Priyal Makwana
TY 04
B batch 35
23UF17826CM156

Aim : Implementation of Clustering Algorithm (K-means / Agglomerative) using Python

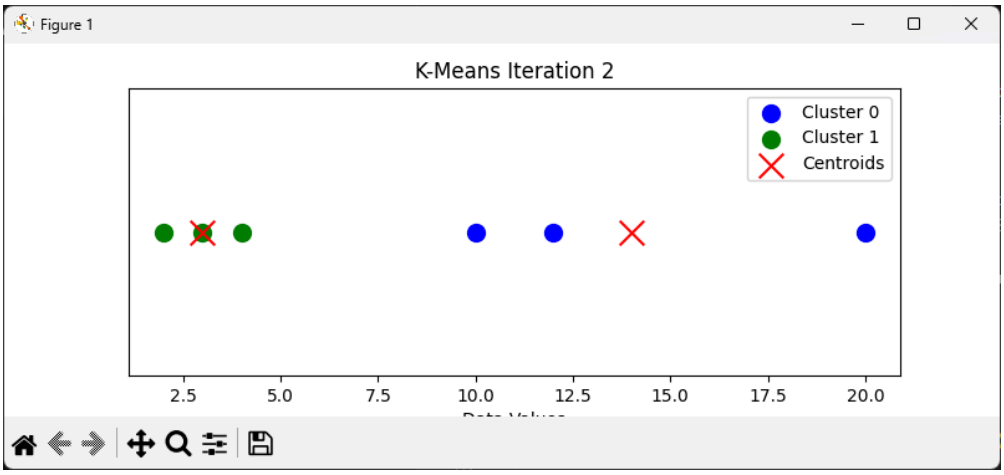
Program Codes :

```
import numpy as np
import matplotlib.pyplot as plt

# Data
data = np.array([2, 3, 4, 10, 20, 12])
k = 2

# Initialize centroids
centroids = np.random.choice(data, size=k, replace=False)
print("Initial centroids:", centroids)

def assign_clusters(data, centroids):
    clusters = {i: [] for i in range(len(centroids))}

    for point in data:
        distances = np.abs(point - centroids)
        cluster_index = np.argmin(distances)
        clusters[cluster_index].append(int(point))

        # CLI print for each point
        print(f"Point {point} -> distances {distances} -> Cluster {cluster_index}")

    return clusters

def update_centroids(clusters):
    new_centroids = []
    for i in clusters:
        mean_val = np.mean(clusters[i])
        new_centroids.append(mean_val)
        print(f"New centroid for Cluster {i}: mean({clusters[i]}) = {mean_val}")
    return np.array(new_centroids)

colors = ['blue', 'green']
```

```
plt.figure(figsize=(8, 3))

for iteration in range(5):
    print("\n" + "="*40)
    print(f"Iteration {iteration + 1}")

    clusters = assign_clusters(data, centroids)

    print("Clusters formed:", clusters)

    plt.clf()

    # Plot points
    for i, points in clusters.items():
        plt.scatter(points, np.zeros(len(points)),
                    color=colors[i], s=100, label=f'Cluster {i}')

    # Plot centroids
    plt.scatter(centroids, np.zeros(len(centroids)),
                color='red', marker='x', s=200, label='Centroids')

    plt.title(f'K-Means Iteration {iteration + 1}')
    plt.yticks([])
    plt.xlabel("Data Values")
    plt.legend()
    plt.pause(1)

    new_centroids = update_centroids(clusters)

    # Convergence check
    if np.allclose(centroids, new_centroids):
        print("Centroids converged. Stopping.")
        break

    centroids = new_centroids
    print("Updated centroids:", centroids)

plt.show()

print("\nFinal Result")
print("Final clusters:", clusters)
print("Final centroids:", centroids)
```

Implementation/Output snap shot :

```
Initial centroids: [ 2 20]
========================================
Iteration 1
Point 2 -> distances [ 0 18] -> Cluster 0
Point 3 -> distances [ 1 17] -> Cluster 0
Point 4 -> distances [ 2 16] -> Cluster 0
Point 10 -> distances [ 8 10] -> Cluster 0
Point 20 -> distances [18  0] -> Cluster 1
Point 12 -> distances [10  8] -> Cluster 1
Clusters formed: {0: [2, 3, 4, 10], 1: [20, 12]}
New centroid for Cluster 0: mean([2, 3, 4, 10]) = 4.75
New centroid for Cluster 1: mean([20, 12]) = 16.0
Updated centroids: [ 4.75 16.  ]
========================================
Iteration 2
Point 2 -> distances [ 2.75 14.  ] -> Cluster 0
Point 3 -> distances [ 1.75 13.  ] -> Cluster 0
Point 4 -> distances [ 0.75 12.  ] -> Cluster 0
Point 10 -> distances [5.25 6.  ] -> Cluster 0
Point 20 -> distances [15.25  4.  ] -> Cluster 1
Point 12 -> distances [7.25 4.  ] -> Cluster 1
Clusters formed: {0: [2, 3, 4, 10], 1: [20, 12]}
New centroid for Cluster 0: mean([2, 3, 4, 10]) = 4.75
New centroid for Cluster 1: mean([20, 12]) = 16.0
Centroids converged. Stopping.

Final Result
Final clusters: {0: [2, 3, 4, 10], 1: [20, 12]}
Final centroids: [ 4.75 16. ]
```



Conclusion :

K-means clustering was implemented to group one-dimensional data into two clusters by iteratively assigning points to the nearest centroid and updating centroids using the mean.
The algorithm converged when the centroids stabilized, resulting in meaningful separation of smaller and larger values.

Review Questions :

1. What is the K-means clustering algorithm & how does it work?

K-means is an unsupervised machine learning algorithm used to group data into K clusters based on similarity.

How it works (steps):

1. Choose K (number of clusters).
2. Initialize K centroids (randomly).
3. Assign each data point to the nearest centroid (usually using distance).
4. Update centroids by calculating the mean of all points in each cluster.
5. Repeat steps 3 & 4 until centroids no longer change (convergence).

Example: Grouping customers based on spending behavior.

2. How do you determine the optimal number of clusters in K-means?

Common methods:

1. Elbow Method

- Plot K vs. WCSS (Within-Cluster Sum of Squares).
- The point where the curve bends like an “elbow” is the optimal K.

2. Silhouette Score

- Measures how well data points fit within their cluster.
- Score ranges from –1 to +1.
- Higher score → better clustering.

3. Gap Statistic

- Compares clustering results with random data.
- The K with the largest gap is chosen.

3. Common distance metrics used in Agglomerative Clustering

Agglomerative clustering is a bottom-up hierarchical clustering method.

Common distance metrics:

1. Euclidean Distance
    - Straight-line distance (most common).
2. Manhattan Distance
    - Distance along grid paths.
3. Cosine Distance
    - Measures angle between vectors (text/data mining).
4. Chebyshev Distance

    - Maximum difference along any dimension.
5. Minkowski Distance
    - Generalized form of Euclidean & Manhattan.

Github Link : - https://github.com/Priyal0410/dwmexp