Econ 7035, Spring 2021, Assignment 4

SIU Chung Pan 20444990

21MAR2021

# Applied Exercises

## Exercise 8 in section 6.8

In this exercise, we will generate simulated data, and will then use this data to perform best subset selection.

(a). Use the **rnorm()** function to generate a predictor X of length n = 100, as well as a noise vector of length n = 100.

set.seed(10)  
X <- rnorm(100)  
epsilon <- rnorm(100)

(b). Generate a response vector Y of length n = 100 according to the model

where and are constants of your choice.

Y <- 2 + 3\*X - 1\*X^2 + 0.5\*X^3 + epsilon

(c).Use the **regsubsets()** function to perform best subset selection in order to choose the best model containing the predictors X, ,…,. 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. Note you will need to use the **data.frame()** function to create a single data set containing both X and Y .

library(leaps)  
df <- data.frame(Y,X)  
mod <- regsubsets(Y ~ poly(X, 10, raw=TRUE), data = df, nvmax = 10)  
mod\_summary <- summary(mod)  
  
which.min(mod\_summary$cp)

## [1] 3

which.min(mod\_summary$bic)

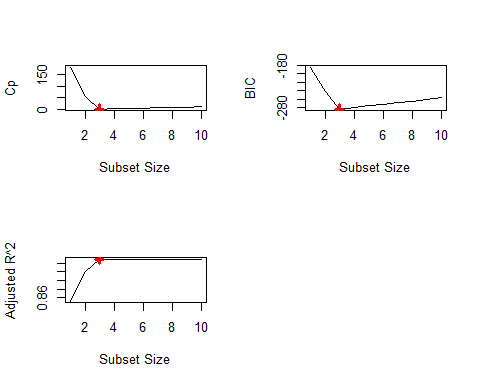
## [1] 3

which.max(mod\_summary$adjr2)

## [1] 3

par(mfrow=c(2,2))  
  
plot(mod\_summary$cp, xlab="Subset Size", ylab="Cp", type="l")  
points(which.min(mod\_summary$cp), mod\_summary$cp[3], pch=3, col="red", lwd=3)  
  
plot(mod\_summary$bic, xlab="Subset Size", ylab="BIC", type="l")  
points(which.min(mod\_summary$cp), mod\_summary$bic[3], pch=3, col="red", lwd=3)  
  
plot(mod\_summary$adjr2, xlab="Subset Size", ylab="Adjusted R^2", type="l")  
points(which.max(mod\_summary$adjr2), mod\_summary$adjr2[3], pch=3, col="red", lwd=3)  
  
coef(mod,which.min(mod\_summary$bic))

## (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2   
## 1.9289736 2.8842119 -1.0363781   
## poly(X, 10, raw = TRUE)3   
## 0.5211132



The best model obtained according to , is the one that with lowest , which is the model with three variables. The best model obtained according to BIC, is the one that with lowest BIC, which is the model with three variables. The best model obtained according to adjusted , is the one that with largest adjusted R^2, which is the model with three variables.

The best model has the intercept term of 1.9289736, X^1 term with coefficient of 2.8842119, X^2 term with coefficient of -1.0363781, and X^3 term with coefficient of 0.5211132.

(d). Repeat (c), using forward stepwise selection and also using backwards stepwise selection. How does your answer compare to the results in (c)?

# Forward stepwise selection  
mod\_fwd <- regsubsets(Y ~ poly(X, 10, raw=TRUE), data = df, nvmax = 10, method = 'forward')  
mod\_fwd\_summary <- summary(mod\_fwd)  
  
which.min(mod\_fwd\_summary$cp)

## [1] 3

which.min(mod\_fwd\_summary$bic)

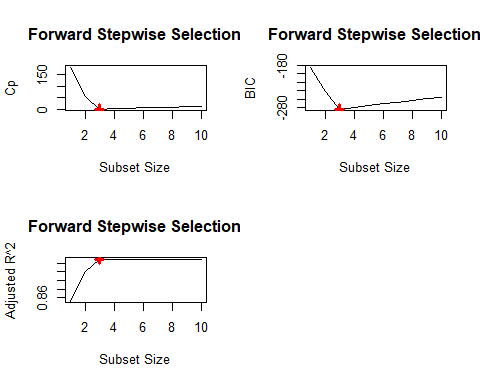
## [1] 3

which.max(mod\_fwd\_summary$adjr2)

## [1] 3

par(mfrow=c(2,2))  
  
plot(mod\_fwd\_summary$cp, main="Forward Stepwise Selection", xlab="Subset Size", ylab="Cp", type="l")  
points(which.min(mod\_fwd\_summary$cp), mod\_fwd\_summary$cp[3], pch=3, col="red", lwd=3)  
  
plot(mod\_fwd\_summary$bic, main="Forward Stepwise Selection", xlab="Subset Size", ylab="BIC", type="l")  
points(which.min(mod\_fwd\_summary$cp), mod\_fwd\_summary$bic[3], pch=3, col="red", lwd=3)  
  
plot(mod\_fwd\_summary$adjr2, main="Forward Stepwise Selection", xlab="Subset Size", ylab="Adjusted R^2", type="l")  
points(which.max(mod\_fwd\_summary$adjr2), mod\_fwd\_summary$adjr2[3], pch=3, col="red", lwd=3)  
  
coef(mod,which.min(mod\_fwd\_summary$bic))

## (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2   
## 1.9289736 2.8842119 -1.0363781   
## poly(X, 10, raw = TRUE)3   
## 0.5211132



# Backward stepwise selection  
mod\_bwd <- regsubsets(Y ~ poly(X, 10, raw=TRUE), data = df, nvmax = 10, method = 'forward')  
mod\_bwd\_summary <- summary(mod\_bwd)  
  
which.min(mod\_bwd\_summary$cp)

## [1] 3

which.min(mod\_bwd\_summary$bic)

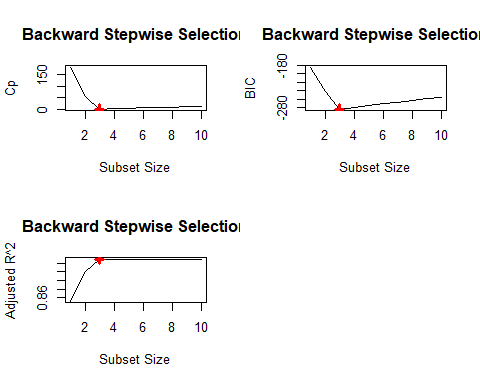
## [1] 3

which.max(mod\_bwd\_summary$adjr2)

## [1] 3

par(mfrow=c(2,2))  
  
plot(mod\_bwd\_summary$cp, main="Backward Stepwise Selection", xlab="Subset Size", ylab="Cp", type="l")  
points(which.min(mod\_bwd\_summary$cp), mod\_bwd\_summary$cp[3], pch=3, col="red", lwd=3)  
  
plot(mod\_bwd\_summary$bic, main="Backward Stepwise Selection", xlab="Subset Size", ylab="BIC", type="l")  
points(which.min(mod\_bwd\_summary$cp), mod\_bwd\_summary$bic[3], pch=3, col="red", lwd=3)  
  
plot(mod\_bwd\_summary$adjr2, main="Backward Stepwise Selection", xlab="Subset Size", ylab="Adjusted R^2", type="l")  
points(which.max(mod\_bwd\_summary$adjr2), mod\_bwd\_summary$adjr2[3], pch=3, col="red", lwd=3)  
  
coef(mod,which.min(mod\_bwd\_summary$bic))

## (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2   
## 1.9289736 2.8842119 -1.0363781   
## poly(X, 10, raw = TRUE)3   
## 0.5211132



Both forward stepwise selection and backwards stepwise selection produce the same result with (c).

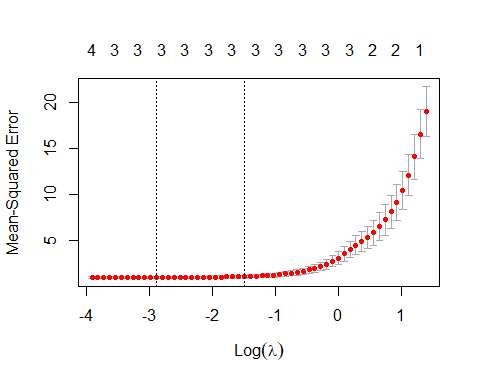
(e). Now fit a lasso model to the simulated data, again using X, ,…, as predictors. Use cross-validation to select the optimal value of . Create plots of the cross-validation error as a function of . Report the resulting coefficient estimates, and discuss the results obtained.

set.seed(100)  
library(glmnet)  
xmat <- model.matrix(Y ~ poly(X, 10, raw=TRUE), data = df)[, -1]  
cv\_lasso <- cv.glmnet(xmat, Y, alpha = 1)  
(optimal\_lambda <- cv\_lasso$lambda.min)

## [1] 0.05578232

The optimal value of is 0.05578232.

plot(cv\_lasso)



fit\_lasso <- glmnet(xmat, Y, alpha = 1)  
predict(fit\_lasso, s = optimal\_lambda, type = "coefficients")[1:11, ]

## (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2   
## 1.8848034 2.8596122 -0.9949697   
## poly(X, 10, raw = TRUE)3 poly(X, 10, raw = TRUE)4 poly(X, 10, raw = TRUE)5   
## 0.5101443 0.0000000 0.0000000   
## poly(X, 10, raw = TRUE)6 poly(X, 10, raw = TRUE)7 poly(X, 10, raw = TRUE)8   
## 0.0000000 0.0000000 0.0000000   
## poly(X, 10, raw = TRUE)9 poly(X, 10, raw = TRUE)10   
## 0.0000000 0.0000000

The resulting coefficient estimates are 1.8848034, 2.8596122, -0.9949697, 0.5101443 for the intercept term, X^1, X^2, X^3 respectively. Comparing these coefficients with that in (c), all coefficients move towards zero. Since lasso adds penalty term to the loss function, and then shrinks all coefficients.

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

# best subset selection  
Y <- 50 + 100\*X^7 + epsilon  
df <- data.frame(Y,X)  
mod <- regsubsets(Y ~ poly(X, 10, raw=TRUE), data = df, nvmax = 10)  
mod\_summary <- summary(mod)  
  
which.min(mod\_summary$cp)

## [1] 1

which.min(mod\_summary$bic)

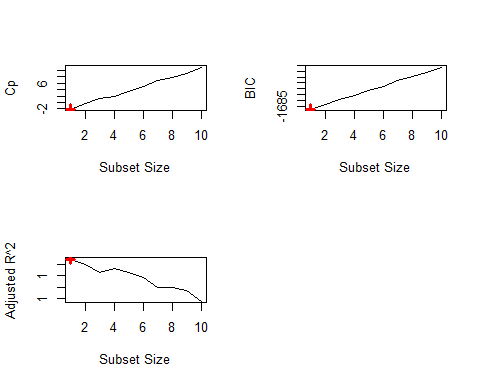
## [1] 1

which.max(mod\_summary$adjr2)

## [1] 1

par(mfrow=c(2,2))  
  
plot(mod\_summary$cp, xlab="Subset Size", ylab="Cp", type="l")  
points(which.min(mod\_summary$cp), mod\_summary$cp[1], pch=3, col="red", lwd=3)  
  
plot(mod\_summary$bic, xlab="Subset Size", ylab="BIC", type="l")  
points(which.min(mod\_summary$cp), mod\_summary$bic[1], pch=3, col="red", lwd=3)  
  
plot(mod\_summary$adjr2, xlab="Subset Size", ylab="Adjusted R^2", type="l")  
points(which.max(mod\_summary$adjr2), mod\_summary$adjr2[1], pch=3, col="red", lwd=3)  
  
coef(mod,which.min(mod\_summary$bic))

## (Intercept) poly(X, 10, raw = TRUE)7   
## 49.90476 99.99987



The best model obtained according to , is the one that with lowest , which is the model with one variable. The best model obtained according to BIC, is the one that with lowest BIC, which is the model with one variable. The best model obtained according to adjusted , is the one that with largest adjusted R^2, which is the model with one variable.

The best model has the intercept term of 49.90476 and X^7 term with coefficient of 99.99987.

# lasso  
xmat <- model.matrix(Y ~ poly(X, 10, raw=TRUE), data = df)[, -1]  
cv\_lasso <- cv.glmnet(xmat, Y, alpha = 1)  
optimal\_lambda <- cv\_lasso$lambda.min  
  
fit\_lasso <- glmnet(xmat, Y, alpha = 1)  
predict(fit\_lasso, s = optimal\_lambda, type = "coefficients")[1:11, ]

## (Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2   
## 43.83898 0.00000 0.00000   
## poly(X, 10, raw = TRUE)3 poly(X, 10, raw = TRUE)4 poly(X, 10, raw = TRUE)5   
## 0.00000 0.00000 0.00000   
## poly(X, 10, raw = TRUE)6 poly(X, 10, raw = TRUE)7 poly(X, 10, raw = TRUE)8   
## 0.00000 97.08482 0.00000   
## poly(X, 10, raw = TRUE)9 poly(X, 10, raw = TRUE)10   
## 0.00000 0.00000

The lasso also choose only one variable, which is X^7. The intercept term and the coefficient of X^7 are shrank by lasso regularization, so they move towards to zero.

## Exercise 9 in section 6.8

In this exercise, we will predict the number of applications received using the other variables in the College data set.

(a). Split the data set into a training set and a test set.

# Remove the # below  
library(ISLR)  
attach(College)  
set.seed(1)  
  
train <- sample(dim(College)[1], dim(College)[1] / 2)  
college\_train <- College[train, ]  
college\_test <- College[-train, ]

(b). Fit a linear model using least squares on the training set, and report the test error obtained.

fit\_lm <- lm(Apps ~ ., data = college\_train)  
pred\_lm <- predict(fit\_lm, college\_test)  
mean((pred\_lm - college\_test$Apps)^2)

## [1] 1135758

The test error obtained is 1135758.

(c). Fit a ridge regression model on the training set, with chosen by cross-validation. Report the test error obtained.

train\_mat <- model.matrix(Apps ~ ., data = college\_train)  
test\_mat <- model.matrix(Apps ~ ., data = college\_test)  
grid <- 10 ^ seq(4, -2, length = 500)  
fit\_ridge <- glmnet(train\_mat, college\_train$Apps, alpha = 0, lambda = grid, thresh = 1e-12)  
cv\_ridge <- cv.glmnet(train\_mat, college\_train$Apps, alpha = 0, lambda = grid, thresh = 1e-12)  
(bestlam\_ridge <- cv\_ridge$lambda.min)

## [1] 0.01

pred\_ridge <- predict(fit\_ridge, s = bestlam\_ridge, newx = test\_mat)  
mean((pred\_ridge - college\_test$Apps)^2)

## [1] 1135716

The best λ for ridge regression is 0.01. The test error obtained is 1135716.

(d). Fit a lasso model on the training set, with chosen by cross-validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

fit\_lasso <- glmnet(train\_mat, college\_train$Apps, alpha = 1, lambda = grid, thresh = 1e-12)  
cv\_lasso <- cv.glmnet(train\_mat, college\_train$Apps, alpha = 1, lambda = grid, thresh = 1e-12)  
(bestlam\_lasso <- cv\_lasso$lambda.min)

## [1] 0.01

pred\_lasso <- predict(fit\_lasso, s = bestlam\_lasso, newx = test\_mat)  
mean((pred\_lasso - college\_test$Apps)^2)

## [1] 1135657

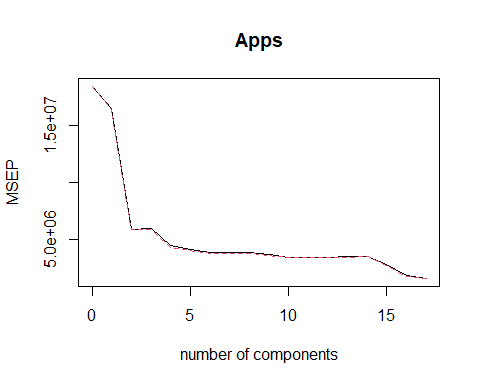
predict(fit\_lasso, s = bestlam\_lasso, type = "coefficients")

## 19 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -7.900312e+02  
## (Intercept) .   
## PrivateYes -3.070156e+02  
## Accept 1.779326e+00  
## Enroll -1.469479e+00  
## Top10perc 6.672190e+01  
## Top25perc -2.230428e+01  
## F.Undergrad 9.258478e-02  
## P.Undergrad 9.409305e-03  
## Outstate -1.083491e-01  
## Room.Board 2.115148e-01  
## Books 2.912133e-01  
## Personal 6.120479e-03  
## PhD -1.547181e+01  
## Terminal 6.409350e+00  
## S.F.Ratio 2.282604e+01  
## perc.alumni 1.130492e+00  
## Expend 4.856685e-02  
## Grad.Rate 7.487998e+00

The best λ for lasso regression is 0.01. The test error obtained is 1135657.

(e). Fit a PCR model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

library(pls)  
  
fit\_pcr <- pcr(Apps~., data=college\_train, scale=T, validation="CV")  
validationplot(fit\_pcr, val.type="MSEP")



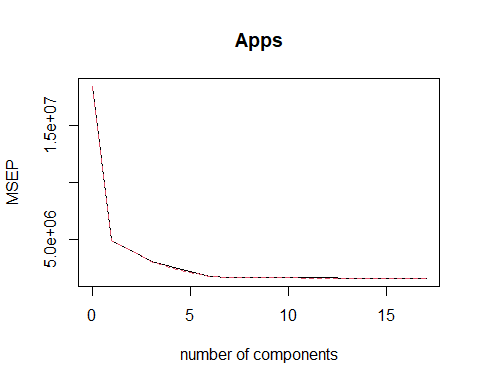
pred\_pcr <- predict(fit\_pcr, college\_test, ncomp=5)  
mean((college\_test[, "Apps"] - c(pred\_pcr))^2)

## [1] 1983650

From the plot, we choose M = 5, and the test error obtained is 1983650.

(f). Fit a PLS model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

fit\_pls <- plsr(Apps ~ ., data = college\_train, scale = TRUE, validation = "CV")  
validationplot(fit\_pls, val.type = "MSEP")



pred\_pls <- predict(fit\_pls, college\_test, ncomp = 6)  
mean((pred\_pls - college\_test$Apps)^2)

## [1] 1066991

From the plot, we choose M = 6, and the test error obtained is 1066991.

(g). Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches.

We make use of the test error to compute the R^2, to assess the accuracy using different approaches.

test\_avg <- mean(college\_test$Apps)  
cat("R-squared of least suqares: ",lm\_r2 <- 1 - mean((pred\_lm - college\_test$Apps)^2) / mean((test\_avg - college\_test$Apps)^2))

## R-squared of least suqares: 0.9015413

cat("\nR-squared of ridge: ",ridge\_r2 <- 1 - mean((pred\_ridge - college\_test$Apps)^2) / mean((test\_avg - college\_test$Apps)^2))

##   
## R-squared of ridge: 0.901545

cat("\nR-squared of lasso: ",lasso\_r2 <- 1 - mean((pred\_lasso - college\_test$Apps)^2) / mean((test\_avg - college\_test$Apps)^2))

##   
## R-squared of lasso: 0.9015501

cat("\nR-squared of pcr: ",pcr\_r2 <- 1 - mean((pred\_pcr - college\_test$Apps)^2) / mean((test\_avg - college\_test$Apps)^2))

##   
## R-squared of pcr: 0.8280378

cat("\nR-squared of pls: ",pls\_r2 <- 1 - mean((pred\_pls - college\_test$Apps)^2) / mean((test\_avg - college\_test$Apps)^2))

##   
## R-squared of pls: 0.9075028

Except for PCR, which only has R-squared of 0.8280378, all other approaches have accurate results. The R-squared of least squares is 0.9015413, the R-squared of ridge is 0.901545, the R-squared of lasso is 0.9015501, the R-squared of pcr is 0.8280378, and the R-squared of pls is 0.9075028.

## Exercise 10 in section 6.8

We have seen that as the number of features used in a model increases, the training error will necessarily decrease, but the test error may not. We will now explore this in a simulated data set.

(a). Generate a data set with p = 20 features, n = 1,000 observations, and an associated quantitative response vector generated according to the model

where has some elements that are exactly equal to zero.

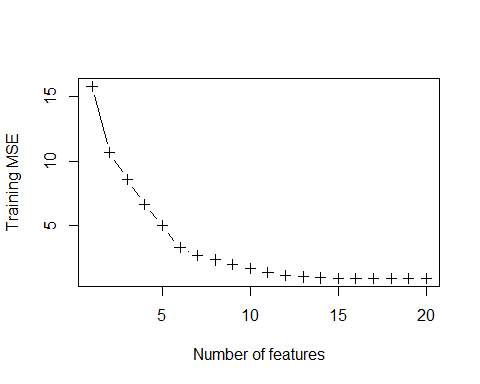
set.seed(100)  
X <- matrix(rnorm(1000 \* 20), 1000, 20)  
b <- rnorm(20)  
b[5] <- 0  
b[8] <- 0  
b[9] <- 0  
b[14] <- 0  
b[18] <- 0  
epsilon <- rnorm(1000)  
Y <- X %\*% b + epsilon

(b). Split your data set into a training set containing 100 observations and a test set containing 900 observations.

set.seed(200)  
train <- sample(seq(1000), 100)  
X\_train <- X[train, ]  
X\_test <- X[-train, ]  
Y\_train <- Y[train, ]  
Y\_test <- Y[-train, ]

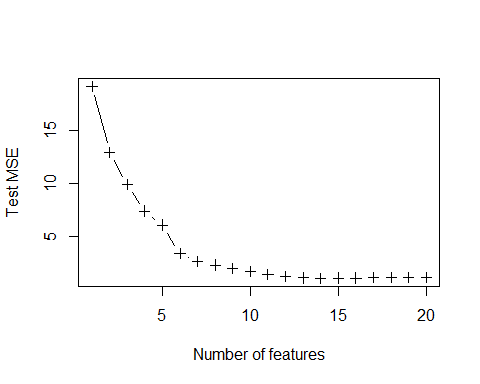
(c). Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size.

set.seed(100)  
df\_train <- data.frame(Y\_train, X\_train)  
regfit\_full <- regsubsets(Y\_train ~ ., data = df\_train, nvmax = 20)  
train\_mat <- model.matrix(Y\_train ~ ., data = df\_train, nvmax = 20)  
val\_errors <- rep(0, 20)  
  
for (i in 1:20) {  
 coef <- coef(regfit\_full, id = i)  
 pred <- train\_mat[, names(coef)] %\*% coef  
 val\_errors[i] <- mean((pred - Y\_train)^2)  
}  
plot(val\_errors, xlab = "Number of features", ylab = "Training MSE", pch = 3, type = "b")



(d). Plot the test set MSE associated with the best model of each size.

set.seed(100)  
df\_test <- data.frame(Y\_test, X\_test)  
test\_mat <- model.matrix(Y\_test ~ ., data = df\_test, nvmax = 20)  
  
val\_errors <- rep(0, 20)  
  
for (i in 1:20) {  
 coef <- coef(regfit\_full, id = i)  
 pred <- test\_mat[, names(coef)] %\*% coef  
 val\_errors[i] <- mean((pred - Y\_test)^2)  
}  
plot(val\_errors, xlab = "Number of features", ylab = "Test MSE", pch = 3, type = "b")



(e). For which model size does the test set MSE take on its minimum value? Comment on your results. If it takes on its minimum value for a model containing only an intercept or a model containing all of the features, then play around with the way that you are generating the data in (a) until you come up with a scenario in which the test set MSE is minimized for an intermediate model size.

which.min(val\_errors)

## [1] 14

The model with 14 variables has the lowest test set MSE. With more features used in the model, the training error must decrease and the testing error increases as a result of overfitting. The model fails to generalize to new observations, causing the testing error increases again when more and more features are added to the model.

(f). How does the model at which the test set MSE is minimized compare to the true model used to generate the data? Comment on the coefficient values.

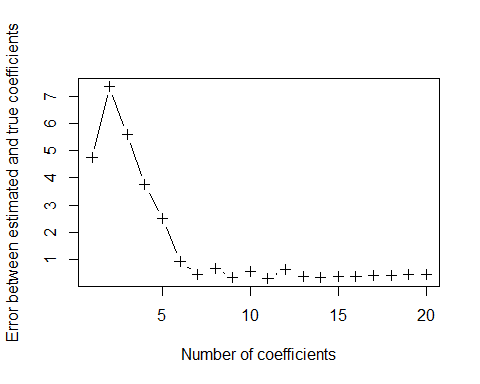
set.seed(100)  
coef(regfit\_full, which.min(val\_errors))

## (Intercept) X1 X2 X3 X4 X6   
## 0.1059887 -0.5282429 1.4160775 -0.5300182 -0.2966303 1.9441823   
## X10 X11 X12 X13 X15 X16   
## -0.4083995 0.9971357 0.6192924 -2.5479723 1.6632232 1.5132903   
## X17 X19 X20   
## 0.5102922 -3.0923406 0.6045765

The best subset model could identify all the coefficients with the value of zero in the true model.

(g). Create a plot displaying for a range of values of r, where is the jth coefficient estimate for the best model containing r coefficients. Comment on what you observe. How does this compare to the test MSE plot from (d)?

set.seed(200)  
val\_errors <- rep(0, 20)  
X\_cols <- colnames(X, do.NULL = FALSE, prefix = "X")  
  
for (i in 1:20) {  
 coef <- coef(regfit\_full, id = i)  
 val\_errors[i] <- sqrt(sum((b[X\_cols %in% names(coef)] - coef[names(coef) %in% X\_cols])^2) + sum(b[!(X\_cols %in% names(coef))])^2)  
}  
plot(val\_errors, xlab = "Number of coefficients", ylab = "Error between estimated and true coefficients", pch = 3, type = "b")



which.min(val\_errors)

## [1] 11

The model with 11 variables minimizes the error between the estimated and true coefficients. On the other hand, test error is minimized by the model with 14 variables. Therefore, a model with best fit of true coefficients may not be the model which gives the lowest test error.

## Exercise 11 in section 6.8

We will now try to predict per capita crime rate in the **Boston** dataset.

(a). Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

# Remove the # below  
library(MASS)  
attach(Boston)  
set.seed(1)  
  
# best subset selection  
  
train <- sample(dim(Boston)[1], dim(Boston)[1]\*0.8)  
Boston\_train <- Boston[train, ]  
Boston\_test <- Boston[-train, ]  
  
mod <-regsubsets(crim~.,data=Boston\_train,nbest=1,nvmax=13)  
  
Boston\_test\_mat <- model.matrix(crim~ ., data = Boston\_test, nvmax = 13)  
val\_errors <- rep(NA, 13)  
for (i in 1:13) {  
 coef <- coef(mod, id = i)  
 pred <- Boston\_test\_mat[, names(coef)] %\*% coef  
 val\_errors[i] <- mean((pred - Boston\_test$crim)^2)  
}  
  
which.min(val\_errors)

## [1] 11

(mod\_MSE<-val\_errors[11])

## [1] 68.93374

The best subset selection model has 11 variables, and the cross validation error is 68.93374.

# lasso  
set.seed(1)  
train\_mat <- model.matrix(crim ~ ., data = Boston\_train)  
test\_mat <- model.matrix(crim ~ ., data = Boston\_test)  
grid <- 10 ^ seq(4, -2, length = 500)  
  
lasso\_mod<-glmnet(train\_mat,Boston\_train$crim,alpha=1,lambda=grid)  
cv\_lasso<-cv.glmnet(train\_mat,Boston\_train$crim,alpha=1,lambda=grid)  
  
# find the best lambda which give smallest CV error  
bestlam<-cv\_lasso$lambda.min  
  
# predict with the model with the best lambda  
pred<-predict(lasso\_mod,s=bestlam,newx=test\_mat)  
  
# MSE  
(lasso\_MSE<-mean((Boston\_test$crim - pred)^2))

## [1] 69.67614

The lasso model has the cross validation error of 69.67614.

# ridge  
set.seed(1)  
ridge\_mod<-glmnet(train\_mat,Boston\_train$crim,alpha=0,lambda=grid)  
cv\_ridge<-cv.glmnet(train\_mat,Boston\_train$crim,alpha=0,lambda=grid)  
  
# find the best lambda which give smallest CV error  
bestlam<-cv\_ridge$lambda.min  
  
# predict with the model with the best lambda  
pred<-predict(ridge\_mod,s=bestlam,newx=test\_mat)  
  
# MSE  
(ridge\_MSE<-mean((Boston\_test$crim - pred)^2))

## [1] 69.97034

The ridge model has the cross validation error of 69.97034.

# pcr  
fit\_pcr <- pcr(crim~., data=Boston\_train, scale=T, validation="CV")  
  
pred\_pcr <- predict(fit\_pcr, Boston\_test, ncomp=5)  
mean((Boston\_test[, "crim"] - c(pred\_pcr))^2)

## [1] 76.07335

The pcr model has the cross validation error of 76.07335.

The cross validation errors using different models are:  
best subset selection: 68.93374  
lasso: 69.67614  
ridge: 69.97034  
pcr: 76.07335

Therefore, the best model is the best subset selection model.

(b). Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, cross-validation, or some other reasonable alternative, as opposed to using training error.

From (a), the models that seem to perform well on the Boston data set are best subset selection, lasso model, and ridge model. Since these models has relatively small cross validation errors. Furthermore, the best model is the best subset selection model because it has the smallest cross validation error.

(c). Does your chosen model involve all of the features in the dataset? Why or why not?

No. The model I chose is the best selection model, which only has 11 features instead of all 13 features. In model selection, the objective is to select a model that gives the smallest test error. If all features are included in the model, even though the model fits training data very well, the model fails to generalize to new observations, as a result of overfitting.