NBA 4920/6921 Lecture 13

Ridge Regression Application

Murat Unal

10/14/2021

```r
rm(list=ls())
options(digits = 3, scipen = 999)
library(tidyverse)
library(ISLR)
library(cowplot)
library(ggcorrplot)
library(stargazer)
library(corrr)
library(lmtest)
library(sandwich)
library(MASS)
library(car)
library(jtools)
library(caret)
library(leaps)
library(future.apply)
library(glmnet)
Hitters <- ISLR::Hitters
Hitters <- na.omit(Hitters)
set.seed(2)
```

Ridge Regression

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

- The **model.matrix()** function is particularly useful for creating x; not only does it produce a matrix corresponding to the 19 predictors but it also automatically transforms any qualitative variables into dummy variables.
- The latter property is important because **glmnet()** can only take numerical, quantitative inputs.

- The **glmnet()** engine has an `alpha` argument that determines what type of model is fit.
- If `alpha=0` then a ridge regression model is fit, and if `alpha=1` then a lasso model is fit. We first fit a ridge regression model.
- By default, the **glmnet()** engine standardizes the variables so that they are on the same scale.

- **glmnet()** will fit ridge models across a wide range of λ values by default, 100 λ values that are data derived
- We could also run the engine for a grid of 100 values ranging from high to low λ

```r
grid=10^seq(10,-2,length=100)
grid[1]
```

```
[1] 10000000000
```

```r
grid[100]
```

```
[1] 0.01
```

- For regression problems set `familty="gaussian"`, and `familty="binomial"` for classification

```
ridge.mod=glmnet(x,y,alpha=0,lambda=grid,family="gaussian")
names(ridge.mod)
```

```
 [1] "a0"          "beta"        "df"          "dim"         "lamb
 [7] "nulldev"     "npasses"     "jerr"        "offset"      "call
```

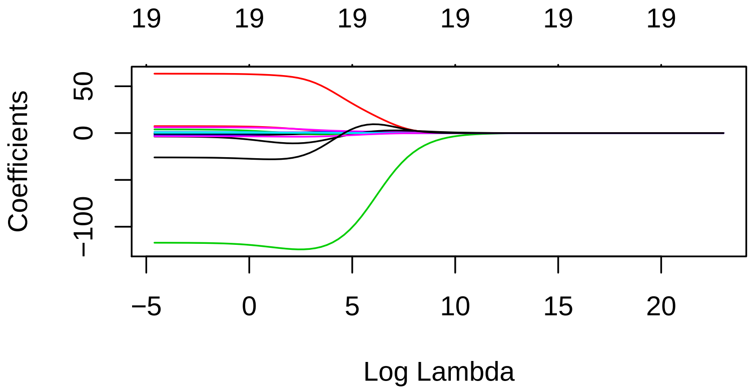
- Associated with each value of λ is a vector of ridge regression coefficients, stored in a matrix that can be accessed by `coef()`.
- In this case, it is a $\mathbf{20 \times 100}$ matrix, with 20 rows (one for each predictor, plus an intercept) and 100 columns (one for each value of λ).

```
dim(coef(ridge.mod))
```

```
[1]  20 100
```

- As λgrows larger, our ridge regression coefficient magnitudes are more constrained.

```
plot(ridge.mod, xvar = "lambda")
```



```r
#Display 1st lambda value
ridge.mod$lambda[1]
```

```
[1] 10000000000
```

```r
# Display coefficients associated with
# 1st lambda value
coef(ridge.mod)[,1]
```

```
    (Intercept)           AtBat            Hits           H
535.92569352090   0.00000005443   0.00000019746   0.0000007
            RBI           Walks           Years          CA
  0.00000035272   0.00000041513   0.00000169771   0.000000
         CHmRun           CRuns            CRBI          CW
  0.00000012972   0.00000003451   0.00000003561   0.000000
      DivisionW         PutOuts         Assists          Er
 -0.00000780726   0.00000002180   0.00000000356  -0.000000
```

- We can use the `predict()` function for a number of purposes. For instance, we can obtain the ridge regression coefficients for a new value of λ , say 50:
- If the desired λ value is not included in the initial fit, then `glmnet()` performs linear interpolation to make predictions for the desired λ value.
- Linear interpolation usually works fine, but we could change this by calling `exact=TRUE`. This way predictions are to be made at values of λ not included in the original fit and the model is refit before predictions are made.

- Let's check if $\lambda = 50$ is used in the original fit

```
any(ridge.mod$lambda==50)
```

```
[1] FALSE
```

- Let's see if there's any difference

```
coef.approax <- predict(ridge.mod, s = 50, type = "coeffic
                        exact = FALSE)
coef.exact <- predict(ridge.mod, s = 50, type = "coefficien
                        exact = TRUE, x=x,y=y)

cbind(coef.approax, coef.exact)
```

```
20 x 2 sparse Matrix of class "dgCMatrix"
                     1        1
(Intercept)   48.76610   48.272
AtBat         -0.35810   -0.353
Hits           1.96936    1.951
HmRun         -1.27825   -1.290
Runs           1.14589    1.156
```

- Let's now split the samples into a training set and a test set in order to estimate the test error of ridge regression

```
train=sample(1:nrow(x), 0.7*nrow(x))
```

- Next we fit a ridge regression model on the training set, and evaluate its RMSE on the test set, using $\lambda = 4$.
- Note the use of the `predict()` function again.
- This time we get predictions for a test set, by replacing `type="coefficients"` with the `newx` argument

```
ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid)
ridge.pred.lambda4=predict(ridge.mod,s=4,newx=x[-train,])
rmse.ridge.lambda4 = sqrt(mean((ridge.pred.lambda4-y[-trai
rmse.ridge.lambda4
```

```
[1] 296
```

- Note that if we had instead simply fit a model with just an intercept, we would have predicted each test observation using the mean of the training observations.
- In that case, we could compute the test set MSE like this:

```
sqrt(mean((mean(y[train])-y[-train])^2))
```

```
[1] 405
```

- We could also get the same result by fitting a ridge regression model with a very large value of λ. Note that 1e10 means 10^{10}

```
ridge.pred.lambdabig=predict(ridge.mod,s=1e10,newx=x[-trai
rmse.ridge.lambdabig <- sqrt(mean((ridge.pred.lambdabig-y[-
rmse.ridge.lambdabig
```

```
[1] 405
```

- So fitting a ridge regression model with $\lambda = 4$ leads to a much lower -train RMSE than fitting a model with just an intercept.
- Let's now check whether there is any benefit to performing ridge regression with $\lambda = 4$ instead of just performing least squares regression.
- Recall that least squares is simply ridge regression with $\lambda = 0$

```
ridge.pred.lambda0=predict(ridge.mod,s=0,exact = TRUE,
          x=x[train,],y=y[train],newx=x[-train,])
rmse.ridge.lambda0 <- sqrt(mean((ridge.pred.lambda0-y[-tra
rmse.ridge.lambda0
```

```
[1] 300
```

- It looks like we are indeed improving over regular least-squares!

Cross-validation

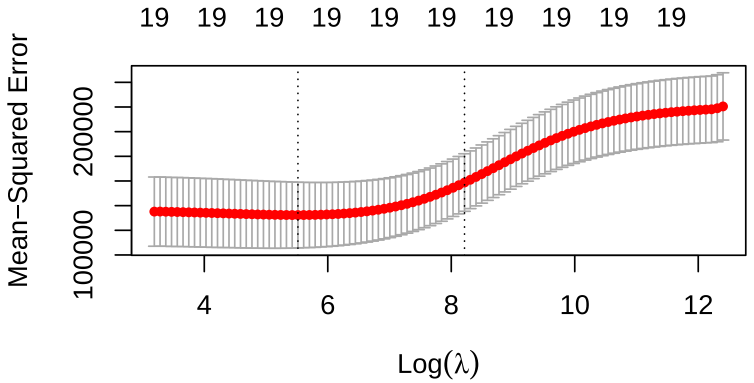
- Instead of arbitrarily choosing $\lambda = 4$, it would be better to use cross-validation to choose the tuning parameter λ.
- We can do this by writing our own function as we did before or we could just use the built-in cross-validation function, `cv.glmnet()`.
- By default, the function performs `10-fold` cross-validation, though this can be changed using the argument `folds`.
- We can define the loss used for cross-validation using `type.measure`

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

```
[1] 249
```

- The first vertical dashed line gives the value of λ that gives minimum mean cross-validated error. The second is the λ that gives an MSE within one standard error of the smallest.

```
plot(cv.out)
```



- Finally, we obtain the coefficients from our fitted model using the value of λ chosen by cross-validation.
- As expected, none of the coefficients are zero—ridge regression does not perform variable selection!

```
predict(ridge.mod,type ="coefficients",s=bestlam,
        exact=TRUE, x=x[train,], y=y[train])
```

```
20 x 1 sparse Matrix of class "dgCMatrix"
                     1
(Intercept)    6.57462
AtBat         -0.00753
Hits           0.94106
HmRun          1.42067
Runs           1.21523
RBI            1.18983
Walks          2.16999
Years          0.43523
CAtBat         0.00963
CHits          0.05862
```

```
final.pred = predict(ridge.mod, newx=x[-train,],s=bestlam,
                     exact=TRUE,x=x[train,],y=y[train])
rmse.ridge.lambdabest <- sqrt(mean((y[-train]-final.pred)^2
rmse.ridge.lambdabest
```

```
[1] 292
```

- Let's compare all test RMSEs:

```r
rmse.ridge.lambdabig
```

```
[1] 405
```

```r
rmse.ridge.lambda4
```

```
[1] 296
```

```r
rmse.ridge.lambda0
```

```
[1] 300
```

```r
rmse.ridge.lambdabest
```

```
[1] 292
```

- Ridge regression indeed obtains the minimum test RMSE!

```
ridge.pred.lambdamedian=predict(ridge.mod,s=10098,newx=x[-t
rmse.ridge.lambdamedian <- sqrt(mean((ridge.pred.lambdamedi
rmse.ridge.lambdamedian
```

```
[1] 358
```