**K-means Clustering Algorithm**

k-means is  one of  the simplest unsupervised  learning  algorithms  that  solve  the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other.

**Algorithmic steps for k-means clustering**

Let  X = {x1,x2,x3,……..,xn} be the set of data points and V = {v1,v2,…….,vc} 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:  , where,*‘ci’* represents the number of data points in *ith* 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.

**Advantages**

* Fast, robust and easier to understand.
* Gives best result when data set are distinct or well separated from each other.

**Disadvantages**

* The learning algorithm requires apriori specification of the number of cluster centers.
* Randomly choosing of the cluster center cannot lead us to the fruitful result.
* Applicable only when mean is defined i.e. fails for categorical data.
* Algorithm fails for non-linear data set.

*Example*

Divide the data points {(1, 1), ((2, 1), (4, 3), (5, 4)} into two clusters.

***Solution***

Let p1=(1,1) p2=(2,1) p3=(4,3) p4=(5,4)

***Initial step***

Let c1=(1,1) and c2=(2,1) are two initial cluster centers.

***Iteration 1***

Calculate distance between clusters centers and each data points

Thus after first iteration

*Cluster 1= {p1}*

*Cluster 2={p2,p3,p4}*

Now, new cluster centers are:

c1=(1,1) and c2={(2+4+5)/3,(1+3+4)/3}=(11/3,8/3)

***Iteration 2***

Calculate distance between new cluster centers and each data points

Thus after second iteration

*Cluster 1= {p1,p2}*

*Cluster 2={p3,p4}*

Now, new cluster centers are:

c1={(1+2)/2,(1+1)/2}={3/2,1} and c2={(2+4+5)/3,(1+3+4)/3}=(11/3,8/3)

Repeat this process till no re-assignment of points to groups.

**K-medoid Clustering Algorithm**

A *medoid* can be defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal. i.e. it is a most centrally located point in the cluster. In contrast of K-means algorithm, K-medoid algorithm chooses data point as centers and works with arbitrary matrix of distances instead of *l2*.*K-medoid* is a classical partitioning technique of clustering that clusters the data set of *n* objects into *k* clusters known *a priori*. It is more robust to noise and outliers as compared to because it may minimize a sum of pair-wise dissimilarities instead of a sum of squared Euclidean distances.

Algorithms

The most common realization of *k*-medoid clustering is the **Partitioning around Medoid (PAM)** algorithm and is as follows:

1. Initialize: randomly select (without replacement) *k* of the *n* data points as the medoid
2. Associate each data point to the closest medoid.
3. While the cost of the configuration decreases:

* For each medoid *m*, for each non-medoid data point *o*:
* Swap *m* and *o*, re-compute the cost (sum of distances of points to their medoid)
* If the total cost of the configuration increased in the previous step, undo the swap

***Example***

Cluster the following data set of ten objects into two clusters i.e. *k* = 2. Data Points are {(1,3), (4,2), (6,2), (3,5), (4,1)}

***Solution***

Let p1=(1,3) p2=(4,5) p3=(6,3) p4=(3,4) p5=(2,1) in two clusters using k-mediods algorithm

***Initial step***

Let m1=p1=(1,3) and m2=p4=(3,4) are two initial mediod

d(m1,p2) = mode(x2-x1)+mode(y2-y1)

***Iteration 1***

Calculate distance between clusters centers and each data point

d(m1,p1)=0

Thus after first iteration

*Cluster 1= {p1, p5}*

*Cluster 2={p2,p3,p4}*

Total Cost= {d(m1,p1)++++d(m2,p4)}= {0+3+1+4+0} =8

***Iteration 2***

Now m1=p2= (4,5) and m2=p4=(3,4) are two mediod

Let p1=(1,3) p2=(4,5) p3=(6,3) p4=(3,4) p5=(2,1)

Calculate distance between new cluster centers and each data points

Thus after second iteration

*Cluster 1= {p2,p3}*

*Cluster 2={p1,p4,p5}*

Total Cost= {d(m1,p2)+++d(m2,p4)+}=0+3+3+0+4=10

Total cost = 10

**Hierarchical Clustering**

In data mining and statistics, **hierarchical clustering** (also called **hierarchical cluster analysis** or **HCA**) is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

* **Agglomerative**: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
* **Divisive**: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

Distance matrix is used for deciding which clusters to merge/split. There are losts of alternatives to define distance between two sets of points.

**Agglomerative Clustering Algorithm**

Algorithm

* 1. Compute the distance matrix between the input data points
  2. Let each data point be a cluster
  3. **Repeat**
     + Merge the two closest clusters
     + Update the distance matrix
  4. **Until** only K clusters remains

***Example***

Cluster the data points (1,1), (1.5,1.5), (5,5), (3,4), (4,4), (3, 3.5) into two clusters.

***Solution***

Assume A=(1,1), B= (1.5,1.5), C=(5,5), D=(3,4), E=(4,4), F=(3,3.5)

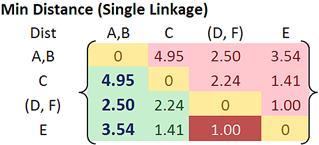
Distance Matrix



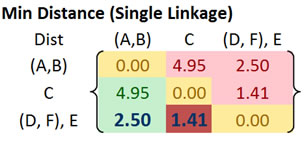
In this case, the closest cluster is between cluster F and D with shortest distance of 0.5. Thus, we group cluster D and F into cluster (D, F). Then we update the distance matrix (see distance matrix below). Distance between ungrouped clusters will not change from the original distance matrix. Now the problem is how to calculate distance between newly grouped clusters (D, F) and other clusters?



Looking at the lower triangular updated distance matrix, we found out that the closest distance between cluster B and cluster A is now 0.71. Thus, we group cluster A and cluster B into a single cluster name (A, B). Now we update the distance matrix. Aside from the first row and first column, all the other elements of the new distance matrix are not changed.



Observing the lower triangular of the updated distance matrix, we can see that the closest distance between clusters happens between cluster E and (D, F) at distance 1.00. Thus, we cluster them together into cluster ((D, F), E ). The updated distance matrix is given below



After that, we merge cluster ((D, F), E) and cluster C into a new cluster name (((D, F), E), C). The updated distance matrix is shown in the figure below

