Introduction to Data Science CS61 June 12 - July 12, 2018

Dr. Ash Pahwa

Lesson 9: Clustering

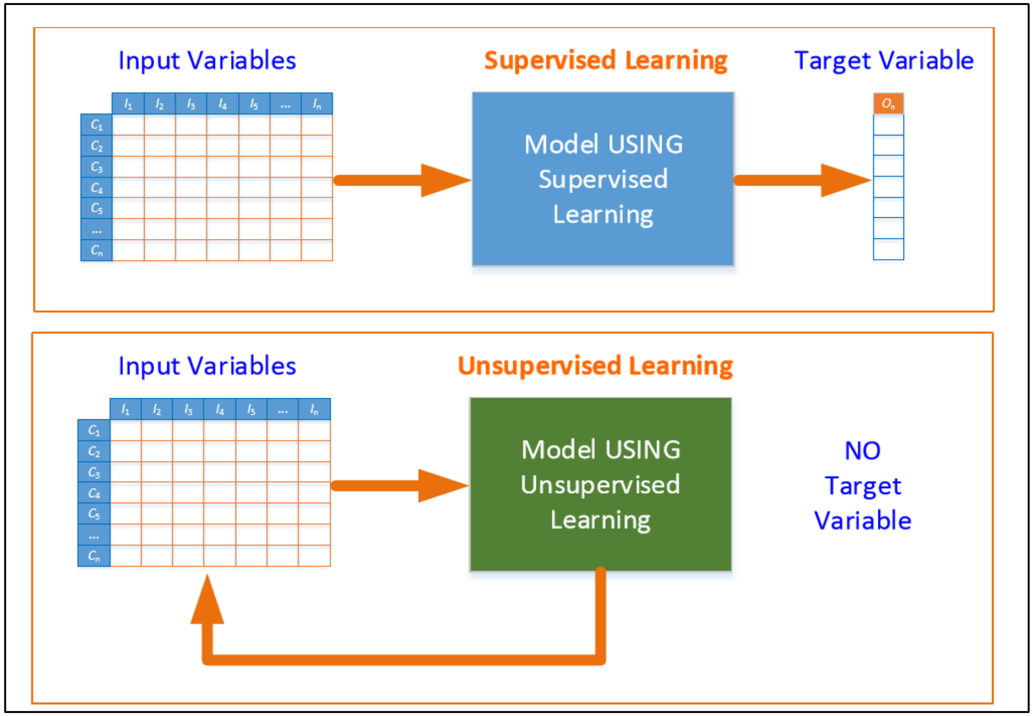
Lesson 9.1: Clustering - kMeans

Outline

- What is Clustering
- Computing Within-Cluster-Variation
- K-means Algorithm
- K-means Clustering in R
- K-means Clustering in
  - Python/Scikit-Learn

Supervised vs. Unsupervised Learning in PA

- Supervisor learning is the most common learning type where there is a target/output variable (which is also called supervisor)
  - Supervisor (target variable) teaches the algorithm how to build/learn the pattern model
  - In PA, supervised learning ≈ predictive modeling
- Unsupervised learning has NO target variable
  - No supervisor to teach → algorithm has to learn by itself
  - In PA, unsupervised learning ≈ descriptive modeling



Unsupervised Learning

- Clustering
  - K-Means Clustering
  - Hierarchical Clustering
- Principal Component Analysis

Unsupervised Learning in Predictive Analytics

- Unsupervised learning is part of Machine Learning family of methods
- Although, it may not be as popular as supervised learning, it has a significant footprint in Analytics
  - Clustering Application
    - Customer Segmentation

Clustering Applications

- Decrease the size and complexity of problems for other data mining methods
- Identify outliers in a specific domain

Business Applications of Clustering

- Procter & Gamble want to test new cosmetic products
  - Create clusters of cities which has similar demographics
    - %Asians, %Blacks, %Hispanics etc.
- Coca-Cola test the new drink
  - Create clusters of cities
    - Consumer preference of soft drink market
- Eli Lilly test their new drug
  - Create clusters of doctors
    - Number of prescriptions they write

Difference between kNN Classifier (kNN Nearest Neighbor) & k-Means Clustering

- kNN
  - Classification Method
  - If a new object is given
    - Model can predict which class that object belongs to
- Algorithm
  - kNN – k Nearest Neighbor
- Supervised Learning Method

- K-Means
  - Clustering Method
  - Group a bunch of objects into clusters
- Algorithm
  - K-Means
- Unsupervised Learning Method

How Many Clusters?

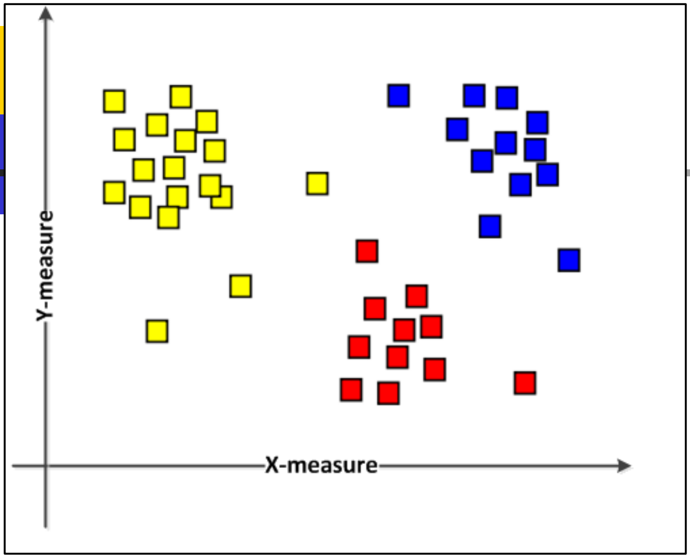


Image 1

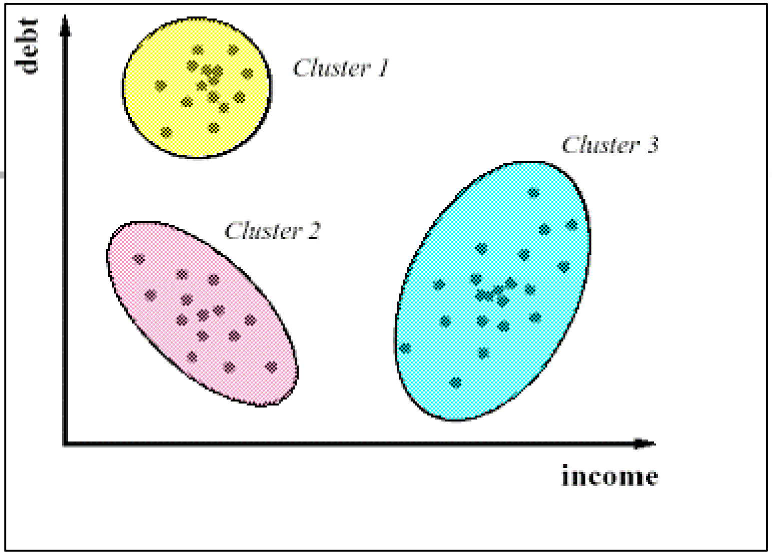
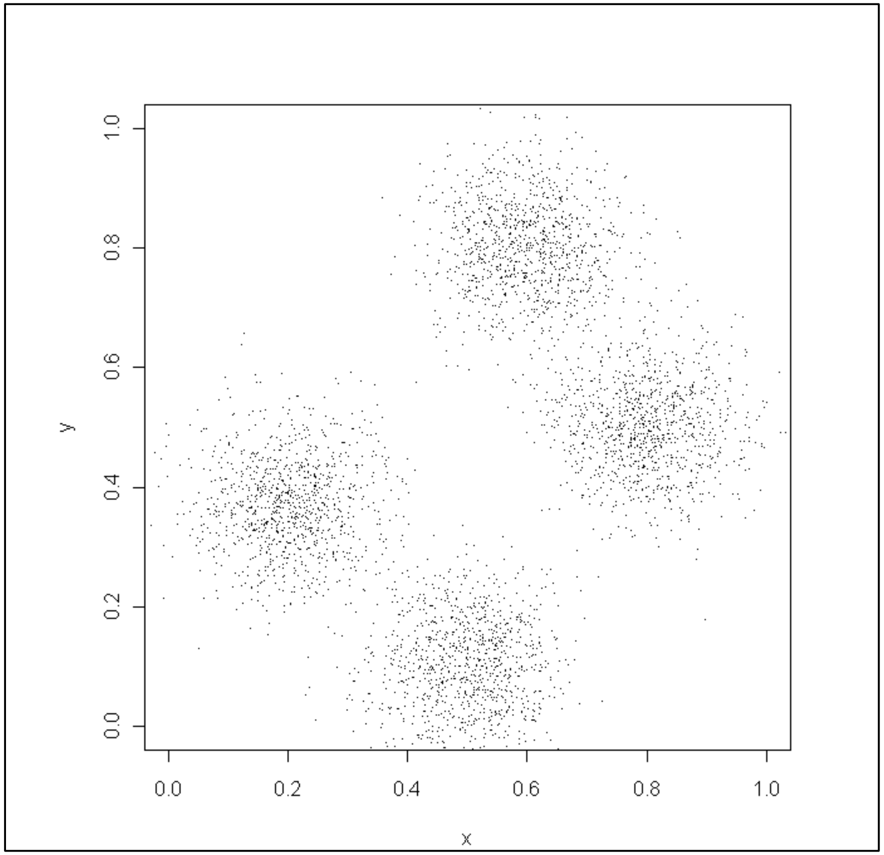


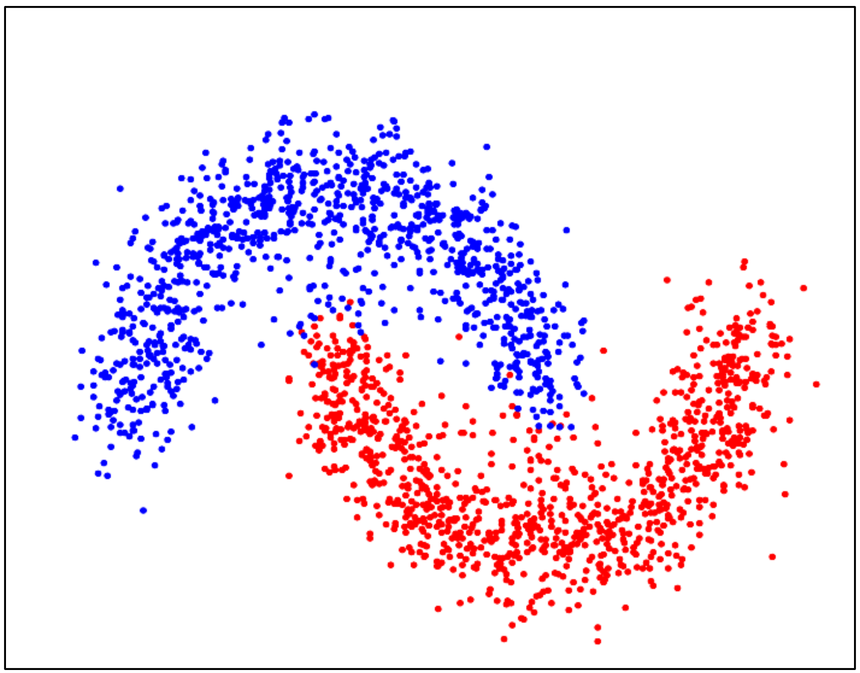
Image 2

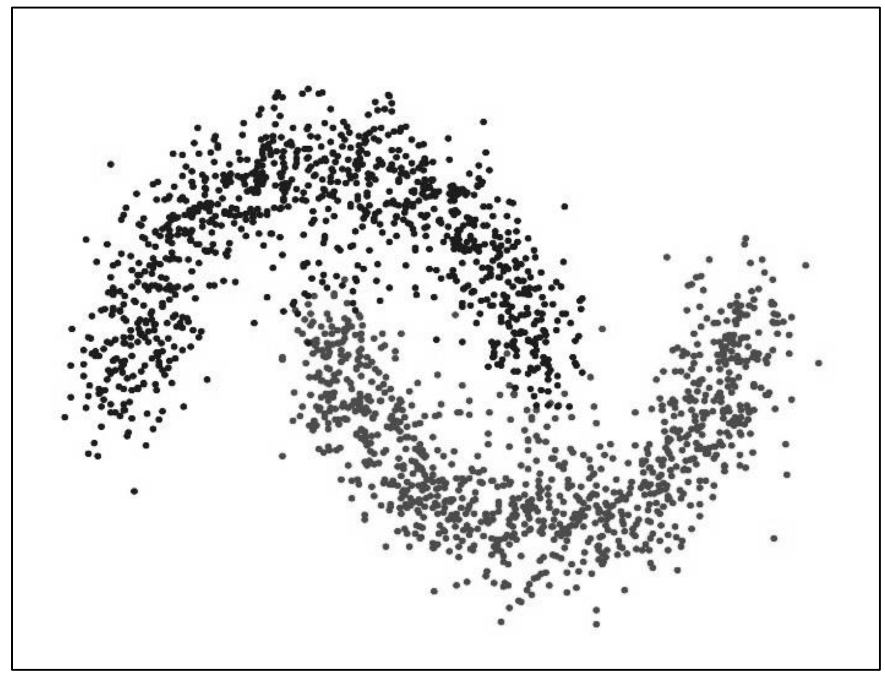
Image 3

How Many Clusters?



How many Clusters?
If Color is not the Criteria





What is Clustering?

- Categorize objects into groups (or clusters) so that
  - Objects in each group are similar
  - Objects in each group are different from objects in other groups

The Challenge of Unsupervised Learning

- Model assessment
  - We cannot tell if the model we have built is good
    - Because we do not have the test data with known response variable information
    - We cannot do cross validation

Clustering Assessment

Computing

Within-Cluster-Variation

Clustering Definition

- Suppose 'n' observations
- $Let\ C_1, C_2, \ldots, C_k\ are\ sets\ containing$
- $the\ indices\ of\ the\ observations\ in\ each\ cluster$
  - $C_1 \cup C_2 \cup C_3 \ldots \cup C_k = \{1, \ldots, n\}$. Each observation belongs to at least one of the 'k' clusters.
  - $C_k \cap C_{k'} = 0\ for\ all\ k \neq k'$. Clusters are non-overlapping: no observation belongs to more than one cluster.

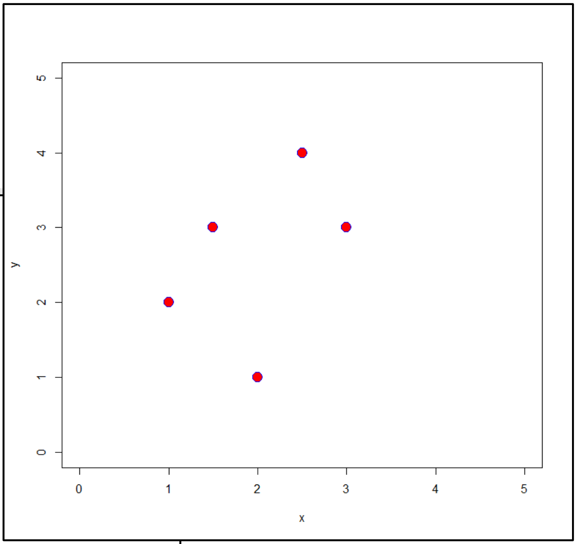
Clustering Assessment

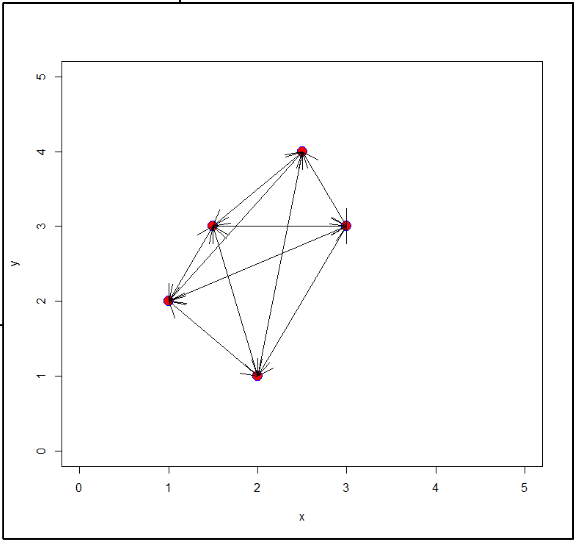
- How to assess a cluster?
  - A good cluster should have the *within-cluster-variation* is as small as possible.
    - $Within - cluster - variation = \ W(C_k)$
    - $Good\ cluster : minimize\{\sum_1^k W(C_k)\}$
    - $W(C_k) = \frac{1}{|C_k|} \sum_{i,i'} \sum_{j=1}^{p} \left(x_{ij} - x_{i^i j}\right)^2$

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

Within-Cluster-Variation

```
> ############################################
> # Within Cluster Variation
> # Euclidian Distance
> #
> rm(list=ls(all=TRUE))
> #install.packages("proxy")
> library(proxy)
> x = c(2,1,1.5,2.5,3)
> y = c(1,2,3,4,3)
> plot(x,y,pch=21,col="blue",bg="red",xlim=c(0,5),ylim=c(0,5),cex=2)
>
> for ( i in 1:5 ) {
+   if ( i != 1 )  { arrows(x[i],y[i],x[1],y[1]) }
+   if ( i != 2 )  { arrows(x[i],y[i],x[2],y[2]) }
+   if ( i != 3 )  { arrows(x[i],y[i],x[3],y[3]) }
+   if ( i != 4 )  { arrows(x[i],y[i],x[4],y[4]) }
+   if ( i != 5 )  { arrows(x[i],y[i],x[5],y[5]) }
+ }
```

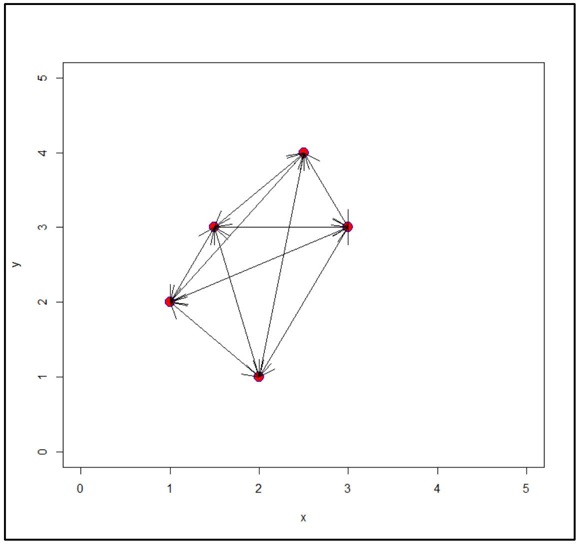




Within-Cluster-Variation

- $W(C_k) = \frac{1}{|C_k|} \sum_{i,i'} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2$
- $Good\ cluster : minimize\{\sum_1^k W(C_k)\}$
- $minimize\left\{\frac{1}{|C_k|} \sum_{i,i'} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2\right\}$

```
> ###########################################
> df1 = data.frame(x,y)
>
> (d = dist(df1, df1, method = "euclidean"))
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 0.000000 1.414214 2.061553 3.041381 2.236068
[2,] 1.414214 0.000000 1.118034 2.500000 2.236068
[3,] 2.061553 1.118034 0.000000 1.414214 1.500000
[4,] 3.041381 2.500000 1.414214 0.000000 1.118034
[5,] 2.236068 2.236068 1.500000 1.118034 0.000000
> (square.dist = d^2)
     [,1] [,2] [,3] [,4] [,5]
[1,] 0.00 2.00 4.25 9.25 5.00
[2,] 2.00 0.00 1.25 6.25 5.00
[3,] 4.25 1.25 0.00 2.00 2.25
[4,] 9.25 6.25 2.00 0.00 1.25
[5,] 5.00 5.00 2.25 1.25 0.00
> ##########################################
> # Since the distance matrix contain all the numbers twice
> # Divide the total sum by 2
> (sum.square.dist = sum(square.dist)/2)
[1] 38.5
> (within.cluster.variation = sum.square.dist/length(x))
[1] 7.7
```



k-means Algorithm

K-means Algorithm

- Given a *K*, find a partition of *K clusters*
- *K-means* algorithm

  K-means algorithm (MacQueen'67):

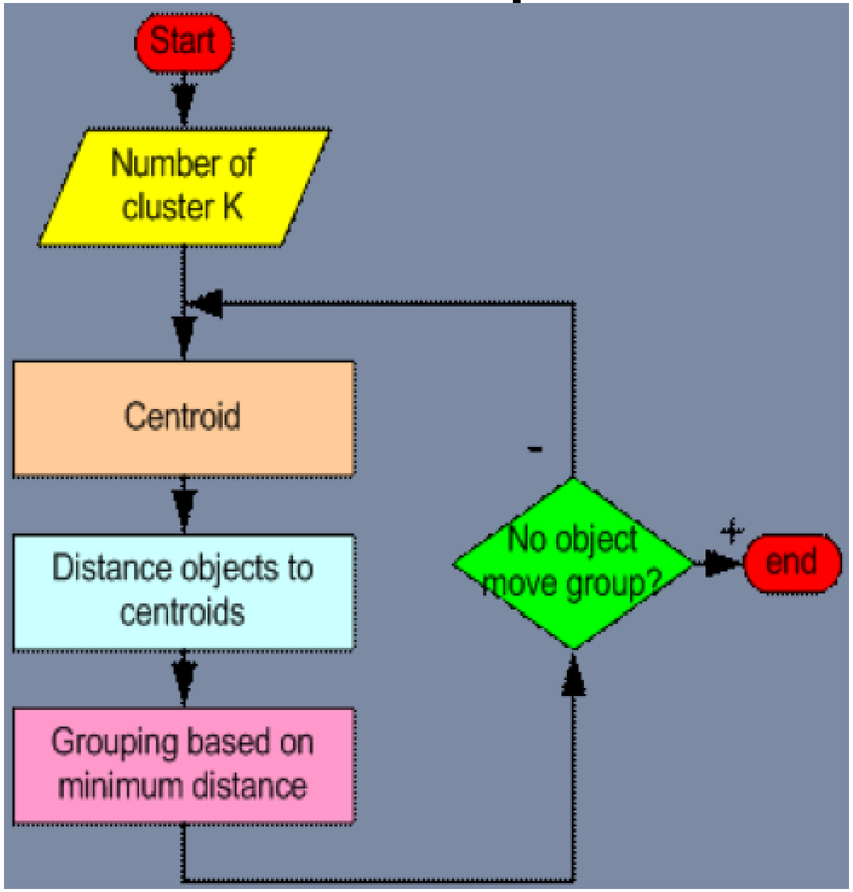
  - Each cluster is represented by the center of the cluster and the algorithm converges to stable centers of clusters.

K-means Algorithm

- Given the cluster number *K*, the *K-means* algorithm is carried out in three steps:

Initialisation: set seed points

- Assign each object to the cluster with the nearest seed point
- Compute seed points as the centroids of the clusters of the current partition (the centroid is the centre, i.e., *mean point*, of the cluster)
- Go back to Step 1), stop when no more new assignment

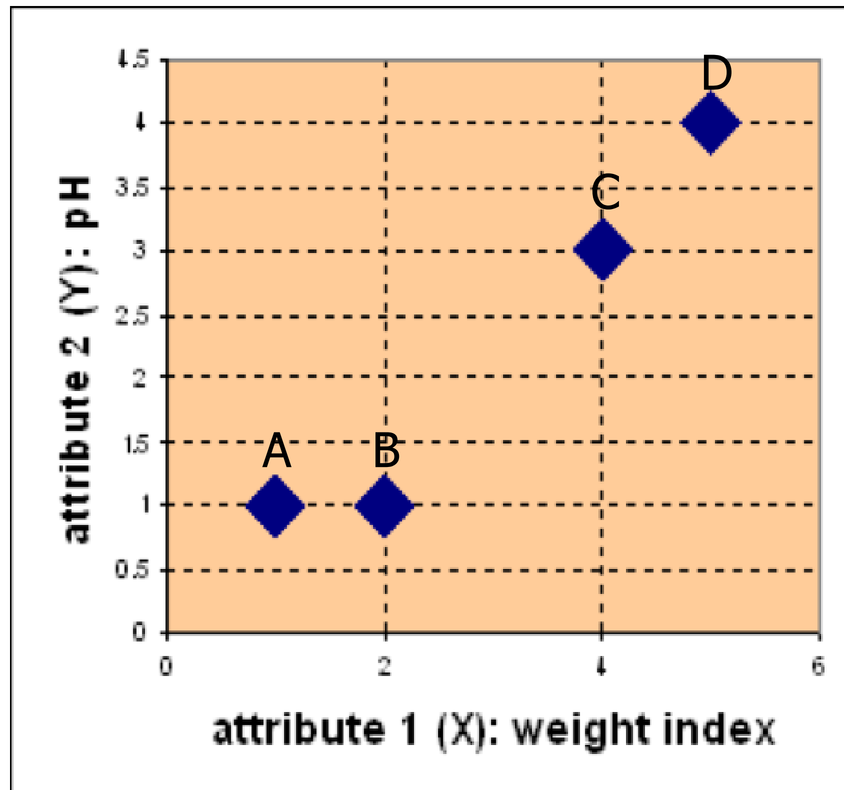


Example

Problem

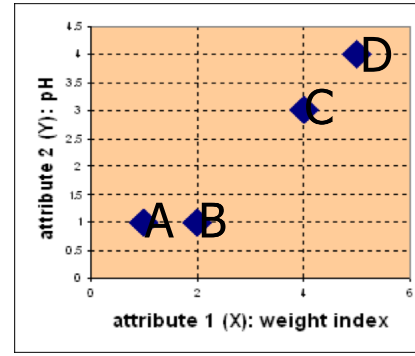
Suppose we have 4 types of medicines and each has two attributes (pH and weight index). Our goal is to group these objects into *K=2* group of medicine.

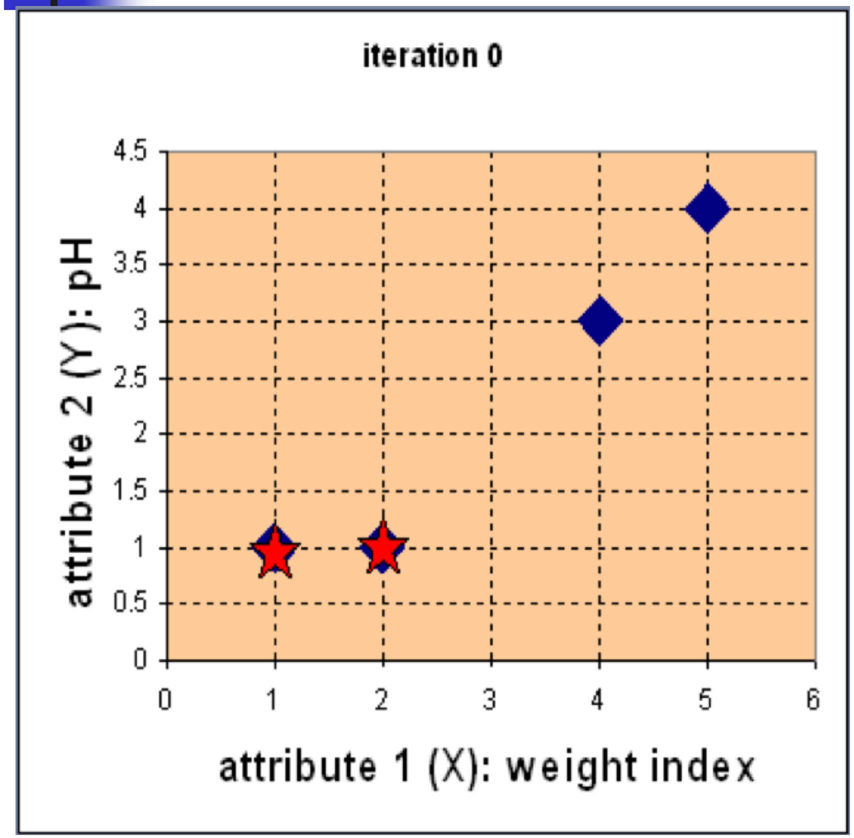
Medicine	**Weight**	**pH-Index**
A	1	1
B	2	1
C	4	3
D	5	4



Example

- Step 1: Use initial seed points for partitioning + create distance matrix





$$c_1 = A, c_2 = B$$

$$\mathbf{D}^0 = \begin{bmatrix} 0 & 1 & 3.61 & \boxed{5} \\ 1 & 0 & 2.83 & \boxed{4.24} \end{bmatrix} \quad \begin{array}{ll} \mathbf{c}_1 = (1,1) & group-1 \\ \mathbf{c}_2 = (2,1) & group-2 \end{array}$$

$$\begin{array}{cc} \begin{array}{cccc} A & B & C & D \end{array} & \\ \begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} & \begin{array}{c} X \\ Y \end{array} \end{array}$$

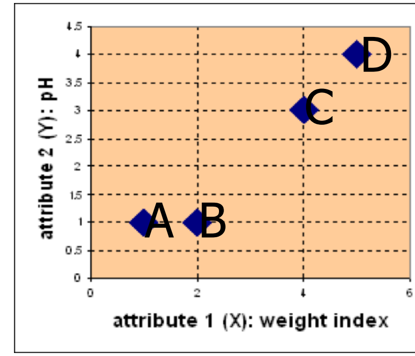
Euclidean distance

$$d(D, c_1) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

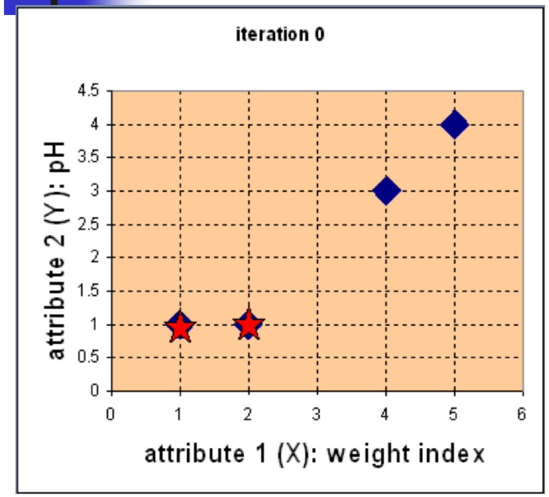
$$d(D, c_2) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

Assign each object to the cluster with the nearest seed point

Example



- Step 2: Assign points to clusters



$$c_1 = A, c_2 = B$$

$$\mathbf{D}^0 = \begin{array}{c} \begin{bmatrix} 0 & 1 & 3.61 & \boxed{5} \\ 1 & 0 & 2.83 & \boxed{4.24} \end{bmatrix} \\ \begin{array}{cccc} A & B & C & D \end{array} \end{array} \quad \begin{array}{ll} \mathbf{c}_1 = (1,1) & group-1 \\ \mathbf{c}_2 = (2,1) & group-2 \end{array}$$

$$\begin{array}{cc} \begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} & \begin{array}{c} X \\ Y \end{array} \end{array}$$

$$d(D, c_1) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

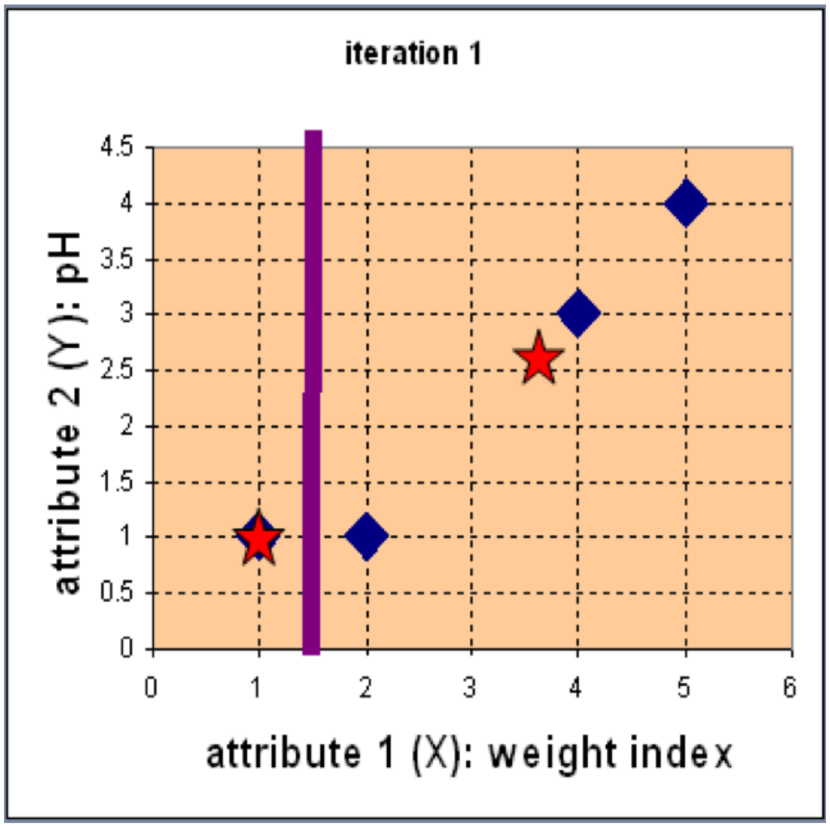
$$d(D, c_2) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

Distance Matrix : For every point select the cluster which has the shorter distance

Centroids	A	B	C	D
C1 = A	0	1	3.61	5
C2 = B	1	0	2.83	4.24

Example

- Step 3: Compute new centroids of the current partition



Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

$$c_1 = (1,\ 1)$$

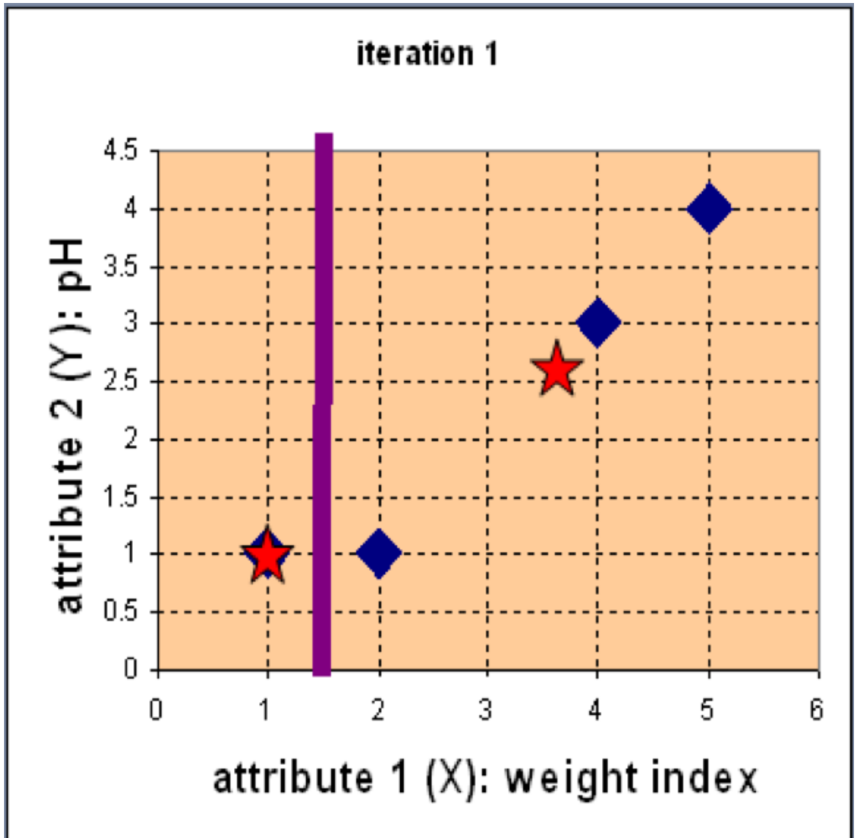
$$c_2 = \left(\frac{2+4+5}{3},\ \frac{1+3+4}{3} \right)$$

$$= (11/3,\ 8/11)$$

$$= (3.67,\ 2.67)$$

Example

- Step 4: Repeat the first two steps



Compute the distance of all objects to the new centroids

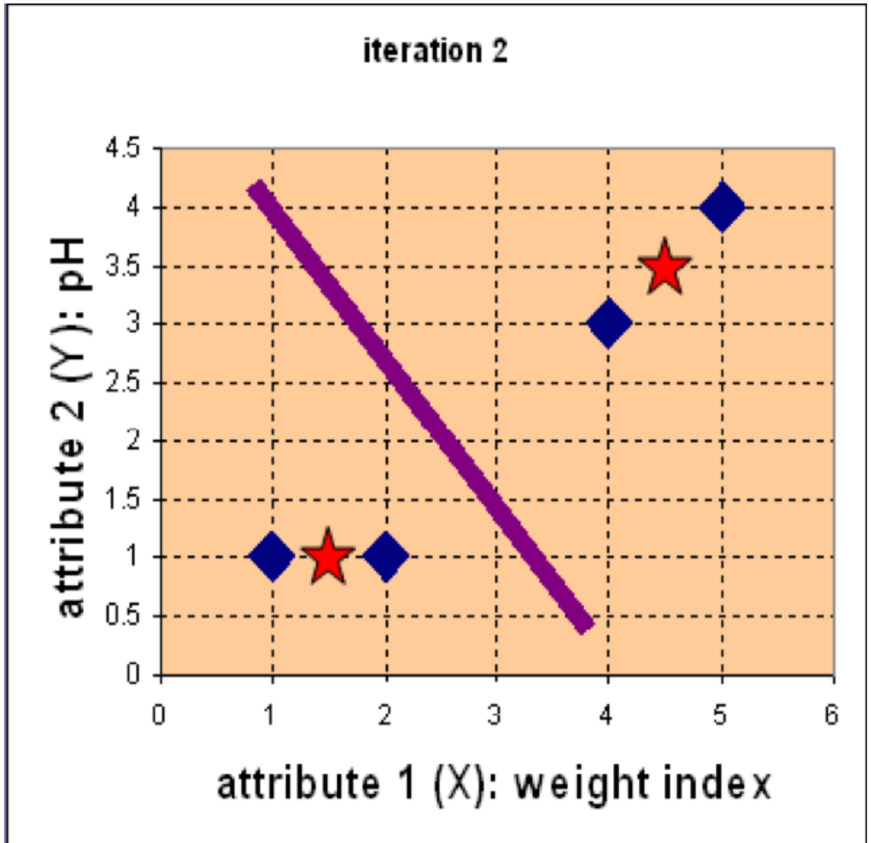
$$\mathbf{D}^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \quad \begin{matrix} \mathbf{c}_1 = (1,1) & group-1 \\ \mathbf{c}_2 = (\frac{11}{3}, \frac{8}{3}) & group-2 \end{matrix}$$

$$\begin{array}{c} \begin{matrix} A & B & C & D \end{matrix} \\ \begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \end{array} \begin{matrix} X \\ Y \end{matrix}$$

Assign the membership to objects

Example

- Step 5: Repeat the first two steps



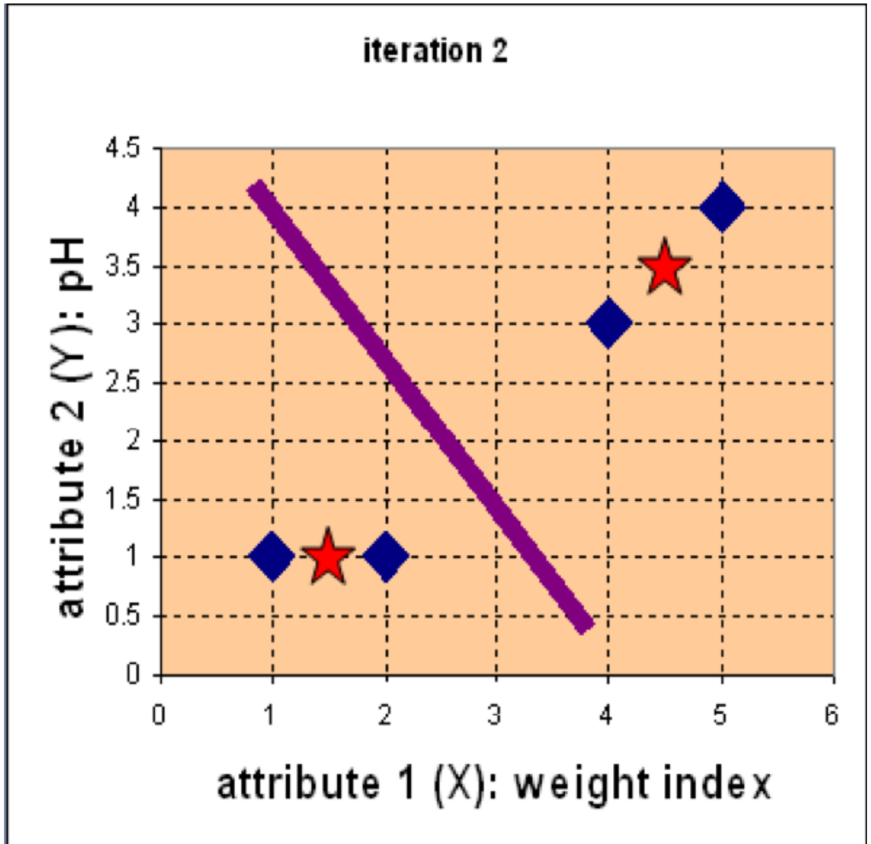
Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2} \right) = (1\frac{1}{2}, 1)$$

$$c_2 = \left(\frac{4+5}{2}, \frac{3+4}{2} \right) = (4\frac{1}{2}, 3\frac{1}{2})$$

Example

- Step 6: Repeat the first two steps



Compute the distance of all objects to the new centroids

$$\mathbf{D}^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \quad \begin{array}{ll} \mathbf{c}_1 = (1\tfrac{1}{2}, 1) & group-1 \\ \mathbf{c}_2 = (4\tfrac{1}{2}, 3\tfrac{1}{2}) & group-2 \end{array}$$

$$\begin{array}{cc} \begin{array}{cccc} A & B & C & D \end{array} & \\ \begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} & \begin{array}{c} X \\ Y \end{array} \end{array}$$

Stop due to no new assignment

K-means Clustering in R

Example 1

Read Data

Medici ne	Weight	pH-Index
A	1	1
B	2	1
C	4	3
D	5	4

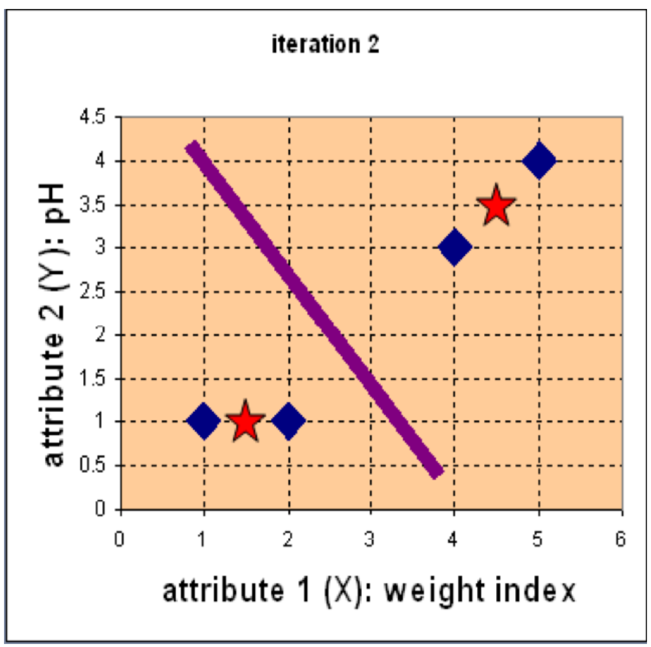
```
> #################################################
> medicine <- c("A","B","C","D")
> weight <- c(1,2,4,5)
> ph <- c(1,1,3,4)
>
> (dataClustering = data.frame(medicine, weight, ph))
  medicine weight ph
1        A      1  1
2        B      2  1
3        C      4  3
4        D      5  4
>
> (dataClustering2 = data.frame(weight, ph))
  weight ph
1      1  1
2      2  1
3      4  3
4      5  4
```

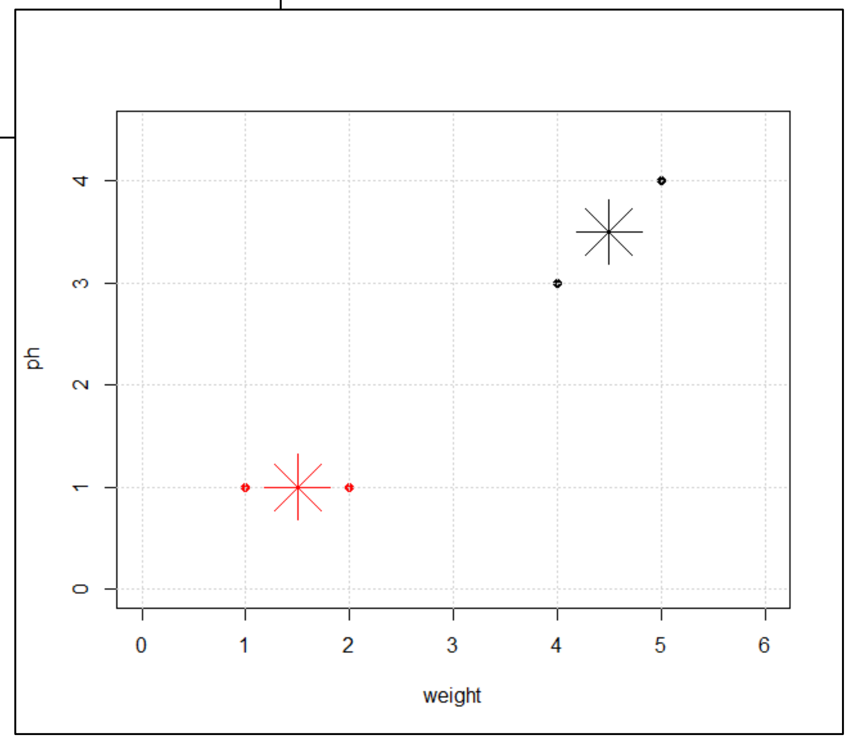
Build Clusters

```
> ####################################################
> numClusters = 2
>
> (kmeans.result <- kmeans(dataClustering2,numClusters))
K-means clustering with 2 clusters of sizes 2, 2
Cluster means:
  weight  ph
1    4.5 3.5
2    1.5 1.0
Clustering vector:
[1] 2 2 1 1
Within cluster sum of squares by cluster:
[1] 1.0 0.5
 (between_SS / total_SS =  91.0 %)
Available components:
[1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
>
> table(dataClustering$medicine, kmeans.result$cluster)

    1 2
  A 0 1
  B 0 1
  C 1 0
  D 1 0
```

Plot the Clusters

```
> ########################################################
> plot(dataClustering2[c("weight", "ph")], pch=21,
+      col = kmeans.result$cluster,
+      bg=kmeans.result$cluster,
+      xlim=c(0,6),ylim=c(0,4.5))
> grid()
>
> points(kmeans.result$centers[,c("weight", "ph")],
+         col = 1:numClusters, pch = 8, cex = 5)
>
> ########################################
>
```



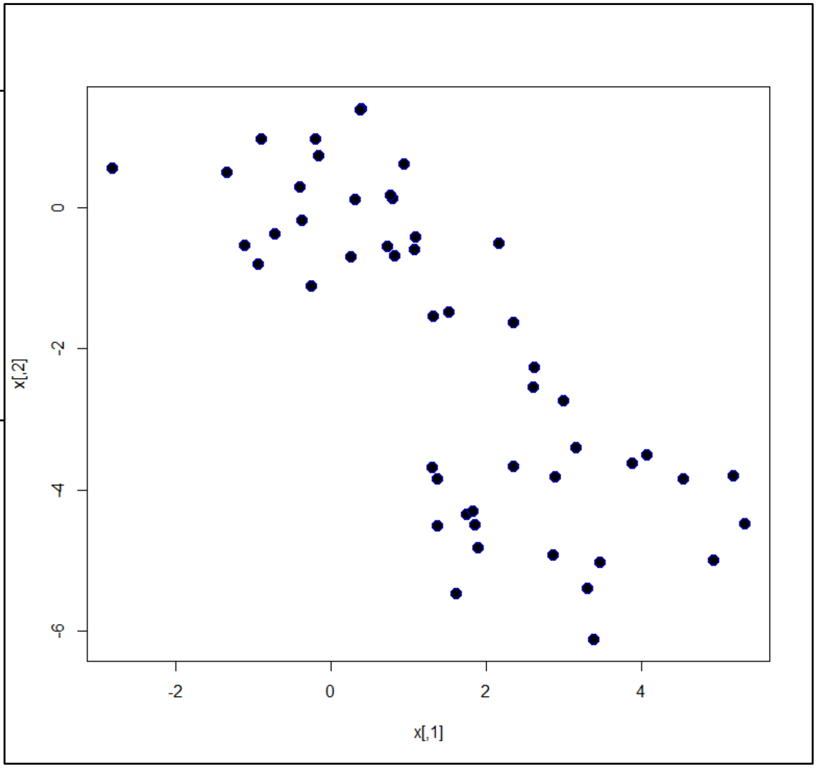


K-means Clustering in R

Example 2

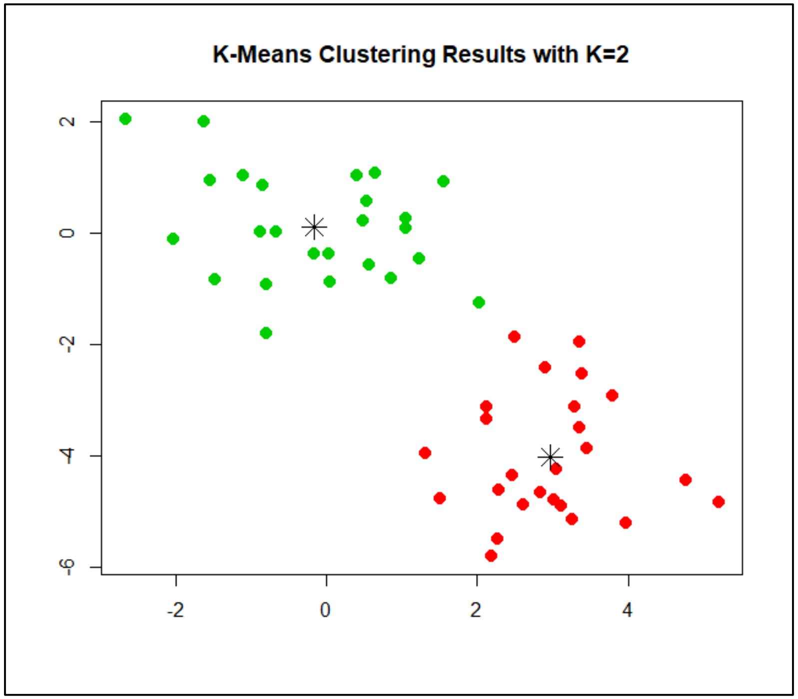
Dataset

```
> ###############################################
> # 1. Generate Dataset
> #
> x=matrix(rnorm(50*2), ncol=2)
> #x
> x[1:25,1]=x[1:25,1]+3
> x[1:25,2]=x[1:25,2]-4
> plot(x,pch=21,col='blue',bg='black',cex=1.5)
```



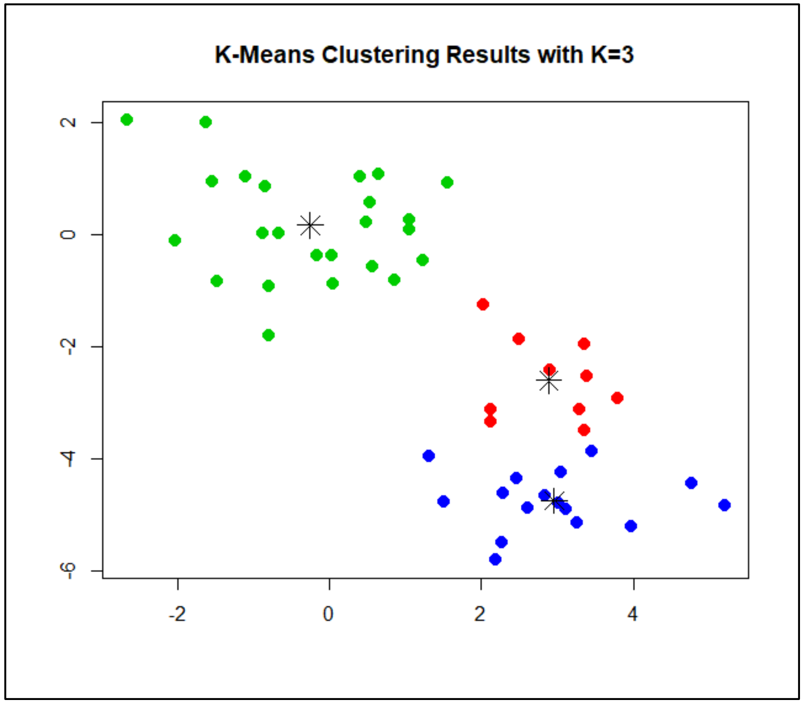
Clustering: k=2

```
>
############################################################
> # 2. k-means clustering
> # k = 2
> set.seed(2)
> km.out=kmeans(x,2,nstart=20)
> km.out$centers
        [,1]      [,2]
1  2.9646980 -4.033903
2 -0.1661289  0.110885
> km.out$withinss
[1] 49.17541 57.40236
> plot(x, col=(km.out$cluster+1), main="K-Means
Clustering Results with K=2", xlab="", ylab="",
pch=20, cex=2)
>
>
points(km.out$centers[1,1],km.out$centers[1,2],
pch=8,col='black',bg='black',cex=2)
>
points(km.out$centers[2,1],km.out$centers[2,2],
pch=8,col='black',bg='black',cex=2)
>
```



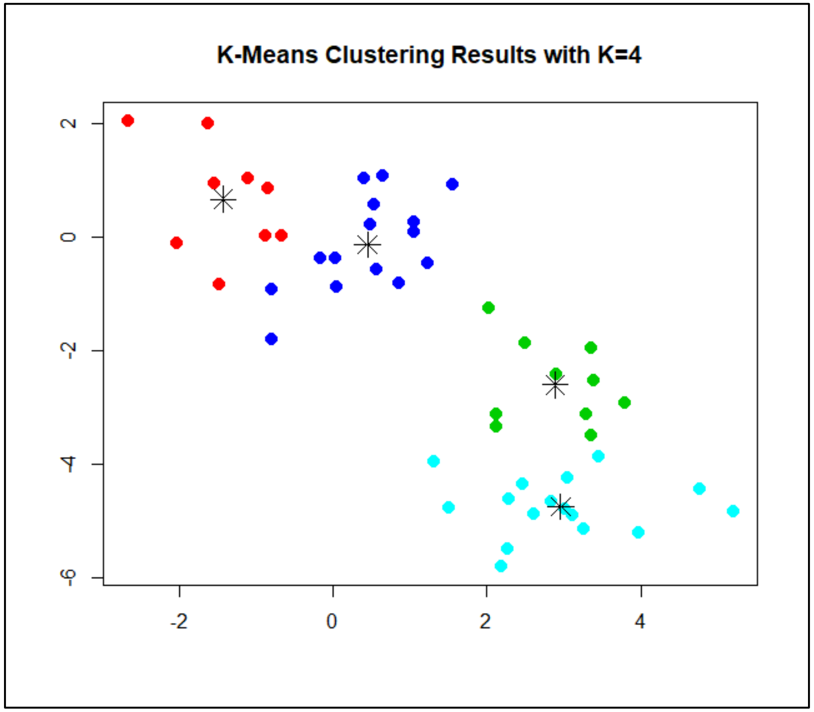
Clustering: k=3

```
>
###############################################
> # 4. k-means clustering
> # k = 3
> set.seed(3)
> km.out=kmeans(x,3,nstart=20)
> km.out$centers
        [,1]       [,2]
1  2.8855294 -2.6045028
2 -0.2572725  0.1675683
3  2.9552172 -4.7532545
> km.out$withinss
[1]  8.59029 50.49027 20.45248
> plot(x, col=(km.out$cluster+1), main="K-Means
Clustering Results with K=3", xlab="", ylab="",
pch=20, cex=2)
>
>
points(km.out$centers[1,1],km.out$centers[1,2],
pch=8,col='black',bg='black',cex=2)
>
points(km.out$centers[2,1],km.out$centers[2,2],
pch=8,col='black',bg='black',cex=2)
>
points(km.out$centers[3,1],km.out$centers[3,2],
pch=8,col='black',bg='black',cex=2)
>
```



Clustering: k=4

```
> ############################################
> # 4.1 k-means clustering
> # k = 4
> set.seed(4)
> km.out=kmeans(x,4,nstart=20)
> km.out$centers
        [,1]       [,2]
1 -1.4268114  0.6696754
2  2.8855294 -2.6045028
3  0.4444509 -0.1336960
4  2.9552172 -4.7532545
> km.out$withinss
[1] 10.93799  8.59029 16.22524 20.45248
> plot(x, col=(km.out$cluster+1), main="K-Means
Clustering Results with K=4", xlab="", ylab="",
pch=20, cex=2)
>
>
points(km.out$centers[1,1],km.out$centers[1,2],
pch=8,col='black',bg='black',cex=2)
>
points(km.out$centers[2,1],km.out$centers[2,2],
pch=8,col='black',bg='black',cex=2)
>
points(km.out$centers[3,1],km.out$centers[3,2],
pch=8,col='black',bg='black',cex=2)
>
points(km.out$centers[4,1],km.out$centers[4,2],
pch=8,col='black',bg='black',cex=2)
>
```



Parameter: nstart

- Clustering algorithm will give slightly different results if we start with different initial values
- The ‘kmeans’ algorithm implemented in R has a parameter ‘nstart’ which indicates multiple random initial assignments
- Suppose ‘nstart’ = n
  - Algorithm builds ‘n’ clusters and only the best cluster is reported
  - Best cluster is the one which has minimum within-cluster-variation

Parameter: nstart

- It is recommended that ‘nstart’ value should be high
- The ‘withinss’ output
  - = Sum of square within a cluster
  - = within-cluster-variation

```
> set.seed(5)
> km.out=kmeans(x,3,nstart=1)
> km.out$tot.withinss
[1] 68.72622
> #################################################
> #
> km.out=kmeans(x,3,nstart=20)
> km.out$tot.withinss
[1] 66.60633
>
```

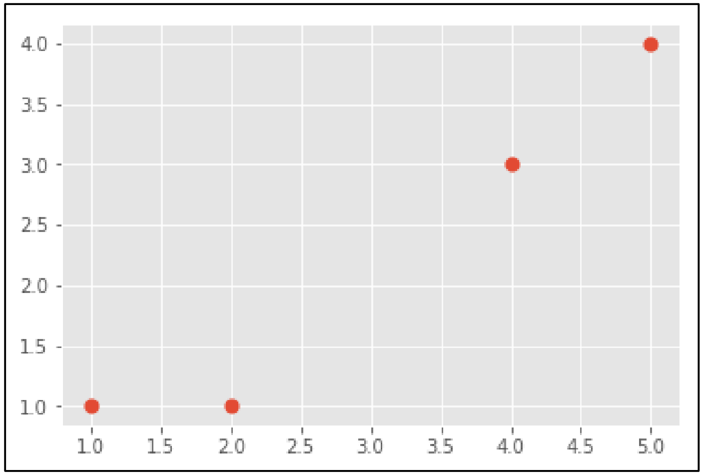
K-Means Clustering in Python/Scikit-Learn

Example-3

Read Data

Medicine	Weight	pH-Index
A	1	1
B	2	1
C	4	3
D	5	4

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
style.use("ggplot")
from sklearn.cluster import KMeans

X = np.array([ [1,1],
        [2,1],
        [4, 3],
        [5,4]])
print(X)
plt.scatter(X[:,0], X[:,1], s=10, linewidth=5)
[[1 1]
 [2 1]
 [4 3]
 [5 4]]
```

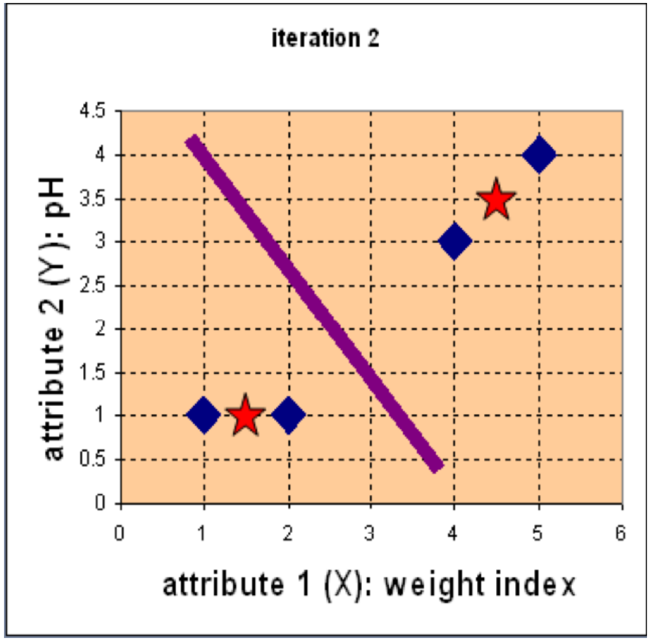


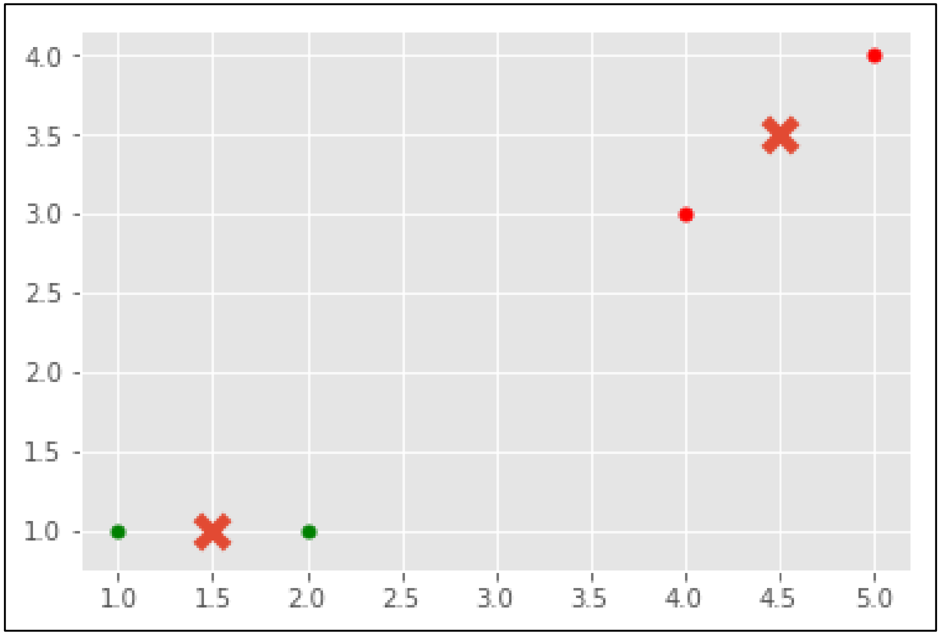
Build Clusters

```
clf = KMeans(n_clusters=2)

clf.fit(X)
Out[8]:
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)

centroids = clf.cluster_centers_

labels = clf.labels_

print(centroids)
[[ 1.5  1. ]
 [ 4.5  3.5]]

print(labels)
[0 0 1 1]
```

Plot the Clusters

```
colors = ["g.","r.","c.","b.","k.","g."]

for i in range(len(X)):
    plt.plot(X[i][0], X[i][1], colors[labels[i]], markersize = 10)


plt.scatter(centroids[:,0], centroids[:,1], marker='x', s=150, linewidth=5)
Out[16]: <matplotlib.collections.PathCollection at 0x1ec15f1a7f0>
```





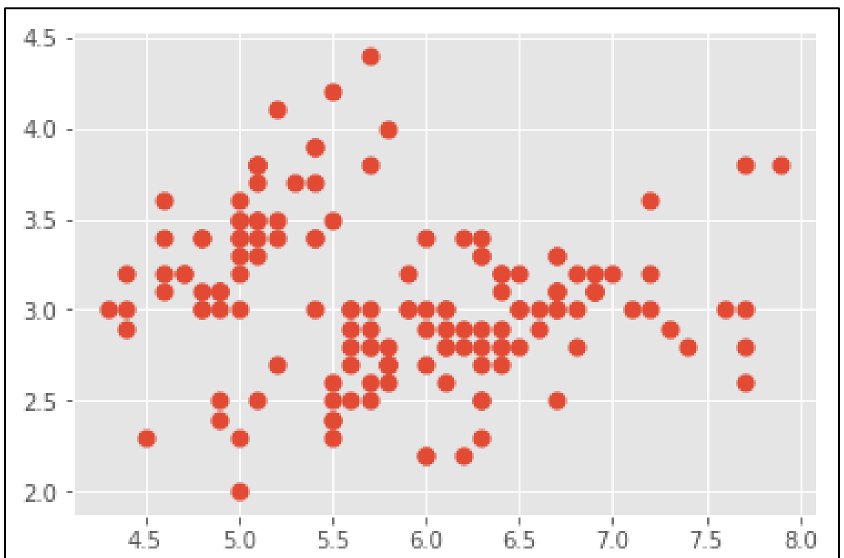
K-Means Clustering in Python/Scikit-Learn

Example-4

Iris Dataset

Read Data

```
from sklearn import datasets
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from matplotlib import style
style.use("ggplot")

########################################
# Load Data
#
iris = datasets.load_iris()
X_iris = iris.data
# Retrieve only the first 2 variables
# Petal Length + Petal Width from the data
#
X2_iris = iris.data[:,0:2]

plt.scatter(X2_iris[:,0], X2_iris[:,1], s=10, linewidth=5)
```

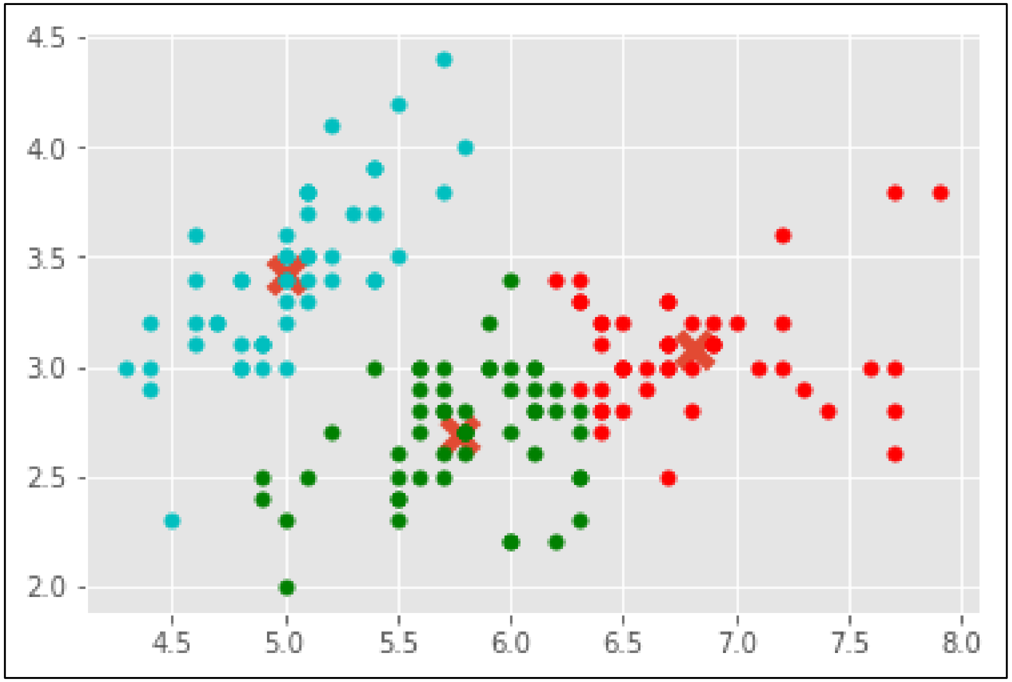


Build Clusters

```
clf = KMeans(n_clusters=3)

clf.fit(X2_iris)
Out[22]:
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)

centroids = clf.cluster_centers_

labels = clf.labels_

print(centroids)
[[ 5.77358491  2.69245283]
 [ 6.81276596  3.07446809]
 [ 5.006       3.418     ]]

print(labels)
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 1 1 1
 1 1 0 0 1 1 1 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1
 1 0]
```

Plot the Clusters

```
colors = ["g.","r.","c.","b.","k.","g."]

for i in range(len(X2_iris)):
    plt.plot(X2_iris[i][0], X2_iris[i][1], colors[labels[i]], markersize = 10)

plt.scatter(centroids[:,0], centroids[:,1], marker='x', s=150, linewidth=5)
```



Summary

- What is Clustering
- Computing Within-Cluster-Variation
- K-means Algorithm
- K-means Clustering in R
- K-means Clustering in
  - Python/Scikit-Learn