practical-4-mids

April 1, 2025

[1]:

**import numpy as np**

**from scipy.spatial.distance import** euclidean

[2]:

[3]:

*# Given points*

points = np.array([ [0.1, 0.6],

[0.15, 0.71],

[0.08, 0.9],

[0.16, 0.85],

[0.2, 0.3],

**def** k\_means\_clustering(points, initial\_centroids, max\_iterations=100, tol=1e-4): centroids = initial\_centroids.copy()

clusters = {0: [], 1: []}

**for** \_ **in** range(max\_iterations): new\_clusters = {0: [], 1: []}

*# Assign points to the nearest cluster*

**for** point **in** points:

distances = [euclidean(point, centroids[i]) **for** i **in** range(2)] cluster\_idx = np.argmin(distances) new\_clusters[cluster\_idx].append(point)

*# Compute new centroids*

new\_centroids = [np.mean(new\_clusters[i], axis=0) **if** new\_clusters[i]␣

𝗌**else** centroids[i] **for** i **in** range(2)]

*# Check for convergence*

**if** all(euclidean(centroids[i], new\_centroids[i]) < tol **for** i **in**␣

𝗌range(2)):

**break**

centroids = new\_centroids clusters = new\_clusters

**return** clusters, centroids

[0.25, 0.5],

[0.24, 0.1],

[0.3, 0.2]

])

[4]:

*# Initial centroids*

initial\_centroids = [points[0], points[7]]

[5]:

*# Run k-means*

clusters, final\_centroids = k\_means\_clustering(points, initial\_centroids)

[6]:

*# Output results*

print("Final Clusters:")

**for** i, cluster **in** clusters.items(): print(f"Cluster **{**i+1**}**: **{**cluster**}**")

print("**\n**Final Centroids:")

**for** i, centroid **in** enumerate(final\_centroids): print(f"Centroid **{**i+1**}**: **{**centroid**}**")

[7]:

Final Clusters:

Cluster 1: [array([0.1, 0.6]), array([0.15, 0.71]), array([0.08, 0.9 ]),

array([0.16, 0.85]), array([0.25, 0.5 ])]

Cluster 2: [array([0.2, 0.3]), array([0.24, 0.1 ]), array([0.3, 0.2])]

Final Centroids:

Centroid 1: [0.148 0.712]

Centroid 2: [0.24666667 0.2 ]

*# Find which cluster contains P6 (0.25, 0.5)*

P6 = [0.25, 0.5]

**for** i, cluster **in** clusters.items():

**if** any(np.allclose(p, P6) **for** p **in** cluster): print(f"**\n**P6 belongs to Cluster **{**i+1**}**") **break**

P6 belongs to Cluster 1

[8]:

*# Count number of points in Cluster 1*

print(f"**\n**Number of points in Cluster 1: **{**len(clusters[0])**}**") print(f"**\n**Number of points in Cluster 2: **{**len(clusters[1])**}**")

Number of points in Cluster 1: 5 Number of points in Cluster 2: 3

[9]:

*#Count number of points in Cluster around m2 (Cluster 2)*

population\_C2 = len(clusters[1])

print(f"**\n**2] Population of cluster around m2: **{**population\_C2**}**")

1. Population of cluster around m2: 3

[10]:

*# Updated centroids*

print(f"**\n**3] Updated values of centroids: **\n**

𝗌 **{**final\_centroids[1]**}**")

m1: **{**final\_centroids[0]**} \n**

m2:

1. Updated values of centroids: m1: [0.148 0.712]

m2: [0.24666667 0.2 ]

[ ]: