STOR 565 Spring 2020 Homework 3

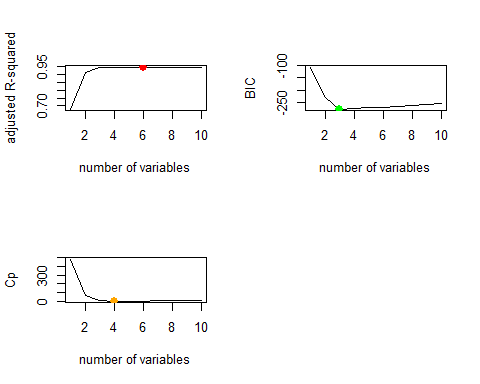
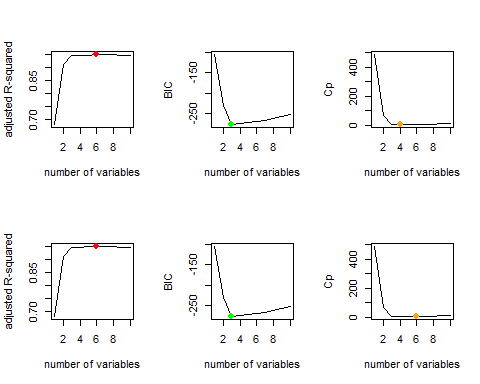
Yimei Fan

*Remark.* This homework aims to help you further understand the model selection techniques in linear models. For the **Computational Part**, please complete your answers in the **RMarkdown** file and print your generated PDF file created by it.

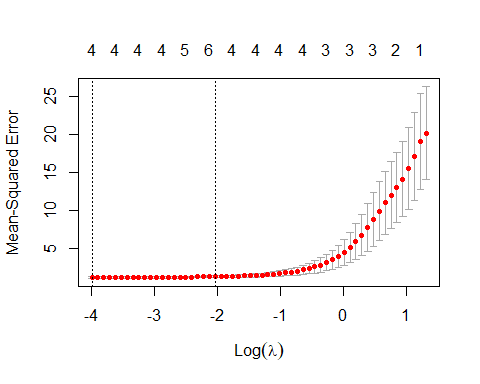
## Computational Part

**Hint.** Before starting your work, carefully read Textbook Chapter 6.5-6.7 (Labs 1-3). Mimic the related analyses you learn from it. Related packages have been loaded in the setup.

1. (Model Selection and Best Subset Prediction, *25 pt*) In this exercise, we will generate simulated data, and will then use the data to perform model selection.
   1. Use the rnorm function to generate a predictor of length , as well as a noise vector of length .

* set.seed(1)  
  x = rnorm(200)  
  e = rnorm(200)
  1. Generate a response vector of length according to the model
  2. where , , , . Split the data set into a size- training set and a size- test set. (You can just let the first 100 samples be the training set)
* b0 = 4;b1 = 3;b2 = -2;b3 = 0.5  
  y = b0 + b1\*x + b2\*x^2 + b3\*x^3 + e  
    
  d = data.frame(cbind(x,y))  
    
  set.seed(1)  
  train = sample(1:nrow(d),nrow(d)/2)  
  test = (-train)  
    
  train\_set = d[train,]  
  test\_set = d[test,]  
    
  nrow(test\_set)
* ## [1] 100
* nrow(train\_set)
* ## [1] 100
  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. Calculate the test errors of these coefficients.
* x.new = x  
  for(i in 2:10){  
   x.new = cbind(x.new,x^i)  
  }  
  colnames(x.new) = paste('x',1:ncol(x.new),sep = ' ')  
    
  dta = data.frame(cbind(y,x.new))  
    
  trainset = dta[train,]  
  testset = dta[test,]  
    
  regfit = regsubsets(y~.,data = trainset, nvmax = 10)  
  regfit.summary = summary(regfit)  
    
  par(mfrow = c(2,2))  
    
  plot(regfit.summary$adjr2,xlab = "number of variables", ylab = 'adjusted R-squared',type = 'l')  
  points(which.max(regfit.summary$adjr2),regfit.summary$adjr2[which.max(regfit.summary$adjr2)],col = 'red',cex = 2,pch = 20)  
  which.max(regfit.summary$adjr2)#6
* ## [1] 6
* plot(regfit.summary$bic,xlab = "number of variables",ylab = 'BIC',type = 'l')  
  points(which.min(regfit.summary$bic),regfit.summary$bic[which.min(regfit.summary$bic)],col = 'green',cex = 2,pch = 20)  
  which.min(regfit.summary$bic)#3
* ## [1] 3
* plot(regfit.summary$cp,xlab = 'number of variables',ylab = 'Cp',type = 'l')  
  points(which.min(regfit.summary$cp),regfit.summary$cp[which.min(regfit.summary$cp)],col = 'orange',cex = 2,pch = 20)  
  which.min(regfit.summary$cp)#4
* ## [1] 4
* #If we define ‘best’ as the most marginal reduction in error for each variable added, then all models indicate by their shape that 3 variables provide the best fit.   
    
    
    
  #then fit model with the full dataset.  
    
  #BIC  
  regfit.full = regsubsets(y~.,data = trainset, nvmax = 3)  
  regfit.full.summary = summary(regfit.full)  
  coef(regfit.full,3)
* ## (Intercept) x.1 x.2 x.3   
  ## 4.0011721 2.7340096 -1.9800634 0.5980388
* #It shows x.1, x2,x3 will ve the best variables.  
  # y = 4.001 + 2.734\*x -1.980\*x^2 + 0.598\*x^3  
    
    
  #calculate test error  
    
  test.mat1 = model.matrix(y~., testset)  
    
  #BIC  
  coefi2 = coef(regfit.full,3)  
   pred2 = test.mat1[,names(coefi2)]%\*%coefi2  
   errors2 = mean((dta$y[test]-pred2)^2)  
   errors2
* ## [1] 0.9400014
* #the test error is 0.9400014.
* 
  1. Repeat (c), using forward stepwise selection and also using backwards stepwise selection. Report the best coefficients and calculate their test errors. How does your answer compare to the results in (c)?
* regfit.fwd = regsubsets(y~.,data = trainset,nvmax = 10,method = 'forward')  
  fwd.summary = summary(regfit.fwd)  
    
  regfit.bwd = regsubsets(y~.,data = trainset,nvmax = 10,method = 'backward')  
  bwd.summary = summary(regfit.bwd)  
    
  par(mfrow = c(2,3))  
    
  #fwd  
  plot(fwd.summary$adjr2,xlab = "number of variables", ylab = 'adjusted R-squared',type = 'l')  
  points(which.max(fwd.summary$adjr2),fwd.summary$adjr2[which.max(fwd.summary$adjr2)],col = 'red',cex = 2,pch = 20)  
  which.max(fwd.summary$adjr2)
* ## [1] 6
* plot(fwd.summary$bic,xlab = "number of variables",ylab = 'BIC',type = 'l')  
  points(which.min(fwd.summary$bic),fwd.summary$bic[which.min(fwd.summary$bic)],col = 'green',cex = 2,pch = 20)  
  which.min(fwd.summary$bic)
* ## [1] 3
* plot(fwd.summary$cp,xlab = 'number of variables',ylab = 'Cp',type = 'l')  
  points(which.min(fwd.summary$cp),fwd.summary$cp[which.min(fwd.summary$cp)],col = 'orange',cex = 2,pch = 20)  
  which.min(fwd.summary$cp)
* ## [1] 4
* #bwd  
  plot(bwd.summary$adjr2,xlab = "number of variables", ylab = 'adjusted R-squared',type = 'l')  
  points(which.max(bwd.summary$adjr2),bwd.summary$adjr2[which.max(bwd.summary$adjr2)],col = 'red',cex = 2,pch = 20)  
  which.max(bwd.summary$adjr2)
* ## [1] 6
* plot(bwd.summary$bic,xlab = "number of variables",ylab = 'BIC',type = 'l')  
  points(which.min(bwd.summary$bic),bwd.summary$bic[which.min(bwd.summary$bic)],col = 'green',cex = 2,pch = 20)  
  which.min(bwd.summary$bic)
* ## [1] 3
* plot(bwd.summary$cp,xlab = 'number of variables',ylab = 'Cp',type = 'l')  
  points(which.min(bwd.summary$cp),bwd.summary$cp[which.min(bwd.summary$cp)],col = 'orange',cex = 2,pch = 20)
* 
* which.min(bwd.summary$cp)
* ## [1] 6
* #for both forward and backward selection, three-variable model could be the best.  
    
    
    
  ##forward  
    
  fwd.full = regsubsets(y~.,data = trainset, nvmax = 3,method = 'forward')  
  coef(fwd.full,3)
* ## (Intercept) x.1 x.2 x.3   
  ## 4.0011721 2.7340096 -1.9800634 0.5980388
* #It shows x1,x2,x3 will ve the best variable, and the model is y= 4.001 + 2.734\*x - -1.980\*x^2 + 0.598\*x^3.  
    
    
  ##backward  
  #one feature  
  bwd.full = regsubsets(y~.,data = trainset, nvmax = 3,method = 'backward')  
  coef(bwd.full,3)
* ## (Intercept) x.1 x.2 x.3   
  ## 4.0011721 2.7340096 -1.9800634 0.5980388
* #It shows x1, x2,x3 will ve the best variable, and the model is y = 4.001 + 2.734\*x - 1.980\*x^2 + 0.598\*x^3.  
    
    
  #calculate test error  
  test.mat2 = model.matrix(y~., testset)  
    
  mean((dta$y[test]-test.mat2[,names(coef(bwd.full,3))]%\*%coef(bwd.full,3))^2)
* ## [1] 0.9400014
* #the test error is 0.9400014.  
    
  #The result of one-feature forward-selected model in (d) is the same as (c).
  1. Now fit a LASSO model with glmnet function from glmnet package to the simulated data, again using as predictors. Use 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.

xe = model.matrix(y~.,dta)[,-1]  
ye = dta$y  
  
#grid = 10^seq(10,-2,length = 100)  
  
  
set.seed(2)  
cv.out = cv.glmnet(xe[train,],ye[train],alpha = 1)  
plot(cv.out)



bestlam = which.min(cv.out$lambda)  
bestlam

## [1] 58

#refit the model on the total training data.  
  
lasso = glmnet(xe[train,],ye[train],alpha = 1)  
  
lasso.predict = predict(lasso,s = bestlam,newx = xe[test,])  
mean((lasso.predict-ye[test])^2)

## [1] 14.03736

#the test error is 14.03736  
  
lasso.coef = predict(lasso,type = 'coefficients',s = bestlam)[1:10,]  
lasso.coef

## (Intercept) x.1 x.2 x.3 x.4 x.5   
## 2.512023 0.000000 0.000000 0.000000 0.000000 0.000000   
## x.6 x.7 x.8 x.9   
## 0.000000 0.000000 0.000000 0.000000

#all the coefficients are zero, except for intercept, which is 2.512023.  
  
#Lasso regression can set coefficients to zero, leading to a sparse model.

(f) Now generate a response vector $Y$ according to the model $$Y = \beta\_0 + \beta\_6 X^6 + \epsilon,$$ where $\beta\_6 = 6$, and perform best subset selection and the LASSO. Discuss the results carefully.   
  
```r  
#best subset selection  
b6 = 6  
Y\_new = b0 + b6\*x^6 + e  
  
  
dta\_new = data.frame(cbind(x.new,Y\_new))  
  
  
regfit.new = regsubsets(Y\_new~.,dta\_new,nvmax = 10)  
regfit.new.summary= summary(regfit.new)  
  
par(mfrow = c(2,2))  
  
####  
  
plot(regfit.new.summary$adjr2,xlab = "number of variables", ylab = 'adjusted R-squared',type = 'l')  
points(which.max(regfit.new.summary$adjr2),regfit.new.summary$adjr2[which.max(regfit.new.summary$adjr2)],col = 'red',cex = 2,pch = 20)  
which.max(regfit.new.summary$adjr2)  
```  
  
```  
## [1] 4  
```  
  
```r  
 coef(regfit.new,which.max(regfit.new.summary$adjr2))# x1,x2,x3,x6  
```  
  
```  
## (Intercept) x.1 x.2 x.3 x.6   
## 3.9219693 -0.2870086 0.2037227 0.1201462 5.9894742  
```  
  
```r  
####  
  
plot(regfit.new.summary$bic,xlab = "number of variables",ylab = 'BIC',type = 'l')  
points(which.min(regfit.new.summary$bic),regfit.new.summary$bic[which.min(regfit.new.summary$bic)],col = 'green',cex = 2,pch = 20)  
which.min(regfit.new.summary$bic)  
```  
  
```  
## [1] 1  
```  
  
```r  
coef(regfit.new,which.min(regfit.new.summary$bic))#x6  
```  
  
```  
## (Intercept) x.6   
## 4.032580 6.001114  
```  
  
```r  
####  
plot(regfit.new.summary$cp,xlab = 'number of variables',ylab = 'Cp',type = 'l')  
  
points(which.min(regfit.new.summary$cp),regfit.new.summary$cp[which.min(regfit.new.summary$cp)],col = 'orange',cex = 2,pch = 20)  
which.min(regfit.new.summary$cp)  
```  
  
```  
## [1] 4  
```  
  
```r  
coef(regfit.new,which.min(regfit.new.summary$cp))# x1,x2,x3,x6  
```  
  
```  
## (Intercept) x.1 x.2 x.3 x.6   
## 3.9219693 -0.2870086 0.2037227 0.1201462 5.9894742  
```  
  
```r  
# Lasso Regression  
   
 set.seed(12)  
 grid.f = 10^seq(10,-2,length= nrow(dta\_new))  
   
 train.f <- sample(1:nrow(dta\_new), nrow(dta\_new)/2) #create training set  
 test.f <- (-train.f)  
 y.test.f <- dta\_new[test.f]  
   
 xf = model.matrix(Y\_new~.,dta\_new)[,-1]  
 yf = dta\_new$Y\_new  
   
   
 lasso.f = cv.glmnet(xf[train.f,],yf[train.f],alpha = 1)  
 plot(lasso.f)  
```  
  
![](Hw3\_regularization\_COMP\_files/figure-docx/unnamed-chunk-6-1.png)<!-- -->  
  
```r  
 lambda.best = lasso.f$lambda.min  
   
 lambda.best  
```  
  
```  
## [1] 5.281132  
```  
  
```r  
 lasso.full = glmnet(xf[train.f,],yf[train.f],alpha = 1,lambda = lambda.best)  
 predict(lasso.full,type = 'coefficients',s = lambda.best)# x6  
```  
  
```  
## 11 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 5.760289  
## x.1 .   
## x.2 .   
## x.3 .   
## x.4 .   
## x.5 .   
## x.6 5.825365  
## x.7 .   
## x.8 .   
## x.9 .   
## x.10 .  
```  
  
```r  
 #For best subset selection, both Cp and adjusted R-squared propose x1,x2,x3,x6. BIC proposes x6, whose coefficient and intercept are close to the real value.  
 # For lasso regression, it propses x6, and coefficient and intercept are close to real values.  
```

1. (Prediction, *25 pt*) In this exercise, we will predict the number of applications received using the other variables in the Boston data set from MASS package.
   1. Randomly split the data set into a training set and a test set (2:1).

library(MASS)  
  
#head(Boston)  
  
dim(Boston)

## [1] 506 14

#506  
  
set.seed(648)  
  
btrain = sample(1:nrow(Boston),nrow(Boston)\*2/3)  
btest = (-btrain)  
  
btrainset = Boston[btrain,]#337  
nrow(btrainset)

## [1] 337

btestset = Boston[btest,]#169  
nrow(btestset)

## [1] 169

(b) Fit a linear model using least squares on the training set, and report the test error obtained.  
  
```r  
linear = lm(crim~.,btrainset)  
lm.predict = predict(linear,btestset)  
  
error = mean((btestset$crim-lm.predict)^2)  
error  
```  
  
```  
## [1] 22.98334  
```  
  
```r  
#test error is 22.98334  
```  
  
(c) Fit a ridge regression model on the training set, with $\lambda$ chosen by 5-fold cross-validation. Report the test error obtained.  
  
```r  
grid2 = 10^seq(10,-2,length = 100)  
  
ridge.train = model.matrix(crim~.,btrainset)  
ridge.test = model.matrix(crim~.,btestset)  
  
ridge.cv = cv.glmnet(ridge.train,btrainset$crim,alpha = 0,lambda = grid2,nfolds = 5)  
  
lamb = ridge.cv$lambda.min  
  
ridge.full = glmnet(ridge.train,btrainset$crim,alpha = 0,lambda = grid2)  
ridge.predict = predict(ridge.full,s = lamb,newx = ridge.test)  
  
mean((btestset$crim- ridge.predict)^2)  
```  
  
```  
## [1] 25.10415  
```  
  
```r  
# the test error is 25.10415  
```  
  
(d) Fit a LASSO model on the training set, with $\lambda$ chosen by 5-fold cross-validation. Report the test error obtained, along with the number of non-zero coefficient estimates.  
  
```r  
 lasso.train = model.matrix(crim~.,btrainset)  
 lasso.test = model.matrix(crim~.,btestset)  
  
 lasso.cv = cv.glmnet(lasso.train,btrainset$crim,alpha = 1,lambda = grid2,nfolds = 5)  
  
lam = lasso.cv$lambda.min  
  
lasso.full = glmnet(lasso.train,btrainset$crim,alpha = 0,lambda = lam)  
lasso.predict = predict(lasso.full,s = lam,newx = lasso.test)  
  
l.cof = predict(lasso.full,s = lam,type = 'coefficients')  
l.cof  
```  
  
```  
## 15 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 14.173002475  
## (Intercept) .   
## zn 0.053942794  
## indus -0.049008227  
## chas -0.585423046  
## nox -10.084407912  
## rm 0.910988797  
## age 0.007403446  
## dis -1.143767690  
## rad 0.569777979  
## tax -0.003036313  
## ptratio -0.234182175  
## black -0.004715275  
## lstat 0.045170560  
## medv -0.253261074  
```  
  
```r  
mean((btestset$crim- lasso.predict)^2)  
```  
  
```  
## [1] 22.8967  
```  
  
```r  
# the test error is 22.8967, and the number of non-zero coefficient is 13.  
```  
(e) Fit a PCR model on the training set, with $M$ chosen by 5-fold cross-validation. Report the test error obtained, along with the value of $M$ selected by cross-validation.  
  
```r  
pcr.model = pcr(crim~., data = btrainset,scale = TRUE,validation = 'CV',segments = 5)  
validationplot(pcr.model, val.type = "MSEP")  
```  
  
![](Hw3\_regularization\_COMP\_files/figure-docx/unnamed-chunk-11-1.png)<!-- -->  
  
```r  
# when m = 3, cross-validation error is minimum.  
  
pcr\_pre = predict(pcr.model,btestset,ncomp = 3)  
  
mean((btestset$crim-pcr\_pre)^2)  
```  
  
```  
## [1] 26.44724  
```  
  
```r  
# the test error is 26.44724.  
```  
(f) Comment on the results obtained. How accurately can we predict the number of crimes? Is there much difference among the test errors resulting from these four approaches?  
  
```r  
# The model that has minimum test error is LASSO regression, which is 22.8967; The model has largest test error is PCR model, which is 26.44724.  
# No, there is not much difference among the test errors, and they are all around 25.  
```