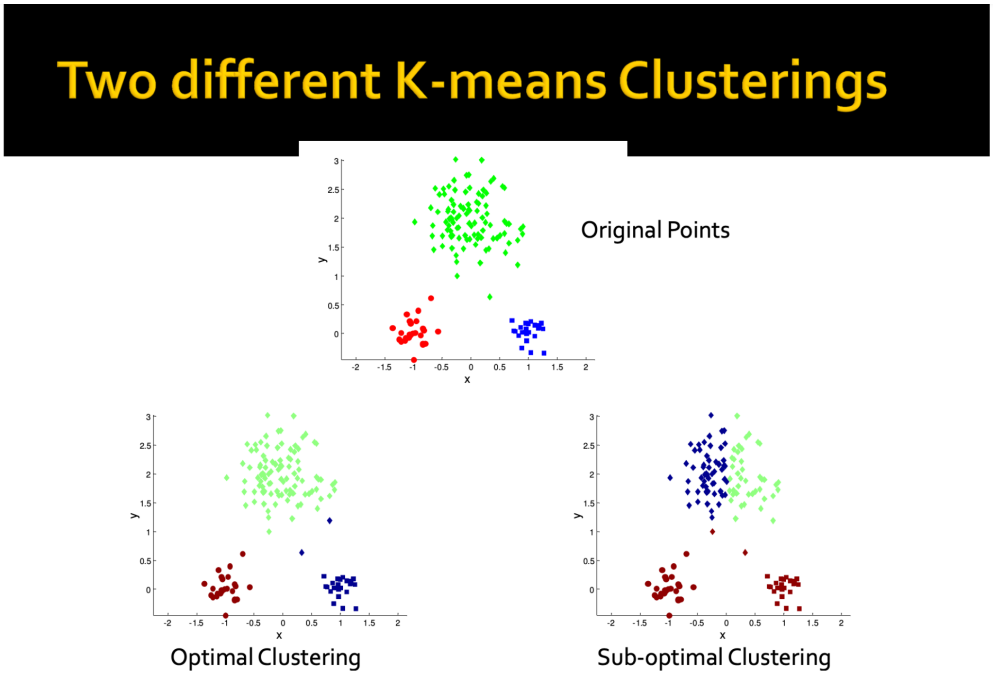
# 02 - K-Means++ & Hierarchical

While the K-Means algorithm is the simplest model for clustering, there are some drawbacks that come with it.

## Drawbacks of K-Means

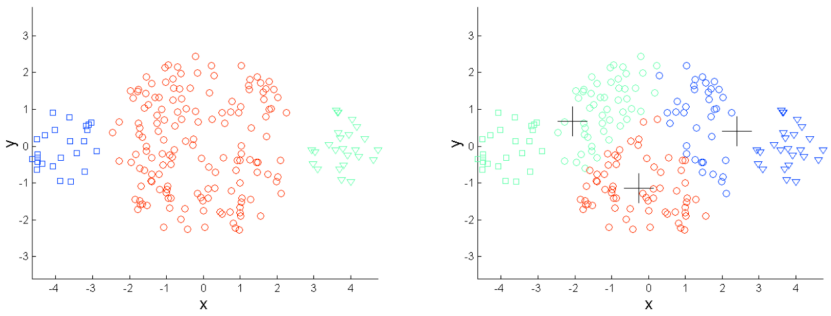
1. K-means is initialization dependent. This means, that the same data, with different initialization, will get different results (different clusters).
   For example;




   Use this visualization tool to see this problem and try it out by yourself!

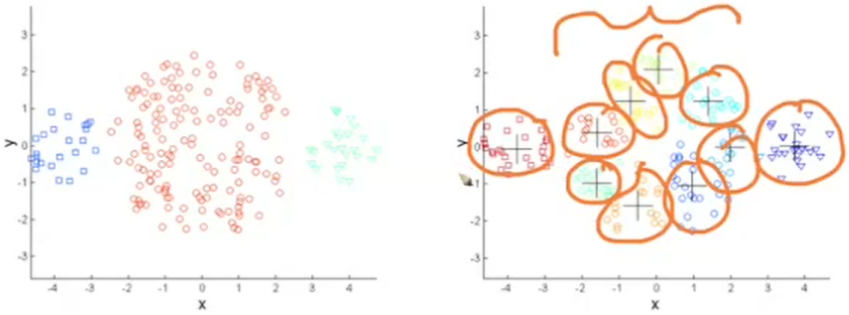
2. The k-means algorithm may not give the best results for data where the clusters are of varying size or density.

# Limitations of K-means: Differing Sizes



Original Points K-means (3 Clusters)

- One way of solving this problem would be to increase the value of K.
- Once clusters are formed, similar clusters can be grouped together to form a mega cluster.

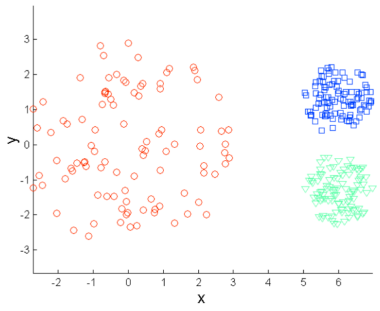


Original Points K-means Clusters

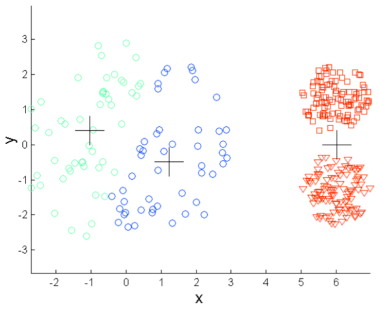
One solution is to use many clusters.
Find parts of clusters, but need to put together.

- The problem with this approach is the grouping of similar clusters is not easy

## Limitations of K-means: Differing Density




Original Points

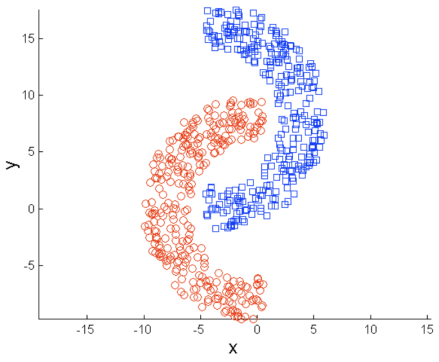


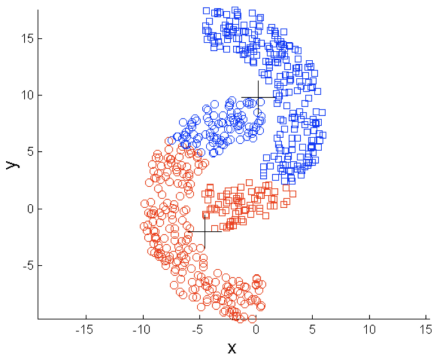

K-means (3 Clusters)

3. The number of clusters (k) needs to be defined prior to clustering.

4. It does not work well with non-globular clusters.

## Limitations of K-means: Non-globular Shapes



# K-Means++

- To overcome the drawback due to the random initialization of centroids in K-means clustering, we use K-means++. It is smarter to initialize the centroids in order to improve the clustering algorithm.

- Consider data where we want to initialize 3 centroids.
    - We pick the first centroid at random
    - Now, to pick the second centroid, we want to pick a point that is as far away as possible
- If you think about it, we would want to pick a point that is far away, because if two centroids are closer to each other, two clusters for that region of data points will be formed

- Most of the time data points belonging to the same region will share similar characteristics and they should ideally belong to one cluster, instead of two.
- So, what we do is compute the distance from the centroid C1 of all the data points present in our dataset $D$ such as: $D - \{C_1\}$
- But there's a little risk with this. If we select a datapoint as a second centroid with the farthest distance, then an outlier might be picked as a centroid, and we might have a cluster with the centroid $C_2$ only.

- So, what we do is pick a centroid probabilistically, instead of picking it deterministically.
- It is done in such a way that the probability of picking a centroid is proportional to the distance from the first centroid $C_1$.

- The steps involved in the initialization of centroids are:
    - ➔ Select the first centroid randomly from the data points.
    - ➔ Choose the next center as the farthest point from the first center.
    - ➔ The next center would be a data point farthest from both the first and second centers.

- Repeat steps 2 and 3 until **k** centroids have been sampled.

- If there are **outliers** in our data, then instead of choosing them as centroid, we can choose the farthest point as the centroid with a **probability proportional to the distance.** This is the implementation that sklearn follows by default..

# Hierarchical Clustering

- Broadly categorizing, there are two ways of performing Hierarchical Clustering.

    1. **Agglomerative Clustering:**
        - ➔ The word agglomerative suggests combining things
        - ➔ It is a bottom-up approach
        - ➔ Agglomerative clustering starts with the assumption that every data point is a cluster
        - ➔ Then, it groups the clusters which are close to each other until there is only a single cluster left

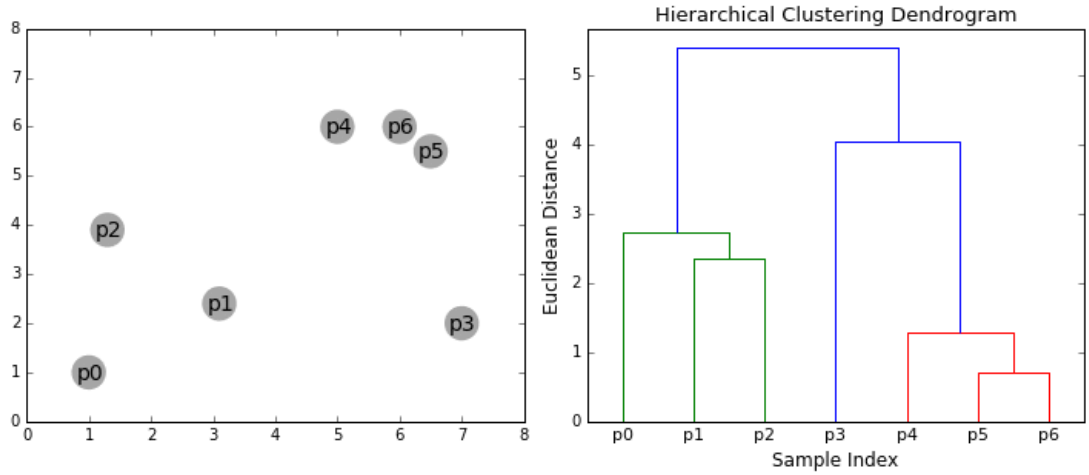
    2. **Divisive Clustering:**
        - ➔ It is the complete opposite of the agglomerative approach
        - ➔ It is a top-down approach
        - ➔ It starts with one big cluster that contains all the data points.
        - ➔ It then divides the points into different clusters till each data point is a cluster itself

## Agglomerative Clustering

- The steps involved in Agglomerative Clustering are:

    1. Assume each point is a cluster (n datapoints -> n clusters)
    2. Compute Proximity Matrix ($Pn*n$)
    3. Repeat until a single cluster is left:
        a. Merge the closest clusters
        b. Update the proximity matrix

- If we visualize this, this looks like a Tree, but there is another name that is often used in Data Mining terminology which is called Dendrogram.




## Proximity Matrix

- Proximity matrix is a matrix of distances or similarity.
- The word proximity suggests how close things are
- Say, at any point we're having $C_m$ clusters. For each of the pairs of clusters, the proximity matrix $P$ will indicate the similarity between clusters $C_i$ and $C_j$.

- Initially the proximity matrix $P$ will be $N*N$ matrix.
- Suppose cluster $C_i$ and $C_j$, where $i \neq j$, are similar and they have the smallest value in the proximity matrix, then those clusters will be combined and proximity matrix will get updated
- The new matrix will be a $N-1*N-1$ matrix, as two clusters have combined.

- One can use the following distances for computing the values of proximity matrices.

1. Using Euclidean Distance between the centroids of two clusters $C_i$ and $C_j$.
2. Maximum distance between two points $x_i$ and $x_j$, such that $x_i \in C_i$ and $x_j \in C_j$.

3. Minimum distance between two points $x_i$ and $x_j$, such that $x_i \in C_i$ and $x_j \in C_j$.
4. Average Distance: $\sum_{x_i \in C_i} \sum_{x_j \in C_j} \frac{dist(x_i x_j)}{|C_i||C_j|}$
5. Ward's Distance: $\sum_{x_i \in C_i} \sum_{x_j \in C_j} \frac{dist(x_i x_j)^2}{|C_i||C_j|}$

## Limitations of Hierarchical Clustering

1. With large datasets, Agglomerative Clustering does not work well
    a. Space Complexity = $O(n)$: Proximity Matrix
    b. Time Complexity = $O(n^2)$
2. Unlike K-means where we try to minimize **within-cluster distance**, there is **no mathematical objective** that is being minimized in Agglomerative clustering.