k-means clustering

K-means clustering is an unsupervised machine learning algorithm used to group a dataset into k clusters. It is an iterative algorithm that starts by randomly selecting k centroids in the dataset. After selecting the centroids, the entire dataset is divided into clusters based on the distance of the data points from the centroid. In the new clusters, the centroids are calculated by taking the mean of the data points.

With the new centroids, we regroup the dataset into new clusters. This process continues until we get a stable cluster. K-means clustering is a partition clustering algorithm. We call it partition clustering because of the reason that the k-means clustering algorithm partitions the entire dataset into mutually exclusive clusters.

K-means Clustering Algorithm

To understand the process of clustering using the k-means clustering algorithm and solve the numerical example, let us first state the algorithm. Given a dataset of N entries and a number K as the number of clusters that need to be formed, we will use the following steps to find the clusters using the k-means algorithm.

1. First, we will select K random entries from the dataset and use them as centroids.
2. Now, we will find the distance of each entry in the dataset from the centroids. You can use any distance metric such as euclidean distance, Manhattan distance, or squared euclidean distance.
3. After finding the distance of each data entry from the centroids, we will start assigning the data points to clusters.We will assign each data point to the cluster with the centroid to which it has the least distance.
4. After assigning the points to clusters, we will calculate the new centroid of the clusters. For this, we will use the mean of each data point in the same cluster as the new centroid. If the newly created centroids are the same as the centroids in the previous iteration, we will consider the current clusters to be final. Hence, we will stop the execution of the algorithm. If any of the newly created centroids is different from the centroids in the previous iteration, we will go to step 2.

Q 1: Suppose we have the following data points in two dimensions:
(2, 3), (3, 5), (4, 6), (6, 8), (7, 7), (8, 1), (9, 4), (10, 2)
Perform *k*-means clustering with k=2 (if not given).
Cluster 1 centroid: (3, 5)
Cluster 2 centroid: (8, 1)

Solution:
We can start by randomly initializing two cluster centroids:

Next, we assign each data point to the nearest centroid. We calculate the distance from each point to both centroids and assign each point to the centroid it's closest to.
After assigning points to clusters, we update the centroids by computing the mean of all points assigned to each cluster.
We repeat this process iteratively until convergence, meaning the centroids no longer change significantly between iterations.

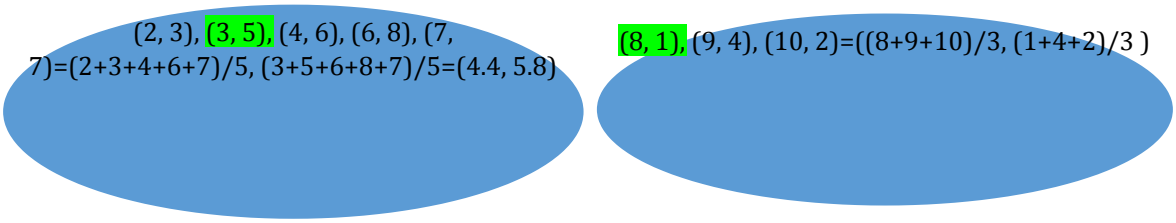
1st iteration:

Assigning points to clusters:

Data Objects	Seed 1/ Cluster 1-(3, 5)	Seed 2(Cluster 2) ((8, 1))
(2, 3),	$\sqrt{(2-3)^2+(3-5)^2}=\sqrt{1+4}=\sqrt{5}$	$\sqrt{(2-8)^2+(3-1)^2}=\sqrt{36+4}=\sqrt{40}=$
(3, 5),		
(4, 6),		
(6, 8),		
(7, 7),		
(8, 1),		
(9, 4),		
(10, 2)		

Updating centroids:

New centroid for Cluster 1: Mean of (2, 3), (3, 5), (4, 6), (6, 8), (7, 7)
New centroid for Cluster 1: ((2+3+4+6+7)/5, (3+5+6+8+7)/5) = (4.4, 5.8)
New centroid for Cluster 2: Mean of (8, 1), (9, 4), (10, 2)
New centroid for Cluster 2: ((8+9+10)/3, (1+4+2)/3) = (9, 2.3)



(2, 3), (3, 5), (4, 6), (6, 8), (7, 7) are closer to Cluster 1 centroid.
(8, 1), (9, 4), (10, 2) are closer to Cluster 2 centroid

Data Objects	Seed 1/ Cluster 1-(4.4, 5.8)	Seed 2(Cluster 2) (9, 2.3)
(2, 3),		
(3, 5),		
(4, 6),		
(6, 8),		
(7, 7),		
(8, 1),		
(9, 4),		
(10, 2)		

Now, we repeat the process until convergence. This iterative process continues until the centroids stabilize or a predefined number of iterations is reached.

Using the updated centroids from the previous step, we reassign the points to the nearest centroid and update the centroids again until convergence.

Iteration 2:

Assigning points to clusters:
(2, 3) and (3, 5) are closer to Cluster 1 centroid (2.5, 4).
(6, 8), (8, 1), and (9, 4) are closer to Cluster 2 centroid (7.67, 4.33).
Updating centroids:

New centroid for Cluster 1: (2.5, 4)
New centroid for Cluster 2: (7.67, 4.33)
Since there is no change in the assignment of points to clusters, the algorithm has converged. So, the final clusters are:
Cluster 1:
(2, 3)
(3, 5)
Cluster 2:
(6, 8)
(8, 1)
(9, 4)
Q2:
Perform k-means clustering on this dataset with k=2. Provide step-by-step details of your solution, including the initialization of centroids, the assignment of data points to clusters, and the update of centroids until convergence. Finally, present the final clusters along with their centroids.

(2, 3)
(3, 2)
(5, 8)
(6, 7)
(8, 2)

(1, 4)
(3, 6)
(7, 8)
(2, 7)
(8, 6)

Solution: Perform k-means clustering with k=2:

Step 1: Initialization

Randomly choose initial centroids:

Centroid 1: (2, 3)

Centroid 2: (6, 7)

Step 2: Assign Points to Clusters

Calculate the Euclidean distance between each data point and each centroid:

For example, the distance between (2, 3) and each centroid would be calculated as follows:

Distance to Centroid 1: sqrt((2-2)^2 + (3-3)^2) = 0
Distance to Centroid 2: sqrt((2-6)^2 + (3-7)^2) ≈ 5.66
Assign each data point to the cluster with the nearest centroid:

Data points (2, 3), (3, 2), (8, 2), and (1, 4) are closer to centroid 1.
Data points (5, 8), (6, 7), (7, 8), (3, 6), and (8, 6) are closer to centroid 2.
Step 3: Update Centroids

Compute the mean of all data points in each cluster:

Centroid 1: (3.5, 2.75) (Mean of (2, 3), (3, 2), (8, 2), and (1, 4))

Centroid 2: (5.8, 7) (Mean of (5, 8), (6, 7), (7, 8), (3, 6), and (8, 6))

Step 4: Repeat Steps 2 and 3 Until Convergence

Continue iterating steps 2 and 3 until the centroids no longer change significantly.

In this case, the algorithm converges after a single iteration.

Step 5: Final Clusters

After convergence, the final clusters and centroids would be:

Cluster 1: {(2, 3), (3, 2), (8, 2), (1, 4)} (Centroid: (3.5, 2.75))

Cluster 2: {(5, 8), (6, 7), (7, 8), (3, 6), (8, 6)} (Centroid: (5.8, 7))

This concludes the k-means clustering process for k=2 on the given dataset.

There are several methods to identify outliers in a dataset. Here are some commonly used techniques:

Standard Deviation Method: Identify data points that fall outside a specified number of standard deviations from the mean. Typically, values more than 2 or 3 standard deviations away from the mean are considered outliers.

- Interquartile Range (IQR) Method: Calculate the interquartile range (IQR), which is the difference between the third quartile (Q3) and the first quartile (Q1). Outliers are often defined as data points that fall below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR.
- Box Plot Method: Use a box plot to visually identify outliers. Data points lying outside the "whiskers" of the box plot are considered outliers.
- Z-Score Method: Calculate the Z-score for each data point, which represents the number of standard deviations a data point is from the mean. Outliers are typically defined as data points with a Z-score greater than a certain threshold (e.g., 2 or 3).
- Distance-based Methods (e.g., DBSCAN): Utilize clustering algorithms to identify outliers based on the density of data points. Data points that do not belong to any cluster or belong to clusters with very few members may be considered outliers.
- Local Outlier Factor (LOF) Method: Calculate the local outlier factor for each data point, which measures the degree of outlier-ness based on the density of its neighbors. Data points with a high LOF are considered outliers.
- Tukey's Method: Similar to the IQR method, but with a different threshold for defining outliers. Outliers are defined as data points that fall below Q1 - k * IQR or above Q3 + k * IQR, where k is a user-defined constant (typically 1.5 or 3).
- Visualization Techniques: Plot histograms, scatter plots, or other visualizations to identify data points that appear unusual or deviate significantly from the rest of the data.

Each method has its advantages and limitations, and the choice of method depends on factors such as the distribution of the data, the presence of noise, and the specific characteristics of the dataset. It's often a good idea to use multiple methods and compare results to ensure robust outlier detection.

Solution:

To find outliers using the 1.5*IQR method, follow these steps:

Step 1: Calculate the Quartiles

Arrange the dataset in ascending order: 65, 68, 70, 71, 72, 75, 76, 77, 80, 85
Calculate the first quartile (Q1), which represents the 25th percentile.
Calculate the third quartile (Q3), which represents the 75th percentile.
Step 2: Calculate the Interquartile Range (IQR)

The IQR is calculated as the difference between the third quartile (Q3) and the first quartile (Q1).

Step 3: Determine the Lower and Upper Bounds

Calculate the lower bound (LB) as Q1 - 1.5 * IQR and the upper bound (UB) as Q3 + 1.5 * IQR.

Step 4: Identify Outliers

Any data point outside the range [LB, UB] is considered an outlier.

Now, let's proceed with the calculations:

Quartiles:

Q1 = 70
Q3 = 77
Interquartile Range (IQR):

IQR = Q3 - Q1 = 77 - 70 = 7
Lower and Upper Bounds:

LB = Q1 - 1.5 * IQR = 70 - 1.5 * 7 = 70 - 10.5 = 59.5
UB = Q3 + 1.5 * IQR = 77 + 1.5 * 7 = 77 + 10.5 = 87.5
Identify Outliers:

Any data point below 59.5 or above 87.5 is considered an outlier.
Checking the dataset, all values fall within the range [59.5, 87.5]. Hence, there are no outliers in the given dataset.