Lab 6: Cross-validation & Regularisation

ACTL3142 and ACTL5110

Questions

Conceptual Questions

1. ⋆ (ISLR2, Q5.3) We now review k-fold cross-validation.

   a. Explain how k-fold cross-validation is implemented.

   b. What are the advantages and disadvantages of k-fold cross-validation relative to:

      - The validation set approach?
      - LOOCV?

   Solution

2. ⋆ (ISLR2, Q6.2) For parts (a) through (c), indicate which of i. through iv. is correct. Justify your answer.

   a. The lasso, relative to least squares, is:

      i. More flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

      ii. More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

      iii. Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

      iv. Less flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

   b. Repeat (a) for ridge regression relative to least squares.

   c. Repeat (a) for non-linear methods relative to least squares.

Solution

3. ⋆ (ISLR2, Q6.3) Suppose we estimate the regression coefficients in a linear regression model by minimizing

$$\sum_{i=1}^{n}\Big(y_i - \beta_0 - \sum_{j=1}^{p}\beta_i x_{ij}\Big)^2 \text{ subject to } \sum_{j=1}^{p}|\beta_j| \leq s$$

for a particular value of s. For parts (a) through (e), indicate which of i. through v. is correct. Justify your answer.

   a. As we increase s from 0, the training RSS will:
      i. Increase initially, and then eventually start decreasing in an inverted U shape.
      ii. Decrease initially, and then eventually start increasing in a U shape.
      iii. Steadily increase.
      iv. Steadily decrease.
      v. Remain constant.
   b. Repeat (a) for test RSS.
   c. Repeat (a) for variance.
   d. Repeat (a) for (squared) bias.
   e. Repeat (a) for the irreducible error.

Solution

4. (ISLR2, Q6.5) It is well-known that ridge regression tends to give similar coefficient values to correlated variables, whereas the lasso may give quite different coefficient values to correlated variables. We will now explore this property in a very simple setting.
Suppose that $n = 2$, $p = 2$, $x_{11} = x_{12}$, $x_{21} = x_{22}$. Furthermore, suppose that $y_1 + y_2 = 0$ and $x_{11} + x_{21} = 0$ and $x_{12} + x_{22} = 0$, so that the estimate for the intercept in a least squares, ridge regression, or lasso model is zero: $\hat{\beta}_0 = 0$.

   a. Write out the ridge regression optimization problem in this setting.
   b. Argue that in this setting, the ridge coefficient estimates satisfy $\hat{\beta}_1 = \hat{\beta}_2$.
   c. Write out the lasso optimization problem in this setting.
   d. Argue that in this setting, the lasso coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$ are not unique—in other words, there are many possible solutions to the optimization problem in (c). Describe these solutions.

Solution

Additional Questions

1. Derive the LOOCV error for linear models:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2$$

Take note of the Sherman-Morrison formula:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}$$

where A is an invertible square matrix, and u, v are column vectors.

Solution

Applied Questions

1. ⋆ (ISLR2, Q5.5) In Chapter 4, we used logistic regression to predict the probability of `default` using `income` and `balance` on the `Default` data set. We will now estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

   a. Fit a logistic regression model that uses `income` and `balance` to predict `default`.

   b. Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

      i. Split the sample set into a training set and a validation set.

      ii. Fit a multiple logistic regression model using only the training observations.

      iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the `default` category if the posterior probability is greater than 0.5.

      iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

   c. Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

   d. Now consider a logistic regression model that predicts the probability of `default` using `income`, `balance`, and a dummy variable for student. Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for `student` leads to a reduction in the test error rate.

Solution

2. (ISLR2, Q5.7) In Sections 5.3.2 and 5.3.3, we saw that the `cv.glm()` function can be used in order to compute the LOOCV test error estimate. Alternatively, one could compute those quantities using just the `glm()` and `predict.glm()` functions, and a for loop. You will now take this approach in order to compute the LOOCV error for a simple logistic regression model on the `Weekly` data set. Recall that in the context of classification problems, the LOOCV error is given in (5.4).

   a. Fit a logistic regression model that predicts `Direction` using `Lag1` and `Lag2`.

   b. Fit a logistic regression model that predicts `Direction` using `Lag1` and `Lag2` using all but the first observation.

   c. Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if $\mathbb{P}(\texttt{Direction = "Up"}|\texttt{Lag1}, \texttt{Lag2}) > 0.5$. Was this observation correctly classified?

   d. Write a for loop from $i = 1$ to $i = n$, where n is the number of observations in the data set, that performs each of the following steps:

      i. Fit a logistic regression model using all but the ith observation to predict `Direction` using `Lag1` and `Lag2`.

      ii. Compute the posterior probability of the market moving up for the ith observation.

      iii. Use the posterior probability for the ith observation in order to predict whether or not the market moves up.

      iv. Determine whether or not an error was made in predicting the direction for the ith observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

   e. Take the average of the n numbers obtained in (d)iv in order to obtain the LOOCV estimate for the test error. Comment on the results.

Solution

3. $\star$ (ISLR2, Q5.8) We will now perform cross-validation on a simulated data set.

   a. Generate a simulated data set as follows:

```
set.seed(1)
x <- rnorm(100)
y <- x - 2 * x^2 + rnorm(100)
```

In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

b. Create a scatterplot of X against Y. Comment on what you find.

c. Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

   i. $Y = \beta_0 + \beta_1 X + \epsilon$

   ii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$

   iii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$

   iv. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$

   Note you may find it helpful to use the `data.frame()` function to create a single data set containing both X and Y .

d. Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

e. Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

f. Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?

Solution

4. ⋆ (ISLR2, Q6.9) 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.

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

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

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

   e. 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?

Solution

5. (ISLR2, Q6.10) 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

   $$Y = X\beta + \epsilon$$

   where β has some elements that are exactly equal to zero.

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

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

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

   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.

   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.

   g. Create a plot displaying $\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$ for a range of values of r, where $\hat{\beta}_j^r$ 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)?

   Solution

Solutions

Conceptual Questions

1. Refer to Module 5 lecture slides and Chapter 5 of the book for this question.
2. a. iii. It sets certain variables to zero, and hence is less flexible than least squares.

   b. iii. It restricts the size of the parameter estimates, whereas least squares does not.

   c. ii. It is more flexible since least squares only does a linear fit.

3. a. Increasing s increases the model's flexibility, which improves the model's fit on the training data.

   b. Initially, it will decrease as the model fits more signal. However, at some point, the model will start overfitting the training data, which means the performance on the test data will worsen.

   c. iii. As the model becomes more flexible, its variance will increase.

   d. iv. As the model becomes more flexible, its bias will decrease.

   e. v. Irreducible error is irreducible.

4. a. Minimise on β:

$$\sum_{i=1}^{2}(y_i - (\beta_1 + \beta_2)x_{i1})^2 \text{ given } \beta_1^2 + \beta_2^2 \leq s$$

   b. To ensure the RSS is minimised as far as possible, we must ensure the range of $\beta_1 + \beta_2$ is maximised under the constraint. Consider the case where $\beta_1 > \beta_2$, in other words, where $\beta_1 = \beta_2 + \epsilon$ for some $\epsilon > 0$. Then, the maximum value of β_2 obtainable is:

$$\begin{aligned} s &= (\beta_2 + \epsilon)^2 + \beta_2^2 \\ &= 2\beta_2^2 + 2\epsilon\beta_2 + \epsilon^2 \\ \frac{s}{2} - \frac{\epsilon^2}{4} &= \beta_2^2 + \epsilon\beta_2 + \frac{\epsilon^2}{4} \\ &= \left(\beta_2^2 + \frac{\epsilon}{2}\right)^2 \\ \beta_2 &= \sqrt{\frac{s}{2} - \frac{\epsilon^2}{4}} - \frac{\epsilon}{2} \\ \beta_1 + \beta_2 &= 2\sqrt{\frac{s}{2} - \frac{\epsilon^2}{4}} \end{aligned}$$

   Clearly, this is maximised when $\epsilon = 0$, in other words, when $\beta_1 = \beta_2$.

   c. Minimise on β:

$$\sum_{i=1}^{2}(y_i - (\beta_1 + \beta_2)x_{i1})^2 \text{ given } |\beta_1| + |\beta_2| \leq s$$

   d. Once again, we wish to maximise the range of $\beta_1 + \beta_2$ subject to the constraint. Assume $\beta_1 = \beta_2 + \epsilon$, and $\beta_1 > 0, \beta_2 \geq 0$. This assumption is robust under all possible β_1 and β_2. Then $\beta_2 = (s - \epsilon)/2$, hence $\beta_1 + \beta_2 = s$. Therefore, any combination of β_1, β_2, provided they consume the entire budget implied by λ, they have the same sign, and the sign is the same as $y_i x_{i1}$.

Additional Questions

1. Let:

   - Y_{-i} denote the response vector without the ith observation
   - X_{-i} denote the design matrix without the ith observation
   - $\hat{\beta}_{-i}$ denote the parameter estimates after regressing without the ith observation
   - $\boldsymbol{x}_i^T$ denote the ith observation's predictor values (i.e. the ith row of the design matrix).

   All other variables have their normal meaning.

   To start with, note that

   $$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_{i,-i})^2$$

   Where $\hat{y}_{i,-i}$ is the predicted value of y_i after the ith observation has been left out.

   Now,

   $$\hat{y}_{i,-i} = \boldsymbol{x}_i^T\hat{\beta}_{-i}$$

   We shall work on $\hat{\beta}_{-i}$

   $$\hat{\beta}_{-i} = (X_{-i}^T X_{-i})^{-1} X_{-i}^T Y_{-i}$$

   Use Sherman-Morrison to obtain a value of $(X_{-i}^T X_{-i})^{-1}$.

   $$\begin{aligned}
   X_{-i}^T X_{-i} &= X^T X - \boldsymbol{x}_i\boldsymbol{x}_i^T \\
   (X_{-i}^T X_{-i})^{-1} &= (X^T X)^{-1} + \frac{(X^T X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^T(X^T X)^{-1}}{1 - \boldsymbol{x}_i^T(X^T X)^{-1}\boldsymbol{x}_i} \\
   &= (X^T X)^{-1} + \frac{(X^T X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^T(X^T X)^{-1}}{1 - h_i} \\
   X_{-i}^T Y_{-i} &= X^T Y - \boldsymbol{x}_i y_i
   \end{aligned}$$

   Multiply $(X_{-i}^T X_{-i})^{-1}$ and $X{-}i^T Y_{-i}$ and simplify to obtain an expression for $\hat{\beta}_{-i}$.

   $$\begin{aligned}
   \hat{\beta}_{-i} &= \left((X^T X)^{-1} + \frac{(X^T X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^T(X^T X)^{-1}}{1 - h_i}\right)(X^T Y - \boldsymbol{x}_i y_i) \\
   &= (X^T X)^{-1}X^T Y + \frac{(X^T X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^T(X^T X)^{-1}X^T Y}{1 - h_i} - (X^T X)^{-1}\boldsymbol{x}_i y_i - \\
   &\frac{(X^T X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^T(X^T X)^{-1}\boldsymbol{x}_i y_i}{1 - h_i} \\
   &= \hat{\beta} + \frac{(X^T X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^T\hat{\beta}}{1 - h_i} - (X^T X)^{-1}\boldsymbol{x}_i y_i - \frac{(X^T X)^{-1}\boldsymbol{x}_i h_i y_i}{1 - h_i}
   \end{aligned}$$

Now right-multiply by $\boldsymbol{x}_i$ to find the prediction for $\hat{y}_{-i,i}$

$$\begin{aligned}
\hat{y}_{i,-i} &= \boldsymbol{x}_i^T\hat{\beta} + \frac{\boldsymbol{x}_i^T(X^TX)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^T\hat{\beta}}{1-h_i} - \boldsymbol{x}_i^T(X^TX)^{-1}\boldsymbol{x}_i y_i - \frac{\boldsymbol{x}_i^T(X^TX)^{-1}\boldsymbol{x}_i h_i y_i}{1-h_i} \\
&= \boldsymbol{x}_i^T\hat{\beta} + \frac{h_i\boldsymbol{x}_i^T\hat{\beta}}{1-h_i} - h_i y_i - \frac{h_i^2 y_i}{1-h_i} \\
&= \frac{\boldsymbol{x}_i^T\hat{\beta} - h_i y_i}{1-h_i} \\
y_i - \hat{y}_{i,-i} &= \frac{y_i - h_i y_i - \boldsymbol{x}_i^T\hat{\beta} + h_i y_i}{1-h_i} \\
&= \frac{y_i - \boldsymbol{x}_i^T\hat{\beta}}{1-h_i} \\
&= \frac{y_i - \hat{y}_i}{1-h_i}
\end{aligned}$$

$$\therefore CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1-h_i}\right)^2$$

Applied Questions

1. a.
```
require(ISLR2)
```
```
Loading required package: ISLR2
```
```
fit <- glm(default ~ income + balance, family = binomial, data = Default)
```

b. i.
```
train <- sample(dim(Default)[1], dim(Default)[1] / 2)
train <- seq(1, dim(Default)[1]) %in% train
```

ii.
```
fit.train <- glm(default ~ balance + income,
  data = Default, subset = train,
  family = binomial
)
```

iii.
```
predicted <- rep("No", dim(Default)[1] / 2) # create a vector "predicted"
responses <- predict(fit.train, newdata = Default[!train, ], type = "response")
predicted[responses > 0.5] <- "Yes"
```

iv.
```
table(true = Default$default[!train], pred = predicted)
```
```
      pred
true    No  Yes
  No  4801   25
  Yes  121   53
```

```
100 * mean(predicted != Default$default[!train]) # test error rate
```

```
[1] 2.92
```

```
100 * (108 + 19) / (4824 + 19 + 108 + 49) # test error rate check
```

```
[1] 2.54
```

The error rate is about 2.54%.

c.
```
for (i in 2:4) {
  set.seed(i)
  train <- sample(dim(Default)[1], dim(Default)[1] / 2)
  train <- seq(1, dim(Default)[1]) %in% train
  fit.train <- glm(default ~ balance + income,
    data = Default, subset = train,
    family = binomial
  )
  predicted <- rep("No", dim(Default)[1] / 2)
  responses <- predict(fit.train, newdata = Default[!train, ], type = "response")
  predicted[responses > 0.5] <- "Yes"
  curr.table <- table(true = Default$default[!train], pred = predicted)
  print(100 * (curr.table[2, 1] + curr.table[1, 2]) / 5000)
}
```

```
[1] 2.38
[1] 2.64
[1] 2.56
```

Error rates are not constant, but are hovering around the mid 2% mark.

d.
```
for (i in 1:4) {
  set.seed(i)
  train <- sample(dim(Default)[1], dim(Default)[1] / 2)
  train <- seq(1, dim(Default)[1]) %in% train
  fit.train <- glm(default ~ balance + income + student,
    data = Default, subset = train,
    family = binomial
  )
  predicted <- rep("No", dim(Default)[1] / 2)
  responses <- predict(fit.train, newdata = Default[!train, ], type = "response")
  predicted[responses > 0.5] <- "Yes"
  curr.table <- table(true = Default$default[!train], pred = predicted)
  print(100 * (curr.table[2, 1] + curr.table[1, 2]) / 5000)
}
```

```
[1] 2.6
[1] 2.46
[1] 2.72
[1] 2.62
```

The error rates are about the same. The inclusion of the new predictor didn't improve the fit.

2. a.
```
require(ISLR2)
fit <- glm(Direction ~ Lag1 + Lag2, data = Weekly, family = binomial)
```

b.
```
fit2 <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-1, ], family = binomial)
```

c.
```
pred <- predict(fit2, newdata = Weekly[1, ], type = "response")
class <- "No"
if (pred > 0.5) {
  class <- "Yes"
}
pred # probability predicted
```

```
        1
0.5713923
```

```
class # class predicted
```

```
[1] "Yes"
```

```
Weekly$Direction[1] # real class
```

```
[1] Down
Levels: Down Up
```

```
class %in% Weekly$Direction[1] # prediction not correctly classified
```

```
[1] FALSE
```

No. It's not correctly classified.

d.
```
errors <- rep(1, dim(Weekly)[1])
for (i in 1:dim(Weekly)[1]) {
  fit <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-i, ], family = binomial)
  pred <- predict(fit2, newdata = Weekly[i, ], type = "response")
  class <- "Down"
  if (pred > 0.5) {
    class <- "Up"
  }
  if (class == Weekly$Direction[i]) {
```

```
    errors[i] <- 0
  }
}
```

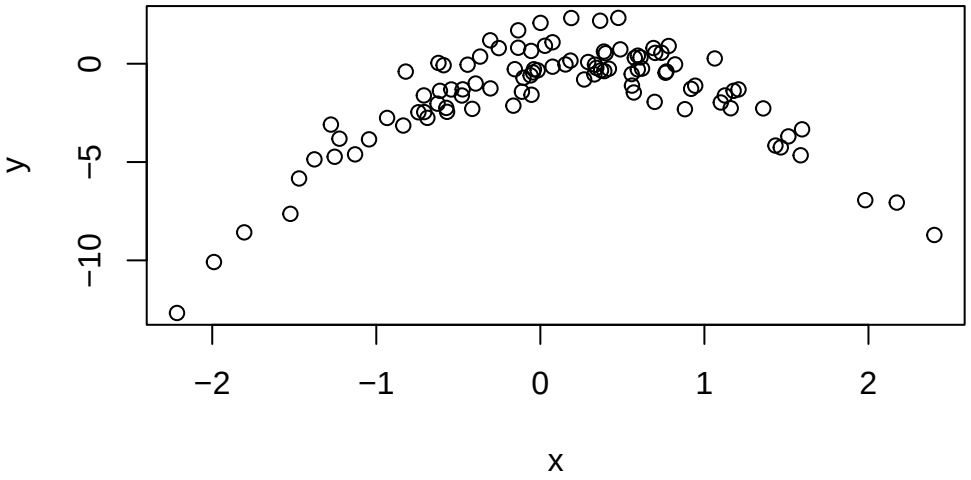
e. `sum(errors) / length(errors)`

```
[1] 0.4435262
```

About 44%. Not really much better than tossing a coin.

3. a. $n = 100, p = 2$. Model equation is $Y = X - 2X^2 + \epsilon$.

b.
```
set.seed(1)
x <- rnorm(100) # mean 0 and sd 1
y <- x - 2 * x^2 + rnorm(100)
plot(x, y)
```



The plot is quadratic. X from about -2 to 2. Y from about -10 to 2.

c.
```
myData <- data.frame(Y = y, X1 = x, X2 = x^2, X3 = x^3, X4 = x^4)

set.seed(1)
fit <- lm(Y ~ X1, data = myData)
fit.hat <- hat(myData$X1)
errors[1] <- mean(((myData$Y - fit$fitted.values) / (1 - fit.hat))^2)
# LOOCV shortcut for linear models
# summary(fit)

fit <- lm(Y ~ X1 + X2, data = myData)
fit.hat <- hat(cbind(myData$X1, myData$X2))
errors[2] <- mean(((myData$Y - fit$fitted.values) / (1 - fit.hat))^2)
# summary(fit)
```

```
fit <- lm(Y ~ X1 + X2 + X3, data = myData)
fit.hat <- hat(cbind(myData$X1, myData$X2, myData$X3))
errors[3] <- mean(((myData$Y - fit$fitted.values) / (1 - fit.hat))^2)
# summary(fit)

fit <- lm(Y ~ X1 + X2 + X3 + X4, data = myData)
fit.hat <- hat(cbind(myData$X1, myData$X2, myData$X3, myData$X4))
errors[4] <- mean(((myData$Y - fit$fitted.values) / (1 - fit.hat))^2)
# summary(fit)
```

d. (Regardless of the seed chosen) LOOCV returned the same results. This method, unlike validation set, is not random, and the method of fitting is not random either.

e. The second model with X^1 and X^2 as predictors. This is what was expected, since that is the model used to generate the data in the first place.

f. Use `summary(fit)` with the previous commands to quickly check coefficient estimate significance. The conclusions drawn by checking the parameter estimates' significance is the same as that of LOOCV. For the underfitted model 1, the parameter is not significant. For the overfitted models 3 and 4, the extra parameters (X^3, X^4) are not significant, suggesting that $Y = X + X^2$ is the best model to use.

4. a.
```
library(ISLR2)
set.seed(1)
train.set <- sample(length(College[, 1]), length(College[, 1]) / 2)
train <- (seq(1, length(College[, 1])) %in% train.set)
```

b.
```
fit <- lm(Apps ~ ., data = College, subset = train)
pred <- predict(fit, newdata = College[!train, ])
mean(fit$residuals^2) # training error
```

```
[1] 1118449
```

```
mean((pred - College[!train, ]$Apps)^2) # test error
```

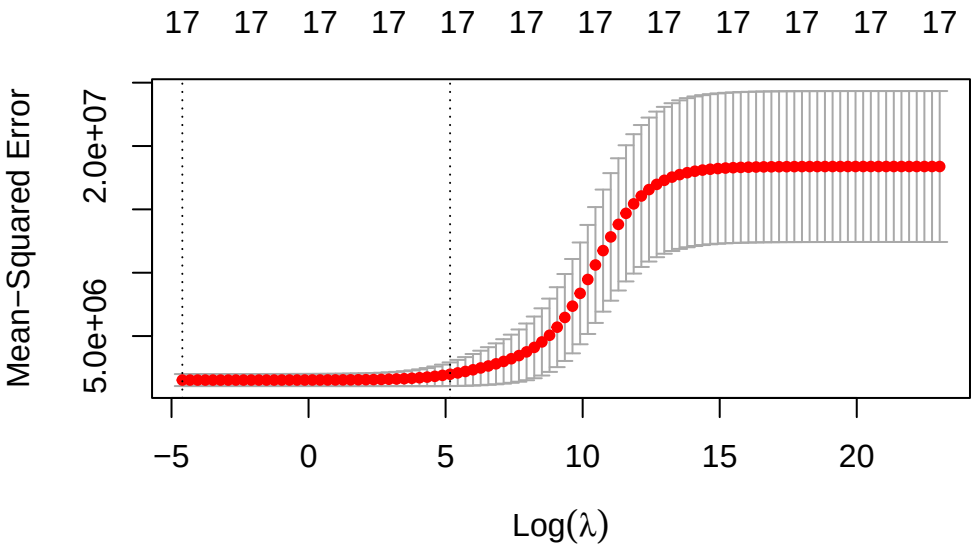
```
[1] 1135758
```

c.
```
library(glmnet)
```

```
Loading required package: Matrix
```

```
Loaded glmnet 4.1-8
```

```
lambda.grid <- 10^seq(10, -2, length.out = 100)
set.seed(1)
College.modelmatrix <- model.matrix(Apps ~ ., College)[, -1]
fit2 <- glmnet(College.modelmatrix[train, ], as.matrix(College[train, 2]),
  alpha = 0, lambda = lambda.grid
)
cv.fit2 <- cv.glmnet(College.modelmatrix[train, ],
  as.matrix(College[train, 2]),
  alpha = 0, lambda = lambda.grid
)
plot(cv.fit2)
```

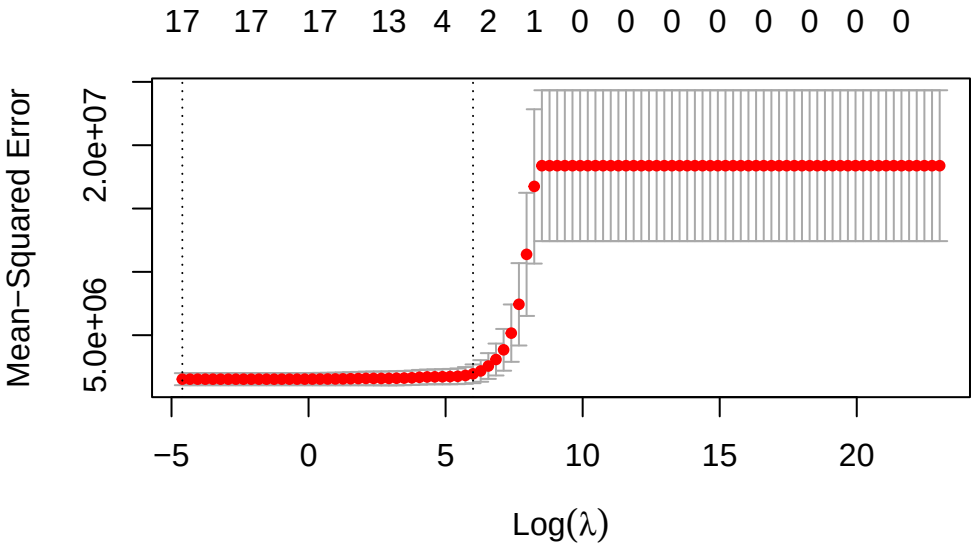



```
mean((College[!train, "Apps"] - predict(fit2,
  s = cv.fit2$lambda.min,
  newx <- College.modelmatrix[!train, ]
))^2)
```

```
[1] 1134677
```

The test mean-squared error is 1.1346768×10^6

d.
```
set.seed(1)
fit2 <- glmnet(College.modelmatrix[train, ], as.matrix(College[train, 2]),
  alpha = 1, lambda = lambda.grid
)
cv.fit2 <- cv.glmnet(College.modelmatrix[train, ], as.matrix(College[train, 2]),
  alpha = 1, lambda = lambda.grid
)
plot(cv.fit2)
```




```
predict(fit2, type = "coefficients", s = cv.fit2$lambda.min)
```

```
18 x 1 sparse Matrix of class "dgCMatrix"
                        s1
(Intercept) -7.931498e+02
PrivateYes  -3.078903e+02
Accept       1.777242e+00
Enroll      -1.450532e+00
Top10perc    6.659456e+01
Top25perc   -2.221506e+01
F.Undergrad  8.983869e-02
P.Undergrad  1.005260e-02
Outstate    -1.082871e-01
Room.Board   2.118762e-01
Books        2.922508e-01
Personal     6.234085e-03
PhD         -1.542914e+01
Terminal     6.364841e+00
S.F.Ratio    2.284667e+01
perc.alumni  1.114025e+00
Expend       4.861825e-02
Grad.Rate    7.466015e+00
```

```
mean((College[!train, "Apps"] - predict(fit2,
  s = cv.fit2$lambda.min,
  newx <- College.modelmatrix[!train, ]
))^2)
```

```
[1] 1133422
```

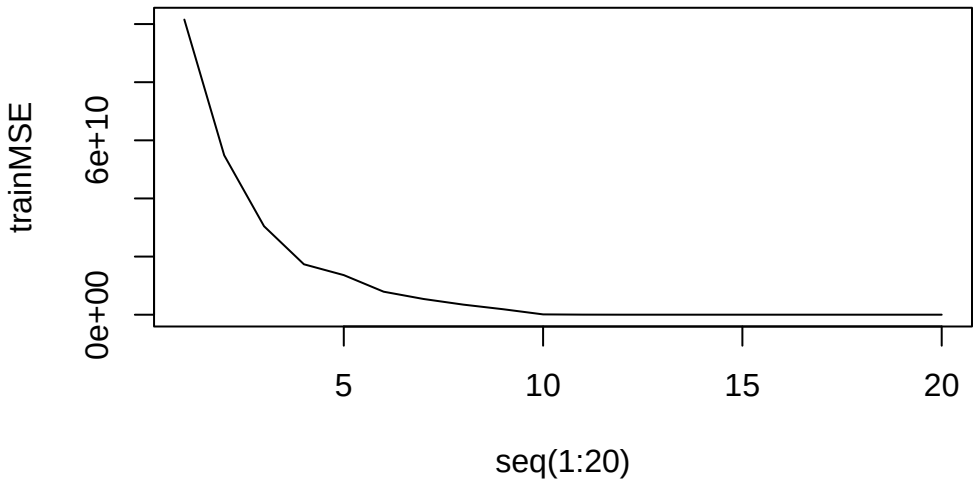
There is a different fit here. The MSE is now 1.1334221×10^6.

e. The MSE in the case of Lasso seems to have done the best, although almost all models have achieved relatively close MSE values.

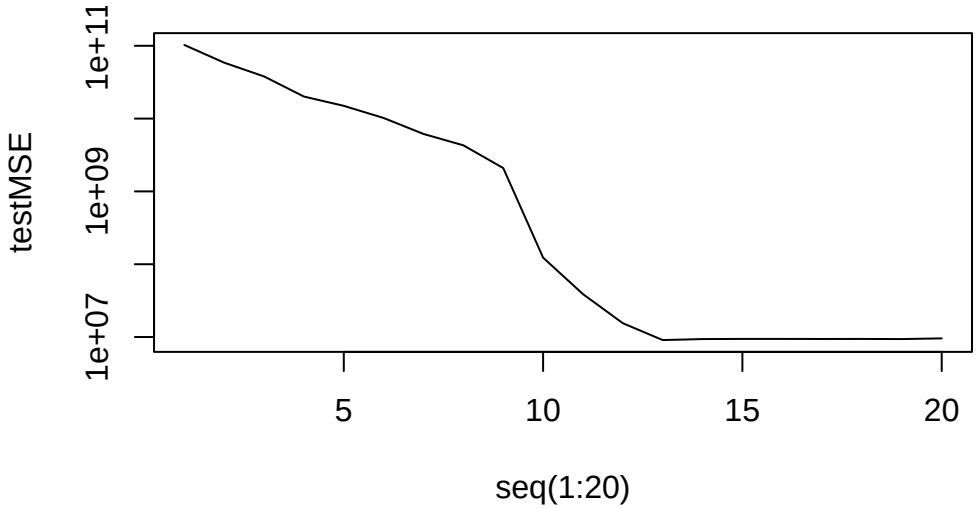
5. a.
```r
set.seed(1)
myData <- matrix(nrow = 1000, ncol = 20)
myMeans <- runif(20, min = 100, max = 1000)
myVarScaling <- runif(20, min = 1, max = 4)
trueCoef <- rep(0, 20)
for (i in 1:20) {
  myData[, i] <- rnorm(1000, mean = myMeans[i], sd = myMeans[i] * myVarScaling[i])
  if (rnorm(1) < 0) {
    trueCoef[i] <- runif(1, min = myMeans[i] * 0.01, max = myMeans[i] * 0.2)
  }
}
trueConst <- 5000
noise <- rnorm(1000, 0, max(myMeans * myVarScaling))
Y <- myData %*% trueCoef + noise + trueConst
myData <- data.frame(Y = Y, myData)
```

b.
```r
train.set <- sample(1000, 100)
train <- (seq(1, 1000) %in% train.set)
```

c.
```r
library(leaps)
fit3 <- regsubsets(x = myData[, -1], y = Y, nvmax = 20)
fit3.sum <- summary(fit3)
trainMSE <- rep(0, 20)
for (i in 1:20) {
  currfit <- lm(Y ~ ., data = myData[train, fit3.sum$which[i, ]])
  trainMSE[i] <- mean((Y[train] - currfit$fitted.values)^2)
}
plot(seq(1:20), trainMSE, type = "l")
```



d.
```
testMSE <- rep(0, 20)
for (i in 1:20) {
  currfit <- lm(Y ~ ., data = myData[train, fit3.sum$which[i, ]])
  testMSE[i] <- mean((Y[!train] - predict(currfit, newdata = myData[!train, ]))^2)
}
plot(seq(1:20), testMSE, type = "l", log = "y")
```



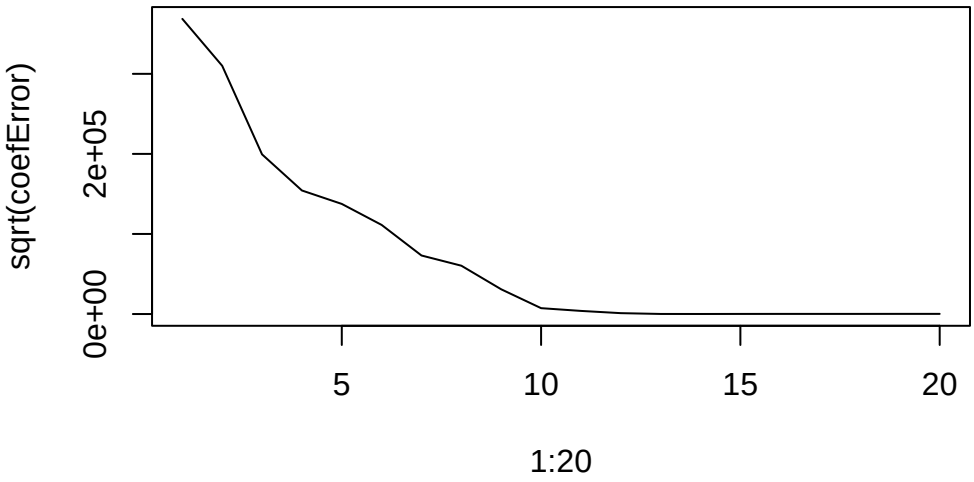
e.
```
coef(fit3, 13)
```

```
  (Intercept)            X1            X2            X3            X4            X5
 5091.100465    53.681512    85.890521    70.809849   121.863743     8.966963
           X8           X11           X12           X14           X15           X16
   29.818868     3.412561    12.050038    67.342103    75.577635    41.301243
          X17           X19
  107.697093    33.601406
```

In our case, the test MSE is minimised at when the model has 14 predictors.

f. The model's estimates are all very close to the true coefficients in the vector `trueCoef`.

g.
```
coefError <- rep(0, 20)
for (i in 1:20) {
  currNum <- 1
  coefError[i] <- (trueConst - coef(fit3, i)[currNum])^2
  for (j in 2:21) {
    if (fit3.sum$which[i, j]) {
      currNum <- currNum + 1
      coefError[i] <- coefError[i] + (trueCoef[j - 1]
      - coef(fit3, i)[currNum])^2
    } else {
      coefError[i] <- coefError[i] + trueCoef[j - 1]^2
    }
  }
}
plot(1:20, sqrt(coefError), type = "l")
```



We attain a similar (but slightly different) result here in that this error is minimised when there are 14 parameters within the model (as opposed to 13 which minimised the test MSE). However, we can see that the extra parameter added `X13` has a small coefficient value.