Deep Learning

Indrajith Wasala Mudiyanselage

Question 01

a) The classes are linearly separable. A good guess would be X2=X1 line (the 45 degree line). Check figure 2 black line.

b) Yes, the algorithm converge. The final weights are,

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0.000000 \\ 7.591054 \\ -7.649974 \end{bmatrix}$$

The training error rate is 0 and the test error rate is 0.005.

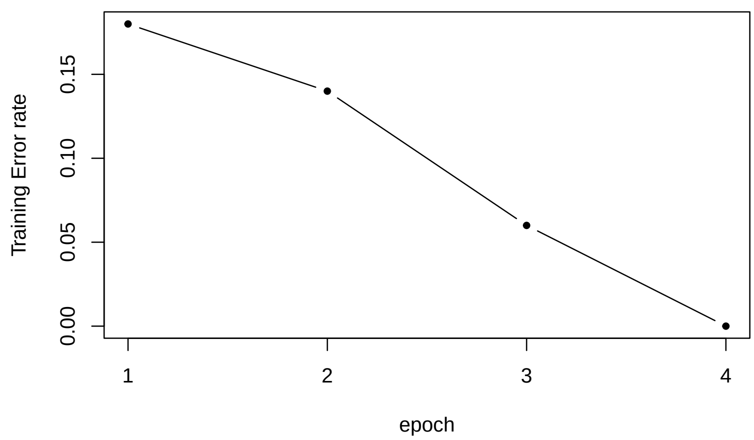

Figure 1: Training Error rate

c) Note that both Perceptron loss and Hinge loss have the same gradient $-y\mathbf{x} = -\{(y - \hat{y}/2)\}\mathbf{x}$ (Shifted).Since two classes are labeled as ± 1 no change in the code. The only difference is the loss function (question does not ask to calculate the loss). Yes, the algorithm converge. The final weights are same as above w (Part b). The training error rate is 0 and the test error rate is 0.005.

d) The decision boundary of the classifiers (for both b and c) is $7.591054X_1 - 7.649974X_2 = 0 \Rightarrow X_2 = \frac{7.591054}{7.649974}X_1$.The slope of the decision boundary is $0.992298 \approx 1$.Check the blue line of figure 2. Therefore the guess and actual boundaries are very close to each other.

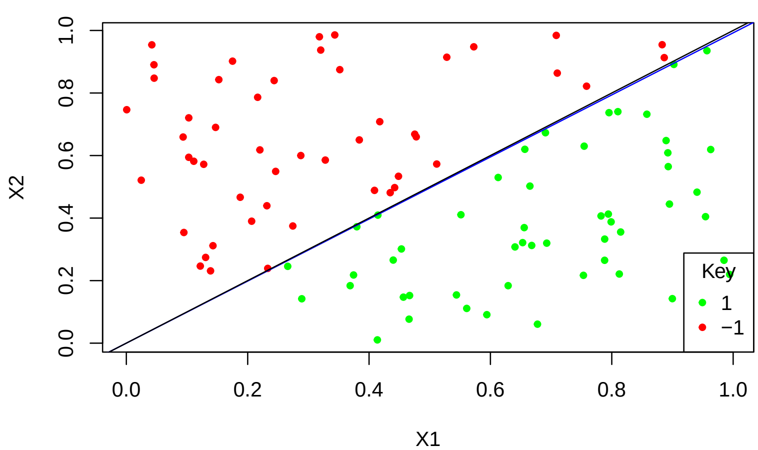

Figure 2: Plot of training data

e) Since both classifiers give the same boundary. There is no difference in the performance. Both methods perform well with a test error rate of 0.005.

Question 02

a) The pairwise scatter plot visually reveals that the suggested response variable (sales) has a moderately positive linear relationship with the variables TV, and radio and weak positive linear relationship between newspaper. Also, note that variable newspaper has a moderate positive linear relationship between radio. The correlation matrix confirms the above information.The Histograms visually suggest that the response variable sales has a symmetric distribution. But visually, the predictors does not have symmetric distributions.

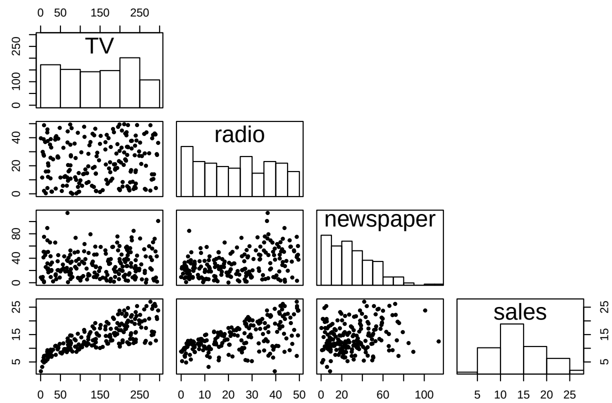

Figure 3: Pairwise scatterplots

	TV	radio	newspaper	sales
TV	1.00	0.05	0.06	0.78
radio	0.05	1.00	0.35	0.58
newspaper	0.06	0.35	1.00	0.23
sales	0.78	0.58	0.23	1.00

Table 1: Correlation Matrix

b)

```
##                Estimate Std. Error     t value      Pr(>|t|)
## (Intercept) 14.02250000  0.1191836 117.6546285 9.153929e-184
## TV           3.92908869  0.1197578  32.8086244  1.509960e-81
## radio        2.79906919  0.1278493  21.8934961  1.505339e-54
## newspaper   -0.02259517  0.1278625  -0.1767146  8.599151e-01
```

These estimates are obtained by solving normal equations or minimizing sum of squares i.e. $(X'X)\hat{\beta} = X'Y \Rightarrow \hat{\beta} = (X'X)^{-1}X'Y$.

c) Yes, the algorithm converge. The final weights are,

$$\mathbf{w} = \begin{bmatrix} Intercept(w_0) \\ TV(w_1) \\ radio(w_2) \\ newspaper(w_3) \end{bmatrix} = \begin{bmatrix} 14.02249998 \\ 3.92908866 \\ 2.79906577 \\ -0.02259174 \end{bmatrix}$$

The estimates reported in part (b) are very close to these estimates. If we run more iterations (>1000), we can get even more closer estimates.

R Codes

```r
knitr::opts_chunk$set(echo = TRUE}


## ----setup, include=FALSE--------------------------------------------------------
knitr::opts_chunk$set(echo = TRUE)


## ----include=FALSE---------------------------------------------------------------
### Question 01

training<-read.csv("/Users/indrajithwasalamudiyanselage/Documents/UTD/Academic/10-Spring 2023/STAT 6390/Mini Proj
test<-read.csv("/Users/indrajithwasalamudiyanselage/Documents/UTD/Academic/10-Spring 2023/STAT 6390/Mini Projects


## ----include=FALSE---------------------------------------------------------------
## b)

# Sign activation function
sgn<-function(x){
  if(x<0){return(-1)}
  else{return(1)}
}


## ----include=FALSE---------------------------------------------------------------

# Initializing
w<-matrix(c(0,0,0),byrow = T) # Initial weights
alpha<-1 # learning rate
n<-length(training$x1.train) # Number of observations
b<-rep(1,n) # bias

X<-t(matrix(c(b,training$x1.train,training$x2.train),nrow = n,ncol = 3,byrow = F)) # X as a matrix
y<-matrix(training$y.train)
yhat<-matrix(nrow = n,ncol = 1)
err.rate<-c()

for (epoch in 1:10) { # Maximum number of epoch is 10
  for (i in 1:n) {
    yhat[i,1]<-sgn(t(w)%*%X[,i])
    w<-w+(alpha*(y[i,1]-yhat[i,1])*X[,i]) # Update weights
  }

  num.match<-sum(yhat==y) # Number of correct classifications for each epoch
  err.rate[epoch]<-1-(num.match/n) # Error rate for each epoch
  if (num.match==n) {
  break
  }
}


## ----eval=FALSE, include=FALSE---------------------------------------------------
## # Final weights
## w


## ----echo=FALSE,fig.align="center",fig.cap="Training Error rate",out.width = "50%"----
# Error rate
plot(1:epoch,err.rate, xlab = "epoch", ylab = "Training Error rate",type = "b",pch=20,xaxt = "n")
axis(1, at = seq(1,epoch,1), las=1)
```

```r
## ----include=FALSE-------------------------------------------------------------
# Calculating test error rate

nt<-length(test[,1])
bt<-rep(1,nt) # bias

Xt<-t(matrix(c(bt,test$x1.test,test$x2.test),nrow = nt,ncol = 3,byrow = F)) # X as a matrix
yt<-matrix(test$y.test)
ythat<-matrix(nrow = nt,ncol = 1)

for (i in 1:nt) {
    ythat[i,1]<-sgn(t(w)%*%Xt[,i])
}

t.num.match<-sum(ythat==yt) # Number of correct classifications
t.err.rate<-1-(t.num.match/nt) # Error rate for each epoch
t.err.rate


## ----include=FALSE-------------------------------------------------------------
## c)

# Note that both Perceptron loss and Hinge loss have the same gradient (Shifted).
# Initializing
w<-matrix(c(0,0,0),byrow = T) # Initial weights
alpha<-1 # learning rate
n<-length(training$x1.train) # Number of observations
b<-rep(1,n) # bias

X<-t(matrix(c(b,training$x1.train,training$x2.train),nrow = n,ncol = 3,byrow = F)) # X as a matrix
y<-matrix(training$y.train)
yhat<-matrix(nrow = n,ncol = 1)
err.rate<-c()

for (epoch in 1:10) { # Maximum number of epoch is 10
  for (i in 1:n) {
    yhat[i,1]<-sgn(t(w)%*%X[,i])
    w<-w+(alpha*(y[i,1]-yhat[i,1])*X[,i]) # Update weights
  }

  num.match<-sum(yhat==y) # Number of correct classifications for each epoch
  err.rate[epoch]<-1-(num.match/n) # Error rate for each epoch
  if (num.match==n) {
  break
  }
}


## ----echo=FALSE,fig.align="center",fig.cap="Plot of training data",out.width = "50%"----
## a) and d)

plot(training$x1.train,training$x2.train,col = ifelse(training$y.train == 1, "green", "red"),pch = 20, xlab = "X1
legend("bottomright", pch = 20, col = c("green", "red"), legend = c(1, -1),title = "Key")
abline(0,7.591054/7.649974,col="blue")
abline(0,1,col="black")
```

```r
## ----include=FALSE---------------------------------------------------------------
### Question 02

Advertising<-read.csv("/Users/indrajithwasalamudiyanselage/Documents/UTD/Academic/10-Spring 2023/STAT 6390/Mini P


## ----echo=FALSE,warning=FALSE,fig.align="center",fig.cap="Pairwise scatterplots",out.width = "40%"----
## a)

# Pairwise scatterplots

panel.hist <- function(x, ...)
{
usr <- par("usr"); on.exit(par(usr))
par(usr = c(usr[1:2], 0, 1.5) )
h <- hist(x, plot = FALSE)
breaks <- h$breaks; nB <- length(breaks)
y <- h$counts; y <- y/max(y)
rect(breaks[-nB], 0, breaks[-1], y, ...)
}

pairs(Advertising[,2:5],pch = 20,cex = 0.9,upper.panel=NULL,diag.panel = panel.hist)


## ---- echo=FALSE,results="asis"--------------------------------------------------
# Correlation Matrix
library(xtable)
options(xtable.comment=FALSE)
xtable(cor(Advertising[,2:5]),caption = "Correlation Matrix",placement = "H")


## ----echo=FALSE------------------------------------------------------------------
## b)

Adver<-apply(Advertising[,2:4],2,function(x) (x-mean(x))/sd(x)) # Standardize X
Adver<-as.data.frame(cbind(Adver,Advertising$sales))
colnames(Adver)[4]<-"sales"
reg<-lm(sales~TV+radio+newspaper, data=Adver) # Fitting regression
summary(reg)$coefficient


## ----include=FALSE---------------------------------------------------------------
## c)

# Initializing
w2<-matrix(c(0,0,0,0),byrow = T) # Initial weights
alpha2<-0.0001 # learning rate
n2<-length(Adver[,1]) # Number of observations
b<-rep(1,n2) # bias

X2<-matrix(c(b,Adver$TV,Adver$radio,Adver$newspaper),nrow = n2,ncol = 4,byrow = F) # X as a matrix
y2<-matrix(Adver$sales)
RSS<-matrix(nrow = 1000,ncol = 1)
err.rate2<-c()
for (epoch in 1:1000) { # Number of iterations is 1000
    sm<-t(y2-t(t(w2)%*%t(X2)))%*%X2
    RSS[epoch]<-sum(y2-t(t(w2)%*%t(X2))) # Calculate RSS for each iteration
    w2<-w2+(alpha2*t(sm)) # Update weights
}

w2
```