The K-Medoids Clustering Method

K-Medoids (also called as Partitioning Around Medoid) algorithm was proposed in 1987 by Kaufman and Rousseeuw. A medoid can be defined as the point in the cluster, whose dissimilarities with all the other points in the cluster is minimum.

- *PAM* (**P**artitioning **A**round **M**edoids, 1987)
  - 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 effectively for small data sets, but does not scale well for large data sets

K-medoids algorithm

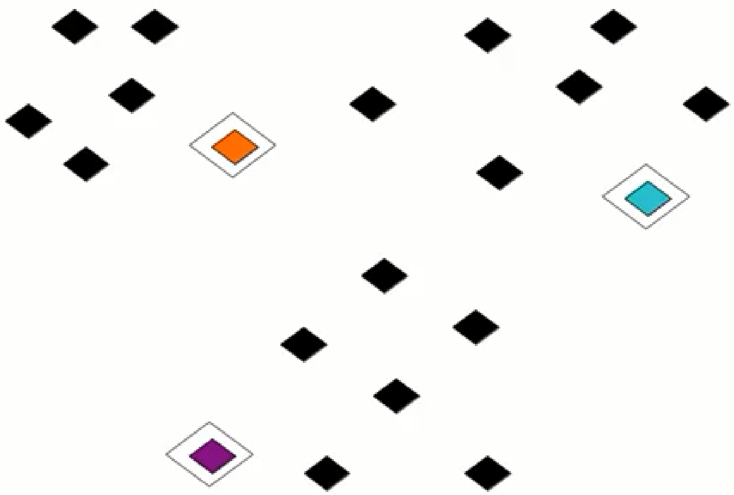
1. Initialize: select k random points out of the n data points as the medoids.
2. Associate each data point to the closest medoid by using any common distance metric methods.
3. While the cost decreases:

   For each medoid m, for each data o point which is not a medoid:

   1. Swap m and o, associate each data point to the closest medoid, recompute the cost.
   2. If the total cost is more than that in the previous step, undo the swap.

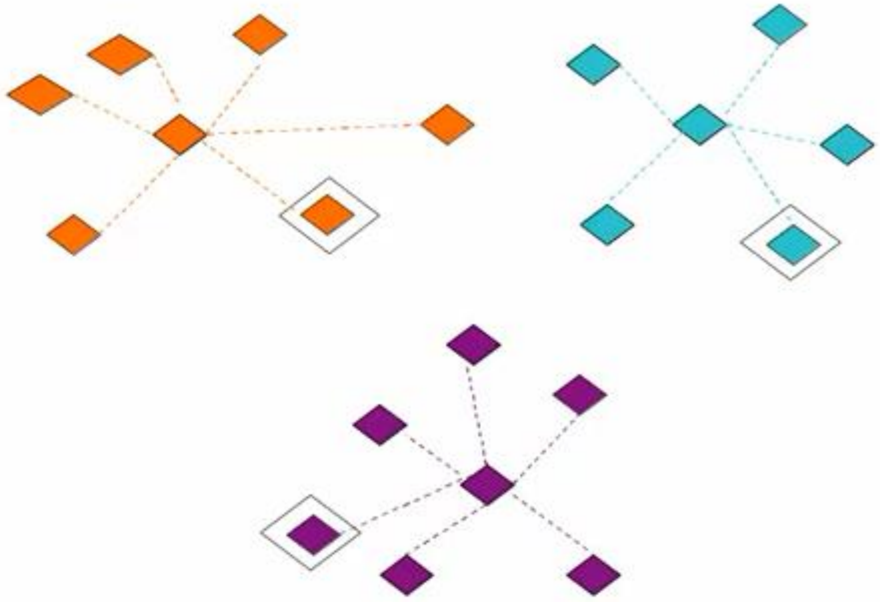
The K-Medoids Clustering Method

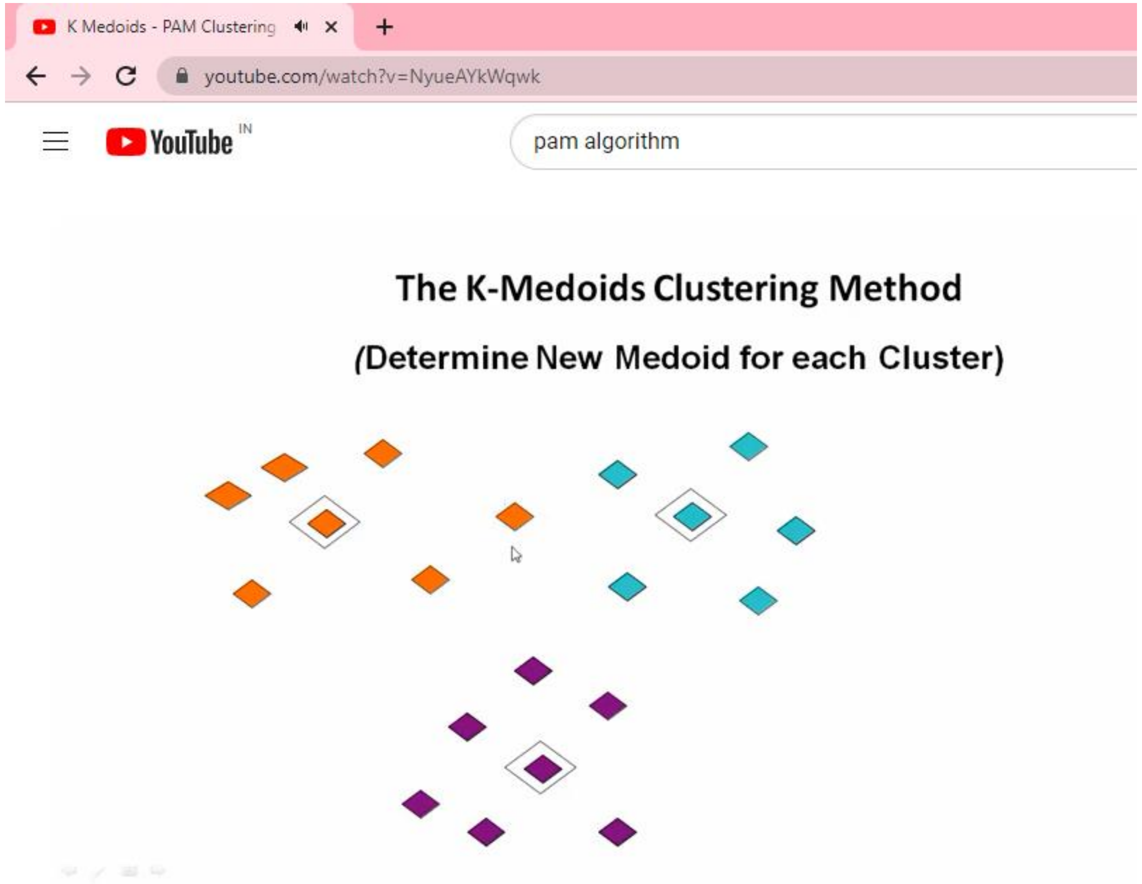
(select the randomly K-Medoids)

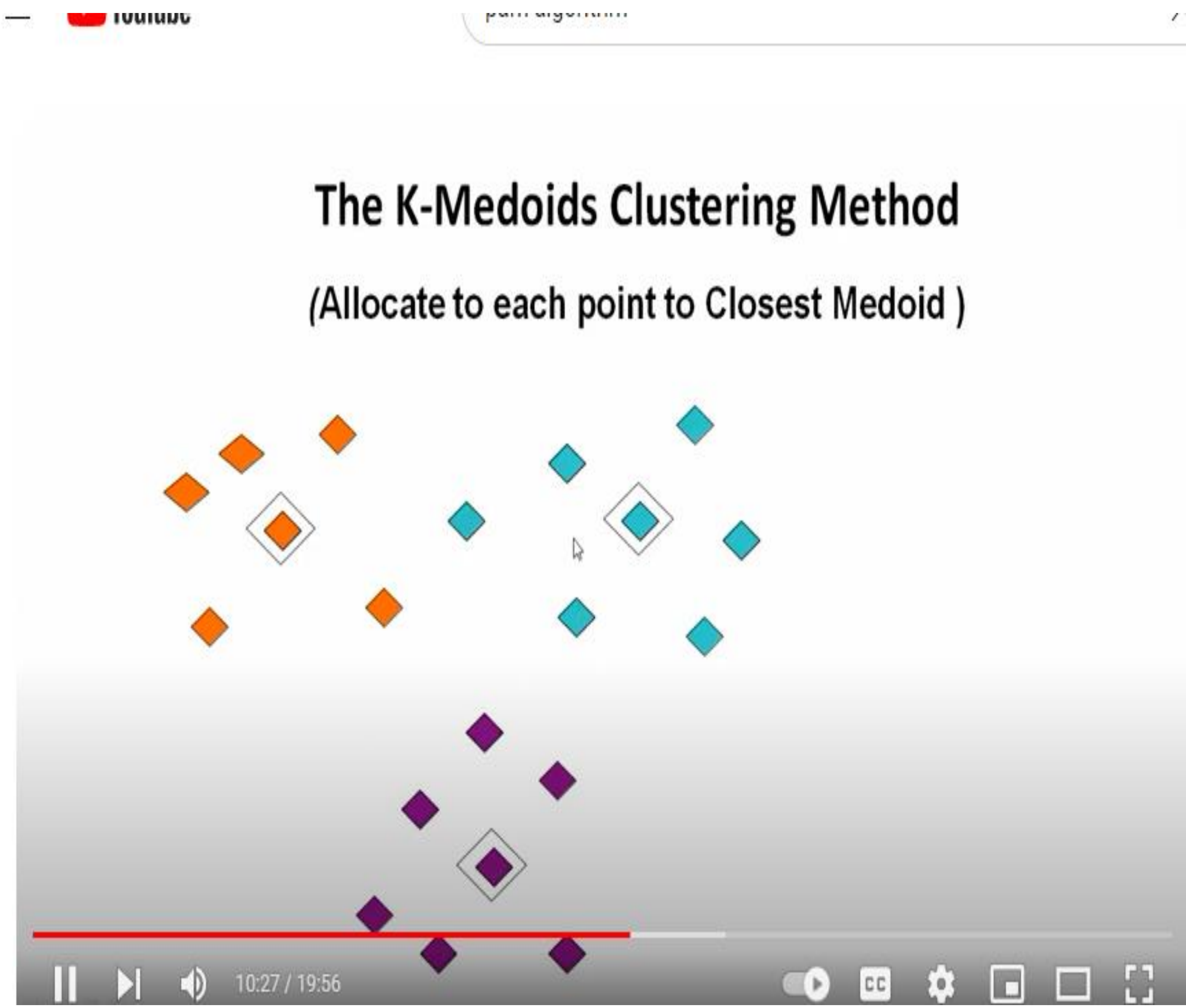


The K-Medoids Clustering M

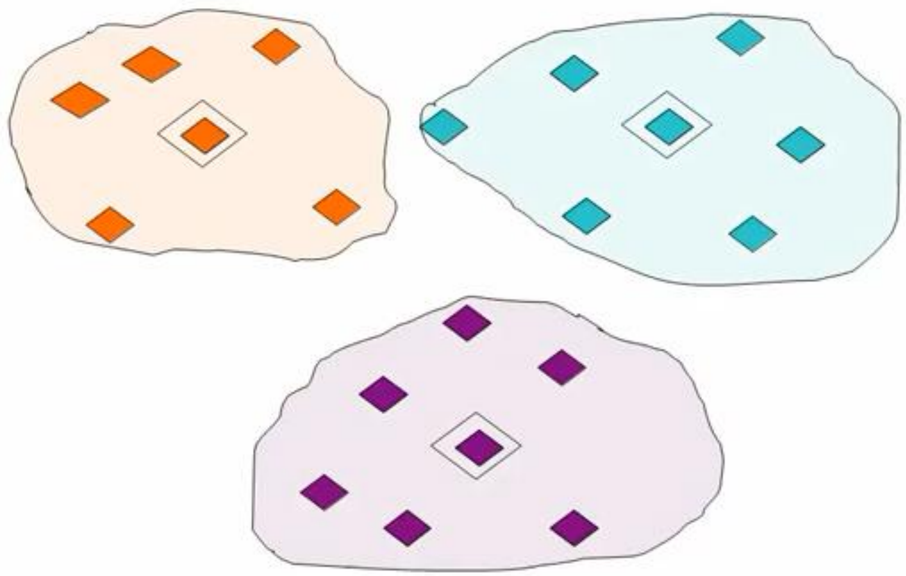
(Allocate to Each Point to Closes




K Medoids - PAM Clustering
youtube.com/watch?v=NyueAYkWqwk
YouTube
IN
pam algorithm
The K-Medoids Clustering Method
(Determine New Medoid for each Cluster)


The K-Medoids Clustering Method
(Allocate to each point to Closest Medoid)
10:27 / 19:56

The K-Medoids Clustering Method (Stop the process)



Cost

- The dissimilarity of the medoid(Ci) and object(Pi) is calculated by using E = |Pi - Ci|
- The cost in K-Medoids algorithm is given as:

$$c = \sum_{Ci} \sum_{Pi E Ci} | Pi - Ci |$$

PAM (Partitioning Around Medoids)

To overcome the problem of sensitivity to outliers (K-means), instead of taking the mean value as the centroid, K-medoid take actual data point to represent the cluster.

K-Medoids Algorithm(PAM)

PAM : Partitioning Around Medoids

- **Input**
  - K: the number of clusters
  - D: a data set containing n objects
- **Output**: A set of k clusters
- **Method**:

  (1) Arbitrary choose k objects from D as representative objects (seeds)

  (2) **Repeat**

  (3) Assign each remaining object to the cluster with the nearest representative object

  (4) For each representative object O_j

  (5) Randomly select a non representative object O_{random}

  (6) Compute the total cost **S** of swapping representative object Oj with O_{random}

  (7) **if S<0 then replace** O_j with O_{random}

  (8) **Until** no change

Input:

$D = \{t_1, t_2, ..., t_n\}$ // **Set of elements**

A // **Adjacency matrix showing distance between elements.**

k // **Number of desired clusters.**

Output:

K // **Set of clusters.**

PAM Algorithm:

```
arbitrarily select k medoids from D;
repeat
    for each t_h not a medoid do
        for each medoid t_i do
            calculate TC_ih;
    find i, h where TC_ih is the smallest;
    if TC_ih < 0 then
        replace medoid t_i with t_h;
until TC_ih ≥ 0;
for each t_i ∈ D do
    assign t_i to K_j where dis(t_i, t_j) is the smallest over all medoids;
```

Data set: {1,2,3,20,21,22} with K=2

Initial Assumption:

Cluster Medoids are 1 and 2 which is represented by M1(1) and M2(2)

Absolute Distance is used

Data Set	M1(1)	M2(2)	Min (2 and 3rd Column)
1	0	1	0
2	1	0	0
3	2	1	1
20	19	18	18
21	20	19	19
22	21	20	20
			Old Cost $\sum$=58

Other candidates for Medoids are 3,20,21,22

We can change only one medoid at a time.

We keep Medoid=2 as it is.

Instead of 1 as medoid, find best choices from 3,20,21 and 22.

TC-Total Cost

$TC_{1,3}$: Cost of changing from 1 to 3

$TC_{1,20}$: Cost of changing from 1 to 20

$TC_{1,21}$: Cost of changing from 1 to 21

$TC_{1,22}$: Cost of changing from 1 to 22

Data Set	M(2)	M(3) [Min with M(2)]	M(20) [Min with M(2)]	M(21) [Min with M(2)]	M(22) [Min with M(2)]
1	1	2 [1]	19 [1]	20 [1]	21 [1]
2	0	1 [0]	18 [0]	19 [0]	20 [0]
3	1	0 [0]	17 [1]	18 [1]	19 [1]
20	18	17 [17]	0 [0]	1 [1]	2 [2]
21	19	18 [18]	1 [1]	0 [0]	1 [1]
22	20	19 [19]	2 [2]	1 [1]	0 [0]
New Cost		**$TC_{1,3}$=55**	**$TC_{1,20}$=5**	**$TC_{1,21}$=4**	**$TC_{1,22}$=5**
New-Old		**-3**	**-53**	**-54**	**-53**

$TC_{1,21}$=-54 is suitable option as cost is minimum.

We keep Medoid=1 as it is.

Instead of 2 as medoid, find best choices from 3,20,21 and 22.

TC-Total Cost

$TC_{2,3}$: Cost of changing from 2 to 3

$TC_{2,20}$: Cost of changing from 2 to 20

$TC_{2,21}$: Cost of changing from 2 to 21

$TC_{2,22}$: Cost of changing from 2 to 22

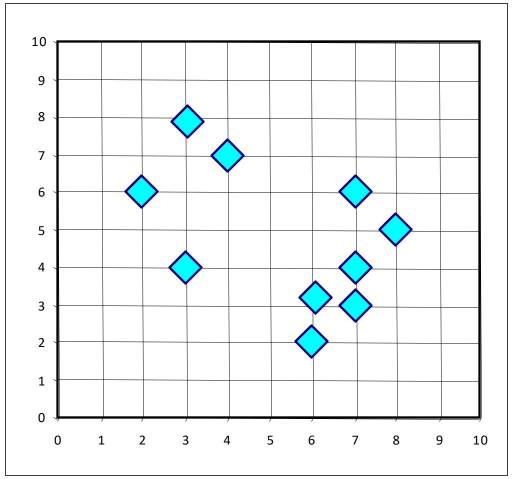
Data Set	M(1)	M(3) [Min with M(1)]	M(20) [Min with M(1)]	M(21) [Min with M(1)]	M(22) [Min with M(1)]
1	0	2 [0]	19 [0]	20 [0]	21 [0]
2	1	1 [1]	18 [1]	19 [1]	20 [1]
3	2	0 [0]	17 [2]	18 [2]	19 [2]
20	19	17 [17]	0 [0]	1 [1]	2 [2]
21	20	18 [18]	1 [1]	0 [0]	1 [1]
22	21	19 [19]	2 [2]	1 [1]	0 [0]
New Cost		**$TC_{2,3}$=55**	**$TC_{2,20}$=6**	**$TC_{2,21}$=5**	**$TC_{2,22}$=6**
New-Old		**-3**	**-52**	**-53**	**-52**

$TC_{2,21}$=-53 is suitable option as cost is minimum.

Therefore minimum cost is $TC_{1,21}$=-54

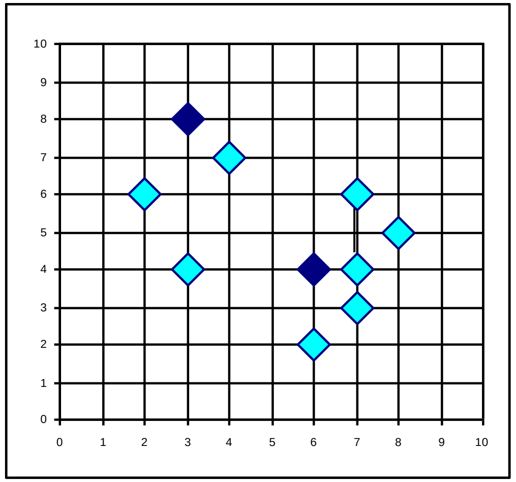
Therefore New Medoid are (2, 21)

PAM: A Typical K-Medoids Algorithm



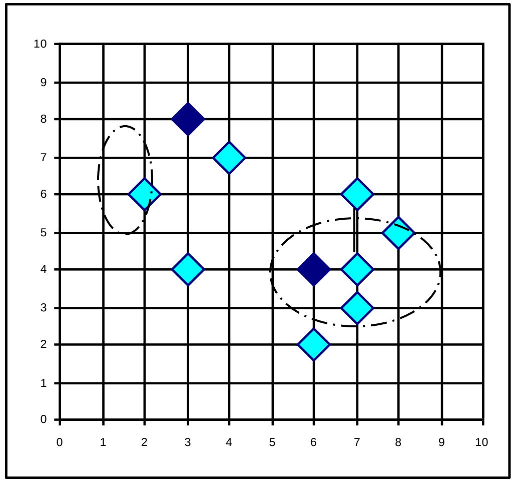
K=2

Arbitrary choose k object as initial medoids

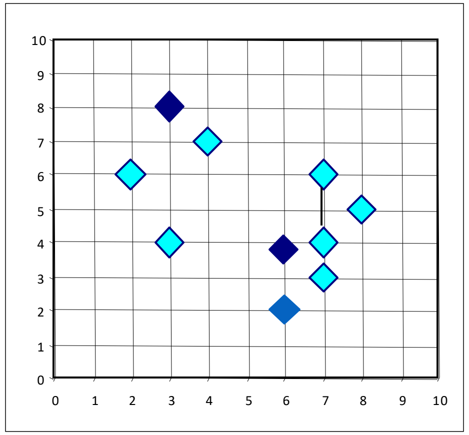


Assign each remaining object to nearest medoids

Total Cost = 20

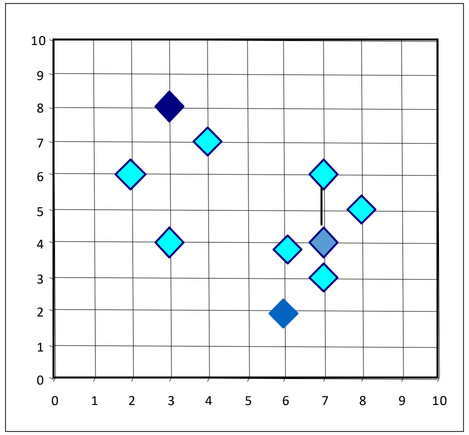


Randomly select a nonmedoid object,O_{ramdom}



Compute total cost of swapping

Total Cost = 26



Swapping O and O_{ramdom}

If quality is improved.

Do loop

Until no change