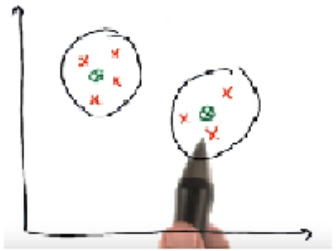
K-Means Clustering



In this example, we can see two clusters of red crosses. And cluster centres can also be visualized (green cross).

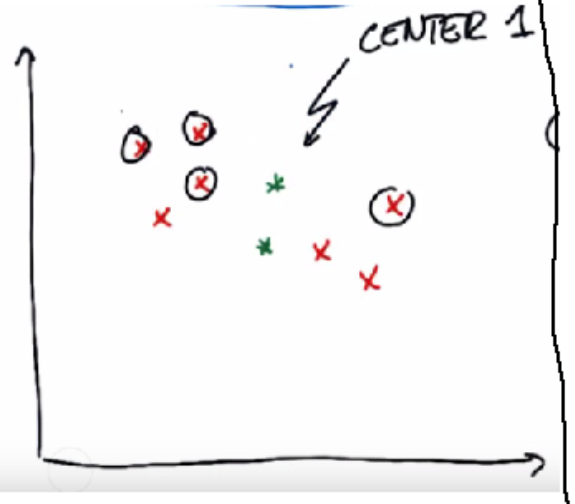
Lets apply K-Means algorithm to find those cluster centers:

In K-Means, we randomly assign cluster centers. Obviously, those two cluster centers are not correct.

So, k-Means works in 2 steps:

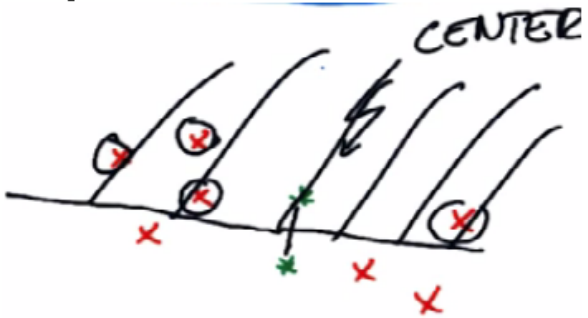
- ASSIGN
- OPTIMIZE

Lets start 1. Assign.

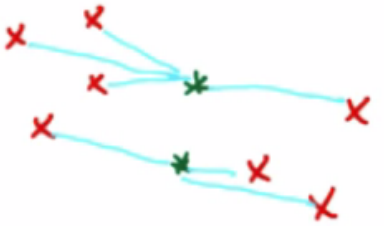


We can see that those 4 circled X are closer to center 1 than center.

We can visualize this by drawing orthogonal line

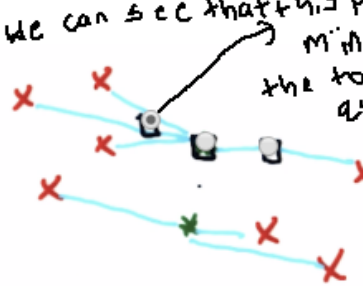


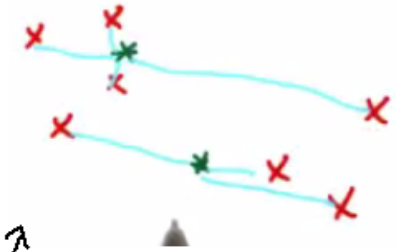
Now we shall assign those points to the clusters.



Next step is Optimization

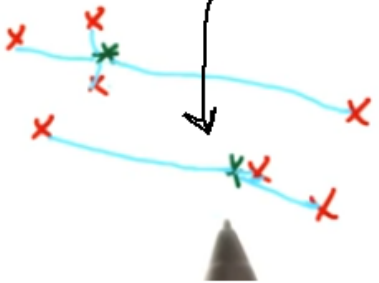
Optimization is done by minimizing the total quadratic dist of a cluster center to the points. So we need move our cluster centers





After moving cluster 1

Similarly, for cluster 2



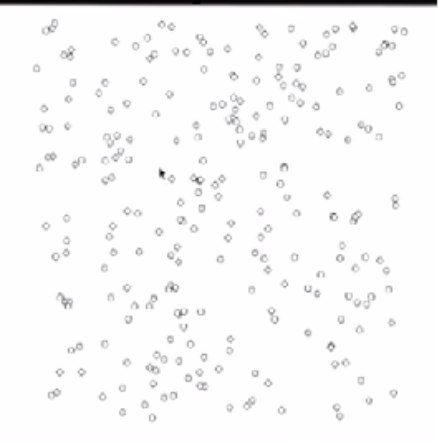
We shall repeat the process.

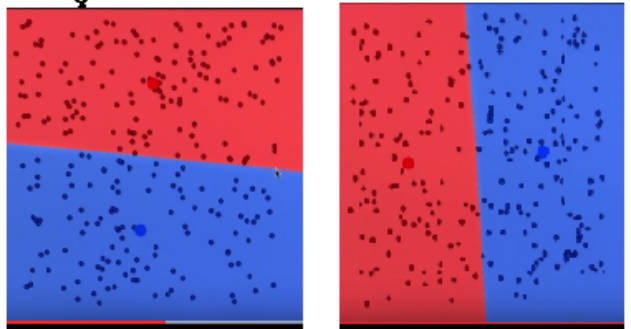
① Assign, for cluster 1

for cluster 2 as well

So, we have iteratively assigned and optimized to get the correct cluster centers as expected. This is called K-Means clustering algorithm

Uniform points Data
Lets apply kmeans to this

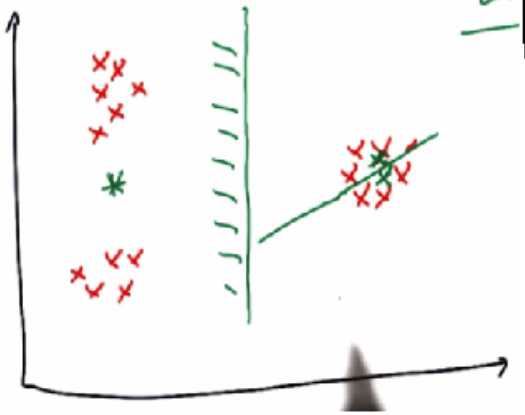




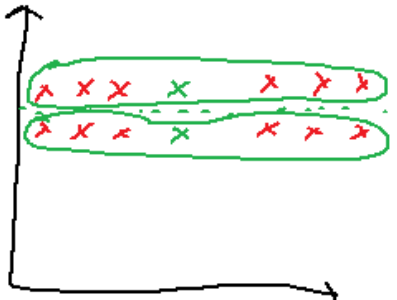
So for the same data, just because the initial points were placed differently we got different results. Thus, in K-Means, the initial conditions (which is random) will affect the final result.

- Given, a fixed training dataset, the output will not always be the same.

This can happen in KMeans bcoz of the clusters chosen initially. Its called "LOCAL MINIMUM"



So the Local "Hill Climbing" algorithm will give better result when run again.



This can happen in kMeans Based on initially points chosen.

If we re-run it by initializing differently,

The clusters will now resolve themselves

But still, chances are, that Bad results may come.

sklearn.cluster.KMeans

class sklearn.cluster. **KMeans** (*n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=1, algorithm='auto'*) [source]

K-Means clustering

Number of clusters to choose

Number of iterations of assign and optimize

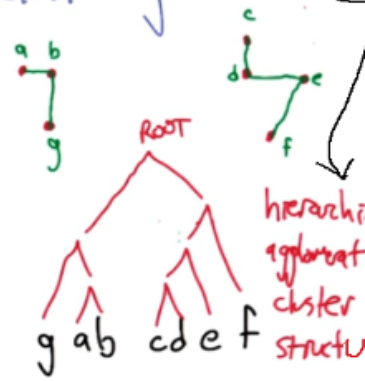
300 is good. most of the time iteration stops even before reaching this value

number of different initializations. So the ensemble of those clusters with different initializations will give you clusters that makes sense.

Single Linkage Clustering

Also called → SLC / HAC

- consider each object a cluster (n objects)
- define intercluster distance as the distance between the closest two points in the two clusters
- merge two closest clusters
- repeat n-K times to make K clusters (K=2 Here)



Issues with SLC