K-medoids Clustering Algorithm

Partitioning Around Medoids or the K-medoids algorithm is a partitional clustering algorithm which is slightly modified from the K-means algorithm. They both attempt to minimize the squared-error but the K-medoids algorithm is more robust to noise than K-means algorithm. In K-means algorithm, they choose means as the centroids but in the K-medoids, data points are chosen to be the medoids. A medoid can be defined as that object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal.

The difference between k-means and k-medoids is analogous to the difference between mean and median: where mean indicates the average value of all data items collected, while median indicates the value around that which all data items are evenly distributed around it. The basic idea of this algorithm is to first compute the K representative objects which are called as medoids. After finding the set of medoids, each object of the data set is assigned to the nearest medoid. That is, object *i* is put into cluster $v_{i,}$ when medoid mv_i is nearer than any other medoid m_w.

The algorithm proceeds in two steps:

- *BUILD-step:* This step sequentially selects *k* "centrally located" objects, to be used as initial medoids
- *SWAP-step:* If the objective function can be reduced by interchanging (swapping) a selected object with an unselected object, then the swap is carried out. This is continued till the objective function can no longer be decreased.

The algorithm is as follows:

1. Initially select k random points as the medoids from the given n data points of the data set.
2. Associate each data point to the closest medoid by using any of the most common distance metrics.
3. For each pair of non-selected object *h* and selected object *i*, calculate the total swapping cost TC_{ih}.
    - If $TC_{ih} < 0$, *i* is replaced by *h*
4. Repeat the steps 2-3 until there is no change of the medoids.

There are four situations to be considered in this process:

i. *Shift-out membership*: an object p_i may need to be shifted from currently considered cluster of o_j to another cluster;
ii. *Update the current medoid*: a new medoid o_c is found to replace the current medoid o_j ;
iii. *No change*: objects in the current cluster result have the same or even smaller square error criterion(SEC) measure for all the possible redistributions considered;
iv. *Shift-in membership*: an outside object pi is assigned to the current cluster with the new (replaced) medoid o_c.

PAM works efficiently for small data sets but does not scale well for large data sets. The complexity of this algorithm is $O(k(n-k)^2)$.

Example:

For a given k=2, cluster the following data set using PAM.

Point	x-axis	y-axis
1	7	6
2	2	6
3	3	8
4	8	5
5	7	4
6	4	7
7	6	2
8	7	3
9	6	4
10	3	4

Let us choose that (3, 4) and (7, 4) are the medoids. Suppose considering the Manhattan distance metric as the distance measure,

So, now if we calculate the distance from each point:
For (7, 6), Calculating the distance from the medoids chosen, this point is nearest to (7, 4)
For (2, 6) , Calculating the distance from the medoids chosen, this point is nearest to (3, 4)
For (3, 8) , Calculating the distance from the medoids chosen, this point is at same distance from both the points. So choosing that it is nearest to (3, 4)
For (8, 5) , Calculating the distance from the medoids chosen, this point is nearest to (7, 4)
For (4, 7) , Calculating the distance from the medoids chosen, this point is nearest to (3, 4)
For (6, 2) , Calculating the distance from the medoids chosen, this point is nearest to (7, 4)
For (7, 3) , Calculating the distance from the medoids chosen, this point is nearest to (7, 4)
For (6, 4) , Calculating the distance from the medoids chosen, this point is nearest to (7, 4)

So, now after the clustering, the clusters formed are: {(3,4), (2,6), (3,8), (4,7)} and{(7,4), (6,2), (6,4), (7,3), (8,5), (7,6)}. Now calculating the cost which is nothing but the sum of distance of each non-selected point from the selected point which is medoid of the cluster it belongs to.

Total Cost = cost((3, 4), (2, 6)) + cost((3, 4), (3, 8)) + cost((3, 4), (4, 7)) + cost((7, 4), (6, 2))+ cost((7, 4), (6, 4))+ cost((7, 4), (7, 3))+ cost((7, 4), (8, 5))+ cost((7, 4), (7, 6))
= 3 + 4 + 4 + 3 + 1 + 1 + 2 + 2
= 20.

So, now let us choose some other point to be a medoid instead of (7, 4). Let us randomly choose (7, 3). Not the new medoid set is: (3, 4) and (7, 3). Now repeating the same task as earlier:

So, now if we calculate the distance from each point:
For (7, 6), Calculating the distance from the medoids chosen, this point is nearest to (7, 3)
For (2, 6) , Calculating the distance from the medoids chosen, this point is nearest to (3, 4)
For (3, 8) , Calculating the distance from the medoids chosen, this point is nearest to (3, 4)
For (8, 5) , Calculating the distance from the medoids chosen, this point is nearest to (7, 3)
For (4, 7) , Calculating the distance from the medoids chosen, this point is nearest to (3, 4)
For (6, 2) , Calculating the distance from the medoids chosen, this point is nearest to (7, 3)
For (7, 4) , Calculating the distance from the medoids chosen, this point is nearest to (7, 3)
For (6, 4) , Calculating the distance from the medoids chosen, this point is nearest to (7, 3)

Calculating the total cost = cost((3, 4), (2, 6)) + cost((3, 4), (3, 8)) + cost((3, 4), (4, 7)) + cost((7, 3), (7, 6)) + cost((7, 3), (8, 5)) + cost((7, 3), (6, 2)) + cost((7, 3), (7, 4)) + cost((7, 3), (6, 4))
= 3 + 4 + 4 + 3 + 3 + 2 + 1 + 2
= 22.

The total cost when (7, 3) is the medoid > the total cost when (7, 4) was the medoid earlier. Hence, (7, 4) should be chosen instead of (7, 3) as the medoid. Since there is no change in the medoid set, the algorithm ends here. Hence the clusters obtained finally are: {(3,4), (2,6), (3,8), (4,7)} and{(7,4), (6,2), (6,4), (7,3), (8,5), (7,6)}.