Homework 12

Christina Lee

11/29/2021

1. Use your dataset with a continuous dependent variable:

a. Divide your data into two equal-sized samples, the in-sample and the out-sample. Estimate the elastic net model using the in-sample data with at least three levels of alpha (ie, three positions in between full lasso and full ridge; eg, alpha = 0, 0.5, and 1), using cv.glmnet to find the best lambda level for each run. (Remember that glmnet prefers that data be in a numeric matrix format rather than a data frame.)

I will be using the BostonHousing data set from the mlbench package. The data set contains 14 variables and 506 observations. The data set is used to predict the house price in Boston from house details.

```
#install.packages("mlbench")
library(mlbench)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-3
```

```
#Load data set
data("BostonHousing")
```

```
str(BostonHousing) # all variables are numeric except for "chas" which is a factor.
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : num  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ b      : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
numericvars1 <- c(1:3,5:13)  # extract numeric variables only
x <- as.matrix(BostonHousing[,numericvars1])  # numeric matrix
y <- BostonHousing$medv # numeric vector
```

```
# Divide the data set into training and testing sets in two equal-size samples.
set.seed(1)
Train <- sample(1:nrow(x),nrow(x)/2) # half the rows at random
```

```
Test <- setdiff(1:nrow(x), Train) # the other half

Trainx <- x[Train,] # matrix
Testx <- x[Test,] # matrix

Trainy <- y[Train] # vector
Testy <- y[Test] # vector
```

```
# Estimate the elastic net model with alpha = 0, 0.5 and 1 using in-sample data.

Lambdalevels <- 10^seq(7,-7, length =100) # set lambda levels for glmnet to try.

set.seed(222)
ridge <- cv.glmnet(Trainx, Trainy, alpha=0, lambda = Lambdalevels)
lasso <- cv.glmnet(Trainx, Trainy, alpha=1, lambda = Lambdalevels)
elnet <- cv.glmnet(Trainx, Trainy, alpha=0.5, lambda = Lambdalevels)
```

```
ridge # find best (minimum) lambda value and MSE for ridge
```

```
##
## Call:  cv.glmnet(x = Trainx, y = Trainy, lambda = Lambdalevels, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  0.167    56   21.48 4.281      12
## 1se  4.329    46   24.45 4.839      12
```

```
lasso # find best (minimum) lambda value and MSE for lasso
```

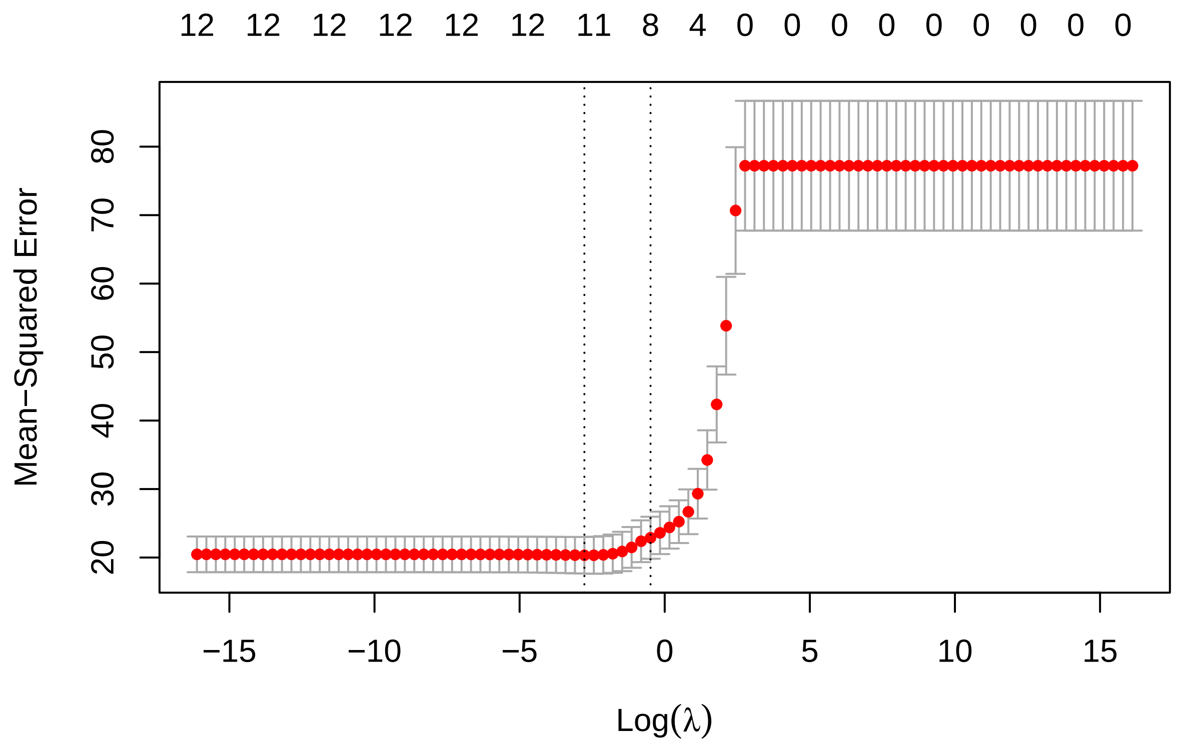
```
##
## Call:  cv.glmnet(x = Trainx, y = Trainy, lambda = Lambdalevels, alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min 0.0327    61   21.03 4.352      12
## 1se 0.8498    51   25.01 4.802       5
```

```
elnet # find best (minimum) lambda value and MSE for elnet
```

```
##
## Call:  cv.glmnet(x = Trainx, y = Trainy, lambda = Lambdalevels, alpha = 0.5)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min 0.0628    59   20.33 2.676      12
## 1se 0.6136    52   22.89 3.066       8
```

Results: The best (minimum) MSE value for full ridge (alpha=0) is 4.281, full lasso (alpha=1) is 4.352 and the elastic net (mix of both lasso/ridge) (alpha=0.5) is 2.676. Hence, for elastic net regression, when alpha = 0.5 it gives the best MSE value which is 2.676.

```
plot(elnet) #shows the MSE values for each lambda value for elnet.
```



b. Choose the alpha (and corresponding lambda) with the best results (lowest error), and then test that model out-of-sample using the out-sample data. To find the lowest MSE associated the best lambda value from a cv.glmnet ouput, note that my_cv_output has inside it an object called cvm, which is a vector of MSE values associated with each lambda level; min(my_cv_outputcvm) will give you the best MSE from that model.

```
yhat.elnet <- predict(elnet$glmnet.fit, s=elnet$lambda.min, newx=Testx)

# mean square error for elnet
mse.e <- sum((Testy - yhat.elnet)^2)/nrow(Testx)
mse.e
```

```
## [1] 27.38717
```

c. Compare your out-of-sample results from b to regular multiple regression: fit a regression model using the in-sample data, predict the out-of-sample yhat using the out-of-sample X, and estimate the error between the predicted yhat and the true out-of-sample y. Which worked better, the model from b or the regression model?

```
lmout <- lm(Trainy ~ Trainx)
yhat.r <- cbind(1,Testx) %*% lmout$coefficients

# mean square error for Multiple regression fit
mse.r <- sum((Testy - yhat.r)^2)/nrow(Testx)
mse.r
```

```
## [1] 27.37977
```

Based on the results, the MSE for elnet (27.38717) and multiple regression (27.37977) suggests that there is not much of a major difference in MSE, however, multiple regression's MSE is a tiny better than elnet's MSE.

d. Which coefficients are different between the multiple regression and the elastic net model? What, if anything, does this tell you substantively about the effects of your independent variables on your dependent variable?

```
coef(elnet, elnet$lambda.min) # coefficients for elnet
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)  3.283760e+01
## crim        -8.504572e-02
## zn           2.751278e-02
## indus        2.669564e-03
## nox         -1.109664e+01
## rm           4.152022e+00
## age         -9.947314e-04
## dis         -1.205499e+00
## rad          3.212675e-01
## tax         -1.672554e-02
## ptratio     -9.878115e-01
## b            6.863393e-03
## lstat       -4.897790e-01
```

```
lmout$coefficients # coefficients for multiple regression
```

```
##   (Intercept)    Trainxcrim      Trainxzn   Trainxindus     Trainxnox
##  35.715258040  -0.095874905   0.032904857   0.040767472 -13.123787402
##      Trainxrm     Trainxage     Trainxdis     Trainxrad     Trainxtax
##   4.090773901  -0.001938844  -1.306547398   0.398985374  -0.020467226
## Trainxptratio       Trainxb   Trainxlstat
##  -1.024567157   0.007256553  -0.493202713
```

The coefficients left to use for both elnet and multiple regression were fairly similar, leaving me with the same number of coefficients to be used for each final model. This suggests that most of the coefficients (independent variables) do have an effect on predicting the dependent variable y (medv = median value of owner-occupied homes in USD 1000's). Perhaps elnet's reducing feature and capabilities may be greater and more obvious with a larger data set that consists more variables.

2. Repeat the same process using your dataset with a binary dependent variable:

a. Divide your data into an in-sample and out-sample as before, and using the in-sample data, estimate an SVM using at least two different kernels and tune to find the best cost level for each. (Remember that SVM expects the dependent variable to be a factor, so if your dependent variable is numeric 0s and 1s, just us as.factor() to convert it first.)

This is a data set from the "mlbench" package called Sonar, Mines vs. Rocks (named "Sonar" in the package), which it consists 208 observations on 61 variables with the objective being to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. Each pattern is a set of 60 numbers in the range 0.0 to 1.0 and each number represents the energy within a particular frequency brand, integrated over a certain period of time. The target variable y is the last (61th) variable "Class" (R = rock or M = mine/metal cylinder).

```
library(e1071)


data("Sonar") # load data set
#str(Sonar)
```

```
x <- Sonar[,c(1:60)]
y <- Sonar$Class

dat <- data.frame(x=x, y=as.factor(y))
```

```r
# Divide data set into equal training and testing sets.
set.seed(123)
trainSonar <- sample(1:nrow(x), nrow(x)/2) # first half at random
testSonar <- setdiff(1:nrow(x), trainSonar) # other half

trainSonardat <- dat[trainSonar,]
testSonardat <- dat[testSonar,]
```

```r
costvalues <- 10^seq(-3,2,1)

# Estimate linear kernel
tuned.svm1 <- tune(svm, y~., data= trainSonardat, ranges = list(cost=costvalues),
                   kernel= "linear")
summary(tuned.svm1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.01
##
## - best performance: 0.2036364
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.4418182  0.1732422
## 2 1e-02 0.2036364  0.1371493
## 3 1e-01 0.2609091  0.1141141
## 4 1e+00 0.2918182  0.1433849
## 5 1e+01 0.3018182  0.1397098
## 6 1e+02 0.3018182  0.1397098
```

```r
# Estimate radial kernel
tuned.svm2 <- tune(svm, y~., data= trainSonardat, ranges = list(cost=costvalues),
                   kernel= "radial")
summary(tuned.svm2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.1345455
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.4509091  0.1403263
## 2 1e-02 0.4509091  0.1403263
## 3 1e-01 0.4509091  0.1403263
## 4 1e+00 0.1454545  0.1501147
## 5 1e+01 0.1345455  0.1310969
## 6 1e+02 0.1345455  0.1310969
```

According to the results, radial kernel outperformed linear kernel and has the highest in-sample accuracy with best performance being about 13% wrong, or 87% correct (best cost setting at 10), while linear kernel's best performance being about 20% wrong, or 80% correct (best cost setting at 0.01).

b. Chose the kernel and cost with the best in-sample accuracy, and then test that model out-of-sample using the out-sample data. How does the out-of-sample accuracy compare to the in-sample accuracy?

Since radial kernel outperformed linear kernel in (2a), I will be using radial kernel here.

```r
tuned.svm3 <- tune(svm, y~., data= trainSonardat, ranges = list(cost=costvalues),
                   kernel= "radial")

yhat.radial <- predict(tuned.svm3$best.model, newdata=testSonardat) # predict y with test data
sum(yhat.radial==testSonardat$y) / length(testSonardat$y) # percentage correct
```

```
## [1] 0.8269231
```

```r
table(predicted=yhat.radial, truth=testSonardat$y) # confusion matrix
```

```
##          truth
## predicted  M  R
##         M 46 10
##         R  8 40
```

Based on the results, the out-of-sample accuracy and the in-sample accuracy for radial kernel are not much different, with the out-of-sample accuracy being ~83% correct and the in-sample accuracy being ~87% correct.