**Kmean Clustering Algorithm:**

**Import Libraries (Nhập thư viện)**:

import numpy as np

import matplotlib.pyplot as plt

import cv2

* **numpy** (as **np**): Thư viện Python cho tính toán khoa học, cung cấp các cấu trúc dữ liệu và công cụ cho việc làm việc với mảng đa chiều.
* **matplotlib.pyplot** (as **plt**): Thư viện Python cho việc vẽ đồ thị và trực quan hóa dữ liệu.
* **cv2**: Thư viện OpenCV (Open Source Computer Vision Library) được sử dụng để xử lý hình ảnh.

**Define K-means Function (Định nghĩa hàm K-means)**:

def kmeans(X, K, max\_iters=10):

centroids = X[np.random.choice(range(len(X)), K, replace=False)]

for i in range(max\_iters):

# Assign each data point to its closest centroid

distances = np.linalg.norm(X[:, np.newaxis, :] - centroids, axis=-1)

labels = np.argmin(distances, axis=-1)

# Update the centroids based on the assigned data points

for j in range(K):

centroids[j] = np.mean(X[labels == j], axis=0)

return labels, centroids

* **kmeans**: Hàm thực hiện thuật toán phân cụm K-means. Nhận ba tham số: **X** là tập dữ liệu đầu vào, **K** là số cụm mong muốn, **max\_iters** là số lần lặp tối đa (mặc định là 10).

Hàm **kmeans** thực hiện thuật toán K-means để phân cụm dữ liệu. Dưới đây là mô tả chi tiết về cách hàm này hoạt động:

1. **Khởi tạo các centroids ban đầu**:
   * Dòng code: **centroids = X[np.random.choice(range(len(X)), K, replace=False)]**
   * Các centroids ban đầu được chọn một cách ngẫu nhiên từ tập dữ liệu *X*. Chúng được chọn bằng cách lấy ngẫu nhiên *K* mẫu từ tập dữ liệu mà không lặp lại.
2. **Lặp lại cho đến khi hội tụ hoặc đạt đến số lần lặp tối đa**:
   * Vòng lặp: **for i in range(max\_iters):**
   * Thuật toán sẽ tiếp tục lặp lại các bước sau cho đến khi một trong các điều kiện sau được đáp ứng:
     + Hội tụ: Khi sự thay đổi giữa các centroids giữa hai lần lặp liên tiếp là không đủ lớn.
     + Đạt đến số lần lặp tối đa: Khi số lần lặp đạt đến giá trị được chỉ định bởi **max\_iters**.
3. **Gán mỗi điểm dữ liệu cho centroid gần nhất**:
   * Dòng code: **distances = np.linalg.norm(X[:, np.newaxis, :] - centroids, axis=-1)**
   * Dòng code này tính toán khoảng cách từ mỗi điểm dữ liệu đến tất cả các centroids.
   * **np.linalg.norm**: Tính toán norm của mỗi điểm dữ liệu đến các centroids.
   * **axis=-1** chỉ định chiều cuối cùng của mảng, tức là tính norm theo chiều của các features của mỗi điểm dữ liệu.
   * Dòng code tiếp theo sử dụng **np.argmin** để gán mỗi điểm dữ liệu cho centroid gần nhất, dựa trên khoảng cách tính được.
4. **Cập nhật lại các centroids**:
   * Dòng code: **centroids[j] = np.mean(X[labels == j], axis=0)**
   * Vòng lặp trong dòng code này lặp qua từng centroid.
   * Dòng code bên trong tính toán trung bình của các điểm dữ liệu đã được gán cho centroid đó và cập nhật lại vị trí của centroid.
5. **Trả về nhãn cho mỗi điểm dữ liệu và các centroids cuối cùng**:
   * Hàm trả về **labels**, một mảng chứa nhãn cho mỗi điểm dữ liệu (chỉ số của centroid gần nhất) và **centroids**, mảng chứa các centroids cuối cùng sau khi thuật toán hội tụ hoặc đã đạt đến số lần lặp tối đa.

**Read the Original Image (Đọc ảnh gốc)**:

original\_img = cv2.imread("D:\Python\_CV\Segment and detection\kodim02.png")

original\_img = cv2.cvtColor(original\_img, cv2.COLOR\_BGR2RGB)

* **cv2.imread**: Đọc ảnh từ đường dẫn cụ thể.
* **cv2.cvtColor**: Chuyển đổi không gian màu từ BGR (Blue-Green-Red) sang RGB (Red-Green-Blue), vì OpenCV đọc ảnh dưới dạng BGR còn matplotlib sử dụng RGB.

**Convert to HSV Color Space (Chuyển đổi sang không gian màu HSV)**:

HSV\_img = cv2.cvtColor(original\_img, cv2.COLOR\_BGR2HSV)

* Chuyển đổi ảnh gốc từ không gian màu RGB sang không gian màu HSV (Hue-Saturation-Value), thường được sử dụng trong việc xử lý hình ảnh và phân đoạn.

**Vectorize the Image (Vector hóa ảnh)**:

vectorized = HSV\_img.reshape((-1, 3))

vectorized = np.float32(vectorized)

* **reshape**: Chuyển đổi ma trận hình ảnh 3D thành một vector 2D để phù hợp với đầu vào của thuật toán K-means.
* **np.float32**: Chuyển đổi kiểu dữ liệu của ma trận thành **float32** để tính toán chính xác hơn.

**Run K-means Algorithm (Chạy thuật toán K-means)**:

K = 15

labels, centroids = kmeans(vectorized, K)

* Chạy thuật toán K-means trên dữ liệu đã vector hóa với số lượng cụm **K** là 15.

**Assign Labels and Create Result Image (Gán nhãn và tạo ảnh kết quả)**:

centroids = np.uint8(centroids)

res = centroids[labels.flatten()]

result\_image = res.reshape((HSV\_img.shape))

result\_image = cv2.cvtColor(result\_image, cv2.COLOR\_HSV2BGR)

* **np.uint8**: Chuyển đổi các giá trị của centroids sang kiểu dữ liệu **uint8**.
* **reshape**: Chuyển đổi vector kết quả thành ma trận 3D cùng kích thước với ảnh gốc.
* **cv2.cvtColor**: Chuyển đổi ảnh kết quả từ không gian màu HSV trở lại BGR để hiển thị bằng Matplotlib.

**Display Original and Segmented Images (Hiển thị ảnh gốc và ảnh phân vùng)**:

figure\_size = 15

plt.figure(figsize=(figure\_size, figure\_size))

plt.subplot(1, 2, 1), plt.imshow(original\_img)

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(1, 2, 2), plt.imshow(result\_image)

plt.title('Segmented Image when K = %i' % K), plt.xticks([]), plt.yticks([])

plt.show()

* Tạo một plot với hai hình ảnh cạnh nhau: ảnh gốc và ảnh phân vùng.
* Hiển thị tiêu đề cho từng hình ảnh và ẩn các trục.

Full source code Kmean Clustering:

import numpy as np

import matplotlib.pyplot as plt

import cv2

def kmeans(X, K, max\_iters=10):

    centroids = X[np.random.choice(range(len(X)), K, replace=False)]

    for i in range(max\_iters):

     # Assign each data point to its closest centroid

     distances = np.linalg.norm(X[:, np.newaxis, :] - centroids, axis=-1)

     labels = np.argmin(distances, axis=-1)

     # Update the centroids based on the assigned data points

     for j in range(K):

      centroids[j] = np.mean(X[labels == j], axis=0)

    return labels, centroids

# Đọc ảnh màu

original\_img = cv2.imread("D:\Python\_CV\Segment and detection\kodim02.png")

original\_img = cv2.cvtColor(original\_img, cv2.COLOR\_BGR2RGB)

# Chuyen sang khong gian mau HSV

HSV\_img = cv2.cvtColor(original\_img, cv2.COLOR\_BGR2HSV)

# Vector hóa ảnh

vectorized = HSV\_img.reshape((-1, 3))

vectorized = np.float32(vectorized)

# Chạy K-means với K=10

K = 10

labels, centroids = kmeans(vectorized, K)

# Gán nhãn cho từng pixel và lấy giá trị trung bình của mỗi cluster để tạo ảnh kết quả

centroids = np.uint8(centroids)

res = centroids[labels.flatten()]

result\_image = res.reshape((HSV\_img.shape))

result\_image = cv2.cvtColor(result\_image, cv2.COLOR\_HSV2BGR)

# Hiển thị ảnh gốc và ảnh phân vùng

figure\_size = 15

plt.figure(figsize=(figure\_size, figure\_size))

plt.subplot(1, 2, 1), plt.imshow(original\_img)

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(1, 2, 2), plt.imshow(result\_image)

plt.title('Segmented Image when K = %i' % K), plt.xticks([]), plt.yticks([])

plt.show()

**Ảnh kết quả đầu ra:**

A close-up of a door

Description automatically generated