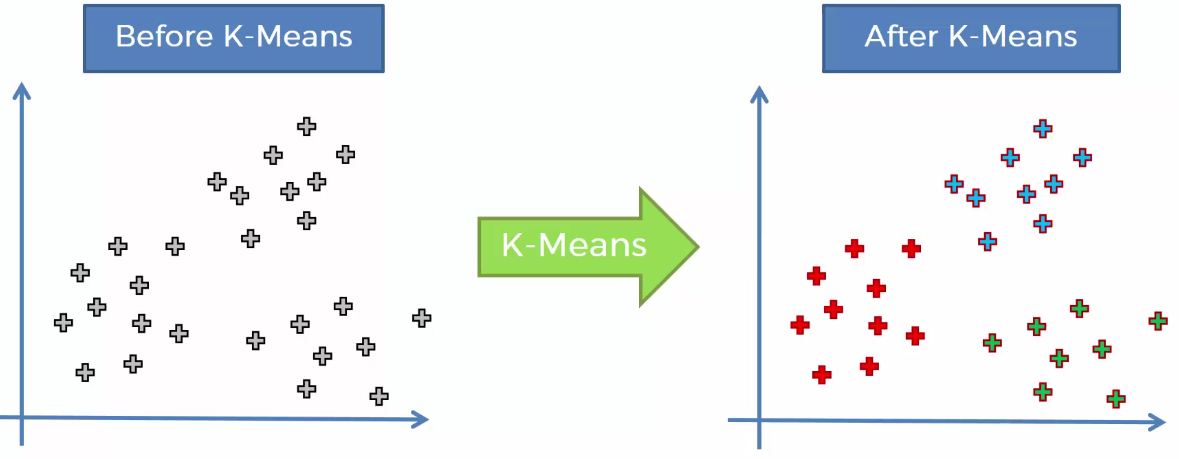
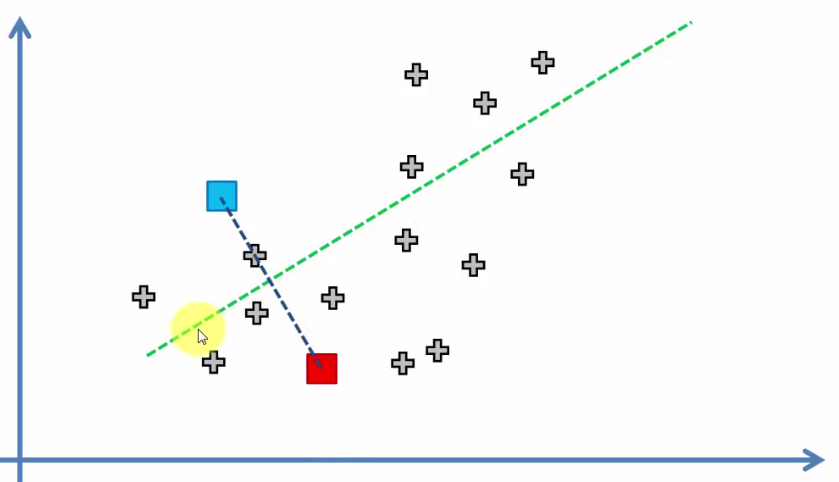
**K-Means Clustering**

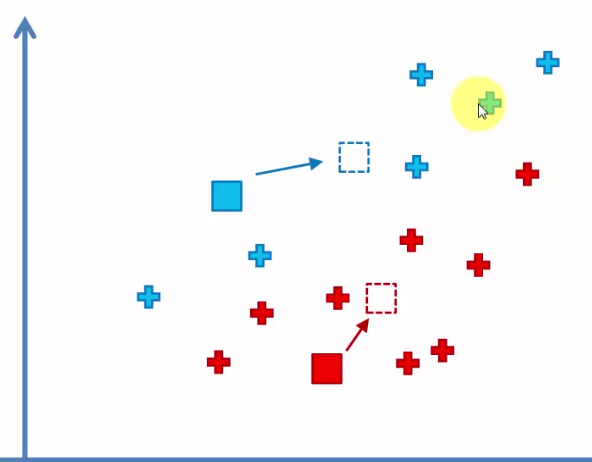
* How can we ID groups w/in a dataset



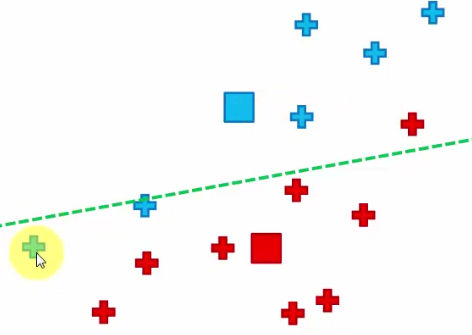
* 1) Choose optimal number of clusters K
* 2) Select random K points to act as **centroids** of clusters (not necessarily from the dataset, can be any {x,y} values on the grid)
* 3) Assign each DP to the closest centroid (for example via Euclidean distance), which will create K starting clusters
* Distance measurements depend on business case



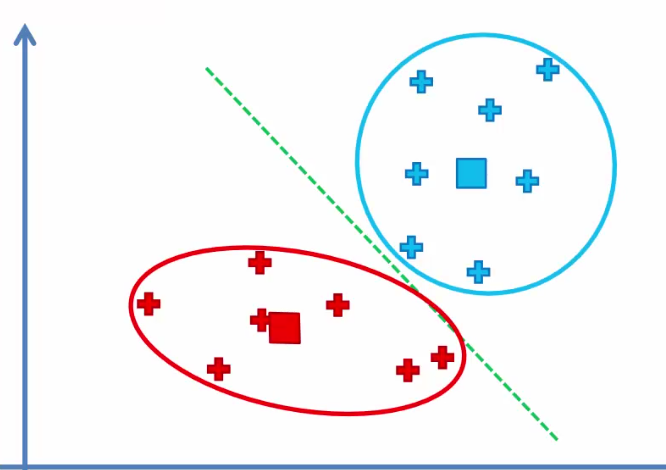
* 4) Compute new centroid of each cluster

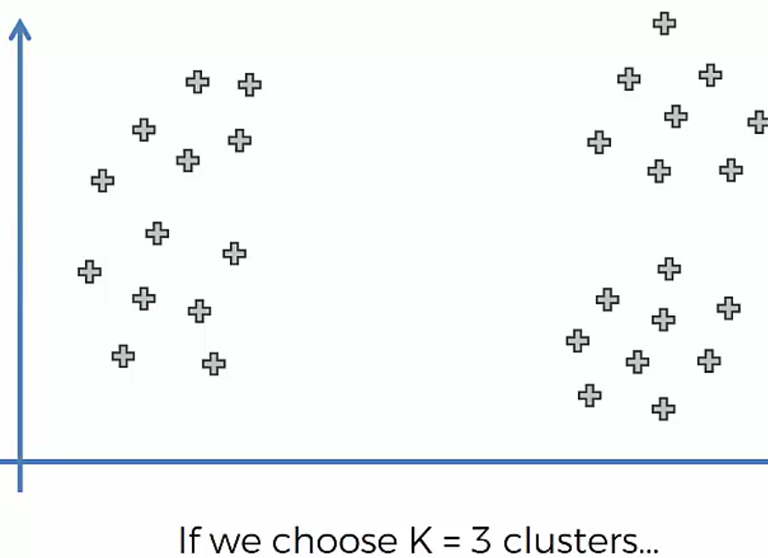
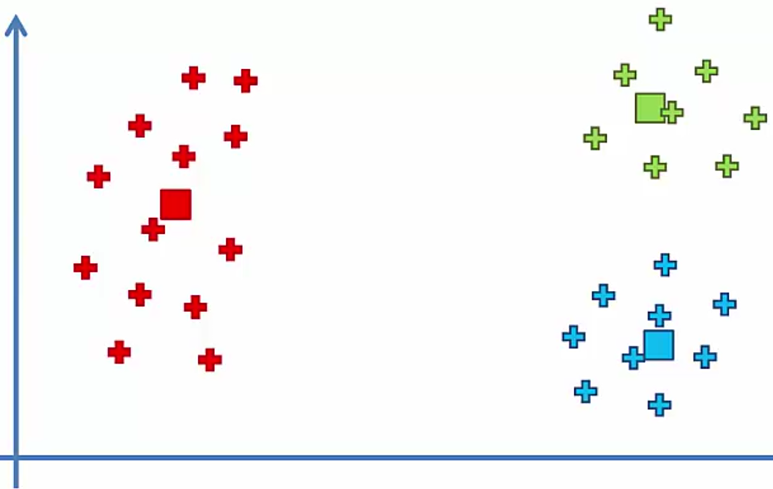


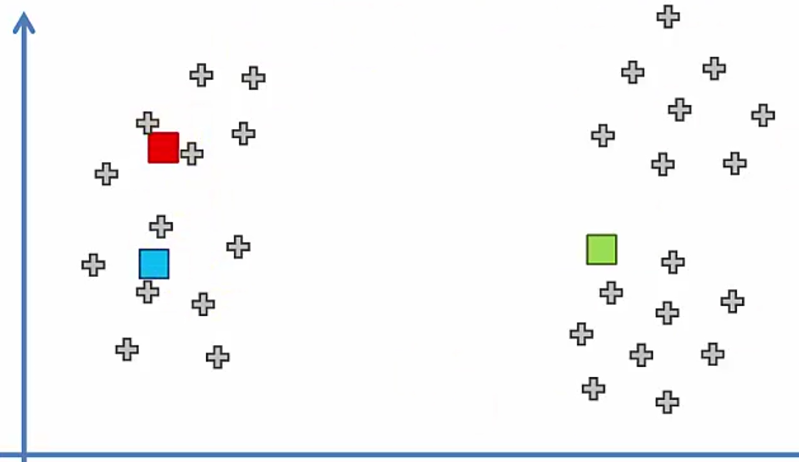
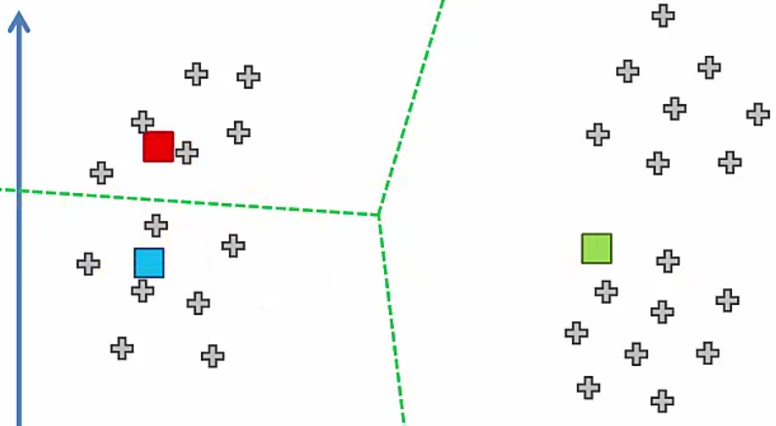
* 5) Reassign each DP to new closes centroid

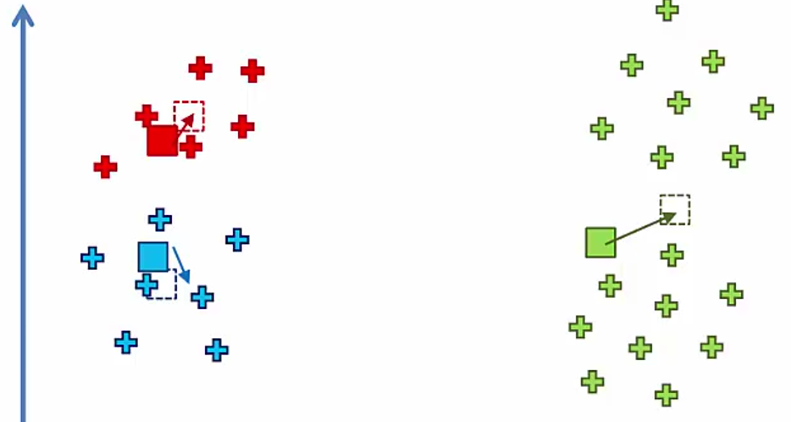
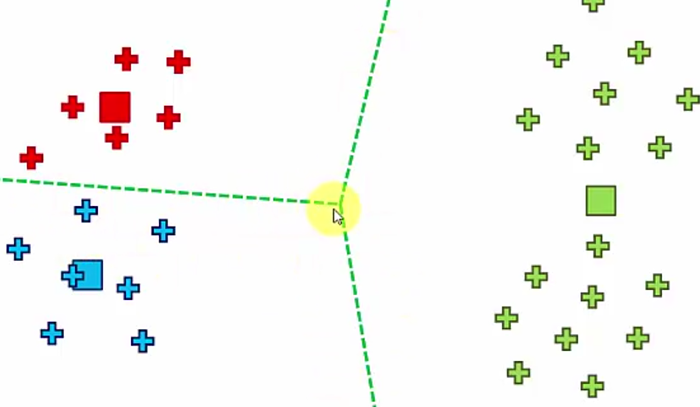


* 6) If reassignment occurred, go back to step 4 and repeat, otherwise algorithm is done



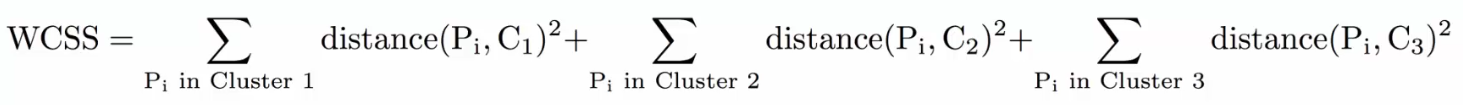
* **Random Initialization Trap**
*  🡺 
* If we select the centroids in a different locations, can we change the results?
* If we had a “bad” random initialization:

 🡺 

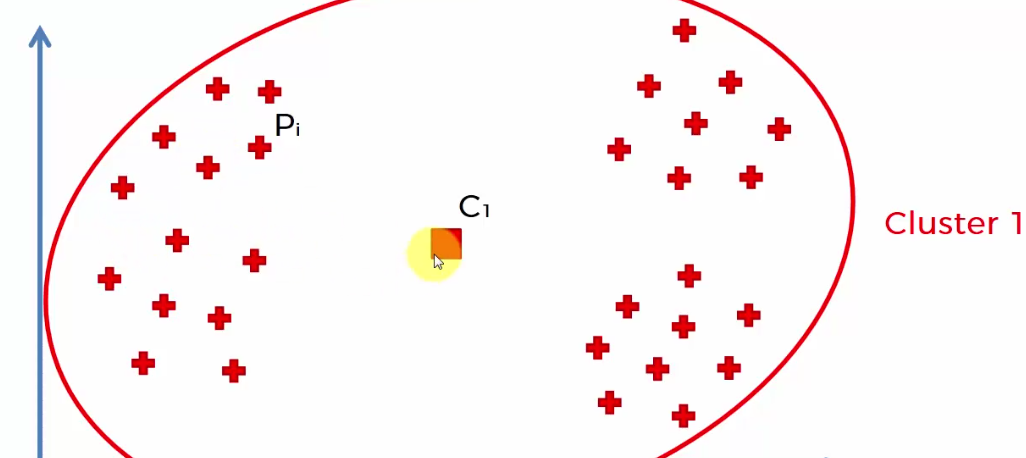
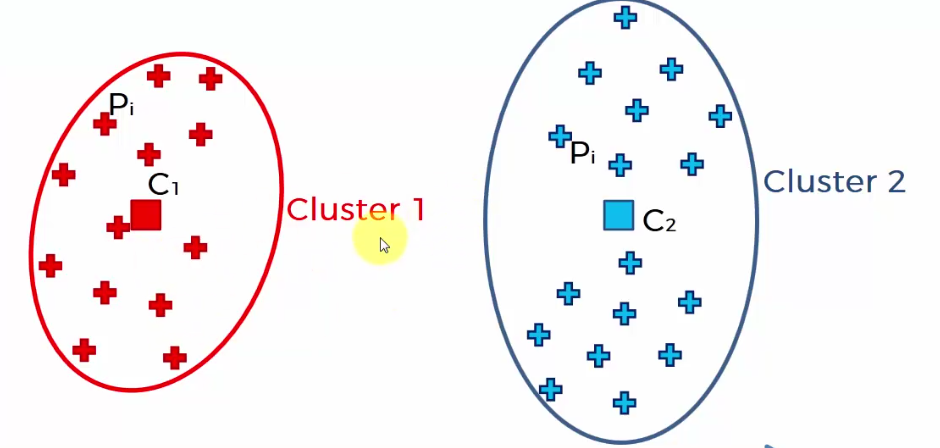
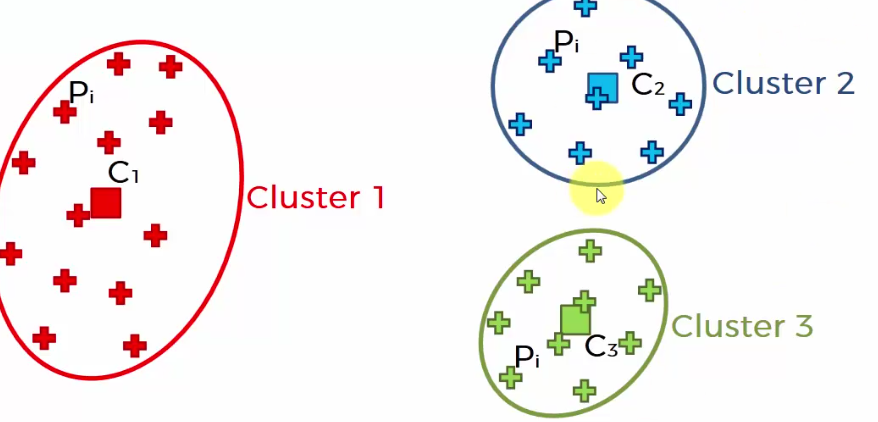
*  🡺 
* These clusters are completely different
* So, selection of the initial K centroids can have a significant effect on the end result of the algorithm
* The **K-means ++** algorithm is supposed to help correctly select the centroids by modifying the K-means algorithm in the background

**Choosing Right # of Clusters**

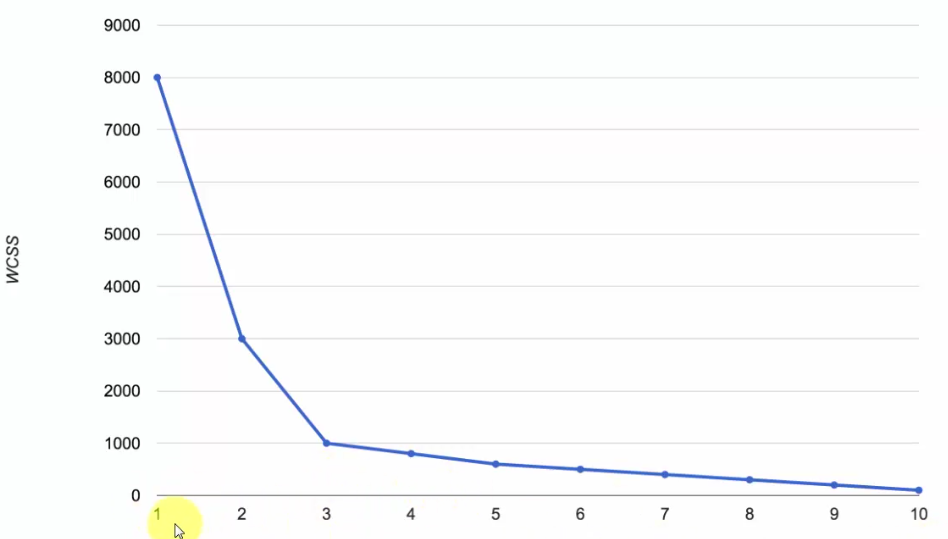
* Need a way to evaluate how a certain # of clusters performs
* **W/in Cluster Sum of Squares:**



* Squared sum of distance of each points w/in a cluster P(i) from centroid of cluster C(k), then sum for all clusters
* As we increase # of clusters, it decrease WCSS

* Ex: Can make each DP its own cluster, so WCSS would be = 0



* **Elbow method** = good for trying to find optimal # of clusters (a bit arbitrary)
* 1st 2 chances in # of clusters has huge drop in WCSS, and then as we move on, there is minimal decrease (K = 3)