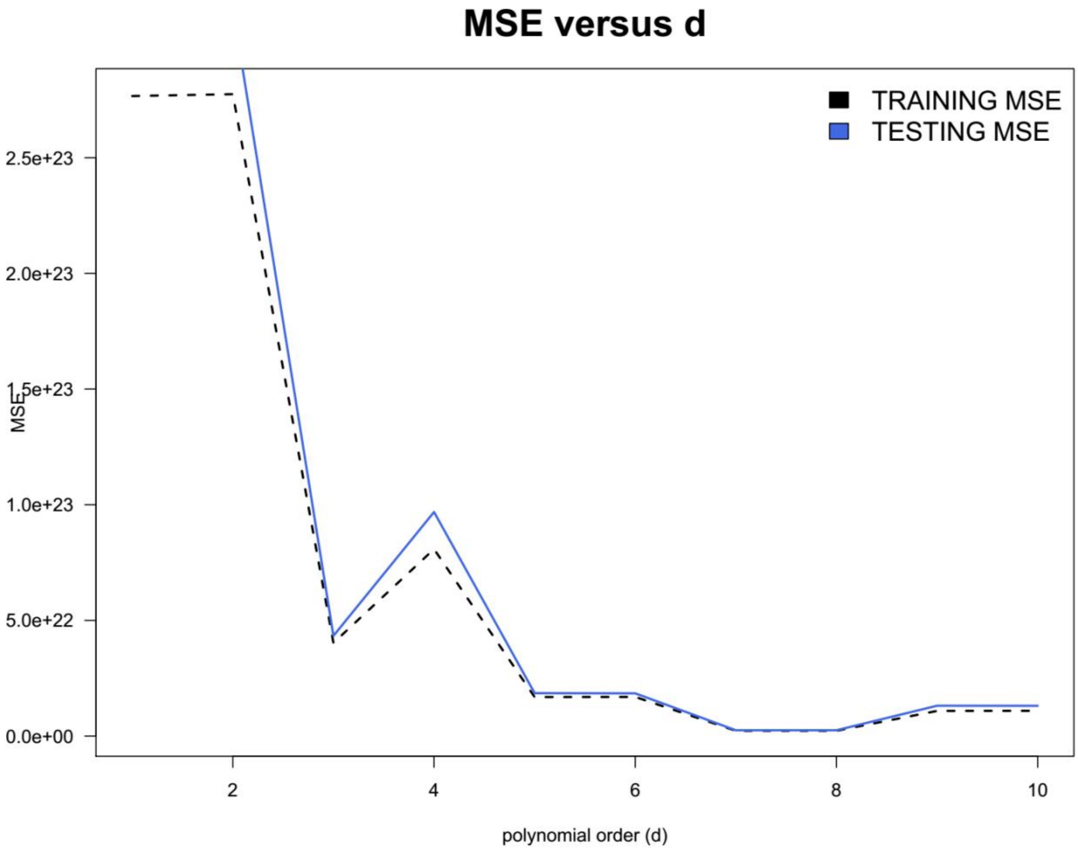
SDGB7847
Minxia Ji
Machine Learning Homework1

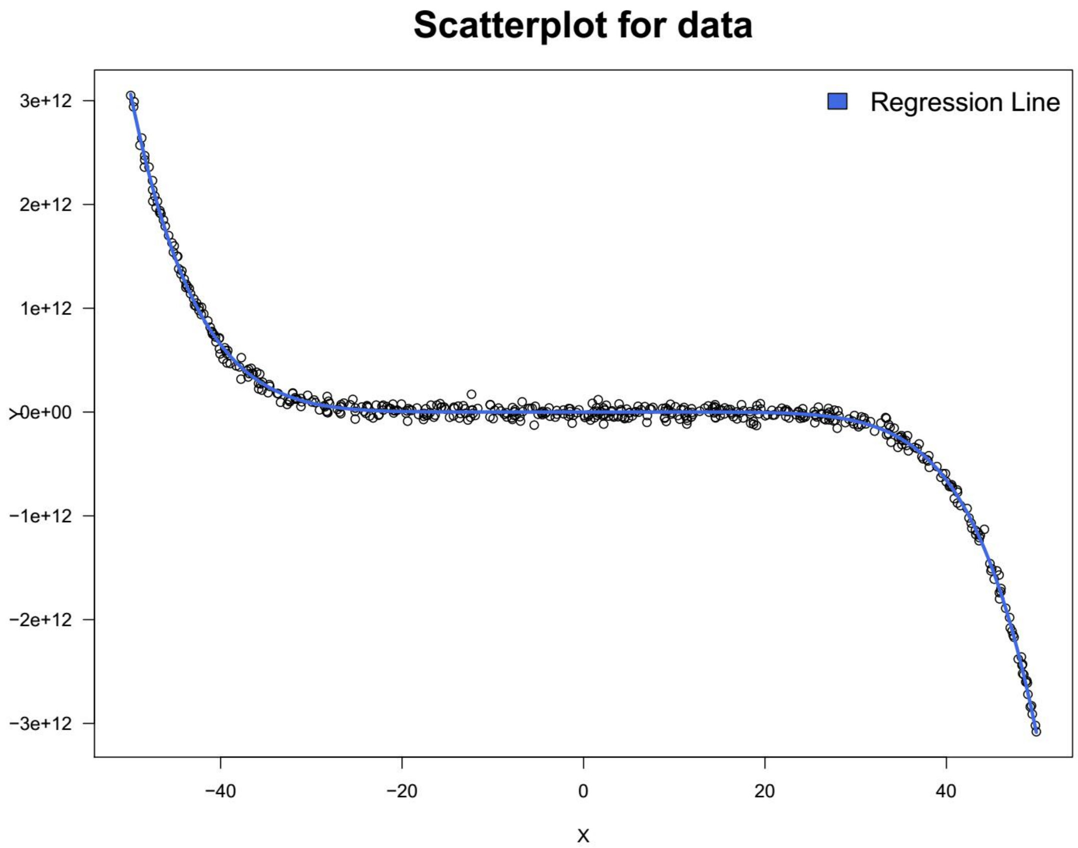
Question 1

Polynomial order (d)	Training MSE	Testing MSE
1	2.868854e+23	3.062569e+23
2	2.798735e+23	3.203375e+23
3	4.344710e+22	3.920112e+22
4	8.756376e+22	9.066979e+22
5	1.890361e+22	1.635627e+22
6	1.885520e+22	1.665282e+22
7	2.416234e+21	2.345898e+21
8	2.428463e+21	2.350603e+21
9	1.294889e+22	1.061315e+22
10	1.296568e+22	1.071135e+22

I will select d = 7 for my model, since when d= 7, my model has a minimum MSE for the training data.

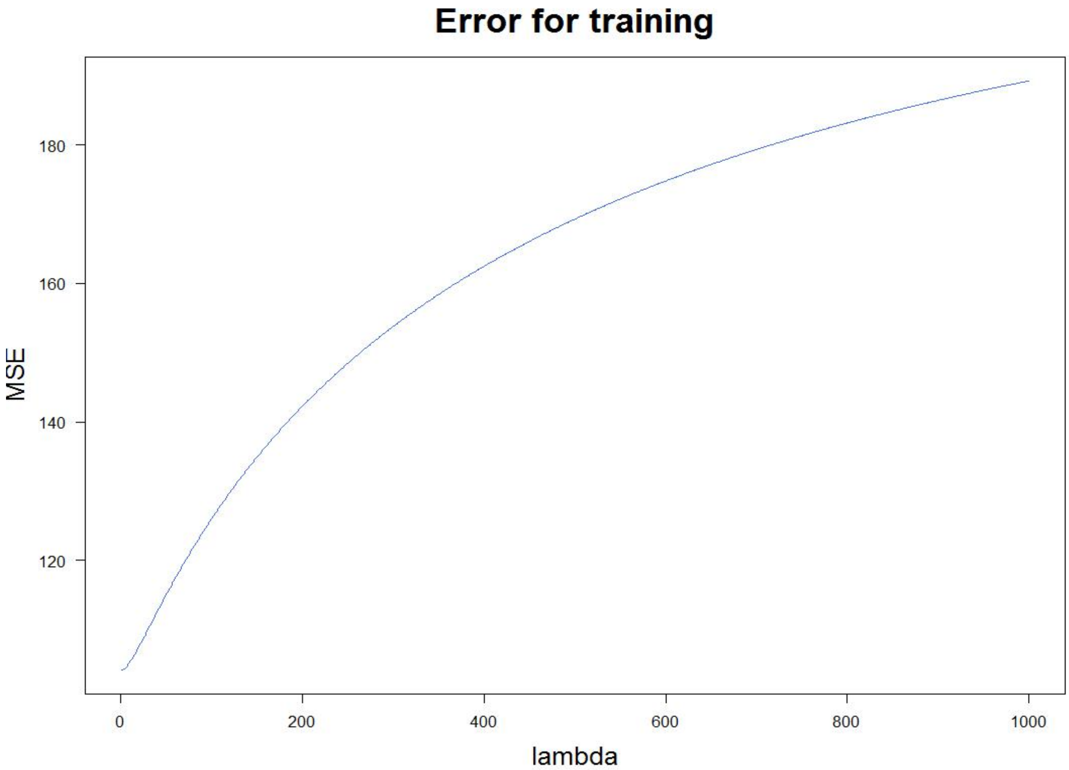


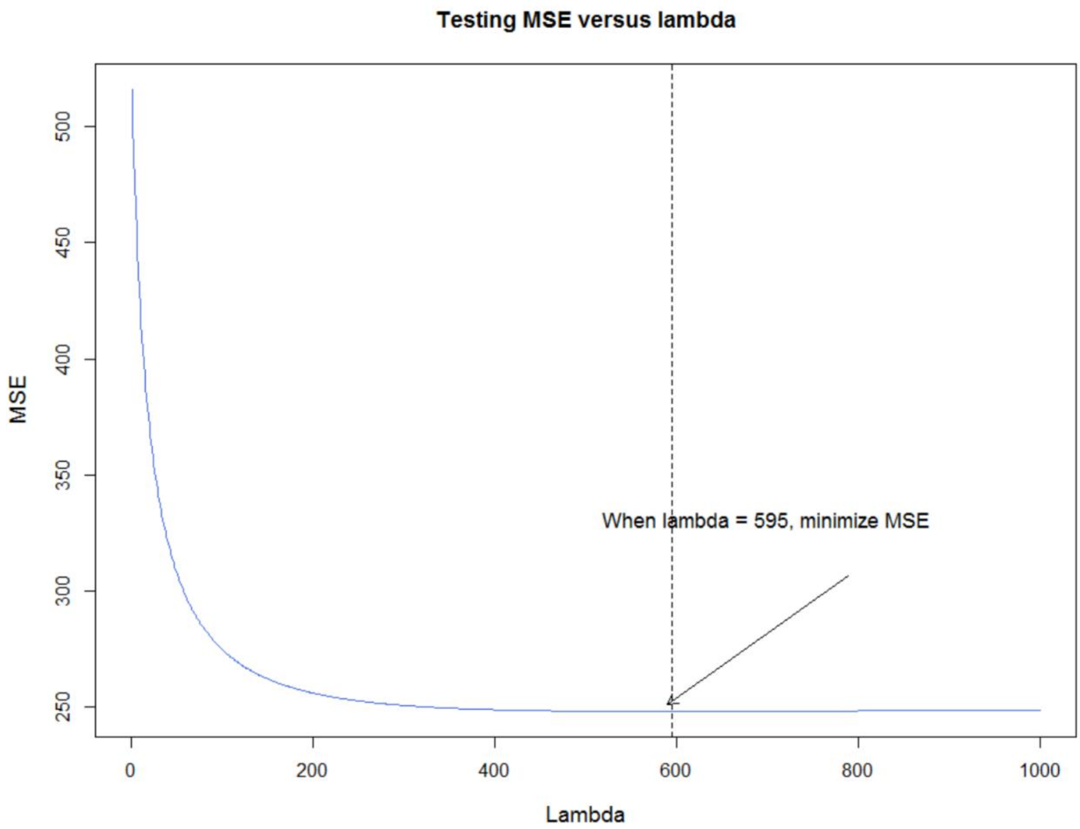
The figure，MSE versus d，shows that when d = 7, the model get minimum MSE for both training and testing data.



The figure，Scatter plot for data, shows that when d=7, the regression model fit the data well.

Question 2





Based on the plots,as lambda increasing, the MSE for training data increase while that for testing data decrease.

When lambda=595, the MSE for testing data is smallest.

Question 3

1.

$\sigma'(x) = e^{-x} * (1+e^{-x})^{-2}$;

while $\sigma(1-\sigma) = \frac{1}{1+e^{-x}} * \frac{e^{-x}}{1+e^{-x}} = e^{-x} * (1+e^{-x})^{-2}$,

therefore proved the first derivative of function $\sigma(x) = \frac{1}{1+e^{-x}}$ is equal to $\sigma(1-\sigma)$.

2.$\sigma(-x) = \frac{1}{1+e^{x}} = \frac{1*e^{-x}}{(1+e^{x})*e^{-x}} = \frac{e^{-x}}{e^{-x}+1} = 1 - \frac{1}{1+e^{-x}} = \sigma(1-\sigma)$

Let

$$y = \sigma(x) = \frac{1}{1+e^{-x}}$$
$$\frac{1}{y} = 1 + e^{-x}$$
$$e^{-x} = \frac{1-y}{y}$$
$$e^{x} = \frac{y}{1-y}$$
$$x = ln(\frac{y}{1-y})$$

Therefore $\sigma^{-1}(y) = x = ln(\frac{y}{1-y})$

3.

$$\sinh(x) = \frac{e^x - e^{-x}}{2}$$
$$\cosh(x) = \frac{e^x + e^{-x}}{2}$$
$$\Rightarrow \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Therefore,

$$\frac{1+\tanh(x)}{1-\tanh(x)} = \frac{\frac{2e^x}{e^x+e^{-x}}}{\frac{2e^{-x}}{e^x+e^{-x}}} = \frac{2e^x}{2e^{-x}} = e^{2x}$$

Question 4

$$\nabla_\theta LL = \sum_{i=1}^{N} (\alpha_i - y_i) x_{ij}$$

Using Linear Algebra, we can get

$$= (\alpha - y)^T X$$
$$= X^T(\alpha - y)$$

$\therefore$

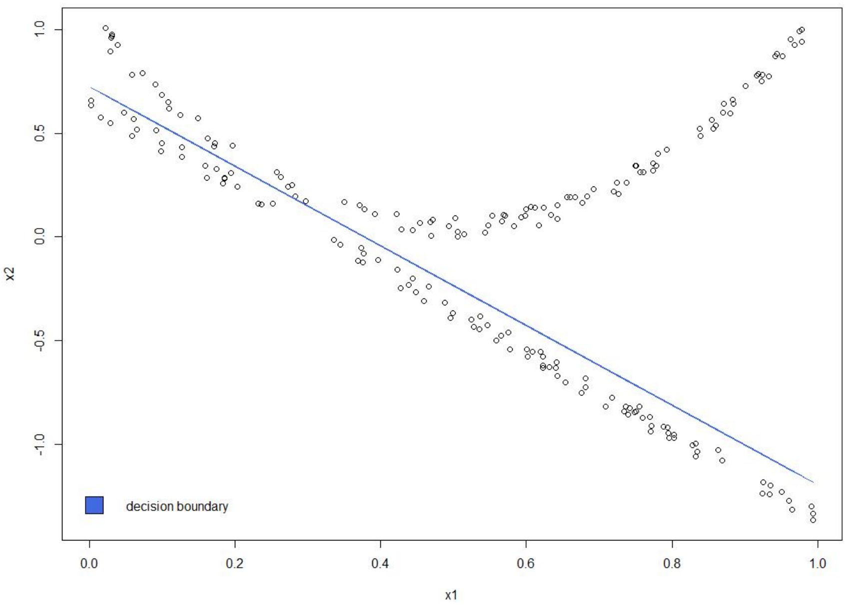
$$\theta_{t+1} = \theta_t - \eta * X^T(\alpha - y)$$

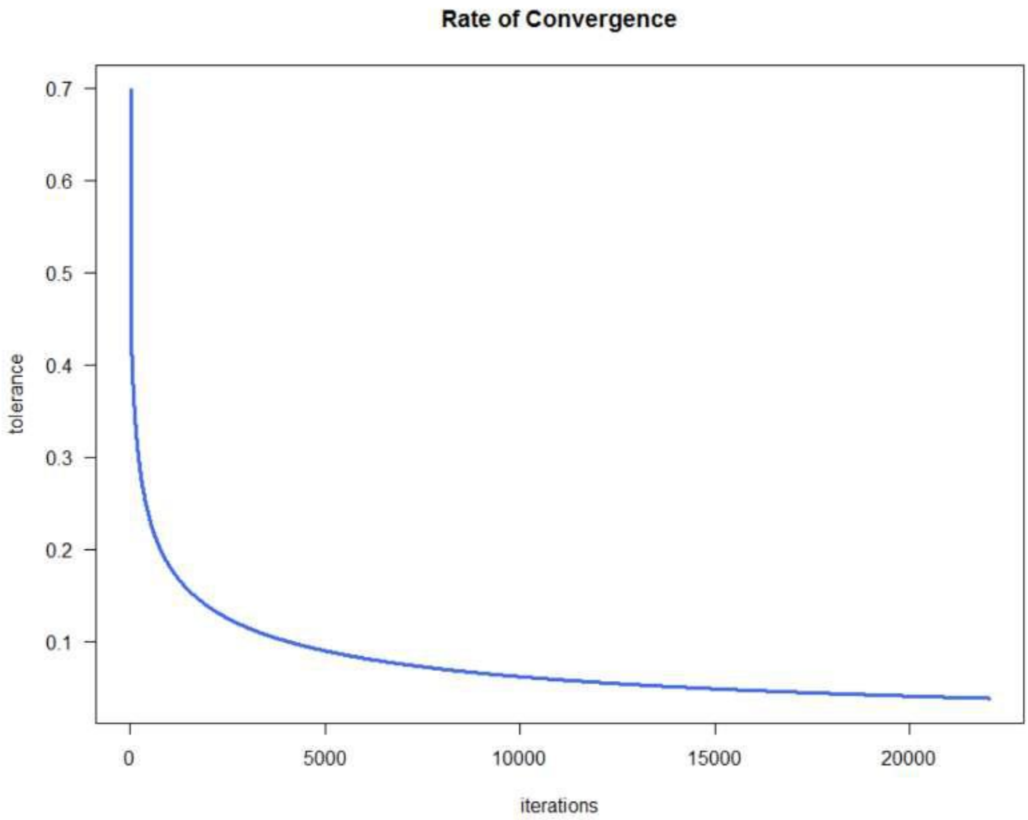
Question 4

Table of parameter estimates

Stepsize	Tolerance	Iterations	Theta1	Theta2	Theta3
0.5	0.1	3182	44.13702	24.49384	-17.38490
0.5	0.01	26547	98.33215	51.14036	-37.13213
0.5	0.001	49975	195.66345	98.86386	-72.47661
0.1	0.1	2054	9.207652	7.117595	-4.079840
0.1	0.01	19768	49.76636	27.26594	-19.48302
0.1	0.001	42896	124.53653	63.98038	-46.63449

Decision Boundary(when stepsize=0.5,tolerance=0.01)





Question 5

$$\frac{\partial}{\partial\theta}LL = \sum_{i=1}^{N}(\alpha_i - y_i)x_{ij}$$

$$\frac{\partial^2}{\partial\theta_j\partial\theta_j}LL = \sum_{i=1}^{N} x_{ij}(\frac{\partial}{\partial\theta}\alpha_i) = \sum_{i=1}^{N} x_{ij}x_{ik}\alpha_i(1-\alpha_i)$$

$$= \vec{z_j}^T S\vec{z_k} \qquad \text{where } \vec{z_j} = (x_{ij}, \cdots, x_{nj})^T$$

$$S = \begin{pmatrix} \alpha_1(1-\alpha_1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \alpha_n(1-\alpha_n) \end{pmatrix} \qquad X = \begin{pmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nd} \end{pmatrix}$$

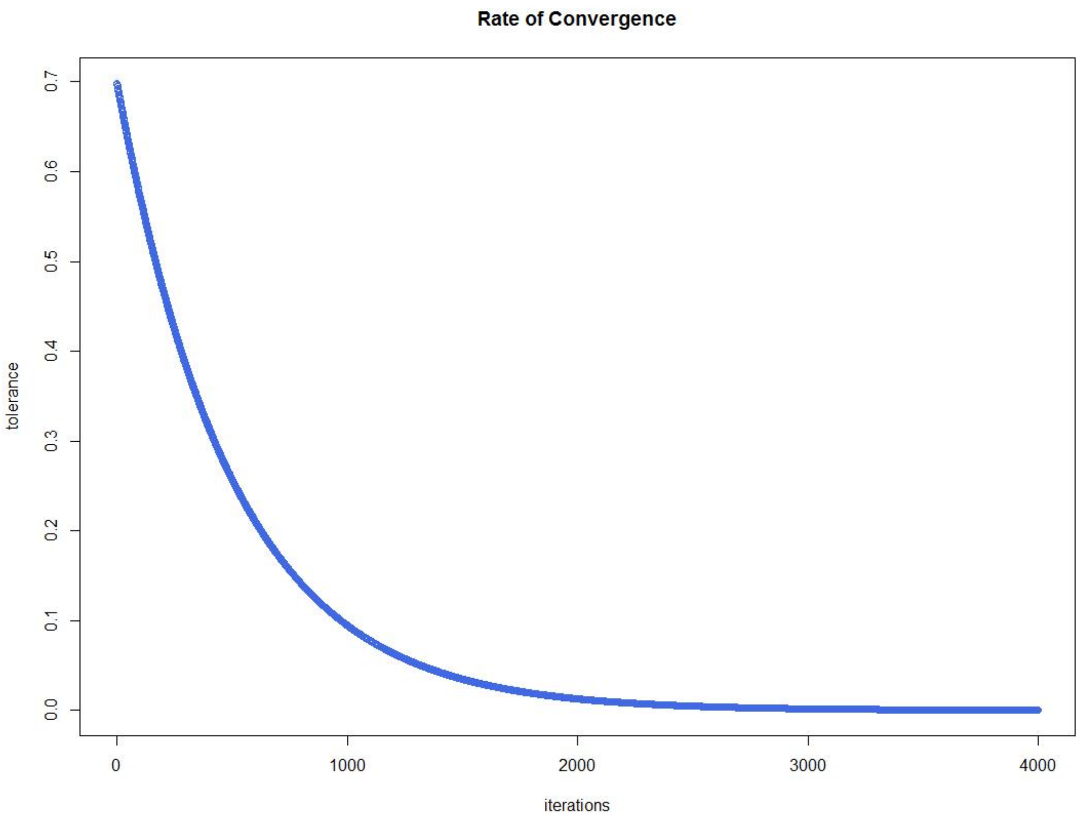
$$\nabla^2_\theta LL = \frac{\partial^2}{\partial\theta_j\partial\theta_j}LL = X^TSX$$

$$H = \nabla^2 LL = X^TSX \qquad \nabla LL = X^T(\alpha - y)$$

$$\theta_{t+1} = \theta_t - H^{-1}g$$
$$= \theta_t - (X^TSX)^{-1}X^T(\alpha - y)$$

Table of parameter estimates

Stepsize	Tolerance	Iterations	Theta1	Theta2	Theta3
0.1	0.1	973	57.15769	29.70850	-21.69674
0.1	0.01	1866	133.80551	67.67338	-49.70085
0.1	0.001	2945	238.60526	119.29102	-87.88332
0.5	0.1	896	81.25300	42.20398	-30.83149
0.5	0.01	1503	176.67818	89.15783	-65.51135
0.5	0.001	2761	298.4268	149.0758	-109.8346



Convergence rate for Newton’s method is much higher than that for gradient descent.

```
#code
#SDGB7847
#Minxia Ji
#SDGB7847
#Minxia Ji
#Machine Learning Homework1
library(MASS)
library(onion)
###############
# Question 1    #
###############
data.q1 <- read.table('data1.txt',header = TRUE,sep = '\t')
plot(data.q1$X,data.q1$Y)
#split the data into training and testing data
temp <- sample(1:nrow(data.q1),ceiling(nrow(data.q1))/2)
training <- data.q1[temp,]
testing <- data.q1[-temp,]
#save training and testing as matrics
x.training <- as.matrix(training[,1])
y.training <- as.matrix(training[,2])

x.testing <- as.matrix(testing[,1])
y.testing <- as.matrix(testing[,2])
#write the function
f <- function(x.train,y.train,x.test,y.test,d){
  a<-matrix(rep(1,nrow(x.train)),nrow(x.train),1)
  b<-matrix(rep(1,nrow(x.test)),nrow(x.test),1)
  SSE.train<-rep(NA,d)
  MSE.train<-rep(NA,d)
  SSE.test<-rep(NA,d)
  MSE.test<-rep(NA,d)
  B.container<-list(NULL)
  for (i in 1:d) {
    a<-cbind(a,x.train^i)
    b<-cbind(b,x.test^i)
    B<-matrix(nrow = ncol(a),ncol = 1)
    B<-ginv(t(a)%*%a)%*%(t(a)%*%y.train)
    #MSE of training data
    SSE.train[i]<-t(y.train-a%*%B)%*%(y.train-a%*%B)
    MSE.train[i]<-SSE.train[i]/(nrow(a)-ncol(a))
    #MSE of testing data
    SSE.test[i]<-t(y.test-b%*%B)%*%(y.test-b%*%B)
    MSE.test[i]<-SSE.test[i]/(nrow(b)-ncol(b))
    #list of beta
```

```
        B.container[[i]] <-B
    }
    return(list("MSE.training"=MSE.train,"MSE.testing"=MSE.test,
                        "Beta"=B.container))
}
#save result to save.1
save.1 <- f(x.training,y.training,x.testing,y.testing,1000)

which.min(save.1$MSE.training)
which.min(save.1$MSE.testing)

# build X varibales matrix
X <- matrix(rep(1,nrow(data.1)),nrow(data.1),1)
for (i in 1:7) {
    X <- cbind(X,as.matrix((data.1[,"X"])^i))
}
beta <- result.1$Beta[[7]]
Y.predict <- as.vector( X %*% beta )
r.predict <- data.frame(data.1$X,Y.predict)
#plot MSE
plot(y=result.1$MSE.training,x=1:10,las = TRUE,type = "l",
          main = "MSE versus d", xlab = "polynomial order (d)",ylab = "MSE",
          col="black",lwd =2,lty=2, cex.main = 2)
lines(result.1$MSE.testing,col = "royalblue",lwd =2)
legend("topright",legend = c("TRAINING MSE","TESTING MSE"),
             fill=c("black", "royalblue"), cex=1.4, bty="n")
#scatterplot
plot(y = data.q1$Y, x = data.q1$X, las = TRUE, cex.main = 2,
          main = "Scatterplot for data",xlab = "X",ylab = "Y")
lines(x=r.predict$data.q1.X,y=r.predict$Y.predict, col="royalblue",lwd=3)
legend("topright",legend = "Regression Line",
             fill="royalblue", cex=1.4, bty="n")
###############
# Question 2    #
###############
data.q2 <- read.csv("q2.txt",header = TRUE,sep = "\t")
temp2 <- sample(1:nrow(data.q2),nrow(data.q2)/2)
training.2 <- data.q2[temp2,]
testing.2 <- data.q2[-temp2,]

x.training.2 <- as.matrix(training.2[,-1])
y.training.2 <- as.matrix(training.2[,1])

x.testing.2 <- as.matrix(testing.2[,-1])
```

```
y.testing.2 <- as.matrix(testing.2[,1])

f.2 <- function(x.train,y.train,x.test,y.test,lambda){
   a <- matrix(rep(1,nrow(x.train)),nrow(x.train),1)
   a <- cbind(a,x.train)
   b <- matrix(rep(1,nrow(x.test)),nrow(x.test),1)
   b <- cbind(b,x.test)
   I <- diag(rep(1,ncol(a)))
   B <- matrix(rep(NA,ncol(a)),ncol(a),1)
   SSE.train <- rep(NA,lambda)
   SSE.test <- rep(NA,lambda)
   MSE.train <- rep(NA,lambda)
   MSE.test <- rep(NA,lambda)
   for (i in 0:lambda) {
      B <- ginv((t(a)%*%a+i*I))%*%(t(a)%*%y.train)
      SSE.train[i]<-t(y.train-a%*%B)%*%(y.train-a%*%B)
      SSE.test[i]<-t(y.test-b%*%B)%*%(y.test-b%*%B)
      MSE.train[i]<-t(y.train-a%*%B)%*%(y.train-a%*%B)/(nrow(b)-ncol(b))
      MSE.test[i]<-t(y.test-b%*%B)%*%(y.test-b%*%B)/(nrow(b)-ncol(b))
   }
   return(list("errorfortraining"=MSE.train,"errorfortesting"=MSE.test))
}
#error plots
save.2 <- f.2(x.training.2,y.training.2,x.testing.2,y.testing.2,1000)
plot(save.2$errorfortraining,las = TRUE,cex.lab = 1.6 ,cex.main =2.2,type = "l", col = "royalblue",
         xlab = "lambda",ylab = "MSE",main = "Error for training")
plot(save.2$errorfortesting, type = "l",xlab = "Lambda",ylab = "MSE",main = "Testing MSE versus
lambda",
         cex.lab = 1.2, col = "royalblue")
abline(v=which.min(result.ridge$Testing.SSE),col = "black",lty = 2)
arrows(x0=788.1744, y0=306.201, x1=590.6369, y1=251.0786, length=0.1, lwd=1.8)
text(700, 330.201,cex = 1.1, labels="When lambda = 595, minimize MSE")

###############
# Question 4    #
###############
data.q4 <- read.csv("q4.txt",header = TRUE,sep = "\t")
#spliting data
temp4 <- sample(1:nrow(data.q4),nrow(data.q4)/2)
training.4 <- data.q4[temp4,]
testing.4 <- data.q4[-temp4,]

x.training.4 <- as.matrix(training.4[,-4])
y.training.4 <- as.matrix(training.4[,4])
```

```
x.testing.4 <- as.matrix(testing.4[,-4])
y.testing.4 <- as.matrix(testing.4[,4])

f.4<-function(x.train,y.train,x.test,y.test,tolerance,stepsize){
  #initialize theta
  theta <- matrix(rep(1,ncol(x.training.4)),nrow=ncol(x.training.4),1)
  #initialize error and index
  error <- 10
  index <- 0
  while (error>tolerance) {
    temporary <- theta - stepsize*(t(x.train)%*%(1/(1+exp(-x.train%*%theta))-y.train))
    error <- as.numeric(sqrt(t(theta-temporary)%*%(theta-temporary)))
    theta <- temporary
    index <- index+1
  }
  return(list("iterationtimes"=index,"theta"=theta))
}

plot(x=save.4,y=tolerance,main = "when steosize = 0.5")
plot(x=data.q4$X1,y=data.q4$X2,xlab = "x1",ylab = "x2")
lines(x=data.q4$X1,y=(37.13-98.33*data.q4$X1)/51.14,col="royalblue")
legend("bottomleft",legend = "decision boundary",fill = "royalblue",bty = "n")
#
#for loop
tolerance.4<-seq(0.1,0.01,by=-0.0001)
result.4<-rep(NA,length(tolerance.4))
theta.container<-list(NULL)
index.4<-1
for (i in tolerance.4) {
  result.4[index.4]<-save.4<-f.4(x.training.4,y.training.4,x.testing.4,
                                  y.testing.4,stepsize     =     0.1,tolerance     =
0.01)$iterationtimes

  theta.container[index.4]<-f.4(x.training.4,y.training.4,x.testing.4,
                                 y.testing.4,stepsize = 0.1,tolerance = 0.01)$theta

  index.4 <- index.4 +1

}

plot(x=result.4,y=tolerance.4,las = TRUE,cex.lab = 1.6 ,cex.main =2.2,
     type = "l", col = "royalblue",xlab = "iterations", ylab = "tolerance",
     main = "Rate of Convergence")
```

```
###############
# Question 5    #
###############
data.q4 <- read.csv("q4.txt",header = TRUE,sep = "\t")

f.5<-function(x,y,stepsize,tolerance){
  theta <- matrix(rep(1,ncol(x)),nrow=ncol(x),1)
  g <- t(x)%*%(1/(1+exp(-x%*%theta))-y)
  theta2 <- -stepsize*g
  index <- 0
  while(Mod.onion(theta2)>tolerance){
    g <- t(x)%*%(1/(1+exp(-x%*%theta))-y)
    h <- ginv(t(x)%*%diag(as.vector(1/(1+exp(-x%*%theta))*(1-1/(1+exp(-x%*%theta)))))%*%x)
    temp <- theta - stepsize*h%*%g
    theta2 <- -stepsize*g
    theta <- temp
    index <- index +1

  }
return(list("iterationtimes"=index,"theta"=theta))
}
#for loop
tolerance.nt<-seq(0.1,0.01,by=-0.0001)
result.nt<-rep(NA,length(tolerance.nt))
theta.container<-list(NULL)
index.nt<-1
for (i in tolerance.nt) {
  result.nt[index.nt]<-f.5(x.training.4,y.training.4,0.5,i)$interationtimes

  theta.container[index.nt]<-f.5(x.training.4,y.training.4,0.5,i)$theta

  index.nt <- index.nt +1

}

plot(x=result.nt,y=tolerance.nt,las = TRUE,cex.lab = 1.6 ,cex.main =2.2,
      type = "l", col = "royalblue",xlab = "iterations", ylab = "tolerance",
      main = "Rate of Convergence")
```