Homework 3: Linear Model Selection and Regularization

Wen Li Teng

February 8 2020

Conceptual Exercises

```r
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 3.6.2

## Loading required package: tidyr
```

```r
library(tidyverse)
```

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

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v readr   1.3.1     v stringr 1.4.0
## v ggplot2 3.2.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.6.2
```

```r
library(ggplot2)

set.seed(1234)
```

1. Generate a data set and an associated quantitative response variable

```r
n_obs <- 1000
min <- -1
max <- 1

for (i in 1:20) {
  feature <- runif(n_obs, min, max)
  assign(paste0("x", i), feature)
}

random_beta <- sample(1:20, 5, replace = F)
random_beta
```

```
## [1]  5 17  4 14  9
```

```
beta <- runif(20)
for (i in 1:5) {
    beta[random_beta[i]] <- 0
}

error <- runif(1000)
q1_data <- cbind(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10,
                 x11, x12, x13, x14, x15, x16, x17, x18, x19, x20)
y <- as.vector(q1_data %*% beta + error)
q1_data <- cbind(q1_data, error, y)
q1_data <- as.data.frame(q1_data)
```

2. Split data into a training set and a test set

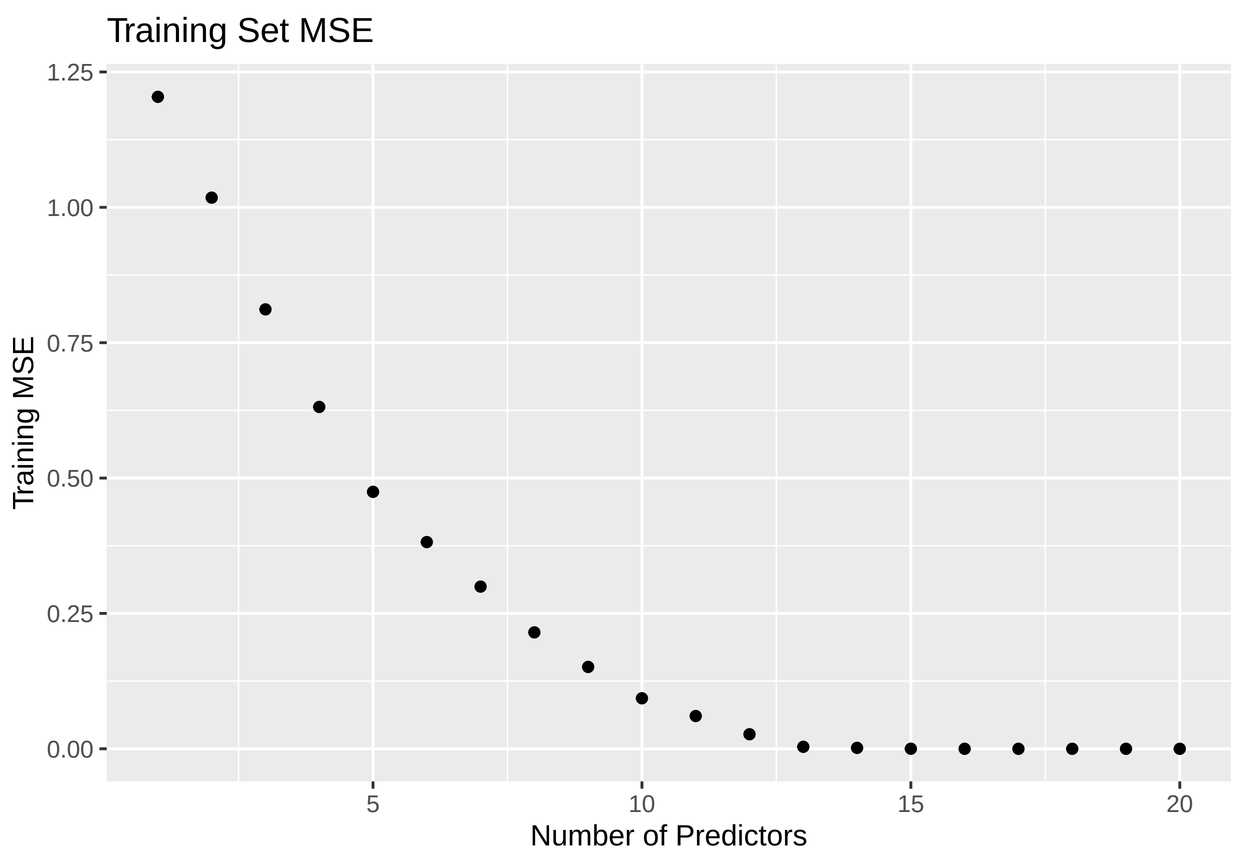
```
split <- initial_split(q1_data, prop = 0.1)
train <- training(split)
test <- testing(split)
```

3. Perform best subset selection on training set and plot the training set MSE

```
q1_select <- regsubsets(y ~ ., data = train, nvmax = 20)
q1_matrix_train <- model.matrix(y ~ ., data = train, nvmax = 20)
na_vector_1 <- rep(NA, 20)
for (i in 1:20) {
  coefi <- coef(q1_select, id = i)
  pred <- q1_matrix_train[, names(coefi)] %*% coefi
  na_vector_1[i] <- mean((pred - train$y)^2)
}

training_mse <- as.data.frame(cbind(1:20, na_vector_1))
training_mse <- training_mse %>%
  rename(num_pred = V1,
         training_MSE = na_vector_1)

ggplot(training_mse,aes(x = num_pred, y = training_MSE)) +
  geom_point() +
  labs(title = "Training Set MSE",
       x = "Number of Predictors",
       y = "Training MSE")
```



```
min_train_set <- which(training_mse$training_MSE ==
                         min(training_mse$training_MSE))
min_train_set
```

```
## [1] 18
```

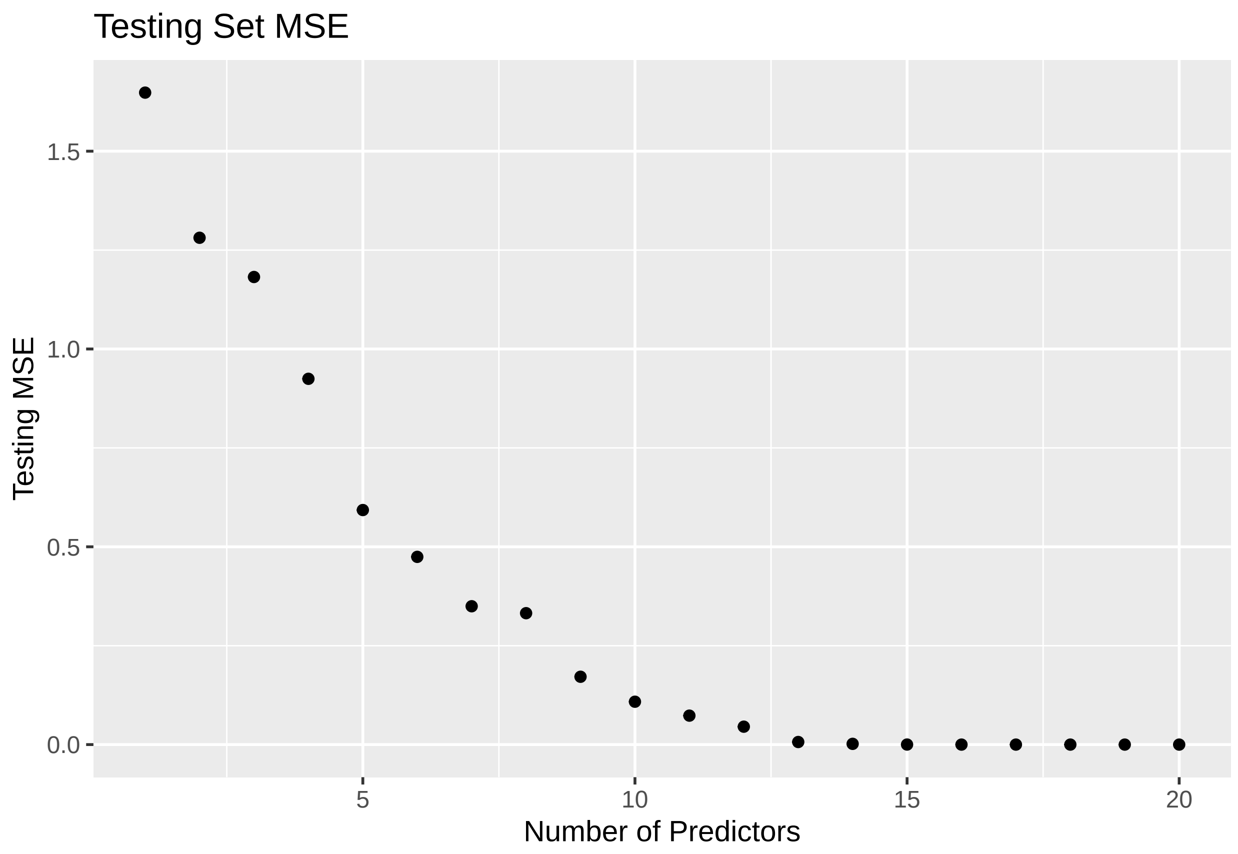
The model of size 18 yields the minimum training set MSE value.

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

```
q1_matrix_test <- model.matrix(y ~ ., data = test, nvmax = 20)
na_vector_2 <- rep(NA, 20)
for (i in 1:20) {
  coefi <- coef(q1_select, id = i)
  pred <- q1_matrix_test[, names(coefi)] %*% coefi
  na_vector_2[i] <- mean((pred - test$y)^2)
}

testing_mse <- as.data.frame(cbind(1:20, na_vector_2))
testing_mse <- testing_mse %>%
  rename(num_pred = V1,
         testing_MSE = na_vector_2)
```

```r
ggplot(testing_mse,aes(x = num_pred, y = testing_MSE)) +
  geom_point() +
  labs(title = "Testing Set MSE",
       x = "Number of Predictors",
       y = "Testing MSE")
```



5. Model for minimum value of test set MSE

```r
min_value <- which.min(testing_mse$testing_MSE)
min_value
```

```
## [1] 16
```

The model with 16 variables has the smallest test set MSE. According to James et al. (2013), if there are a set of models that appear to be more or less equally good, one should choose the simplest model - the one with the fewest predictors. From the plot, it appears that the models from 16 to 20 onwards have similar MSEs. Therefore, one should choose the model with 16 variables.

6. Compare model for minimum value of test set MSE to true model by comparing coefficient sizes

```r
model_coef <- coef(q1_select, min_value)
model_coef
```

```
##   (Intercept)            x1            x2            x3            x6
## -5.568462e-16  5.496697e-01  9.274793e-01  2.743850e-01  1.690863e-02
##            x7            x8           x10           x11           x12
##  1.101205e-01  4.468726e-01  3.344711e-01  9.867715e-01  5.349801e-01
##           x13           x15           x16           x18           x19
##  6.762125e-02  5.335923e-01  8.239707e-01  8.336309e-01  3.110404e-01
##           x20         error
##  4.972922e-01  1.000000e+00
```

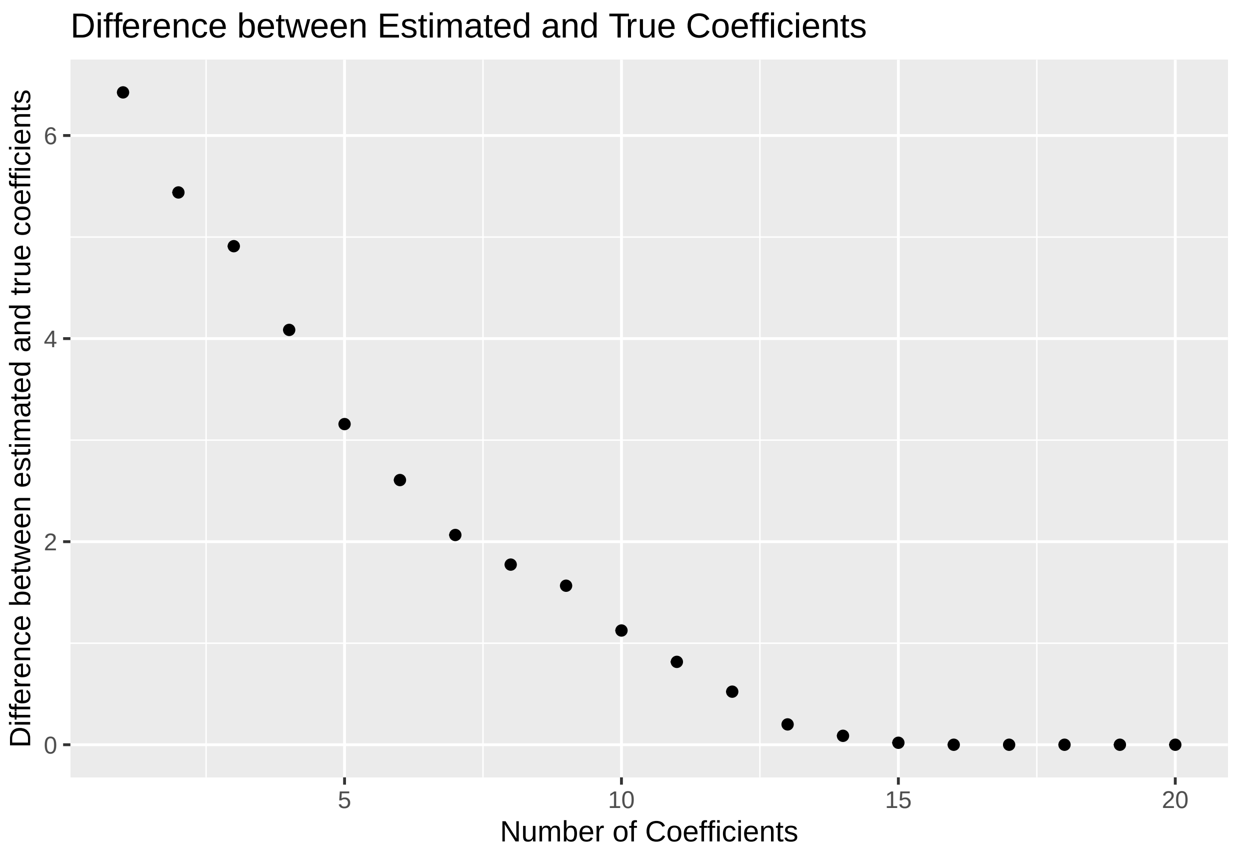
There are only 15 x variables listed among the coefficients. x4, x5, x9, x14, and x17 are missing. These are the variables for which a beta of 0 was specified (by the random_beta vector). Therefore, the 16-variable model is indeed a "good" model that balances the number of predictors and MSE.

7. Create plot displaying function and compare to test MSE plot

```r
na_vector_3 <- rep(NA, 20)
x_cols = colnames(q1_data[,1:20])
for (i in 1:20) {
  coefi <- coef(q1_select, id = i)
  na_vector_3[i] <- sqrt(sum((beta[x_cols %in% names(coefi)]
                              - coefi[names(coefi) %in% x_cols])^2)
                         + sum(beta[!(x_cols %in% names(coefi))])^2)
}

est_v_true <- as.data.frame(cbind(1:20, na_vector_3))
est_v_true <- est_v_true %>%
  rename(num_coef = V1,
         error_diff = na_vector_3)

ggplot(est_v_true,aes(x = num_coef, y = error_diff)) +
  geom_point() +
  labs(title = "Difference between Estimated and True Coefficients",
       x = "Number of Coefficients",
       y = "Difference between estimated and true coefficients")
```



For this particular set of data (uniform distribution), the shape of the test set MSE curve is similar to the curve of the difference between estimated and true coefficients. This suggests that the model with 16 variables is likely to have both the lowest test MSE as well as a smallest difference between estimated and true coefficients.

Application Exercises

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.2
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 3.0-2
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.2

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
train <- read.csv("data/gss_train.csv")
test <- read.csv("data/gss_test.csv")
```

1. Fit a least squares linear model on training set

```r
linear_mod <- lm(egalit_scale ~ ., data = train)
linear_pred <- predict(linear_mod, test)
linear_err <- mean((linear_pred - test$egalit_scale)^2)
linear_err
```

```
## [1] 63.21363
```

2. Fit a ridge regression model on training set

```r
train_matrix <- model.matrix(egalit_scale ~ ., data = train)
test_matrix <- model.matrix(egalit_scale ~ ., data = test)

lambda_vals <- 10 ^ seq(4, -2, length = 100)

ridge_fit <- glmnet(train_matrix, train$egalit_scale, alpha = 0, lambda = lambda_vals, nfolds = 10)
ridge_cv <- cv.glmnet(train_matrix, train$egalit_scale, alpha = 0, lambda = lambda_vals, nfolds = 10)
ridge_bestlam <- ridge_cv$lambda.min
ridge_pred <- predict(ridge_fit, s = ridge_bestlam, newx = test_matrix)
ridge_err <- mean((ridge_pred - test$egalit_scale)^2)
ridge_err
```

```
## [1] 61.01179
```

3. Fit a lasso regression on training set

```
lasso_fit <- glmnet(train_matrix, train$egalit_scale, alpha = 1, lambda = lambda_vals, nfolds = 10)
lasso_cv <- cv.glmnet(train_matrix, train$egalit_scale, alpha = 1, lambda = lambda_vals, nfolds = 10)
lasso_bestlam <- lasso_cv$lambda.min
lasso_pred <- predict(lasso_fit, s = lasso_bestlam, newx = test_matrix)
lasso_err <- mean((lasso_pred - test$egalit_scale)^2)
lasso_err
```

```
## [1] 61.31515
```

4. Fit an elastic net regression model

```
alpha_vals <- seq(0, 1, 0.1)
min_lambda_vals <- vector(mode = "numeric", length = 11)
min_mse_vals <- vector(mode = "numeric", length = 11)

for (i in 0:10) {
  enet_cv <- cv.glmnet(train_matrix,
                       train$egalit_scale,
                       alpha = i*0.1,
                       lambda = lambda_vals,
                       nfolds = 10)
  min_lambda_vals[(i+1)] <- enet_cv$lambda.min
  min_mse <- enet_cv$cvm[enet_cv$lambda == enet_cv$lambda.min]
  min_mse_vals[(i+1)] <- min_mse
  remove(enet_cv, min_mse)
}

enet_table <- as.data.frame(cbind(alpha_vals, min_lambda_vals, min_mse_vals))
rownum <- which(enet_table$min_mse_vals == min(enet_table$min_mse_vals))

enet_fit <- glmnet(train_matrix,
                   train$egalit_scale,
                   alpha = enet_table$alpha_vals[rownum],
                   lambda = enet_table$min_lambda_vals[rownum],
                   nfolds = 10)

enet_bestlam <- enet_table$min_lambda_vals[rownum]
enet_pred <- predict(enet_fit, s = enet_bestlam, newx = test_matrix)
enet_err <- mean((enet_pred - test$egalit_scale)^2)
enet_err
```

```
## [1] 61.22729
```

```
non_zero <- coef(enet_fit, s=enet_fit$lambda.1se)
non_zero <- non_zero[which(non_zero != 0 ) ]
length(non_zero)
```

```
## [1] 30
```

5. Comment on results

```
methods <- c("least squares linear", "ridge", "lasso", "elastic net")
errors <- c(linear_err, ridge_err, lasso_err, enet_err)
q2_results <- as.data.frame(cbind(methods, errors), stringsAsFactors = F)

avg_egalit <- mean(test$egalit_scale)
avg_egalit_calc <- mean((avg_egalit - test$egalit_scale)^2)
linear_r2 <- 1 - linear_err/avg_egalit_calc

rsquared <- vector(mode = "numeric", length = 4)
for (i in 1:4){
  rsquared[i] <- 1 - as.numeric(q2_results$errors[[i]])/avg_egalit_calc
}

q2_results <- cbind(q2_results, rsquared)
q2_results <- q2_results[order(rsquared), ]
q2_results
```

```
##                methods           errors  rsquared
## 1 least squares linear 63.2136296230151 0.3004585
## 3                lasso  61.315154054766 0.3214677
## 4          elastic net 61.2272865455112 0.3224400
## 2                ridge 61.0117917456826 0.3248248
```

There is not much difference between the test errors resulting from these approaches. Given that the r-squared values are all around 0.3, the models do not predict an individual's egalitarianism accurately.