DS 6030 HW03 Penalized Regression

Ben Wilson

'09/21/2022

Contents

```r
knitr::opts_chunk$set(echo = TRUE)
data.dir = 'https://mdporter.github.io/DS6030/data/' # data directory
library(mlbench)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-4
```

```r
library(R6030)     # functions for DS-6030
library(tidyverse) # functions for data manipulation
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x tidyr::unpack() masks Matrix::unpack()
```

Problem 1: Optimal Tuning Parameters

1a. Code for the simulation and performance results

```r
#-- Settings
n.train = 100       # number of training obs
n.test = 10000        # number of test obs
K = 10            # number of CV folds
alpha = 0.1         # glmnet tuning alpha (1 = lasso, 0 = ridge)
M = 200             # number of simulations
n.folds = 10

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

#capture lambda valyes
lambda_values = tibble()
MSE_values = tibble()

#-- Simulations
# Set Seed Here
set.seed(200)

for(m in 1:M) {

# 1. Generate Training Data
  data_train <- getData(n.train)

  X.train = data_train$x
  Y.train = data_train$y


# 2. Build Training Models using cross-validation, e.g., cv.glmnet()
  #ols
  lasso_cv = cv.glmnet(X.train, Y.train, alpha = alpha, nfolds = K)


 # 3. get lambda that minimizes cv error and 1 SE rule
    #Ridge
  min_lambda = lasso_cv$lambda.min
  se_lambda = lasso_cv$lambda.1se


  # 4. Generate test data
  data_test <- getData(n.test)
  X.test = data_test$x
  Y.test = data_train$y

  # 5. Predict y values for test data (for each model: min, 1SE)

  yhat_min = predict(lasso_cv, X.test, s = "lambda.min")
  yhat_1se = predict(lasso_cv, X.test, s = "lambda.1se")
  # 6. evaluate predictions
  MSE_min = mean((Y.test - yhat_min)^2)
  MSE_lse = mean((Y.test - yhat_lse)^2)
  MSE_values = rbind(MSE_values, c(MSE_min, MSE_lse))

}
```

1b. Description and results of a hypothesis test comparing lambda min and lambda 1SE.

```
names(lambda_values)[1] <- "min"
```

```
## Warning: The `value` argument of `names<-` must have the same length as `x` as of tibble 3.0.0.
## `names` must have length 0, not 1.
```

```
names(MSE_values)[1] <- "min"
names(MSE_values)[2] <- "SE1"

colMeans(MSE_values)
```

```
##      min      SE1
## 43.41062 35.73048
```

```
t.test(MSE_values$min, MSE_values$SE1, paired = TRUE, alternative = "two.sided")
```

```
##
##  Paired t-test
##
## data:  MSE_values$min and MSE_values$SE1
## t = 69.95, df = 199, p-value < 2.2e-16
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  7.463625 7.896646
## sample estimates:
## mean difference
##        7.680136
```

The results for the test above show a low value returned which allows us to reject the null and conclude that the methods are significant.

Problem 2 Prediction Contest: Real Estate Pricing

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

```
#load data
realestate_test <- read.csv("C:\\Users\\brwil\\Desktop\\SY MSDS\\DS 6030 Stat Learning\\Week 4\\realest

realestate_train <- read.csv("C:\\Users\\brwil\\Desktop\\SY MSDS\\DS 6030 Stat Learning\\Week 4\\reales
```

Convert categorical variables to numeric & create null price column in test data

```
realestate_test[, c('BldgType', 'CentralAir', 'HouseStyle')] <- sapply(realestate_test[, c('BldgType',

realestate_train[, c('BldgType', 'CentralAir', 'HouseStyle')] <- sapply(realestate_train[, c('BldgType'

realestate_test$yhat <- NA
```

2b. Use an elastic net model to predict the price of the test data.

Set folds & create data matrix

```
#set folds
n.folds = 10

#create glmnet matrix
X = glmnet::makeX(
  train = realestate_train %>% select(-price),
  test = realestate_train %>% select(-price)
  )

X.train = X$x
Y.train = realestate_train %>% pull(price)

X.test = X$x
Y.test = realestate_train %>% pull(price)

#create folds
fold = sample(rep(1:n.folds, length=nrow(X.train)))
```

Identify optimal alpha

```
#loop lambda values
models <- list()
for (i in 0:20) {
  name <- paste0("alpha", i/20)


  models[[name]] <-
    cv.glmnet(X.train, Y.train, type.measure="mse", alpha=i/20,
              family="gaussian")
}

#predict results
results <- data.frame()
for (i in 0:20) {
  name <- paste0("alpha", i/20)

  ## Use each model to predict 'y' given the Testing dataset
  predicted <- predict(models[[name]],
                       s=models[[name]]$lambda.1se, newx=X.test)
  ## Calculate the Mean Squared Error...
  mse <- mean((Y.test - predicted)^2)

  ## Store the results
  temp <- data.frame(alpha=i/20, mse=mse, name=name)
  results <- rbind(results, temp)
}

#print results
print(results)
```

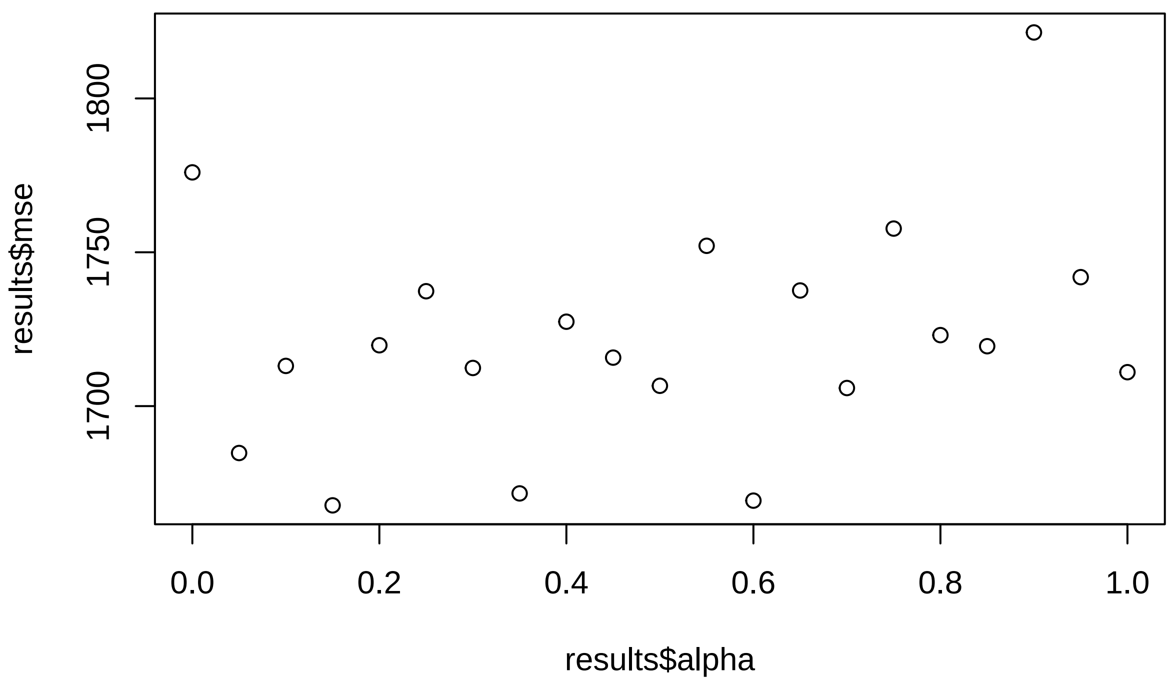
```
##    alpha      mse      name
## 1   0.00 1775.960    alpha0
## 2   0.05 1684.757 alpha0.05
## 3   0.10 1713.062  alpha0.1
## 4   0.15 1667.749 alpha0.15
## 5   0.20 1719.779  alpha0.2
## 6   0.25 1737.338 alpha0.25
## 7   0.30 1712.399  alpha0.3
## 8   0.35 1671.626 alpha0.35
## 9   0.40 1727.430  alpha0.4
## 10  0.45 1715.757 alpha0.45
## 11  0.50 1706.608  alpha0.5
## 12  0.55 1752.097 alpha0.55
## 13  0.60 1669.282  alpha0.6
## 14  0.65 1737.584 alpha0.65
## 15  0.70 1705.880  alpha0.7
## 16  0.75 1757.692 alpha0.75
## 17  0.80 1723.070  alpha0.8
## 18  0.85 1719.467 alpha0.85
## 19  0.90 1821.434  alpha0.9
## 20  0.95 1741.911 alpha0.95
## 21  1.00 1711.022    alpha1
```

Print and plot results of alpha values

```
#print results
print(results)
```

```
##    alpha      mse      name
## 1   0.00 1775.960    alpha0
## 2   0.05 1684.757 alpha0.05
## 3   0.10 1713.062  alpha0.1
## 4   0.15 1667.749 alpha0.15
## 5   0.20 1719.779  alpha0.2
## 6   0.25 1737.338 alpha0.25
## 7   0.30 1712.399  alpha0.3
## 8   0.35 1671.626 alpha0.35
## 9   0.40 1727.430  alpha0.4
## 10  0.45 1715.757 alpha0.45
## 11  0.50 1706.608  alpha0.5
## 12  0.55 1752.097 alpha0.55
## 13  0.60 1669.282  alpha0.6
## 14  0.65 1737.584 alpha0.65
## 15  0.70 1705.880  alpha0.7
## 16  0.75 1757.692 alpha0.75
## 17  0.80 1723.070  alpha0.8
## 18  0.85 1719.467 alpha0.85
## 19  0.90 1821.434  alpha0.9
## 20  0.95 1741.911 alpha0.95
## 21  1.00 1711.022    alpha1
```

```
#plot results
plot(results$alpha, results$mse)
```



Determine optimal alpha (min returned)

```
#min results
results %>% slice_min(mse)
```

```
##   alpha      mse      name
## 1  0.15 1667.749 alpha0.15
```

Generate elastic net

```r
#Build Training Models using cross-validation, e.g., cv.glmnet()
#ols
fit.ls = lm(Y.train~X.train)
beta.ls = coef(fit.ls)
yhat.ls = cbind(1, X.test) %*% coef(fit.ls)

#get lambda that minimizes cv error and 1 SE rule
#Ridge
fit.ridge = cv.glmnet(X.train, Y.train, alpha=0, foldid=fold)
beta.ridge = coef(fit.ridge, s="lambda.min")
yhat.ridge = predict(fit.ridge, newx = X.test, s="lambda.min")

#Lasso
fit.lasso = cv.glmnet(X.train, Y.train, alpha=1, foldid=fold)
beta.lasso = coef(fit.lasso, s="lambda.min")
yhat.lasso = predict(fit.lasso, newx = X.test, s="lambda.min")

#Elastic Net
a = .7 # set alpha for elastic net
fit.enet = cv.glmnet(X.test, Y.test, alpha=a, foldid=fold)
beta.enet = coef(fit.enet, s="lambda.min")
yhat.enet = predict(fit.enet, newx = X.test, s="lambda.min")
```

Evaluate Predictions

```r
#evaluate predictions
YHAT = list(ols = yhat.ls, ridge = yhat.ridge,
            lasso=yhat.lasso, enet=yhat.enet)
purrr::map_dbl(YHAT, ~mean( (Y.test-.x)^2 )) %>% sort
```

```
##     enet    lasso    ridge
## 1500.363 1500.915 1537.381
```

Return min lambda

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

```
## [1] 0.8853807
```

Elastic net alpha used 0.7 due to the test identifying alpha of 0.7 producing the lowest MSE in additon to the secondary test above evaluating predictions against lasso and ridge where elastic net of alpha=0.7 outperformed each. The minimum lambda which was used is 0.9717042.

2c. 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.

```r
elastic_net_final = glmnet(X.train, Y.train, alpha = 0.7, lambda = 0.9717042)
yhat = predict(elastic_net_final, X.test, s = "lambda.lse")
yhat = tibble(yhat)
names(yhat)[1] <- "yhat"
```

```r
#export csv file
write.csv(yhat,"C:\\Users\\brwil\\Desktop\\SY MSDS\\DS 6030 Stat Learning\\Week 4\\wilson_benjamin.csv"
```

2d. Report the anticipated performance of your method in terms of RMSE. We will see how close your performance assessment matches the actual value.

```r
sqrt(min(yhat))
```

```
## [1] 4.377862
```