

Sujet TP2

Nous travaillerons encore dans le repertoire **TP_ANANUM/** avec l'environnement virtuel **.venv** activé:

```
source .venv/bin/activate
python test_config.py
```

- Téléchargez et **décompressez** dans ce dossier l'archive tp2.zip présente sur moodle.

Objectifs du TP:

- Formuler un problème réel sous forme mathématique et le résoudre à l'aide d'un ordinateur,
- Implémenter la décomposition QR d'une matrice rectangulaire A ,
- Résoudre un problème des moindres carrés avec cette décomposition,
- Observer l'instabilité de la procédure de Gram-Schmidt,
- (Bonus) Implémenter la méthode de Householder et montrez sa stabilité.

Création du TP

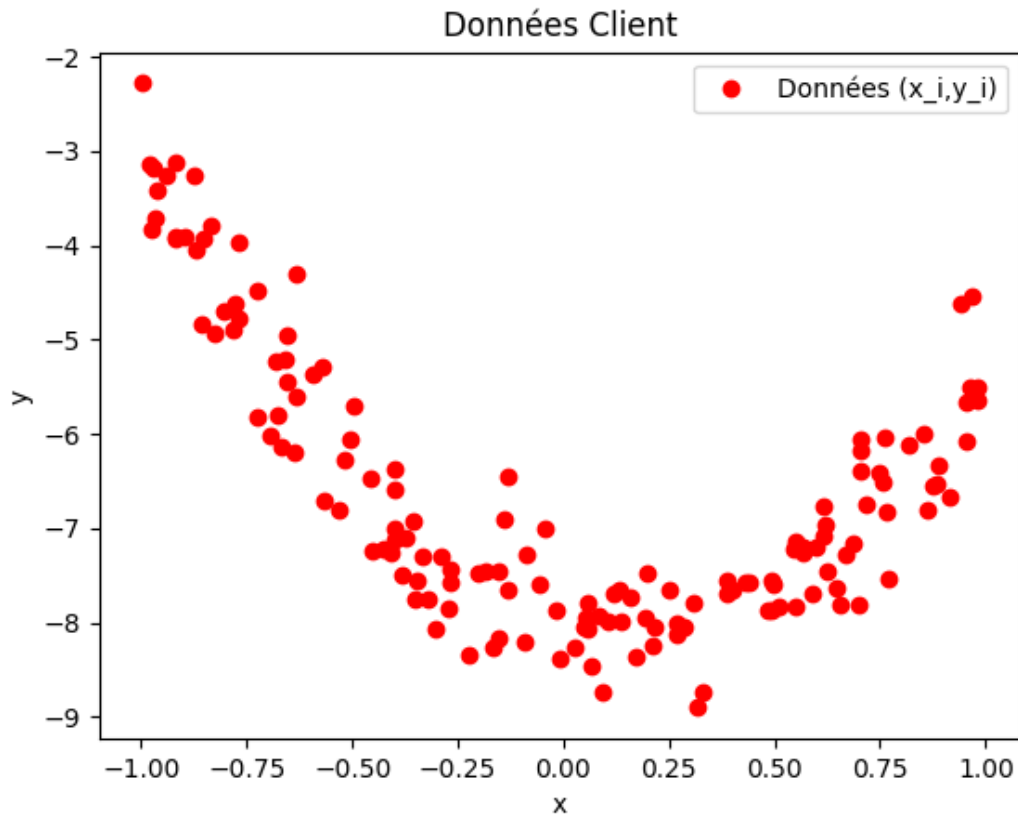
Dans ce TP vous allez chacun générer un jeu de données différents. Pour cela, exécuter la commande suivante:

```
python generate_tp2.py
```

S'il n'ya pas d'erreurs, vous pouvez continuer sinon appelez moi.

Description du problème

Vous travaillez pour l'entreprise **DataCorp**. Un client vient vous voir avec des données de la forme suivante:



Votre client vous demande de trouver le polynôme de degré 2 qui explique le mieux les données.

Modélisation Mathématique

Nous cherchons un triple $(A, B, C) \in \mathbb{R}^3$ tel que le polynôme

$$P(X) = AX^2 + BX + C$$

approxime au mieux les données (x_i, y_i) c'est à dire $P(x_i) \approx y_i$ pour tout $1 \leq i \leq N$.

Les données sont sous la forme suivante:

- Un vecteur de taille N , contenant x_i ;
- Un vecteur de taille N , contenant y_i ;
- Notons $d = 2$ le degré du polynôme P ;

Formuler le problème comme un problème de minimisation des moindres carrés, i.e.

$$\text{Trouver } x \text{ réalisant le } \min_{x \in \mathbb{R}^{d+1}} \|Ax - b\|_2^2.$$

Avec, A une matrice de taille $(N, d + 1)$, x est un vecteur inconnu de taille $(d + 1)$ et b est un vecteur connu de taille N .

Question: Donnez la matrice A , le vecteur x et le vecteur b . *Bonus:* comment s'appelle une matrice de la forme de A ?

Investigation Mathématique

La matrice A admet la décomposition suivante:

$$A = QR$$

où Q est une matrice orthogonale ($Q^t Q = Id_{d+1}$) de taille $(N, d+1)$ et R est une matrice triangulaire supérieure de taille $(d+1, d+1)$.

Nous avons vu au dernier TD que Q et R se calculent de la façon suivante:

- Les colonnes de Q résultent du procédé de Gram-Schmidt appliqué aux colonnes de A :

$$\tilde{q}_i = A_i - \sum_{k=1}^{i-1} (A_i, q_k) q_k, \quad \text{et} \quad q_i = \frac{\tilde{q}_i}{\|\tilde{q}_i\|_2}.$$

- Les éléments de R sont calculés par:

$$R_{i,j} = \begin{cases} 0 & \text{si } i > j; \\ \|\tilde{q}_j\|_2 & \text{si } i = j; \\ (A_i, q_j) & \text{si } i < j. \end{cases}$$

Enfin, nous avons montré que le problème de chercher le x réalisant le $\min_x \|Ax - b\|_2^2$ est le même problème que de chercher le x réalisant le

$$\min_x \|Rx - Q^t b\|_2^2.$$

Dont la solution est donnée par $x = R^{-1} Q^t b$.

Implémentation - 1

Dans le fichier *main_tp2.py* complétez la fonction suivante qui aux données (x_i, y_i) construit la matrice A et le vecteur du second membre b :

```
'''
Construct matrix A and right hand side vector b
'''
def construct_Ab(N, d, x, y):
    A = np.zeros(shape=(N, d+1), dtype=np.double)
    b = np.zeros(shape=(N), dtype=np.double)

    # Compléter ici

    return A, b
```

Implémenter la fonction *QR_GS(A)* qui calcule la décomposition QR de A avec la procédure de Gram-Schmidt.

```
'''
Compute QR decomposition of a matrix A of size Nx x Ny (Nx > Ny) with Gram-
Schmidt algorithm
Q is orthogonal of size Nx x Ny
R is upper triangular of size Ny x Ny
'''
def QR_GS(A):
    nx, ny = A.shape
    R = np.zeros(shape=(ny, ny), dtype=np.double)
    Q = np.zeros(shape=(nx, ny), dtype=np.double)

    # Compléter ici
```

```

    print("La matrice Q donnée par la procédure de G-S est orthogonale à
    ||Id_{d+1} - Q^t * Q|| = " + str(np.linalg.norm(np.eye(ny) -
    np.matmul(np.transpose(Q), Q))) + " près.")
    return Q, R

```

Implémentez la fonction $\text{solve_lsq}(A, b)$ qui retourne le vecteur x réalisant le $\min_x \|Ax - b\|_2^2$.

```

'''
Solve least square problem min_x ||Ax - b||^2 via QR décomposition of A
'''
def solve_lsq(A, b):
    nx, ny = A.shape
    sol = np.zeros(shape=(ny), dtype=np.double)

    # Compléter ici

    return sol

```

Quelques commandes numpy

```

np.transpose(A)    # transposée de la matrice A: A^t
np.matmul(A, B)    # produit de matrices A et B
np.matmul(A, v)    # produit matrice vecteur
np.outer(u, v)     # produit u * v^t
np.dot(u, v)       # produit scalaire de u et v : u^t * v
np.linalg.norm(u)  # norm du vecteur u : ||u||
A[:, j]            # j-ème coloone de la matrice A /\ en python les indices
commencent à 0

```

Implémentation - 2

Modifier votre programme pour chercher la meilleure approximation polynomiale de degré d et complétez le tableau suivant:

Degré de $P(X)$	Variance Expliquée	Orthogonalité de Q
0		
1		
2		
10		
50		
100		
120		
140		

Commentez:

Implémentation - 3

Nous avons vu en CM que la procédure de Gram-Schmidt est instable, on se propose de modifier l'algorithme pour utiliser la méthode de Householder. Implémenter une fonction *householder(u)* qui construit la matrice de Householder $H(u)$ associée à un vecteur $u \in \mathbb{R}^n$:

$$H(u) = Id_n - 2 \frac{u \cdot u^t}{\|u\|_2^2}$$

Commentaire mathématique

Nous voulons obtenir la décomposition QR de A via la méthode de Householder. A la différence du cas précédent, nous allons obtenir une décomposition QR de la forme:

- Q une matrice orthogonale de taille (N, N) ;
- R une matrice triangulaire supérieure de taille $(N, d+1)$;
- /!\ On supposera toujours que $N \geq d+1$ c'est à dire que nous avons plus de données que le degré du polynôme qui interpole.

C'est à dire que nous obtiendrons:

$$A = Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$$

où R_1 est triangulaire supérieure de taille $(d+1, d+1)$. Pour obtenir la décomposition, nous procéderons avec la méthode de Householder:

$$\begin{aligned} A^{(1)} &= A \\ A^{(2)} &= H^{(1)} A^{(1)} \\ &\vdots \\ A^{(k+1)} &= H^{(k)} A^{(k)} \end{aligned}$$

où,

$$H^{(k)} = H \left(\tilde{A}_k^{(k)} + \|\tilde{A}_k^{(k)}\| e_k \right), \quad \tilde{A}_k^{(k)} = \left(0, \dots, 0, a_{kk}^{(k)}, \dots, a_{nk}^{(k)} \right)^t.$$

Observez que $\tilde{A}_k^{(k)}$ est la k -ème colonne de $A^{(k)}$ à la quelle nous avons mis des zéros au dessus du coefficient $a_{kk}^{(k)}$. En itérant ce processus jusqu'à la n -ème étape nous obtiendrons:

$$A^{(n)} = R = H^{(n-1)} H^{(n-2)} \dots H^{(2)} H^{(1)} A.$$

Ainsi nous aurons,

$$Q = H^{(1)} \cdot H^{(2)} \cdot \dots \cdot H^{(n-1)} \quad \text{et} \quad R = Q^t A.$$

Implémentez une fonction *QR_Householder(A)* qui retourne la décomposition QR de A avec la méthode de Householder. Effectuez le changement nécessaire dans la fonction *solve_ls(A,b)* pour faire appel à cette nouvelle méthode de décomposition. Complétez et commentez les résultats obtenus par la méthode de Householder.

Degré de $P(X)$	Variance Expliquée	Orthogonalité de Q
0		
1		
2		
10		
50		
100		
120		
140		

Commentez: