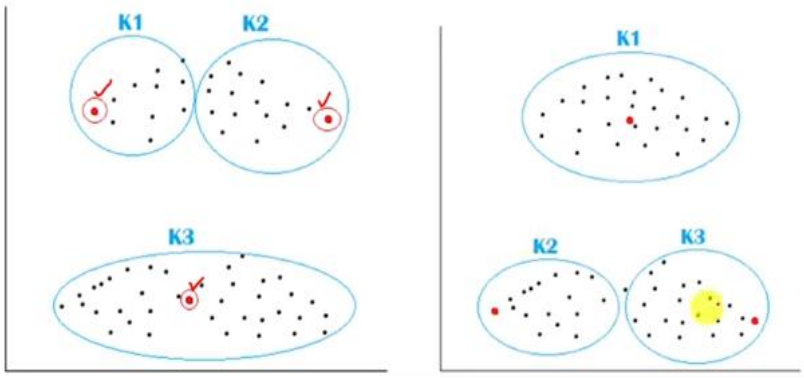
Reason for using KMean++ Algorithm

- The K-means++ algorithm is used to improve the **initialization phase(**selects initial centroids that are more likely to be spread out in the data space) of the K-means clustering algorithm.
- Traditional K-means can suffer from **poor cluster quality** and **slow convergence** if the initial centroids are not chosen carefully.
- K-means is **highly sensitive to the initial positions of the centroids**, which can result in the algorithm converging to suboptimal solutions (local minima).
  - Also, we need to select the initial centroids for each of the clusters.



K-means++ addresses these issues by providing a smarter way to initialize the centroids.

KMean++ Algorithm Basics (It is a probabilistic method)

- **Select the First Centroid:**
  - Choose one data point randomly from the dataset as the first centroid.
- **Calculate Distances:**
  - For each data point that has not been selected as a centroid, calculate the distance (usually using Euclidean distance) from that point to the nearest centroid that has already been chosen.
- **Select Subsequent Centroids:**
  - The next centroid is chosen from the remaining data points with a probability proportional to the square of the distance from the nearest existing centroid. This ensures that points that are far from existing centroids are more likely to be selected as new centroids.
- **Repeat Step 2 and 3:**
  - Repeat the distance calculation and centroid selection process until the desired number of centroids is reached.

Example: K-means++ Initialization: Step-by-Step Explanation

Let's assume we have the following 8 set of 2D points:

(2,3),(3,7),(4,2),(6,5),(8,3),(8,6),(9,1),(10,4)

and we want to cluster these points into k=3 clusters.

Step 1: Randomly select the first centroid.
Assume the first centroid C1 is chosen randomly as (4,2).

Step 2: Compute the distance D(x) from each point x to the nearest centroid that has already been chosen.

We compute the distances from each point to (4,2).

$$D((2,3)) = \sqrt{(2-4)^2 + (3-2)^2} = \sqrt{4+1} = \sqrt{5} \approx 2.24$$

$$D((3,7)) = \sqrt{(3-4)^2 + (7-2)^2} = \sqrt{1+25} = \sqrt{26} \approx 5.10$$

$$D((4,2)) = 0$$

$$D((6,5)) = \sqrt{(6-4)^2 + (5-2)^2} = \sqrt{4+9} = \sqrt{13} \approx 3.61$$

$$D((8,3)) = \sqrt{(8-4)^2 + (3-2)^2} = \sqrt{16+1} = \sqrt{17} \approx 4.12$$

$$D((8,6)) = \sqrt{(8-4)^2 + (6-2)^2} = \sqrt{16+16} = \sqrt{32} \approx 5.66$$

$$D((9,1)) = \sqrt{(9-4)^2 + (1-2)^2} = \sqrt{25+1} = \sqrt{26} \approx 5.10$$

$$D((10,4)) = \sqrt{(10-4)^2 + (4-2)^2} = \sqrt{36+4} = \sqrt{40} \approx 6.32$$

Step 3: Select the next centroid using a weighted probability distribution.

Calculate the probability $P(x)$ for each point x to be chosen as the next centroid. This is done by dividing each $D(x)^2$ by the sum of all $D(x)^2$.

$$\sum D(x)^2 = (2.24)^2 + (5.10)^2 + 0 + (3.61)^2 + (4.12)^2 + (5.66)^2 + (5.10)^2 + (6.32)^2$$

$$\sum D(x)^2 = 5 + 26 + 0 + 13 + 17 + 32 + 26 + 40 = 159$$

$$P((2,3)) = \frac{(2.24)^2}{159} = \frac{5}{159} \approx 0.0314$$

$$P((3,7)) = \frac{(5.10)^2}{159} = \frac{26}{159} \approx 0.1635$$

$$P((4,2)) = 0$$

$$P((6,5)) = \frac{(3.61)^2}{159} = \frac{13}{159} \approx 0.0818$$

$$P((8,3)) = \frac{(4.12)^2}{159} = \frac{17}{159} \approx 0.1069$$

$$P((8,6)) = \frac{(5.66)^2}{159} = \frac{32}{159} \approx 0.2013$$

$$P((9,1)) = \frac{(5.10)^2}{159} = \frac{26}{159} \approx 0.1635$$

$$P((10,4)) = \frac{(6.32)^2}{159} = \frac{40}{159} \approx 0.2516$$

Step 4: Repeat steps 2 and 3 to choose the remaining centroids.

Compute the distances from each point to the nearest of the centroids $(4, 2)$ and $(10, 4)$.

$$D((2,3)) = \min(\sqrt{5}, \sqrt{61}) = \sqrt{5} \approx 2.24$$

$$D((3,7)) = \min(\sqrt{26}, \sqrt{29}) = \sqrt{26} \approx 5.10$$

$$D((4,2)) = 0$$

$$D((6,5)) = \min(\sqrt{13}, \sqrt{16}) = \sqrt{13} \approx 3.61$$

$$D((8,3)) = \min(\sqrt{17}, \sqrt{4}) = \sqrt{4} = 2$$

$$D((8,6)) = \min(\sqrt{32}, \sqrt{4}) = \sqrt{4} = 2$$

$$D((9,1)) = \min(\sqrt{26}, \sqrt{10}) = \sqrt{10} \approx 3.16$$

$$D((10,4)) = 0$$

Update the probability $P(x)$ for each point x to be chosen as the next centroid.

$$\sum D(x)^2 = (2.24)^2 + (5.10)^2 + 0 + (3.61)^2 + 2^2 + 2^2 + (3.16)^2 + 0$$

$$\sum D(x)^2 = 5 + 26 + 0 + 13 + 4 + 4 + 10 = 62$$

$$P((2,3)) = \frac{5}{62} \approx 0.0806$$

$$P((3,7)) = \frac{26}{62} \approx 0.4194$$

$$P((4,2)) = 0$$

$$P((6,5)) = \frac{13}{62} \approx 0.2097$$

$$P((8,3)) = \frac{4}{62} \approx 0.0645$$

$$P((8,6)) = \frac{4}{62} \approx 0.0645$$

$$P((9,1)) = \frac{10}{62} \approx 0.1613$$

$$P((9,1)) = \frac{10}{62} \approx 0.1613$$

$$P((10,4)) = 0$$

Assume the next centroid c3 is chosen to be (3,7).

Disadvantage of KMean++ algorithm:

- **Still Requires Specification of k**:
  - K-means++ does not eliminate the need to specify the number of clusters (k) in advance, which can be challenging in practice.

- **Sensitivity to Outliers**:
  - Both K-means and K-means++ are sensitive to outliers and noise. Outliers can significantly affect the placement of centroids, leading to suboptimal clustering.
- **Scalability**:
  - While K-means++ can converge faster than standard K-means due to better initial centroids, it can still be computationally intensive for very large datasets, especially in the initialization phase.
- **Assumption of Spherical Clusters**:
  - K-means++, like K-means, assumes that clusters are spherical and evenly sized. This can be a limitation when dealing with data that have clusters of different shapes and densities.
- **Complexity of Initialization**:
  - The initialization step in K-means++ is more complex and computationally expensive than the random initialization in standard K-means. For each new centroid, distances from all points to the existing centroids must be computed and probabilities updated, which can be slow for large datasets.
- **Local Optima**:
  - Although K-means++ provides a better starting point than random initialization, it can still converge to a local optimum. It doesn't guarantee a global optimum solution.

K-median clustering algorithm

- The K-Median algorithm is a clustering method that aims to partition a set of data points into k clusters by minimizing the sum of the distances from each point to the median of its assigned cluster.
- Unlike the K-Means algorithm, which minimizes the sum of squared distances, K-Median uses the median, making it more robust to outliers.

K-Median Algorithm Steps

Step1: **Initialization:** Select k initial medians randomly from the data points.

Step 2: **Assignment:** Assign each data point to the nearest median based on a chosen distance metric (usually Manhattan distance).

Step 3: **Update:** Calculate the new median for each cluster by finding the median of the points in the cluster.

Step4: **Repeat:** Repeat the assignment and update steps until the medians no longer change or change minimally.

Numerical Example

Suppose we have the following data points and we want to cluster them into k=2 clusters.

Data Points: {(2,3),(3,3),(6,7),(8,8),(3,6),(5,8),(9,10)}

Objective: Cluster the data points into k=2 clusters.

Step 1: Initialization
Randomly select 2 initial medians. Let's say we pick (2,3) and (8,8).

Step 2: Assignment
Calculate Manhattan distances and assign each point to the nearest median:

Distance from (2, 3) to (2, 3) =0

Distance from (2, 3) to (8, 8) =$|2-8|+|3-8|=6+5=11$

Distance from (3, 3) to (2, 3) =$|3-2|+|3-3|=1$

Distance from (3, 3) to (8, 8) =$|3-8|+|3-8|=5+5=10$

Distance from (6, 7) to (2, 3) $=|6-2|+|7-3|=4+4=8$

Distance from (6, 7) to (8, 8) $=|6-8|+|7-8|=2+1=3$

Distance from (8, 8) to (2, 3) $=|8-2|+|8-3|=6+5=11$

Distance from (8, 8) to (8, 8) $=0$

Distance from (3, 6) to (2, 3) $=|3-2|+|6-3|=1+3=4$

Distance from (3, 6) to (8, 8) $=|3-8|+|6-8|=5+2=7$

Distance from (5, 8) to (2, 3) $=|5-2|+|8-3|=3+5=8$

Distance from (5, 8) to (8, 8) $=|5-8|+|8-8|=3+0=3$

Distance from (9, 10) to (2, 3) $=|9-2|+|10-3|=7+7=14$

Distance from (9, 10) to (8, 8) $=|9-8|+|10-8|=1+2=3$

Based on the distances, we assign the points to the nearest median:

Cluster 1: (Median (2, 3)): (2, 3), (3, 3), (3, 6)

Cluster 2: (Median (8, 8)): (6, 7), (8, 8), (5, 8), (9, 10)

Step 3: Update

Calculate the new median for each cluster:

For Cluster 1: (2, 3), (3, 3), (3, 6)

- Median x-coordinate: median(2, 3, 3) = 3
- Median y-coordinate: median(3, 3, 6) = 3
- New Median: (3, 3)

For Cluster 2: (6, 7), (8, 8), (5, 8), (9, 10)

- Median x-coordinate: median(5, 6, 8, 9) = 7
- Median y-coordinate: median(7, 8, 8, 10) = 8
- New Median: (7, 8)

Step 4: Repeat

- Reassign the points to the nearest new medians:

Distance from (2, 3) to (3, 3))=|2−3|+|3−3|=1

Distance from (2, 3) to (7, 8) =|2−7|+|3−8|=5+5=10

Distance from (3, 3) to (3, 3) =0

Distance from (3, 3) to (7, 8) =|3−7|+|3−8|=4+5=9

Distance from (6, 7) to (3, 3) =|6−3|+|7−3|=3+4=7

Distance from (6, 7) to (7, 8) =|6−7|+|7−8|=1+1=2

Distance from (8, 8) to (3, 3) =|8−3|+|8−3|=5+5=10

Distance from (8, 8) to (7, 8) =|8−7|+|8−8|=1

Distance from (3, 6) to (3, 3) =|3−3|+|6−3|=0+3=3

Distance from (3, 6) to (7, 8) =|3−7|+|6−8|=4+2=6

Distance from (5, 8) to (3, 3) =|5−3|+|8−3|=2+5=7

Distance from (5, 8) to (7, 8) =|5−7|+|8−8|=2+0=2

Distance from (9, 10) to (3, 3) =|9−3|+|10−3|=6+7=13

Distance from (9, 10) to (7, 8) =|9−7|+|10−8|=2+2=4

New assignment based on distances:

- Cluster 1 (Median (3, 3)): (2, 3), (3, 3), (3, 6)
- Cluster 2 (Median (7, 8)): (6, 7), (8, 8), (5, 8), (9, 10)

Update the Medians

For Cluster 1: (2, 3), (3, 3), (3, 6)

- Median x-coordinate: median(2, 3, 3) = 3
- Median y-coordinate: median(3, 3, 6) = 3
- New Median: (3, 3)

For Cluster 2: (6, 7), (8, 8), (5, 8), (9, 10)

- Median x-coordinate: median(5, 6, 8, 9) = 7
- Median y-coordinate: median(7, 8, 8, 10) = 8
- New Median: (7, 8)

Since the medians did not change, the algorithm stops.

Final Clusters:

Cluster 1: (2, 3), (3, 3), (3, 6)

Cluster 2: (6, 7), (8, 8), (5, 8), (9, 10)

Qus. Divide the given (1,2), (2, 4), (3, 6), (8, 8), (9, 10) 2D-points into K=2 clusters using K-Median clustering algorithm.

Ans. **Step 1: Initialization**
Randomly select 2 initial medians. Let's say we pick (2, 4) and (8, 8).

Step 2: Assignment
Calculate Manhattan distances and assign each point to the nearest median:

Distance from (1, 2) to (2, 4) $=|1-2|+|2-4|=1+2=3$

Distance from (1, 2) to (8, 8) $=|1-8|+|2-8|=7+6=13$

Distance from (3, 6) to (2, 4) $=|3-2|+|6-4|=1+2=3$

Distance from (3, 6) to (8, 8) $=|3-8|+|6-8|=5+2=7$

Distance from (9, 10) to (2, 4) $=|9-2|+|10-4|=7+6=13$

Distance from (9, 10) to (8, 8) $=|9-8|+|10-8|=1+2=3$

Step 3: Update

Calculate the new median for each cluster:

For Cluster 1: (1, 2), (2, 4), (3, 6)

- Median x-coordinate: median(1, 2, 3) = 2
- Median y-coordinate: median(2, 4, 6) = 4
- New Median**: (2, 4)**

For Cluster 2: (8, 8), (9, 10)

- Median x-coordinate: median(8, 9) = 8.5 = 8
- Median y-coordinate: median(8, 10) = 9
- New Median: **(8, 9)**

Step 4: Repeat

- Reassign the points to the nearest new medians based on the minimum distance.

Distance of (1, 2) to (2, 4))=|1−2|+|2−4|=1+2=3

Distance of (1, 2) to (8, 9) =|1−8|+|2−9|=7+7=14

Distance of (3, 6) to (2, 4) =|3−2|+|6−4|=1+2=3

Distance of (3, 6) to (8, 9) =|3−8|+|6−9|=5+3=8

Distance of (9, 10) to (2, 4) =|9−2|+|10−4|=7+6=13

Distance of (9, 10) to (8, 9) =|9−8|+|10−9|=1+1=2

Update the median

Calculate the new median for each cluster:

For Cluster 1: (1, 2), (2, 4), (3, 6)

- Median x-coordinate: median(1, 2, 3) = 2
- Median y-coordinate: median(2, 4, 6) = 4
- New Median**: (2, 4)**

For Cluster 2: (8, 8), (9, 10)

- Median x-coordinate: median(8, 9) = 8.5 = 8
- Median y-coordinate: median(8, 10) = 9
- New Median: **(8, 9)**

Since the medians did not change, the algorithm stops.

So, the Final Clusters:

Cluster 1: (1, 2), (2, 4), (3, 6)

Cluster 2: (8, 8), (9, 10)

K-medoid (Partitioned based clustering):

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

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

Algorithm step:

1. First, we select K random data points from the dataset and use them as medoids.
2. Now, we will calculate the distance of each data point from the medoids. You can use any of the Euclidean, **Manhattan distance**, or squared Euclidean distance as the distance measure.
3. Once we find the distance of each data point from the medoids, we will assign the data points to the clusters associated with each medoid. The data points are assigned to the medoids at the closest distance.
4. After determining the clusters, we will calculate the sum of the distance of all the non-medoid data points to the medoid of each cluster. Let the cost be Ci.
5. Now, we will select a random data point Dj from the dataset and swap it with a medoid Mi. Here, Dj becomes a temporary medoid. After swapping, we will calculate the distance of all the non-medoid data points to the current medoid of each cluster. Let this cost be Cj.
6. If Ci>Cj, the current medoids with Dj as one of the medoids are made permanent medoids. Otherwise, we undo the swap, and Mi is reinstated as the medoid.
7. Repeat 4 to 6 until no change occurs in the clusters.

The dataset for clustering is as follows.

Point	Coordinates
A1	(2, 6)
A2	(3, 8)
A3	(4, 7)
A4	(6, 2)
A5	(6, 4)
A6	(7, 3)
A7	(7,4)
A8	(8, 5)
A9	(7, 6)
A10	(3, 4)

Iteration 1

Suppose that we want to group the above dataset into two clusters. So, we will randomly choose two medoids.

Here, the choice of medoids is important for efficient execution. Hence, we have selected two points from the dataset that can be potential medoid for the final clusters. Following are two points from the dataset that we have selected as medoids.

M1 = (3, 4)

M2 = (7, 3)

Now, we will calculate the distance between each data point and the medoids using the Manhattan distance measure. The results have been tabulated as follows.

Point	Coordinates	Distance From M1 (3,4)	Distance from M2 (7,3)	Assigned Cluster
A1	(2, 6)	3	8	Cluster 1
A2	(3, 8)	4	9	Cluster 1
A3	(4, 7)	4	7	Cluster 1
A4	(6, 2)	5	2	Cluster 2
A5	(6, 4)	3	2	Cluster 2
A6	(7, 3)	5	0	Cluster 2
A7	(7,4)	4	1	Cluster 2
A8	(8, 5)	6	3	Cluster 2
A9	(7, 6)	6	3	Cluster 2
A10	(3, 4)	0	5	Cluster 1

The clusters made with medoids (3, 4) and (7, 3) are as follows.
Points in cluster1= {(2, 6), (3, 8), (4, 7), (3, 4)}
Points in cluster 2= {(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}

After assigning clusters, we will calculate the cost for each cluster and find their sum. The cost is nothing but the sum of distances of all the data points from the medoid of the cluster they belong to.
Hence, the cost for the current cluster will be 3+4+4+2+2+0+1+3+3+0=22.

Iteration 2

Now, we will select another non-medoid point (7, 4) and make it a temporary medoid for the second cluster but fix the medoid (3,4) of first cluster for simplicity. We can also change it but we don't change it, if we change the medoid of first cluster then the problem will be lengthier. Hence,
M1 = (3, 4), M2 = (7, 4)

Now, let us calculate the distance between all the data points and the current medoids.

Point	Coordinates	Distance From M1 (3,4)	Distance from M2 (7,4)	Assigned Cluster
A1	(2, 6)	3	7	Cluster 1
A2	(3, 8)	4	8	Cluster 1
A3	(4, 7)	4	6	Cluster 1
A4	(6, 2)	5	3	Cluster 2
A5	(6, 4)	3	1	Cluster 2
A6	(7, 3)	5	1	Cluster 2
A7	(7,4)	4	0	Cluster 2
A8	(8, 5)	6	2	Cluster 2
A9	(7, 6)	6	2	Cluster 2
A10	(3, 4)	0	4	Cluster 1

The data points haven't changed in the clusters after changing the medoids. Hence, clusters are:

Points in cluster1:{(2, 6), (3, 8), (4, 7), (3, 4)}
Points in cluster 2:{(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}
Now, let us again calculate the cost for each cluster and find their sum. The total cost this time will be 3+4+4+3+1+1+0+2+2+0=20.

Here, the current cost is less than the cost calculated in the previous iteration. Hence, we will make the swap permanent and make (7,4) the medoid for cluster 2. If the cost this time was greater than the previous cost i.e. 22, we would have to revert the change. New medoids after this iteration are (3, 4) and (7, 4) with no change in the clusters.

Iteration 3

Now, let us again change the medoid of cluster 2 to (6, 4). Hence, the new medoids for the clusters are M1=(3, 4) and M2= (6, 4).

Let us calculate the distance between the data points and the above medoids to find the new cluster. The results have been tabulated as follows.

Point	Coordinates	Distance From M1 (3,4)	Distance from M2 (6,4)	Assigned Cluster
A1	(2, 6)	3	6	Cluster 1
A2	(3, 8)	4	7	Cluster 1
A3	(4, 7)	4	5	Cluster 1
A4	(6, 2)	5	2	Cluster 2
A5	(6, 4)	3	0	Cluster 2
A6	(7, 3)	5	2	Cluster 2
A7	(7,4)	4	1	Cluster 2
A8	(8, 5)	6	3	Cluster 2
A9	(7, 6)	6	3	Cluster 2
A10	(3, 4)	0	3	Cluster 1

Iteration 3

Again, the clusters haven't changed. Hence, clusters are:

Points in cluster1:{(2, 6), (3, 8), (4, 7), (3, 4)}
Points in cluster 2:{(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}
Now, let us again calculate the cost for each cluster and find their sum. The total cost this time will be 3+4+4+2+0+2+1+3+3+0=22.

The current cost is 22 which is greater than the cost in the previous iteration i.e. 20. Hence, we will revert the change and the point (7, 4) will again be made the medoid for cluster 2.

So, the clusters after this iteration will be cluster1 = {(2, 6), (3, 8), (4, 7), (3, 4)} and cluster 2= {(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}. The medoids are (3,4) and (7,4).

We keep replacing the medoids with a non-medoid data point. The set of medoids for which the cost is the least, the medoids, and the associated clusters are made permanent. So, after all the iterations, you will get the final clusters and their medoids.

The K-Medoids clustering algorithm is a computation-intensive algorithm that requires many iterations. In each iteration, we need to calculate the distance between the medoids and the data points, assign clusters, and compute the cost. Hence, K-Medoids clustering is not well suited for large data sets.

Difference between K-mean and K-medoid

S No	Factor	K-mean	K-medoid
1	Objective	Minimizing the sum of squared distances between data points and their assigned cluster centroids.	Minimizing the sum of dissimilarities between data points and their assigned cluster medoids.
2	Cluster Center Metric	Use centroids, which are the arithmetic means of all data points in a cluster.	Use medoids, which are representative data points within each cluster that are most centrally located concerning all other data points in the cluster.
3	Robustness	Less robust to noise and outliers.	More robust to noise and outliers.
4	Computational Complexity	Faster and more efficient for large datasets.	Slower and less efficient for large datasets.
5	Cluster Shape	Assumes spherical clusters and is not suitable for non-convex clusters	Can handle non-convex clusters
6	Initialization	Requires initial centroids to be randomly selected.	Requires initial medoids to be randomly selected.
7	Applications	Suitable for applications such as customer segmentation, image segmentation, and anomaly detection.	Suitable for applications where robustness to noise and outliers is important, such as clustering DNA sequences or gene expression data.