NBA 4920/6921 Lecture 15

Elastic Net Application

Murat Unal

10/21/2021

```
rm(list=ls())
options(digits = 3, scipen = 999)
library(tidyverse)
library(ISLR)
library(ggplot2)
library(jtools)
library(caret)
library(leaps)
library(glmnet)
Hitters <- ISLR::Hitters
Hitters <- na.omit(Hitters)
set.seed(2)
```

- Let's create 2.order variables and interaction terms

```
x=model.matrix(Salary~.,Hitters)[,-1]
y=Hitters$Salary

x.squared <- sapply(as.data.frame(x), function(i) i^2)

colnames(x.squared) <- paste0(colnames(x),"_sq", sep ="")

x.interact = model.matrix(~.^2,as.data.frame(x) )[,21:191]

x <- cbind(x, x.squared, x.interact)
rm(x.interact,x.squared)
dim(x)
```

```
[1] 263 209
```

```
train=sample(1:nrow(x), 0.7*nrow(x))
ols.data <- as.data.frame(cbind(Salary=y[train],x[train,]))
```

- Matrix to store RMSEs:

```
RMSE <- matrix(NA,ncol = 1, nrow = 6)
rownames(RMSE) <- c("rmse.ridge.lambda0",
"rmse.ridge.lambdabest","rmse.lasso.lambda1se",
"rmse.lasso.lambdabest", "rmse.elnet.lambda1se",
"rmse.elnet.lambdabest")
```

OLS

```
ols.mod <- lm(Salary~.,ols.data)
summ(ols.mod)
```

```
MODEL INFO:
Observations: 184
Dependent Variable: Salary
Type: OLS linear regression

MODEL FIT:
F(183,0) = NaN, p = NaN
R² = 1.00
Adj. R² = NaN

Standard errors: OLS
------------------------------------------------------------------
                                    Est.    S.E.    t val.
---------------------------- ----------- ------ -------- --
(Intercept)                      1323.31
```

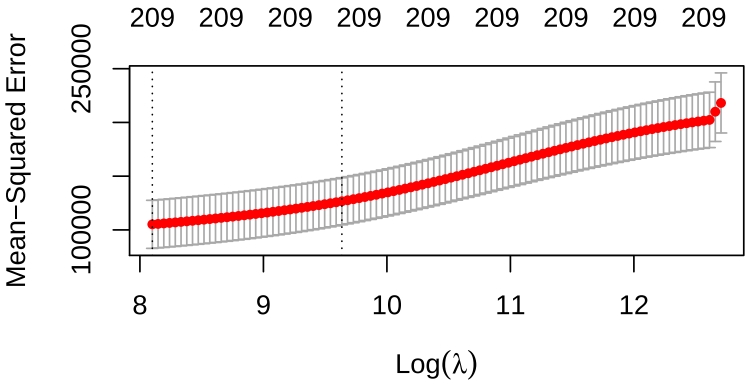
Ridge Regression

```
ridge.cv=cv.glmnet(x[train,],y[train]
                  ,alpha=0,nfold=10,
                 type.measure="mse")
bestlam=ridge.cv$lambda.min
bestlam
```

```
[1] 3296
```

- Plot

```
plot(ridge.cv)
```



- Predictions for *lambda* $= 0$

```
ridge.pred.lambda0 = predict(ridge.cv,newx=x[-train,],
                    s=0,exact=TRUE,
                    x=x[train,],y=y[train])

rmse.ridge.lambda0 <- sqrt(mean((y[-train]-
                          ridge.pred.lambda0)^2))
rmse.ridge.lambda0
```

```
[1] 556
```

- Predictions for best λ

```
ridge.pred.lambdabest = predict(ridge.cv,newx=x[-train,],
                        s=bestlam,exact=TRUE,
                        x=x[train,],y=y[train])

rmse.ridge.lambdabest <- sqrt(mean((y[-train]-
                          ridge.pred.lambdabest)^2))

rmse.ridge.lambdabest
```

```
[1] 312
```

- Store the values in the RMSE matrix

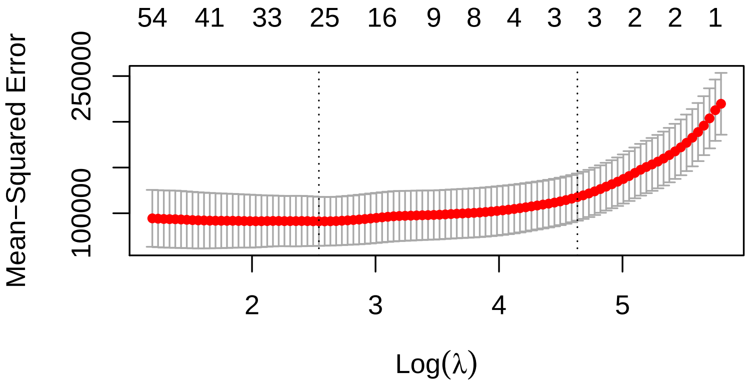
```
RMSE[1:2,] <- c(rmse.ridge.lambda0,rmse.ridge.lambdabest)
```

Lasso Regression

```
lasso.cv=cv.glmnet(x[train,],y[train],alpha=1,
                   nfold=10,type.measure="mse")
```

- Plot

```
plot(lasso.cv)
```



- Prediction using both values of λ, **lambda.min** and **lambda.1se**, the value of λ that gives the most regularized model such that the cross-validated error is within one standard error of the minimum.
- Save the outputs as **lasso.lambdabest** and **lasso.lambda1se**

```
lasso.pred.lambdabest=predict(lasso.cv,newx=x[-train,],
              s=lasso.cv$lambda.min,exact=TRUE,
              x=x[train,],y=y[train])

lasso.pred.lambda1se=predict(lasso.cv,newx=x[-train,],
              s=lasso.cv$lambda.1se,exact=TRUE,
              x=x[train,],y=y[train])
```

- Compute RMSE of the predictions

```
rmse.lasso.lambdabest <- sqrt(mean((y[-train]-
                          lasso.pred.lambdabest)^2))

rmse.lasso.lambda1se <- sqrt(mean((y[-train]-
                          lasso.pred.lambda1se)^2))

rmse.lasso.lambdabest
```

```
[1] 361
```

```
rmse.lasso.lambda1se
```

```
[1] 287
```

- Store in RMSE

```
RMSE[3:4,] <- c(rmse.lasso.lambda1se,rmse.lasso.lambdabest)
RMSE
```

```
                      [,1]
rmse.ridge.lambda0     556
rmse.ridge.lambdabest  312
rmse.lasso.lambda1se   287
rmse.lasso.lambdabest  361
rmse.elnet.lambda1se    NA
rmse.elnet.lambdabest   NA
```

Elastic net

- Now, there are two parameters to tune: λ and α.
- The **glmnet** package allows to tune λ via cross-validation for a fixed α, but it does not support α-tuning.

- Let's write our own loop that does the tuning
- First, we create a common `fold_id`, which just allows us to apply the same CV folds to each model.
- We then create a tuning grid that searches across a range of αs from 0-1, and empty columns where we'll dump our model results into.

```
# maintain the same folds across all models
fold_id <- sample(1:10, size = length(y[train]),
                                    replace=TRUE)

# search across a range of alphas
tuning_grid <- data.frame(
  alpha      = seq(0, 1, by = .1),
  mse_min    = NA,mse_1se    = NA,
  lambda_min = NA,lambda_1se = NA)
```

- Now we can iterate over each α value, apply a CV elastic net, and extract the minimum and one standard error MSE values and their respective λ values.

```r
for(i in seq_along(tuning_grid$alpha)  ) {
  # fit CV model for each alpha value
  fit <- cv.glmnet(x[train,], y[train],
                   alpha = tuning_grid$alpha[i],
                              foldid = fold_id)


  # extract MSE and lambda values
  tuning_grid$mse_min[i]    <- fit$cvm[fit$lambda==
                                           fit$lambda.min]
  tuning_grid$mse_1se[i]    <- fit$cvm[fit$lambda==
                                           fit$lambda.1se]
  tuning_grid$lambda_min[i] <- fit$lambda.min
  tuning_grid$lambda_1se[i] <- fit$lambda.1se
}
```

```r
tuning_grid %>% arrange(mse_min)
```

alpha	mse_min	mse_1se	lambda_min	lambda_1se
0.7	90193	108536	9.03	101.4
0.8	90290	108465	6.87	88.8
1.0	90428	109678	11.57	77.9
0.6	90454	108687	10.54	118.3
0.5	90880	110071	13.87	148.8
0.9	90992	109461	6.11	82.6
0.4	91105	109946	18.17	177.5
0.3	91258	109891	23.12	225.9
0.2	91314	110403	34.69	323.5
0.1	91535	110824	66.22	562.7
0.0	105378	122038	3295.63	11571.6

- Extract the optimum *alpha* and λ values

```
best.index <- which.min(tuning_grid$mse_min)
best.alpha <- tuning_grid[best.index ,"alpha"]
best.lambda <- tuning_grid[best.index ,"lambda_min"]
best.lambda.1se <- tuning_grid[best.index ,"lambda_1se"]

best.alpha
```

```
[1] 0.7
```

```
best.lambda
```

```
[1] 9.03
```

```
best.lambda.1se
```

```
[1] 101
```

- Now that have identified the preferred model, we retrain the model and simply use `predict` to predict the same model on a new data set.

```
elnet.mod <- glmnet(x[train,], y[train],alpha=best.alpha)

elnet.pred.lambdabest <- predict(elnet.mod,
                    s=best.lambda,newx=x[-train,],
                    exact=TRUE,x=x[train,],y=y[train])

elnet.pred.lambda1se <- predict(elnet.mod,
                    s=best.lambda.1se,newx=x[-train,],
                    exact=TRUE,x=x[train,],y=y[train])

rmse.elnet.lambdabest<-sqrt(mean((y[-train] -
                              elnet.pred.lambdabest)^2))
rmse.elnet.lambda1se<-sqrt(mean((y[-train] -
                              elnet.pred.lambda1se)^2))
```

```r
RMSE[5:6,1]<-c(rmse.elnet.lambda1se,
               rmse.elnet.lambdabest)
RMSE
```

```
                      [,1]
rmse.ridge.lambda0     556
rmse.ridge.lambdabest  312
rmse.lasso.lambda1se   287
rmse.lasso.lambdabest  361
rmse.elnet.lambda1se   295
rmse.elnet.lambdabest  357
```