CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Machine Learning

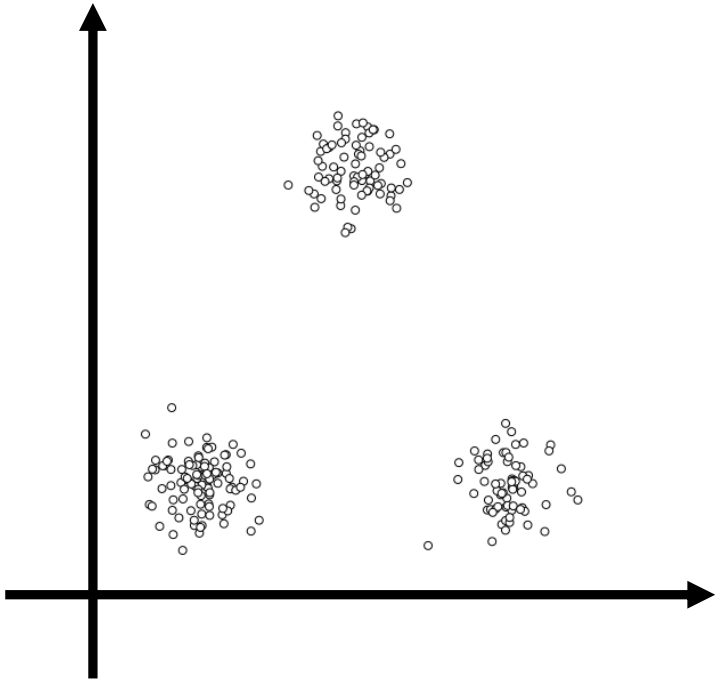
Machine Learning

Lecture: K-Means Clustering

Ted Scully

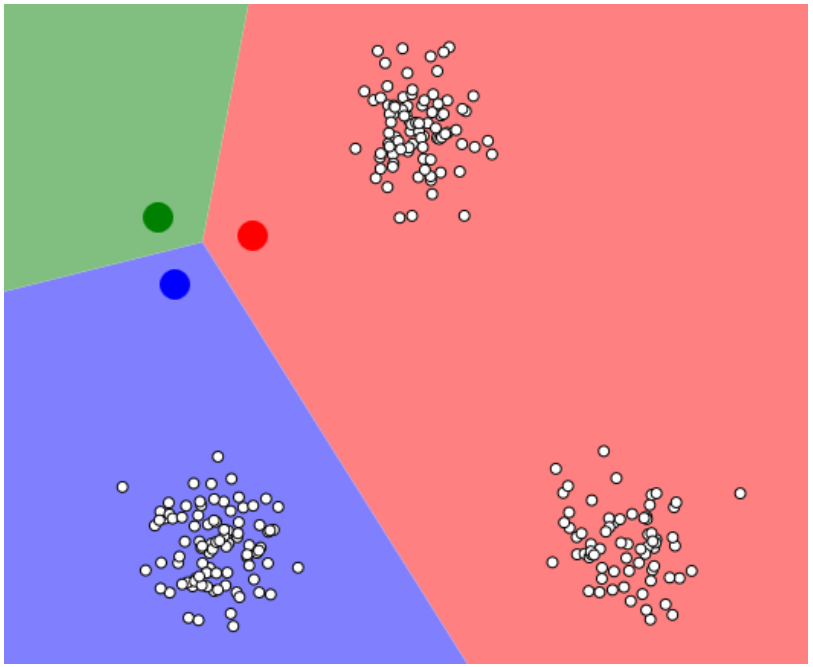
K-Means Clustering

- K- Means clustering is the most widely used unsupervised learning technique.
- It is also a very simple algorithm to understand and implement.
- It is **parameterized**. We must specify the number of centroids in advance.
- It is an iterative algorithm that involves two main steps:
  - **Cluster Assignment**
  - **Move Centroid (Move Cluster Centre)**



K-Means Clustering

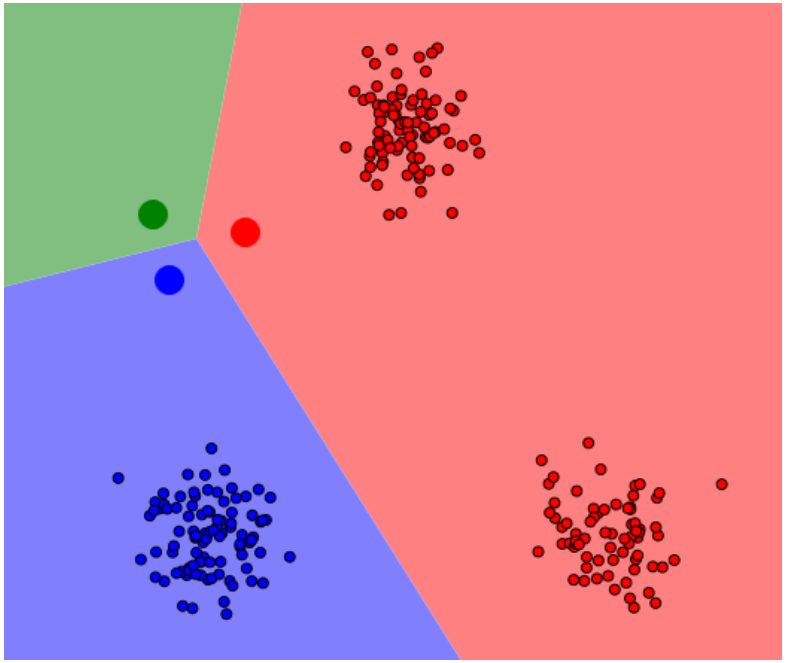
- The very first thing we do is we 'randomly' pick **k points** in space, we call each of these points **centroids**.
- In the example below I have "randomly" picked three centroid points in space (green, blue and red circles).
- We then iteratively perform the **cluster assignment** and **move centroid** steps.



```
for x iterations:
        cluster assignment
        move centroids
```

K-Means Clustering – Cluster Assignment Step

- In the cluster assignment step we assign each training example to the nearest cluster.
- You will notice that all training points are assigned to either the blue or red centroid. The green centroid is further away from all training points.
- We have coloured the training example accordingly.



for x iterations:
    cluster assignment
    move centroids

Distortion Cost Function

- The KMeans algorithm is attempting to minimize a specific cost function (often referred to as the distortion cost function).

- **$D(c_1, c_2, \ldots c_m, u_1, .., u_k) = \frac{1}{m}\sum_{i=1}^{m} | |x^i - U_{c(i)}| |^2$**

- K-Means attempts to find min $D(c_1, c_2, \ldots c_m, u_1, .., u_k)$

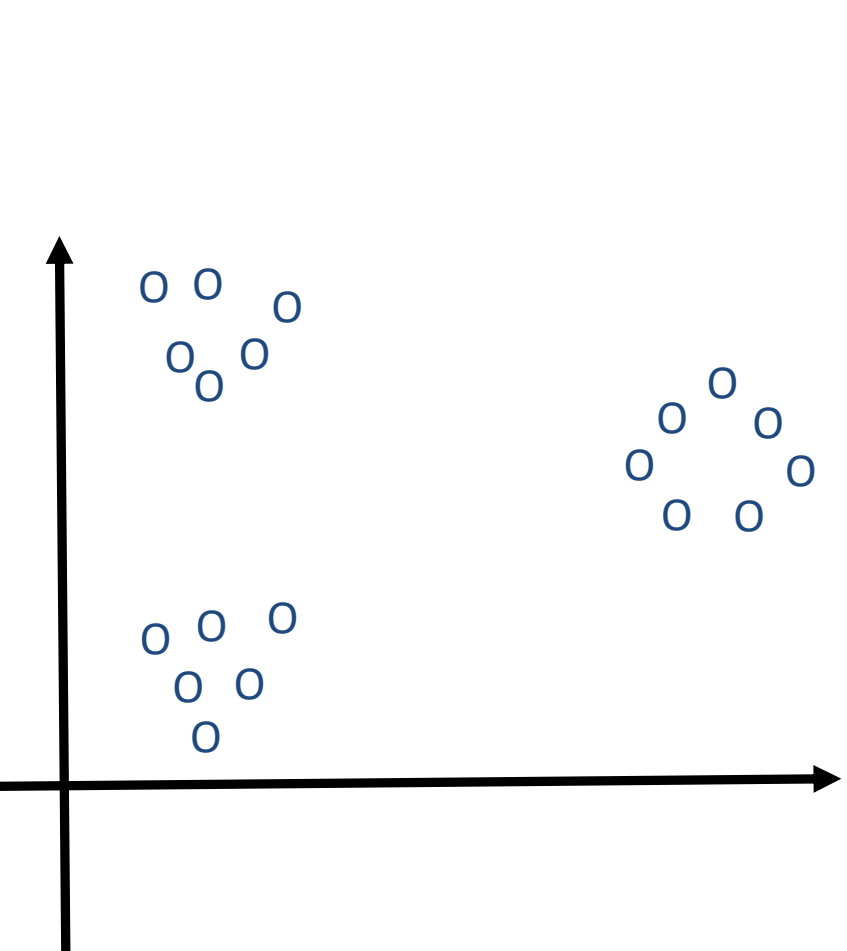
- Remember
  - c_i is the index of the cluster centroid closest to training example i
  - u_n is the cluster centroid n
  - $U_{c(i)}$ is the cluster centroid that training example x_i is connected to.

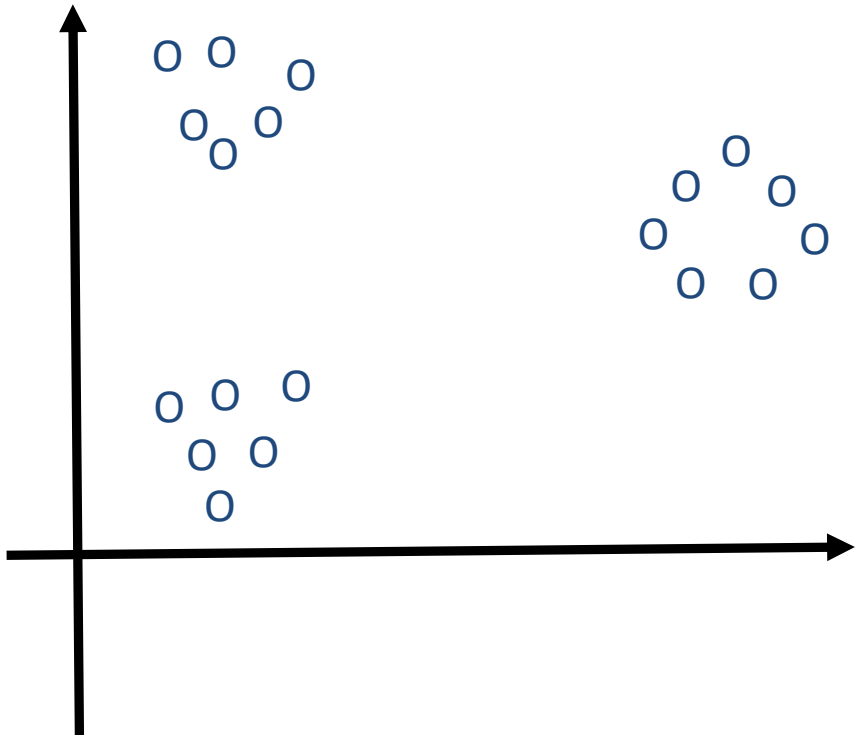
The distortion function D finds the **average** (across all *m* training examples) **squared distance** between each training example and the centroid to which it is connected to.

This function is useful as it can enable us to **monitor the performance** of our k means algorithm to make sure it is working as anticipated.

Local Optima

- KMeans is susceptible to arriving at **local optimums** (a local optima of the distortion function *D*).
- In other words it can converge incorrectly and provides a poor clustering of the training data points.





Local Optima

- KMeans is susceptible to arriving at local optimums (a local optima of the distortion function *D*). In other words it can converge incorrectly and provide a poor clustering of the training data points.

- The typical approach to solving this problem is to **run KMeans many times** and select the version that achieves the minimum distortion function value.

for 1 to 10:
    Randomly initialise Kmeans algorithm
    Run Kmeans
    Compute distortion function D
        (D(c1, c2, … cm, u1, .. , uk))

Select clustering solution that gives minimum distortion cost function value.
min D(c1, c2, … cm, u1, .. , uk)

Random Initialisation? – Not Really!

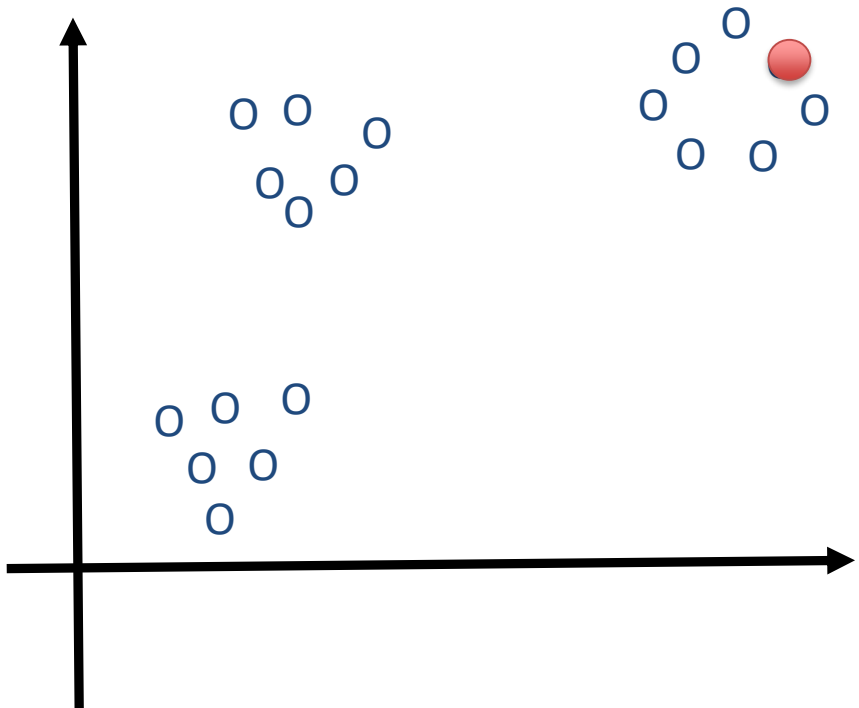
- One of the limitations of the standard k-means algorithm is that it is **very sensitive to its initialization**.

- The KMeans algorithm specifies that the centroids are randomly initialized.
- However, the random initialization is generally "**controlled**".
- If we have specified n clusters then we randomly selected **n training points** to become the centroids for each cluster.
- This has been shown empirically to outperform versions of KMeans where there is absolute random initialization but is still very sensitive.

- An alternative version of KMeans called **KMeans++** implements an initialization strategy that is more robust.
- The following study presents a comparative analysis of different initialization strategies for KMeans.

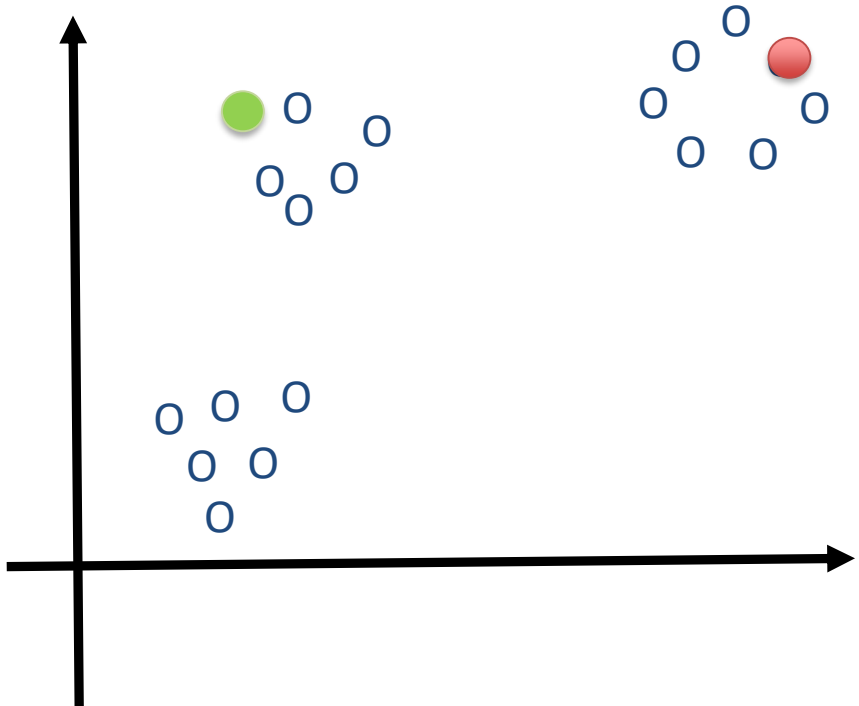
K-Means ++

- kMeans++ is a variant of kMeans that utilizes a different initialization strategy for the centroids.
- With kMeans the **probability** of being chosen as a **centroid** is related to the **minimum distance of a data points from the existing set of centroids**.
  - The closer a data point is to an existing centroid the less likely it will be chosen as a new centroid
  - The further away a data point from it's near centroid the more likely it is to be chosen as a new centroid.



K-Means ++

- kMeans++ is a variant of kMeans that utilizes a different initialization strategy for the centroids.
- With kMeans the **probability** of being chosen as a **centroid** is related to the **minimum distance of a data points from the existing set of centroids**.
  - The closer a data point is to an existing centroid the less likely it will be chosen as a new centroid
  - The further away a data point from it's near centroid the more likely it is to be chosen as a new centroid.



K-Means ++

- Starting with a dataset X of n points (x^1, x^2, … , x^n).

1. Select the first centroid $\mathbf{C_1}$ uniformly at random from the set of data points
2. Compute a vector containing the square distances between all points in the dataset and C_1:

$$D_i = || X^i - C_1 ||^2$$

3. Choose a second centroid $\mathbf{C_2}$ from X randomly based on the probability distribution defined by: $\mathbf{D_i / sum(D)}$

K-Means ++

- Starting with a dataset X of n points (x^1, x^2, … , x^n).

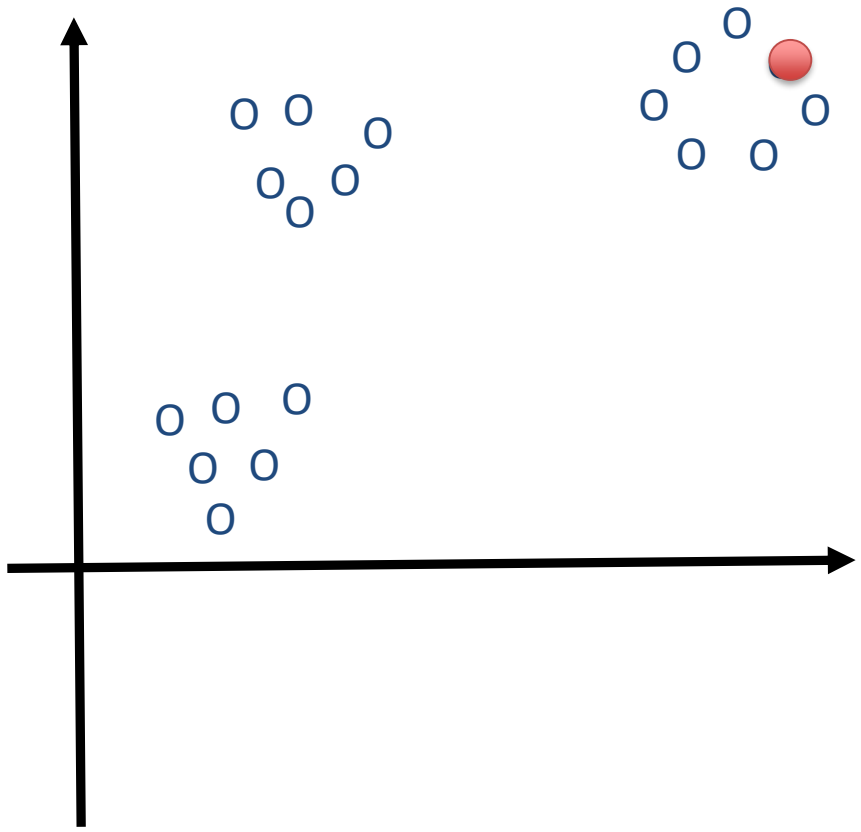
1. Select the first centroid C_1 uniformly at random from the set of data points
2. Compute a vector containing the square distances between all points in the dataset and C_1:

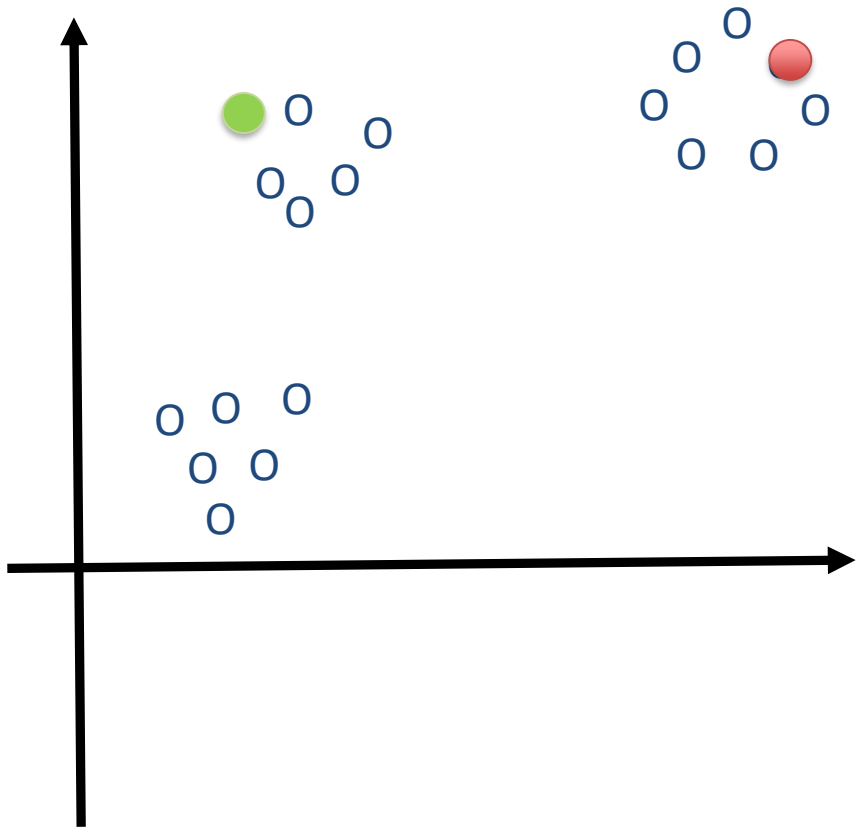
$$D_i = || X^i - C_1 ||^2$$

3. Choose a second centroid $\mathbf{C_2}$ from X randomly based on the probability distribution defined by: $\mathbf{D_i / sum(D)}$

4. Re-compute the distance vector as for each instance x^i we calculate
5. $D_i = \min(||X^i - \mathbf{C_1}||^2, ||X^i - \mathbf{C_2}||^2)$ and select C_3 as described in step 3.

…….



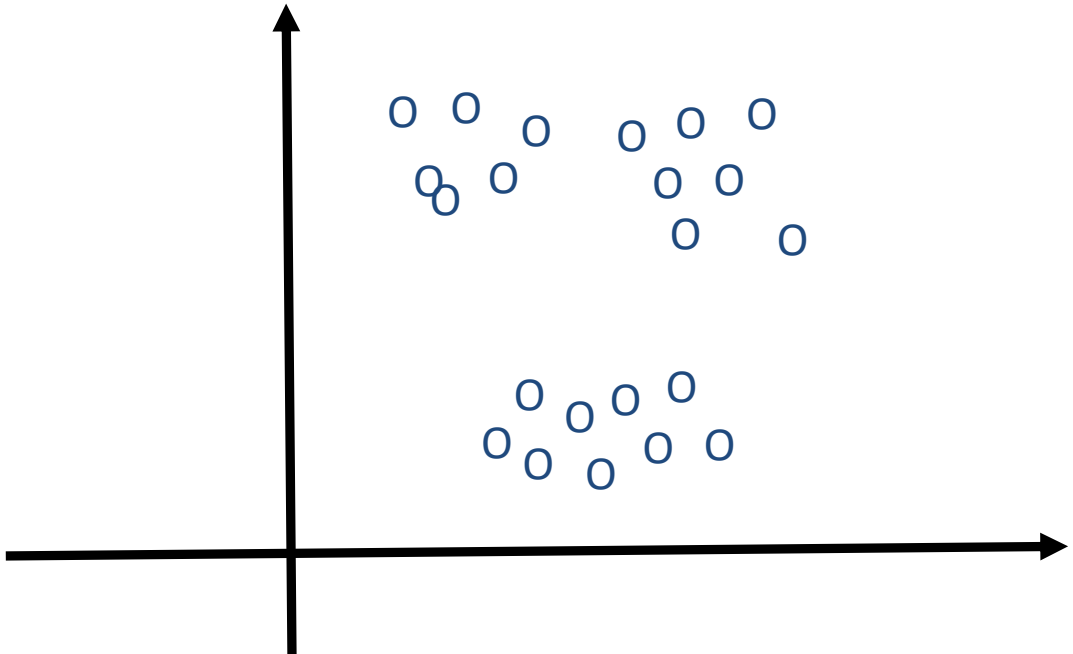


Choosing a Value of K

- Selecting an **appropriate value of K** for the K-Means algorithm is very important and can have a significant impact on the results.

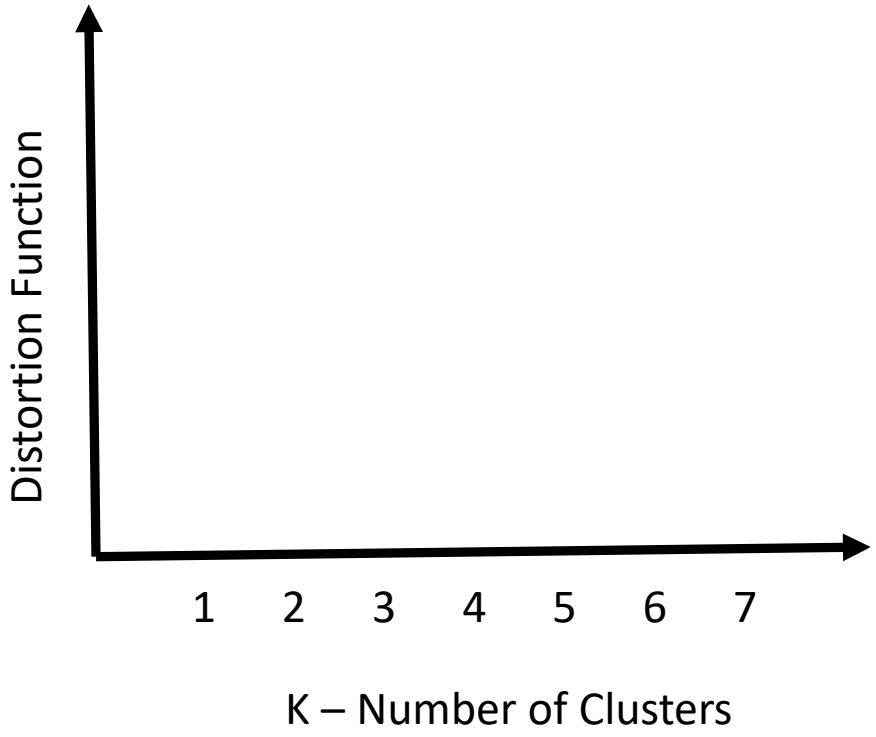
- Unfortunately there is no proven method of obtaining the optimal value for k.

- It can often be very ambiguous how many clusters there are in the data. In the example presented earlier it was very clear there were three clusters.



Choosing a Value of K – Elbow Method

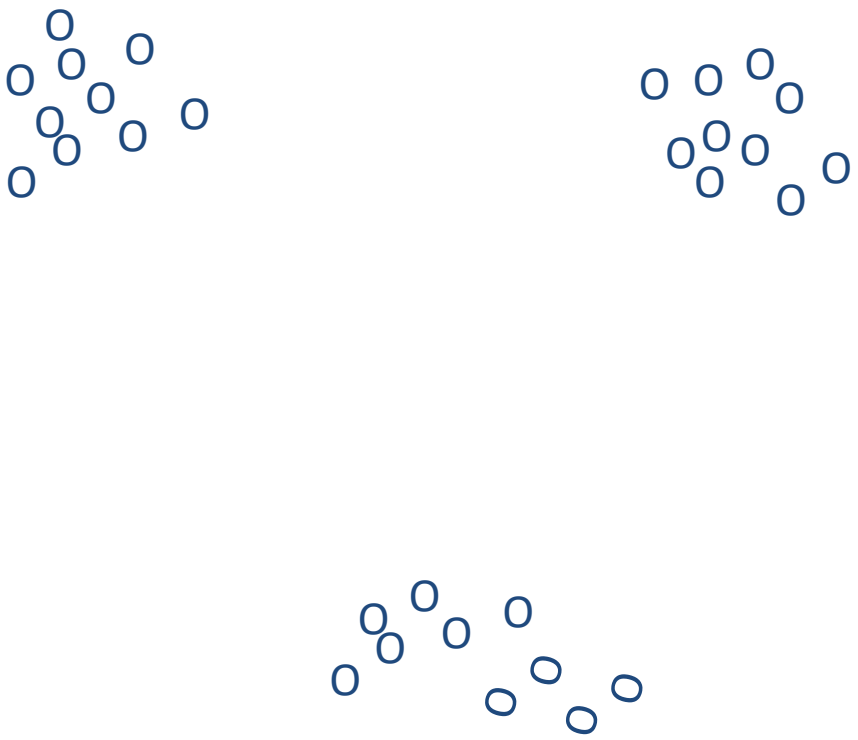
- One method that can be employed is referred to as the elbow method or elbow plot.
- It maps the **distortion value** achieved by selecting different values of k.

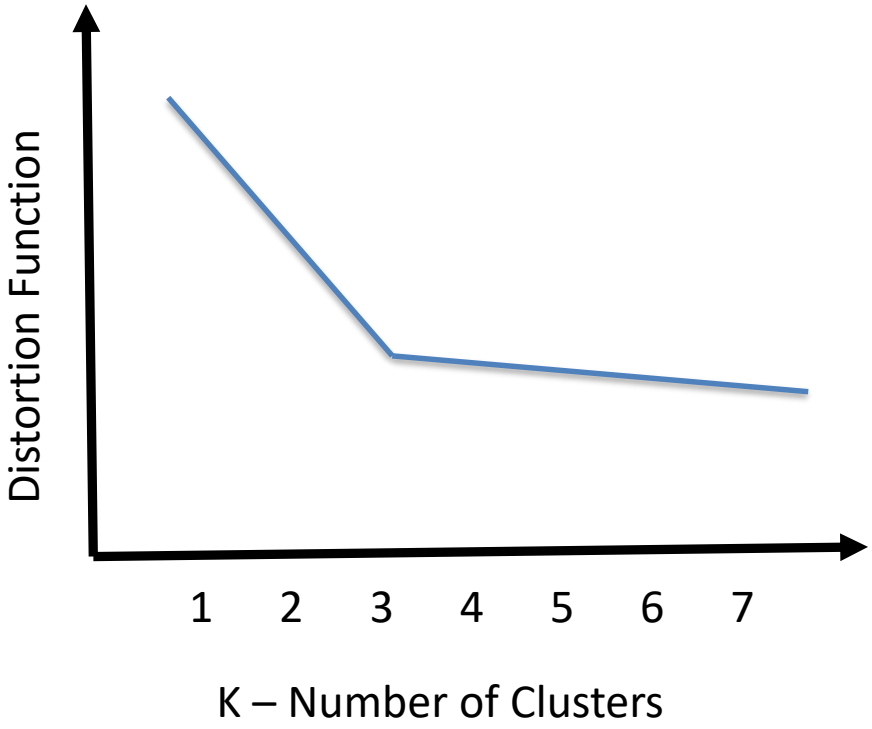


Choosing a Value of K – Elbow Method

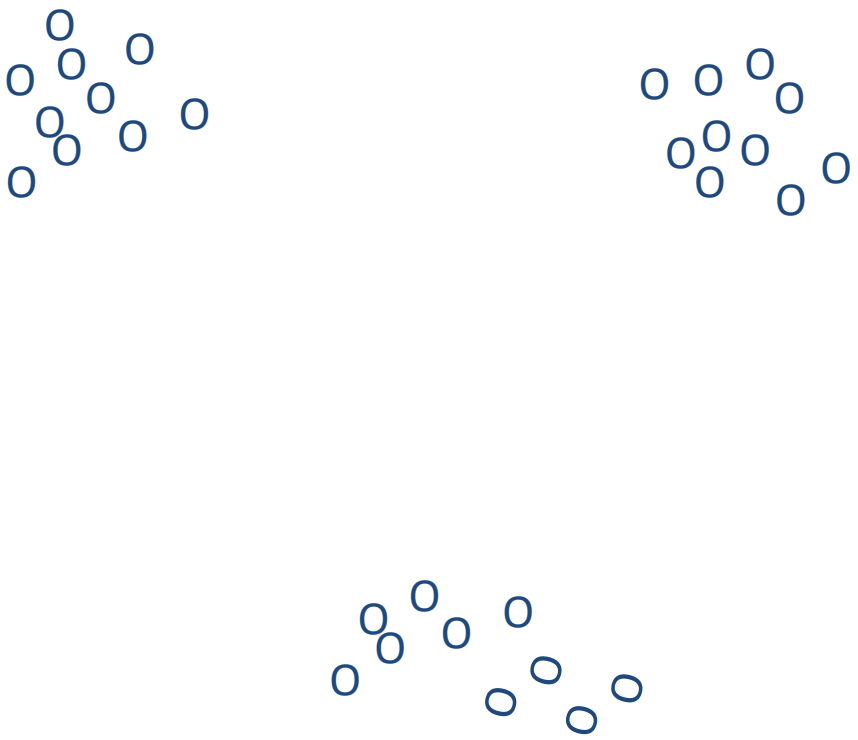
- One method that can be employed is referred to as the elbow method or elbow plot.
- It maps the **distortion value** achieved by selecting different values of k.
- If you end up with a graph such as the one below that begins to plateaux after K=3, it can give a strong indication that K = 3 is the most appropriate value of K. Note this is still not a very reliable method as the graph may often not take this ideal form.

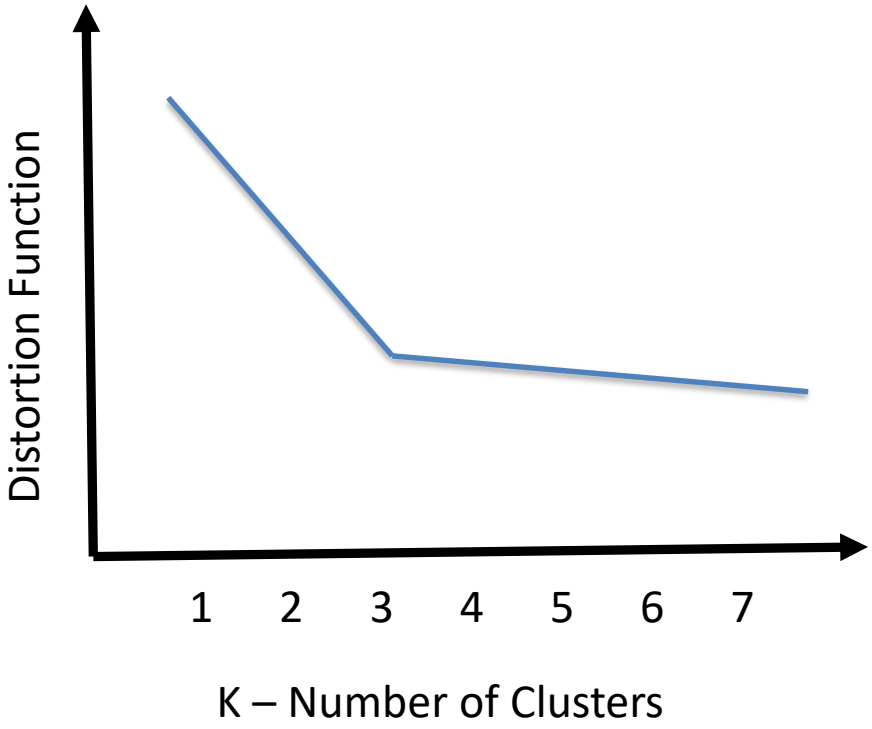
$$\frac{1}{m}\sum_{i=1}^{m}| |x^i - U_{c(i)}| |^2$$



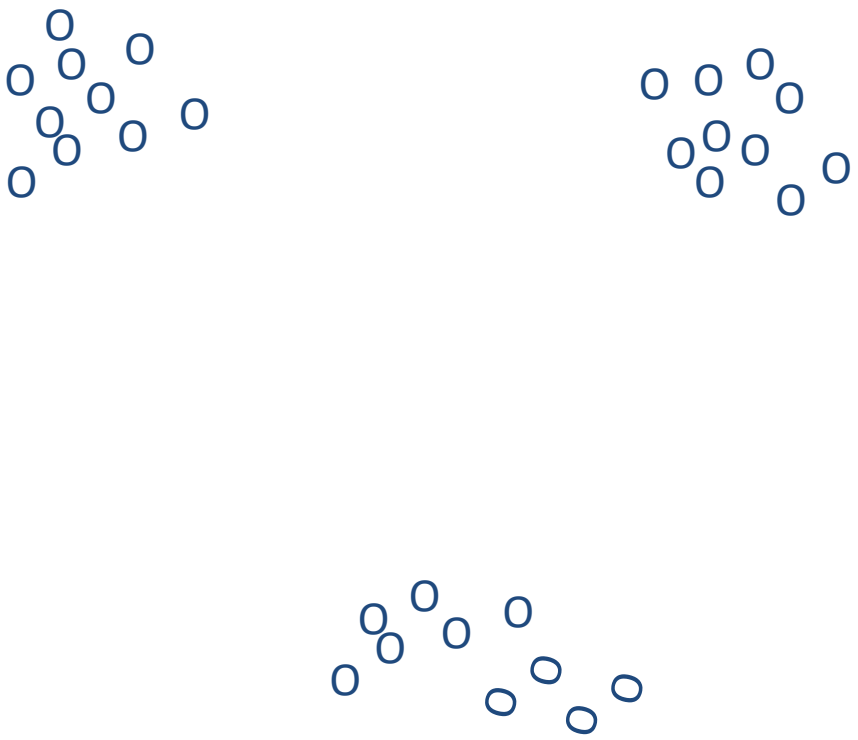


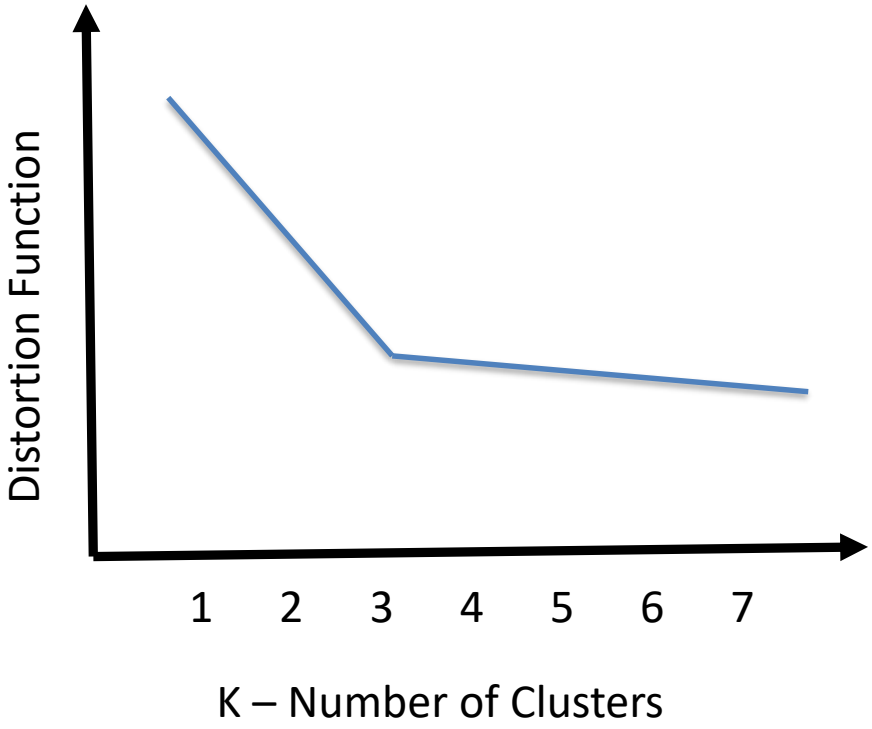
$$\frac{1}{m}\sum_{i=1}^{m}| |x^i - U_{c(i)}| |^2$$



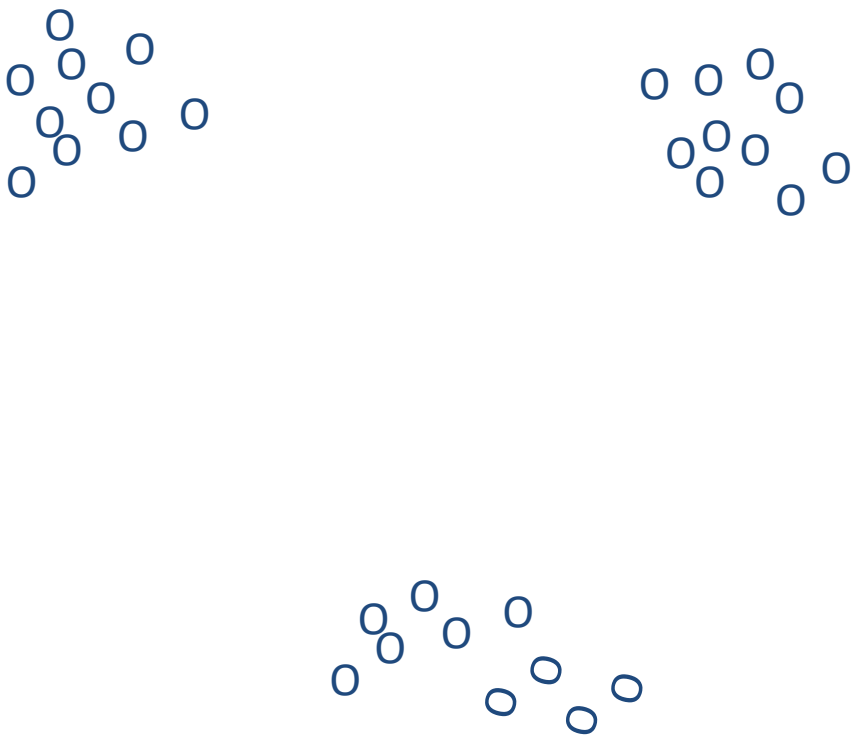


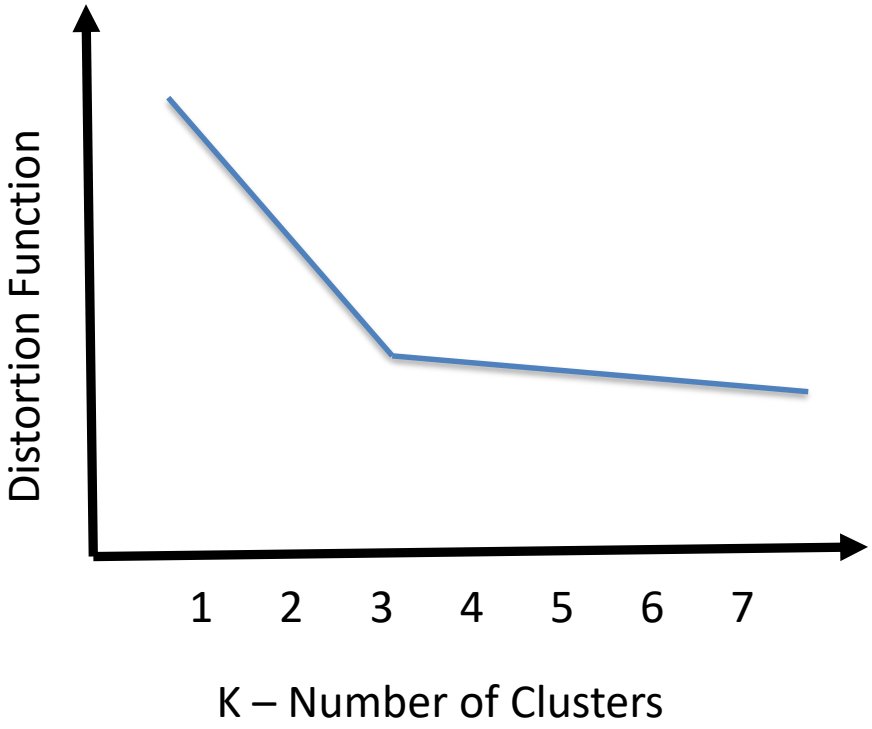
$$\frac{1}{m}\sum_{i=1}^{m}| |x^i - U_{c(i)}| |^2$$





$$\frac{1}{m}\sum_{i=1}^{m}| |x^i - U_{c(i)}| |^2$$



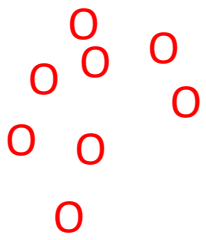


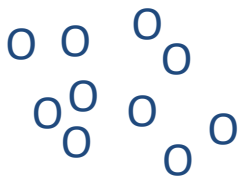
Measuring Cluster Quality

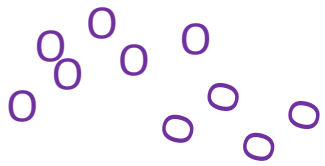
- While we can use an **elbow plot** as a rough estimate of the cluster quality of a k-means solution, often it may not exhibit the pronounced "elbow" that we are looking for. Also this approach cannot be directly applied to other clustering methods (which don't use centroids).

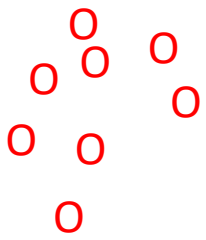
- Another common technique used is called a **Silhouette plot**. The silhouette plot looks at the relationship between two different aspects of a clustering solution.

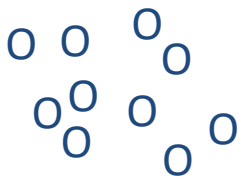
- More specifically, for each cluster it looks at:
  - The **cohesion** of each individual cluster (this is a measure of the density and ideally we want cohesion to be small … dense clusters)
  - The **cluster separation**, which looks at the distance between the current cluster and their nearest neighbouring cluster (the greater the distance the better)

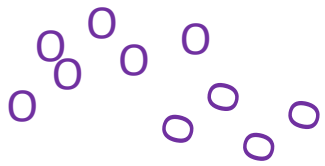












Building a Silhouette plot

- To calculate the silhouette coefficient of a single sample in our dataset, we can apply the following three steps for **each data point i** :

1. Calculate the **cluster cohesion $a^{(i)}$** as the **average distance** between a sample $x^{(i)}$ and all other points in the same cluster.

2. Calculate **the cluster separation $b^{(i)}$** from the next closest cluster as the average distance between the sample $x^{(i)}$ and all samples in the nearest cluster.

3. Calculate the **silhouette $s^{(i)}$** as shown here:

$$s^{(i)} \begin{cases} 1 - (a^{(i)} / b^{(i)}) & \text{if } a^{(i)} < b^{(i)} \\ 0 & \text{if } a^{(i)} = b^{(i)} \\ -1 + (b^{(i)} / a^{(i)}) & \text{if } a^{(i)} > b^{(i)} \end{cases}$$

Building a Silhouette plot

$1 - a^{(i)} / b^{(i)}$	if $a^{(i)} < b^{(i)}$
0	if $a^{(i)} = b^{(i)}$
$-1 + b^{(i)} / a^{(i)}$	if $a^{(i)} > b^{(i)}$

- We are ideally looking for clusters that have an $s^{(i)}$ value that is close to 1. Notice above that for $s^{(i)}$ to be close to 1 then $a^{(i)}$ must be small and $b^{(i)}$ must be large. That is, the cluster must be dense and must be quite a distance from the nearest neighbouring cluster.

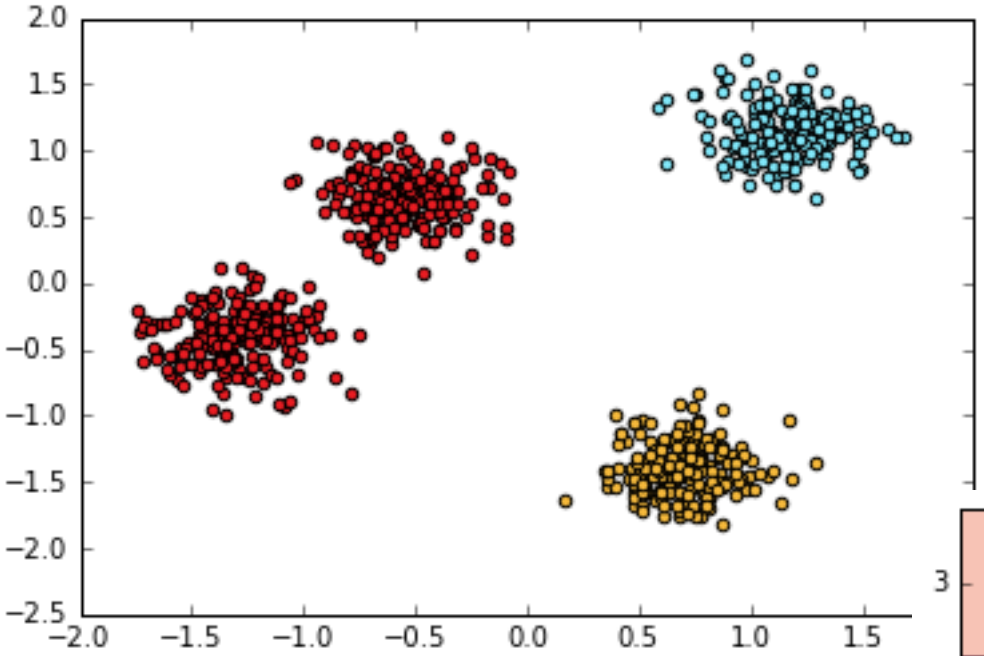
- **If we obtain an $s^{(i)}$ value close to -1, then it means $b^{(i)}$ (the average distance to the nearest cluster) is greater than $a^{(i)}$** .(the average distance to other points within the same cluster). That is the point $x^{(i)}$ should really be clustered with the nearest neighbouring cluster.

- An $s^{(i)}$ value close to zero means that the sample data point is on the border of two clusters (as it's average distance to data points in the nearest cluster and within it's own is the same).

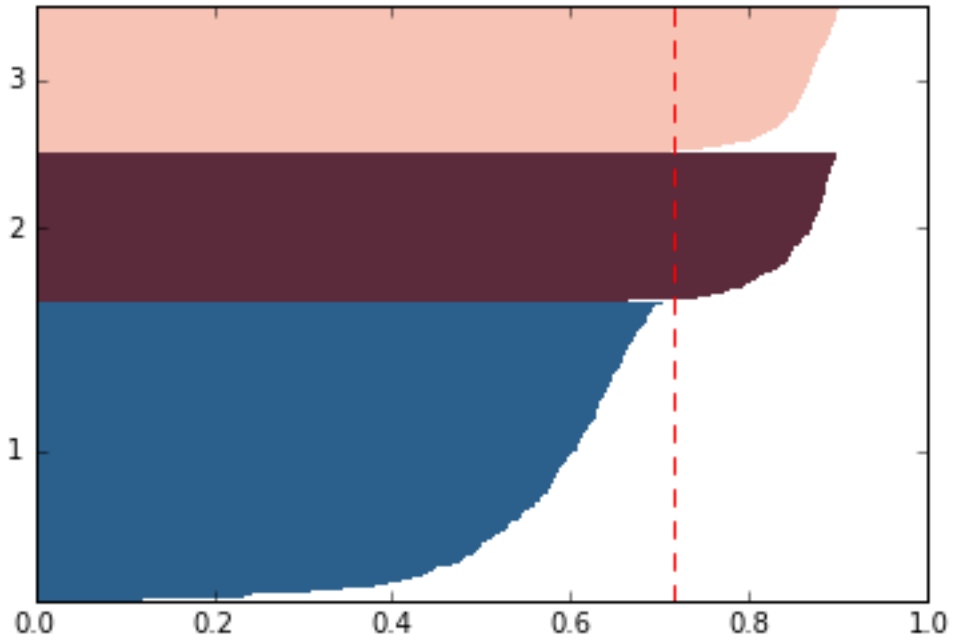
Creating a Silhouette Plot in Scikit Learning

- So for each training data point we will calculate the silhouette value.
- We can get the **mean silhouette (value)** coefficient for each cluster by **averaging** the silhouette coefficient of all data points assigned to each cluster.
- The following is a good rule of thumb for the average silhouette value.
  - **0.7 – 1.0** : A strong structure has been found
  - **0.51 – 0.7** : A reasonable structure has been found
  - **0.26 – 0.5** : A weak structure that could be artificial.
  - **<0.25** : No reasonable structure has been found in the data

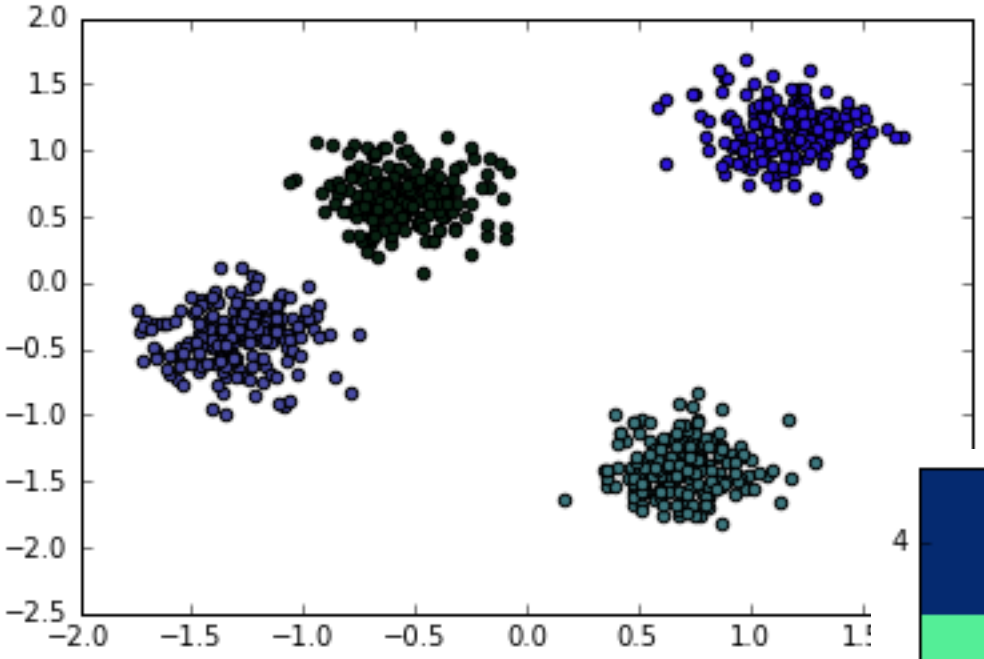
Example of a Silhouette Plot



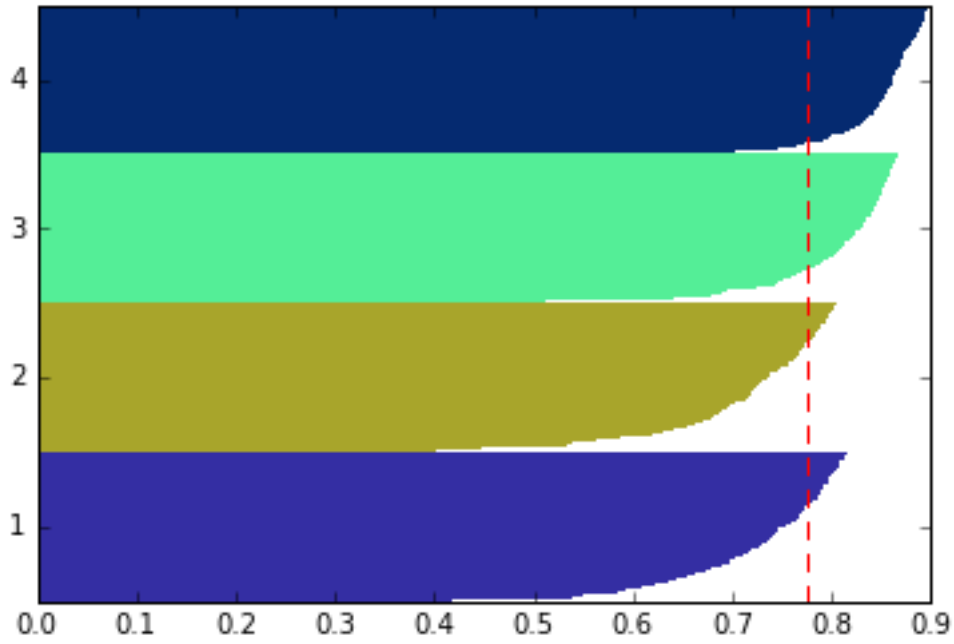
Average Silhouette Coefficient 0.715



Example of a Silhouette Plot



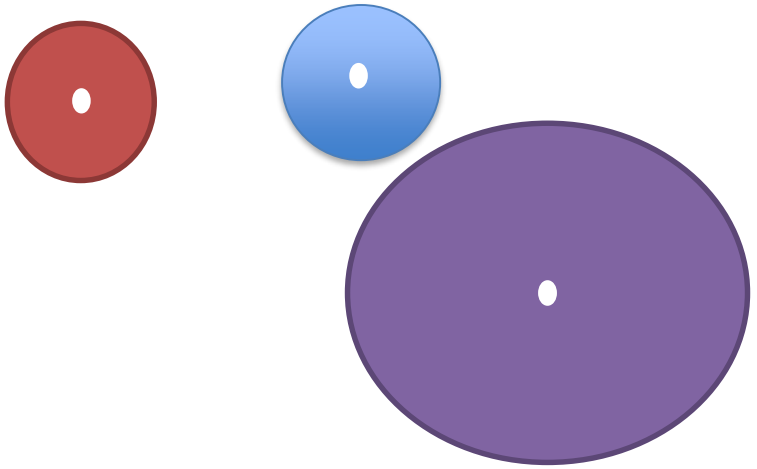
Average Silhouette Coefficient 0.775



Advantages and Disadvantages of K-Means

<u>Advantages</u>

- Relatively easy to implement.
- Easy to interpret the clustering results.
- No underlying distributional assumptions.
- Works well for large scale data



<u>Disadvantages</u>

- Can be difficult to select appropriate value for k
- Very sensitive to **initialization** strategy.
- KMeans has real difficultly when faced with clusters of **different overall size**. For example the dispersion or spread of data points within one cluster might be much greater than the spread of data points in another cluster.
- For example, we might end up with a situation as shown on the right where we have three centroids (shown as white circles). Notice that many of the points in the purple cluster are closer to the centroid in the blue cluster and as such will be assigned to that cluster.
- Susceptible to curse of dimensionality.