Initialization of K means

K means clustering in its most simple form, assigns initial cluster means randomly. Unfortunately, such an approach to initialization leads to inconsistency in the algorithms final prediction of k classes. The first section of this report will seek to show that the initialization step in k means clustering is non trivial, in that the initial positions chosen as cluster means can significantly effect the algorithms final classification of the data set into k clusters.

The initialization step for k means clustering is usually done by sampling k data points from the data set and assigning these as the initial positions for the centroids. Thus sampling from a uniform distribution where every datapoint has an equal chance of being chosen. Sometimes, one or more centroids finds what we will call a *local optimum.* This is usually brought about by centroids that are initialized close to each other, which influences the final classification of data points to an extent that the sum of squared error is large. An example of this will be demonstrated after the following assumptions and notations are observed.

Let us assume that for a particular value of k there exists *k final centroids* that will result in the data set being clustered in a way that will minimize the sum of squared error. Let us refer to these k clusters as the *k best fitting clusters* as they give a sum of squared error that is small for that particular value of k.

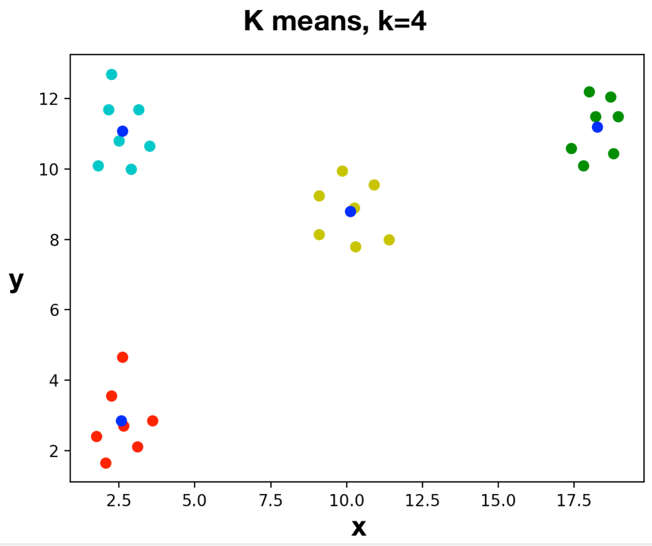
The sum of squared error can be calculated as follows:

$$SSE = \sum\limits_{i=1}^k \sum\limits_{p \in C_i} (p-m_i)^2$$

- Where k is the number of clusters
- C is the set of objects in a cluster
- M the centre point of a cluster

Fig 1.0 demonstrates a case when k=4, where the centroids find the 4 best clusters that minimize the sse.

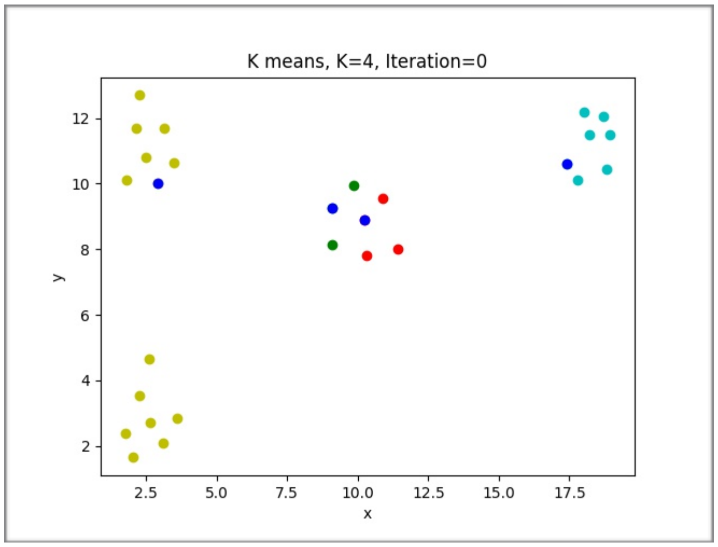
MSE = 128.12

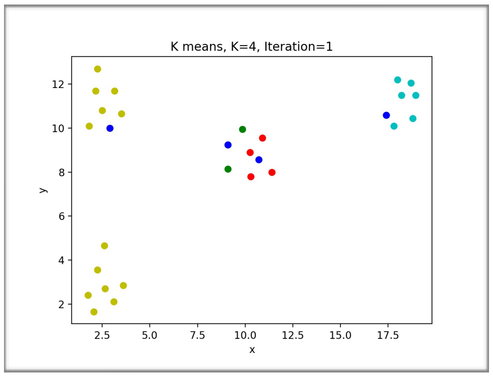


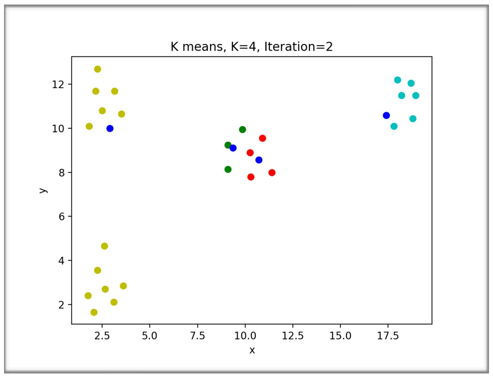
The above is an example of the best case classification for this particular dataset as it minimizes the sum of squared error for k=4. Ideally, every time the k-means algorithm is executed on this dataset for k=4 it will classify the data into 4 clusters as shown above.

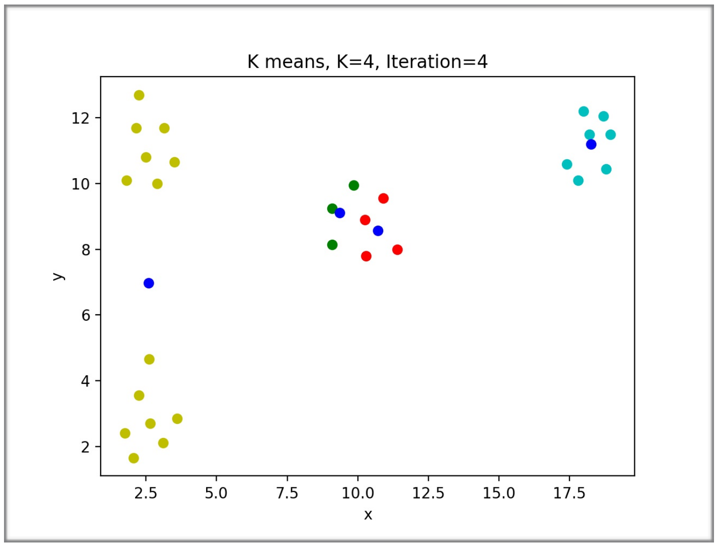
Unfortunately, this is not always the case. Fig 1.1 shows the initialization step and figures 1.2 to 1.x show the following iterations of the k means algorithm resulting in a situation where a *local optimum* (as previously mentioned) is observed.

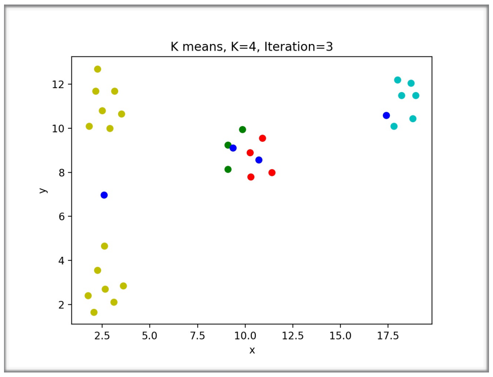
Fig 1.1 to 1.5 (clockwise)







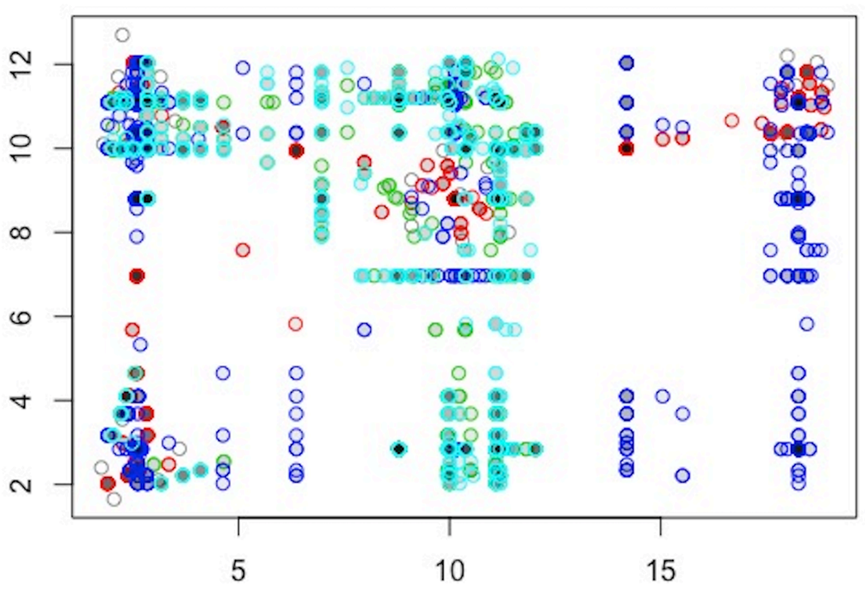




Note that the final clusters are shown on each figure from fig 1.1 to 1.5 and are distinguished by colour.

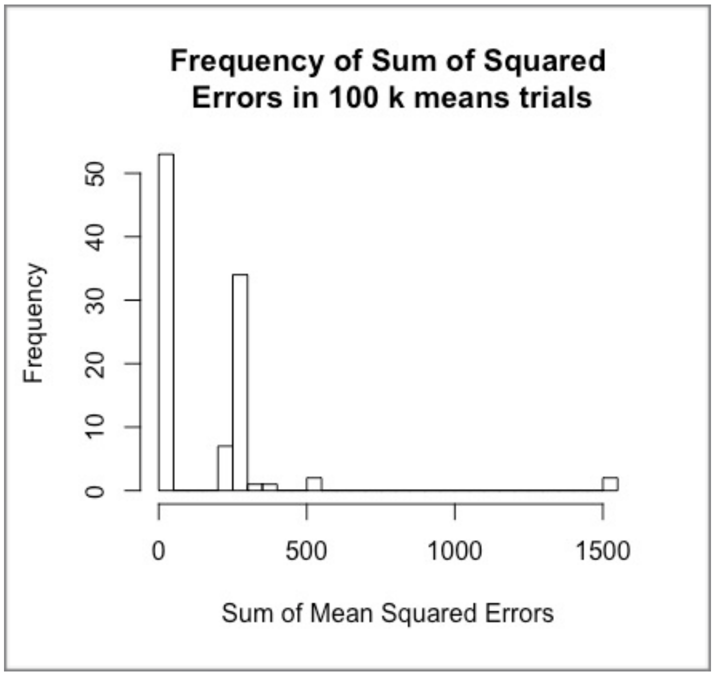
Two of the closely initialized centroids seen in the centre most cluster (FIG 1.1) do not find their way to the k best fitting clusters. Instead the two centroids remain close to each other, partitioning the data points around them into two individual clusters when they actually should have been classified into a single cluster. This means that there remains k - 2 centroids to cluster the remaining data points yet there remains k - 1 best clusters in the data, i.e there is one more best cluster than remaining centroids. This forces one of the remaining centroids to cluster two of the best fitting clusters as one single cluster, resulting in a sum of squared error that is large.

The following figure shows the final position of the centroids for the k means algorithm performed 100 times on the same data set, where the centroids are simply plotted on top of each other each time k means is performed.



It can be observed that the areas of highest densities of the final centroids for 100 trials of the k means algorithm lie over the four best clusters in the dataset. This may indicate to us that k-means has been able to find the four best clusters most of the time. That being said, there is a noticeable amount of dispersion of centroids on the figure which demonstrates that the algorithm was inconsistent in its predictions. It is the intent of this report to show that the majority of the inconsistency observed in the final positions of centroids is as a direct consequence of the method of initialization in plain k means clustering.

The corresponding figure below shows the distribution of the sum of squared errors for the 100 trials of the k means algorithm on the same data set, where each sum of squared error corresponds to the outcome of one of the executions of k means plotted in the previous figure.



Mean = 174.73

Variance = 55109.19

IQR = 239.1805

The histogram above of the sum of squared error for 100 k-means trials has large spread and quite a large variance. It is unimodal with the greatest frequency of mean squared errors around the mean. It can be deduced that there is large degree of inconsistency in the final positions of the centroids that is leading to large spread of sum of squared errors and a large mean for the sum of squared error.

K++ Initialization

K++ is a method of initialization that seeks to reduce the inconsistency and increase the accuracy of the k means algorithm with a more sophisticated initialization step. The general idea behind k++ initialization is to initialize centroids far away from each other such that two (or more centroids) are unlikely to find themselves within a single cluster where they would likely find local optimums (as previously defined).

Unlike regular k means where initial centroids are sampled from a uniform distribution where every data point has an equal chance of being selected, k++ initializes its k centroids by sampling from a weighted distribution where the probability of a data point being chosen is proportionate to its distance from all other initialized centroids. It was proposed in 2007 by David Arthur and Sergei Vassilvitskii[1].

The initialization step is performed as follows:

Let x be an element of X, a subset of the data set whose elements are closets to the last chosen centroid. Then let D(x) denote the shortest distance from a data point to the last centre we have already chosen.

1) Choose the first centre uniformly at random from among the data points.

2) For each data point x, compute D(x) the distance between x and the nearest centre that has already been chosen.

3) Choose one new data point at random as a new centre, using a weighted probability distribution where a point *x* is chosen with probability of choosing x proportional to $\frac{D(x)^2}{\sum_{x \in \mathcal{X}} D(x)^2}$, I.e how far away x is from the last centroid over the sum of the distances of all the points closest to the last centroid.

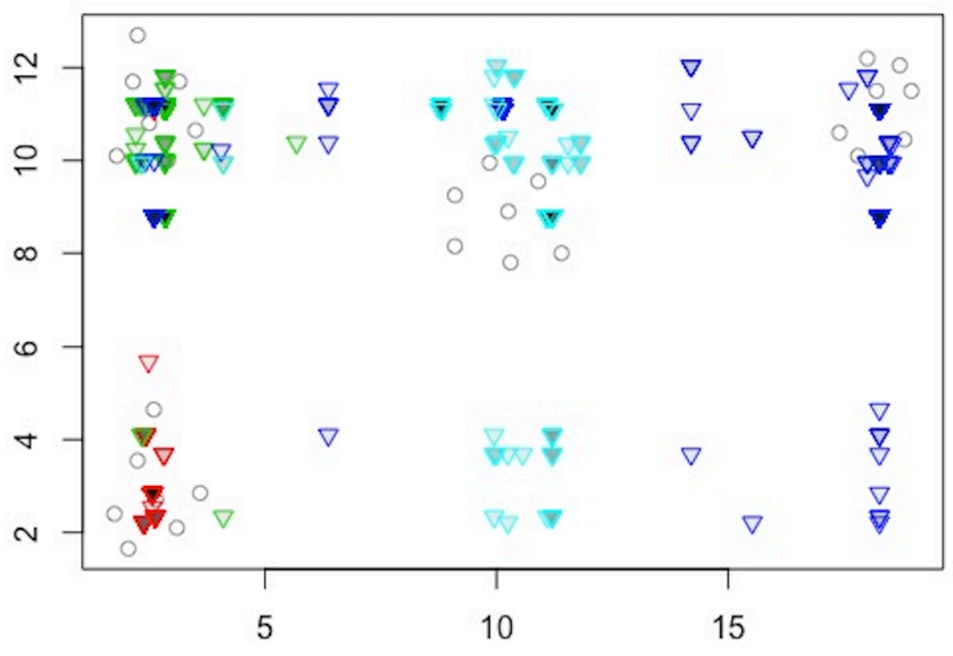
4) Repeat Steps 2 and 3 until k centers have been chosen.

5) Now that the initial centers have been chosen, proceed using standard k-means.

K ++ means uses a more sophisticated method of initialization that requires more computation than regular k means. Despite this, k++ means usually performs quicker than regular k means as it reduces the subsequent computation required for the means to reach their final positions. This is because k++ initialization places the centroids close to their final positions which for the same reason reduces the likelihood of finding a local optimum.

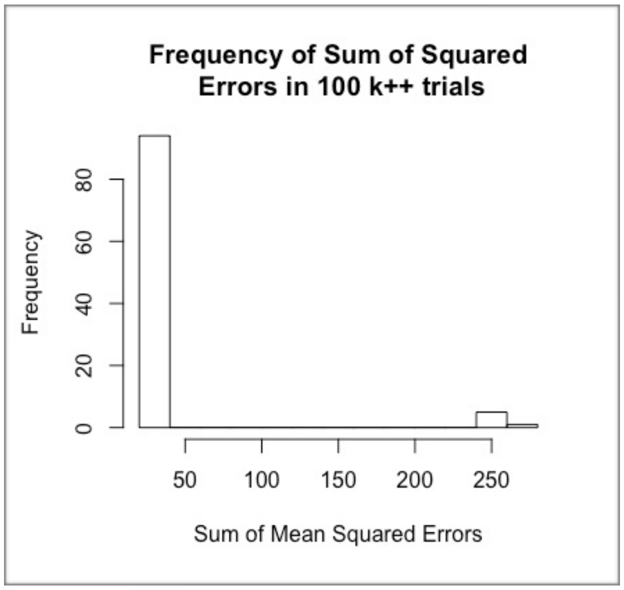
In order to perform a direct comparison between k++ means initialization and plain k means, the figures previously used have been replotted but this time using k++ initialization. The following figure shows the final position of the centroids for the k means algorithm with k++ initialization performed 100 times on the same data set, where the centroids are simply plotted on top of each other each time k means is performed.

It should be quite apparent that there is much less dispersion of centroids when k++ initialization is performed. While there still exists centroids that lie well outside any cluster



of data points, occurrences of this are far reduced compared to regular k means initialization. The centroids within their respective clusters are more densely situated which shows us that k++ initialization results in greater consistency in the final position of centroids.

The corresponding figure below shows the distribution of the sum of squared errors for the 100 trials of the k means algorithm with k++ initialization on the same data set, where each sum of squared error corresponds to the outcome of one of the executions of the k means algorithm plotted in the previous figure.



Mean = 44.06

Variance = 2735.90

IQR = 0

Compared to the similar histogram for regular k means. It is immediately apparent that the distribution of sum of squared errors is tighter than with regular k mans initialization. A greater amount of the mean squared errors are centred around their mean which suggests that this method of initialization fits the data more consistently, this is supported a large drop in variance. The actual mean of the sum of squared errors if drastically less, meaning that the clustering algorithm fit the data better more frequently. From the analysis of both methods of initialization performed, it seems that k++ yields far better results.