STOR 565 Fall 2019 Homework 3

Ted Henson

*Remark.* This homework aims to help you further understand the model selection techniques in linear model. Credits for **Theoretical Part** and **Computational Part** are in total 100 pt. For **Computational Part** , please complete your answer in the **RMarkdown** file and summit your printed PDF homework created by it.

## Computational Part

**Hint.** Before starting your work, carefully read Textbook Chapter 6.5-6.7 (Lab 1-3). Mimic the related analyses you learn from it. Also look at the demonstrations I showed you in class (see Sakai/Resources/Lectures and click on each Lecture for demonstrations). Some related packages have been loaded in setup.

1. (Model Selection, Textbook 6.8, *18 pt*) In this exercise, we will generate simulated data, and will then use this data to perform model selection.
2. Use the rnorm function to generate a predictor of length , as well as a noise vector of length . Do not print the entire vector.

set.seed(336)  
  
x = matrix(rnorm(100),ncol=1)  
  
  
e = matrix(rnorm(100),ncol=1)

**Hint.** Before generating random numbers, fix your favourite random seed by set.seed so that your result is reproducible as you carry forward your exploration.

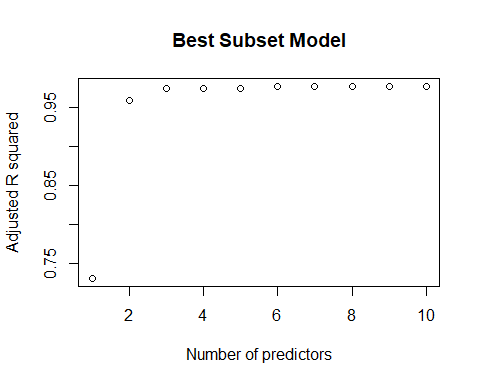
1. Generate a response vector of length according to the model
2. where , , , . Do not print the entire vector.

beta = function(x){3 + 2\*x - 3\*(x^2) + .3\*(x^3)}  
  
y = beta(x) + e

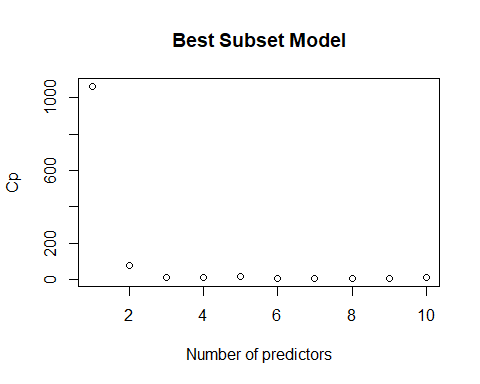
1. Use the regsubsets function from leaps package to perform best subset selection in order to choose the best model containing the predictors .

What is the best model obtained according to , BIC, and adjusted ? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained.

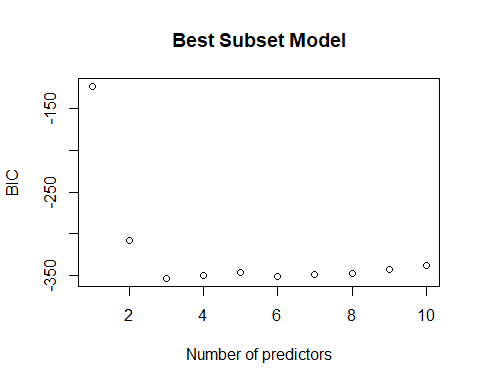
data = matrix(nrow = nrow(x),  
 ncol = 10)  
data[,1] = x[,1]  
data[,2] = data[,1]^2  
data[,3] = data[,1]^3  
data[,4] = data[,1]^4  
data[,5] = data[,1]^5  
data[,6] = data[,1]^6  
data[,7] = data[,1]^7  
data[,8] = data[,1]^8  
data[,9] = data[,1]^9  
data[,10] = data[,1]^10  
  
colnames(data) = c('x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9', 'x10')  
  
reduced = regsubsets(y = y,  
 x = data,  
 nvmax = 10)  
results = summary(reduced)  
  
plot(y = results$adjr2[1:10], x = c(1:10),  
 main = 'Best Subset Model',  
 ylab = 'Adjusted R squared',   
 xlab = 'Number of predictors')



plot(y = results$cp[1:10], x = c(1:10),  
 main = 'Best Subset Model',  
 ylab = 'Cp', xlab = 'Number of predictors')



plot(y = results$bic[1:10], x = c(1:10),  
 main = 'Best Subset Model',  
 ylab = 'BIC', xlab = 'Number of predictors')



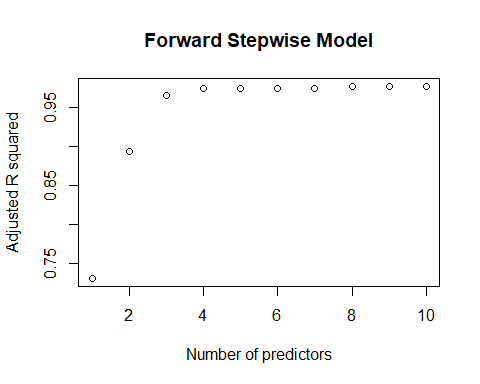
The model with the lowest cp has 8 predictors. The model with the highest adjusted R squared has 8 predictors. The model with the lowest bic has 3 predictors. Overall the best model is probably the model with 3 predictors, as the changes in cp, BIC, and adjr2 are marginal with additional predictors. The coeffecients for our chosen model are:

coef(reduced, 3)

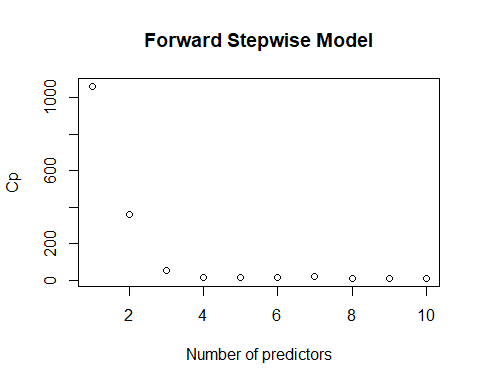
## (Intercept) x1 x2 x3   
## 2.995934 1.973766 -2.978722 0.340765

1. Repeat (c), using forward stepwise selection and also using backwards stepwise selection. How does your answer compare to the results in (c)? You must show a summary of the selected model or other evidence to support your statements.

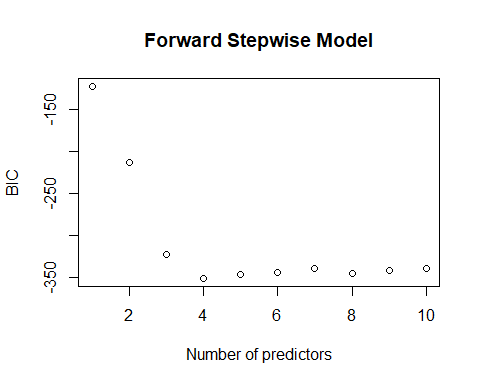
forward = regsubsets(y = y,  
 x = data,  
 nvmax = 10,  
 method = 'forward')  
  
backward = regsubsets(y = y,  
 x = data,  
 nvmax = 10,  
 method = 'backward')  
  
results = summary(forward)  
  
  
  
plot(y = results$adjr2[1:10], x = c(1:10),  
 main = 'Forward Stepwise Model',  
 ylab = 'Adjusted R squared', xlab = 'Number of predictors')



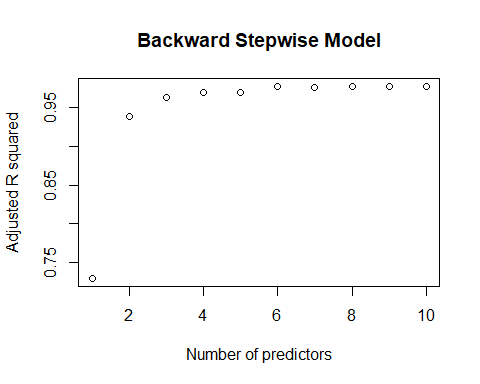
plot(y = results$cp[1:10], x = c(1:10),  
 main = 'Forward Stepwise Model',  
 ylab = 'Cp', xlab = 'Number of predictors')



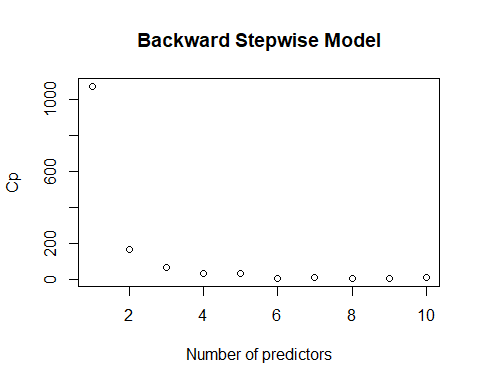
plot(y = results$bic[1:10], x = c(1:10),  
 main = 'Forward Stepwise Model',  
 ylab = 'BIC', xlab = 'Number of predictors')



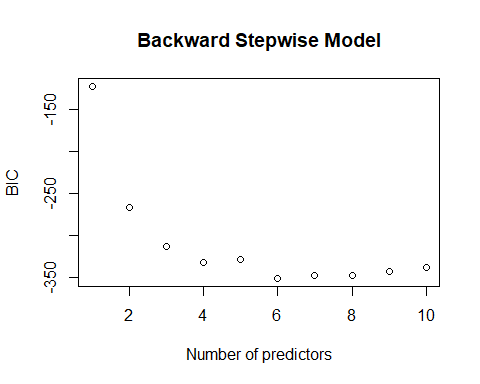
results = summary(backward)  
  
plot(y = results$adjr2[1:10], x = c(1:10),  
 main = 'Backward Stepwise Model',  
 ylab = 'Adjusted R squared', xlab = 'Number of predictors')



plot(y = results$cp[1:10], x = c(1:10),  
 main = 'Backward Stepwise Model',  
 ylab = 'Cp', xlab = 'Number of predictors')



plot(y = results$bic[1:10], x = c(1:10),  
 main = 'Backward Stepwise Model',  
 ylab = 'BIC', xlab = 'Number of predictors')



Using forward stepwise selection, the best model has 4 predictors as the changes in CP and adjr2 begin to level out at 3 predictors, and the BIC levels out (and increases) after adding the fifth predictor. Using backward stepwise selection, the best model has 6 predictors as the BIC begins to increase after adding a 7th predictor, and the CP and adjr2 have leveled out. The coeffecients for our forward model are:

print(coef(forward, 4))

## (Intercept) x1 x2 x3 x4   
## 3.0949108 1.8468069 -3.1748960 0.3987835 0.0388525

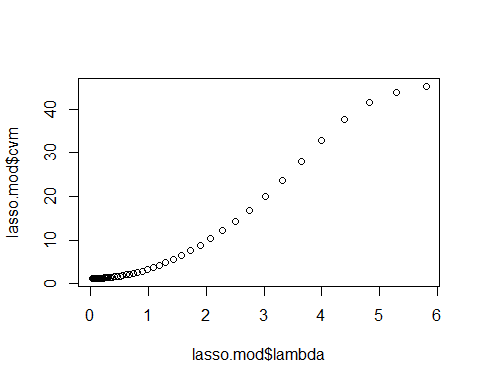
The coeffecients for our backward model are:

print(coef(backward, 6))

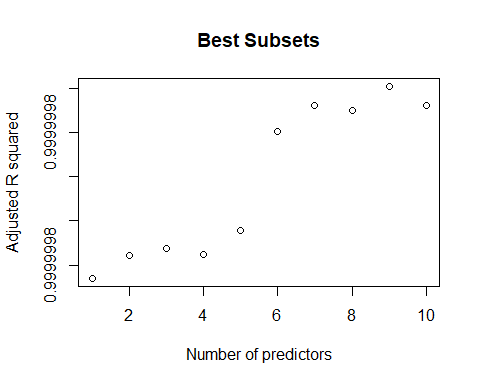
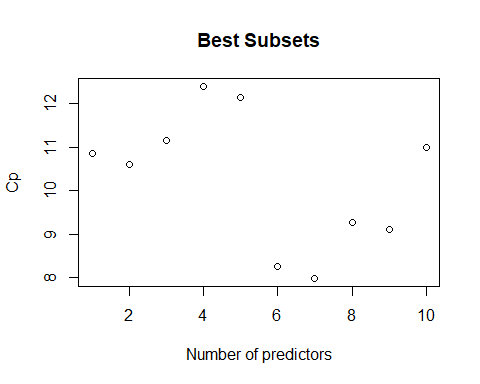
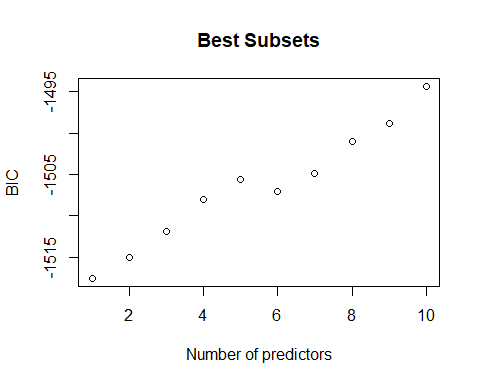
## (Intercept) x2 x3 x5 x6 x7   
## 2.90572768 -2.48126339 3.56272035 -1.37025326 -0.19200768 0.16660286   
## x8   
## 0.03773801

All three models contain X2 and X3 as predictor variables. Best Subsets and Forward Stepwise also both contain X1 as a predictor.

1. Now fit a LASSO model with glmnet function from glmnet package to the simulated data, again using as predictors. Use 5-fold cross-validation to select the optimal value of . Create plots of the cross-validation error as a function of . Report the resulting coefficient estimates, and discuss the results obtained.

* lasso.mod = cv.glmnet(y = y, x = data, alpha = 1,  
   nfolds = 5)  
  plot(x = lasso.mod$lambda, y = lasso.mod$cvm)
* 
* print('Lasso Regression Coefficients:')
* ## [1] "Lasso Regression Coefficients:"
* coef(lasso.mod)
* ## 11 x 1 sparse Matrix of class "dgCMatrix"  
  ## 1  
  ## (Intercept) 2.8100565  
  ## x1 1.7397857  
  ## x2 -2.8267160  
  ## x3 0.3490796  
  ## x4 .   
  ## x5 .   
  ## x6 .   
  ## x7 .   
  ## x8 .   
  ## x9 .   
  ## x10 .
* The lasso model found that the only nonzero coefficients were X1, X2, and X3. These were the only coefficients used to generate our data so the lasso model will be better at predicting future data generated from our function than the previous models trained.

1. Now generate a response vector according to the model
2. where , and perform best subset selection and the LASSO. Discuss the results obtained.

* beta = function(x){0 + 7\*x^7}  
    
  y = beta(x) + e  
    
  reduced = regsubsets(y = y,  
   x = data,  
   nvmax = 10)  
  subsets.results = summary(reduced)  
    
  lasso.mod = cv.glmnet(y = y, x = data, alpha = 1,  
   nfolds = 5)  
    
  print('Lasso Regression Coefficients:')
* ## [1] "Lasso Regression Coefficients:"
* coef(lasso.mod)
* ## 11 x 1 sparse Matrix of class "dgCMatrix"  
  ## 1  
  ## (Intercept) -9.727377  
  ## x1 .   
  ## x2 .   
  ## x3 .   
  ## x4 .   
  ## x5 .   
  ## x6 .   
  ## x7 6.730511  
  ## x8 .   
  ## x9 .   
  ## x10 .
* plot(y = subsets.results$adjr2[1:10], x = c(1:10),  
   main = 'Best Subsets',  
   ylab = 'Adjusted R squared', xlab = 'Number of predictors')
* 
* plot(y = subsets.results$cp[1:10], x = c(1:10),  
   main = 'Best Subsets',  
   ylab = 'Cp', xlab = 'Number of predictors')
* 
* plot(y = subsets.results$bic[1:10], x = c(1:10),  
   main = 'Best Subsets',  
   ylab = 'BIC', xlab = 'Number of predictors')
* 
* print('Subset Regression Coefficients:')
* ## [1] "Subset Regression Coefficients:"
* coef(reduced, 6)
* ## (Intercept) x1 x3 x4 x5   
  ## 0.117721254 -1.212712618 2.010430891 -0.091432326 -0.832500783   
  ## x7 x10   
  ## 7.099656961 0.001212406

Depending on your criterion, the best subset model would chose 6 or 7 predictors despite our function only being generated from one variable (x7). Our lasso regression was able to shrink all other variables to 0, so we conclude that the lasso regression model is better.

1. (Prediction, *20 pt*) In this exercise, we will try to develop a prediction model for wins in a basketball season for a team based on a host of other factors. The starting point is to load the nba-teams-2017 data set (which was scraped by Gaston Sanchez at Berkeley).
2. Do some exploratory data analysis by picking 6-7 features that you think might be interesting and explore relationship between these features by making a scatterplot matrix like the following (you **do not** have to use the same features I am using!):

library(readr, quietly = T)  
nba\_teams\_2017 <- read\_csv("nba-teams-2017.csv")

## Parsed with column specification:  
## cols(  
## .default = col\_double(),  
## team = col\_character()  
## )

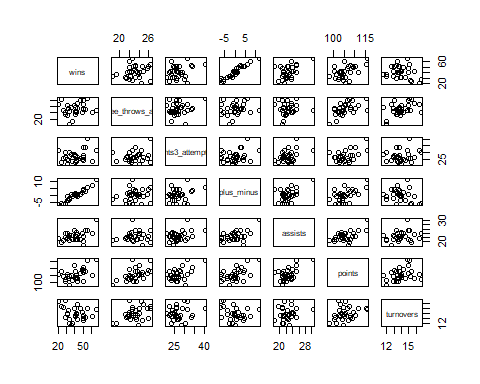
## See spec(...) for full column specifications.

library(caret, quietly = T)

##   
## Attaching package: 'caret'

## The following object is masked from 'package:pls':  
##   
## R2

pairs(wins ~ free\_throws\_att + points3\_attempted + plus\_minus + assists + points + turnovers, data = nba\_teams\_2017)



*NOTE: You may remove the includegraphics statements below when knitting your own response, if they are giving you trouble*

1. The aim is now to predict *wins* based on the other features. First explain why you would remove the “losses” column from the above data set? Would you necessarily remove any other columns?

You would remove the losses column because it is redudant: losses is a linear combination of the wins and games columns (games minus wins). Game number is also the same for every observation so we should remove that as well. Winprop will have the same variance as wins so we should remove that to avoid auto correlation. Since we only have one observation per team, we should also remove that column.

1. Use ridge regression with 5 fold cross-validation to choose the optimal tuning parameter and report your model along with your test error as found by cross-validation for that choice of .

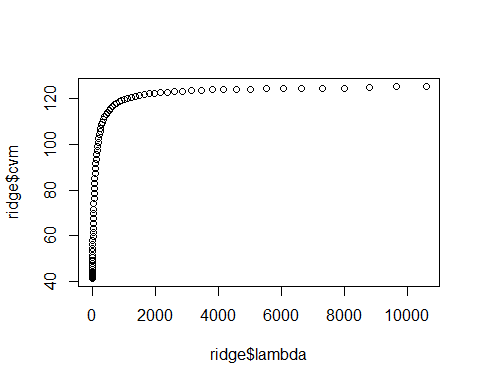
library(tidyverse)

## -- Attaching packages -------------------------------------------- tidyverse 1.2.1 --

## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 0.8.3 v stringr 1.4.0  
## v purrr 0.3.2 v forcats 0.4.0

## -- Conflicts ----------------------------------------------- tidyverse\_conflicts() --  
## x purrr::accumulate() masks foreach::accumulate()  
## x tidyr::expand() masks Matrix::expand()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x purrr::lift() masks caret::lift()  
## x purrr::when() masks foreach::when()

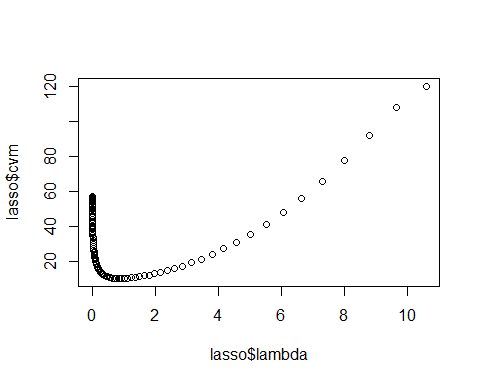
nba\_teams\_2017.reduced = nba\_teams\_2017 %>% dplyr::select(-c(team,  
 games\_played,  
 losses,  
 wins,  
 win\_prop))  
  
ridge = cv.glmnet(x = as.matrix(nba\_teams\_2017.reduced),  
 y = nba\_teams\_2017$wins,  
 alpha = 0, nfolds = 5)  
plot(x = ridge$lambda, y = ridge$cvm)



The best test error was 41.3723017 wins from the optimal lambda of 3.5533312.

1. Fit a LASSO model on the training set, with chosen by 5-fold cross-validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

lasso = cv.glmnet(x = as.matrix(nba\_teams\_2017.reduced),  
 y = nba\_teams\_2017$wins,  
 alpha = 1, nfolds = 5)  
 plot(x = lasso$lambda, y = lasso$cvm)



print('Coefficients:')

## [1] "Coefficients:"

coef(lasso)

## 23 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 41.000000  
## minutes .   
## points .   
## field\_goals .   
## field\_goals\_attempted .   
## field\_goals\_prop .   
## points3 .   
## points3\_attempted .   
## points3\_prop .   
## free\_throws .   
## free\_throws\_att .   
## free\_throws\_prop .   
## off\_rebounds .   
## def\_rebounds .   
## rebounds .   
## assists .   
## turnovers .   
## steals .   
## blocks .   
## block\_fga .   
## personal\_fouls .   
## personal\_fouls\_drawn .   
## plus\_minus 2.111394

The only non-zero coefficient was plus\_minus. The best test error was 10.4874123 wins wins from the optimal lambda of 0.8599517.

1. Let us now try to understand the performance of the various techniques on simulated data to get insight.

Suppose your true model is

where:

* Uncorrelated predictors:
  + . Precisely for the -th individual, where are independent and identically distributed normal random variables with mean zero and variance one.
* . Precisely: where are independent and identically distributed normal random variables with mean zero and variance 1.

(a)(2 pt) Generate the above data with seed = 1234

set.seed(1234)  
beta = c(rep(1,20), rep(0,1980))  
  
  
x = matrix(data = rnorm(mean = 0,   
 n = 1000\*2000,  
 sd = 1),   
 nrow = 1000,  
 ncol = 2000)  
  
  
e = rnorm(n = 1000, mean = 0, sd = 1)  
  
  
y = x %\*% beta + e

1. (6 pt) Using glmnet fit Lasso, ridge regression and elastic net with

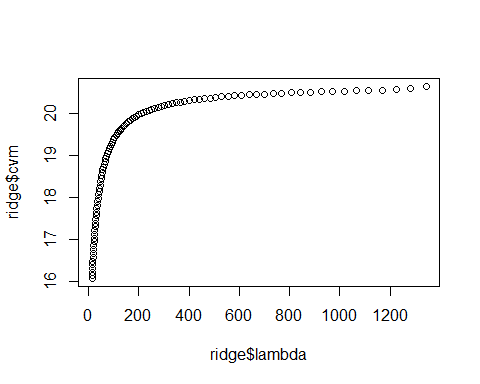
What I am looking for: (outputting the entire model for each one of the above is not-trivial so):

* code showing the fitting of each of the above models

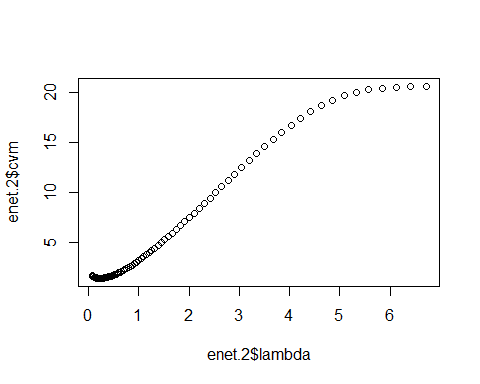
alpha = c(0, .2, .3, .4, .5, .6, .7, .8, .9, 1)  
  
  
ridge = cv.glmnet(x = x, y = y,  
 alpha = alpha[1],  
 nfolds = 6  
 )  
  
enet.2 = cv.glmnet(x = x, y = y,  
 alpha = alpha[2],  
 nfolds = 6  
 )  
  
enet.3 = cv.glmnet(x = x, y = y,  
 alpha = alpha[3],  
 nfolds = 6)  
  
enet.4 = cv.glmnet(x = x, y = y,  
 alpha = alpha[4],  
 nfolds = 6)  
  
  
enet.5 = cv.glmnet(x = x, y = y,  
 alpha = alpha[5],  
 nfolds = 6)  
  
enet.6 = cv.glmnet(x = x, y = y,  
 alpha = alpha[6],  
 nfolds = 6)  
  
enet.7 = cv.glmnet(x = x, y = y,  
 alpha = alpha[7],  
 nfolds = 6)  
  
enet.8 = cv.glmnet(x = x, y = y,  
 alpha = alpha[8],  
 nfolds = 6)  
  
enet.9 = cv.glmnet(x = x, y = y,  
 alpha = alpha[9],  
 nfolds = 6)  
  
lasso = cv.glmnet(x = x, y = y,  
 alpha = alpha[10],  
 nfolds = 6)

* For ridge, Lasso and for plot the cross-validated (6 fold) MSE versus lambda as well as your optimal value of for ridge, Lasso and for

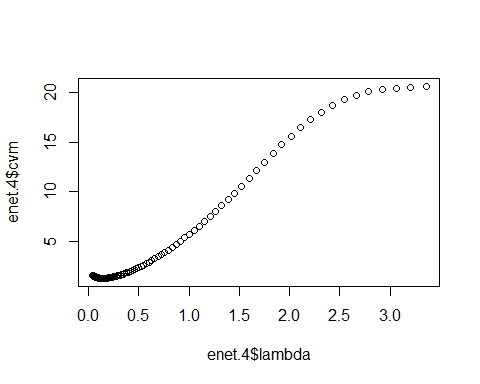
plot(x = ridge$lambda, y = ridge$cvm)



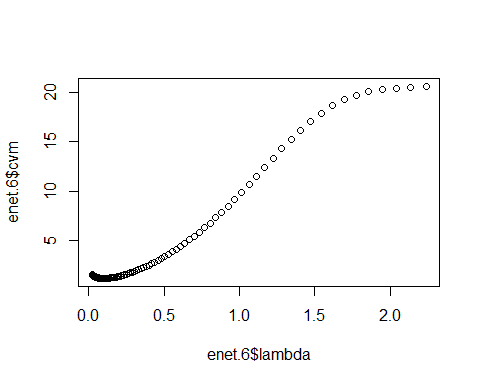
plot(x = enet.2$lambda, y = enet.2$cvm)



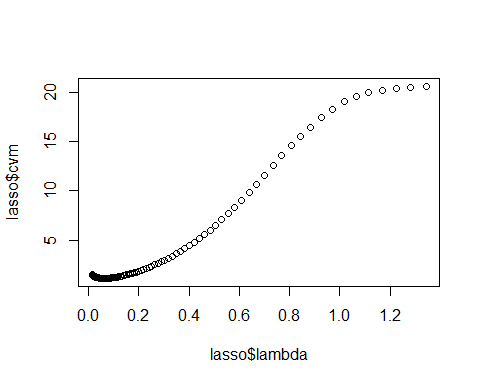
plot(x = enet.4$lambda, y = enet.4$cvm)



plot(x = enet.6$lambda, y = enet.6$cvm)



plot(x = lasso$lambda, y = lasso$cvm)



Ridge optimal lambda value: 13.4447287

Enet .2 optimal lambda value: 0.2253069

Enet .4 optimal lambda value: 0.1421526

Enet .6 optimal lambda value: 0.0947684

Lasso optimal lambda value: 0.0595685

* The number of non-zero regression coeffecients for each of the above techniques.

ridge.coeff = nrow(summary(coef(ridge, s= 'lambda.min')))  
enet.2.coeff = nrow(summary(coef(enet.2, s= 'lambda.min')))  
enet.4.coeff = nrow(summary(coef(enet.4, s= 'lambda.min')))  
enet.6.coeff = nrow(summary(coef(enet.6, s= 'lambda.min')))  
lasso.coeff = nrow(summary(coef(lasso, s= 'lambda.min')))

Number of nonzero coefficients in ridge model: 2001

Number of nonzero coefficients in enet.2 model: 227

Number of nonzero coefficients in enet.4 model: 134

Number of nonzero coefficients in enet.6 model: 128

Number of nonzero coefficients in lasso model: 119

1. (2 pt) Simulate an independent {} data set of the same type as above (response and features per subject) with . Use seed = 4567.

set.seed(4567)  
beta = c(rep(1,20), rep(0,1980))  
  
  
x = matrix(data = rnorm(mean = 0,   
 n = 10000\*2000,  
 sd = 1),   
 nrow = 10000,  
 ncol = 2000)  
  
  
e = rnorm(n = 10000, mean = 0, sd = 1)  
  
  
y = x %\*% beta + e

1. (2 pt) Using the models you obtained above using the training data set and the 11 models above, compute average test error for each of the 11 models. Which one is the “best” model?

ridge.pred = predict(ridge, x)  
enet.2.pred = predict(enet.2, x)  
enet.4.pred = predict(enet.4, x)  
enet.6.pred = predict(enet.6, x)  
lasso.pred = predict(lasso, x)  
  
ridge.rmse = mean(sqrt((ridge.pred - y)^2))  
enet.2.rmse = mean(sqrt((enet.2.pred - y)^2))  
enet.4.rmse = mean(sqrt((enet.4.pred - y)^2))  
enet.6.rmse = mean(sqrt((enet.6.pred - y)^2))  
lasso.rmse = mean(sqrt((lasso.pred - y)^2))

RMSEs by increasing alpha:

3.2358333

0.9468024

0.9099804

0.8998722

0.8612684

The lasso had the smallest mean test error.

1. Do all the 4 parts of problem 3 but where the underlying model is:

(a)(2 pt) Generate the above data with seed = 8910

set.seed(8910)  
beta = c(rep(1,1000), rep(0,1000))  
  
  
x = matrix(data = rnorm(mean = 0,   
 n = 10000\*2000,  
 sd = 1),   
 nrow = 10000,  
 ncol = 2000)  
  
  
e = rnorm(n = 10000, mean = 0, sd = 1)  
  
  
y = x %\*% beta + e

1. (6 pt) Using glmnet fit Lasso, ridge regression and elastic net with

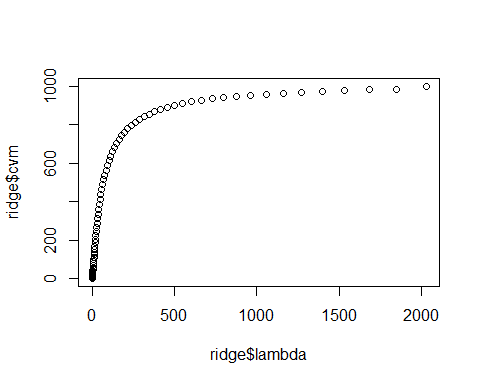
What I am looking for: (outputting the entire model for each one of the above is not-trivial so):

* code showing the fitting of each of the above models

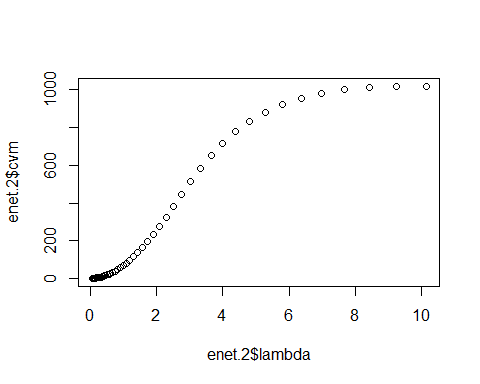
alpha = c(0, .2, .3, .4, .5, .6, .7, .8, .9, 1)  
  
  
ridge = cv.glmnet(x = x, y = y,  
 alpha = alpha[1],  
 nfolds = 6  
 )  
  
enet.2 = cv.glmnet(x = x, y = y,  
 alpha = alpha[2],  
 nfolds = 6  
 )  
  
enet.3 = cv.glmnet(x = x, y = y,  
 alpha = alpha[3],  
 nfolds = 6)  
  
enet.4 = cv.glmnet(x = x, y = y,  
 alpha = alpha[4],  
 nfolds = 6)  
  
  
enet.5 = cv.glmnet(x = x, y = y,  
 alpha = alpha[5],  
 nfolds = 6)  
  
enet.6 = cv.glmnet(x = x, y = y,  
 alpha = alpha[6],  
 nfolds = 6)  
  
enet.7 = cv.glmnet(x = x, y = y,  
 alpha = alpha[7],  
 nfolds = 6)  
  
enet.8 = cv.glmnet(x = x, y = y,  
 alpha = alpha[8],  
 nfolds = 6)  
  
enet.9 = cv.glmnet(x = x, y = y,  
 alpha = alpha[9],  
 nfolds = 6)  
  
lasso = cv.glmnet(x = x, y = y,  
 alpha = alpha[10],  
 nfolds = 6)

* For ridge, Lasso and for plot the cross-validated (6 fold) MSE versus lambda as well as your optimal value of for ridge, Lasso and for

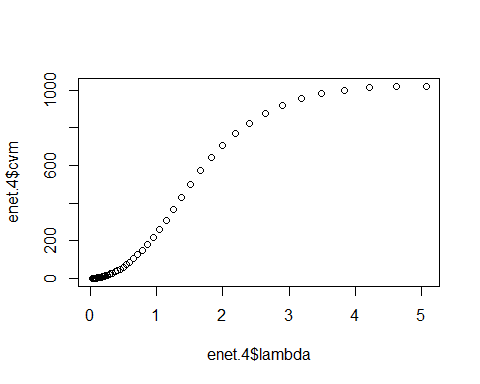
plot(x = ridge$lambda, y = ridge$cvm)



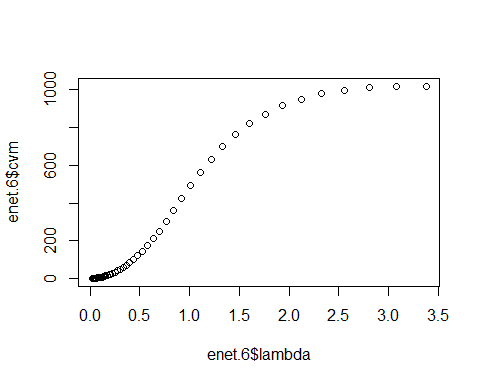
plot(x = enet.2$lambda, y = enet.2$cvm)



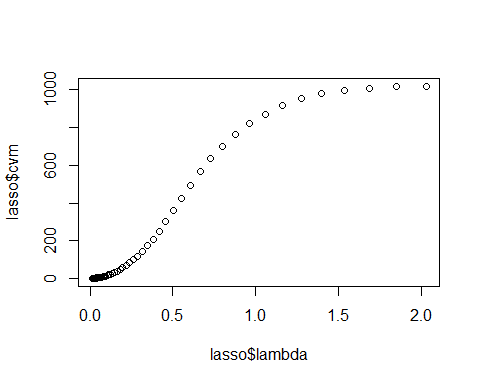
plot(x = enet.4$lambda, y = enet.4$cvm)



plot(x = enet.6$lambda, y = enet.6$cvm)



plot(x = lasso$lambda, y = lasso$cvm)



Ridge optimal lambda value: 0.4269841

Enet .2 optimal lambda value: 0.0554017

Enet .4 optimal lambda value: 0.0277009

Enet .6 optimal lambda value: 0.0184672

Lasso optimal lambda value: 0.0121607

* The number of non-zero regression coeffecients for each of the above techniques.

Number of nonzero coefficients in ridge model: 2001

Number of nonzero coefficients in enet.2 model: 227

Number of nonzero coefficients in enet.4 model: 134

Number of nonzero coefficients in enet.6 model: 128

Number of nonzero coefficients in lasso model: 119

1. (2 pt) Simulate an independent {} data set of the same type as above (response and features per subject) with . Use seed = 1112.

set.seed(1112)  
beta = c(rep(1,1000), rep(0,1000))  
  
  
x = matrix(data = rnorm(mean = 0,   
 n = 10000\*2000,  
 sd = 1),   
 nrow = 10000,  
 ncol = 2000)  
  
  
e = rnorm(n = 10000, mean = 0, sd = 1)  
  
  
y = x %\*% beta + e

1. (2 pt) Using the models you obtained above using the training data set and the 11 models above, compute average test error for each of the 11 models. Which one is the “best” model?

ridge.pred = predict(ridge, x)  
enet.2.pred = predict(enet.2, x)  
enet.4.pred = predict(enet.4, x)  
enet.6.pred = predict(enet.6, x)  
lasso.pred = predict(lasso, x)  
  
ridge.rmse = mean(sqrt((ridge.pred - y)^2))  
enet.2.rmse = mean(sqrt((enet.2.pred - y)^2))  
enet.4.rmse = mean(sqrt((enet.4.pred - y)^2))  
enet.6.rmse = mean(sqrt((enet.6.pred - y)^2))  
lasso.rmse = mean(sqrt((lasso.pred - y)^2))

RMSEs in order of increasing alpha:

1.0235113

0.934744

0.9359549

0.9320257

0.9288984

The lasso model is the best model.