STAD80: Homework #3

Yulun Wu

Due: 2023-03-12

Contents

Question 4 (30 Points) Big Data. Big Profit. 1
Question 5 (20 Points) Discover Gravitational Waves in Your Room 8
Question 6 (25 Points) Upright Human Detection in Photos 9
Question 7 (30 Points) Sentiment Analysis on Amazon Product Reviews 14
Question 8 (15 Points) Identifying Risk Factors for the Heart Disease 19

Question 4 (30 Points) Big Data. Big Profit.

Answer:

Q1

```
library('fastDummies')
load("q1.RData")

# Create dummy variables
dataTrainAll = dummy_cols(dataTrainAll, select_columns = c("Region","City","AdX","Domain","Key_Page","Ad
remove_first_dummy = TRUE, ignore_na = TRUE, remove_selected_columns = FALSE
)

# Column bind dummy variables
dataTrainAll$Region = cbind(dataTrainAll$Region_3,dataTrainAll$Region_6)
dataTrainAll$City = cbind(dataTrainAll$City_2,dataTrainAll$City_3,dataTrainAll$City_4,dataTrainAll$City
dataTrainAll$AdX = cbind(dataTrainAll$AdX_2,dataTrainAll$AdX_3)
dataTrainAll$Domain = cbind(dataTrainAll$Domain_5KFUl5p0Gxsvgmd4wspENpn,dataTrainAll$Domain_trqRTu5Jg9q
dataTrainAll$Key_Page = cbind(dataTrainAll$Key_Page_9f4e2f16b6873a7eb504df6f61b24044,dataTrainAll$Key_Pa
dataTrainAll$Ad_Vis = cbind(dataTrainAll$Ad_Vis_1,dataTrainAll$Ad_Vis_2)
dataTrainAll$Ad_Form = dataTrainAll$Ad_Form_1

# Standardized 3 columns
dataTrainAll$Ad_Width = (dataTrainAll$Ad_Width - mean(dataTrainAll$Ad_Width))/sd(dataTrainAll$Ad_Width)
dataTrainAll$Ad_Height = (dataTrainAll$Ad_Height - mean(dataTrainAll$Ad_Height))/sd(dataTrainAll$Ad_Heig
dataTrainAll$Floor_Price = (dataTrainAll$Floor_Price - mean(dataTrainAll$Floor_Price))/sd(dataTrainAll$F

# Put needed features in a dataframe
dataTrain = cbind(dataTrainAll$Region,dataTrainAll$City,dataTrainAll$AdX,dataTrainAll$Domain,dataTrainAl

# Modify Click column
dataTrainAll$Click[dataTrainAll$Click >= 1] = 1
```

From the definition of dummy_cols, remove_first_dummy = TRUE will remove the first dummy category in

each column. In Region, Region 1 will be baseline category. In City, City 1 will be baseline category. In AdX, AdX 1 will be baseline category.In Domain, Domain 5Fa-expoBTTR1m58uG will be baseline category. In Key_Page, Key_Page 3a7eb50444df6f61b2409f4e2f16b687 will be baseline category. In Ad_Vis, Ad_Vis 0 will be baseline category. In Ad_Form, Ad_Form 0 will be baseline category.

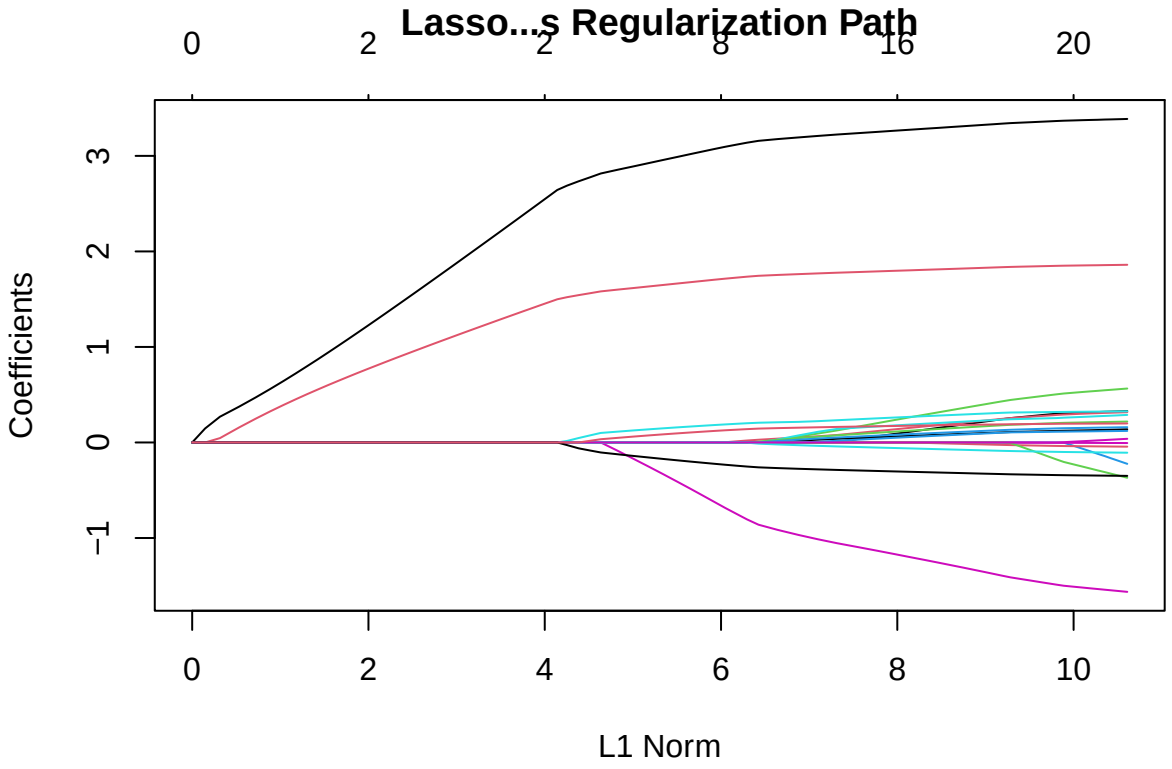
Part a

```
library(glmnet)
```

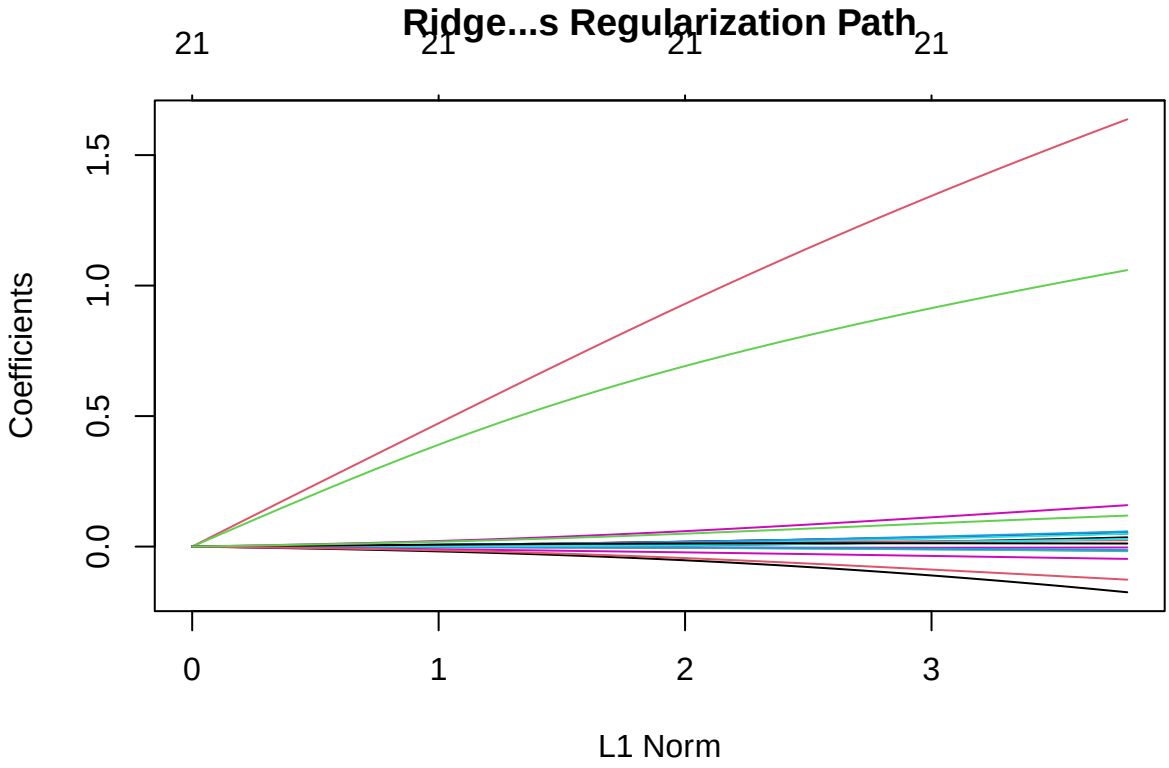
```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```
# Fit LASSO and Ridge model
fit.lasso = glmnet(dataTrain,dataTrainAll$Click,family="binomial",standardize=FALSE,alpha=1)
fit.ridge = glmnet(dataTrain,dataTrainAll$Click,family="binomial",standardize=FALSE,alpha=0)

# Plot graph
plot(fit.lasso,main="Lasso's Regularization Path",xlab="L1 Norm",ylab="Coefficients")
```




```
plot(fit.ridge,main="Ridge's Regularization Path",xlab="L1 Norm",ylab="Coefficients")
```



Part b

```
# Coefficients of LASSO model in last column of beta
dim_lassobeta = dim(fit.lasso$beta)
abs(fit.lasso$beta[,dim_lassobeta[2]])
```

```
##          V1          V2          V3          V4          V5          V6
## 0.328327302 0.316747742 0.564559336 0.223820136 0.288483703 0.038449854
##          V7          V8          V9         V10         V11         V12
## 0.138374944 0.000000000 0.043816121 0.369073039 0.121074354 0.325991793
##         V13         V14         V15         V16         V17         V18
## 1.563563657 3.386598177 1.859926722 0.218876930 0.159707350 0.107043787
##         V19         V20         V21
## 0.005933028 0.349789052 0.199048029
```

```
# Coefficients of Ridge model in last column of beta
dim_ridgebeta = dim(fit.ridge$beta)
abs(fit.ridge$beta[,dim_ridgebeta[2]])
```

```
##          V1          V2          V3          V4          V5          V6
## 0.035307143 0.050432066 0.056810182 0.017564327 0.056305359 0.003434069
##          V7          V8          V9         V10         V11         V12
## 0.050430785 0.024361039 0.015590457 0.013324720 0.049368638 0.158586037
##         V13         V14         V15         V16         V17         V18
## 0.174961513 1.636953518 1.059298698 0.057802169 0.029305423 0.047219580
##         V19         V20         V21
## 0.011823948 0.126742968 0.118714798
```

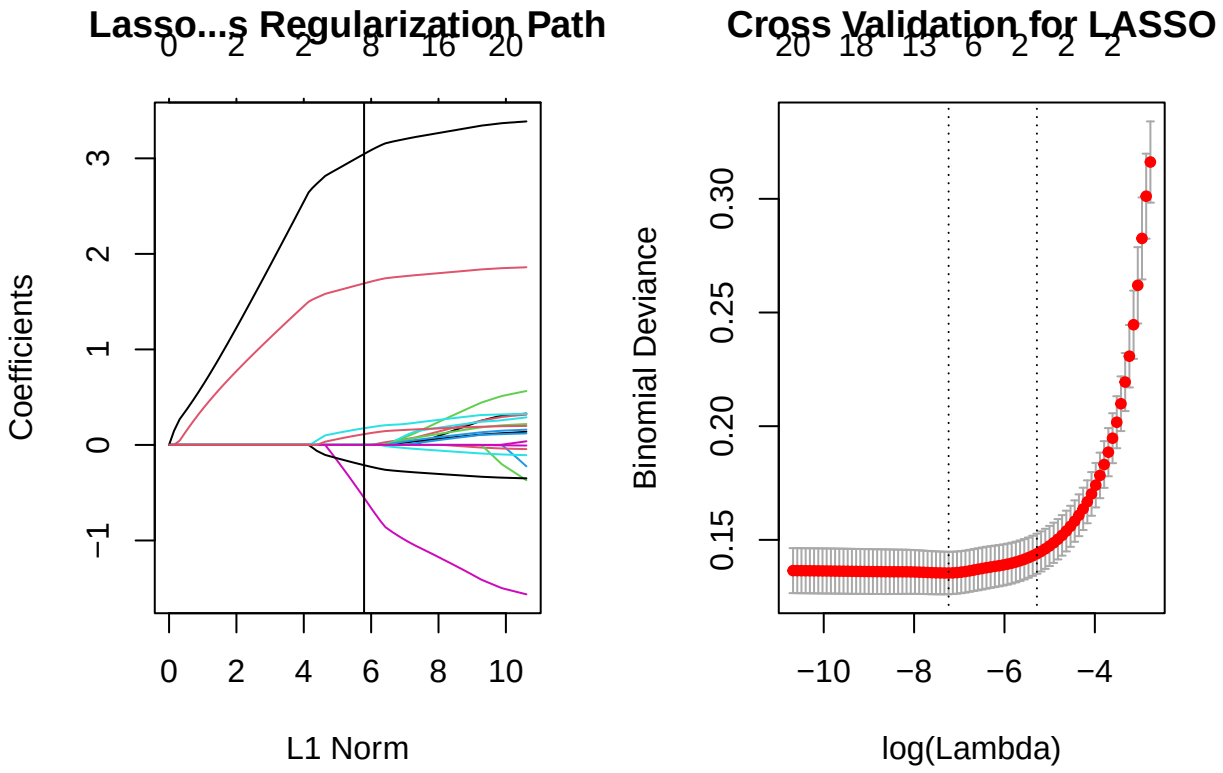
According to Google, the importance of features in LASSO and Ridge Regression are indicated by the absolute value of corresponding coefficient. The larger the absolute value of coefficient, the more important the feature is.

In LASSO model, there are 3 features have coefficients such that the absolute value of coefficient is greater than 1, ordering them in the decreasing order of importance (absolute value of coefficients), they are: Ad_Width, Ad_Height ,Domain="trqRTuT-GNTYJNKbuKz".

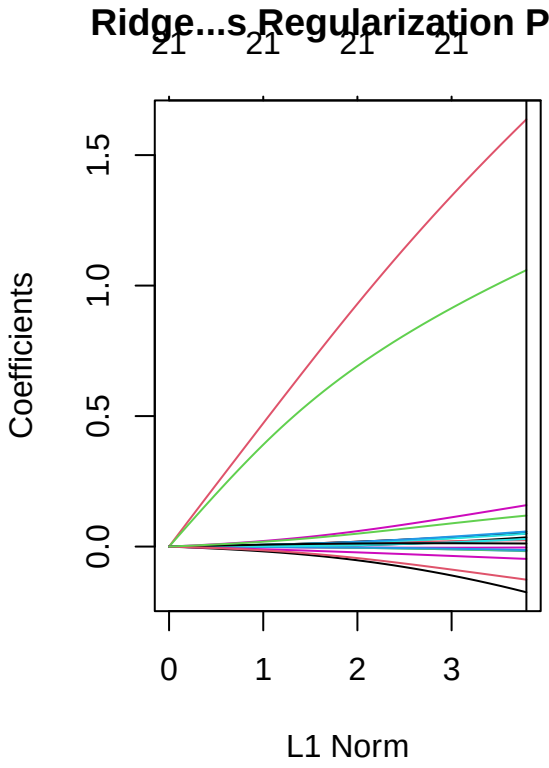
In Ridge model, there are 2 features have coefficients such that the absolute value of coefficient is greater than 1, ordering them in the decreasing order of importance (absolute value of coefficients), they are: Ad_Width, Ad_Height.

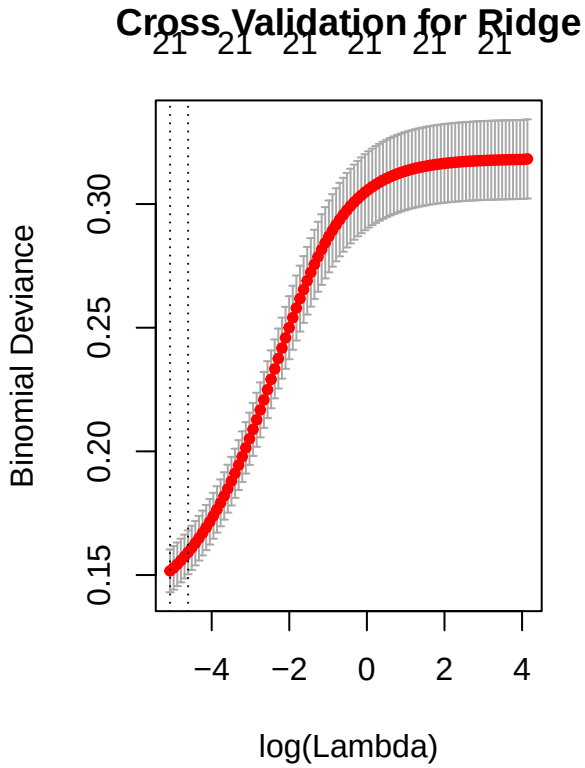
Part c

```
# Fit LASSO and Ridge 5 folds model
cv.lasso = cv.glmnet(dataTrain,dataTrainAll$Click,family="binomial",standardize=FALSE,alpha=1,nfolds = 5

cv.ridge = cv.glmnet(dataTrain,dataTrainAll$Click,family="binomial",standardize=FALSE,alpha=0,nfolds = 5

par(mfrow=c(1,2))
plot(fit.lasso,main="Lasso's Regularization Path",xlab="L1 Norm",ylab="Coefficients")
abline(v=sum(abs(coef(cv.lasso,s="lambda.min")[2:22])))
plot(cv.lasso,main="Cross Validation for LASSO",xlab="log(Lambda)",ylab="Binomial Deviance")
```




```
par(mfrow=c(1,2))
plot(fit.ridge,main="Ridge's Regularization Path",xlab="L1 Norm",ylab="Coefficients")
abline(v=sum(abs(coef(cv.ridge,s="lambda.min")[2:22])))
plot(cv.ridge,main="Cross Validation for Ridge",xlab="log(Lambda)",ylab="Binomial Deviance")
```





According to Part b and regularization path graph, 3 features play a more important role in LASSO model when predicting whether there is at least 1 click. The cross validation plot of LASSO Regression shows that lambda that gives minimum cvm is around log(lambda)=-8, and the model has around 16 non-zero features with lambda that gives minimum cvm.

According to Part b and regularization path graph, 2 features play a more important role in Ridge model when predicting whether there is at least 1 click. The cross validation plot of Ridge Regression shows that lambda that gives minimum cvm is around log(lambda)=-5, and the model has around 21 non-zero features with lambda that gives minimum cvm.

The reason why models with larger degrees of freedom do not necessarily do better in the cross validation is because larger degrees of freedom can cause overfitting, and models that overfit have higher cvm (mean cross-validated error) in cross validation, and cross validation wants to find a lambda such that cvm is minimized.

Part d

```
# Modify dataTest in the way model wants
# Create dummy variables
dataTest = dummy_cols(dataTest, select_columns = c("Region","City","AdX","Domain","Key_Page","Ad_Vis","A
remove_first_dummy = TRUE, ignore_na = TRUE, remove_selected_columns = FALSE
)

# Column bind dummy variables
dataTest$Region = cbind(dataTest$Region_3,dataTest$Region_6)
dataTest$City = cbind(dataTest$City_2,dataTest$City_3,dataTest$City_4,dataTrainAll$City_5,dataTest$City_
dataTest$AdX = cbind(dataTest$AdX_2,dataTest$AdX_3)
dataTest$Domain = cbind(dataTest$Domain_5KFUl5p0Gxsvgmd4wspENpn,dataTest$Domain_trqRTu5Jg9q9wMKYvmpENpn
dataTest$Key_Page = cbind(dataTest$Key_Page_9f4e2f16b6873a7eb504df6f61b24044,dataTest$Key_Page_df6f61b24
dataTest$Ad_Vis = cbind(dataTest$Ad_Vis_1,dataTest$Ad_Vis_2)
dataTest$Ad_Form = dataTest$Ad_Form_1
```

```
# Standardized 3 columns
dataTest$Ad_Width = (dataTest$Ad_Width - mean(dataTest$Ad_Width))/sd(dataTest$Ad_Width)
dataTest$Ad_Height = (dataTest$Ad_Height - mean(dataTest$Ad_Height))/sd(dataTest$Ad_Height)
dataTest$Floor_Price = (dataTest$Floor_Price - mean(dataTest$Floor_Price))/sd(dataTest$Floor_Price)

# Put needed features in a dataframe
dataTest_modified = cbind(dataTest$Region,dataTest$City,dataTest$AdX,dataTest$Domain,dataTest$Ad_Width,

# Modify Click column
dataTestRes$Click[dataTestRes$Click >= 1] = 1
```

```
# Number of rows in dataTestRes
n = dim(dataTestRes)
n = n[1]
# Test for LASSO model
pred_lasso = predict(cv.lasso,dataTest_modified,s="lambda.min")
predAndTruth.lasso = cbind(pred_lasso,dataTestRes$Click) # bind prediction and truth
# True yi = 1, but prediction is 0
nrow(predAndTruth.lasso[dataTestRes$Click==1&pred_lasso<0,])/n
```

```
## [1] 0.0235
```

```
# True yi = 0, but prediction is 1
nrow(predAndTruth.lasso[dataTestRes$Click==0&pred_lasso>0,])/n
```

```
## [1] 0.0023
```

```
# Test for Ridge model
pred_ridge = predict(cv.ridge,dataTest_modified,s="lambda.min")
predAndTruth.ridge = cbind(pred_ridge,dataTestRes$Click) # bind prediction and truth
# True yi = 1, but prediction is 0
nrow(predAndTruth.ridge[dataTestRes$Click==1&pred_ridge<0,])/n
```

```
## [1] 0.0238
```

```
# True yi = 0, but prediction is 1
nrow(predAndTruth.ridge[dataTestRes$Click==0&pred_ridge>0,])/n
```

```
## [1] 0.002
```

Q2

```
load("q1.RData")

# Standardize 3 columns
dataTrainAll$AdX = (dataTrainAll$AdX - mean(dataTrainAll$AdX))/sd(dataTrainAll$AdX)
dataTrainAll$iPinYou_Bid = (dataTrainAll$iPinYou_Bid - mean(dataTrainAll$iPinYou_Bid))/sd(dataTrainAll$
dataTrainAll$Comp_Bid = (dataTrainAll$Comp_Bid - mean(dataTrainAll$Comp_Bid))/sd(dataTrainAll$Comp_Bid)
dataTrain2 = cbind(dataTrainAll$AdX,dataTrainAll$iPinYou_Bid)
```

```
# Fit linear model
fit.linear = lm(Comp_Bid~AdX+iPinYou_Bid,data=dataTrainAll)
fit.linear$coefficients[2:3] # MLE coefficients of linear model
```

```
##         AdX iPinYou_Bid
##  -0.1664490   0.7721763
```

```
# Fit LASSO model
fit.lasso = glmnet(dataTrain2,dataTrainAll$Comp_Bid,family="gaussian",standardize = FALSE,alpha=1)
```

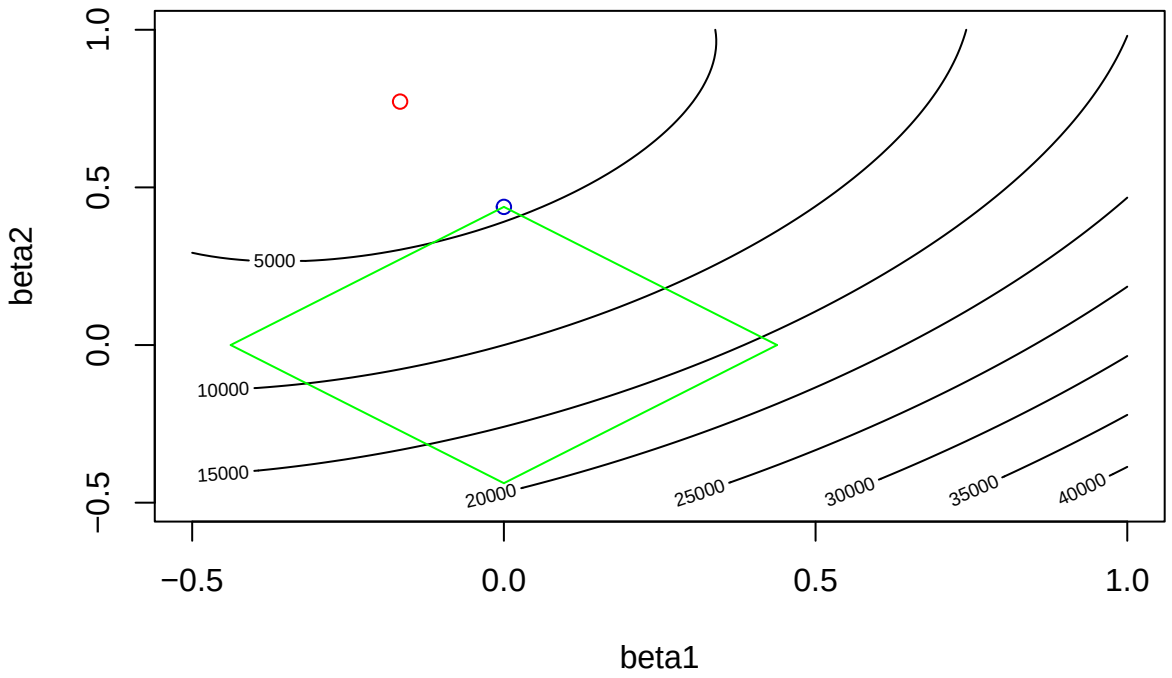
```r
half_L1norm = 0.5*sum(abs(fit.linear$coefficients[2:3]))
half_L1norm # 0.5*L1 norm of MLE coefficients of linear model
```

```
## [1] 0.4693127
```

```r
beta_leq_half_L1norm = fit.lasso$beta[,colSums(abs(fit.lasso$beta))<=half_L1norm] # All beta that have
lasso.coef = beta_leq_half_L1norm[,dim(beta_leq_half_L1norm)[2]] # Beta that have max L1 norm among bet
lasso.coef
```

```
##        V1        V2
## 0.0000000 0.4382291
```

```r
beta1 = seq(-.5,1,length.out=100)
beta2 = seq(-.5,1,length.out=100)
beta = expand.grid(beta1, beta2) # Expand a grid of all combinations of beta1 and beta2

# Function to calculate MSE of a single combination of beta1 and beta2
calc_MSE = function(beta1,beta2){
  return(sum((dataTrainAll$Comp_Bid-beta1*dataTrainAll$AdX-beta2*dataTrainAll$iPinYou_Bid)^2))
}

MSE = mapply(calc_MSE,beta[,1],beta[,2]) # Use mapply to apply calc_MSE to all combinations
MSE = matrix(MSE,100,100)
L1_lasso = sum(abs(lasso.coef))
contour(beta1,beta2,MSE,xlab="beta1",ylab="beta2") # Contour
points(fit.linear$coefficients[2],fit.linear$coefficients[3],col="red") # MLE coefficients
points(lasso.coef[1],lasso.coef[2],col="blue3") # LASSO coefficients
plot(function(x){x+L1_lasso},-L1_lasso,0,add=T,col="green")
plot(function(x){-x-L1_lasso},-L1_lasso,0,add=T,col="green")
plot(function(x){x-L1_lasso},0,L1_lasso,add=T,col="green")
plot(function(x){-x+L1_lasso},0,L1_lasso,add=T,col="green")
```




```r
# Ridge Regression part
library(wordspace)
```

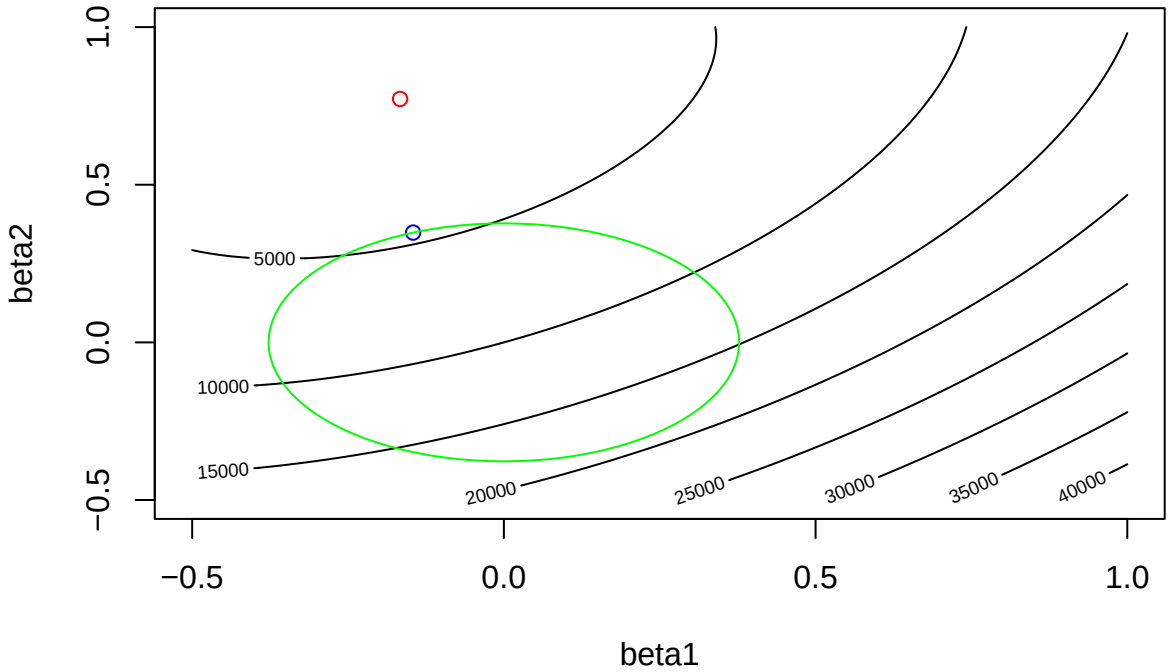
```
library(DescTools)
fit.ridge = glmnet(dataTrain2,dataTrainAll$Comp_Bid,family="gaussian",standardize=FALSE,alpha=0)
half_L2norm = 0.5*norm(fit.linear$coefficients[2:3], type = "2")
half_L2norm # 0.5*L2 norm of MLE coefficients of linear model
```

```
## [1] 0.3949562
```

```
beta_leq_half_L2norm = fit.ridge$beta[,colNorms(fit.ridge$beta)<=half_L2norm] # All beta that have L2 n
ridge.coef = beta_leq_half_L2norm[,dim(beta_leq_half_L2norm)[2]] # Beta that have max L2 norm among bet
ridge.coef
```

```
##         V1         V2
## -0.1455805  0.3482623
```

```
L2_ridge = norm(ridge.coef, type = "2")
contour(beta1,beta2,MSE,xlab="beta1",ylab="beta2") # Contour
points(fit.linear$coefficients[2],fit.linear$coefficients[3],col="red") # MLE coefficients
points(ridge.coef[1],ridge.coef[2],col="blue3") # Ridge coefficients
DrawEllipse(0,0, radius.x = L2_ridge, radius.y = L2_ridge,border="green",col =NA)
```



The MLE coefficients sit at the center of the level curve. MLE coefficients minimize MSE. LASSO and Ridge coefficients sit close to the most inner contour curve (5000), It is the point on L1 or L2 ball that has minimum MSE. LASSO and Ridge try to minimize MSE while coefficients satisfy L1/L2 norm. LASSO encourages sparse because L1 ball is diamond shape, so the L1 diamond have greater chance to have minimum MSE at 4 corner point, and at corner point, there is one coefficient equal to 0 (in 2d case). A simple example is, when you rotate a circle for 360 degree, during the rotation, any point on circle can be a maximum point at some angle with equal occurrence; but if you rotate a diamond, 4 corner point have further higher frequency to become maximum points than others.

Question 5 (20 Points) Discover Gravitational Waves in Your Room

Answer:

```
set.seed(10)
LIGO = read.table("LIGO.Hanford.Data.txt", header=F)
head(LIGO)
```

```
##          V1           V2
## 1 0.2500000  0.026422536
## 2 0.2500610 -0.003132572
## 3 0.2501221 -0.130760718
## 4 0.2501831 -0.061991781
## 5 0.2502441  0.019565419
## 6 0.2503052  0.022075875
```

Q1

```
n_row = nrow(LIGO)
sqrt_n_row = sqrt(n_row)
C = matrix(, nrow = n_row, ncol = n_row)
for (j in 1:n_row){
  for (k in 1:n_row){
    if (j==1){
      C[j,k] = sqrt_n_row
    }
    else {
      C[j,k] = sqrt(2/n_row)*cos(pi*(2*k-1)*(j-1)/(2*n_row))
    }
  }
}

cv.lasso = cv.glmnet(C,as.matrix(LIGO[2]),alpha=1,nfolds = 10)
#w_hat = cv.lasso$beta[,cv.lasso$lambda.min]
#cv.lasso$beta
y_pred = predict(cv.lasso,C,s="lambda.min")
#plot(cv.lasso$beta,)
cv.lasso$beta
```

Q2

Question 6 (25 Points) Upright Human Detection in Photos

Answer:

(a)

```
library(png)
source("functions.R")
subdir_pos = "/pngdata/pos/"
subdir_neg = "/pngdata/neg/"
filename_pos = "5"
filename_neg = "7"
photo_pos = readPNG(paste(file.path(getwd(),subdir_pos,filename_pos), ".png", sep = ""))
photo_neg = readPNG(paste(file.path(getwd(),subdir_neg,filename_neg), ".png", sep = ""))
# Transfer to the black and white version
photo_pos = rgb2gray(photo_pos)
writePNG(photo_pos, target = "bwpos.png")
photo_neg = rgb2gray(photo_neg)
writePNG(photo_neg, target = "bwneg.png")
```

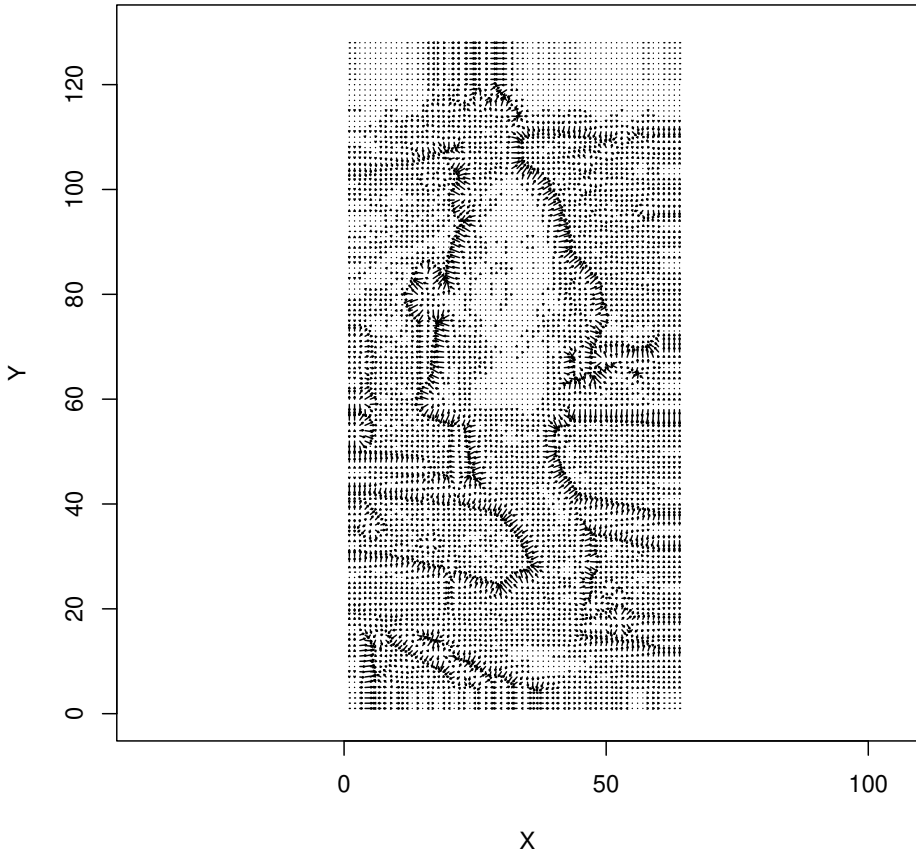
```
photo_neg = crop.r(photo_neg,160,96) # Randomly crop
writePNG(photo_neg, target = "cropneg.png")

photo_pos.grad = grad(photo_pos, 128, 64, F)
setEPS()
postscript("gradpos.eps")
g=grad(photo_pos, 128, 64, T)
dev.off()
```

```
## pdf
##   2
```

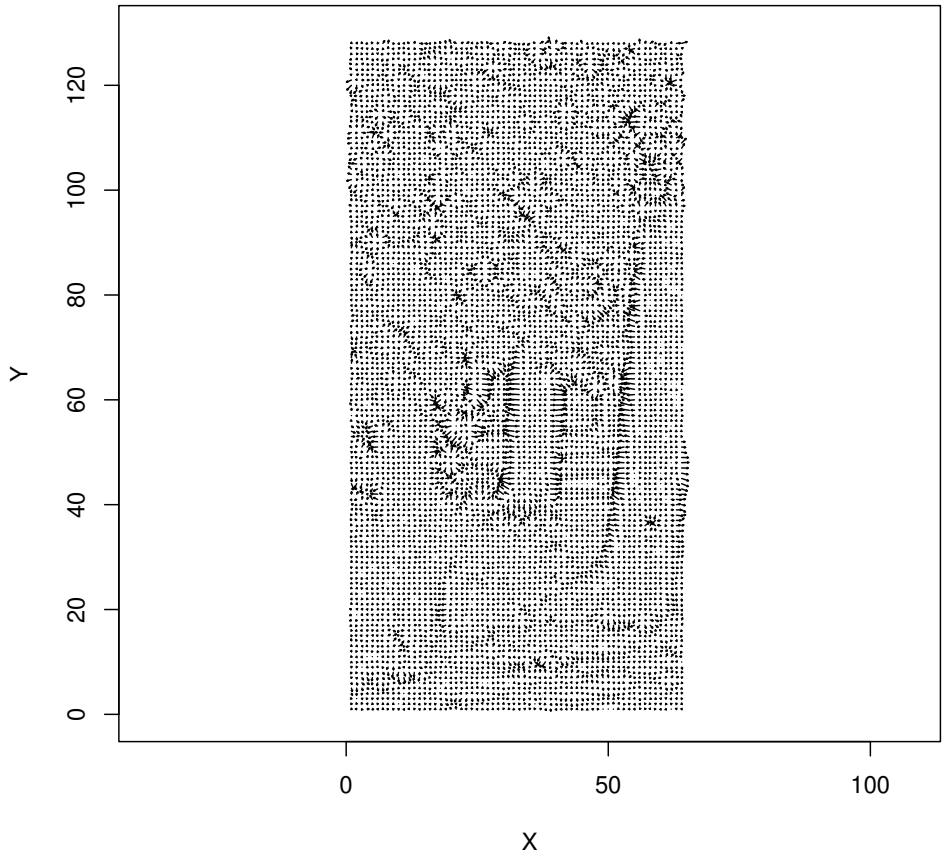
```
photo_neg.grad = grad(photo_neg, 128, 64, F)
setEPS()
postscript("gradneg.eps")
g=grad(photo_neg, 128, 64, T)
dev.off()
```

```
## pdf
##   2
```

```
photo_pos.hog = hog(photo_pos.grad$xgrad, photo_pos.grad$ygrad, 4, 4, 6)
photo_neg.hog = hog(photo_neg.grad$xgrad, photo_neg.grad$ygrad, 4, 4, 6)
```




```
##  [1] 0.23828125 0.45312500 0.10156250 0.02148438 0.07031250 0.05273438
##  [7] 0.10546875 0.17773438 0.13281250 0.12109375 0.27148438 0.18945312
## [13] 0.13281250 0.25585938 0.11328125 0.13085938 0.25585938 0.11132812
## [19] 0.13085938 0.20703125 0.14062500 0.09179688 0.32617188 0.08789062
## [25] 0.25781250 0.12500000 0.21289062 0.12109375 0.13867188 0.14453125
## [31] 0.16406250 0.10937500 0.15429688 0.16406250 0.11328125 0.29296875
## [37] 0.22656250 0.26367188 0.14453125 0.10546875 0.11328125 0.14453125
## [43] 0.23632812 0.21093750 0.14453125 0.11132812 0.19531250 0.10156250
## [49] 0.20117188 0.32226562 0.13085938 0.10546875 0.11132812 0.06445312
## [55] 0.07421875 0.10937500 0.33203125 0.29296875 0.10156250 0.08984375
## [61] 0.13085938 0.19531250 0.22851562 0.20117188 0.16796875 0.07617188
## [67] 0.22070312 0.22656250 0.19726562 0.10937500 0.16601562 0.08007812
## [73] 0.18164062 0.40820312 0.11718750 0.03906250 0.15625000 0.03515625
## [79] 0.05273438 0.21484375 0.06640625 0.11523438 0.41992188 0.13085938
## [85] 0.05078125 0.40234375 0.05664062 0.04492188 0.36718750 0.07812500
## [91] 0.13281250 0.17968750 0.08593750 0.08007812 0.41992188 0.07031250
```




```
##   [1] 0.18359375 0.15625000 0.16406250 0.19335938 0.16406250 0.13867188
##   [7] 0.21289062 0.16992188 0.12695312 0.18750000 0.16796875 0.13476562
```

```
## [13] 0.15234375 0.10937500 0.23828125 0.21875000 0.11914062 0.16210938
## [19] 0.15234375 0.19335938 0.24414062 0.16796875 0.10742188 0.13476562
## [25] 0.18164062 0.16015625 0.16406250 0.18554688 0.16406250 0.14453125
## [31] 0.18750000 0.17187500 0.15625000 0.18359375 0.15234375 0.14843750
## [37] 0.11523438 0.16015625 0.16210938 0.22070312 0.17578125 0.16601562
## [43] 0.16406250 0.22851562 0.22656250 0.14257812 0.12695312 0.11132812
## [49] 0.14843750 0.14257812 0.17773438 0.20898438 0.19335938 0.12890625
## [55] 0.12304688 0.17578125 0.20507812 0.27148438 0.11718750 0.10742188
## [61] 0.07812500 0.20117188 0.23046875 0.17578125 0.21093750 0.10351562
## [67] 0.11914062 0.24804688 0.18359375 0.16406250 0.17578125 0.10937500
## [73] 0.15820312 0.16406250 0.16015625 0.16601562 0.17773438 0.17382812
## [79] 0.26757812 0.12109375 0.15820312 0.08593750 0.10546875 0.26171875
## [85] 0.16210938 0.09570312 0.25390625 0.18164062 0.08593750 0.22070312
## [91] 0.11718750 0.24609375 0.27929688 0.13085938 0.11718750 0.10937500
```

(b)

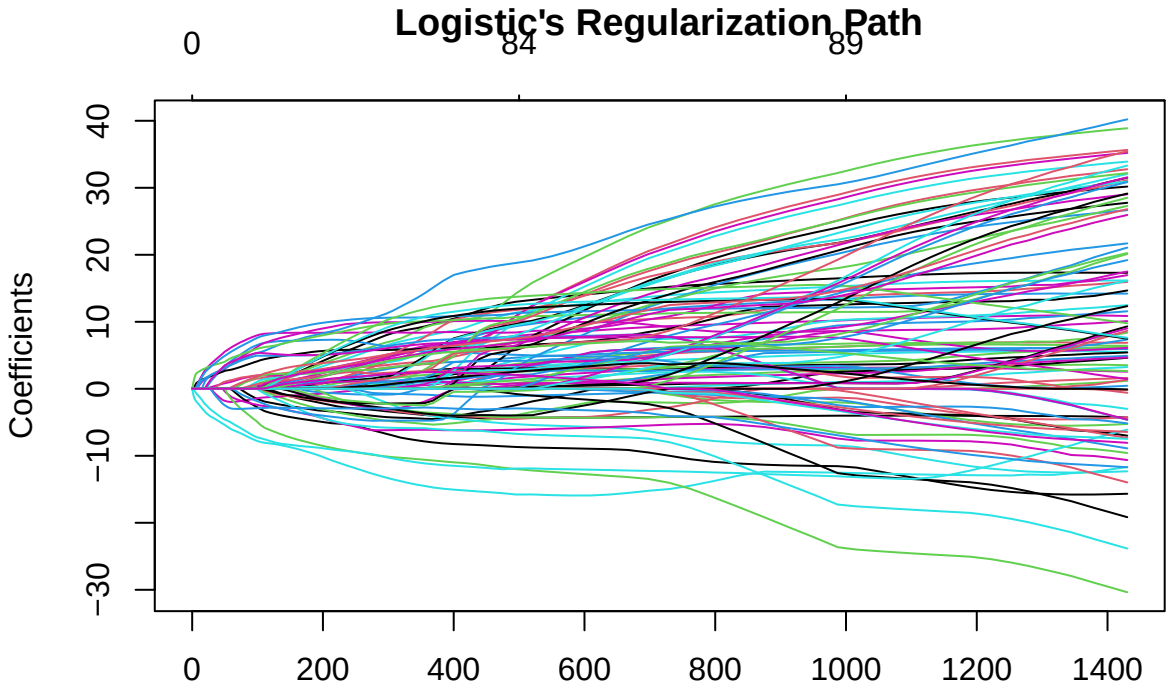
```
getwd()
```

```
## [1] "/Users/yulunwu/Downloads/2023winter/STAD80/A3"
```

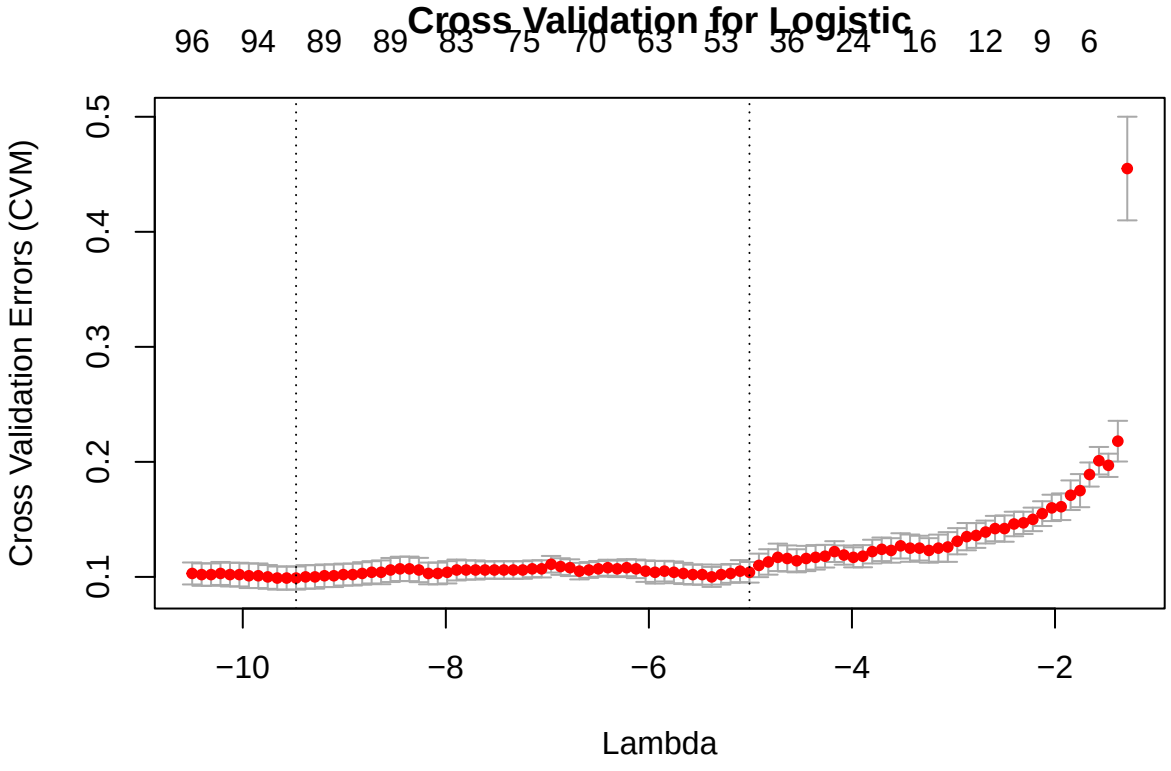
```
n=500
features_pos = data.frame(matrix(ncol = 96, nrow = 0))
features_neg = data.frame(matrix(ncol = 96, nrow = 0))
for (i in 1:500){
  photo_pos = readPNG(paste(file.path(getwd(),subdir_pos,as.character(i)), ".png", sep = ""))
  photo_neg = readPNG(paste(file.path(getwd(),subdir_neg,as.character(i)), ".png", sep = ""))
  # Transfer to the black and white version
  photo_pos = rgb2gray(photo_pos)
  photo_neg = rgb2gray(photo_neg)

  photo_neg = crop.r(photo_neg,160,96) # Randomly crop

  photo_pos.grad = grad(photo_pos, 128, 64, F)
  photo_neg.grad = grad(photo_neg, 128, 64, F)

  photo_pos.hog = hog(photo_pos.grad$xgrad, photo_pos.grad$ygrad, 4, 4, 6)
  photo_neg.hog = hog(photo_neg.grad$xgrad, photo_neg.grad$ygrad, 4, 4, 6)

  features_pos[i,] = photo_pos.hog
  features_neg[i,] = photo_neg.hog
}
features_data = rbind(features_pos,features_neg) # Row bind two dataframe together, row 1 to 500 are PO
features_data[1:500,"POS"] = 1 # 1 indicates POS
features_data[501:1000,"POS"] = 0 # 0 indicates NEG
```

Part II

```
fit.logit = glmnet(features_data[,1:96],features_data$POS,family="binomial")
plot(fit.logit,main="Logistic's Regularization Path",xlab="",ylab="Coefficients")
```




```
fit.logit.cv = cv.glmnet(as.matrix(features_data[,1:96]),features_data$POS,family="binomial",type.measu
plot(fit.logit.cv,main="Cross Validation for Logistic",xlab="Lambda",ylab="Cross Validation Errors (CVM)
```



Question 7 (30 Points) Sentiment Analysis on Amazon Product Reviews

Answer:

(a)

```
library(plyr)
load("Amazon_SML.RData")
colnames(dat) # Column names
```

```
## [1] "name"   "review" "rating"
```

```
nrow(dat) # Number of reviews
```

```
## [1] 1312
```

```
length(unique(dat$name)) # Number of unique products
```

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

```
# Product has the most '5' ratings
prod_r5 = dat[dat$rating==5,] # Product has '5' ratings
count_r5 = count(prod_r5$name) # Count of 5 ratings with respect to names
count_r5
```

```
##                                           x freq
## 1 NTM-910YIC - Sony Baby Call Nursery Monitor  105
## 2       Safety 1st Deluxe 4-in-1 Bath Station   25
## 3            Vulli Sophie the Giraffe Teether  526
```

```
# Product has the most '1' ratings
prod_r1 = dat[dat$rating==1,] # Product has '1' ratings
count_r1 = count(prod_r1$name) # Count of 1 ratings with respect to names
count_r1[count_r1$freq==max(count_r1$freq),]
```

```
##                                                                        x freq
## 5 Infant Optics DXR-5 2.4 GHz Digital Video Baby Monitor with Night Vision   68
```

Column names are name,review and rating. There are 1312 reviews. There are 20 unique products. Vulli Sophie the Giraffe Teether has the most '5' ratings which is 526. Infant Optics DXR-5 2.4 GHz Digital Video Baby Monitor with Night Vision has the most '5' ratings which is 68.

(b)

```
unique(dat$rating) # List of unique rating values
```

```
## [1] 1 5
```

```
nrow_r1 = sum(count_r1$freq) # Number of reviews for rating=1
nrow_r1
```

```
## [1] 656
```

```
nrow_r5 = sum(count_r5$freq) # Number of reviews for rating=5
nrow_r5
```

```
## [1] 656
```

```
source("tdMat.R")
```

```
## Loading required package: NLP
```

There are 656 reviews of each rating value in the entire dataset. The possible rating values are 1 and 5. The best performance of a "constant classifier" is 0.5, because we have half of data with rating=1, and half with rating=5, so a "constant classifier" will get 50% of calssification correct.

(c)

```
source("splitData.R")
set.seed(10)
lambda<-exp(seq(-20, -1, length.out = 99))
cvfit<-cv.glmnet(train.x,train.y,family="binomial",type.measure="class",lambda=lambda)
fit.logit = glmnet(train.x,train.y,family="binomial",lambda = cvfit$lambda.1se)
fit.logit$df # The number of nonzero coefficients
```

```
## [1] 353
```

```
# Twenty words with most positive coefficients
beta_de = fit.logit$beta[order(fit.logit$beta,decreasing = T),] # Sort beta in decreasing order
names(beta_de[1:20]) # 20 words with most positive coefficients
```

```
##  [1] "wimper"       "round"        "endur"        "abov"         "scrape"
##  [6] "whichev"      "love"         "lol"          "neighborhood" "precious"
## [11] "laundri"      "fyi"          "teeth"        "poster"       "grandma"
## [16] "channel"      "sum"          "bet"          "describ"      "result"
```

```
# Twenty words with most negative coefficients
beta_in = fit.logit$beta[order(fit.logit$beta),] # Sort beta in increasing order
names(beta_in[1:20]) # 20 words with most negative coefficients
```

```
##  [1] "swallow"    "downstair"  "tummi"      "solv"       "dissapoint"
##  [6] "unlink"     "avoid"      "philip"     "bin"        "wast"
## [11] "useless"    "click"      "knock"      "sad"        "massiv"
## [16] "scissor"    "cool"       "speaker"    "return"     "ball"
```

The number of covariates that have non-zero coefficients in the model selected by lambda.1se are 353.

(d)

```
# This function will compute number of row in train.x such that column rname has value>0
count_doc=function(rname){
  return(nrow(train.x[train.x[,rname]>0,]))
}

count_pos = mapply(count_doc,names(beta_de[1:20]))
count_pos # Covariates name and frequency, these covariates are 20 words with most positive coefficient
```

```
## $wimper
## [1] 2
##
## $round
## [1] 4
##
## $endur
## [1] 2
##
## $abov
## [1] 7
##
## $scrape
## NULL
##
## $whichev
## [1] 3
##
```

```
## $love
## [1] 427
##
## $lol
## [1] 5
##
## $neighborhood
## [1] 3
##
## $precious
## [1] 2
##
## $laundri
## [1] 3
##
## $fyi
## NULL
##
## $teeth
## [1] 179
##
## $poster
## [1] 2
##
## $grandma
## [1] 2
##
## $channel
## [1] 32
##
## $sum
## [1] 2
##
## $bet
## [1] 2
##
## $describ
## [1] 5
##
## $result
## [1] 6
```

```
count_neg = mapply(count_doc,names(beta_in[1:20]))
count_neg # Covariates name and frequency, these covariates are 20 words with most negative coefficient
```

```
## $swallow
## NULL
##
## $downstair
## [1] 4
##
## $tummi
## [1] 2
##
## $solv
```

```
## [1] 3
##
## $dissapoint
## [1] 3
##
## $unlink
## [1] 2
##
## $avoid
## [1] 4
##
## $philip
## [1] 2
##
## $bin
## [1] 2
##
## $wast
## [1] 86
##
## $useless
## [1] 27
##
## $click
## [1] 9
##
## $knock
## [1] 12
##
## $sad
## [1] 9
##
## $massiv
## [1] 2
##
## $scissor
## [1] 2
##
## $cool
## [1] 4
##
## $speaker
## [1] 4
##
## $return
## [1] 97
##
## $ball
## [1] 2
```

```
# Number of documents using "love" had a rating of 1 or 5
length(train.y[train.y==1&train.x[,"love"]>0]) # Number of documents using "love" had a rating of 5
```

```
## [1] 364
```

```
length(train.y[train.y==0&train.x[,"love"]>0]) # Number of documents using "love" had a rating of 1
```

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

```
using_love = train.tag[train.x[,"love"]>0] # Tag of documents used in training set that has "love"
dat[using_love[1],"review"] # The first document that used this most positive word "love"
```

```
## [1] It\\'s easy to hold by little fingers n gives sound when she presses it. She loves it very much a
## 182643 Levels:  ...
```

```
# Number of documents using "wast" had a rating of 1 or 5
length(train.y[train.y==1&train.x[,"wast"]>0]) # Number of documents using "wast" had a rating of 5
```

```
## [1] 6
```

```
length(train.y[train.y==0&train.x[,"wast"]>0]) # Number of documents using "wast" had a rating of 1
```

```
## [1] 80
```

```
using_wast = train.tag[train.x[,"wast"]>0] # Tag of documents used in training set that has "wast"
dat[using_wast[1],"review"] # The first document that used this most negative word "wast"
```

```
## [1] We have had this monitor for four years now and we are getting ready to purchase another one to u
## 182643 Levels:  ...
```

The word with the most positive weights that appeared in more than 10 documents is "love". The word with the most negative weights that appeared in more than 10 documents is "wast". Among 427 of documents in train.x that using "love", 364 of them have rating of 5, 63 of them have rating of 1. Among 86 of documents in train.x that using "wast", 6 of them have rating of 5, 80 of them have rating of 1.

(e)

```
n_test = nrow(test.x) # Number of rows in testing set
pred_y=predict(fit.logit,test.x) # Predict testing set
# Number of predictions where pred_y don't agree with test.y divided by number of rows in testing set,
length(pred_y[(test.y==1&pred_y<0)|((test.y==0&pred_y>0)),])/n_test
```

```
## [1] 0.07575758
```

The misclassification rate is 0.07575758. This is definitely better than the "constant classifier", because the misclassification rate for "constant classifier" is 0.5.

Question 8 (15 Points) Identifying Risk Factors for the Heart Disease

Answer:

Part I

```
heart = read.csv("framingham.csv",header = T)
n = nrow(heart)
x = heart[,-ncol(heart)] # dataframe with out TenYearCHD
fit.logit = glm(heart$TenYearCHD~as.matrix(x),family = binomial)
summary_fit = summary(fit.logit)
# The statistically significant coefficients
summary_fit$coefficients[summary_fit$coefficients[ ,4] < 0.05,1]
```

```
##            (Intercept)       as.matrix(x)male        as.matrix(x)age
##           -8.328185673            0.555278660            0.063514559
## as.matrix(x)cigsPerDay    as.matrix(x)totChol      as.matrix(x)sysBP
##            0.017913725            0.002332111            0.015403097
```

```
##    as.matrix(x)glucose
##            0.007126942
```

Part II

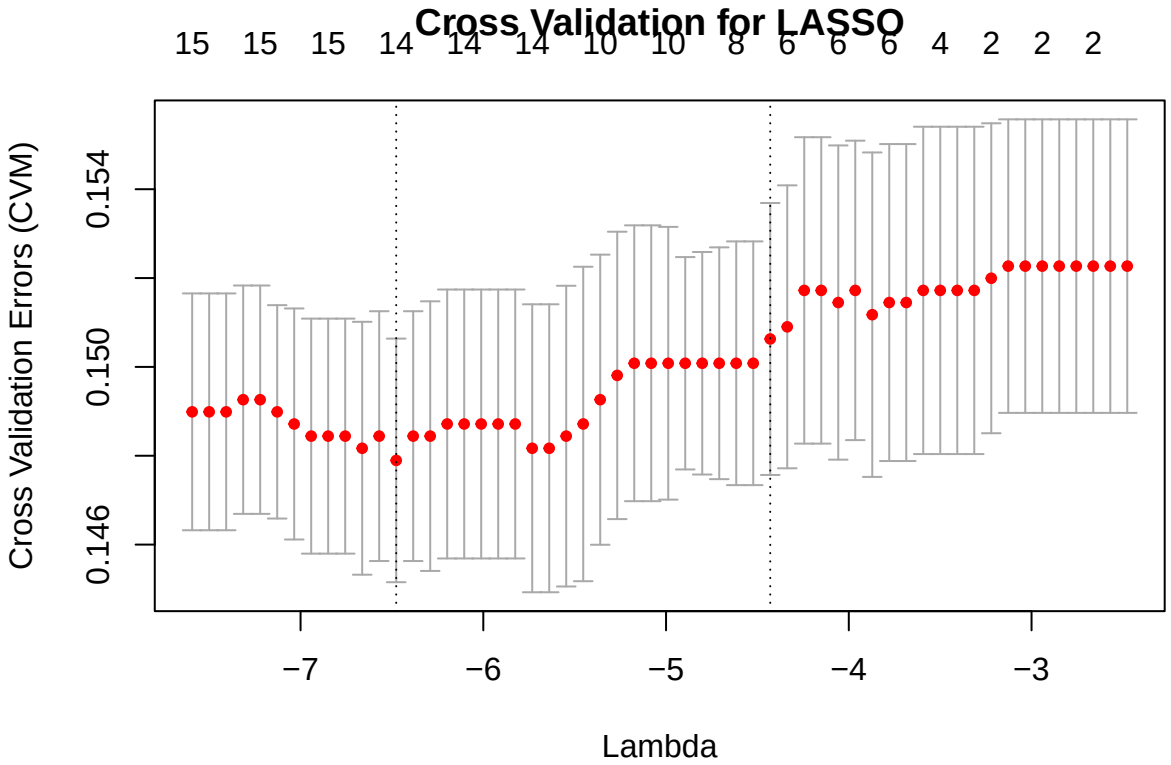
```
set.seed(100)

# First subset
sub_dataset_ind = sample(rownames(heart),n/5,replace=F)
sub_dataset1 = heart[sub_dataset_ind,]
rest = heart[-as.numeric(sub_dataset_ind),]

# Second subset
sub_dataset_ind = sample(rownames(rest),n/5,replace=F)
sub_dataset2 = rest[sub_dataset_ind,]
rest = rest[-as.numeric(sub_dataset_ind),]

# Third subset
sub_dataset_ind = sample(rownames(rest),n/5,replace=F)
sub_dataset3 = rest[sub_dataset_ind,]
rest = rest[-as.numeric(sub_dataset_ind),]

# Fourth subset
sub_dataset_ind = sample(rownames(rest),n/5,replace=F)
sub_dataset4 = rest[sub_dataset_ind,]
rest = rest[-as.numeric(sub_dataset_ind),]

# Fifth subset
sub_dataset_ind = sample(rownames(rest),n/5,replace=F)
sub_dataset5 = rest[sub_dataset_ind,]

train1 = rbind(sub_dataset1,sub_dataset2,sub_dataset3,sub_dataset4)
train1.x = train1[,-ncol(heart)]
fit.logit1 = glm(train1$TenYearCHD~as.matrix(train1.x),family = binomial)
test.x = sub_dataset5[,-ncol(heart)]
pred_y_test = exp(as.matrix(test.x)%*%fit.logit1$coefficients[2:16]+fit.logit1$coefficients[1])/(1+exp(a
length(pred_y_test[(sub_dataset5$TenYearCHD==1&pred_y_test<0.5)|((sub_dataset5$TenYearCHD==0&pred_y_tes
```

```
## [1] 0.2488208
```

Here I find some NA in dataframe, I didn't remove NAs because handout didn't ask so. The prediction error of logistic regression (misclassification rate) is 0.2488208.

Part III

```
heart = na.omit(heart)
x = heart[,-ncol(heart)] # dataframe with out TenYearCHD
fit.lasso = glmnet(x,heart$TenYearCHD,family="binomial",alpha=1)
x = model.matrix(TenYearCHD~., heart)[,-1]
fit.lasso.cv = cv.glmnet(x,heart$TenYearCHD,family="binomial",type.measure="class",nfolds = 5)

plot(fit.lasso.cv,main="Cross Validation for LASSO",xlab="Lambda",ylab="Cross Validation Errors (CVM)")
```



The curve is not a typical U-curve. I don't think we need regularization here, because CVM doesn't change much with different λ.