Ahmed Alhasan

Assignment 1

```
##        Predicted
## Actual  0  1
##      0  9  0
##      1  0 13
```

1p

- The model fit the data quite well, this is because of couple factors, the response “Class” is easily predictable from other features, the test data is small this doese not allow many many missclassifications also the logistic regression is a robust model that does not need many assumptions.

- Probablistic Model: Class ~ Binomial(θ = 0.5)

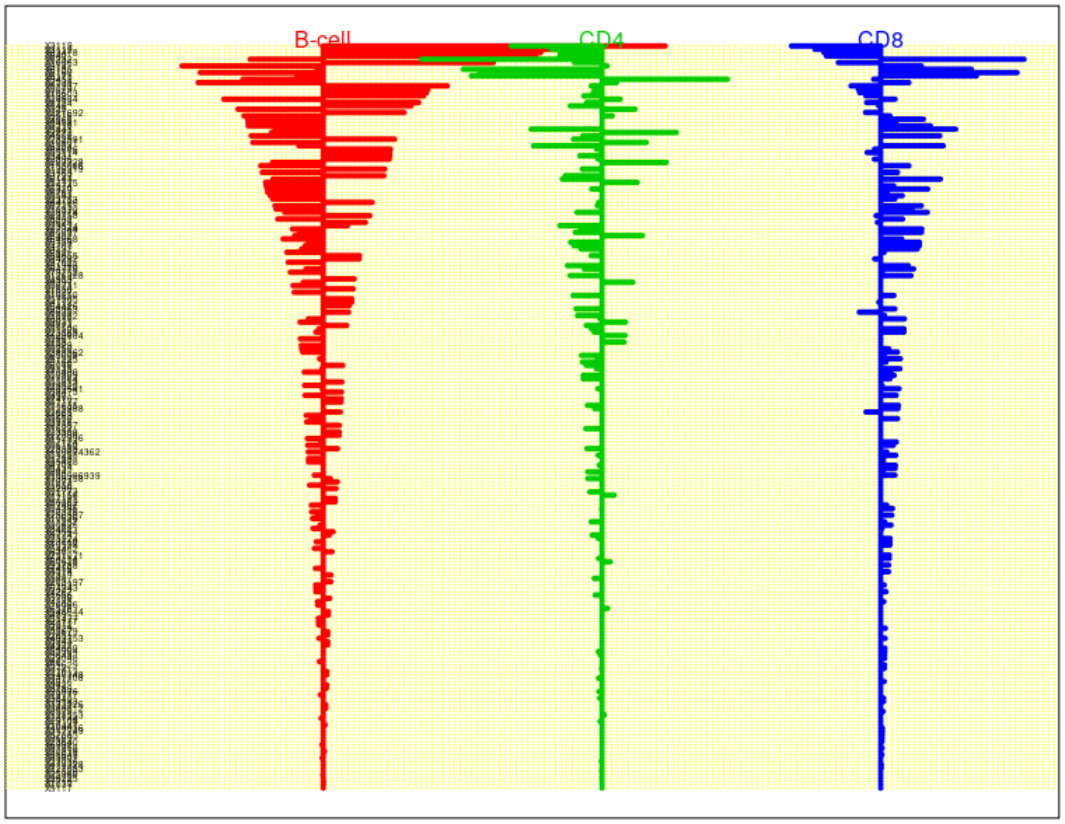
$$P(Class) = \frac{1}{1+e^{-15060.574-1.093ID-6368.893RI-39.533Na-17.752Mg-68.285Al-59.276Si-31.590K-23.270Ca-37.102Ba+15.775Fe}}$$

Assignment 3

Obtimal Threshold: 2.462881

The centroid plot for the optimal model

1.5p



The 5 most important genes

```
[1,] "X3119"
[2,] "X3112"
[3,] "X83478"
[4,] "X640"
[5,] "X6232"
```

- only the genes that have a nonzero shrinkage value for at least one of the classes are selected
- Positive values means that the genes have positive correlation with the cell type and negative have negative correlation with the cell type
- Yes, it is possible for the gene to have all positive values in the centroid plot.

Number of genes selected: 224

Assignment 4

#Neural Nets

• Because each x contribute equally to the value of y, the weights are almost exactly the same,
• And because how y is originated by adding both x1 and x2 and use of the tanh activation function the neural net becomes an approximation of a linear regression.

Appendix:

```
setwd("/mnt/ahmal787/Desktop/given_files/732A99")
#Assignment 1
glass <- read.csv("glass.csv")

n=dim(glass)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.4))
train=glass[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.5))
valid=glass[id2,]
id3=setdiff(id1,id2)
test=glass[id3,]

comb <- rbind(train,valid)

model <- glm(as.factor(Class) ~.,family=binomial, data=comb)
predicted <- predict(model, newdata=test, type = 'response')
classified <- ifelse(predicted > 0.5,1,0)
con_matrix <- table(Actual = test$Class, Predicted = classified)
```

```
con_matrix


#Assignment 2
flies <- read.csv2("mortality_rate.csv")



#Assignment 3
library(pamr)
gene <- read.csv("geneexp.csv")

rownames(gene) <- 1:nrow(gene)
x_gene <- t(gene[,-2086]) # remove dependent variable
y_gene <- gene[[2086]]
mygene_data <- list(x = x_gene,
                    y = y_gene,
                    geneid = as.character(1:nrow(x_gene)),
                    genenames = rownames(x_gene))

cen_model <- pamr.train(mygene_data)
set.seed(12345)
cvmodel <- pamr.cv(cen_model, mygene_data)

#Obtimal Threshold
cvmodel$threshold[which.min(cvmodel$error)]
#The centroid plot for the optimal model
pamr.plotcv(cvmodel)


pamr.plotcen(cen_model,
             mygene_data,
             threshold = cvmodel$threshold[which.min(cvmodel$error)])

features = pamr.listgenes(cen_model,
                          mygene_data,
                          threshold = cvmodel$threshold[which.min(cvmodel$error)],
                          genenames = TRUE)


#The 5 most important genes
as.matrix(features[1:5,2])

#Number of genes selected
nrow(features)


#Assignment 4
#Mixture Models
library(mvtnorm)
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
```

```
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=300 # number of training points
D=2 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
# Sampling the training data
mu1<-c(0,0) # component 1
Sigma1 <- matrix(c(5,3,3,5),D,D)
dat1<-rmvnorm(n = 100, mu1, Sigma1)
mu2<-c(5,7) # component 2
Sigma2 <- matrix(c(5,-3,-3,5),D,D)
dat2<-rmvnorm(n = 100, mu2, Sigma2)
mu3<-c(8,3) # component 3
Sigma3 <- matrix(c(3,2,2,3),D,D)
dat3<-rmvnorm(n = 100, mu3, Sigma3)
plot(dat1,xlim=c(-10,15),ylim=c(-10,15))
points(dat2,col="red")
points(dat3,col="blue")
x[1:100,]<-dat1
x[101:200,]<-dat2
x[201:300,]<-dat3
plot(x,xlim=c(-10,15),ylim=c(-10,15))

K=3 # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional means
Sigma <- array(dim=c(D,D,K)) # conditional covariances
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the parameters
set.seed(12345)
pi <- runif(K,0,1)
pi <- pi / sum(pi)
for(k in 1:K) {
  set.seed(12345)
  mu[k,] <- runif(D,0,5)
  Sigma[,,k]<-c(1,0,0,1)
}
pi
mu
Sigma

for(it in 1:max_it) {
  # E-step: Computation of the fractional component assignments (responsiblities)
  for (n in 1:N) {
    phi = c()
    for (j in 1:K) {
      phi = sum(dmvnorm(x, mean = rep(mu[j], length = ncol(x)), sigma = Sigma[,,j]))
    }
    z[n,] = (pi*phi) / sum(pi*phi)
  }
  #Log likelihood computation.
  # Your code here
```

```
  likelihood <-matrix(0,1000,K)
  llik[it] <-0
  for(n in 1:N)
  {
    for (k in 1:K)
    {
      likelihood[n,k] <- pi[k]*prod( ((mu[k,]^x[n,])*((1-mu[k,])^(1-x[n,]))))
    }
    llik[it]<- sum(log(rowSums(likelihood)))
  }
  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()

  if (it > 1)
  {
    if (llik[it]-llik[it-1] < min_change)
    break()
  }

  mu<- (t(z) %*% x) /colSums(z)
  pi <- colSums(z)/N
  for(n in N){
    for(k in K){
      Sigma <- sum((x[n,k]-mu[k])*(x[n,k]-mu[k])*z[n,k])
    }
  }
pi
mu
plot(llik[1:it],
     type="o",
     main = "Log Likelihood",
     xlab = "Number of Iterations",
     ylab = "Log Likelihood")
}

#Neural Nets
library(neuralnet)
set.seed(1234567890)
x1 <- runif(1000, -1, 1)
x2 <- runif(1000, -1, 1)
tr <- data.frame(x1,x2, y=x1 + x2)
winit <- runif(9, -1, 1)
nn<-neuralnet(formula = y ~ x1 + x2, data = tr, hidden = c(1), act.fct = "tanh")
plot(nn)
```