Homework #3: Penalized Regression

Hyunsuk Ko

Due: Wed Sept 21 | 11:45am

DS 6030 | Fall 2022 | University of Virginia

Required R packages and Directories

```
data.dir = 'https://mdporter.github.io/DS6030/data/' # data directory
library(mlbench)
library(glmnet)
library(R6030)     # functions for DS-6030
library(tidyverse) # functions for data manipulation
library(ggpubr)
library(dplyr)
library(GGally)
```

Problem 1: Optimal Tuning Parameters

In cross-validation, we discussed choosing the tuning parameter values that minimized the cross-validation error. Another approach, called the "one-standard error" rule [ISL pg 214, ESL pg 61], uses the values corresponding to the least complex model whose cv error is within one standard error of the best model. The goal of this assignment is to compare these two rules.

Use simulated data from `mlbench.friedman1(n, sd=2)` in the `mlbench` R package to fit *lasso models*. The tuning parameter λ (corresponding to the penalty on the coefficient magnitude) is the one we will focus one. Generate training data, use k-fold cross-validation to get λ_{min} and λ_{1SE}, generate test data, make predictions for the test data, and compare performance of the two rules under a squared error loss using a hypothesis test.

Choose reasonable values for:

- Number of cv folds (K)
  - Note: you are free to use repeated CV, repeated hold-outs, or bootstrapping instead of plain cross-validation; just be sure to describe what do did so it will be easier to follow.
- Number of training and test observations
- Number of simulations
- If everyone uses different values, we will be able to see how the results change over the different settings.
- Don't forget to make your results reproducible (e.g., set seed)

This pseudo code will get you started:

```
library(mlbench)
library(glmnet)

#-- Settings
n.train =        # number of training obs
n.test =         # number of test obs
K =              # number of CV folds
alpha =          # glmnet tuning alpha (1 = lasso, 0 = ridge)
M =              # number of simulations

#-- Data Generating Function
getData <- function(n) mlbench.friedman1(n, sd=2) # data generating function

#-- Simulations
# Set Seed Here

for(m in 1:M) {

# 1. Generate Training Data
# 2. Build Training Models using cross-validation, e.g., cv.glmnet()
# 3. get lambda that minimizes cv error and 1 SE rule
# 4. Generate Test Data
# 5. Predict y values for test data (for each model: min, 1SE)
# 6. Evaluate predictions

}

#-- Compare
# compare performance of the approaches / Statistical Test
```

a. Code for the simulation and performance results

```
library(mlbench)
library(glmnet)
```

```
#-- Data Generating Function
getData <- function(n) mlbench.friedman1(n, sd=2) # data generating function

#-- Settings
n.train = 8000      # number of training obs
n.test = 2000       # number of test obs
K = 10             # number of CV folds
alpha = 0        # glmnet tuning alpha (1 = lasso, 0 = ridge)
M = 100              # number of simulations
```

```
lambda_min <- c()
lambda_1se <- c()
mse_min <- c()
mse_1se <- c()

for(m in 1:M) {
  # 1. Generate Training Data
  train <- getData(n.train)

  # 2. Build Training Models using cross-validation, e.g., cv.glmnet()
  X.train <- train$x
  Y.train <- train$y
  ridge_cv <- cv.glmnet(X.train, Y.train, alpha = alpha, nfolds = K)

  # 3. get lambda that minimizes cv error and 1 SE rule
  lambda.min <- ridge_cv$lambda.min
  lambda.1se <- ridge_cv$lambda.1se

  lambda_min <- append(lambda_min, lambda.min)
  lambda_1se <- append(lambda_1se, lambda.1se)

  # 4. Generate Test Data
  #set.seed(2022)
  test <- getData(n.test)
  X.test <- test$x
  Y.test <- test$y

  # 5. Predict y values for test data (for each model: min, 1SE)
  yhat_min <- predict(ridge_cv, X.test, s = "lambda.min")
  yhat_1se <- predict(ridge_cv, X.test, s = "lambda.1se")

  # 6. Evaluate predictions

  mse_min <- append(mse_min, mean((Y.test - yhat_min) ^ 2) )
  mse_1se <- append(mse_1se, mean((Y.test - yhat_1se) ^ 2) )
}
```

b. Description and results of a hypothesis test comparing $\lambda_{\min}$ and $\lambda_{1\mathrm{SE}}$.

$H_0 : MSE_{\lambda_{\min}} = MSE_{\lambda_{1\mathrm{SE}}}$

$H_a : MSE_{\lambda_{\min}} \neq MSE_{\lambda_{1\mathrm{SE}}}$

Since p-value for paired t.test is less than 0.05, we cannot reject the null hypothesis and conclude that $MSE_{\lambda_{\min}}$ and $MSE_{\lambda_{1\mathrm{SE}}}$ are actually the same.

```
a <- data.frame(lambda_min = lambda_min, mse_min= mse_min, lambda_1se = lambda_1
se ,mse_1se = mse_1se)

mean(a$mse_min)
```

```
#> [1] 9.961
```

```
mean(a$mse_1se)
```

```
#> [1] 10.12
```

```
t.test(a$mse_min, a$mse_1se, paired = TRUE)
```

```
#> 
#>  Paired t-test
#> 
#> data:  a$mse_min and a$mse_1se
#> t = -26, df = 99, p-value <2e-16
#> alternative hypothesis: true mean difference is not equal to 0
#> 95 percent confidence interval:
#>  -0.1708 -0.1465
#> sample estimates:
#> mean difference 
#>         -0.1586
```

Problem 2 Prediction Contest: Real Estate Pricing

This problem uses the realestate-train (https://mdporter.github.io/DS6030/data//realestate-train.csv) and realestate-test (https://mdporter.github.io/DS6030/data//realestate-test.csv) (click on links for data).

The goal of this contest is to predict sale price (in thousands) (`price` column) using an *elastic net* model. Evaluation of the test data will be based on the root mean squared error $\text{RMSE} = \sqrt{\frac{1}{m}\sum_i (y_i - \hat{y}_i)^2}$ for the m test set observations.

a. Load the data and create necessary data structures for running *elastic net*.

- You are free to use any data transformation or feature engineering
- Note: there are some categorical predictors so at the least you will have to convert those to something numeric (e.g., one-hot or dummy coding).

```
train <- read.csv('realestate-train.csv')
test <- read.csv('realestate-test.csv')
```

```
train <- train %>% mutate(PoolArea = ifelse(PoolArea != 0, 1, 0),
                          CentralAir = ifelse(CentralAir == "Y", 1, 0))

test <- test %>% mutate(PoolArea = ifelse(PoolArea != 0, 1, 0),
                        CentralAir = ifelse(CentralAir == "Y", 1, 0))
```

b. Use an *elastic net* model to predict the `price` of the test data.

- You are free to use any data transformation or feature engineering
- You are free to use any tuning parameters
- **Report the α and λ parameters you used to make your final predictions.**
  α = 0.8, λ = 0.6432

- **Describe how you choose those tuning parameters**
  Opimtal λ was chosen based among lambda candidates that gives the best result.

```
X = glmnet::makeX(
  train = train %>% select(-c(price, PoolArea, CentralAir)),
  test = test %>% select(-c(PoolArea, CentralAir))
)

X.train = X$x

Y.train = train$price

X.test = X$xtest
```

```
set.seed(2022)
#--ElasticNet
a = 0.8

#set alpha for elastic net
fit.enet = cv.glmnet(X.train,Y.train, alpha=a, nfolds= 10)
beta.enet = coef(fit.enet, s="lambda.min")
yhat.enet = predict(fit.enet, newx = X.test, s = "lambda.min")
```

```
fit.enet$lambda.min
```

```
#> [1] 0.6432
```

c. Submit a .csv file (ensure comma separated format) named `lastname_firstname.csv` that includes the column named *yhat* that is your estimates. We will use automated evaluation, so the format must be exact.

- You will receive credit for a proper submission; the top five scores will receive 2 bonus points.

```
test <- test %>% mutate(yhat = yhat.enet[,1])

my_yhat <- data.frame(yhat = yhat.enet[,1])
my_yhat
```

	yhat <dbl>
1161	191.824
1162	54.802
1163	196.248
1164	186.037
1165	282.454
1166	245.404

	yhat <dbl>
1167	115.670
1168	259.634
1169	34.221
1170	306.037

1-10 of 300 rows Previous **1** 2 3 4 5 6 ... 30 Next

```
write_csv(my_yhat, 'ko_hyunsuk.csv')
```

d. Report the anticipated performance of your method in terms of RMSE. We will see how close your performance assessment matches the actual value.

: The anticiapted RMSE of the final model (train MSE) is 38.76.

```
set.seed(2022)
#--ElasticNet
a = 0.8

#set alpha for elastic net
fit.enet = cv.glmnet(X.train,Y.train, alpha=a, nfolds= 10)
beta.enet = coef(fit.enet, s="lambda.min")
yhat.enet = predict(fit.enet, newx = X.train, s = "lambda.min")

train <- train %>% mutate(yhat = yhat.enet[,1])
train
```

price <dbl>	PoolArea <dbl>	GarageCars <int>	Fireplaces <int>	TotRmsAbv... <int>	Ba... <int>	SqF... <int>	Central <d
208.50	0	2	0	8	3	1710	
140.00	0	3	1	7	1	1717	
250.00	0	3	1	9	3	2198	
143.00	0	2	0	5	2	1362	
307.00	0	2	1	7	2	1694	
129.90	0	2	2	8	2	1774	
118.00	0	1	2	5	1	1077	
129.50	0	1	0	5	1	1040	
345.00	0	3	2	11	3	2324	
144.00	0	1	0	4	1	912	

1-10 of 1,160 rows | 1-9 of 14 columns Previous **1** 2 3 4 5 6 ... 116 Next

```
r.train = train$price - train$yhat # residuals on test data
mse.train = sqrt(mean(r.train^2))
mse.train
```

```
#> [1] 38.76
```