.linalg.matrix_rank(C)
            print("rang de A",r,"rang de C",s,"diff=", r-s)
            return (B,C)
        #Retourne 'None' si on ne met pas de return
    #Test sur matrice de l'intro
    print(A,X,Y,"BC",BC(A,X,Y))
```

(b)
```
def testAlea(n=10):
    A = np.random.rand(n,n)
    X = np.random.rand(n,1)
    Y = np.random.rand(1,n)
    BC(A,X,Y)
#On fait tourner 10 fois pour des valeurs de n allant de 3 à 10
for n in range(3,11):
    for _ in range(10):
        testAlea(n=n)
```

Après tous ces essais (qui ont des faiblesses, on ne fabrique jamais de matrice de rang inférieur à la taille) on a l'idée que $rg(C) = rg(A) - 1$.

3. (a) Soit $Z \in \ker(A)$ c'est-à-dire $AZ = 0$. On a alors

$$CZ = (A - \frac{1}{\sigma}AXYA)Z = (I - \frac{1}{\sigma}AXY)AZ = 0$$

et donc $Z \in \ker(C)$. On a donc $\ker(A) \subset \ker(C)$.

(b) On a

$$CX = (A - \frac{1}{\sigma}AXYA)X = AX(1 - \frac{1}{\sigma}YAX) = AX(1-1) = 0$$

et donc $X \in \ker(C)$.

Comme par hypothèse ($\sigma \neq 0$), $X \notin \ker(A)$ et l'inclusion de $\ker(A) \subset \ker(C)$ est stricte. Cela se traduit par un inégalité stricte des dimensions $\dim\ker(C) > \dim\ker(A)$ ou encore via le théorème du rang

$$n - rg(A) < n - rg(C).$$

ou encore, du fait que ce sont des entiers, $rg(C) \leq r - 1$.

(c) La matrice XY est de rang 1 (son espace image est contenu dans $\text{Vect}(X)$) et $XYY^t \neq 0$ car YY^t est réel non nul et X est non nul.

Ainsi par rang de produit de matrices, le rang de B est inférieur à 1 (vu autrement, son espace image est contenu dans $\text{Vect}(AX)$).

Le rang de B est 1 car $BX = AX \neq 0$ (calcul déjà fait).

Comme $A = C + B$, on voit qu'en prenant une base de l'espace image de C, une base de l'espace image de B, en les juxtaposant, on obtient une famille génératrice de l'espace image de A et donc contenant au moins r vecteurs.

Cela démontre à la volée l'inégalité $rg(A) \leq rg(B) + rg(C)$ et, appliquée à notre cas, montre que $rg(C) \geq r - 1$, ce qui au final, garantit l'égalité $rg(C) = r - 1$.

(d) Montrons par récurrence sur $r \in \mathbb{N}$, la proposition : **toute** matrice A de rang r s'écrit comme somme de r matrices de rang 1.

— **Initialisation.** Cet énoncé est logiquement vrai si la matrice A est nulle (rang 0) (la somme sur le vide vaut 0) et il est trivialement vrai si A est de rang 1 : $A = \sum_{i=1}^{1} A_i$ où $A_1 = A$ est de rang 1.

— **Hérédité** : supposons que cet énoncé soit vrai pour un certain entier naturel r. Soit A une matrice $(n \times n)$ de rang $r+1$.

— On peut noter que si $r+1 > n$, il n'existe pas de telle matrice, ce qui n'est pas un problème car alors la proposition voulue, commençant par **toute** est automatiquement logiquement vraie.

— Étant de rang $r+1$ où $1 \leq r+1 \leq n$, son noyau est de dimension $< n$ et il existe un vecteur X de $\mathbb{R}^n$ n'appartenant pas au noyau de A et en prenant $Y = (AX)^t$, la matrice-nombre 1×1 définie par $\sigma = YAX$, qui vaut la norme au carré de AX, est non nulle.

En effectuant la construction du texte, la matrice $A' = C$ est de rang $r+1-1 = r$ et donc, par hypothèse de récurrence, s'écrit comme somme de r matrices de rang 1, $A_1, \ldots, A_r$. En posant $A_{r+1} = B$, matrice de rang 1, on a alors la décomposition de A en somme de $r+1$ matrices de rang 1.

$$A = \sum_{i=1}^{r} A_i + A_{r+1} = \sum_{i=1}^{r+1} A_i$$

4.
```
def decomposition(A,tol = 1e-7):
    A = A.copy() #On travaille sur une copie
    n = A.shape[0] #retrouver le nbre de lignes
    r = np.linalg.matrix_rank(A)
    As = []
    print("A",A,"rang de A",r)
    for i in range(r):
        while True : #Trouver un vecteur pas ds le noyau
            X = np.random.rand(n,1)
            AX = np.dot(A,X)
            if np.sum(np.abs(AX)) > tol:
                print("X")
                break
        Y = AX.T #On prend la transposée
        (B,C) = BC(A,X,Y)
        As.append(B)
        A = C
    return As
#Test
As = decomposition(A)
print("decomposition",As,"\n rangs=",
      [np.linalg.matrix_rank(a) for a in As],
      "\n diff A et somme decomp. =",A-sum(As))
```

On utilise la récurrence précédente pour écrire la fonction voulue. la difficulté est de trouver un vecteur qui n'est pas dans le noyau des matrices considérées. Une méthode pratique est simplement de tirer au sort n vecteur, il tombera (presque) toujours à coté du noyau...