Problem Set 3 - Runhua Li

Runhua Li

2/15/2020

```
rm(list = ls())
library(tidyverse)
```

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

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

```
library(rsample)
library(rcfss)
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
library(ISLR)
library(broom)
```

1 Decision Trees

1.1 Set Up

```
set.seed(3751)
NES <- read.csv("/Users/RunhuaLi/R/nes2008.csv")
lambda <- seq(from = 0.0001, to = 0.04, by = 0.001)
p <- ncol(NES) - 1
```

1.2 Seperating Training and Testing Sets

```
set.seed(3751)
train=sample(1:nrow(NES), 3/4 * nrow(NES))
```

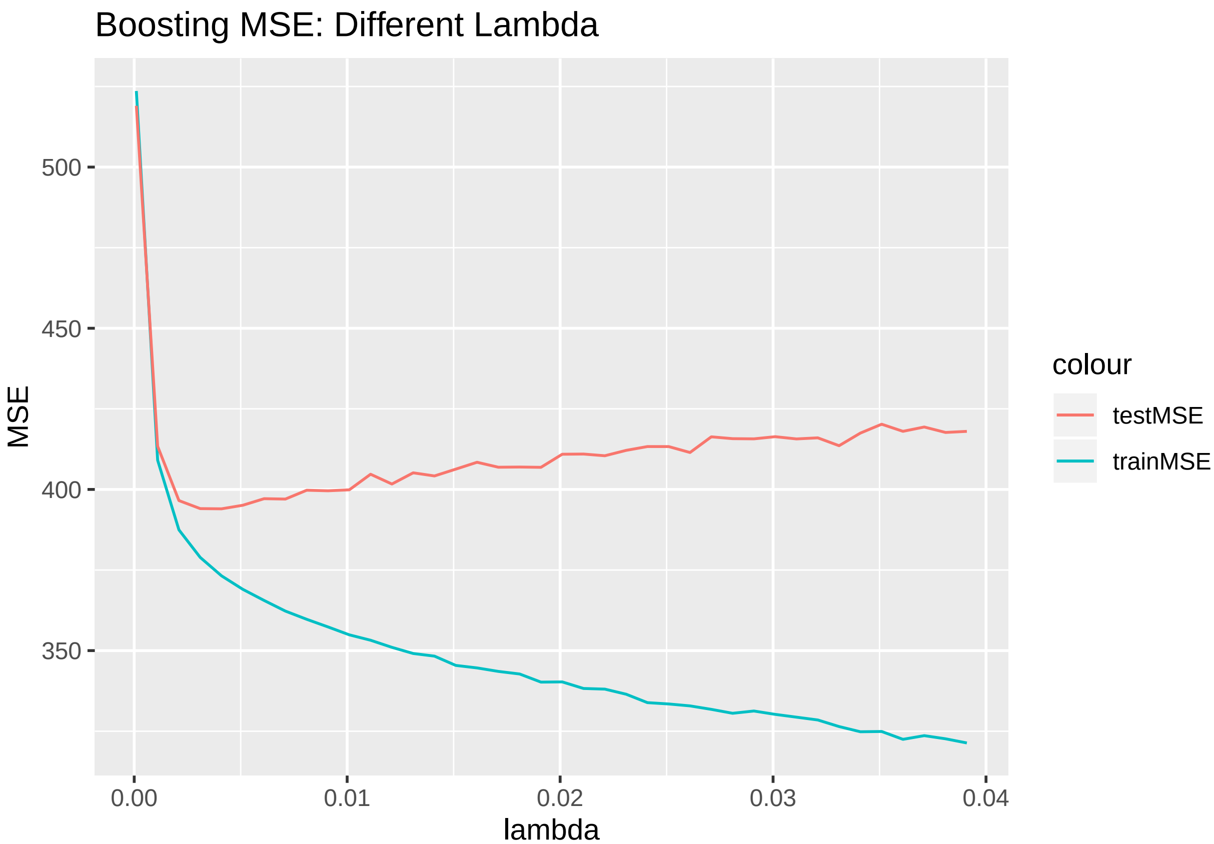
1.3 Boosting with A Range of Shrinkage

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
mse_train <- 0
mse_test <- 0
for(l in lambda){
```

```r
boost.NES <- gbm(biden ~ .,
                 data=NES[train,],
                 distribution = "gaussian",
                 n.trees = 1000,
                 shrinkage = l,
                 interaction.depth = 4)

preds_train = predict(boost.NES, newdata = NES[train,], n.trees = 1000)
preds_test = predict(boost.NES, newdata = NES[-train,], n.trees = 1000)

SE_train = with(NES[train,], (preds_train - biden)^2)
SE_test = with(NES[-train,], (preds_test - biden)^2)

mse_train <- c(mse_train, mean(SE_train))
mse_test <- c(mse_test, mean(SE_test))
}

mse_train <- mse_train[-1]
mse_test <- mse_test[-1]

mse_df <- data.frame(trainMSE = mse_train,
                     testMSE = mse_test,
                     lambda = lambda)

ggplot(data = mse_df) +
  geom_line(aes(x = lambda, y = trainMSE, color = "trainMSE")) +
  geom_line(aes(x = lambda, y = testMSE, color = "testMSE")) +
  labs(title = "Boosting MSE: Different Lambda") +
  ylab("MSE")
```



1.4 Boosting with 0.01 Shrinkage

```
boost.NES <- gbm(biden ~ .,
                 data=NES[train,],
                 distribution = "gaussian",
                 n.trees = 1000,
                 shrinkage = 0.01,
                 interaction.depth = 4)

preds_test = predict(boost.NES, newdata = NES[-train,], n.trees = 1000)

MSE_boost = mean(with(NES[-train,], (preds_test - biden)^2))

MSE_boost
```

```
## [1] 400.0943
```

When λ is set to 0.01, the reported test MSE from boosting is 400.0943. Compare this test MSE with the plot in Section 1.3, I see that 0.01 is not the optimal value of λ, which is around 0.002. Still, the performances (test MSEs) of boosting under these two λ s are very close. The difference in test MSEs is less than 10.

1.5 Bagging

```
library(ipred)
library(rpart)
bagging.NES <- bagging(biden ~ .,
                       data=NES[train,],
```

```
                       nbagg = 100,
                       coob=T)

preds_test = predict(bagging.NES,
                     newdata = NES[-train,],
                     nbagg = 100)
MSE_bagg = mean(with(NES[-train,], (preds_test - biden)^2))

MSE_bagg
```

```
## [1] 393.0304
```

The test set MSE for bagging is $94.0189.

1.6 Random Forest

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rf.NES <- randomForest(biden ~ .,
                       data = NES[train,],
                       ntree = 100,
                       mtry = sqrt(p))

preds_test = predict(rf.NES,
                     newdata = NES[-train,],
                     ntree = 100)
MSE_rf = mean(with(NES[-train,], (preds_test - biden)^2))

MSE_rf
```

```
## [1] 397.3775
```

The test set MSE for random forest is 395.219.

1.7 Linear Model

```
lm.NES <- lm(biden ~ .,
             data = NES[train,])

preds_test = predict(lm.NES,
                     newdata = NES[-train,])
MSE_lm = mean(with(NES[-train,], (preds_test - biden)^2))

MSE_lm
```

```
## [1] 388.4608
```

The test set MSE for linear model is 388.4608.

1.8

```
MSE_boost
```

```
## [1] 400.0943
```

```
MSE_bagg
```

```
## [1] 393.0304
```

```
MSE_rf
```

```
## [1] 397.3775
```

```
MSE_lm
```

```
## [1] 388.4608
```

Based on test set MSEs of boosting, bagging, random forest and linear model, I conclude that linear model fits best. However, I am very aware that this depends on the seed I set to seperate training and testing sets.

2 Support Vector Machine

2.1 Set Up

```
set.seed(3751)
OJsplit <- initial_split(OJ, prop = 800 / dim(OJ))
```

```
## Warning in if (!is.numeric(prop) | prop >= 1 | prop <= 0) stop("`prop` must
## be a number on (0, 1).", : the condition has length > 1 and only the first
## element will be used
```

```
  OJtrain <- training(OJsplit)
  OJtest <- testing(OJsplit)
```

2.2 Fitting

```
library(e1071)
OJ.svm <- svm(Purchase ~ .,
              data = OJtrain,
              kernel = "linear",
```

```r
              cost = 0.01,
              scale = FALSE); summary(OJ.svm)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJtrain, kernel = "linear",
##     cost = 0.01, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  627
##
##  ( 314 313 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

The above result shows that the majority of training observations lie within the margin. To be specific, 627 out of 800 observations lie within the margin, half of which lie on each side.

2.3 Confusion Matrix and Error Rate

```r
predicted <- predict(OJ.svm, newdata = OJtest)
table(predicted = predicted, true = OJtest$Purchase)
```

```
##          true
## predicted  CH  MM
##        CH 157  39
##        MM  12  62
```

```r
library(yardstick)
```

```
## For binary classification, the first factor level is assumed to be the event.
## Set the global option `yardstick.event_first` to `FALSE` to change this.
```

```
##
## Attaching package: 'yardstick'
```

```
## The following object is masked from 'package:readr':
##
##     spec
```

```r
train_accu <- OJtrain %>%
  mutate(estimate = predict(OJ.svm, newdata = OJtrain)) %>%
  accuracy(truth = Purchase, estimate = estimate)
error_train.01 <- 1 - train_accu$.estimate[[1]]

test_accu <- OJtest %>%
  mutate(estimate = predict(OJ.svm, newdata = OJtest)) %>%
```

```
  accuracy(truth = Purchase, estimate = estimate)
error_test.01 <- 1 - test_accu$.estimate[[1]]

error_train.01
```

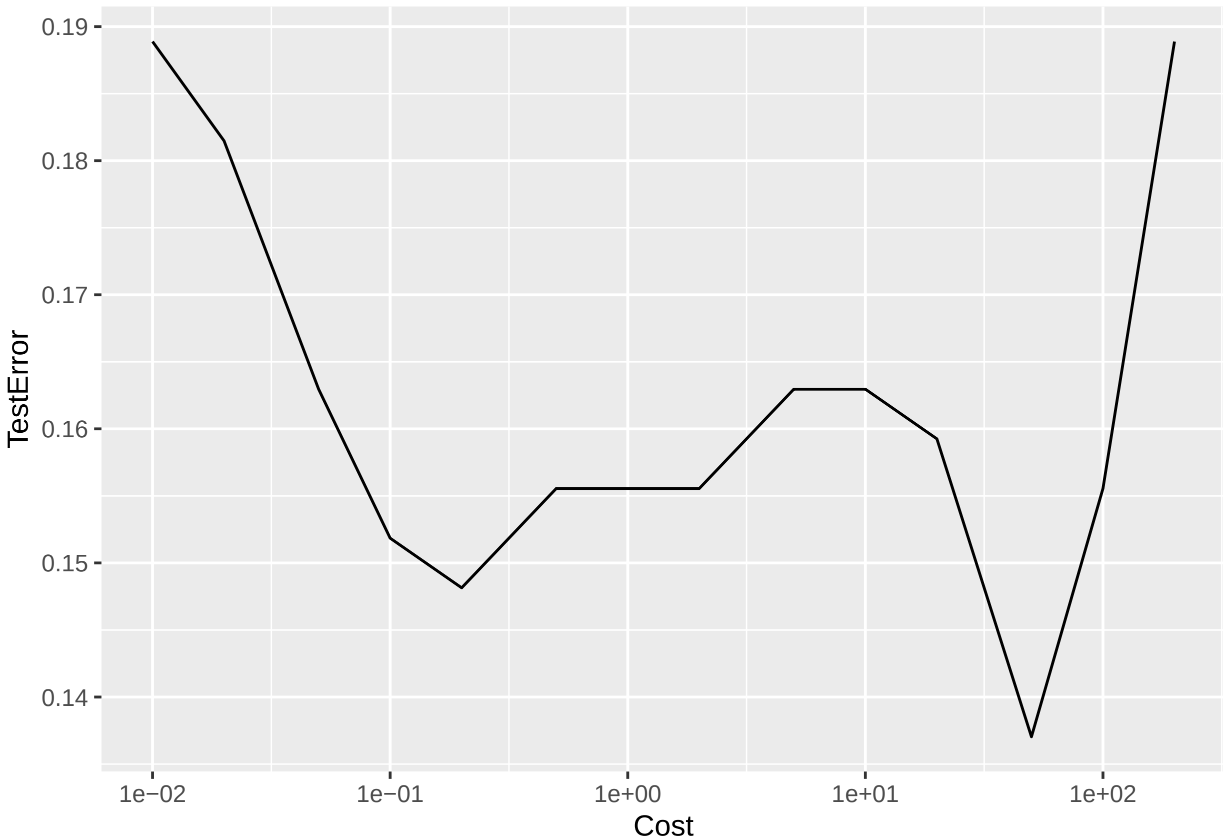
```
## [1] 0.22625
```

```
error_test.01
```

```
## [1] 0.1888889
```

2.4 Finding the Optimal Cost

```
c <- c(1, 2, 5, 10)
  c <- c(c / 100, c / 10, c, c * 10, 200)
# I couldn't do c = 500 or 1000, because when I did,
  # R warns me of "reaching max number of iterations.

error_testc <- 0

for(i in c){
OJ.svm <- svm(Purchase ~ .,
              data = OJtrain,
              kernel = "linear",
              cost = i,
              scale = FALSE)

test_accu <- OJtest %>%
  mutate(estimate = predict(OJ.svm, newdata = OJtest)) %>%
  accuracy(truth = Purchase, estimate = estimate)

error_testc <- c(error_testc, 1 - test_accu$.estimate[[1]])
}

error_testc <- error_testc[-1]

testerror.SVM <- data.frame(TestError = error_testc,
                            Cost = c)

ggplot(data = testerror.SVM,
       aes(x = Cost, y = TestError)) +
  geom_line() +
  scale_x_log10()
```



I found the optimal value for cost to be 50, with 2 as the second best which is not close.

I am aware that this “optimal cost” is sensitive to the seed I used to split the training and testing sets. I used `set.seed(1234)` to check the optimal cost’s robustness to the seed, and I found the optimal cost under seed 1234 is 10, with 1 as the close second best.

2.5 Comparison

```
OJ.svm <- svm(Purchase ~ .,
              data = OJtrain,
              kernel = "linear",
              cost = 50,
              scale = FALSE); summary(OJ.svm)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJtrain, kernel = "linear",
##     cost = 50, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  50
##
## Number of Support Vectors:  288
##
##  ( 143 145 )
```

```
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
predicted <- predict(OJ.svm, newdata = OJtest)
table(predicted = predicted, true = OJtest$Purchase)
```

```
##          true
## predicted  CH  MM
##        CH 157  25
##        MM  12  76
```

```r
train_accu <- OJtrain %>%
  mutate(estimate = predict(OJ.svm, newdata = OJtrain)) %>%
  accuracy(truth = Purchase, estimate = estimate)
error_train50 <- 1 - train_accu$.estimate[[1]]

test_accu <- OJtest %>%
  mutate(estimate = predict(OJ.svm, newdata = OJtest)) %>%
  accuracy(truth = Purchase, estimate = estimate)
error_test50 <- 1 - test_accu$.estimate[[1]]

error_train50
```

```
## [1] 0.175
```

```r
error_test50
```

```
## [1] 0.137037
```

```r
error_train50 / error_train.01
```

```
## [1] 0.7734807
```

```r
error_test50 / error_train.01
```

```
## [1] 0.6056886
```

The training set error is larger than testing set error when cost is set to be 50. They are both substaintially less than errors when cost is set to be 0.01 originally. To be specific, the training set error is around 23% less, and the testing set error is around 39% less.