Hw10

Manuel Alejandro Garcia Acosta

11/20/2019

For this exercise I'll use the 'MASS' package.

```
library(MASS)
setwd('/home/noble_mannu/Documents/PhD/First/STAT_2131_Applied_Statistical_Methods_I/HW10')
```

Exercise 3

First I read the data from 'Fat.txt'. After that, I construct the data frames for the training data and the test data.

```
# Read the data
Data <- read.table('Fat.txt', header = TRUE)

# Creating the training data frame by removing the tenths observations
tenths <- seq(from = 10, to = 252, by =10)
Data_1 <- Data[-tenths, ]

# Create the test data frame only with the tenths observations
Data_2 <- Data[tenths, ]
```

In part (i) and (ii) I ran both linear regression and ridge regression for the training data, where 'siri' is the response and all other variables are predictors.

(i) A simple linear model

For this part I just fitted a linear model.

```
# Create the linear model
linMod <- lm(siri ~ ., data = Data_1)
# Display the summary of the linear model
summary(linMod)
```

```
##
## Call:
## lm(formula = siri ~ ., data = Data_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8314 -0.6722  0.1828  0.9150  6.6619
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -12.591885   6.448868  -1.953 0.052193 .
## age           0.007978   0.012320   0.648 0.517983
## weight        0.362999   0.023314  15.570  < 2e-16 ***
## height        0.049026   0.040315   1.216 0.225315
```

```
## adipos       -0.514032   0.114074  -4.506 1.09e-05 ***
## free         -0.564773   0.014889 -37.933  < 2e-16 ***
## neck          0.016525   0.089863   0.184 0.854272
## chest         0.120219   0.039590   3.037 0.002694 **
## abdom         0.140108   0.042186   3.321 0.001056 **
## hip           0.006197   0.056101   0.110 0.912148
## thigh         0.195057   0.054460   3.582 0.000424 ***
## knee          0.106637   0.093534   1.140 0.255542
## ankle         0.125118   0.081303   1.539 0.125325
## biceps        0.096199   0.064656   1.488 0.138278
## forearm       0.230775   0.073332   3.147 0.001888 **
## wrist         0.139279   0.206804   0.673 0.501378
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.55 on 211 degrees of freedom
## Multiple R-squared:  0.9692, Adjusted R-squared:  0.967
## F-statistic: 442.5 on 15 and 211 DF,  p-value: < 2.2e-16
```

I also created the design matrix separately.

```
# Create X matrix including intercept for the training data
intercept <- rep(1, length(Data_1$siri))
X <- cbind(intercept, data.matrix(subset(Data_1, select = -c(siri))))
Y <- Data_1$siri
```

(ii) Ridge regression

Here I fitted a model using ridge regression. The tuning parameter λ was chosen using Generalized Cross Validation (GCV).

Choosing the right vector of possible values for λ was a little tricky. However, after some attempts I came up with this one that gets the job done.

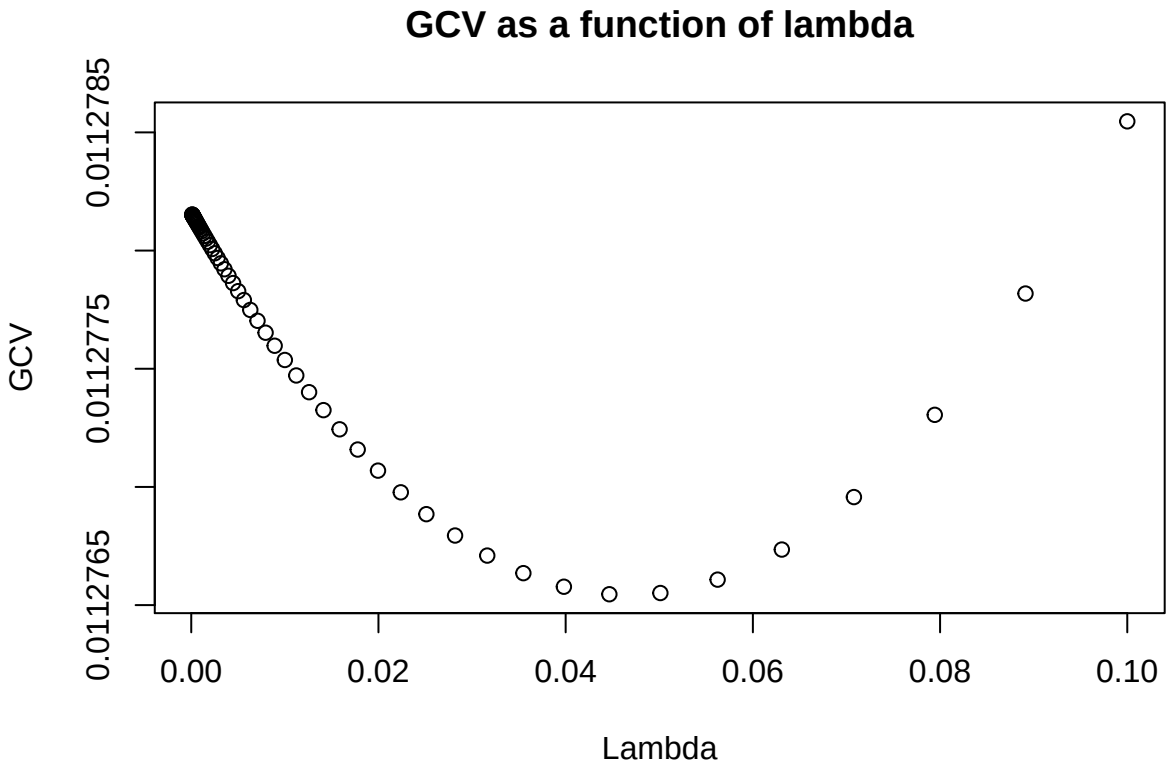
```
# Create vector of lambdas
lambda_seq <- 10^seq(-1, -4, by = -.05)
```

I used the vector defined above to run ridge regression.

```
# Run ridge regression
fit.ridge <- lm.ridge(siri~ ., data=Data_1, lambda = lambda_seq)
# Save the fitted coefficients for each value of lambda on a matrix
all.coefficients <- data.matrix(coef(fit.ridge))
```

After running the model I needed to choose the best λ. Such λ will minimize the GCV. Here is the plot of GCV as a function of λ, this gives us a good idea of which value of λ to pick.

```
# Here I plot GCV as a function of lambda
plot( lambda_seq, fit.ridge$GCV, xlab = 'Lambda', ylab = 'GCV',
      main = 'GCV as a function of lambda')
```



Since the object 'fit.ridge$GCV' contains the GCV results for each value of lambda, it was enough to look for the smallest GCV and from there retrieve the value of λ.

```
# Choosing the best lambda based on GCV
index <- which.min(fit.ridge$GCV)
# Which GCV was the minimum
min_gcv <- fit.ridge$GCV[index]
# Which lambda was the best
best_lambda <- as.numeric(names(fit.ridge$GCV[index]))

best_lambda
```

```
## [1] 0.04466836
```

Now we know that the best value for lambda was $\lambda = 0.04466836$.

(a) Training error

In this part I computed the training error for both models (linear and ridge).

Linear model

First, for the linear model we have the following.

```
# Compute the training error for the linear model
y.lin <- Data_1$siri - linMod$fitted.values
n <- length(Data_1$siri)
err.lin <- 1/n * y.lin%*%y.lin
# Printing the training error for the linear model
err.lin
```

```
##           [,1]
## [1,] 2.232976
```

The training error for the linear model was $\overline{err}_{(lin)} = 2.232976$.

Ridge regression model

Second, I computed the training error for the ridge regression model.

```
# Compute the fitted values for the best lambda (ridge regression)
Y.hat.lambda <- as.vector(X%*%all.coefficients[index,])
# Compute the training error for the ridge regression model
y.ridge <- Data_1$siri - Y.hat.lambda
err.ridge <- 1/n * y.ridge%*%y.ridge
# Printing the training error for the ridge model
err.ridge
```

```
##           [,1]
## [1,] 2.233271
```

The training error for the ridge model was $\overline{err}_{(ridge)} = 2.233271$. Therefore the training error for the linear model was smaller than the training error for the ridge model.

```
# We get that the error from the linear model is smaller than the error using ridge
err.lin < err.ridge
```

```
##       [,1]
## [1,] TRUE
```

The training error is a poor judge of how well the model will predict future data since as we proved in exercise 1 the training error usually underestimates the test (or prediction) error. Moreover, if we try to make the training error in a specific model really small then more often than not our model will be almost useless since the test error when we are predicting values will be high. It's kind of overfitting our training model and then finding out that it doesn't work for prediction.

(b) Test error

In this last section I use the models from part (i) and (ii) to predict the held out data -that is, the data in 'Data_2'-. The metric (loss function) used for this purpose was

$$LF = \frac{1}{n}\sum_{i=1}^{25}\left[y_i - \hat{f}(x_i)\right]^2$$

Where $i \in \{1, ..., 25\}$ because our test data consist of 25 observations.

Next, I define the design matrix and the response I'll use for computing the loss function.

```
# Create X matrix including intercept for the test data
intercept.test <- rep(1, length(Data_2$siri))
X.test <- cbind(intercept.test, data.matrix(subset(Data_2, select = -c(siri))))
Y.test <- Data_2$siri
# Define n = # of observations. I use the same for both models
n.test <- length(Data_2$siri)
```

Linear model

In this section I compute the loss function for the linear regression model.

```
# Using the linear model to predict the held out data
lin.coef <- coef(linMod)
fitted.Y.linear <- as.vector(X.test%*%lin.coef)
y.lin.test <- Data_2$siri - fitted.Y.linear
loss.lin <- 1/n.test * y.lin.test%*%y.lin.test
loss.lin
```

```
##          [,1]
## [1,] 1.280357
```

Here we have that the loss function for the linear model is $LF_{(lin)} = 1.280357$.

Ridge regression model

In this section I compute the loss function for the ridge regression model.

```
# Using the ridge model to predict the held out data
fitted.Y.ridge <- as.vector(X.test%*%all.coefficients[index,])
y.ridge.test <- Data_2$siri - fitted.Y.ridge
loss.ridge <- 1/n.test * y.ridge.test%*%y.ridge.test
loss.ridge
```

```
##          [,1]
## [1,] 1.272906
```

Here we have that the loss function for the ridge model is $LF_{(ridge)} = 1.272906$. Finally, we conclude finally that the ridge model performed better than the linear regression model at predicting the test data.

```
# Now ridge does a better job at predicting the data
loss.ridge < loss.lin
```

```
##      [,1]
## [1,] TRUE
```