# Clustering methods: Kmeans

PGDM Research and Analytics cell
Madras School of Economics
Reference - An introduction to Statistical Learning


## Introduction to Clustering

**Clustering** methods are **unsupervised learning methods** that help us find subgroups or clusters in a dataset of many observations. The observations in one particular cluster tend to be **similar** to each other.

## K-means Clustering

This is an algorithm that partitions our dataset into $K$ distinct clusters or groupings that are **non overlapping** - which means that an observation can only belong to one cluster. Some notations that we will for further reference:

1. We consider that there are $n$ observations or data points, each having a set of $p$ measurements/features/variables.
2. We let $C_1, C_2, \cdots, C_K$ represent the $K$ clusters where all the $n$ obervations are allocated.
3. If the $i^{th}$ observation belongs to the $k^{th}$ cluster, then we write $i \in C_k$.

### A good clustering

A good clustering method is one for which the **within cluster variation** is as small as possible. The within cluster variation for a particular cluster, denoted as $W(C_k)$ - measures the amount by which observations within a cluster differ from each other. The objective is this - **We want to select $K$ clusters such that the total within cluster variation, summed over all $K$ clusters is minimum**.

$$\min_{C_1,\cdots,C_K} \left( \sum_{k=1}^{K} W(C_k) \right)$$

Within cluster variation is measured in terms of **Euclidean distance** given by the following equation:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (X_{ij} - x_{i'j})^2$$

Note that $|C_k|$ denotes the number of observations in the $k^{th}$ cluster.

## The algorithm

1. We first randomly assign 1 till $K$ cluster numbers to all observations - these are randomized initial cluster assignments.
2. Then for each of the $K$ clusters, we find the **centroid** - the centroid is a vector of $p$ feature/variable means for the observations in the $k^{th}$ cluster.
3. Then we assign each individual observation to the cluster for which the **centroid is closest to the particular observation** - where **closeness** is measured by Euclidean distance.
4. We keep repeating the steps 2 and 2 until the cluster assignments **don't change** any further.
5. With each iteration, the value of our objective function is progressively minimized - recall that our objective function is nothing but the summation of within cluster variation.
6. Note that it is crucial in Kmeans to specify the number of clusters before hand. Also - since initial cluster assignments are essentially **random** - we must ideally run this algorithm multiple times to ascertain some amount of coherency in the clusterings.

## Kmeans Simulation

First we simulate some random data upon which we will perform Kmeans clustering - we essentially simulate a dataset with 50 random observations over 2 features.

```
set.seed(2)
x = matrix(rnorm(50*2), ncol = 2)

### shifting the mean (+3) of the 1st 25 obs in 1st col
x[1:25, 1] = x[1:25, 1] + 3

### shiting the mean (-4) of the 2nd 25 obs in 2nd col
x[1:25, 2] = x[1:25, 2] - 4
```

```r
# taking a look at the data
x
```

```
##              [,1]         [,2]
##  [1,]  2.10308545 -4.838287148
##  [2,]  3.18484918 -1.933698644
##  [3,]  4.58784533 -4.562247053
##  [4,]  1.86962433 -2.724284488
##  [5,]  2.91974824 -5.047572627
##  [6,]  3.13242028 -5.965878241
##  [7,]  3.70795473 -4.322971094
##  [8,]  2.76030198 -3.064137473
##  [9,]  4.98447394 -2.860770197
## [10,]  2.86121299 -2.328381233
## [11,]  3.41765075 -5.788242207
## [12,]  3.98175278 -1.968757481
## [13,]  2.60730464 -4.703144333
## [14,]  1.96033102 -3.841835237
## [15,]  4.78222896 -3.493765203
## [16,]  0.68893092 -4.819995106
## [17,]  3.87860458 -5.998846995
## [18,]  3.03580672 -4.479292591
## [19,]  4.01282869 -3.915820096
## [20,]  3.43226515 -4.895486611
## [21,]  5.09081921 -4.921275666
## [22,]  1.80007418 -3.669550497
## [23,]  4.58963820 -4.141660809
## [24,]  4.95465164 -3.565152238
## [25,]  3.00493778 -4.053722626
## [26,] -2.45170639 -0.907110376
## [27,]  0.47723730  1.303512232
## [28,] -0.59655817  0.771789776
## [29,]  0.79220327  1.052525595
## [30,]  0.28963671 -1.410038341
## [31,]  0.73893860  0.995984590
## [32,]  0.31896040 -1.695764903
## [33,]  1.07616435 -0.533372143
## [34,] -0.28415772 -1.372269451
## [35,] -0.77667527 -2.207919779
## [36,] -0.59566050  1.822122519
## [37,] -1.72597978 -0.653393411
## [38,] -0.90258448 -0.284681219
## [39,] -0.55906191 -0.386949604
## [40,] -0.24651257  0.386694975
## [41,] -0.38358623  1.600390852
```

```
## [42,] -1.95910318  1.681154956
## [43,] -0.84170506 -1.183606388
## [44,]  1.90354747 -1.358457254
## [45,]  0.62249393 -1.512670795
## [46,]  1.99092044 -1.253104899
## [47,] -0.30548372  1.959357077
## [48,] -0.09084424  0.007645872
## [49,] -0.18416145 -0.842615198
## [50,] -1.19876777 -0.601160105
```

Now we can perform Kmeans clustering with $K = 2$. Why have we specified $K = 2$? Recall that in the previous **code block** we shifted the mean of the **1st twenty five observations of feature 1 (column 1)** observations by $+3$ and the mean of the **1sd twenty five observations of feature 2 (column 2)** observations by $-4$ - this was to essentially make the **1st twenty five and 2nd twenty five observations across both columns** to be distinct. Now we will **validate the Kmeans algorithm** if it actually splits our data into these two distinct groups.

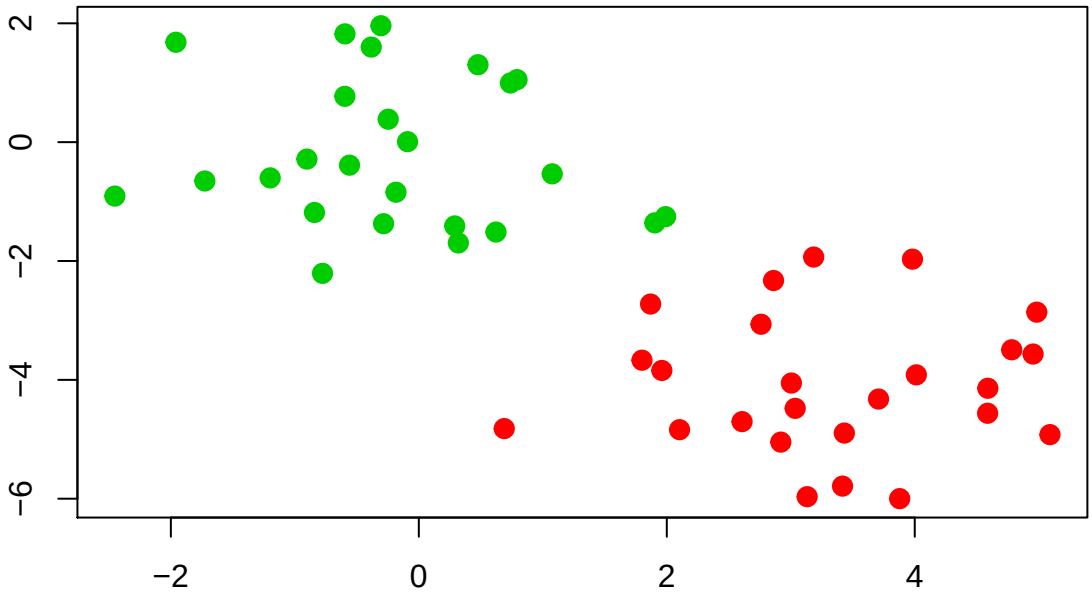
```
# nstart is the number of random assignments
km.out = kmeans(x, 2, nstart = 20)

# viewing the cluster assignments
km.out$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2
```

```
# plotting the data with colors representing clusters

plot(x, col=(km.out$cluster+1), xlab="", ylab="", pch=20, cex=2)
```



```r
# getting a summary output
km.out
```

```
## K-means clustering with 2 clusters of sizes 25, 25
##
## Cluster means:
##         [,1]       [,2]
## 1  3.3339737 -4.0761910
## 2 -0.1956978 -0.1848774
##
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 63.20595 65.40068
##  (between_SS / total_SS =  72.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# viewing the within cluster variation
km.out$tot.withinss
```

```
## [1] 128.6066
```

We can essentially run this Kmeans algorithm with variables **nstart** values (signifying different randomized initialization of clusters) and then compare the **within cluster variation** from each specification - we ultimately go witht he model giving us **least within cluster variation**.