Machine Learning course

Lab 9 - Clustering

Lab Outline:

➢ k-means Summary

➢ k-means from scratch

➢ Built in k-means

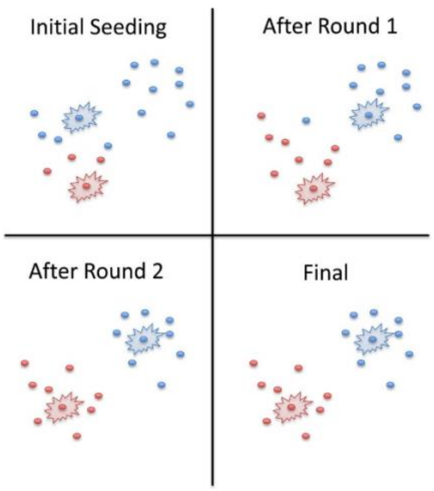
➢ Agglomerative Hierarchical Algorithm Summary

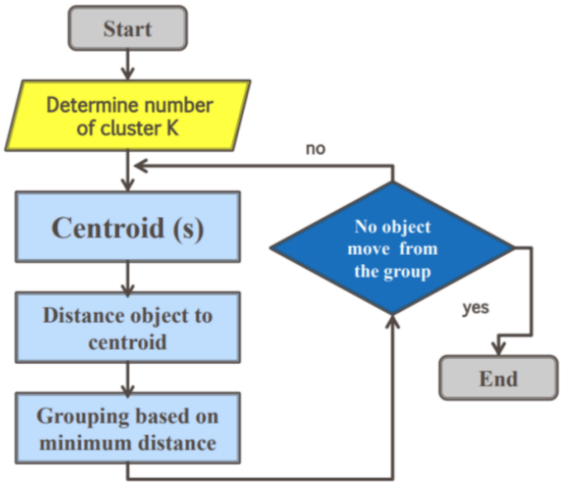
Summary of k-means:

Clustering is like Classification, but the class label of each object is not known. It can be considered the most important unsupervised learning problem; it deals with finding a structure in a collection of unlabeled data.
A cluster is a subset of data that are similar while clustering is the process of grouping the data into classes or clusters so that objects within a cluster have high similarity in comparison to one another but are very dissimilar to objects in other clusters.

K-means Clustering:

K-means clustering is a partitioning algorithm that divides a dataset into K clusters, where each data point belongs to the cluster with the nearest mean. It's an iterative algorithm that aims to minimize the sum of squared distances between data points and their assigned cluster centroids.

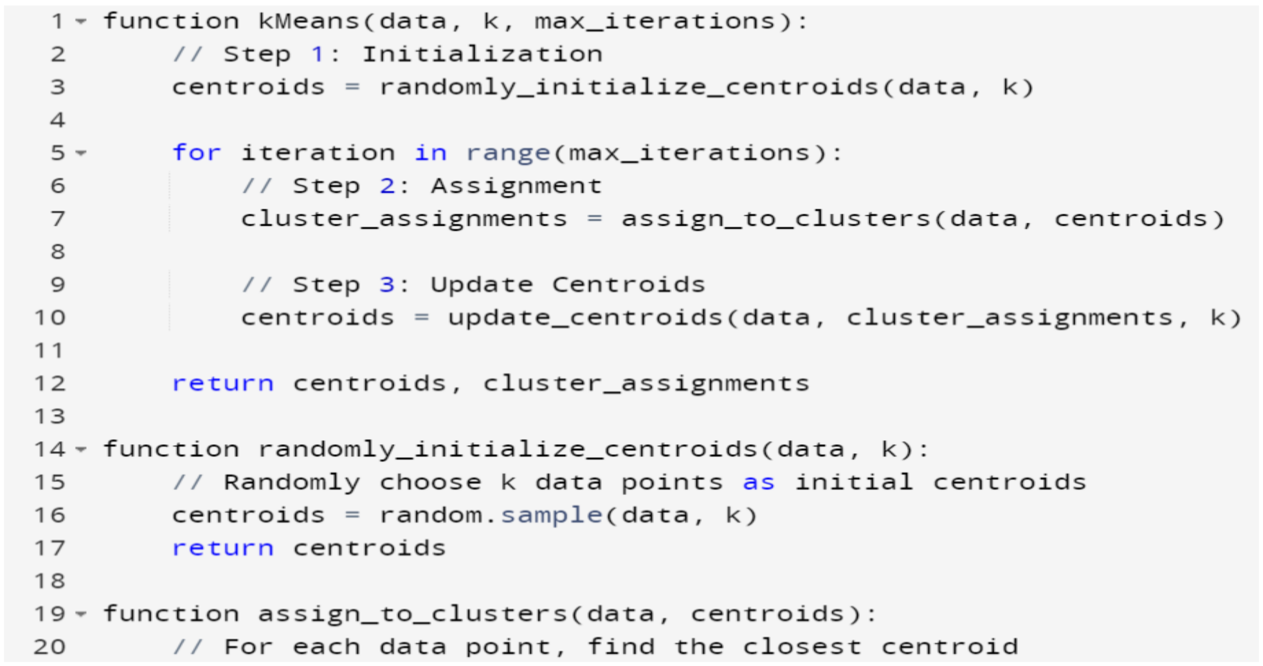



Start
Determine number of cluster K
Centroid (s)
Distance object to centroid
Grouping based on minimum distance
No object move from the group
no
yes
End

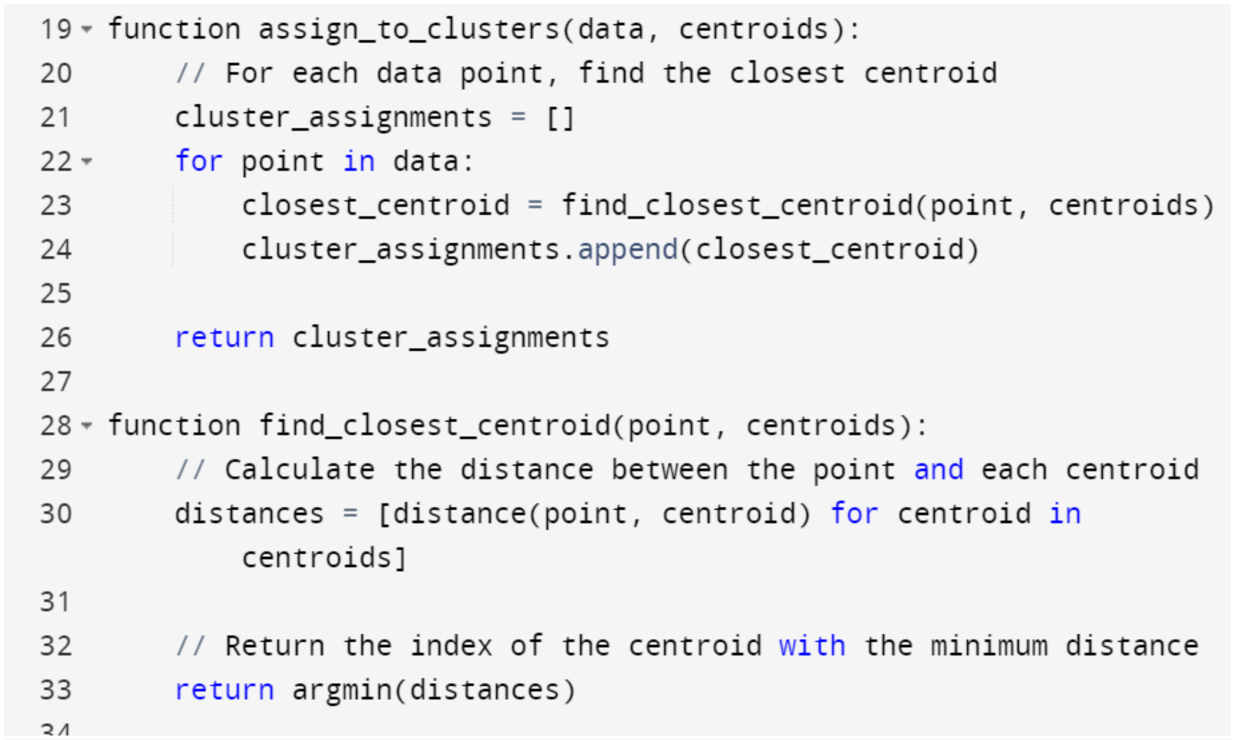
Algorithm:

Step 1: Given n objects, initialize k clusters centers/centroids.
Step 2: Randomly select k points to represent the cluster centers.
Step 3:For each data point, Calculate the distance to each centroids.
Step 4: Assign each object to its closest cluster. (by taking the min. distance)
Step 5: Update the center of each cluster.
Step 6: Repeat 2 and 3 until no change in each cluster center or a specified number of iterations is reached.

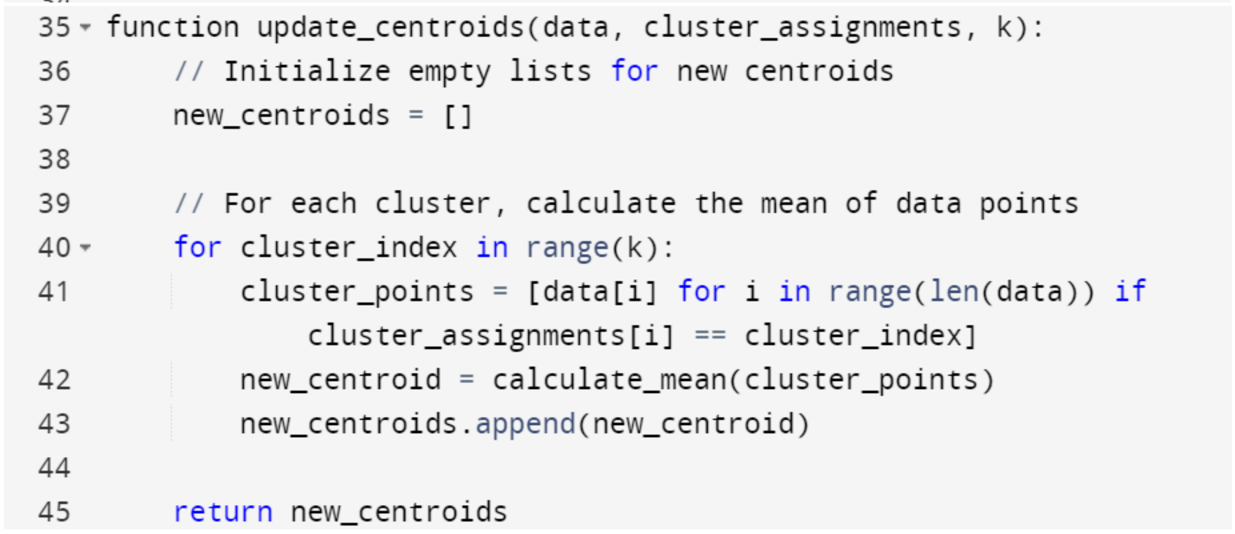
k-means from scratch:



```
function kMeans(data, k, max_iterations):
    // Step 1: Initialization
    centroids = randomly_initialize_centroids(data, k)

    for iteration in range(max_iterations):
        // Step 2: Assignment
        cluster_assignments = assign_to_clusters(data, centroids)

        // Step 3: Update Centroids
        centroids = update_centroids(data, cluster_assignments, k)

    return centroids, cluster_assignments

function randomly_initialize_centroids(data, k):
    // Randomly choose k data points as initial centroids
    centroids = random.sample(data, k)
    return centroids

function assign_to_clusters(data, centroids):
    // For each data point, find the closest centroid
```



```
function assign_to_clusters(data, centroids):
    // For each data point, find the closest centroid
    cluster_assignments = []
    for point in data:
        closest_centroid = find_closest_centroid(point, centroids)
        cluster_assignments.append(closest_centroid)

    return cluster_assignments

function find_closest_centroid(point, centroids):
    // Calculate the distance between the point and each centroid
    distances = [distance(point, centroid) for centroid in centroids]

    // Return the index of the centroid with the minimum distance
    return argmin(distances)

```



```
function update_centroids(data, cluster_assignments, k):
    // Initialize empty lists for new centroids
    new_centroids = []

    // For each cluster, calculate the mean of data points
    for cluster_index in range(k):
        cluster_points = [data[i] for i in range(len(data)) if cluster_assignments[i] == cluster_index]
        new_centroid = calculate_mean(cluster_points)
        new_centroids.append(new_centroid)

    return new_centroids
```

Built in k-means:

```python
from sklearn.cluster import KMeans
import numpy as np

# Sample Data
data = np.array([[2, 10], [2, 5], [8, 4], [5, 8], [7, 5], [6, 4],
    [1, 2], [4, 9]])

# Number of Clusters (k)
k = 3

# K-means Algorithm
kmeans = KMeans(n_clusters=k, random_state=0).fit(data)

# Cluster Labels and Centroids
labels = kmeans.labels_
centroids = kmeans.cluster_centers_

# Print Results
print("Cluster Labels:", labels)
print("Cluster Centers:", centroids)
```

Agglomerative Hierarchical Algorithm (AGNES) :

AGNES is a bottom-up approach that starts with each object individually forming a cluster of its own. These single clusters are then iteratively merged to form larger and larger clusters. The objects nearest to each other are merged together until all the objects are merged into a single cluster.

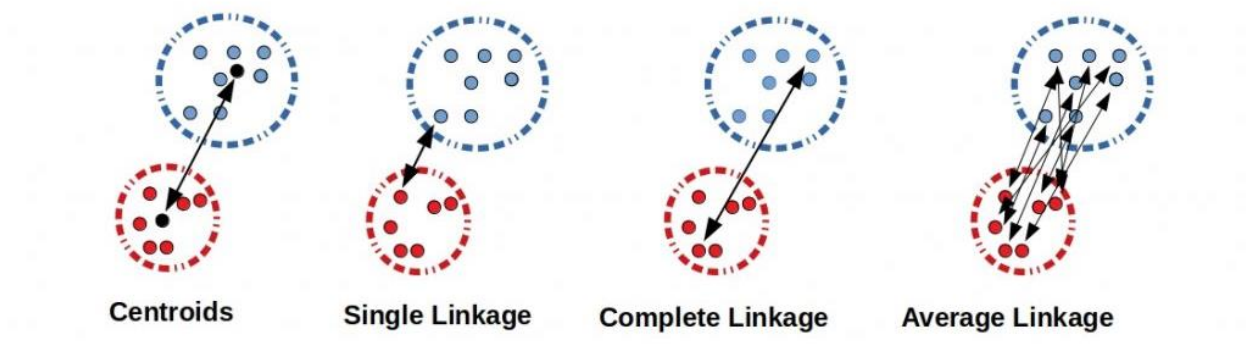
Algorithm:

Step 1: initialize each data point as a single cluster.
Step 2: Iteratively merge the two closest clusters into a new cluster.
Step 3: Recalculate the distances between the new cluster and the remaining clusters.

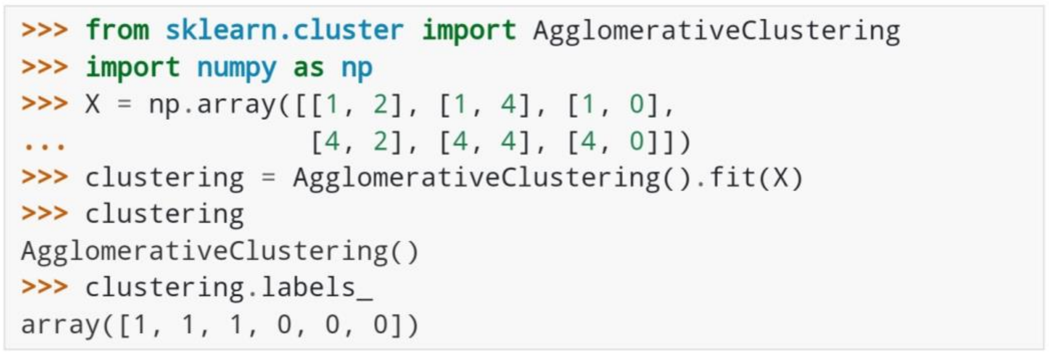
Step 4: Repeat steps 2 and 3 until the desired number of clusters is achieved or until a stopping criterion is met.

There are different methods to measure the distance between clusters, which include Single link, Complete link and Average link.

• Single link: is the shortest distance between two clusters.
This is done by checking the distance between all points in both clusters and selecting the smallest.
• Complete link: is the longest distance between two clusters.
This is done by checking the distance between all points in both clusters and selecting the largest
• Average link: is the average distance between two clusters.
We do so by calculating the average distance between all points in both clusters.



Sklearn's Agglomerative Clustering:



```
>>> from sklearn.cluster import AgglomerativeClustering
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4], [1, 0],
...               [4, 2], [4, 4], [4, 0]])
>>> clustering = AgglomerativeClustering().fit(X)
>>> clustering
AgglomerativeClustering()
>>> clustering.labels_
array([1, 1, 1, 0, 0, 0])
```

Comparison:

K-means is a partitioning algorithm, while Agglomerative is hierarchical.
K-means requires specifying the number of clusters (K) beforehand, while Agglomerative can provide a hierarchy.

Practice:

You are the owner of a shop and you have some basic data about your customers like customer ID, age, gender, occupation, etc. You want to perform customer segmentation (subdivision of a market into discrete customer groups that share similar characteristics) using this data.

The dataset can be downloaded from:

https://www.kaggle.com/datasets/dev0914sharma/customer-clustering/data

Given this data, you are required to cluster customers. Perform the following steps:

- Preprocess the data
- Use k-means clustering with different k values (2, 3, 4)
- Use agglomerative clustering and plot the dendrogram
- Compare the time of k-means and agglomerative clustering on this data