ASSIGNMENT NO	A4
TITLE	**Assignment on K-Means Clustering**
PROBLEM STATEMENT/ DEFINITION	We have given a collection of 8 points. P1=[0.1,0.6] ,P2=[0.15,0.71], P3=[0.08,0.9] P4=[0.16, 0.85], P5=[0.2,0.3], P6=[0.25,0.5], P7=[0.24,0.1], P8=[0.3,0.2]. Perform the k-mean clustering with initial centroids as m1=P1 = Cluster#1=C1 and m2=P8=cluster#2=C2. Answer the following 1] Which cluster does P6 belong to? 2] What is the population of cluster around m2? 3] What is updated value of m1 and m2?
OBJECTIVE	To understand how k-means clustering algorithm works on the given dataset
OUTCOME	To implement k-means clustering algorithm.
S/W PACKAGES AND HARDWARE APPARATUS USED	Core 2 DUO/i3/i5/i7 64-bit processor OS-LINUX 64 bit OS Editor-gedit/Eclipse S/W- Jupyter Notebook/Weka/Python
REFERENCES	1. Giuseppe Bonaccorso, " Machine Learning Algorithms", Packt Publishing Limited, ISBN-10: 1785889621, ISBN-13: 978-1785889622 2. Josh Patterson, Adam Gibson, "Deep Learning : A Practitioners Approach", O'REILLY, SPD, ISBN: 978-93-5213-604-9, 2017 Edition 1st. 3. Nikhil Buduma, "Fundamentals of Deep Learning", O'REILLY publication, Second Edition, 2017,ISBN: 1491925612
STEPS	Let X = $\{x_1,x_2,x_3,……..,x_n\}$ be the set of data points and V = $\{v_1,v_2,…….,v_c\}$ be the set of centers. 1) Randomly select '*c*' cluster centers. 2) Calculate the distance between each data point and cluster

	centers. 3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.. 4) Recalculate the new cluster center using: $$v_i = (1/c_i) \sum_{j=1}^{c_i} x_i$$ where, 'c_i' represents the number of data points in i^{th} cluster. 5) Recalculate the distance between each data point and new obtained cluster centers. 6) If no data point was reassigned then stop, otherwise repeat from step 3).
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment No. 3. Problem Definition 4. Learning Objective 5. Learning Outcome 6. Concepts Related Theory 7. Algorithm 8. Test Cases 9. Conclusion/Analysis

- **Prerequisites:** Basic knowledge about Algorithms and any programming knowledge Java/python
- **Concepts related Theory**

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.

You'll define a target number *k*, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster.

Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares.

In other words, the K-means algorithm identifies *k* number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

The *'means'* in the K-means refers to averaging of the data; that is, finding the centroid.

How the K-means algorithm works

K-means algorithm is an iterative algorithm that tries to partition the dataset into *K* pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

Finally, this algorithm aims at minimizing an objective function know as squared error function given by:

$$J(V) = \sum_{i=1}^{c} \sum_{j=1}^{c_i} \left(\left\| x_i - v_j \right\| \right)^2$$

where,

'$||x_i - v_j||$' is the Euclidean distance between x_i and v_j.

'c_i' is the number of data points in i^{th} cluster.

'c' is the number of cluster centers.

Algorithmic steps for k-means clustering

Let X = {x_1,x_2,x_3,……..,x_n} be the set of data points and V = {v_1,v_2,…….,v_c} be the set of centers.

1) Randomly select 'c' cluster centers.

2) Calculate the distance between each data point and cluster centers.

3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..

4) Recalculate the new cluster center using:

$$v_i = (1/c_i) \sum_{j=1}^{c_i} x_i$$

where, 'c_i' represents the number of data points in i^{th} cluster.

5) Recalculate the distance between each data point and new obtained cluster centers.

6) If no data point was reassigned then stop, otherwise repeat from step 3).

It halts creating and optimizing clusters when either:

- The centroids have stabilized — there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

Notes:

- Since clustering algorithms including kmeans use distance-based measurements to determine the similarity between data points, it's recommended to standardize the data to have a mean of zero and a standard deviation of one since almost always the features in any dataset would have different units of measurements such as age vs income.
- Given kmeans iterative nature and the random initialization of centroids at the start of the algorithm, different initializations may lead to different clusters since kmeans algorithm may *stuck in a local optimum and may not converge to global optimum*. Therefore, it's recommended to run the algorithm using different initializations of centroids and pick the results of the run that that yielded the lower sum of squared distance.

Determining the **optimal number of clusters** in a data set is a fundamental issue in partitioning clustering, such as k-means clustering, which requires the user to specify the number of clusters k to be generated.

Unfortunately, there is no definitive answer to this question. The optimal number of clusters is somehow subjective and depends on the method used for measuring similarities and the parameters used for partitioning.
A simple and popular solution consists of inspecting the dendrogram produced using hierarchical clustering to see if it suggests a particular number of clusters. Unfortunately, this approach is also subjective.

These methods include direct methods and statistical testing methods:

1. Direct methods: consists of optimizing a criterion, such as the within cluster sums of squares or the average silhouette. The corresponding methods are named *elbow* and *silhouette* methods, respectively.
2. Statistical testing methods: consists of comparing evidence against null hypothesis. An example is the *gap statistic*.

In addition to *elbow*, *silhouette* and *gap statistic* methods, there are more than thirty other indices and methods that have been published for identifying the optimal number of clusters.

The Elbow method:

The Elbow method looks at the total WSS as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't improve much better the total WSS.

The optimal number of clusters can be defined as follow:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 10 clusters.
2. For each k, calculate the total within-cluster sum of square (wss).
3. Plot the curve of wss according to the number of clusters k.
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

Note that, the elbow method is sometimes ambiguous. An alternative is the average silhouette method which can be also used with any clustering approach.

Average silhouette method

The average silhouette approach we'll be described comprehensively in the chapter cluster validation statistics. Briefly, it measures the quality of a clustering. That is, it determines how well each object lies within its cluster. A high average silhouette width indicates a good clustering.

Average silhouette method computes the average silhouette of observations for different values of k. The optimal number of clusters k is the one that maximize the average silhouette over a range of possible values for k.

The algorithm is similar to the elbow method and can be computed as follow:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 10 clusters.
2. For each k, calculate the average silhouette of observations (*avg.sil*).
3. Plot the curve of *avg.sil* according to the number of clusters k.
4. The location of the maximum is considered as the appropriate number of clusters

Gap statistic method

The *gap statistic* has been published by R. Tibshirani, G. Walther, and T. Hastie (Standford University, 2001). The approach can be applied to any clustering method.

The gap statistic compares the total within intra-cluster variation for different values of k with their expected values under null reference distribution of the data. The estimate of the optimal clusters will be value that maximize the gap statistic (i.e, that yields the largest gap statistic). This means that the clustering structure is far away from the random uniform distribution of points.

The algorithm works as follow:

1. Cluster the observed data, varying the number of clusters from k = 1, …, k_{max}, and compute the corresponding total within intra-cluster variation W_k.
2. Generate B reference data sets with a random uniform distribution. Cluster each of these reference data sets with varying number of clusters k = 1, …, k_{max}, and compute the corresponding total within intra-cluster variation W_{kb}.
3. Compute the estimated gap statistic as the deviation of the observed W_k value from its expected value W_{kb} under the null hypothesis: $Gap(k)=1/B\sum log(W*kb)-log(Wk)$ where b=1 to B. Compute also the standard deviation of the statistics.
4. Choose the number of clusters as the smallest value of k such that the gap statistic is within one standard deviation of the gap at k+1: $Gap(k)\geq Gap(k+1)-s_{k+1}$.

<u>Advantages of Kmeans clsustering algorithm:</u>

1) Fast, robust and easier to understand.

2) Relatively efficient: O(tknd), where n is # objects, k is # clusters, d is # dimension of each object, and t is # iterations. Normally, k, t, d << n.

3) Gives best result when data set are distinct or well separated from each other.

<u>Disadvantages of Kmeans clsustering algorithm:</u>

1) The learning algorithm requires apriori specification of the number of cluster centers.

2) The use of Exclusive Assignment - If there are two highly overlapping data then k-means will not be able to resolve that there are two clusters.

3) The learning algorithm is not invariant to non-linear transformations i.e. with different representation of data we get

different results (data represented in form of cartesian co-ordinates and polar co-ordinates will give different results).

4) Euclidean distance measures can unequally weight underlying factors.

5) The learning algorithm provides the local optima of the squared error function.

6) Randomly choosing of the cluster center cannot lead us to the fruitful result.

7) Applicable only when mean is defined i.e. fails for categorical data.

8) Unable to handle noisy data and outliers.

9) Algorithm fails for non-linear data set.

Applications

k-means algorithm is very popular and used in a variety of applications such as market segmentation, document clustering, image segmentation and image compression, etc. The goal usually when we undergo a cluster analysis is either:

1. Get a meaningful intuition of the structure of the data we're dealing with.
2. Cluster-then-predict where different models will be built for different subgroups if we believe there is a wide variation in the behaviors of different subgroups. An example of that is clustering patients into different subgroups and build a model for each subgroup to predict the probability of the risk of having heart attack.

Test Cases:
From the given dataset.