Partitioning Methods

- Find mutually exclusive clusters of spherical shape
- Distance-based
- May use mean or medoid (etc.) to represent cluster center
- Effective for small- to medium-size data sets

k-Means: A Centroid-Based Technique

- A centroid-based partitioning technique uses the *centroid of a cluster, Ci , to represent* that cluster
- The difference between an object ***p € Ci and ci, the representative*** of the cluster, is measured by *dist(**p, ci), where dist(x,y) is the Euclidean distance*** between two points ***x and y.***
- ***The quality of cluster Ci can be measured by the withincluster* variation, which is the sum of *squared error between all objects in Ci and the*** centroid ***ci, defined as***

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, c_i)^2,$$

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
(5) **until** no change;

0.2 The *k*-means partitioning algorithm.

Disadvantages

- The *k-means method is not guaranteed to converge to the global optimum and often* terminates at a local optimum. The results may depend on the initial random selection of cluster centers.
- The time complexity of the *k-means algorithm is O(nkt), where n is the total number* of objects, *k is the number of clusters, and t is the number of iterations.*

Disadvantage

A drawback of *k*-means. Consider six points in 1-D space having the values 1, 2, 3, 8, 9, 10, and 25, respectively. Intuitively, by visual inspection we may imagine the points partitioned into the clusters {1, 2, 3} and {8, 9, 10}, where point 25 is excluded because it appears to be an outlier. How would *k*-means partition the values? If we apply *k*-means using $k = 2$ and Eq. (10.1), the partitioning $\{\{1, 2, 3\}, \{8, 9, 10, 25\}\}$ has the within-cluster variation

$$(1-2)^2 + (2-2)^2 + (3-2)^2 + (8-13)^2 + (9-13)^2 + (10-13)^2 + (25-13)^2 = 196,$$

given that the mean of cluster {1, 2, 3} is 2 and the mean of {8, 9, 10, 25} is 13. Compare this to the partitioning $\{\{1, 2, 3, 8\}, \{9, 10, 25\}\}$, for which *k*-means computes the within-cluster variation as

$$(1-3.5)^2 + (2-3.5)^2 + (3-3.5)^2 + (8-3.5)^2 + (9-14.67)^2 + (10-14.67)^2 + (25-14.67)^2 = 189.67,$$

given that 3.5 is the mean of cluster {1, 2, 3, 8} and 14.67 is the mean of cluster {9, 10, 25}. The latter partitioning has the lowest within-cluster variation; therefore, the *k*-means method assigns the value 8 to a cluster different from that containing 9 and 10 due to the outlier point 25. Moreover, the center of the second cluster, 14.67, is substantially far from all the members in the cluster. ■

Partitioning Around Medoids (PAM) algorithm

- It is a popular realization of *k-medoids clustering*
- *To diminish the sensitivity to outliers, i*nstead of taking the mean value of the objects in a cluster as a reference point, we can pick actual objects to represent the clusters, using one representative object per cluster.

Algorithm: *k*-medoids. PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects in *D* as the initial representative objects or seeds;
(2) **repeat**
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, o_{random};
(5) compute the total cost, *S*, of swapping representative object, o_j, with o_{random};
(6) **if** $S < 0$ **then** swap o_j with o_{random} to form the new set of *k* representative objects;
(7) **until** no change;

PAM, a *k*-medoids partitioning algorithm.

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)}.