NestedCV

Chase Enzweiler

10/16/2017

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.3.2
```

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 3.3.2
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.3.2
```

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

```
## Warning: package 'Matrix' was built under R version 3.3.2
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.2
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 3.3.2
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:pls':
##
##     R2
```

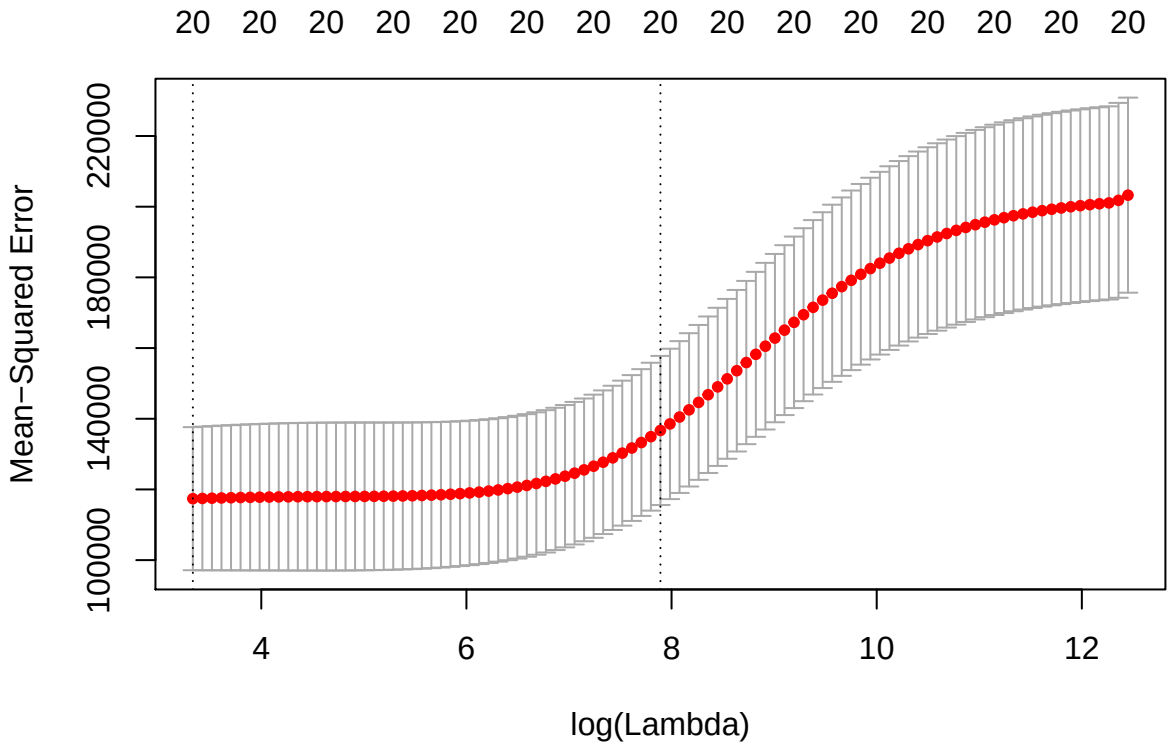
```
# remove NA
data_hitters <- na.omit(Hitters)
```

Cross Validation for Ridge Regression and the Lasso if ridge regression is desired our alpha should be 0 and if we want lasso we should set alpha to zero. Use cv.glmnet and cv.glmnet.plot for the ridge regression and the lasso

```
set.seed(300)
# for ridge regression
hitter_matrix <- model.matrix(Salary ~ . - 1, data = data_hitters)
```

```
ridge_cv <- cv.glmnet(hitter_matrix, data_hitters$Salary, alpha = 0)

plot(ridge_cv)
```

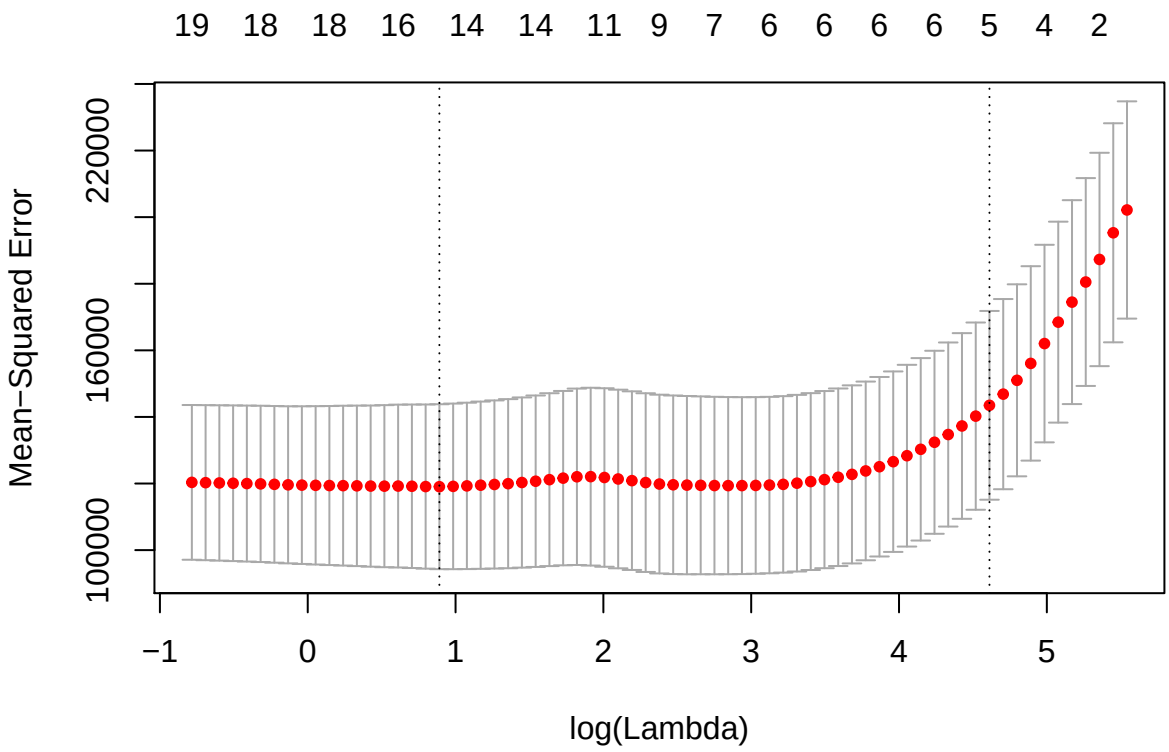



```
ridge_cv$lambda.min
```

```
## [1] 28.01718
```

```
# now for Lasso
set.seed(400)
lass_cv <- cv.glmnet(hitter_matrix, data_hitters$Salary, alpha = 1)

plot(lass_cv)
```




```
lass_cv$lambda.min
```

```
## [1] 2.436791
```

we see that the optimal lambda for ridge is 28.01718 and 2.436791 for the lasso

Nested Cross Validation

```
# nested cross validation for OLS, PCR, PLSR, RIDGE, LASSO
set.seed(1)


# create folds for 10 fold cross validation
folds <- createFolds(data_hitters$Salary, k = 10)

# just an index to help store mse's
i <- 1

# matrix to store test mse's for each model
test_mse <- matrix(NA, ncol = 5, nrow = 10)
colnames(test_mse) <- c("OLS", "PCR", "PLSR", "LASSO", "RIDGE")

for (fold in folds){

  # split data into training and test data

  training <- data_hitters[-fold, ]
```

```r
test <- data_hitters[fold, ]

###====== find the tuning parameters of each model========= ###

# PCR

pcr_fit <- pcr(Salary ~ ., data = training, scale = FALSE, validation = "CV", segments = 10)

# optimal amount of components for pcr
optimal_pcr_comp <- which.min(pcr_fit$validation$PRESS)

# PLSR

plsr_fit <- plsr(Salary ~ ., data = training, scale = FALSE, validation = "CV")

optimal_plsr_comp <- which.min(plsr_fit$validation$PRESS)

# Ridge regression

# this is design matrix without intercept for training
design_matrix <- model.matrix(Salary ~. -1 , data = training)

cv_rr <- cv.glmnet(design_matrix, training$Salary, alpha = 0)

rr_best_lambda <- cv_rr$lambda.min

rr_fit <- glmnet(design_matrix, training$Salary, alpha = 0, lambda = rr_best_lambda)

# Lasso

cv_lasso <- cv.glmnet(design_matrix, training$Salary, alpha = 1)

lasso_best_lambda <- cv_lasso$lambda.min

lasso_fit <- glmnet(design_matrix, training$Salary, alpha = 1, lambda = lasso_best_lambda)

# OLS
# No hyper parameter for ols
ols_fit <- lm(Salary ~. , data = training)

#####================================================######

# now predict the test sets


# PCR prediction
pcr_predict <- predict(pcr_fit, newdata = test, ncomp = optimal_pcr_comp)

pcr_mse <- mean((test$Salary - pcr_predict)^2)

# PLSR predictions

plsr_predict <- predict(plsr_fit, newdata = test, ncomp = optimal_plsr_comp)
```

```
  plsr_mse <- mean((test$Salary - plsr_predict)^2)

  #test design matrix without intercept

  # Ridge Regression predictions

  test_design_matrix <- model.matrix(Salary ~. -1 , data = test)

  rr_predict <- predict(rr_fit, newx = test_design_matrix)

  rr_mse <- mean((test$Salary - rr_predict)^2)

  # Lasso Predictions

  lasso_predict <- predict(lasso_fit, newx = test_design_matrix)

  lasso_mse <- mean((test$Salary - lasso_predict)^2)

  # OLS predictions

  ols_predict <- predict(ols_fit, newdata = test)

  ols_mse <- mean((test$Salary - ols_predict)^2)

  # put mse in the matrix

  # store lm
  test_mse[i,1] <- ols_mse
  # store pcr
  test_mse[i,2] <- pcr_mse
  # store plsr
  test_mse[i,3] <- plsr_mse
  # store lasso
  test_mse[i,4] <- lasso_mse
  # store ridge
  test_mse[i,5] <- rr_mse

  i = i + 1
}

# now we can average our test mse to find the best model

colMeans(test_mse)
```

```
##      OLS      PCR     PLSR    LASSO    RIDGE
## 113365.4 123052.8 115731.5 114611.1 117449.2
```

From our nested cross validation we have that our OLS model performs the best and the next best in order are LASSO, PLSR, RIDGE, and lastly PCR.