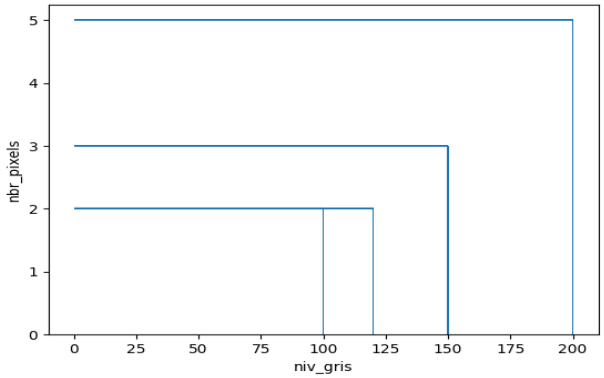
1- Histogramme

Niveau de gris	**100**	**120**	**150**	**200**
Nombre de pixels	2	2	3	5



Niveau de gris	**100**	**120**	**150**	**200**
Nombre de pixels	2	2	3	5
P(i)	0.167	0.167	0.25	0.416

Pour k=100

Classe C1 (<=100)

P1(100) = 0.167

M1(100) = (100 * 0.167) /0.167 = 100

Classe C2 (>100)

P2(100) = 1 - 0.167 = 0.833

M2(100) = (120 * 0.167 + 150 * 0.25 + 200 * 0.416) / 0.833 = 175

$[\sigma B(100)]^2$ = 0.167 * 0.833 * (100 - 175) ^{2} = 863.3

Pour k=120

Classe C1 (<=120)

P1(120) = 0.167 + 0.167 = 0.334

M1(120) = (100 * 0.167 + 120 * 0.167) /0.334 = 110

Classe C2 (>120)

P2(120) = 1 - 0.334 = 0.666

M2(120) = (150 * 0.25 + 200 * 0.416) / 0.666 = 184

$[\sigma_B(120)]^2 = 0.334 * 0.666 * (110 - 184)^2 = 1110.7$

Suivant le même processus on obtient

$[\sigma_B(150)]^2 = 1255.5$

Le seuil optimal et k = 150

$$I\_seg = \begin{vmatrix} 0 & 0 & 0 \\ 255 & 255 & 255 \\ 255 & 255 & 255 \end{vmatrix}$$

2-

```python
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread('noisy2.png', cv.IMREAD_GRAYSCALE)

#================== Seuil global ===========
seul_1, th1 = cv.threshold(img, 127, 255, cv.THRESH_BINARY)
print('seul_1 =  ', seul_1)
#================== Seuil otsu ===========
seul_2, th2 = cv.threshold(img, 0, 255, cv.THRESH_BINARY + cv.THRESH_OTSU)
print('seul_2 =  ', seul_2)

#================== Seuil apres filtrage gaussian ===========
blur = cv.GaussianBlur(img, (5, 5), 0)
seul_3, th3 = cv.threshold(blur, 0, 255, cv.THRESH_BINARY + cv.THRESH_OTSU)
print('seul_3 =  ', seul_3)
# plot all the images and their histograms
images = [img, 0, th1, img, 0, th2, blur, 0, th3]
titles = ['Originale image avec bruit', 'Histogram', 'Global seuil (v=127)',
          'Originaleimage avec bruit', 'Histogram', "seuil avec otsu",
          'filter Gaussian ', 'Histogram', "seuil avec otsu"]

for i in range(3):
    plt.subplot(3, 3, i * 3 + 1), plt.imshow(images[i * 3], 'gray')
    plt.title(titles[i * 3]), plt.xticks([]), plt.yticks([])
    plt.subplot(3, 3, i * 3 + 2), plt.hist(images[i * 3].ravel(), 256)
    plt.title(titles[i * 3 + 1]), plt.xticks([]), plt.yticks([])
    plt.subplot(3, 3, i * 3 + 3), plt.imshow(images[i * 3 + 2], 'gray')
    plt.title(titles[i * 3 + 2]), plt.xticks([]), plt.yticks([])
plt.show()
```

3- Étapes principales de k-means

1. **Initialisation des centres de clusters** : On choisit k=2k = 2k=2 centres initiaux aléatoirement dans l'ensemble des points ou de façon arbitraire.
2. **Assignation des points aux clusters** : Chaque point est attribué au centre de cluster le plus proche (selon la distance euclidienne).
3. **Mise à jour des centres de clusters** : Le nouveau centre d'un cluster est la moyenne (barycentre) des points assignés à ce cluster.
4. **Convergence** : Répéter les étapes 2 et 3 jusqu'à ce que les clusters ne changent plus.

Étape 1 : Initialisation des centres

Choisissons deux centres initiaux aléatoires parmi les points :

- $C_1 = (1, 4)$
- $C_2 = (7, 1)$

Étape 2 : Assignation des points aux clusters

Calculons la distance euclidienne de chaque point aux centres C_1 et C_2 :

$$\text{Distance euclidienne} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Point	$d(C_1)$	$d(C_2)$	Cluster assigné
$(1, 4)$	$\sqrt{(1-1)^2 + (4-4)^2} = 0$	$\sqrt{(1-7)^2 + (4-1)^2} = 6.71$	C_1
$(5, 1)$	$\sqrt{(5-1)^2 + (1-4)^2} = 5.00$	$\sqrt{(5-7)^2 + (1-1)^2} = 2.00$	C_2
$(6, 2)$	$\sqrt{(6-1)^2 + (2-4)^2} = 5.39$	$\sqrt{(6-7)^2 + (2-1)^2} = 1.41$	C_2
$(7, 1)$	$\sqrt{(7-1)^2 + (1-4)^2} = 6.71$	$\sqrt{(7-7)^2 + (1-1)^2} = 0$	C_2
$(2, 5)$	$\sqrt{(2-1)^2 + (5-4)^2} = 1.41$	$\sqrt{(2-7)^2 + (5-1)^2} = 6.40$	C_1

Clusters après assignation :

- C_1 : $(1, 4), (2, 5)$
- C_2 : $(5, 1), (6, 2), (7, 1)$

Étape 3 : Mise à jour des centres

Calculons le nouveau centre de chaque cluster (barycentre) :

- Nouveau C_1 : Moyenne de $(1, 4)$ et $(2, 5)$

$$C_1 = \left(\frac{1+2}{2}, \frac{4+5}{2} \right) = (1.5, 4.5)$$

- Nouveau C_2 : Moyenne de $(5, 1), (6, 2), (7, 1)$

$$C_2 = \left(\frac{5+6+7}{3}, \frac{1+2+1}{3} \right) = (6, 1.33)$$

Étape 4 : Répéter (Assignation des points avec les nouveaux centres)

Recalculons les distances avec les nouveaux centres $(1.5, 4.5)$ et $(6, 1.33)$:

Point	$d(C_1)$	$d(C_2)$	Cluster assigné
$(1, 4)$	$\sqrt{(1-1.5)^2 + (4-4.5)^2} = 0.71$	$\sqrt{(1-6)^2 + (4-1.33)^2} = 5.84$	C_1
$(5, 1)$	$\sqrt{(5-1.5)^2 + (1-4.5)^2} = 4.95$	$\sqrt{(5-6)^2 + (1-1.33)^2} = 1.05$	C_2
$(6, 2)$	$\sqrt{(6-1.5)^2 + (2-4.5)^2} = 5.22$	$\sqrt{(6-6)^2 + (2-1.33)^2} = 0.67$	C_2
$(7, 1)$	$\sqrt{(7-1.5)^2 + (1-4.5)^2} = 6.40$	$\sqrt{(7-6)^2 + (1-1.33)^2} = 1.05$	C_2
$(2, 5)$	$\sqrt{(2-1.5)^2 + (5-4.5)^2} = 0.71$	$\sqrt{(2-6)^2 + (5-1.33)^2} = 5.84$	C_1

Les clusters restent inchangés après cette itération, donc **l'algorithme a convergé**.

---

3 Résultat final

- **Cluster 1 (C1)** : $(1, 4), (2, 5)$, avec centre $C_1 = (1.5, 4.5)$.
- **Cluster 2 (C2)** : $(5, 1), (6, 2), (7, 1)$, avec centre $C_2 = (6, 1.33)$.

4-

```
image = np.array([[0, 1, 2, 1, 0, 0],
                  [0, 3, 4, 4, 1, 0],
                  [2, 4, 5, 5, 4, 0],
                  [1, 4, 5, 5, 4, 1],
                  [0, 3, 4, 4, 3, 1],
                  [0, 2, 3, 3, 2, 0],], dtype=np.uint8)


from sklearn.cluster import KMeans

# Créons une matrice 10x10 avec 3 niveaux de gris
np.random.seed(42)
#matrix = np.random.choice([50, 150, 250], size=(10, 10))
matrix = image
print(matrix)
# Affichage de la matrice d'origine
plt.figure(figsize=(8, 4))
plt.subplot(1, 2, 1)
plt.imshow(matrix, cmap='gray')
plt.title("Matrice d'origine (niveaux de gris)")
plt.colorbar()

# 2. Préparation des données pour K-moyennes
# La matrice est convertie en un tableau de points pour l'algorithme
pixels = matrix.reshape(-1, 1)  # Chaque pixel est une intensité unique
print(pixels)
# 3. Application de l'algorithme des K-moyennes
K = 6  # Nombre de segments
kmeans = KMeans(n_clusters=K, random_state=42)
kmeans.fit(pixels)

# Les étiquettes (clusters) assignées par l'algorithme
labels = kmeans.labels_

# 4. Reconstruction de la matrice segmentée
# Remplace chaque pixel par l'intensité moyenne de son cluster
segmented_matrix = kmeans.cluster_centers_[labels].reshape(matrix.shape)

# 5. Affichage de la matrice segmentée
plt.subplot(1, 2, 2)
plt.imshow(segmented_matrix, cmap='gray')
plt.title(f"Matrice segmentée (K={K})")
plt.colorbar()

plt.tight_layout()
plt.show()
```