Machine Learning HW-4

Sagar Kalauni

2023-12-10

1) Consider a neural network with two hidden layers: p = 4 input units, 2 units in the first hidden layer, 3 units in the second hidden layer, and a single output.

(a) Draw a picture of the network.

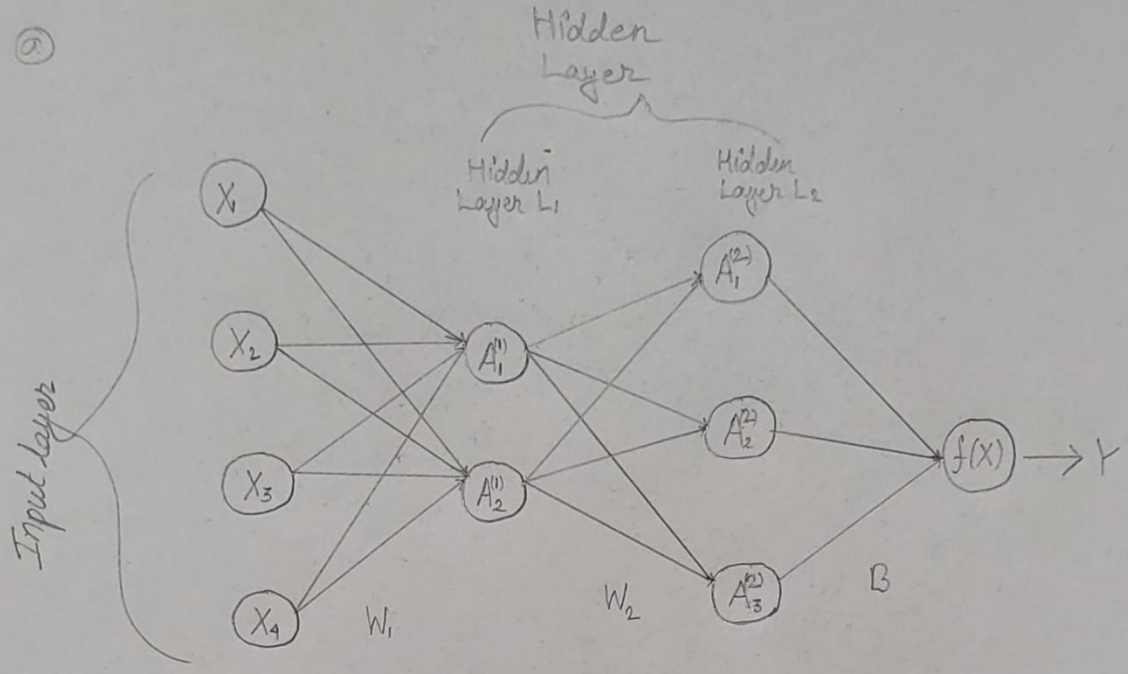
(b) Write out an expression for f(X), assuming ReLU activation functions. Be as explicit as you can!

(c) How many parameters are there? ANSWER:- Please look for the attached figure below:

HW=#4

Solution for Q.NO.-1

(a)



(b) The expression for $f(X)$ is given as

$$f(X) = \beta_0 + \sum \beta_k A_k^{(L)}$$

If our activation function is ReLU (Rectified linear unit). Then

$$f(X) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

(c) The above figure network has $2+3+1 = 6$ neurons, $[4*2]+[2*3]+[3*1]$ $= 8+6+3 = 17$ weights and $2+3+1 = 6$ biases, for a total of 23 learnable parameter

$\therefore$ # of parameters = 23. □

2) Consider the Default data. Split the data into 70% training and 30% test.

```r
set.seed(100)
#install.packages("ISLR2")
library(ISLR2)

## Warning: package 'ISLR2' was built under R version 4.3.2

library(nnet)

## Warning: package 'nnet' was built under R version 4.3.2

standardize=function(x) {(x-min(x))/(max(x)-min(x))}
Default$income=standardize(Default$income)
Default$balance=standardize(Default$balance)

index=sample(1:nrow(Default), 0.7*nrow(Default))
train=Default[index,]
test=Default[-index,]
```

(a) Fit a neural network using a single hidden layer with 10 units.

```r
set.seed(100)
#install.packages("nnet")
#library(nnet)

NN.fit=nnet(default~., data=train, size=10 )  # For linout:-Default logistic
output units

## # weights:  51
## initial  value 6634.591690
## iter  10 value 772.690204
## iter  20 value 564.802404
## iter  30 value 559.909439
## iter  40 value 558.935767
## iter  50 value 558.251088
## iter  60 value 557.674311
## iter  70 value 557.266359
## iter  80 value 556.877075
## iter  90 value 556.663528
## iter 100 value 556.331984
## final  value 556.331984
## stopped after 100 iterations

set.seed(100)
test_prob=predict(NN.fit, test)

test_pred=rep("No", nrow(test))
test_pred[test_prob>0.5]="Yes"

table(test_pred, test$default)
```

```
##
## test_pred   No  Yes
##       No  2900   59
##       Yes   15   26
```

Observation: The test Accuracy of the Neural network with single hidden layer having 10 units in the test data set is:$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{2900+26}{2900+59+15+26} = 0.9753333$

```
set.seed(100)
# Fit a linear logistic regression model
logistic_model=glm(
  formula = default ~ income + balance + student,
  data = train,
  family = binomial
)
glm_test_prob=predict(logistic_model, newdata = test)

glm_test_pred=rep("No", nrow(test))
glm_test_pred[glm_test_prob>0.5]="Yes"

table(glm_test_pred, test$default)
```

```
##
## glm_test_pred   No  Yes
##           No  2909   70
##           Yes    6   15
```

Observation: The test Accuracy of the Logistic regression model in the test data set is:$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{2909+15}{2909+70+6+15} = 0.9746667$

(b) Compare the classification performance of your model with that of linear logistic regression. ANSWER: Both are doing comparatively same but neural network with single hidden layer has slight more accuracy then logistic regression model in our test data.

3) In this problem, you will perform K-means clustering manually, with $K = 2 $, on a small example. The observations are as follows.

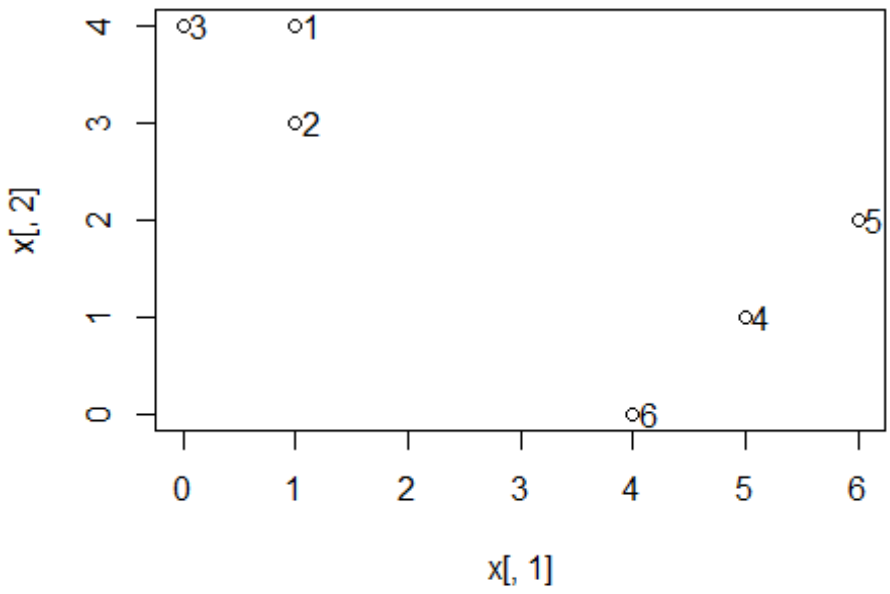
```
mydata <- data.frame(
  Obs = c(1, 2, 3, 4, 5, 6),
  X1 = c(1, 1, 0, 5, 6, 4),
  X2 = c(4, 3, 4, 1, 2, 0)
)
print(mydata)
```

```
##   Obs X1 X2
## 1   1  1  4
## 2   2  1  3
## 3   3  0  4
## 4   4  5  1
```

```
## 5   5  6  2
## 6   6  4  0
```

(a) Sketch the observations.

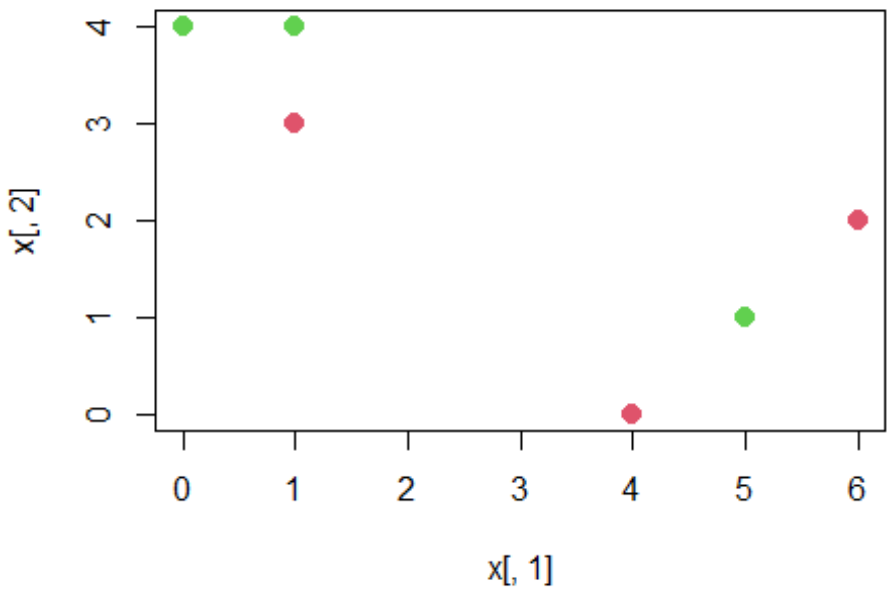
```
x <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
plot(x[,1], x[,2])
text(mydata$X1+0.15, mydata$X2, 1:6)
```




```
# Here I am showing each data point with its observation number in the plot, +0.15 is added not to concide the label and data in one, so at the same y-distance and little more x-distance, I am showing my label of the data point.
```

(b) Randomly assign a cluster label to each observation.

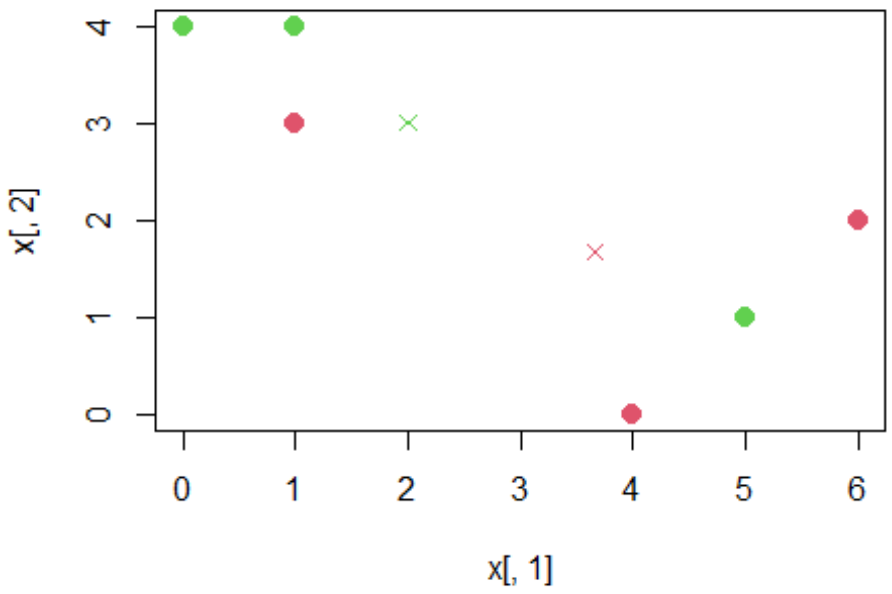
```
set.seed(100)
labels=sample(2, nrow(x), replace = T)
labels

## [1] 2 1 2 2 1 1

plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2)
```



(c) Compute the centroid for each cluster. ANSWER:- We compute the centroid of red cluster as $ {x}*{11}* = *(1 + 4 + 6) = $ and $ {x}*{12} = (3 + 0 + 2) = $

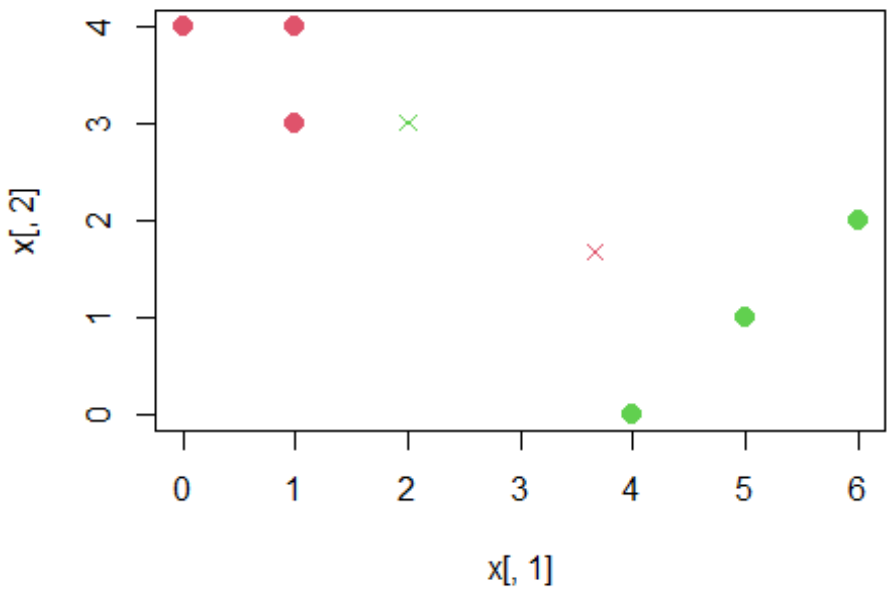
and the centroid of the green cluster as: $ {x}*{21}* = *(0 + 1 + 5) = 2 $ and $ {x}*{22} = (4 + 4 + 1) = 3 $

```
set.seed(100)
centroid1 <- c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
centroid2 <- c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
centroid1

## [1] 3.666667 1.666667

centroid2

## [1] 2 3

plot(x[,1], x[,2], col=(labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



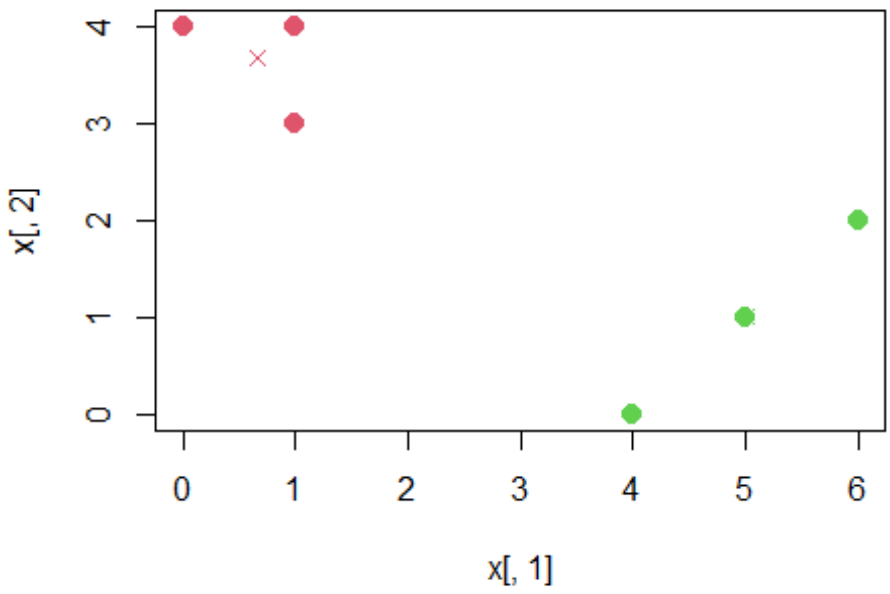
(d) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

```
labels <- c(1, 1, 1, 2, 2, 2)
plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



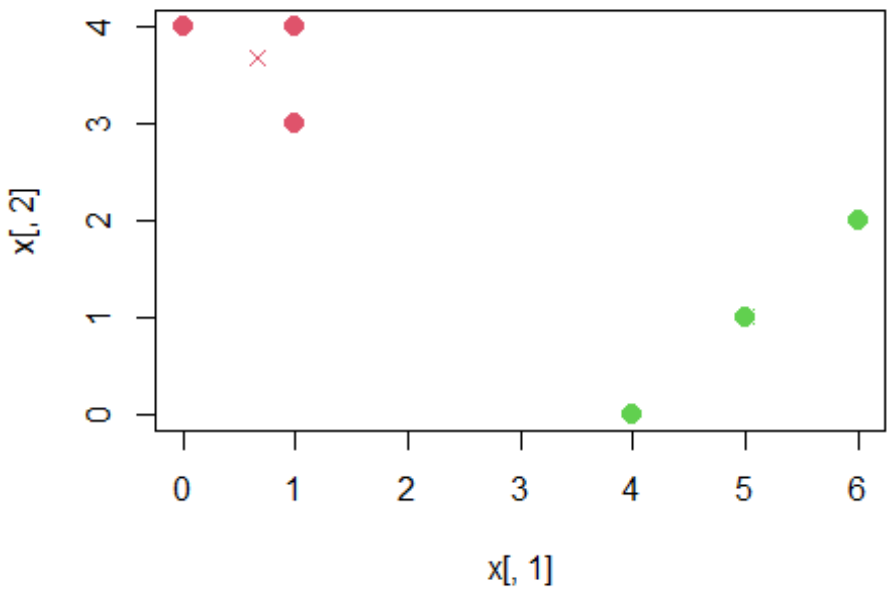
(e) Repeat (c) and (d) until the answers obtained stop changing. Answer:-We compute the centroid of red cluster as $ {x}*{11} = (0 + 1 + 1) = $ and $ {x}*{12} = (3 + 4 + 4) = $ and the centroid of the green cluster as: $ {x}*{21} = (4 + 5 + 6) = 5 $ and $ {x}*{22} = (0 + 1 + 2) = 1 $

```
set.seed(100)
centroid1 <- c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
centroid2 <- c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
centroid1

## [1] 0.6666667 3.6666667

centroid2

## [1] 5 1

plot(x[,1], x[,2], col=(labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



Re- asagining

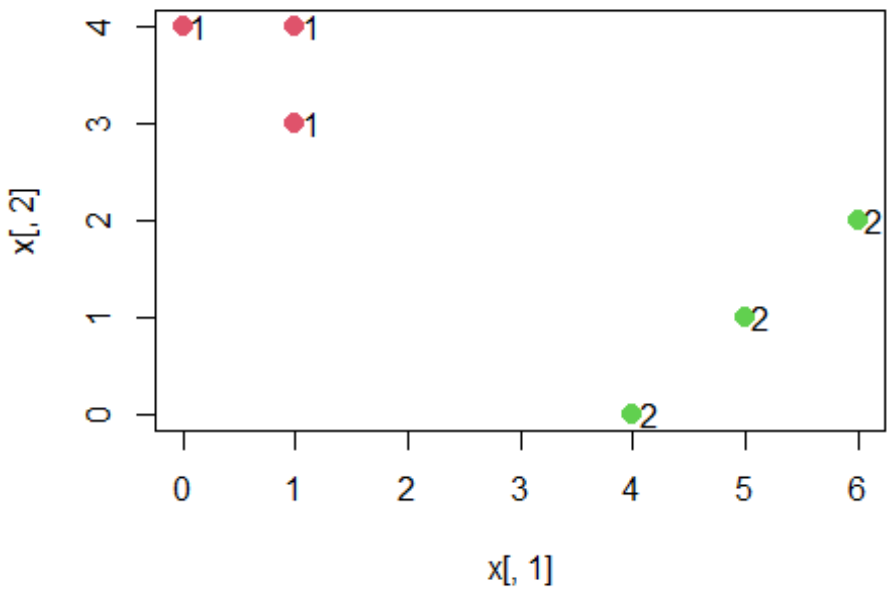
```
labels <- c(1, 1, 1, 2, 2, 2)
plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



If we assign each observation to the centroid to which it is closest, nothing changes, so the algorithm is terminated at this step.

(f) In your plot from (a), label the observations according to the final cluster labels obtained.

```
plot(x[, 1], x[, 2], col=(labels + 1), pch = 20, cex = 2)
text(x[, 1]+0.15, x[, 2], labels)
```



4. In this problem, you consider the gene expression data (Khan, in ISLR), and then perform clustering on the data.

```
# Application to Gene Expression Data
set.seed(500)
library(ISLR)

##
## Attaching package: 'ISLR'

## The following object is masked _by_ '.GlobalEnv':
##
##     Default

## The following objects are masked from 'package:ISLR2':
##
##     Auto, Credit

names(Khan)

## [1] "xtrain" "xtest"  "ytrain" "ytest"

dim(Khan$xtrain)

## [1]   63 2308

dim(Khan$xtest)

## [1]   20 2308
```

```r
length(Khan$ytrain)
```

```
## [1] 63
```

```r
length(Khan$ytest)
```

```
## [1] 20
```

```r
table(Khan$ytrain)
```

```
##
##  1  2  3  4
##  8 23 12 20
```

```r
table(Khan$ytest)
```

```
##
## 1 2 3 4
## 3 6 6 5
```

```r
dat=data.frame(x=Khan$xtrain, y=as.factor(Khan$ytrain))
```

(a) Perform K-means clustering of the "xtrain" with K = 4. How well do the clusters that you obtained in K-means clustering compare to the true class labels ("ytrain")?

```r
set.seed(500)
khan_clust=kmeans(Khan$xtrain,centers=4)
khan_clust$cluster
```

```
##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 V12 V13 V14 V15 V16 V17 V18
V19 V20
##   3   3   3   3   2   4   2   2   2   2   3   3   3   3   1   1   1   1
1   1
## V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V34 V35 V36 V37 V38
V39 V40
##   1   1   1   2   1   1   4   4   4   3   2   4   3   3   3   3   3   1
1   1
## V41 V42 V43 V44 V45 V46 V47 V48 V49 V50 V51 V52 V53 V54 V55 V56 V57 V58
V59 V60
##   1   1   1   3   3   3   4   4   4   4   4   4   4   4   4   3   3   3
3   4
## V61 V62 V63
##   4   4   4
```

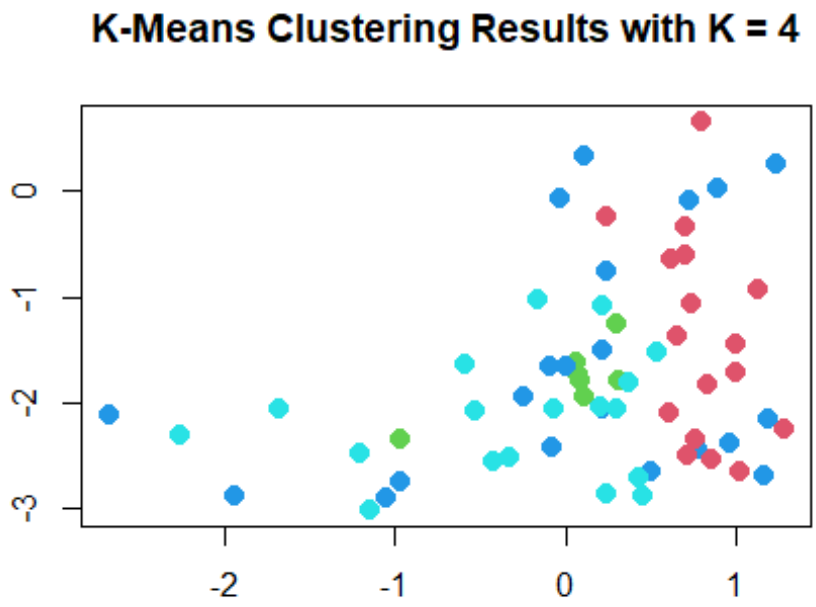
```r
set.seed(500)
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.3.2
```

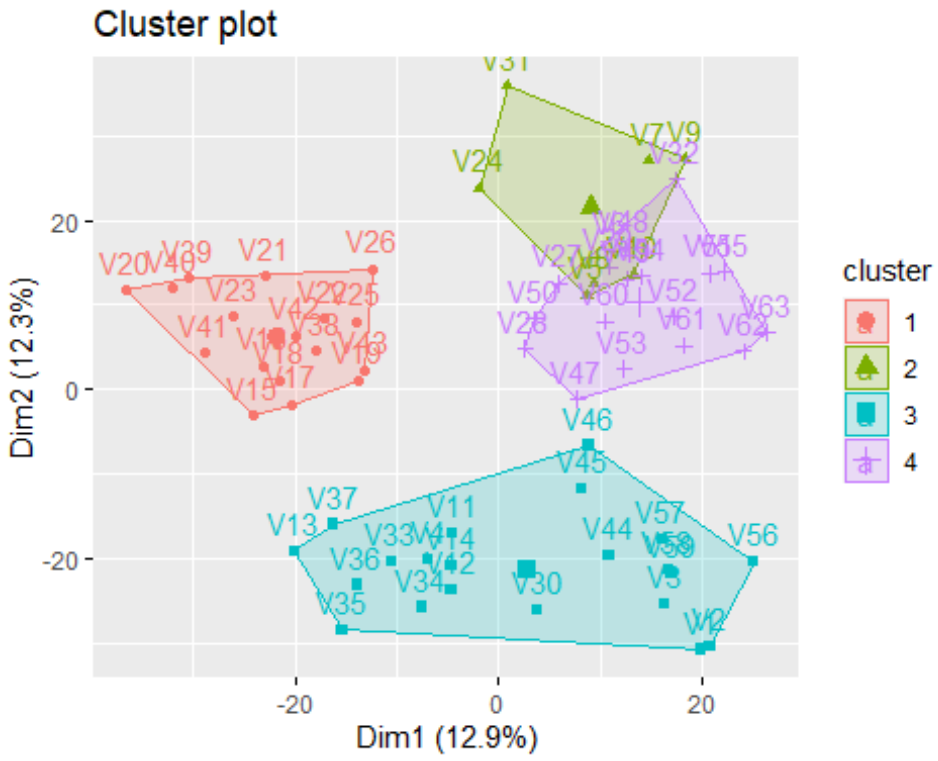
```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.1
```

```
## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa

plot(Khan$xtrain, col = (khan_clust$cluster + 1),
main = "K-Means Clustering Results with K = 4",
xlab = "", ylab = "", pch = 20, cex = 2)
```



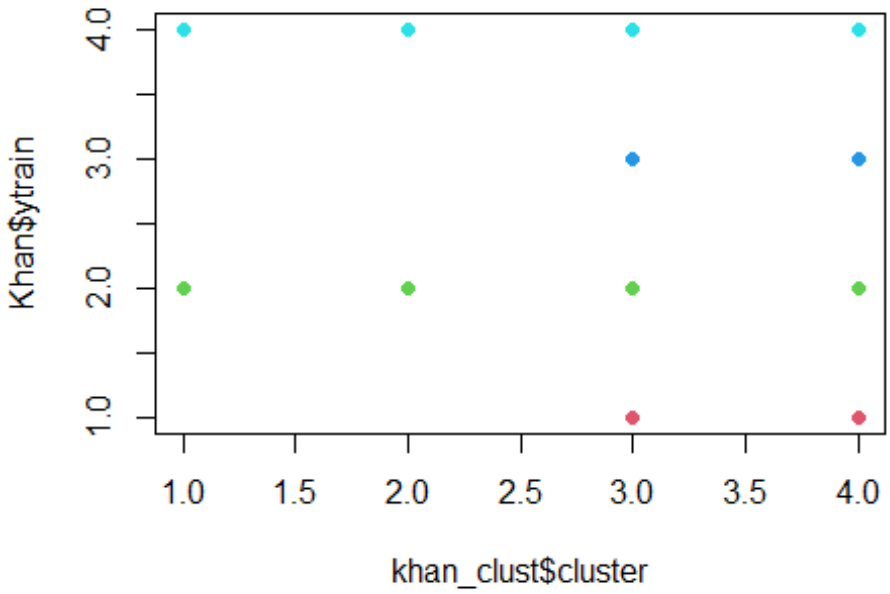

```
fviz_cluster(list(data=Khan$xtrain,cluster=khan_clust$cluster))
```




```
set.seed(500)
table(khan_clust$cluster,Khan$ytrain)

##
##     1 2 3 4
##   1 0 9 0 8
##   2 0 5 0 2
##   3 4 8 3 6
##   4 4 1 9 4
```

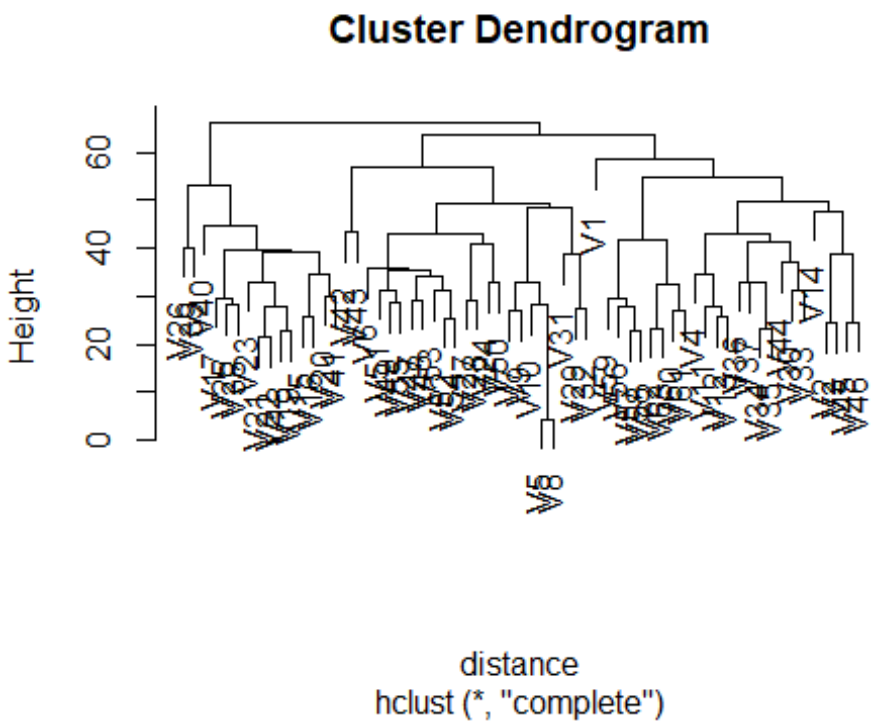
Observation: the clustering done by kmean clustering algorithm has accuracy of: (0+5+3+4)/63 = 0.1904762 i.e 12.69%. so we can say it performs really poor in clustering.

```
set.seed(500)
plot(khan_clust$cluster,Khan$ytrain, col=Khan$ytrain+1, pch=19)
```



(b) Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```
set.seed(500)
distance=dist(dat,method="euclidean")
cc=hclust(distance,method="complete")
plot(cc)
```



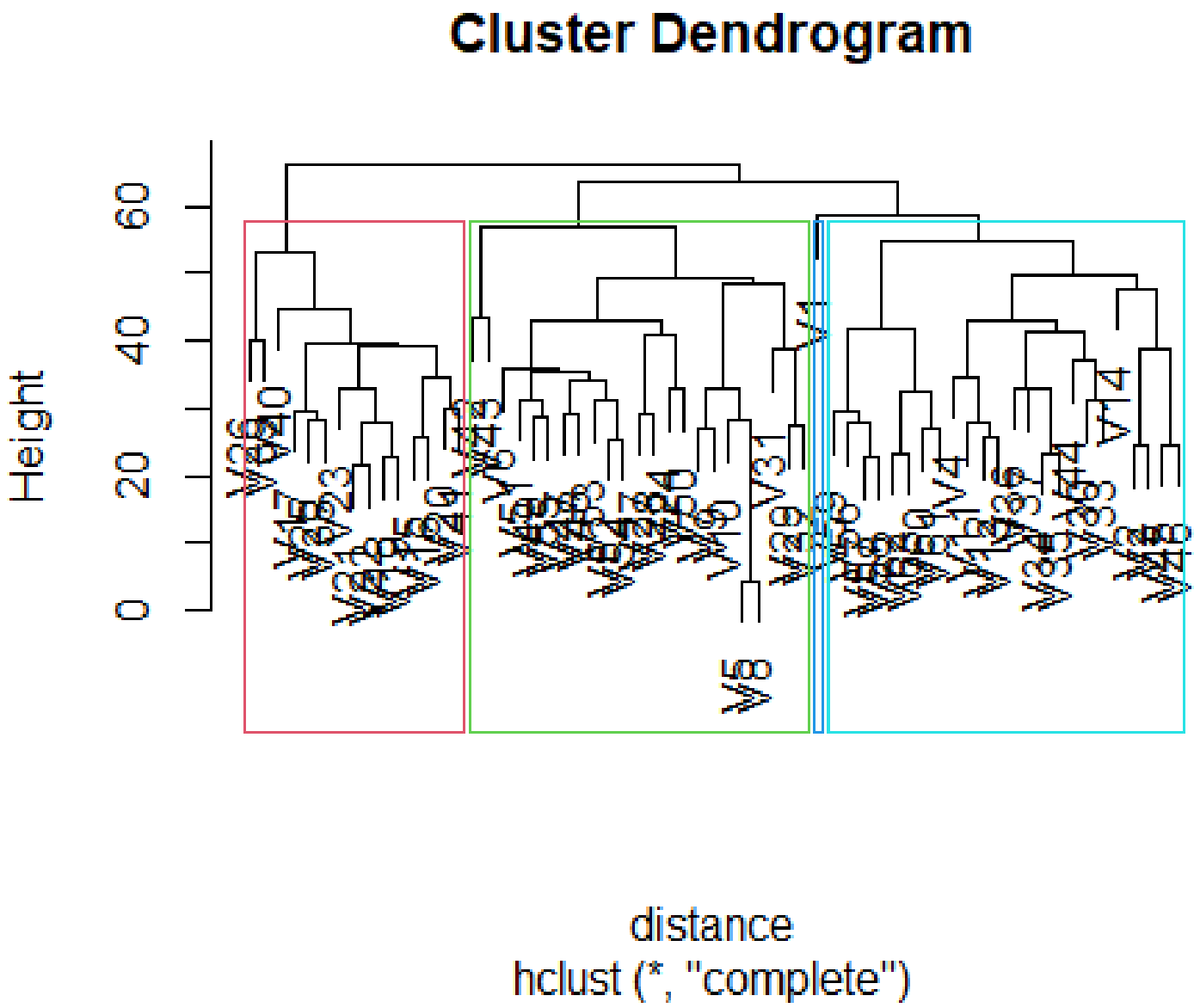
(c) Cut the dendrogram at a height that results in 4 distinct clusters.

```
set.seed(100)
cutree(cc, 4)
```

```
##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 V12 V13 V14 V15 V16 V17 V18
V19 V20
##   1   2   2   2   3   3   3   3   3   3   2   2   2   2   4   4   4   4
4   4
## V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V34 V35 V36 V37 V38
V39 V40
##   4   4   4   3   4   4   3   3   3   2   3   3   2   2   2   2   2   4
4   4
## V41 V42 V43 V44 V45 V46 V47 V48 V49 V50 V51 V52 V53 V54 V55 V56 V57 V58
V59 V60
##   4   3   3   2   2   2   3   3   3   3   3   3   3   3   3   2   2   2
2   2
## V61 V62 V63
##   2   2   2
```

If you want to vissually look the cut point and look at the cluster formed

```
plot(cc)
rect.hclust(cc, k = 4, border = 2:5)
```



I zoomed this picture for you, clearly you can see now you are remaining with only 4 cluster and one of the cluster has V1 as the only entity on it.