# Partitioning methods

Construct a partition of a database D of objects into a set of k clusters such that sum of squared distance is minimized.

Given a set of objects, a partitioning method construct k partitions of the data, where each partition represents a cluster and . That is, divide the data into k groups such that each group must have at least one object. In partitioning methods, each object must belong to exactly one cluster.

Most of these methods are distance-based. Given k, a partitioning method creates an initial partitioning and then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one cluster to another. Given a k, the partition of k clusters that optimizes the chosen partition criterion is found. The clusters are formed to optimize the criterion (such as dissimilarity function based on distance) so that the objects within a cluster are similar and dissimilar to objects in other clusters.

It is computationally demanding to find a global optimal partitioning. Instead heuristic methods like K-means and K-medoids is used where the clustering is improved until a local optimum is reached. These methods are good at finding spherical shapes (but partitioning-based methods need to be extended to find other shapes)

* K-means: Each cluster is represented by the center of the cluster
* K-medoids (PAM): Each cluster is represented by one of the objects in the cluster

**K-Means**

Given k, data D

1. Arbitrarily choose k objects as initial cluster centers
2. Repeat (until no change in cluster means):
   1. Assign each object to the cluster to which the object is most similar based on mean values of the objects in the cluster
   2. Update cluster means (calculate mean value of the objects for each cluster)

K-means often terminates at a local optimum. It is not guaranteed to converge to a global optimum.

Weaknesses:

* K-means only works on numerical data. Can only be used when mean is defined. Therefore, it doesn’t work on categorical data.
* We need to specify k (number of clusters) beforehand
* Unable to handle noisy data and outliers. Very sensitive to outliers, which can distort the distribution of the data
* Can’t discover clusters with non-convex shapes, can only find spherical shapes.

**K-Medoids (PAM = Partitioning Around Medoids)**

Instead of taking the mean value of the object in a cluster as a reference point, medoids are used instead, which is the most centrally located object in a cluster. Find representative objects called medoids in clusters. Also called PAM

PAM starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering.

PAM works by choosing the initial representatives objects (medoids) arbitrarily. We consider whether replacing a representative object by a non-representative object would improve the clustering quality. All possible replacements are tried out. The iterative process of replacing representative objects continues until the quality can’t improve anymore. This quality is measured by a cost function of the average dissimilarity between an object and the representative object of its cluster.

Start by arbitrarily choosing k objects as initial medoids. Assign each remaining object to nearest medoid. Randomly select a non-medoid object . Compute total cost of swapping, here -

Swapping and for best pair if quality is improved. Loop until no improvement can be made. Algorithm:

1. Select k representative objects arbitrarily (this will be your initial k medoids). Assign other objects to clusters defined by the k mediods.
2. For each pair of non-selected objects and selected medoid , calculate total swapping cost
3. Select a pair and which corresponds to the minimum swapping cost  
   If is replaced with   
   Then assign each non-selected object to the most similar representative object
4. Repeat steps 2-3 until there is no change

Comments about PAM:

* PAM is more robust than K-means when there’s outliers and noise in the data. A medoid is less influenced by outliers or other extreme values than a mean
* PAM works effectively for small data sets but does not scale well for large data sets.

**CLARA (Clustering LARge Applications)**

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

Weaknesses: 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.

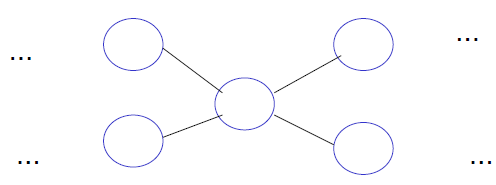
While PAM searches for the best k-medoids among the given data set, CLARA searches for the best k-medoids among the selected sample of the data. CLARA can’t find a good clustering if any of the best sampled medoids is far from the best k-medoids. If an object is one of the best k-medoids but is not selected during sampling, CLARA will never find the best clustering.

CLARA draws multiple samples and gives the best result as output. If the sample is drawn well than the medoids of the sample will approximate the true best medoids in all the data.

Algorithm (n=5, s=40+2k is the recommendation)

1. Repeat n times:
2. Draw samples of s objects from the entire data set and perform PAM to find k medoids of the sample
3. Assign each non-selected object in the entire data set to the most similar medoid
4. Calculate average dissimilarity of the clustering. If the value is smaller than the current minimum, use this value as current minimum and retain the k medoids found in step 2 as the best so far

**Graph representation**

****

* Each node represents k objects (medoids); a potential solution for the clustering.
* Nodes are neighbors if the sets of objects in the nodes differ by one object.  
  Each node has k(n-k) neighbors
* Cost differential between two neighbors is
* PAM searches for the node in the graph with minimum cost
* CLARA searches in smaller graphs (as it uses PAM on samples of the entire data set)
* CLARANS: searches in the original graph. Searches part of the graph. Uses the neighbors to guide the search

**CLARANS (“Randomized” CLARA, Clustering Large Applications based on RANdomized Search)**

Is more efficient and scalable than both PAM and CLARA. First it randomly selects k objects in the data as the current medoids. It then randomly selects a current medoid x and an object y that is not one of the current medoids. If replacing x by y improves the absolute-error criterion x is replaced by y as medoid. This randomized search is done times. The set of current medoids after loops is considered a local optimum. CLARANS repeats this times and returns the best local optimal as final result. Algorithm:

* Numlocal: number of local minima to be found
* Maxneighbor: maximum number of neighbors to compare
* Repeat Numlocal times: (find local minimum)
  1. Take arbitrary node in the graph
  2. Consider random neighbor S of the current node and calculate the cost differential. If S has lower cost, then set S to current and repeat this step. If S does not have lower cost, repeat this step (check at most Maxneighbor number of neighbors)
  3. Compare the cost of the current node with minimum cost so far. If the cost is lower, set minimum cost to cost of the current node, and bestnode to the current node