Statistical Modeling Course

Variable Selection Assignment

In this exercise, we will predict the number of applications, `Apps`, received using the other variables in the `College` data set. Use 5-fold cross-validation to calculate the test error using the following methods. You should fit each model five times leaving out one fifth of the data each time and calculating the test error (RMSE) on the data you left out. The total test error will be the average RMSE from the five folds. You should also calculate the standard error of the test errors, `sd(test_error)/sqrt(5)`. For best subset selection, and forward stepwise selection chose the model using BIC. For lasso, ridge, and pcr you will conduct a second level of cross validation for each of the 5 training sets. You can do this with `cv.glmnet`.

Compare your results with the null model. Code for the null model and splits is given below, make sure to compare all of your models on the same splits of the data.

```r
set.seed(2020)
# Vector of data split
folds <- sample(1:5, nrow(College), replace = TRUE)

# Variable to store error from each fold
error_null <- c()

# Loop through folds and fit models with no predictors
for (i in 1:5){
  test <- College[folds == i, ]
  train <- College[folds != i, ]
  error_null[i] <- sqrt(mean((test$Apps - mean(train$Apps))^2))
}
```

Problem 1

- Least squares linear model
- Best subset selection (chosen using BIC)
- Forward stepwise subset selection (chosen using BIC)

```r
## Least squares model

total_error <- data.frame(Model = character(0), MeanError = integer(0))
```

```r
# Create 5 equally size folds
folds <- cut(seq(1, nrow(College)), breaks = 5, labels = FALSE)
error_ls <- c()
error_best <- c()
error_fwd <- c()
error_null <- c()
# Perform 5 fold cross validation
for(i in 1:5){
  College = na.omit(College)
  test_indexes <- which(folds==i, arr.ind=TRUE)
  test <- College[test_indexes, ]
  train <- College[-test_indexes, ]
  error_null[i] <- sqrt(mean((test$Apps - mean(train$Apps))^2))

  test.mat=model.matrix(data=College[test_indexes ,], Apps ~ .)

  # Use the test and train data partitions however you desire...
  least_squares <- lm(data = train,
                      Apps ~ .)
  prediction_ls <- predict(least_squares, test)
  error_ls[i] <- sqrt(mean((test$Apps - prediction_ls)^2))

  best_sub <- regsubsets(data = train,
                         Apps ~ .)
  sum_best <- summary(best_sub)
  num <- which.min(sum_best$bic)
  coef_best <- coef(best_sub, num)
  #get prediction
  prediction_best = test.mat[,names(coef_best)]%*%coef_best
  error_best[i] <- sqrt(mean((test$Apps - prediction_best)^2))


  fwd_step <- regsubsets(data = train,
                         Apps ~ .,
                         method = "forward")
  sum_fwd <- summary(fwd_step)
  num <- which.min(sum_fwd$bic)
  coef_fwd <- coef(fwd_step, num)
  #get prediction
  prediction_fwd = test.mat[,names(coef_fwd)]%*%coef_fwd
  error_fwd[i] <- sqrt(mean((test$Apps - prediction_fwd)^2))
}

# Get total error
total_error <- rbind(total_error,
                     data.frame(Model = c("null" , "least_squares", "best_sub", "fwd_step"),
                                MeanError = c(mean(error_null),
                                              mean(error_ls),
                                              mean(error_best),
                                              mean(error_fwd)),
                                SEerror = c(sd(error_null)/sqrt(5),
                                            sd(error_ls)/sqrt(5),
                                            sd(error_best)/sqrt(5),
                                            sd(error_fwd)/sqrt(5))
                                ))

total_error
```

```
##           Model MeanError  SEerror
## 1          null  3815.024 412.8009
## 2 least_squares  1185.452 146.2900
## 3      best_sub  1183.346 141.3992
## 4      fwd_step  1196.435 143.0311
```

Problem 2

- Ridge-regression with lambda chosen by cross-validation, report the five λ's chosen
- Lasso with lambda chosen by cross-validation, report the five λ's chosen

```
lambda_ridge <- c()
lambda_lasso <- c()

error_ridge <- c()
error_lasso <- c()

x_college <- model.matrix(Apps ~ . , College)[,-1]
y_college <- College$Apps
```

```r
# Perform 5 fold cross validation
for(i in 1:5){
  test_indexes <- which(folds==i, arr.ind=TRUE)
  x_test <- x_college[test_indexes, ]
  y_test <- y_college[test_indexes]

  x_train <- x_college[-test_indexes, ]
  y_train <- y_college[-test_indexes]

  # Use the test and train data partitions however you desire...
  ridge <- cv.glmnet(x = x_train, y = y_train,
                     alpha = 0)
  prediction_ridge <- predict(ridge, x_test, s = "lambda.min")
  error_ridge[i] <- sqrt(mean((y_test - prediction_ridge)^2))
  lambda_ridge[i] <- ridge$lambda.min

  lasso <- cv.glmnet(x = x_train, y = y_train,
                     alpha = 1)
  prediction_lasso <- predict(lasso, x_test, s = "lambda.min")
  error_lasso[i] <- sqrt(mean((y_test - prediction_lasso)^2))
  lambda_lasso[i] <- lasso$lambda.min
}

# Get total error
total_error <- rbind(total_error,
                     data.frame(Model = "ridge", MeanError = mean(error_ridge),
                                SEerror = sd(error_ridge)/sqrt(5)))

# Get total error
total_error <- rbind(total_error,
                     data.frame(Model = "lasso", MeanError = mean(error_lasso),
                                SEerror = sd(error_lasso)/sqrt(5)))
total_error
```

```
##           Model MeanError  SEerror
## 1          null  3815.024 412.8009
## 2 least_squares  1185.452 146.2900
## 3      best_sub  1183.346 141.3992
## 4      fwd_step  1196.435 143.0311
## 5          ridge  1255.712 260.0541
## 6          lasso  1181.591 148.7972
```

Problem 3

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

```r
error_pcr <- c()
m_val <- c()

for(i in 1:5){
  College = na.omit(College)
  test_indexes <- which(folds==i, arr.ind=TRUE)
  test <- College[test_indexes, ]
  train <- College[-test_indexes, ]

  pcrmod <- pcr(Apps ~ ., data = train, scale = T,
                validation = "CV")
  prediction_pcr <- predict(pcrmod, test, ncomp = pcrmod$validation$ncomp)
  error_pcr[i] <- sqrt(mean((test$Apps - prediction_pcr)^2))
}

total_error <- rbind(total_error,
                     data.frame(Model = "pcr",
                                MeanError = mean(error_pcr),
                                SEerror = sd(error_pcr)/sqrt(5)))
total_error
```

```
##            Model MeanError  SEerror
## 1           null  3815.024 412.8009
## 2 least_squares  1185.452 146.2900
## 3       best_sub  1183.346 141.3992
## 4       fwd_step  1196.435 143.0311
## 5          ridge  1255.712 260.0541
## 6          lasso  1181.591 148.7972
## 7            pcr  1185.452 146.2900
```

Problem 4

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 the approaches?

All of the models are substantial improvements over the null model that does not include predictors. The choice of model doesn't seem important here with least squares and PCR producing identical results indicating that all of the components were used in the model. The standard error in the RMSEs is greater than any of the differences.

Problem 5

Generate a data set with $p = 100$ features, $n = 300$ observations, and an associated quantitative response vector generated according to the model

$$p_i = logit(x_i^T \beta)$$

$$y_i \sim Bin(p_i, n = 1)$$

Simulate some of the features as categorical and some as numeric. Set most of the values of $\beta_p = 0$ for most but not all p.

- Using your simulated dataset split your dataset into at training set and a test set containing using an 80/20 split.
- Perform lasso and ridge regression on the training set.
- Which model has a lower test error (MSE)? Comment on your results.

```r
set.seed(2019)
n <- 300
p1 <- 9
p2 <- 10

X_cont <- matrix(rnorm(n*p1), ncol=p1, nrow=n)
X_cat <- matrix(rbinom(n*p2, size = 1, prob = .2), ncol=p2, nrow = n)
X <- cbind(1, X_cont, X_cat)

beta <- rep(0, 20)

#pick 4 indicies less than 20
beta[c(11, 12, 15, 19)] <- c(4, -1, .4, 3)
```

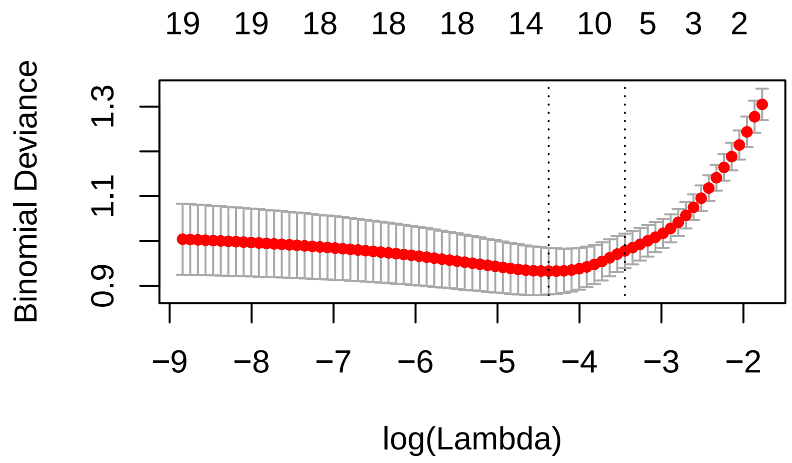
```
eta <- X %*% beta
prob <- exp(eta)/(1+exp(eta))
Y <- rbinom(n, 1, prob = prob)
sim_df <- data.frame(cbind(Y, X_cont, X_cat))
```

```
# rename with sim_df
set.seed(2019)
# Randomly shuffle the data
sim_df <- sim_df[sample(nrow(sim_df)),]
folds <- cut(seq(1, nrow(sim_df)), breaks = 5, labels = FALSE)
test_indexes <- which(folds==i, arr.ind=TRUE)
x_sim <- model.matrix(Y ~ . , sim_df)[,-1]
y_sim <- sim_df$Y

x_test <- x_sim[test_indexes, ]
y_test <- y_sim[test_indexes]

x_train <- x_sim[-test_indexes, ]
y_train <- y_sim[-test_indexes]

lasso <- cv.glmnet(x = x_train, y = y_train,
                   alpha = 1, family = "binomial")
plot(lasso)
```



```
prediction_lasso <- predict(lasso, x_test)
error_lasso[i] <- mean((y_test - prediction_lasso)^2)
lambda_lasso[i] <- lasso$lambda.min

ridge <- cv.glmnet(x = x_train, y = y_train,
```

```r
                   alpha = 0, family = "binomial")
prediction_ridge <- predict(ridge, x_test)
error_ridge[i] <- mean((y_test - prediction_ridge)^2)
lambda_ridge[i] <- ridge$lambda.min

mean(error_lasso)
```

```
## [1] 959.7801
```

```r
mean(error_ridge)
```

```
## [1] 1050.69
```

Lasso has the lower MSE, since many of the parameters have a true value of 0, lasso is able to correctly identify the non-zero variables.