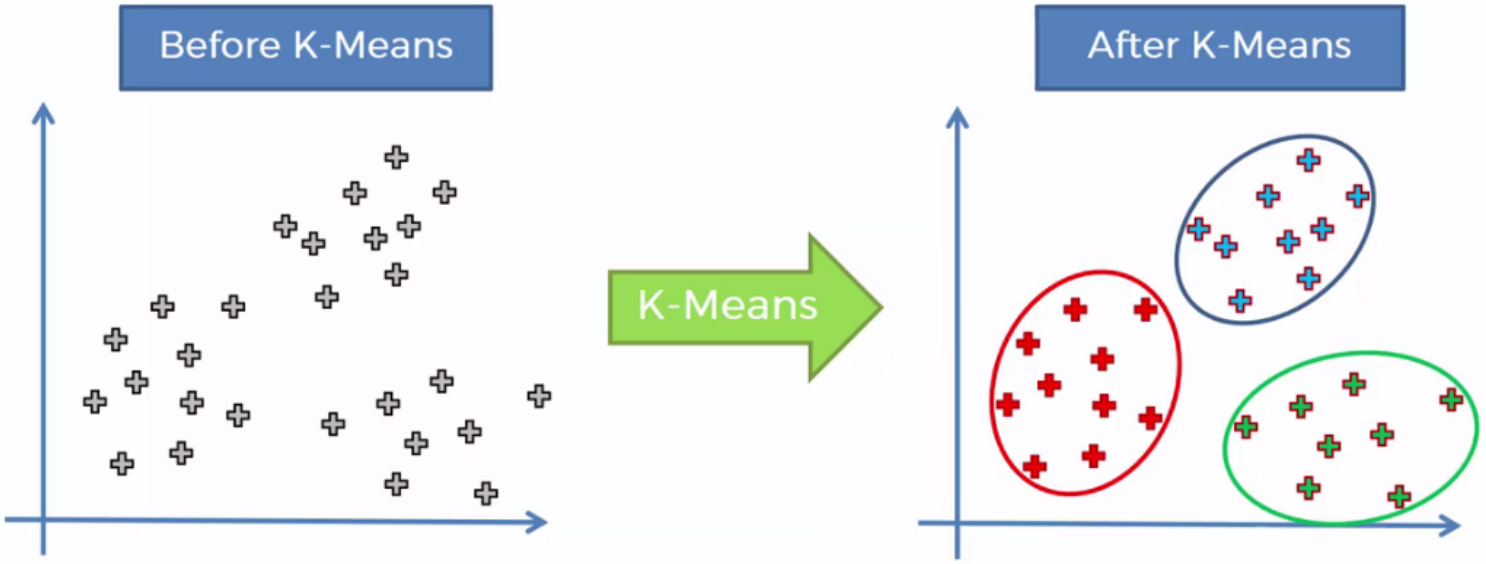



## Table Of Contents

## Main Requirements for clustering Algorithms

1. It should be scalable.
2. It should be able to deal with attributes of different types.
3. It should be able to discover arbitrary shape clusters.
4. It should have an inbuilt ability to deal with NOISE & outliers.
5. The cluster should not vary with the order of input records.
6. It should be able to handle data of high dimensions.
7. It should be easy to interpret & use.

## 10 K-means

Old Algorithm

Proposed in 1957 by Stuart Lloyd, BELL
In 1965 by Edward W. Forgy.
Sometimes called Lloyd-Forgy Algo.

→ K-means is a clustering approach in which data is grouped into K distinct non overlapping clusters based on their distances from the K centres.

→ The value of 'K' needs to be specified first & then the algorithm allocates the points to exactly one cluster.

## Theory of K-means

→ Let $C_1, C_2, \ldots C_K$ be the K-clusters.

→ then we can write :-

$C_1 \cup C_2 \cup C_3 \cup \ldots \cup C_K = \{1, 2, 3, \ldots n\}$ → i.e, Each data point has been assigned to a cluster

and also, $C_K \cap C_{K'} = \phi$ ⟶

this means, clusters are non-overlapping.

→ the idea behind the K-means approach is that within-cluster variation amongst the point should be minimum.

The within-cluster variance is denoted by :- $W(C_K)$

→ Hence, according to statement above, we need to minimise the variance for all cluster mathematically,

$$\underset{C_1, C_2, \ldots C_K}{\text{minimize}} \left\{ \sum_{K=1}^{K} W(C_K) \right\}$$

WCSS :- within cluster summation of squares

→ the next step is to define the criterion for measuring the within cluster variance. Generally, the criterion is the Euclidean distance between two datapoints.

$$W(C_K) = \frac{1}{|C_K|} \sum_{i, i' \in C_K} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

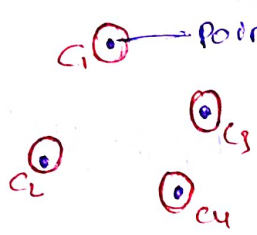
→ the above formula says that we are ① calculating the distances b/w all the points in a cluster ② then we are repeating it for all the K-clusters (so two summations)

Eg:-

Let's take 4 points

**Case ①:-**

4 points
4 clusters




→ Now, each point acts as a cluster and so it is centroid or itself.

∴ Distance b/w the point & centroid (same point) is zero (0).

→ And it is same for all the 4 clusters on this.

i.e) $w(C_4) = 0+0+0+0 = 0$

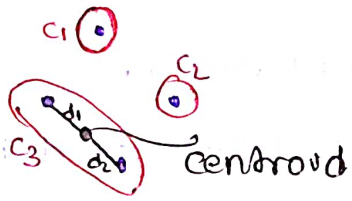
$C_1$ $C_2$ $C_3$ $C_4$

↓

Summation of distances.

**Case ②:-**

4 points
3 clusters




→ Now we need distance b/w points & the centroid of cluster it is present in.

$w(C_3) = C_1 + C_2 + C_3$

$w(C_3) = 0 + 0 + (d_1 + d_2)$

(1) Individual sum of distance of a cluster

$$\sum_{i=1} (x - c)^2$$

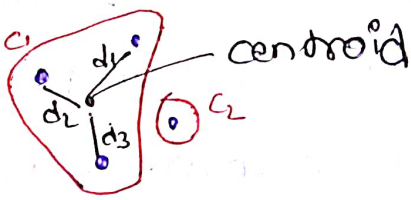
(centroid)

(2) All the clusters sum

$$\sum \sum_{i=1} (x - c)^2$$

**Case ③:-**

4 points
2 cluster




$w(C_4) = C_1 + C_2$

$w(C_4) = (d_1 + d_2 + d_3) + 0$

(distance b/w point & centroid) Summation of distances in a cluster.

Summation of the individual summations of distances in cluster

**Case ④:-** 4 points 1 cluster


$w(C_1) = C_1$

↓

$w(C_4) = (d_1 + d_2 + d_3 + d_4)$

Goal is to minimize $w(c_1)+w(c_2)+w(c_3)+w(c_4)$ → low variance for all clusters

So, we got ↓

$w(c_4) = 0$

$w(c_3) = d_1 + d_2$

$w(c_2) = d_1' + d_2' + d_3'$

$w(c_1) = d_1'' + d_2'' + d_3'' + d_4''$

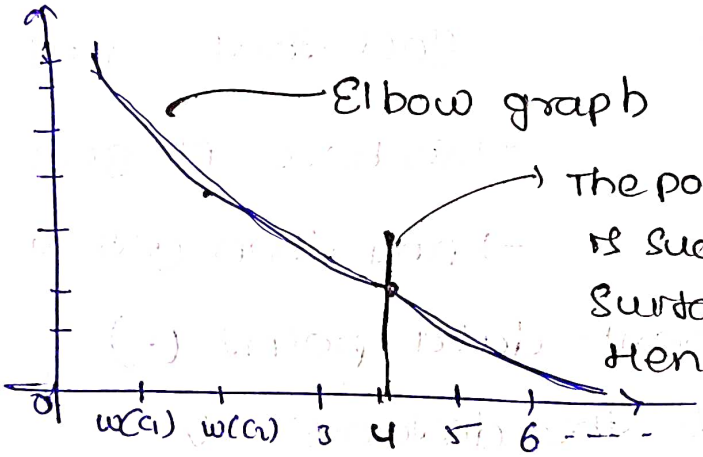
→ By observation, we can say that

As no. of cluster increases, $w(c_k)$ value decreases & tends to '0'

$$w(c_k) \propto \frac{1}{\text{no. of cluster}(k)}$$

$w(c_1) > w(c_2) > w(c_3) > \cdots > w(c_k)$

and the graph goes as -




within cluster summation of squares (WCSS)

→ this is how we find the value of 'k' in k-means

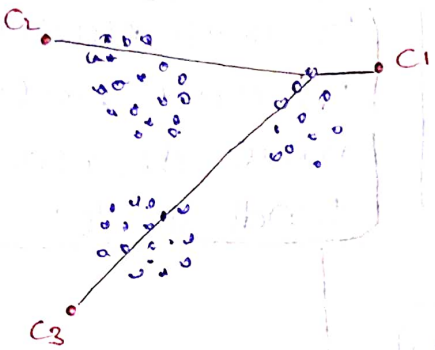
But, here is the problem in k-means,

(*) After selecting k value, it takes k-points as centroids and starts building the cluster normally.

But thing is where do we place those k-points ?

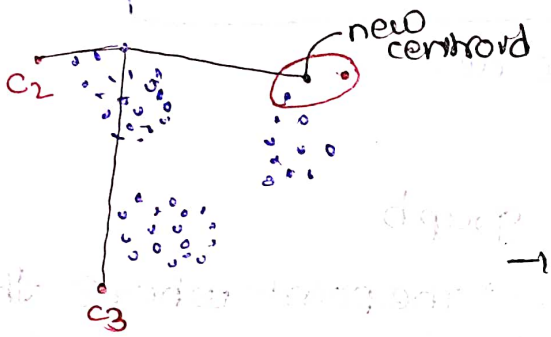
## (*) cluster building process

Let's say we got: (K=3)




→ After, randomly placing the K points,

→ It takes a point from our data (·) and measures the distance to all the K-centroid points (· ), And whichever distance is smaller, the point goes that cluster.



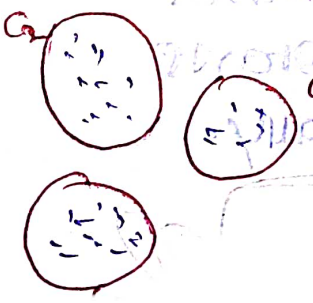

→ So, here it goes to $C_1$

→ And then gets a new centroid.

→ Now for next data point (·) we measure the distance of point to the "new centroids"

→ And in this way, cluster keeps building and centroid gets updated.

→ This process continues until all the data points are finished.



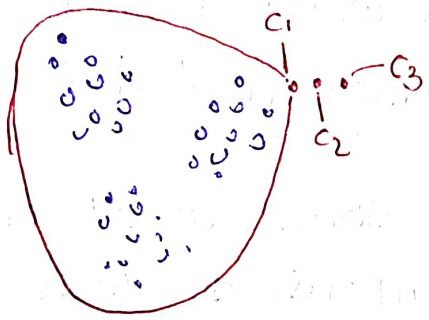

This is how grouping happens.

## Edge case scenario

### Problem with K-means

→ In the above example, we took $k=3$ and then it generated 3 random points and build cluster.

→ So, what if all the 3 points are close?




If all are close to only $c_1$.

And,
$c_2 = 0$
$c_3 = 0$
} elements

In this case, all data points would go to cluster ① ($c_1$)

which is not what we are expecting!!

### Algorithm

① Find WCSS for K-groups possible (max $k = n$) (n = no. of elements)

② Build Elbow graph, and understand K-value (point where there's distinction)

③ Randomly assign K-centres.

④ calculate distance of all points from all K-centres and allocate the points to cluster based on shortest distance to "cluster centroid".

The model's inertia is the mean squared distance b/w each instance & closest centroid.
Goal is to have model with low inertia.

⑤ Once all points assigned to cluster recompute centroids.

(6) Repeat steps 4 & 5 until the location of the centroids stop changing and the cluster allocation of the points becomes constants.

---

Note:-

→ K-means is only recommended for spherical data (or) symmetrical data.

→ For, non spherical data there won't be fair chance for k clusters to form clusters, and it will become high biased cluster.

(*) Challanges and Improvements in K-means:-

(1) We need to specify the no. of clusters beforehand.

(2) It is required to run the algorithm multiple times to avoid a suboptimal solution.

(3) K-means does not behave very well when the clusters have varying sizes, different densities or non-spherical shapes.

(4) An improvement to K-means is propoled in 2003 by charles Elkan.

→ It accelerates the algorithm by many unnecessary distance calculations ; which is achieved by exploiting the "TRIANGLE INEQUALITY"

In a triangle with sides a, b & c) $a+b>c$
i.e, the straight line is always shortest

⑤ Another important improvement to k-means algorithm called "k-means++", was proposed in 2006 by David Arthur & Sergei.

→ they introduced a smarter initialization step that tends to select centroids that are distant from one another, and this makes the k-means algorithm much less likely to converge to a suboptimal solution.

⑥ Yet another important variant of the k-means algorithm was proposed in a 2010 paper by David Sculley.

→ Instead of using the full dataset at each iteration, the algorithm is capable of using mini-batches, moving the centroids just slightly at each iteration. This speeds up the algorithm typically by a factor of 3 or 4 and makes it possible to cluster huge datasets that do not fit in memory. Scikit-Learn implements this algorithm in the "MiniBatchKMeans" class.