R Practice 4 ALY6015

R practice week 4 - Module 4 Assignment — Regularization

by Mohammad Hossein Movahedi

movahed.m@northeastern.edu

14 May 2022

Table of contents

Introduction

This R practice mainly focuses on Regularization. During this assignment, I will use the `Collage` dataset to build regularization models using Ridge and Lasso, perform `stepwise selection`, and then fit a model in 14 steps.

Analytics

This part will go through the steps described in the assignment and use R to predict `grade.rate`.

Step 1: Split the data into a train and test set

this step is straightforward coding.

```
#split the data
set.seed(96)
trainIndex<-sample(x = nrow(dataset) , size = nrow(dataset)*0.7)
train <- dataset[trainIndex,]
test <- dataset[-trainIndex,]

train_x <- model.matrix(Grad.Rate~.,train)[,-1]
test_x <- model.matrix(Grad.Rate~.,test)[,-1]
```

```
train_y <- train$Grad.Rate
test_y <- test$Grad.Rate
```

The results are two datasets for modeling and testing separately.

Step 2: Estimating the lambda.min and lambda.1se values.

I use `cv.glmnet()` to estimate `min` and `1se` of lambda in this part.

```
cv.lasso <- cv.glmnet(train_x,train_y,nfolds = 10)
#object : min perdiction error
log(cv.lasso$lambda.min)
log(cv.lasso$lambda.1se)
```

The resulting plot is shown below.

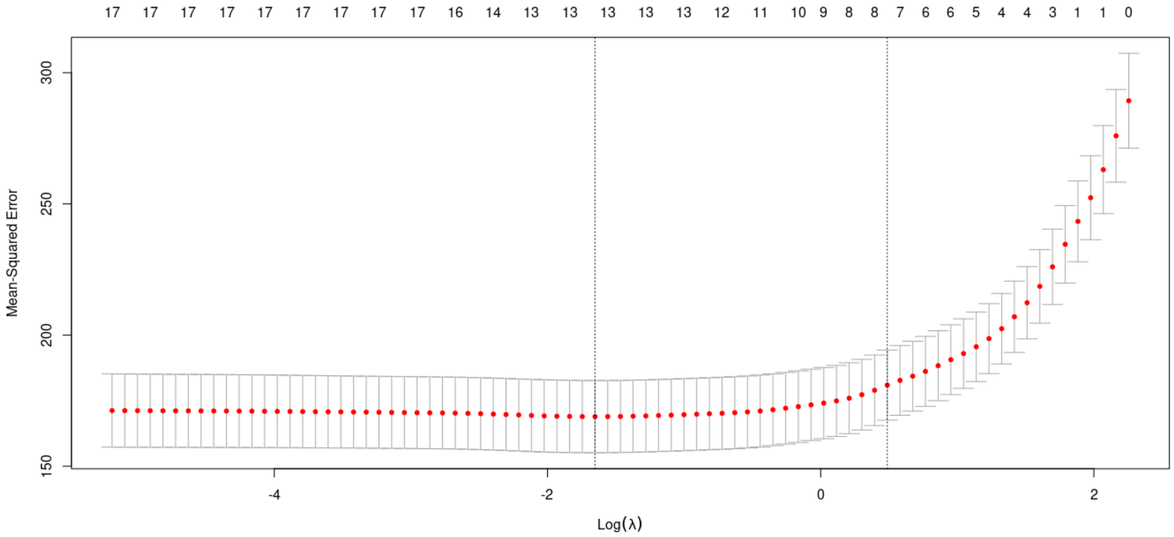
The resulting values for `lambda.min` and `lambda.1se` are shown below.

```
> #object : min perdiction error
> log(cv.lasso$lambda.min)
[1] -1.652088
> log(cv.lasso$lambda.1se)
[1] 0.4876883
```

As it can be seen, the best model has 13 variables in it, while the simplest model that does the same thing has eight variables.

Step 3: Plot the results from the cv.glmnet function

```
#finding the best value of lambda
set.seed(96)
cv.lasso <- cv.glmnet(train_x,train_y,nfolds = 10)
plot(cv.lasso)
```



the `cv.lasso` plot

As seen in the plot, we see a spike in the graph starts after setting lambda between 8 and 7 the mean of squared error starts to rise dramatically.

Step 4: Fitting a Ridge regression model against the training set and reporting on the coefficients.

```
#fit model based on lambda.min with Ridge regression
model.1se <- glmnet(train_x,train_y,alpha = 0,lambda = cv.lasso$lambda.1se)
model.1se
#display reg coeff
coef(model.1se)
#display coeff of train model with no regularization
tm <- lm(Grad.Rate~.,data = train)
coef(tm)
```

The `coef(model.1se)` result is shown below.

```
> coef(model.1se)
18 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept)  3.209466e+01
PrivateYes   4.545804e+00
Apps         5.894800e-04
Accept       3.213770e-05
Enroll       3.249355e-04
Top10perc    8.051476e-02
Top25perc    1.127033e-01
F.Undergrad  5.744338e-05
P.Undergrad -1.254433e-03
Outstate     7.413867e-04
Room.Board   1.965067e-03
Books       -6.233951e-03
Personal    -2.017028e-03
PhD          6.396792e-02
Terminal    -8.927667e-03
S.F.Ratio    1.837197e-01
```

```
perc.alumni  2.659121e-01
Expend      -2.363661e-04
```

Compering these with coefficients of the full model shows a huge difference in Intercept; however, for most existing variables, the coefficients aren't that different between the two models

Step 5: Determine the performance of the fit model against the training set by calculating the root mean square error (RMSE)

```
#train set prediction
preds.train<- predict(model.1se,newx = train_x)
train.rmse <- rmse(train_y,preds.train)
```

Step 6: Determine the performance of the fit model against the test set by calculating the root mean square error (RMSE)

```
#test set prediction
preds.test <- predict(model.1se,newx =test_x)
test.rmse <- rmse(test_y,preds.test)
```

```
#view RMSE of full model
preds.tm <- predict(tm,new = test)
full.rmse <-rmse(test$Grad.Rate,preds.tm)
#compering values of rmse
train.rmse
test.rmse
full.rmse
```

The results are shown below

```
> train.rmse
[1] 12.6653
> test.rmse
[1] 12.79758
> full.rmse
[1] 12.8865
```

As can be seen, the regularized model performed better in reducing RMSE than the initial model.

Step 7 and 8: estimating the lambda.min and lambda.1se values. And creating a plot for glmnet.

It is as same as steps 2 and 3; therefore, I skip it.

Step 9: Fit a LASSO regression model against the training set and report the coefficients.

```
#fit model based on lambda.min with lasso regression
model.1se <- glmnet(train_x,train_y,alpha = 1,lambda = cv.lasso$lambda.1se)
model.1se
#display reg Coeff
coef(model.1se)
```

The results are shown below

```
> coef(model.1se)
18 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept)  4.001468e+01
PrivateYes   .
Apps         5.397571e-05
Accept       .
Enroll       .
Top10perc    3.591467e-02
Top25perc    1.128799e-01
F.Undergrad  .
P.Undergrad -2.024551e-04
Outstate     1.020005e-03
Room.Board   9.719135e-04
Books        .
Personal    -1.011774e-03
PhD          .
Terminal     .
S.F.Ratio    .
perc.alumni  2.067628e-01
Expend       .
```

As seen in the results, nine variables from the original model lost all the coefficients in the new model and were removed.

Step 10: Determine the performance of the fit model against the training set by calculating the root mean square error (RMSE).

```
#train set prediction
preds.train<- predict(model.1se,newx = train_x)
train.rmse <- rmse(train_y,preds.train)
```

Step 11: Determine the performance of the fit model against the test set by calculating the root mean square error (RMSE)

```
#test set prediction
preds.test <- predict(model.1se,newx =test_x)
test.rmse <- rmse(test_y,preds.test)

#compering values of rmse
train.rmse
test.rmse
full.rmse
```

The results are shown below

```
> train.rmse
[1] 13.28165
> test.rmse
[1] 13.16788
> full.rmse
[1] 12.8865
```

As can be seen, the RMSE of the model increased; however, we managed to eliminate nine variables without making much RMSE in test data. The RMSE in testing data is still the same as training data, indicating the model's success.

Conclusion

Step 12: Which model performed better and why?

Each model has its unique advantages. The ridge model reduced RMSE more significantly; however, it used all the variables of the initial model; however, the lasso eliminated some variables making the model more simple.

Step 13: Perform stepwise selection and then fit a model

```
# Forward selection method
tm <- lm(Grad.Rate~1.,data = train)
datan <- (train[, unlist(lapply(train, is.numeric))])
m = cor(datan,datan$Grad.Rate)
#first rearrenging cor matrix and remove duplicates
msort <- m %>%
  as.data.frame() %>%
  mutate(var1 = rownames(.)) %>%
  gather(var2, value, -var1) %>%
  arrange(desc(value)) %>%
  group_by(value) %>%
  filter(row_number()==1)

msort$value <- abs(msort$value)
msort<-msort[(msort$value < 0.999),]
msort <-arrange(msort,desc(value))
head(msort,5)
steps <- step(tm, direction = 'forward',scope ~
                Outstate + perc.alumni + Top10perc + Top25perc + Room.Board)
model_forward <- lm(formula = Grad.Rate ~ Outstate + Top25perc + perc.alumni + Room.Board, data = train)
summary(model_forward)
#view RMSE of full model
preds.model_forward <- predict(model_forward,new = test)
full.rmse <-rmse(test$Grad.Rate,preds.model_forward)
full.rmse
```

the RMSE of model_forward is `13.0137`, which is very close to the ridge model, indicating the effectiveness of this method.

Bibliography

Instructure.com. (2019). *Module 4 Assignment — Regularization*. [online] Available at: https://northeastern.instructure.com/courses/116151/assignments/1369378 [Accessed 14 May 2022].

Pumpkin C (2022). *correlation of one variable to all the other in R*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/45892274/correlation-of-one-variable-to-all-the-other-in-r [Accessed 14 May 2022].

Guru99. (2020). *R Stepwise & Multiple Linear Regression [Step by Step Example]*. [online] Available at: https://www.guru99.com/r-simple-multiple-linear-regression.html [Accessed 14 May 2022].

Appendix

```
install.packages('FSA')
install.packages('FSAdata')
install.packages('magrittr')
install.packages('dplyr')
install.packages('tidyr')
install.packages('plyr')
install.packages('tidyverse')
install.packages('outliers')
install.packages('ggplot2')
install.packages('lubridate')
install.packages('corrplot')

library(ggplot2)
library(outliers)
library(FSA)
library(FSAdata)
library(magrittr)
library(dplyr)
library(tidyr)
library(dplyr)
library(tidyverse)
library(scales)
library(lubridate)
library(corrplot)
install.packages('caret')
library(caret)
install.packages('ggplot2')
library(ggplot2)
install.packages("glmnet")
install.packages('Metrics')
library(glmnet)
library(Metrics)

#load the data
library(ISLR)
data("College")
dataset <- as.data.frame(College)
```

```
#split the data
set.seed(96)
trainIndex<-sample(x = nrow(dataset) , size = nrow(dataset)*0.7)
train <- dataset[trainIndex,]
test <- dataset[-trainIndex,]

train_x <- model.matrix(Grad.Rate~.,train)[,-1]
test_x <- model.matrix(Grad.Rate~.,test)[,-1]

train_y <- train$Grad.Rate
test_y <- test$Grad.Rate

#finding value of lambda

#finding the best value of lambda
set.seed(96)
cv.lasso <- cv.glmnet(train_x,train_y,nfolds = 10)
plot(cv.lasso)

#object : min perdiction error
log(cv.lasso$lambda.min)
log(cv.lasso$lambda.1se)
#Ridge
#fit model based on lambda.min with Ridge regression
model.1se <- glmnet(train_x,train_y,alpha = 0,lambda = cv.lasso$lambda.1se)
model.1se
#display reg coeff
coef(model.1se)

#display coeff of train model with no regularization
tm <- lm(Grad.Rate~.,data = train)
coef(tm)

#view RMSE of full model
preds.tm <- predict(tm,new = test)
full.rmse <-rmse(test$Grad.Rate,preds.tm)

#train set prediction
preds.train<- predict(model.1se,newx = train_x)
train.rmse <- rmse(train_y,preds.train)
#test set prediction
preds.test <- predict(model.1se,newx =test_x)
test.rmse <- rmse(test_y,preds.test)

#compering values of rmse
train.rmse
test.rmse
full.rmse
#lasso
#fit model based on lambda.min with lasso regression
model.1se <- glmnet(train_x,train_y,alpha = 1,lambda = cv.lasso$lambda.1se)
model.1se
#display reg coeff
coef(model.1se)

#display coeff of train model with no regularization
tm <- lm(Grad.Rate~.,data = train)
coef(tm)

#view RMSE of full model
preds.tm <- predict(tm,new = test)
full.rmse <-rmse(test$Grad.Rate,preds.tm)

#train set prediction
preds.train<- predict(model.1se,newx = train_x)
train.rmse <- rmse(train_y,preds.train)
#test set prediction
```

```
preds.test <- predict(model.1se,newx =test_x)
test.rmse <- rmse(test_y,preds.test)

#compering values of rmse
train.rmse
test.rmse
full.rmse

# Forward selection method
tm <- lm(Grad.Rate~1.,data = train)
datan <- (train[, unlist(lapply(train, is.numeric))])
m = cor(datan,datan$Grad.Rate)
#first rearrenging cor matrix and remove duplicates
msort <- m %>%
  as.data.frame() %>%
  mutate(var1 = rownames(.)) %>%
  gather(var2, value, -var1) %>%
  arrange(desc(value)) %>%
  group_by(value) %>%
  filter(row_number()==1)

msort$value <- abs(msort$value)
msort<-msort[(msort$value < 0.999),]
msort <-arrange(msort,desc(value))
head(msort,5)
steps <- step(tm, direction = 'forward',scope ~
                Outstate + perc.alumni + Top10perc + Top25perc + Room.Board)
model_forward <- lm(formula = Grad.Rate ~ Outstate + Top25perc + perc.alumni + Room.Board, data = train)
summary(model_forward)
#view RMSE of full model
preds.model_forward <- predict(model_forward,new = test)
full.rmse <-rmse(test$Grad.Rate,preds.model_forward)
full.rmse
```