**Partitioning Methods (Clustering)**

**1.K-Means**

**2.K-Medoids**

The **k-medoids algorithm** is a [clustering](http://en.wikipedia.org/wiki/Data_clustering) [algorithm](http://en.wikipedia.org/wiki/Algorithm) related to the [k-means](http://en.wikipedia.org/wiki/K-means) algorithm and the medoid shift algorithm. Both the k-means and k-medoids algorithms are partition (breaking the dataset up into groups) and both attempt to minimize the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the k-means algorithm, k-medoids chooses data points as centers ( [medoids](http://en.wikipedia.org/wiki/Medoids" \o "Medoids) or exemplars) and works with an arbitrary matrix of distances between data points instead of. This method was proposed in 1987 for the work with norm and other distances.

* *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 [*k*-means](http://en.wikipedia.org/wiki/K-means) because it minimizes a sum of pair wise dissimilarities instead of a sum of squared Euclidean distances.
* A [medoid](http://en.wikipedia.org/wiki/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.
* The most common realization of *k*-medoid clustering is the **Partitioning around Medoids (PAM)** **algorithm and is as follows:**

1. Initialize: randomly select (without replacement) *k* of the *n* data points as the medoids
2. Associate each data point to the closest medoid. ("closest" here is defined using any valid [distance metric](http://en.wikipedia.org/wiki/Metric_space), most commonly [Euclidean distance](http://en.wikipedia.org/wiki/Euclidean_distance), [Manhattan distance](http://en.wikipedia.org/wiki/Manhattan_distance) or [Minkowski distance](http://en.wikipedia.org/wiki/Minkowski_distance))
3. For each medoid *m*
   * For each non-medoid data point *o*
     + Swap *m* and *o* and compute the total cost of the configuration
4. Select the configuration with the lowest cost.
5. Repeat steps 2 to 4 until there is no change in the medoid.

Example1:

Cluster the following data set of ten objects into two clusters i.e. k = 2.

Consider a data set of ten objects as follows:

|  |  |  |
| --- | --- | --- |
| Samples | Attribute1 | Attribute 2 |
| X1 | 2 | 6 |
| X2 | 3 | 4 |
| X3 | 3 | 8 |
| X4 | 4 | 7 |
| X5 | 6 | 2 |
| X6 | 6 | 4 |
| X7 | 7 | 3 |
| X8 | 7 | 4 |
| X9 | 8 | 5 |
| X10 | 7 | 6 |

Solution

Initialize k centers. Let us assume x2 and x8 are selected as medoids, so the centers are c1 = (3, 4) and c2 = (7,4)

Calculate distances to each center so as to associate each data object to its nearest medoid. Cost is calculated using Euclidian distance. Costs to the nearest medoid are shown bold in the table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Samples | Attribute1 | Attribute 2 | Cost distance or dist mean (3,4) | Cost distance or dist mean (7,4) | clustering | Total cost |
| X1 | 2 | 6 | 3 | 7 | C1 | **3** |
| X2 | 3 | 4 | 0 | 4 | C1 | **0** |
| X3 | 3 | 8 | 4 | 8 | C1 | **4** |
| X4 | 4 | 7 | 4 | 6 | C1 | **4** |
| X5 | 6 | 2 | 5 | 3 | C2 | **3** |
| X6 | 6 | 4 | 3 | 1 | C2 | **1** |
| X7 | 7 | 3 | 5 | 1 | C2 | **1** |
| X8 | 7 | 4 | 4 | 0 | C2 | **0** |
| X9 | 8 | 5 | 6 | 2 | C2 | **2** |
| X10 | 7 | 6 | 6 | 2 | C2 | **2** |

Then the clusters become:

Cluster1 = {(3, 4), (2, 6) (3, 8) (4, 7)} ={x1, x2, x3, x4}

Cluster2 = {(7,4)(6,2)(6,4)(7,3)(8,5)(7,6)}=(X5,X6,X7,X8,X9,X10}

So the total cost involved is 20

**Iteration 2**

Select one of the non medoids O′ Let us assume O′ = (7, 3) =x7

So now the medoids are c1 (3, 4) and O′ (7, 3) If c1 and O′ are new medoids, calculate the total cost involved by using the formula in the step 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Samples | Attribute1 | Attribute 2 | Cost distance or dist mean (3,4) | Cost distance or dist mean (7,3) | clustering | Total cost |
| X1 | 2 | 6 | 3 | 8 | C1 | **3** |
| X2 | 3 | 4 | 0 | 5 | C1 | **0** |
| X3 | 3 | 8 | 4 | 9 | C1 | **4** |
| X4 | 4 | 7 | 4 | 7 | C1 | **4** |
| X5 | 6 | 2 | 5 | 2 | C2 | **2** |
| X6 | 6 | 4 | 3 | 2 | C2 | **2** |
| X7 | 7 | 3 | 5 | 0 | C2 | **0** |
| X8 | 7 | 4 | 4 | 1 | C2 | **1** |
| X9 | 8 | 5 | 6 | 3 | C2 | **3** |
| X10 | 7 | 6 | 6 | 3 | C2 | **3** |

So cost of swapping medoid from c2 to O′ is = 22 so moving to O′ would be a bad idea, so the previous choice was good. So we try other non medoids and found that our first choice was the best. So the configuration does not change and algorithm terminates here (i.e. there is no change in the medoids).

**Analysis in k-medoids**

* Pam is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean
* Pam works efficiently for small data sets but does not scale well for large data sets.
  + O(k(n-k)2 ) for each iteration where n is # of data, k is # of clusters
* Sampling based method,

CLARANS (Clustering LARge Applications based on randomized search)

* *CLARA* (Kaufmann and Rousseeuw in 1990)
  + Built in statistical analysis packages, such as S+
* It draws *multiple samples* of the data set, applies *PAM* on each sample, and gives the best clustering as the output
* Strength: deals with larger data sets than *PAM*
* Weakness:
  + Efficiency depends on the sample size
  + A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased