復旦大學

Project Report

课程名称：统计（机器学习）概论　课程代码：DATA130003.01

姓　　名：张天然　学　　号：15300180104

学　　院：大数据学院　专　　业：数学与应用数学

Table of contents

part1 Linear Regression and Nonlinear Bases

1. Abstract

In this part, I managed to train the model using Linear Regression and improve the predictions by Polynomial basis. I also calculated training error and test error, drew some figures to show the model clearly.

2. Introduction

a. Error Calculation

We defined the mean squared error to see whether our model suit the data as:

$$\text{MSE(Mean squared error)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2$$

and the same way we can use to define our model's training error and test error as well:

$$\text{Training Error} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2$$

$$\text{Test Error} = \frac{1}{n}\sum_{i=1}^{n}(ytest_i - \widehat{ytest_i})^2$$

b. Linear Regression

In class, we have discussed the Linear Regression, now that we have linear regression function:

$$\hat{Y} = C_0 + C_1 X$$

and to mathematically obtain the close form solution of C_0 and C_1 in general:

$$\widehat{C_1} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\widehat{C_0} = \bar{y} - \widehat{C_1}\bar{x}$$

We can make the predictions of data by regression function and the parameters we obtain:

$$\hat{Y} = C_0 + C_1 X$$

c. Polynomial Basis

In class we also discuss the polynomial regression function:

$$\widehat{y_i} = b_0 + b_1 x_i + b_2 {x_i}^2 + \cdots + b_d {x_i}^d \ \ (\mathrm{i} = 1 \ldots \mathrm{d})$$

x, y, b are vectors, x's k degree x polynomial matrix is:

$$X_{poly} = \begin{bmatrix} 1 & x_1 & \cdots & {x_1}^k \\ 1 & x_2 & & {x_2}^k \\ \vdots & & \ddots & \vdots \\ 1 & x_n & \cdots & {x_n}^k \end{bmatrix}$$

then we could rewrite the function in another way:

$$y = \mathrm{X}_{poly} * b$$

we could obtain the vector b:

$$\mathrm{X}_{poly}{}^{T} * \mathrm{X}_{poly} * b = \mathrm{X}_{poly} * y$$

$$\mathrm{b} = (\mathrm{X}_{poly}{}^{T} * \mathrm{X}_{poly})^{-1} * \mathrm{X}_{poly} * y$$

Now we can make the prediction of data by polynomial regression:

$$\hat{y} = \mathrm{X}_{poly} * b$$

3. Methods and Procedures

I've write 2 scripts to train the model for dataset using the methods mentioned before in Introduction.

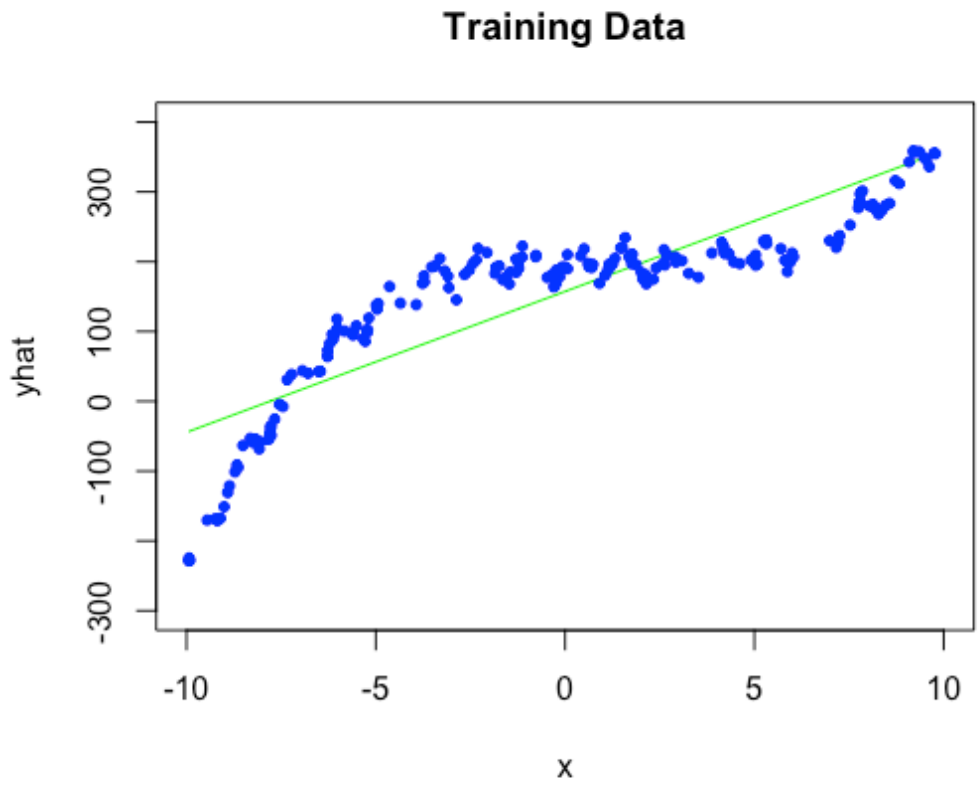
- If you run the script'project1-1.1.R', it will:

1. Load the dataset to get training data of two vectors x, y and test data of two vectors xtest and ytest as well.
2. Using linear regression to train the model as I have mentioned before.
3. Report the training error and test error:

   TrainError:　3571.36

   TestError:　3489.006

4. Draw a figure showing the training data and what the linear model looks like:



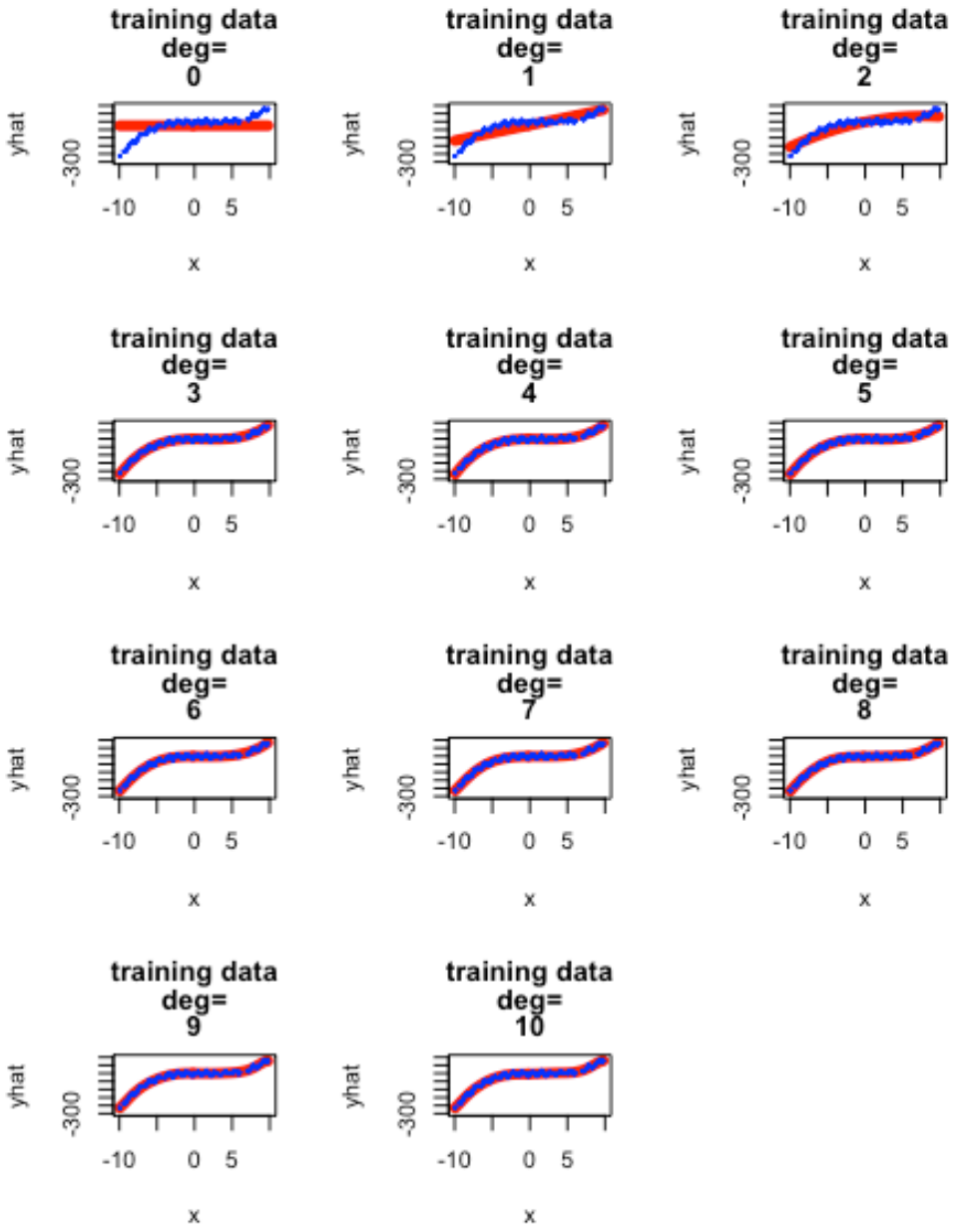
Through the figure we can find that the prediction is still problematic, thus we use Polynomial Basis to improve our predictions.

- If you run the script'project1-1.2.R', it will:

1. Load the dataset to get training data of two vectors x, y and test data of two vectors xtest and ytest as well.
2. For each k(k=0 through k=10):
   a. form x's k degree matrix Xpoly;
   b. make predictions of x and xtest and report training error and test error:

	A	B	C
1	k	TrainError	TestError
2	0	15480.51978	14390.763
3	1	3551.345871	3393.8691
4	2	2167.991943	2480.7254
5	3	252.0461009	242.804944
6	4	251.4615531	242.126427
7	5	251.1434618	239.544855
8	6	248.5828144	246.005402
9	7	247.0110471	242.887502
10	8	241.3064192	245.966196
11	9	244.0824729	248.414
12	10	238.3156265	256.422561

c. draw a figure showing the training data and what the polynomial model looks like:



4. Conclusions

Through the 2 models' report error and figures, we can tell that polynomial regression fits better to the dataset than linear regression, and with the increasing of k, the polynomial model shows lower training error and test error (the model works better).

Part2 regularization

1. Abstract

In this part, I standardize the input data and use ridge regression to train the model.

Besides, I calculated the cross-validation error to revise the model.

2. Introduction

a. Data Standardization

When we get some data attributed on different scales, we need to standardize the data on the same scale by ensuring it has zero mean and unit variance. We can do this by computing:

$$X_{ij} = \frac{X_{ij} - mean(X_j)}{var(X_j)^{\frac{1}{2}}}$$

b. Ridge Regression

Ridge method is a regularized version of least squares, with objective function:

$$\min_{delta} \|y - X * delta\|_2^2 + \mathrm{d}^2 \|delta\|_2^2$$

(d is a scalar and delta is a vector)

we have ridge function:

$$(\mathrm{X}^T X + d^2 I_n) delta = X^T y$$

solve the function, we can obtain:

$$\mathrm{delta} = \mathrm{X}^T (\mathrm{X}^T X + d^2 I_n)^{-1} y$$

Now we can make the prediction of data by ridge regression:

$$\hat{\mathrm{y}} = \mathrm{X} * \mathrm{delta}$$

c. Cross-Validation

As we discussed in class, when we don't have an explicit test set, we could use cross-validation to test our model and report the error.

To proper cross-validation procedure, we divide the data into K roughly equal parts, and for each k=1,2,…K, fit the model with parameter d to the other K-1 parts and get the error $\mathrm{E}_k(d)$, then we get the cross-validation error:

$$\mathrm{CV(d)} = \frac{1}{\mathrm{K}} \sum_{k=1}^{K} \mathrm{E}_k(d)$$

3. Methods and Procedures

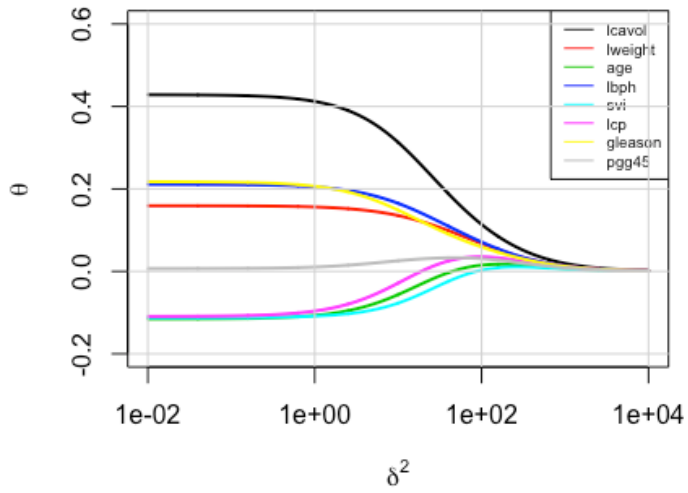
I've write 2 scripts to train the model for dataset using the methods mentioned before in Introduction.

- If you run the script'project1-2.1.R', it will:

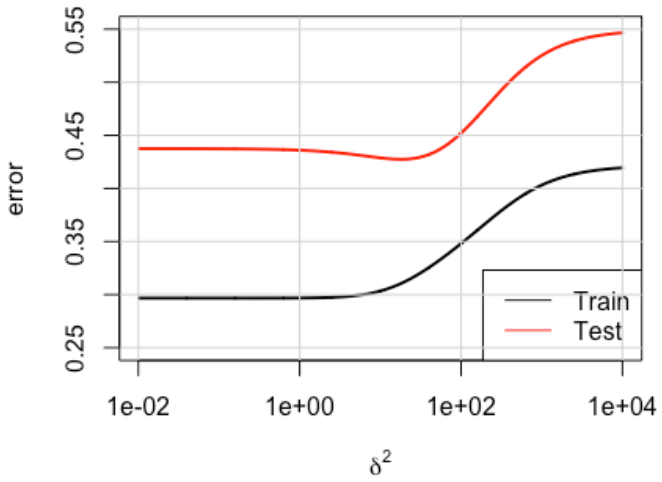
  1. Load data, randomly shuffle the data and choose the first

50 patients as the training data, the remaining patients will be the test data.

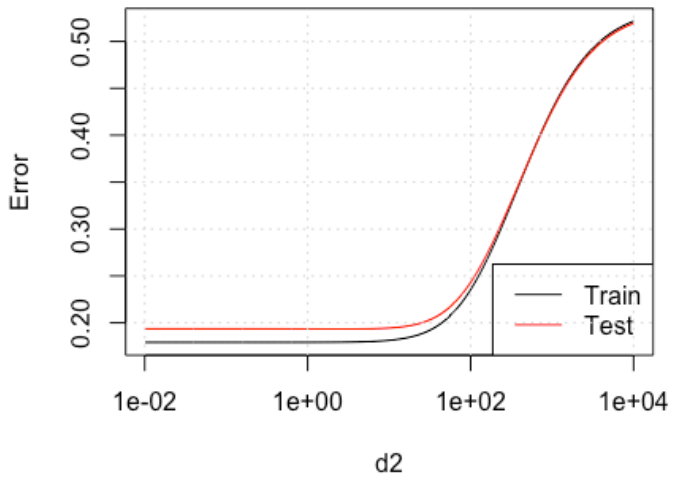
2. Standardize all data to make them on the same scale.
3. Use ridge regression to train the model as mentioned before to get yhat.
4. Draw a figure showing the regularization path:



5. Draw a figure showing the test error and train error of different delta:



- If you run the script'project1-2.2.R', it will:
  1. Load data, randomly shuffle it and divide the data into K roughly equal parts.
  2. For each k:
     a. Set k^{th} part as test data and the others as training data ;
     b. Standardize them in the same scale;
     c. Use ridge regression to train the model as mentioned before to get yhat[k];
     d. Work out the CV-error by cross-validation.
  3. Draw a figure showing train error and test error for different delta:



(k=3 for example)

part3 Rcode

1. Project1-1.1.R

```
basicdata <- read.table(file='file:///Users/ran/Desktop/project1/data/basicData.txt', header=T)
x <- basicdata$X
y <- basicdata$y
d <- dim(basicdata)

model.predict <- function(x,y){
   ymean <- mean(y)
   xmean <- mean(x)

   d1 <- 0
   d2 <- 0
   for (i in d[1]){
      d1 <- d1+(x[i]-xmean)*(y[i]-ymean)
      d2 <- d2+(x[i]-xmean)^2
   }

   c1 <<- d1/d2
   c0 <<- ymean-c1*xmean

   x*c1+c0
}

yhat <- model.predict(x,y)
TrainError <- sum((y-yhat)^2)/d[1]

xtest <- basicdata$Xtest
ytest <- basicdata$Ytest

ytesthat <- xtest*c1+c0
TestError <- sum((ytest-ytesthat)^2)/d[1]

print(TrainError)
print(TestError)

plot(x,yhat, type = 'l',xlim=c(-10,10),ylim = c(-300,400),
        col='green', main = 'Training Data')
points(x,y,pch =20,col='blue')
```

2. Project1-1.2.R

```
install.packages("MASS")
library(MASS)

basicdata <- read.table(file='file:///Users/ran/Desktop/project1/data/basicData.txt', header=T)
x <- basicdata$X
y <- basicdata$y
xtest <- basicdata$Xtest
ytest <- basicdata$Ytest

d <- dim(basicdata)

makeXpoly <- function(x,deg){
   m <- matrix(data=NA,ncol=deg+1,nrow=d)
   for (i in (0 : deg)){
      m[,i+1] <- x^i
   }
   m
}
leastSquaresBasis <- function(x,y,deg){

   Xpoly <- makeXpoly(x,deg)
   b <<- ginv(t(Xpoly)%*%Xpoly,tol=0) %*%t(Xpoly)%*%y

   Xpoly%*%b
}

par(mfrow=c(4,3))
for (i in 0:10){
   yhat <- leastSquaresBasis(x,y,i)
   plot(x,yhat, type = 'l',xlim=c(-10,10),ylim = c(-300,400),
          col='red', main = c("training data",'deg=',i),lwd=5)
   points(x,y,pch =20,col='blue',cex=0.3)

   TrainError <- sum((y-yhat)^2)/d[1]
   ytesthat <- makeXpoly(xtest,i) %*% b
   TestError <- sum((ytest-ytesthat)^2)/d[1]

   print(c('k=',i))
   print(c(' TrainError ', TrainError, ' TestError ', TestError))
}
```

3. Project1-2.1.R

```
#import data
d <- read.table(file='file:///Users/ran/Desktop/project1/data/prostate.data.txt', head=TRUE)

d1 <- d[sample.int(97),]
xmean <- matrix(data = NA, ncol = 9)
xvar <- matrix(data = NA, ncol = 9)

#initialization of xmean, xvar, x, y, xtest, ytest
for (j in 1:8){
   xmean[j] <- mean(d1[1:50, j])
   xvar[j] <- sqrt(sum((d1[1:50, j]-xmean[j])^2)/50)

   for (i in 1:97)
      d1[i,j] <- (d1[i,j]-xmean[j])/xvar[j]
}
ymean <- mean(d1[1:50,9])
for (i in 1:97)
   d1[i,9] <- d1[i,9]-ymean

x <- as.matrix(d1[1:50, 1:8])
y <- as.matrix(d1[1:50, 9])
xtest <- as.matrix(d1[51:97, 1:8])
ytest <- as.matrix(d1[51:97, 9])

#function ridge, get theta for different x,y,d2
ridge <- function(x,y,d2){
   theta <- ginv(t(x)%*%x+diag(d2,8),tol=0) %*% t(x) %*% y
   theta
}

#initialization of TrainError, TestError, d2 and th
th <- matrix(data = NA, nrow = 1000, ncol = 8)
TrainError <- matrix(data = NA, ncol = 1000)
TestError <- matrix(data = NA, ncol = 1000)
d2 <- 10^((1:1000)*0.006-2)

# for 1000 different d2, get theta, TrainError and TestError
for (i in 1:1000){
   th[i, ] <- ridge(x,y,d2[i])
   yhat <- x%*%th[i, ]
   testyhat <- xtest%*%th[i, ]
```

```
  TrainError[i] <- sqrt(sum((y-yhat)^2)/sum((y+ymean)^2))
  TestError[i] <- sqrt(sum((ytest-testyhat)^2)/sum((ytest+ymean)^2))
}

par(mfrow=c(1,1))
#draw the plots of d2-theta
plot(d2, th[,1], xlim = c(0.01,10000), ylim = c(-0.2,0.6),log='x', type='l',col=1,
      ylab=expression(theta), xlab = expression(delta^2), lwd=2)
for (i in 2:8)
  lines(d2, th[,i],col=i,lwd=2)

legend.text <- colnames(d1)[-9]
legend('topright', legend = legend.text,col=c(1:8),lty=1,cex=0.6)
grid(lty=1)

#draw polts of d2-TrainError,TestError
plot(d2, TrainError, ylim= c(0.25,0.55), log='x', type='l',lwd=2,
      xlab=expression(delta^2),ylab='error')
lines(d2,TestError,lwd=2,col=2)
legend('bottomright',legend=c('Train','Test'),col=c(1,2),lty=1,cex=1)
grid(lty=1)
```

4. Project1-2.2.R

```
d <- read.table(file='file:///Users/ran/Desktop/project1/data/prostate.data.txt', head=TRUE)

#initialization of k d1 d0 st(stands for step in each training data）

k=3
d0 <- floor(97/k)
st <- matrix(1)
for (i in 2:k)
  st[i] <- st[i-1]+d0
st[k+1] <- 98

#function ridge get theta for different x,y,d2
ridge <- function(x,y,d2){
  theta <- ginv(t(x)%*%x+diag(d2,8),tol=0) %*% t(x) %*% y
  theta
}

#initialization of TrainError, TestError and th(stands for different theta)
TrainError <- matrix(data = 0, ncol = 1000)
```

```
TestError <- matrix(data = 0, ncol = 1000)
th <- matrix(data = NA, nrow = 1000, ncol = 8)
d2 <- 10^((1:1000)*0.006-2)

# repeat to get k different TrainError and TestError
for (num in 1:k){
   set.seed(123)
   d1 <- d[sample.int(97),]
   #initialization of different x, y, xtest, ytest
   for (j in 1:8){
      xmean[j] <- mean(d1[-st[num]:-(st[num+1]-1), j])
      xvar[j] <- sqrt(sum((d1[-st[num]:-(st[num+1]-1), j]-xmean[j])^2)/(97-(st[num+1]-st[num])))

      for (i in 1:97)
         d1[i,j] <- (d1[i,j]-xmean[j])/xvar[j]
   }
   ymean <- mean(d1[-st[num]:-(st[num+1]-1), 9])
   for (i in 1:97)
      d1[i,j] <- d1[i,j]-ymean

   x <- as.matrix(d1[-st[num]:-(st[num+1]-1), 1:8])
   y <- as.matrix(d1[-st[num]:-(st[num+1]-1), 9])
   xtest <- as.matrix(d1[st[num]:(st[num+1]-1), 1:8])
   ytest <- as.matrix(d1[st[num]:(st[num+1]-1), 9])

   # for 1000 different d2, get theta, TrainError and TestError
   for (i in 1:1000){
      th[i, ] <- ridge(x, y, d2[i])
      yhat <- x %*% th[i, ]
      testyhat <- xtest%*%th[i, ]

      TrainError[i] <- TrainError[i]+sqrt(sum((y-yhat)^2)/sum((y+ymean)^2))
      TestError[i] <- TestError[i]+sqrt(sum((ytest-testyhat)^2)/sum((ytest+ymean)^2))
   }
}

#draw plots of d2-TrainError,TestError
TrainError <- TrainError/k
TestError <- TestError/k

plot(d2, TrainError, log='x', type='l',col=1,ylab='Error')
lines(d2,TestError,col=2)
legend('bottomright',legend=c('Train','Test'),col=c(1,2),lty=1,cex=1)
grid()
```