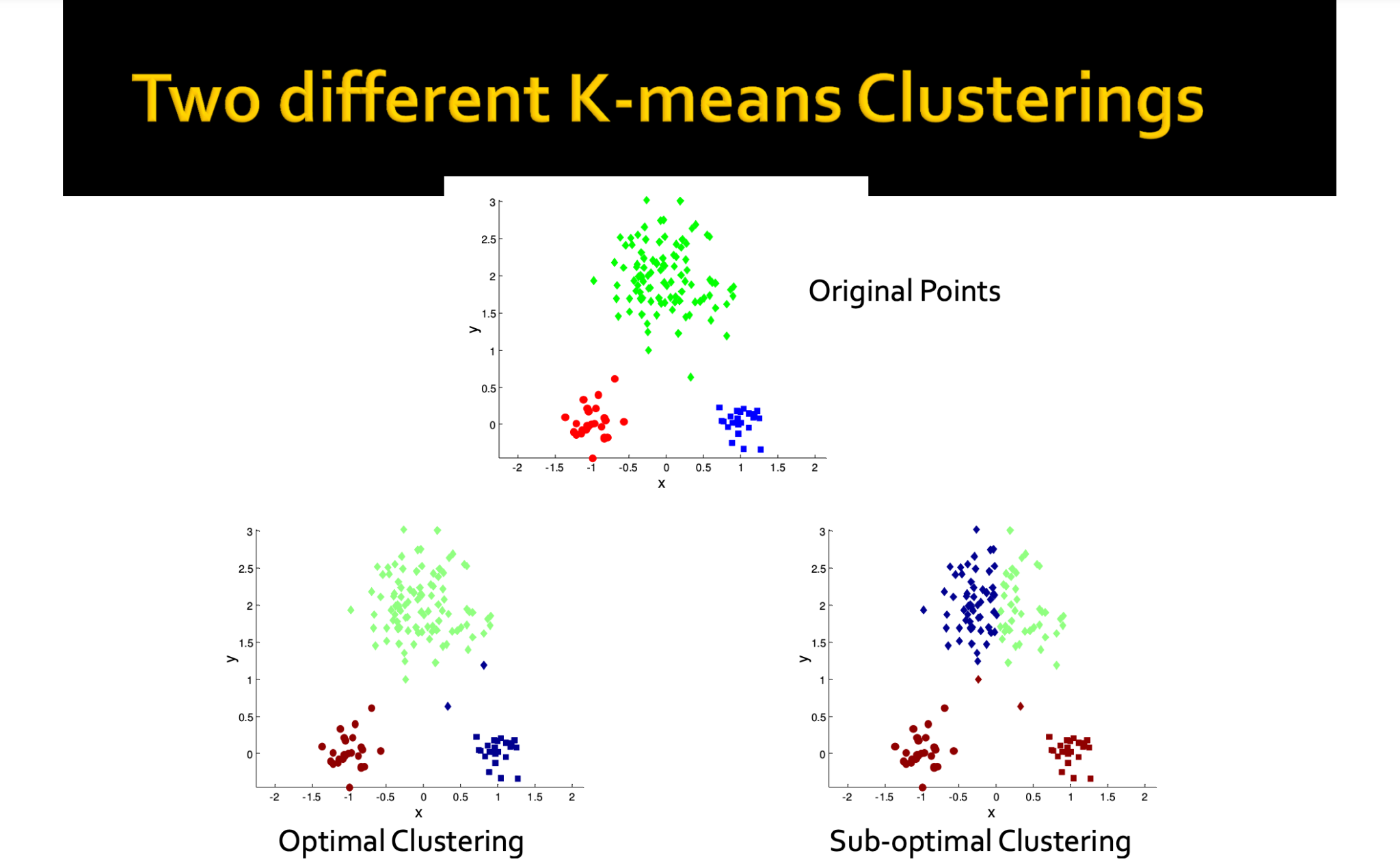
**02 - K-Means++ & Hierarchical**

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

# **Drawbacks of K-Means**

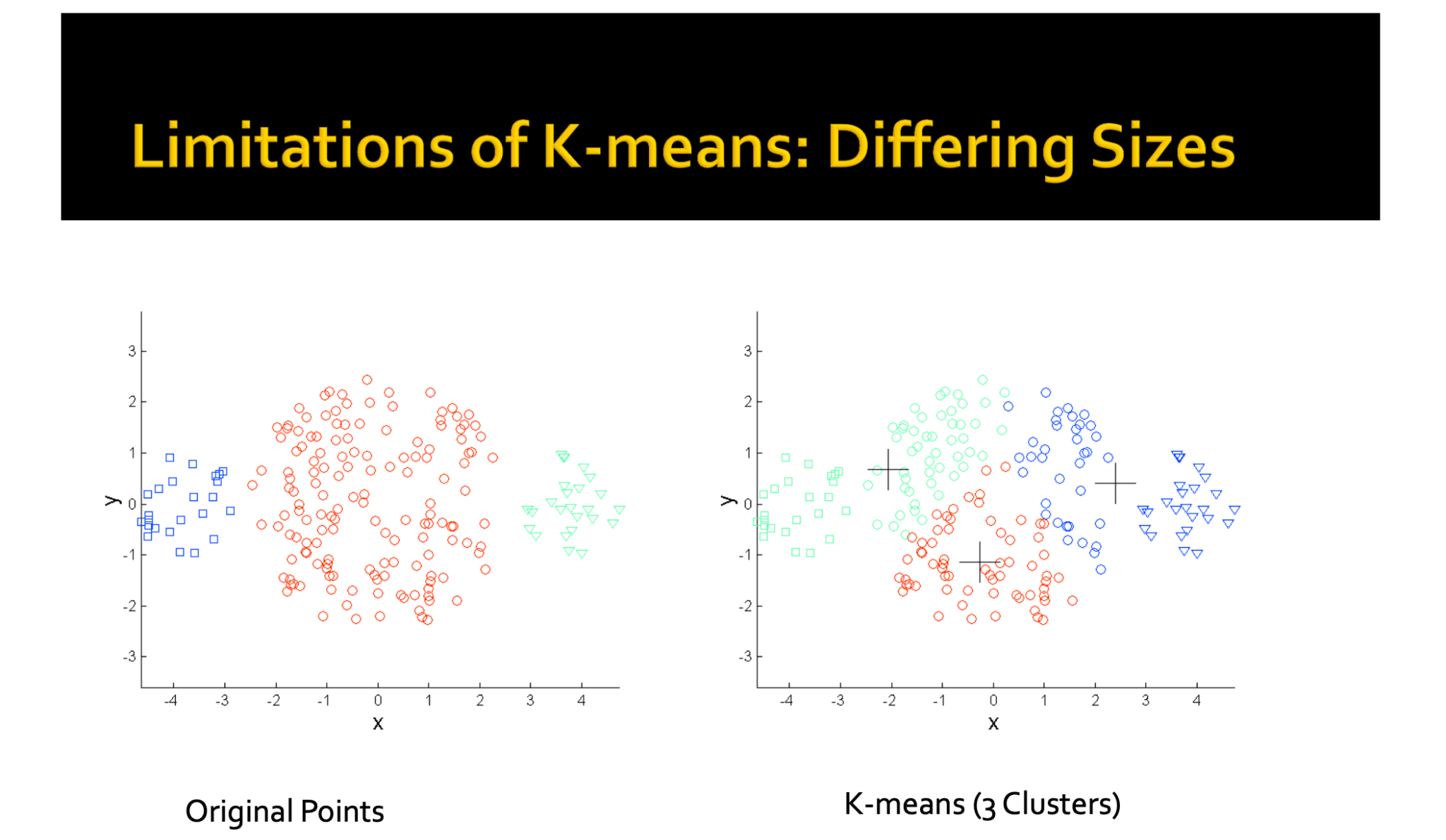
1. K-means is initialization dependent. This means that the same data, with different initialization, will get different results (different clusters).

For example;

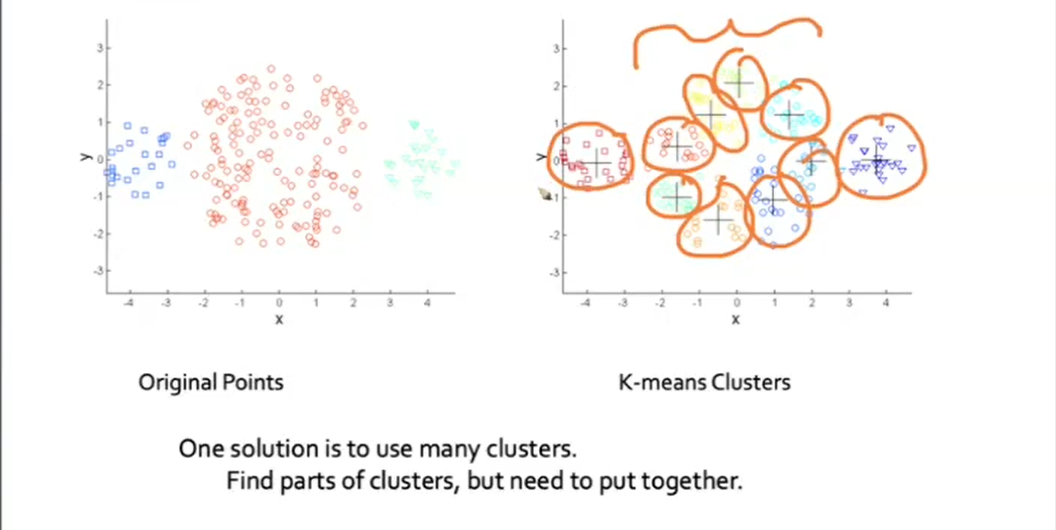


Use this [visualization tool](https://www.naftaliharris.com/blog/visualizing-k-means-clustering/) to see this problem and try it out by yourself!

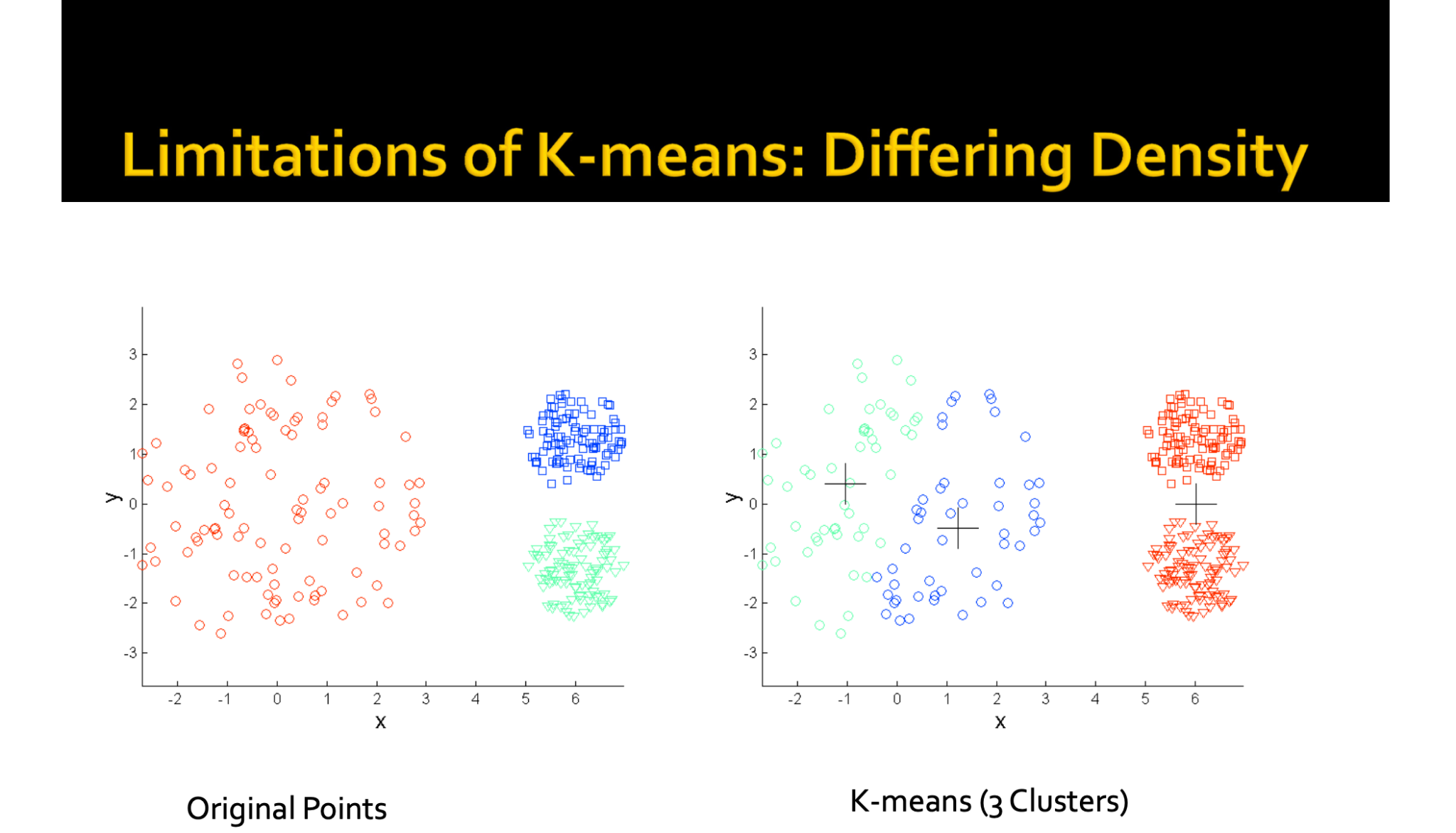
1. The k-means algorithm may not give the best results for data where the clusters are of varying size or density.



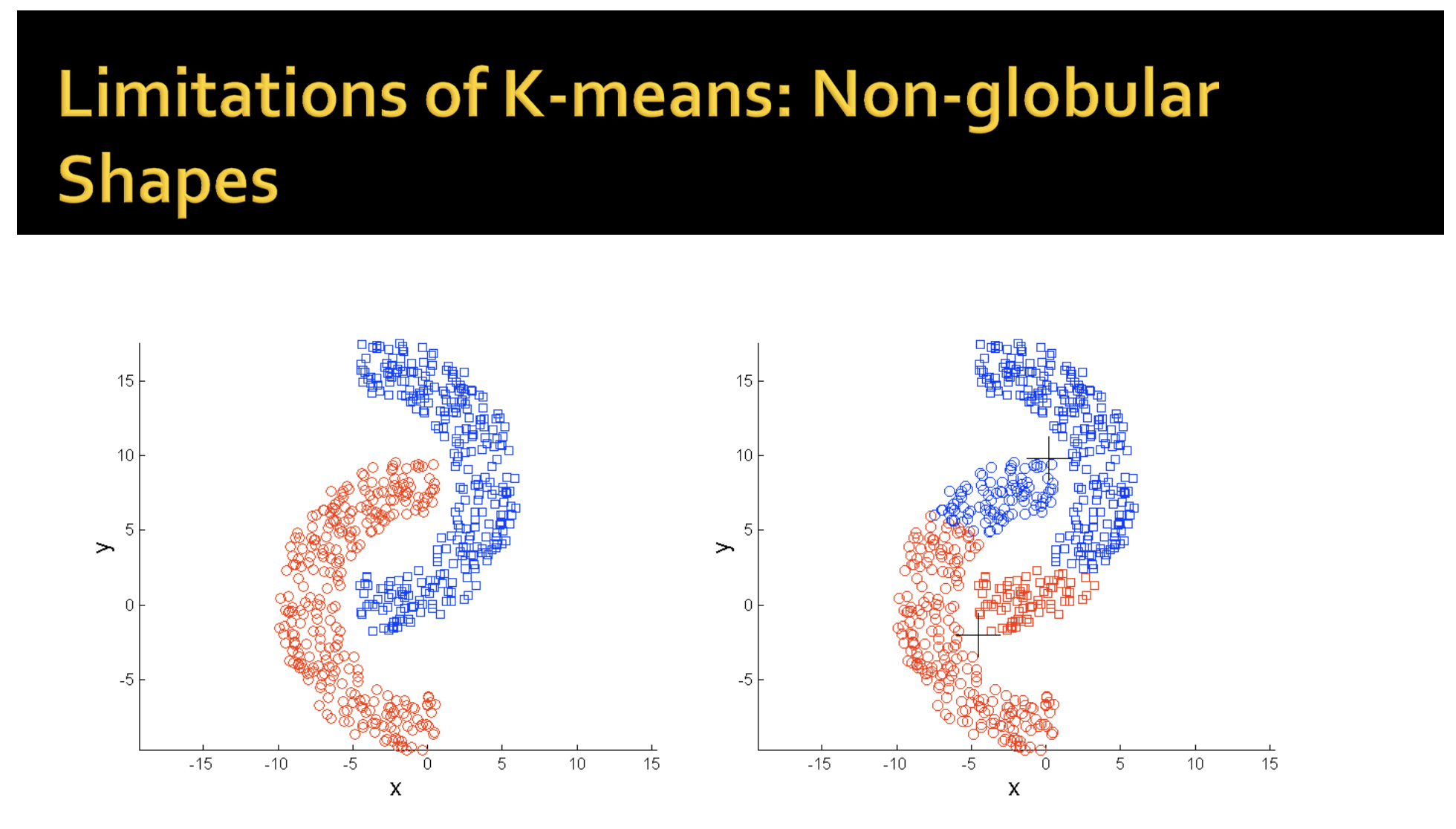
* One way of solving this problem would be to increase the value of K.
* Once clusters are formed, similar clusters can be grouped together to form a mega cluster.



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



1. The number of clusters (k) needs to be defined prior to clustering.
2. It does not work well with non-globular clusters.



# **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 (any point from the data is picked as initial centroid randomly)
  + 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 - {C1}
* We can now select the farthest point from the 1st centroid as the 2nd initial centroid.
* 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 C2 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 C1.
* So now we can run the full algorithm many times so that, a few times, non outlier points will be initial centroids, and one of those trials will give us the best clustering.
* 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

1. **Divisive Clustering:**

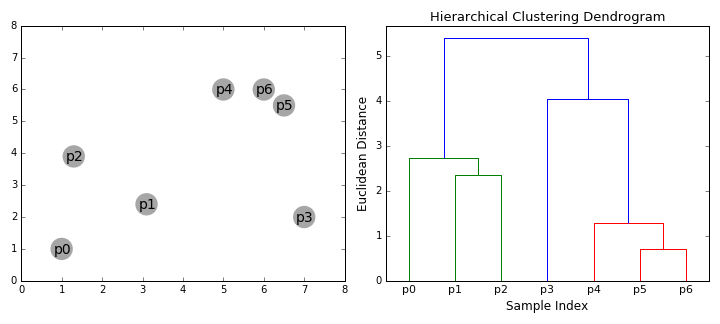
* It is the complete opposite of the agglomerative approach
* It is a top-down approach
* It starts with one big cluster that contains all the data points.
* It then divides the points into different clusters till each data point is a cluster itself

## **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:
   1. Merge the closest clusters
   2. Update the proximity matrix

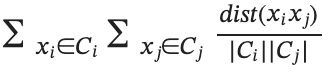
* If we visualize this, this looks like a Tree, but there is another name that is often used in Data Mining terminology which is called Dendrogram.
* Link to animation: [Hierarchical Clustering](https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/015/038/original/hierarch.gif?1663699414)



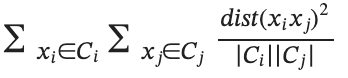
## **Proximity Matrix**

* Proximity matrix is a matrix of distances or similarity.
* The word proximity suggests how close things are
* Say, at any point we're having Cm clusters. For each of the pairs of clusters, the proximity matrix 𝑃 will indicate the similarity between clusters Ci and Cj.
* Initially the proximity matrix 𝑃 will be 𝑁∗𝑁 matrix.
* Suppose cluster 𝐶𝑖 and 𝐶𝑗, where 𝑖≠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 𝐶𝑖 and 𝐶𝑗.
2. Maximum distance between two points 𝑥𝑖 and 𝑥𝑗, such that 𝑥𝑖 ∈ 𝐶𝑖 and 𝑥𝑗 ∈ 𝐶𝑗.
3. Minimum distance between two points 𝑥𝑖 and 𝑥𝑗, such that 𝑥𝑖 ∈ 𝐶𝑖 and 𝑥𝑗 ∈ 𝐶𝑗.



1. Average Distance:



1. Ward’s Distance:

### **Limitations of Hierarchical Clustering**

1. With large datasets, Agglomerative Clustering does not work well
   1. Space Complexity = 𝑂(𝑛): Proximity Matrix
   2. Time Complexity = 𝑂(𝑛2)
2. Unlike K-means where we try to minimize **within-cluster distance**, there is **no mathematical objective** that is being minimized in Agglomerative clustering.