



Artificial Intelligence

Unsupervised Learning-Clustering



Clustering | Applications

In **exploratory data analysis (EDA)** clustering plays a fundamental role in developing initial intuition about features and patterns in data.

In **statistical analysis**, clustering is frequently used to identify the (dis)similarities variables in different samples.

Insurance industries use clustering for **anomaly detection** and potentially catch fraudulent transactions.

Clustering is widely used in **customer segmentation** — e.g. for developing marketing strategies targeting different groups of customers.

In **computer vision**, image segmentation is used to partition data into disjoint groups using pattern recognition.

Clustering is an essential tool in **biological sciences**, especially in genetic and taxonomic classification and understanding evolution of living and extinct organisms.

Clustering algorithms have wide-ranging other applications such as building **recommendation systems**, social media network analysis etc.



Clustering K-Means

- It is an unsupervised iterative clustering technique
- It partitions the given data into predefined K distinct clusters
- A cluster is defined as collection of data points exhibiting certain similarities

It partitions the data set such that-

- Each data point belongs to a cluster with the nearest mean.
- Data points belonging to one cluster have high degree of similarity (intra-cluster).
- Data points belonging to different clusters have high degree of dissimilarity (inter-cluster).



K-Means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid Number of clusters, K , must be specified
- The basic algorithm is very simple

1: Select K points as the initial centroids.

2: **repeat**

3: Form K clusters by assigning all points to the closest centroid.

4: Recompute the centroid of each cluster.

5: **until** The centroids don't change

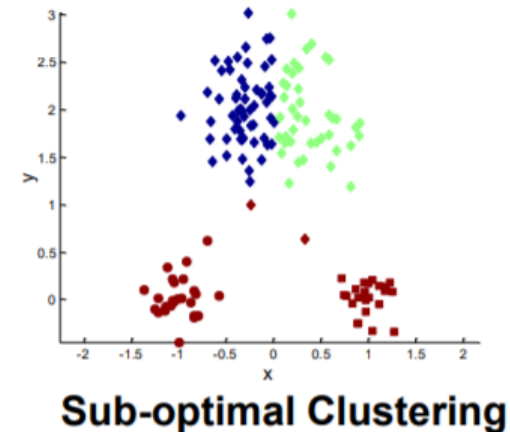
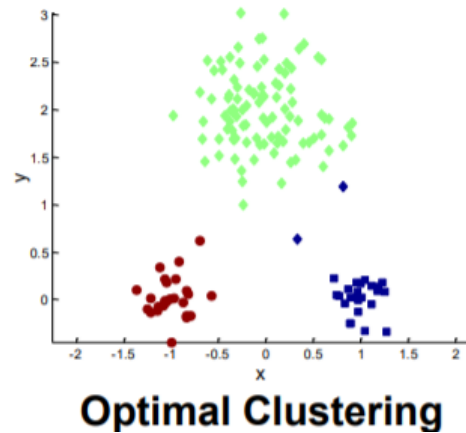
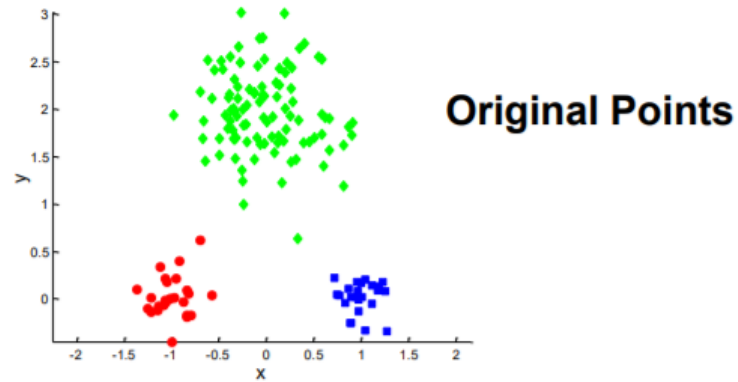


Details

- Initial centroids are often chosen randomly.
 - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- 'Closeness' is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
 - Often the stopping condition is changed to 'Until relatively few points change clusters'
- Complexity is $O(n * K * I * d)$
 - n = number of points, K = number of clusters,
 I = number of iterations, d = number of attributes

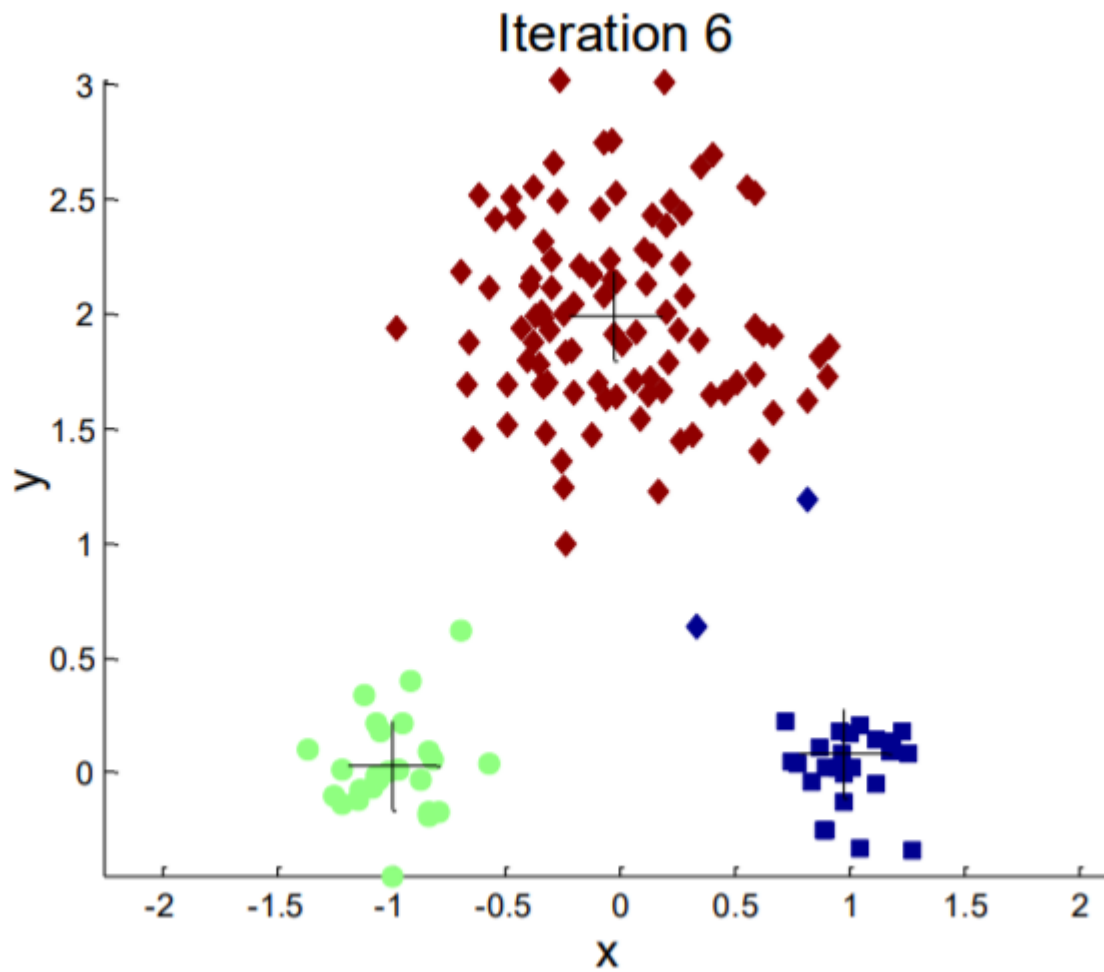


Different K-means Clustering

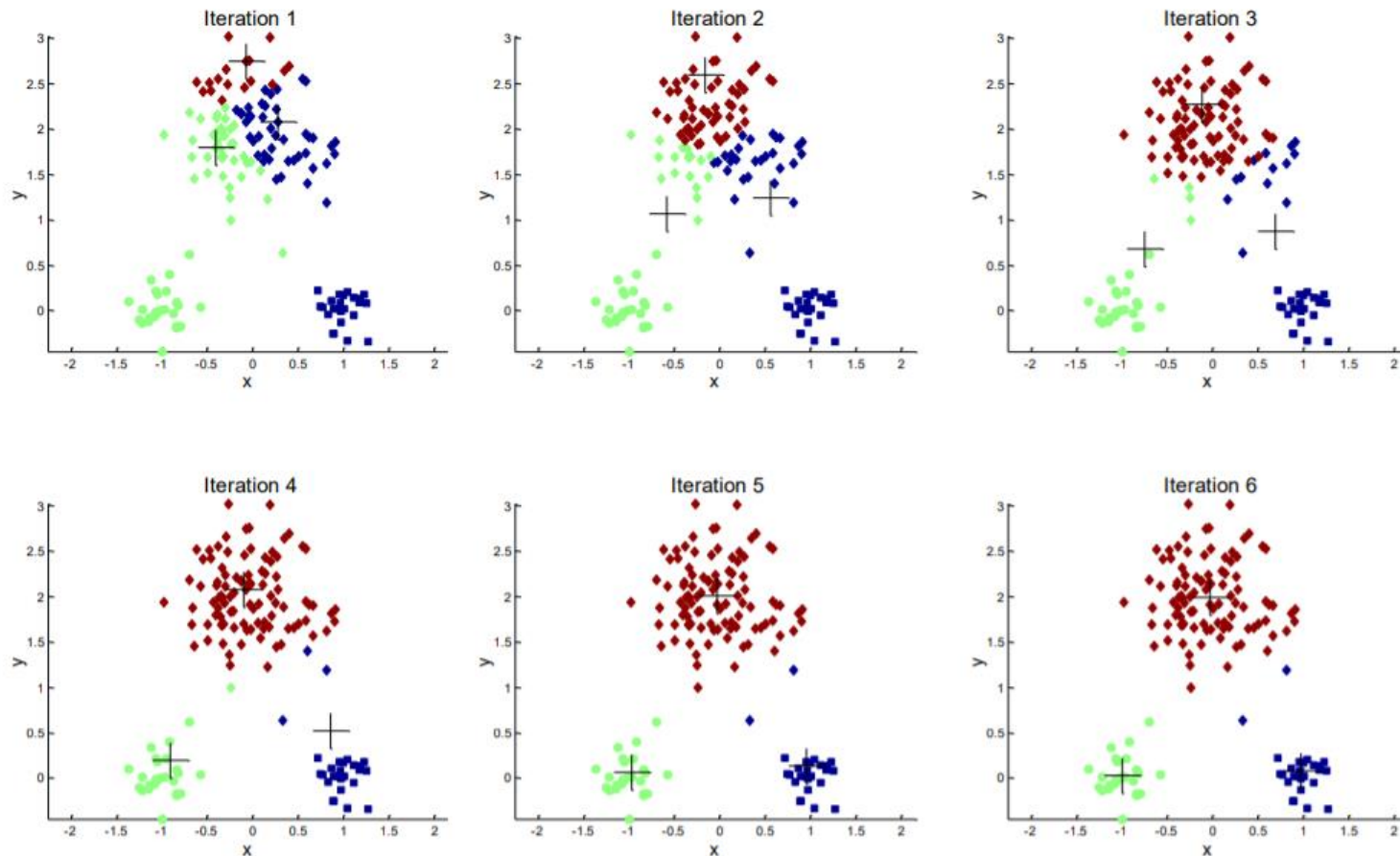




Importance of Choosing Initial Centroid



Importance of choosing initial centroid



Example: K=2

Let K=2

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5



Example: Step 1

Step 1:

Initialization: Randomly we choose following two centroids ($k=2$) for two clusters. In this case the 2 centroid are: $m1=(1.0,1.0)$ and $m2=(5.0,7.0)$.

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Individual	Mean Vector
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)



Example: Step 2

1. Compute distance of each given point from the 2 centroids (1,1) and (5,7)
2. Assign 1-3 points to cluster 1 and 4-7 points to cluster 2

□ Thus, we obtain two clusters containing:
{1,2,3} and {4,5,6,7}.

3. Compute new centroids by taking mean of x, y coordinates for both clusters (to take the mean u calculate by using points assigned to the clusters)

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

$$d(m_1, 2) = \sqrt{|1.0 - 1.5|^2 + |1.0 - 2.0|^2} = 1.12$$

$$d(m_2, 2) = \sqrt{|5.0 - 1.5|^2 + |7.0 - 2.0|^2} = 6.10$$

Example: Step 3

Compute new centroids by taking mean of x, y coordinates for both clusters (to take the mean u calculate by using points assigned to the clusters)

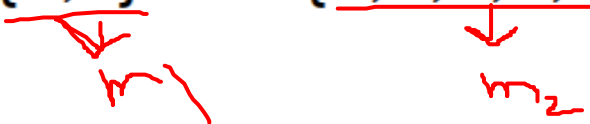
$$m_1 = \left(\frac{1}{3}(1.0 + 1.5 + 3.0), \frac{1}{3}(1.0 + 2.0 + 4.0) \right) = (1.83, 2.33)$$

$$m_2 = \left(\frac{1}{4}(5.0 + 3.5 + 4.5 + 3.5), \frac{1}{4}(7.0 + 5.0 + 5.0 + 4.5) \right) = (4.12, 5.38)$$

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

$$d(m_1, 1) = \sqrt{(1 - 1.83)^2 + (1 - 2.33)^2} = 1.56$$
$$d(m_2, 1) = \sqrt{(4.2 - 1)^2 + (5.38 - 1)^2} = 5.37$$

Example: Step 4

- Now using these centroids we compute the Euclidean distance of each object, as shown in table.
- Therefore, the new clusters are:
 $\{1,2\}$ and $\{3,4,5,6,7\}$

 m_1 and m_2
- Next centroids are:
 $m_1 = (1.25, 1.5)$ and
- $m_2 = (3.9, 5.1)$

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2 (1.5, 2.0)	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

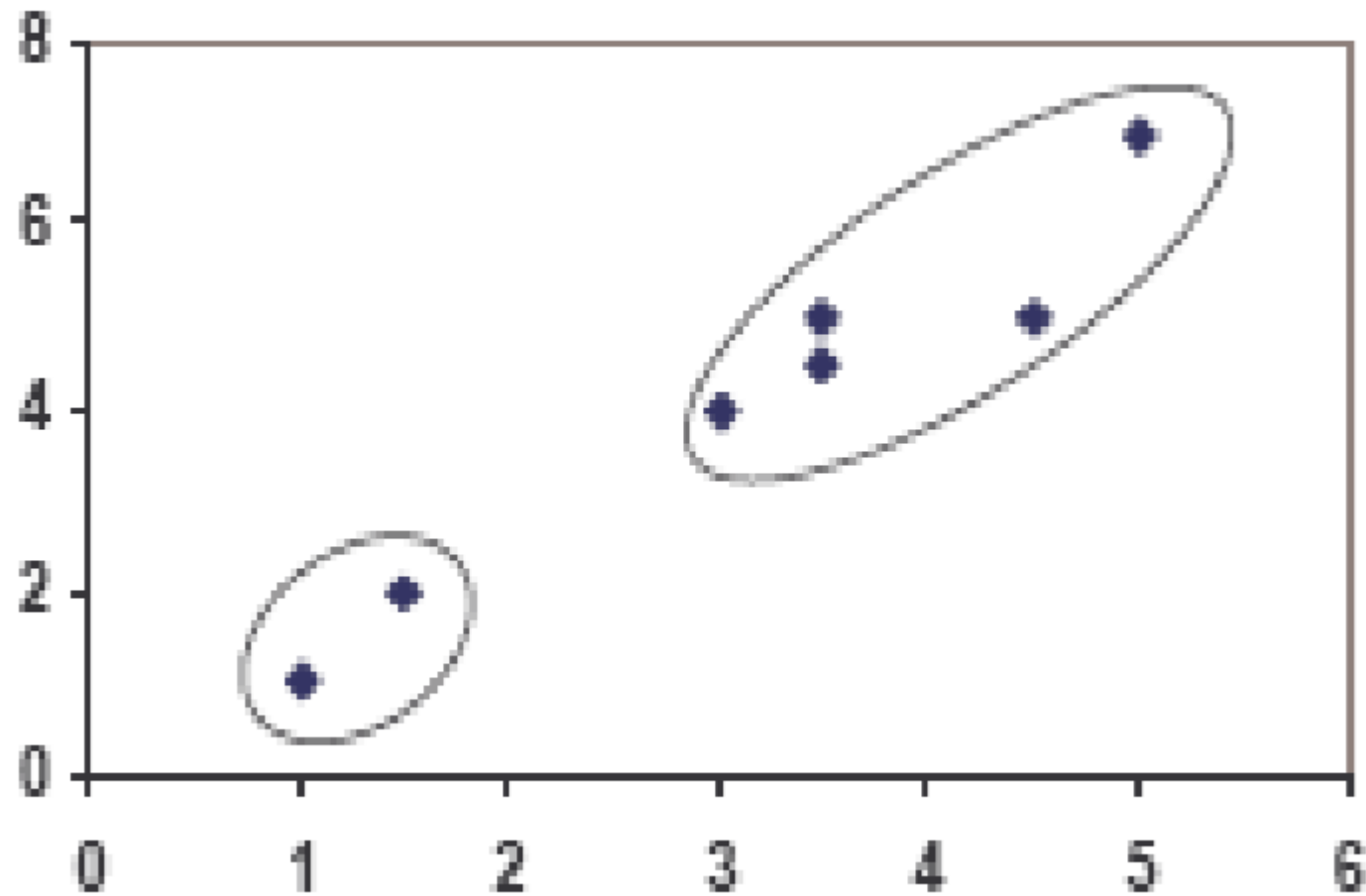
Example: Step 5

- The clusters obtained are:
{1,2} and {3,4,5,6,7}
- Therefore, there is no change in the cluster.
- Thus, the algorithm comes to a halt here and final result consist of 2 clusters:
{1,2} and {3,4,5,6,7}.

Individual	Centroid 1	Centroid 2
1	0.56	5.02
2	0.56	3.92
3	3.05	1.42
4	6.66	2.20
5	4.16	0.41
6	4.78	0.61
7	3.75	0.72



Example: Plot



Example: With $K=3$

Point	M1 (Point 1)	M2 (Point 2)	M3 (Point 3)	Cluster
1	0	1.11	3.61	1
2	1.12	0	2.5	2
3	3.61	2.5	0	3
4	7.21	6.10	3.61	3
5	4.72	3.61	1.12	3
6	5.31	4.24	1.80	3
7	4.30	3.20	0.71	3

Clustering with initial centroids set to the first three points

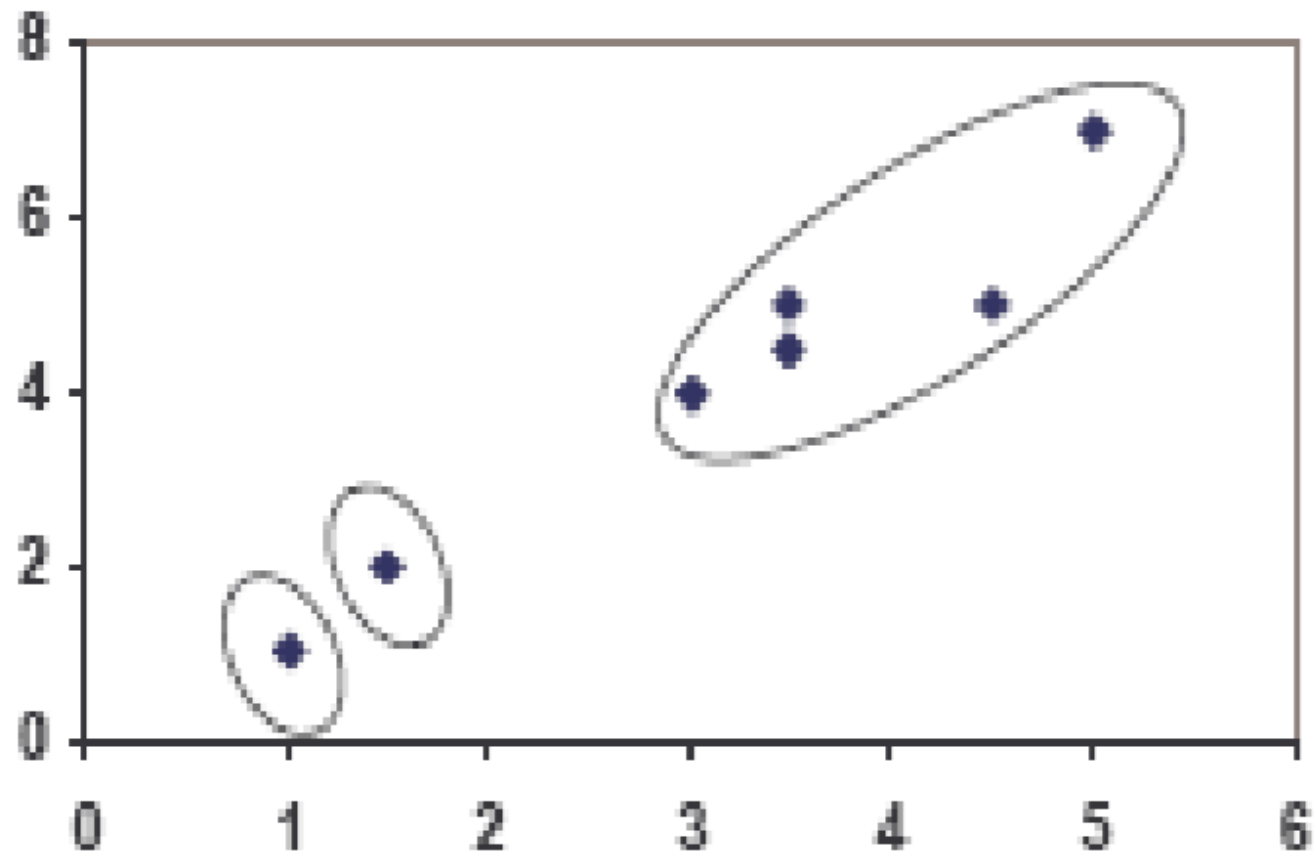
Step 1

Point	M1 (1.0,1.0)	M2 (1.5,2.0)	M3 (3.9,5.1)	Cluster
1	0	1.11	5.02	1
2	1.12	0.00	3.92	2
3	3.61	2.50	1.42	3
4	7.21	6.10	2.20	3
5	4.72	3.61	0.41	3
6	5.31	4.24	0.61	3
7	4.30	3.20	0.72	3

Step 2



Example: Plot





Evaluating K-Means Clusters

- Most common measure is Sum of Squared Error (SSE)

- For each point, the error is the distance to the nearest cluster
- To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^K \sum_{x \in C_i} dist^2(m_i, x)$$

- x is a data point in cluster C_i and m_i is the representative point for cluster C_i
 - ◆ can show that m_i corresponds to the center (mean) of the cluster
- Given two clusters, we can choose the one with the smallest error
- One easy way to reduce SSE is to increase K , the number of clusters
 - ◆ A good clustering with smaller K can have a lower SSE than a poor clustering with higher K



- The quality of the cluster assignments is determined by computing the sum of the squared error (SSE) after the centroids converge

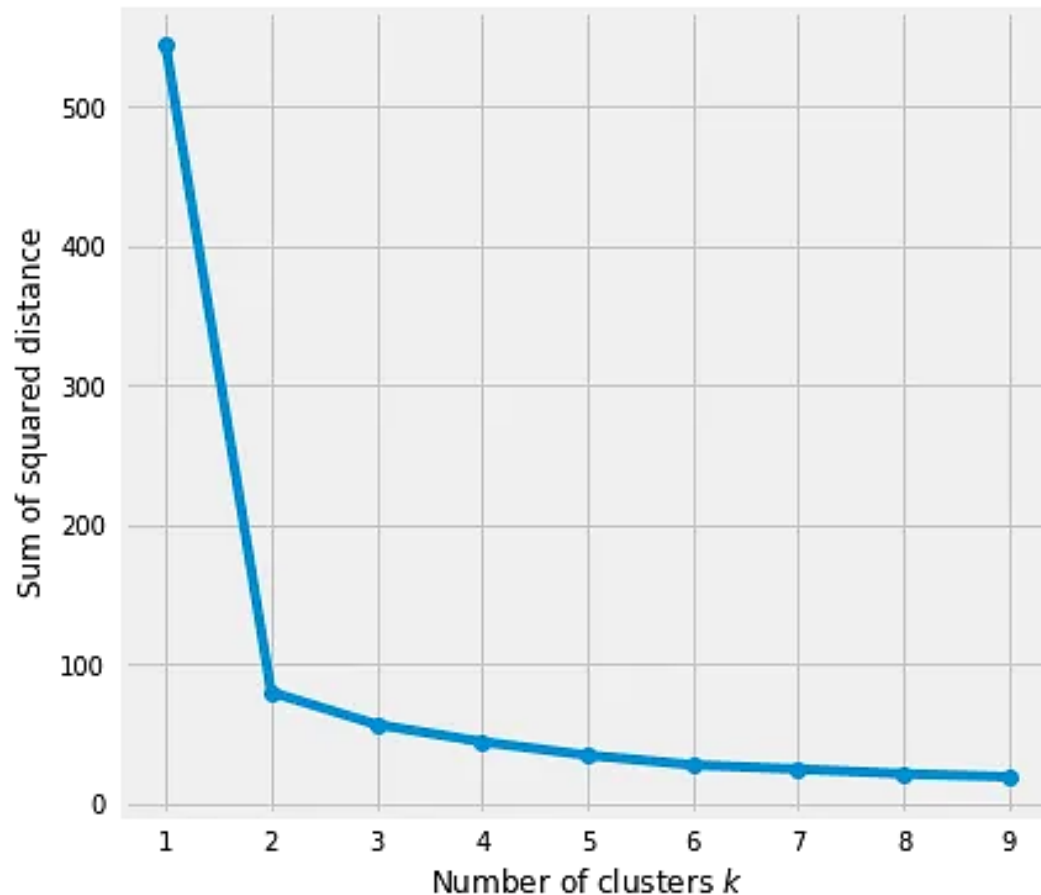
The diagram shows the formula for the sum of squared errors (SSE) with several annotations. The formula is $J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$. Annotations include: 'number of clusters' pointing to k , 'number of cases' pointing to n , 'case i ' pointing to $x_i^{(j)}$, 'centroid for cluster j ' pointing to c_j , and 'Distance function' pointing to the norm $\|x_i^{(j)} - c_j\|^2$. The entire expression is labeled 'objective function' with an arrow pointing to J .

$$\text{objective function } \leftarrow J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$$

Annotations in the diagram:

- number of clusters (points to k)
- number of cases (points to n)
- case i (points to $x_i^{(j)}$)
- centroid for cluster j (points to c_j)
- Distance function (points to $\|x_i^{(j)} - c_j\|^2$)

Evaluation: Elbow Method



Elbow method gives us an idea on what a good k number of clusters would be based on the sum of squared distance (SSE) between data points and their assigned clusters' centroids.

We pick k at the spot where SSE starts to flatten out and forming an elbow.

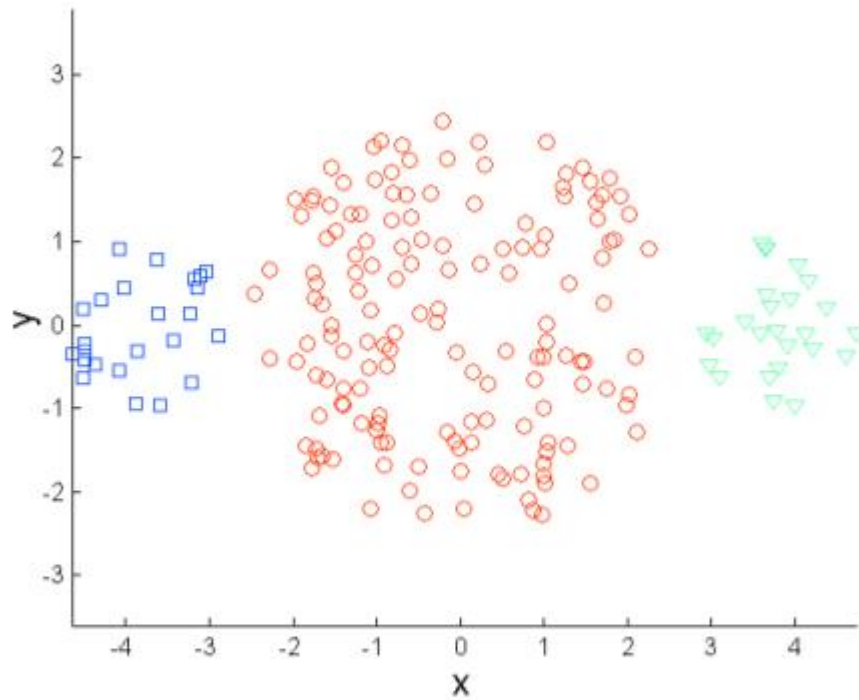


Strength and Weakness of K-Means Algorithm

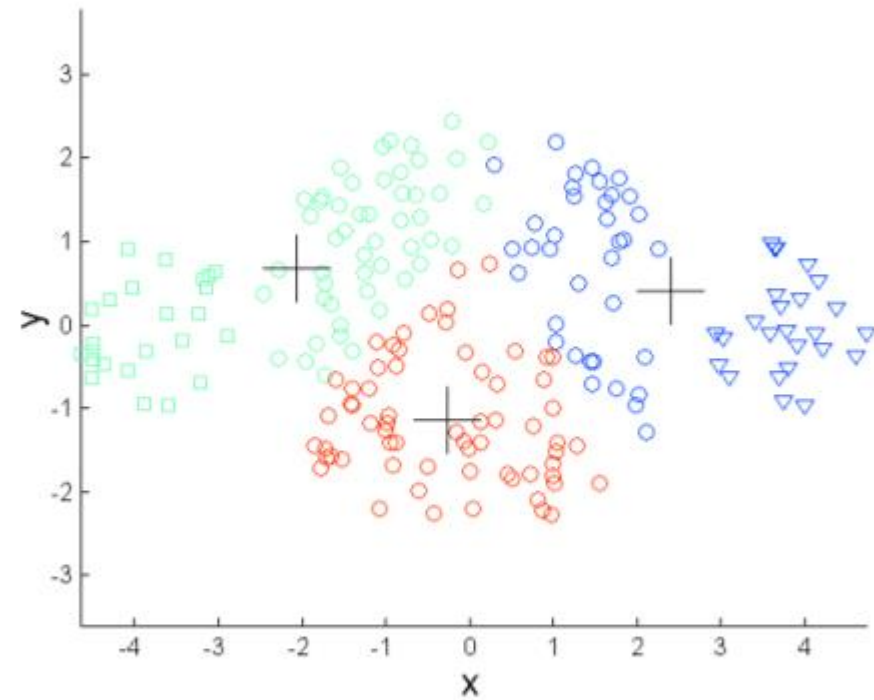
- Strength
 - Simple algorithm
 - Can handle wide variety of data
 - Quite efficient
- Limitations
 - K-means has problems when clusters are of differing
 - Sizes
 - Densities
 - Non-globular shapes
 - K-means has problems when the data contains outliers.



Limitations of K-means (Different Size)

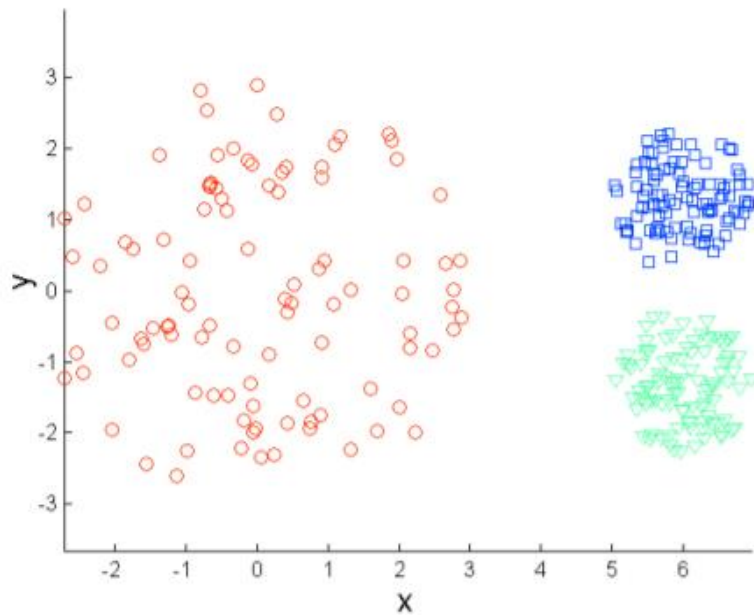


Original Points

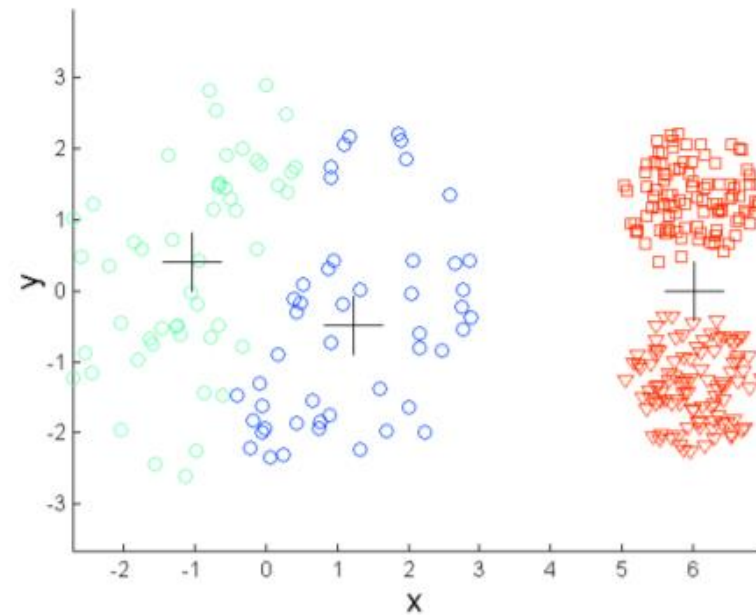


K-means (3 Clusters)

Limitations of K-means (Different Density)

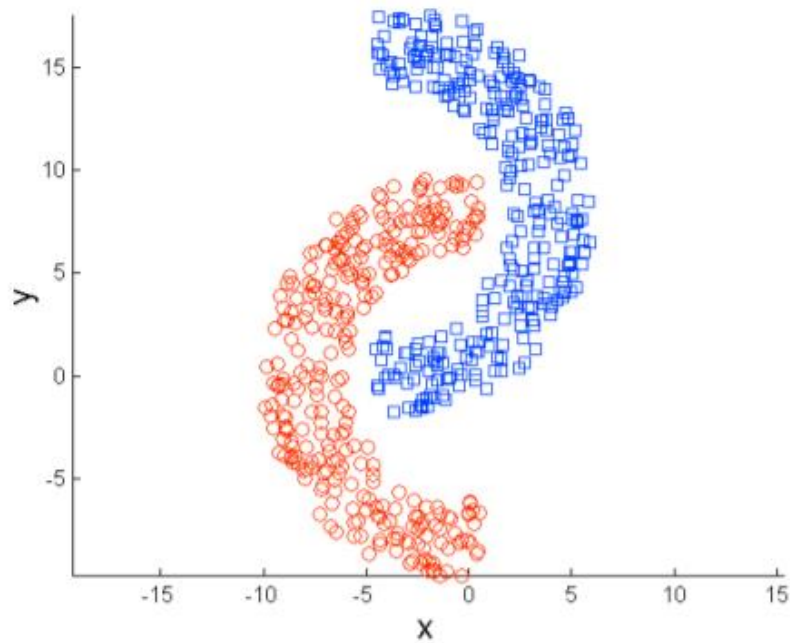


Original Points

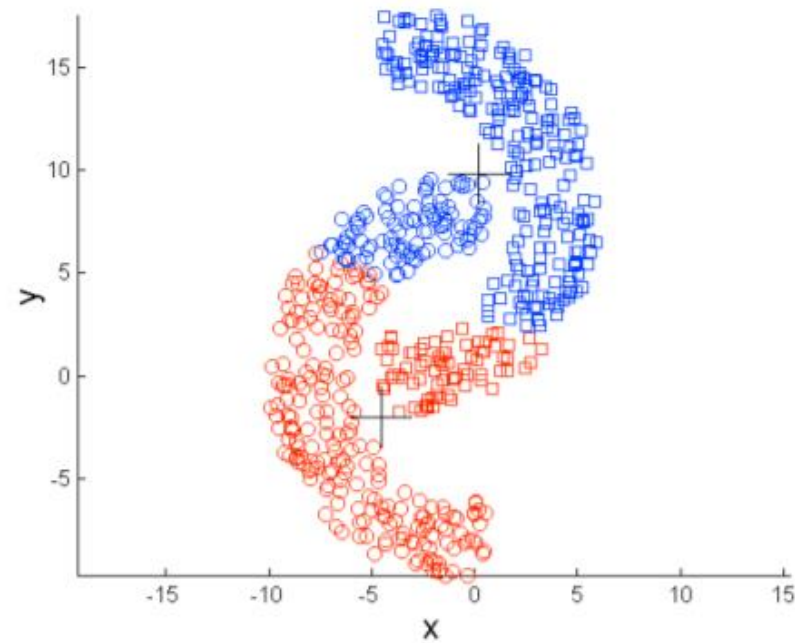


K-means (3 Clusters)

Limitations of K-means (Non Globular shape)



Original Points

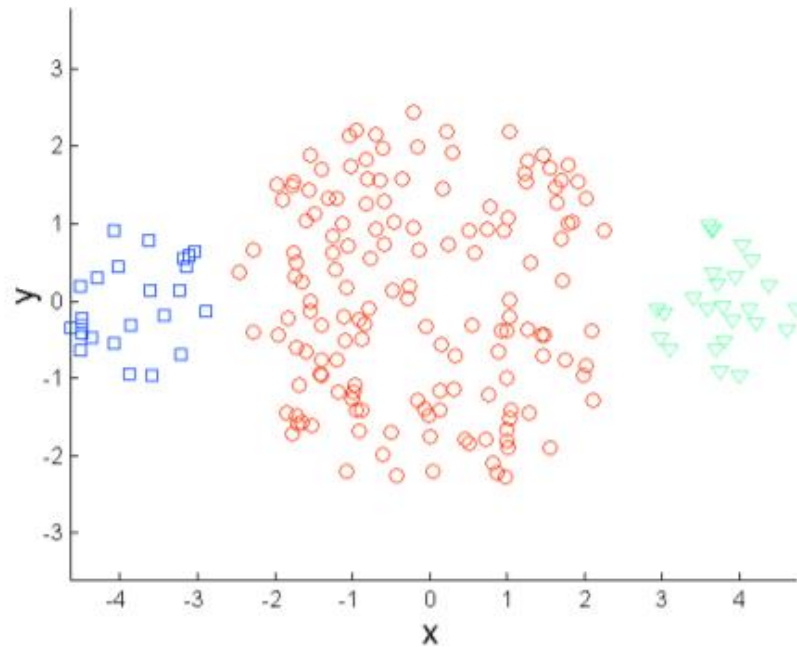


K-means (2 Clusters)

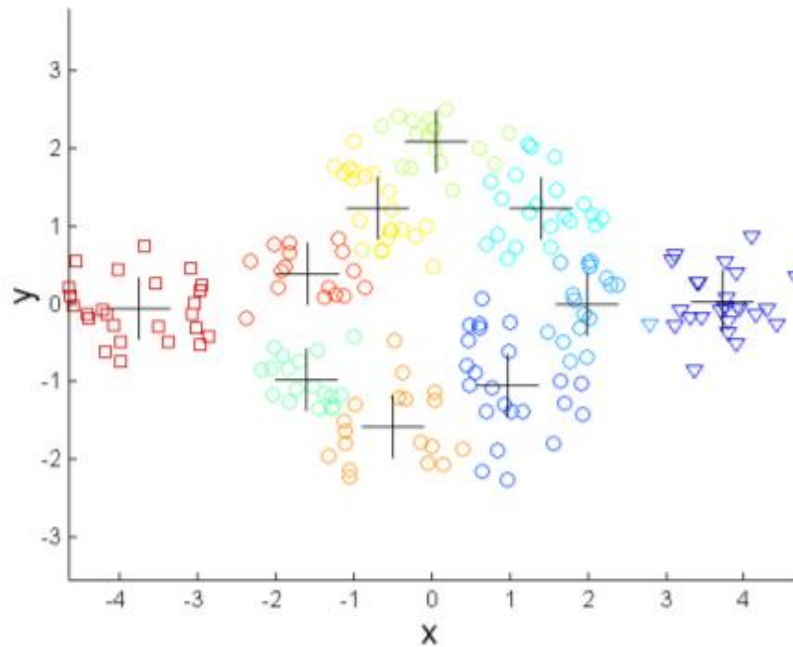


Solution

- One solution is to use many clusters. Find parts of clusters, but need to put together



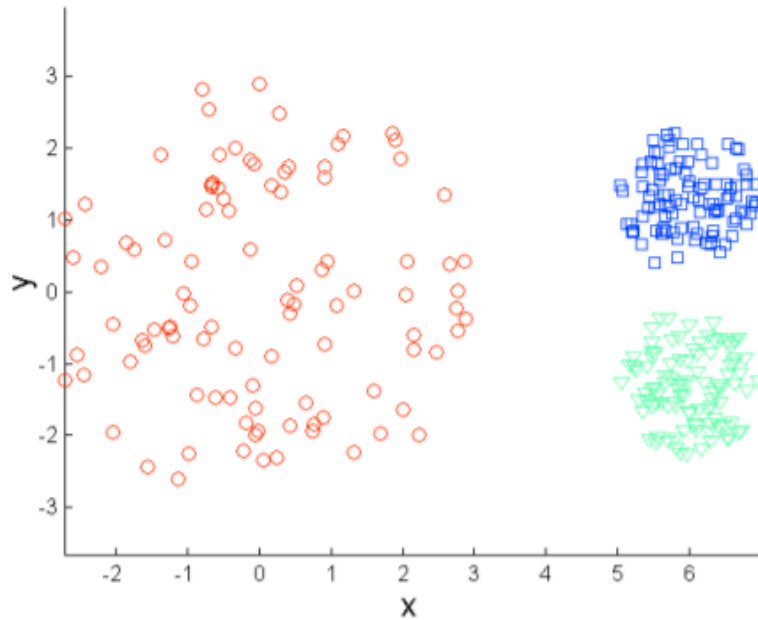
Original Points



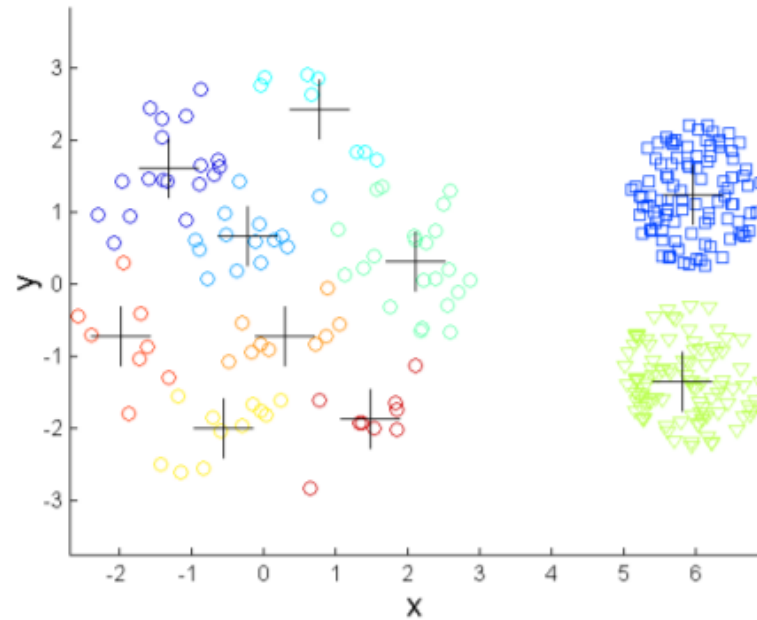
K-means Clusters

Solution

- One solution is to use many clusters. Find parts of clusters, but need to put together



Original Points

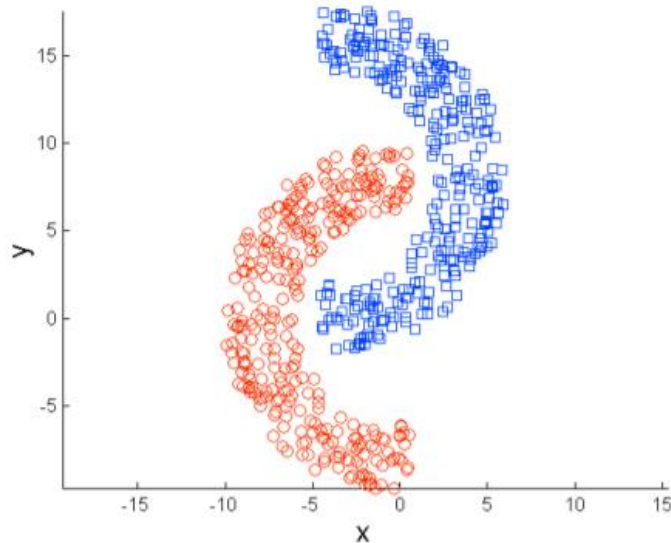


K-means Clusters

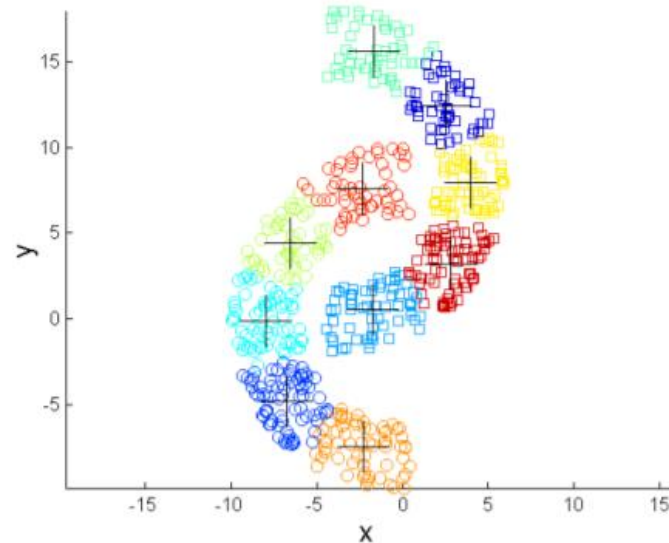


Solution

- One solution is to use many clusters. Find parts of clusters, but need to put together



Original Points



K-means Clusters



K-medoids Clustering

- Always data points are selected as medoids
- Less sensitive to outliers
- For K-Means, we were working with centroids.
 - The relationship between centroids and medoids is similar to the relationship between means and medians.
 - Medoids and medians will always be one of the observations in the data, while that's not necessarily the case for centroids and means.
- PAM: Partitioning around Medoids
- Homework

K-Medoid

- K-Medoids (also called as Partitioning Around Medoid) algorithm
- A medoid can be defined as the point in the cluster, whose dissimilarities with all the other points in the cluster is minimum.
- The dissimilarity of the medoid(C_i) and object(P_i) is calculated by using $E = |P_i - C_i|$
- *The cost in K-Medoids algorithm is given as -*

$$c = \sum_{C_i} \sum_{P_i \in C_i} |P_i - C_i|$$



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 point o 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.*



- Step 1: Let the randomly selected 2 medoids be C1 -(3, 4) and C2 -(7, 4).
- Step 2: The dissimilarity of each non-medoid point with the medoids is calculated and tabulated:

$$7-3 + 6-4 = 4+2=6$$

$$7-7 + 6-4 = 0+2=2$$

	X	Y	Dissimilarity From C1	Dissimilarity From C2
0	7	6	6	2
1	2	6	3	7
2	3	8	4	8
3	8	5	6	2
4	7	4	4	0
5	4	7	4	6
6	6	2	5	3
7	7	3	5	1
8	6	4	3	1
9	3	4	0	4



- Each point is assigned to the cluster of that medoid whose dissimilarity is less.
- The points 1, 2, 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2.
- The cost $C = (3 + 4 + 4) + (3 + 1 + 1 + 2 + 2) = 20$
- Step 3: Now randomly select one non-medoid point and recalculate the cost. Let the randomly selected point be (7, 3). The dissimilarity of each non-medoid point with the medoids – C1 (3, 4) and C2 (7, 3) is calculated and tabulated.



	X	Y	Dissimilarity From C1	Dissimilarity From C2
0	7	6	6	3
1	2	6	3	8
2	3	8	4	9
3	8	5	6	3
4	7	4	4	1
5	4	7	4	7
6	6	2	5	2
7	7	3	-	-
8	6	4	3	2
9	3	4	-	-

- Each point is assigned to that cluster whose dissimilarity is less. So, the points 1, 2, 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2.

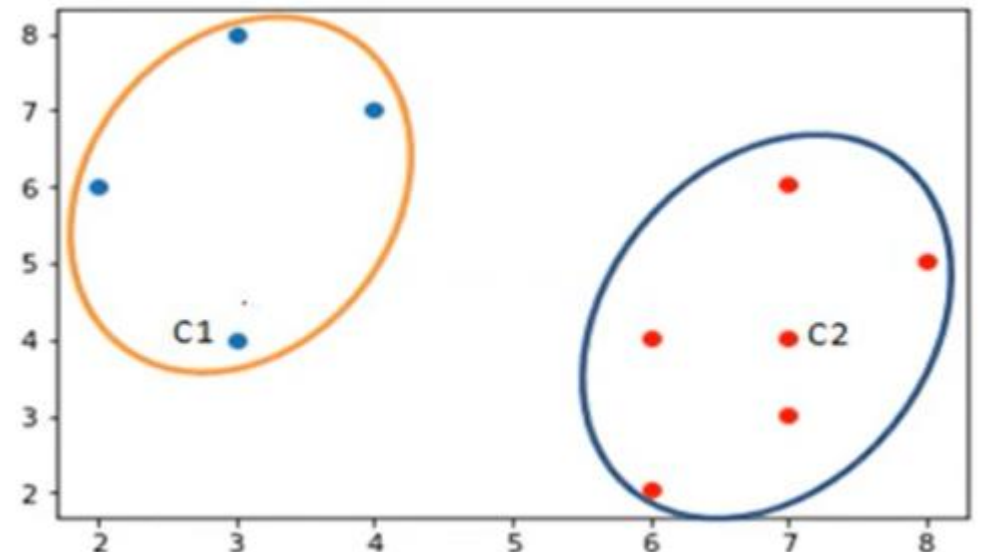
The cost $C = (3 + 4 + 4) + (2 + 2 + 1 + 3 + 3)$

$C = 22$

Swap Cost = Present Cost – Previous Cost

$= 22 - 20 = 2 > 0$

- As the swap cost is not less than zero, we undo the swap. Hence (3, 4) and (7, 4) are the final medoids. The clustering would be in the following way





Pre-processing and Post Processing

□ Pre-processing

- Normalize the data
- Eliminate outliers

□ Post-processing

- Eliminate small clusters that may represent outliers
- Split 'loose' clusters, i.e., clusters with relatively high SSE
- Merge clusters that are 'close' and that have relatively low SSE
- Can use these steps during the clustering process