K-MEANS

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into *K* distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible.

Algorithm for Basic K-Means algorithm

Select **K** points as initial centroids
    Repeat
        Form K clusters by assigning each point to its closest centroid.
        Recompute the centroid of each cluster
Until Centroids do not change

Example:

Cluster the following eight points (with (x, y) representing locations) into three clusters (let us say, we need 3 clusters):

A1(2, 10),
A2(2, 5),
A3(8, 4),
A4(5, 8),
A5(7, 5),
A6(6, 4),
A7(1, 2),
A8(4, 9)

Let, Initial cluster centers are:

A1(2, 10),
A4(5, 8) and
A7(1, 2).

The distance function between two points a = (x1, y1) and b = (x2, y2) is defined as-

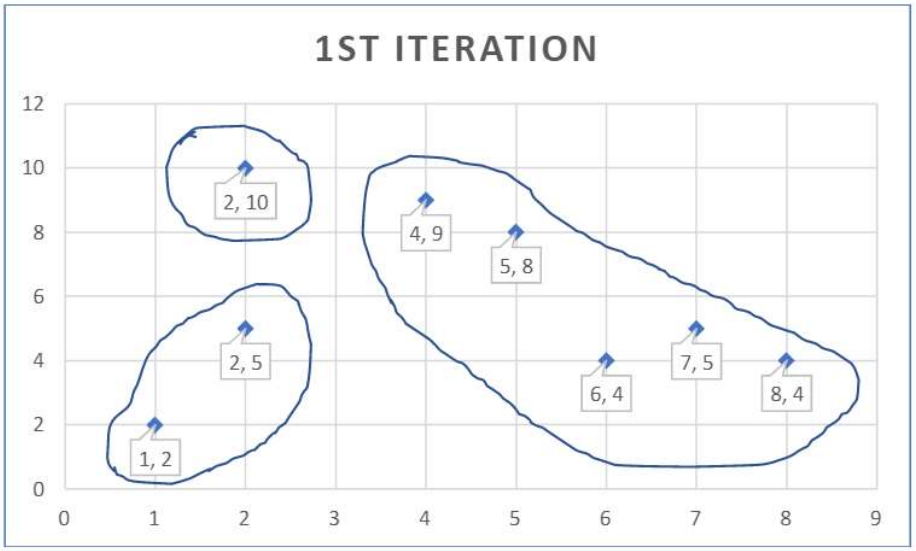
$$P(a, b) = |x2 - x1| + |y2 - y1|$$

Note: You can use any other distance formula like Euclidean distance too.

Now, calculate the distance from each point to the randomly picked centroids. And make the cluster to which the points are ***nearer.***

1st Iteration

Given Points	Distance from centre (2, 10) of Cluster-01	Distance from centre (5, 8) of Cluster-02	Distance from centre (1, 2) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	5	9	C1
A2(2, 5)	5	6	4	C3
A3(8, 4)	12	7	9	C2
A4(5, 8)	5	0	10	C2
A5(7, 5)	10	5	9	C2
A6(6, 4)	10	5	7	C2
A7(1, 2)	9	10	0	C3
A8(4, 9)	3	2	10	C2



For Cluster-01:
We have only one point A1**(2, 10)** in Cluster-01.
So, cluster center remains the same.

For Cluster-02:
Centre of Cluster-02
= ((8 + 5 + 7 + 6 + 4)/5, (4 + 8 + 5 + 4 + 9)/5)
= (6, 6)

For Cluster-03:
Centre of Cluster-03
= ((2 + 1)/2, (5 + 2)/2)
= (1.5, 3.5)

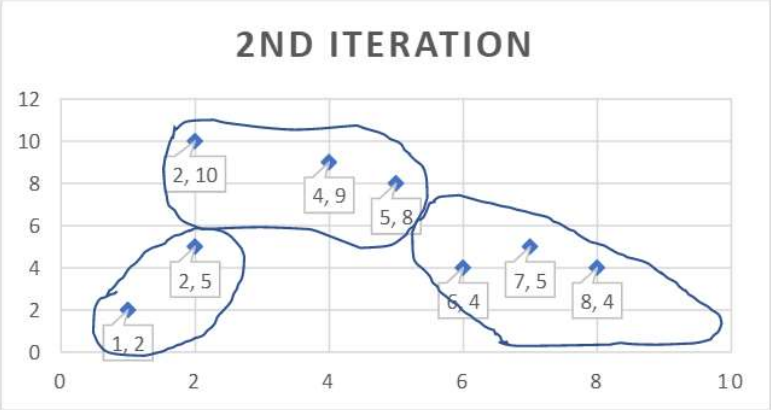
This is completion of Iteration-01.

2nd Iteration

- We calculate the distance of each point from each of the center of the three clusters **(2, 10), (6, 6), (1.5, 3.5)**
- The distance is calculated by using the given distance function.
-

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (6, 6) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	8	7	C1
A2(2, 5)	5	5	2	C3
A3(8, 4)	12	4	7	C2
A4(5, 8)	5	3	8	C2
A5(7, 5)	10	2	7	C2
A6(6, 4)	10	2	5	C2
A7(1, 2)	9	9	2	C3
A8(4, 9)	3	5	8	C1

Note that A8(4,9) has jumped from cluster c2 to cluster c1. So, we should do another iteration.



For Cluster-01:
Center of Cluster-01
= ((2 + 4)/2, (10 + 9)/2) = **(3, 9.5)**

For Cluster-02:
Center of Cluster-02
= ((8 + 5 + 7 + 6)/4, (4 + 8 + 5 + 4)/4) = **(6.5, 5.25)**

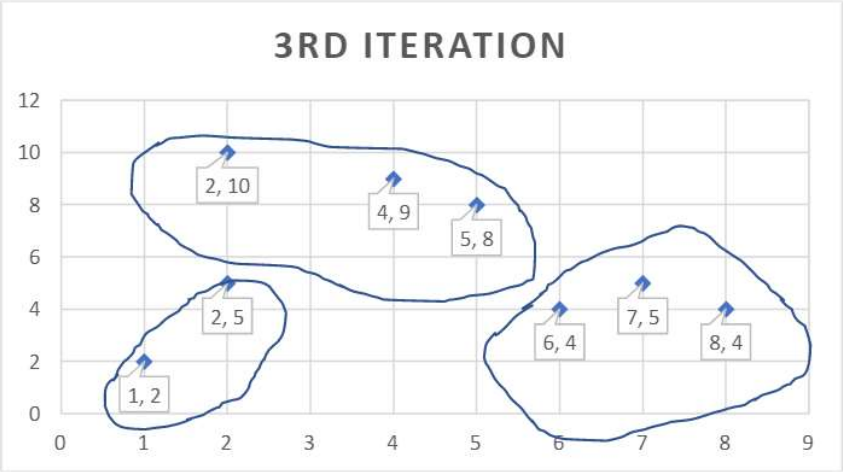
For Cluster-03:
Center of Cluster-03
= ((2 + 1)/2, (5 + 2)/2) = **(1.5, 3.5)**
This is completion of Iteration-02.

3rd iteration,

- We calculate the distance of each point from each of the centre of the three clusters **(3, 9.5), (6.5, 5.25), (1.5, 3.5)**
- The distance is calculated by using the given distance function.

Given Points	Distance from center (3, 9.5) of Cluster-01	Distance from center (6.5, 5.25) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	1.5	9.25	7	C1
A2(2, 5)	5.5	4.75	2	C3
A3(8, 4)	10.5	2.75	7	C2
A4(5, 8)	3.5	4.25	8	C1
A5(7, 5)	8.5	0.75	7	C2
A6(6, 4)	8.5	1.75	5	C2
A7(1, 2)	9.5	8.75	2	C3
A8(4, 9)	1.5	6.25	8	C1

Note that cluster A4(5, 8) has jumped from C2 to C1. So, we should do another iteration.



Repeat the process until no element jumps from one cluster to another cluster,

Strengths, Weaknesses, Time and space complexity of K-Means

Strengths

1. It is simple and useful for all varieties of data types.

2. Even though multiple iterations are performed, it is quite efficient.
3. Bisecting k- means is also efficient.

Weakness

1. It cannot handle non-globular clusters, clusters of different sizes and densities even though it can find pure sub-clusters.
2. Outliers cannot be clustered.
3. K-means is restricted for the notion of centroid.

Time complexity

O(I*K*m*n)

Where,

I- Number of Iteration for making perfect clusters
K- Number of clusters
m- Number of attributes
n- Number of elements

Space complexity

O((m+K)n)