Lecture 4

Shiva Chelluri

20-01-2025

Shrinkage Estimators

Loading Data set

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# glmnet for the conducting ridge and lasso
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(readxl)

cars <- mtcars
```

Ridge Regression

Splitting the data into train-test

```
# setting seed
set.seed(42)

# splitting the sample into parameters matrix and the target variable
x <- model.matrix(mpg~., data= cars)[,-1]
y <- cars$mpg
```

Now we can set $\alpha = 0$, to train on a ridge model as

```
lambdas <- seq(0.1, 100, by = 0.1)
ridge.mod <- glmnet(x, y, alpha = 0, lambda = 1)
```

And you can get the parameter values by the following code:

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

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) 20.814983865
## cyl         -0.312361618
## disp        -0.003796565
## hp          -0.012439077
## drat         1.013868280
## wt          -1.592691035
## qsec         0.224451078
## vs           0.588185454
## am           1.940862840
## gear         0.586899896
## carb        -0.650503206
```

But there are far too many parameters to check and see where the lowest MSE is. So we use the K-Cross Fold Validation.

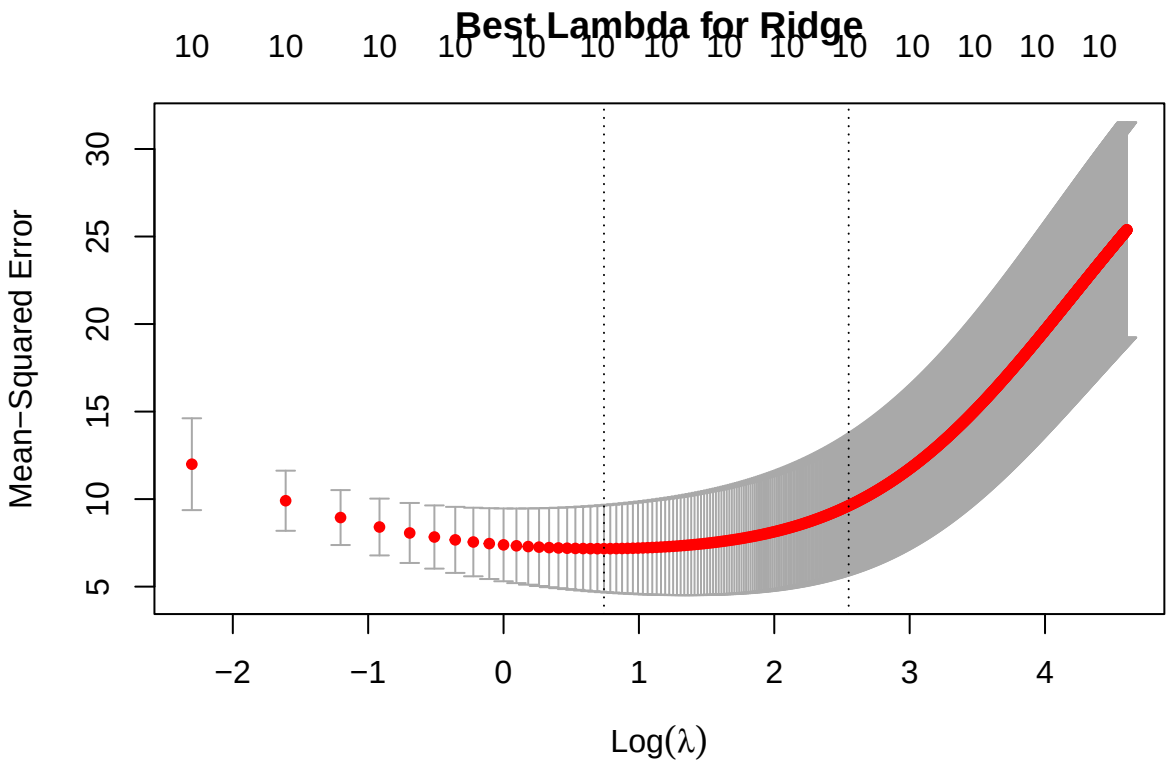
K-Fold Cross Validation

```
ridge.cv <- cv.glmnet(x, y, type.measure = "mse", nfolds = 3, lambda = lambdas, alpha = 0)
ridge.cv
```

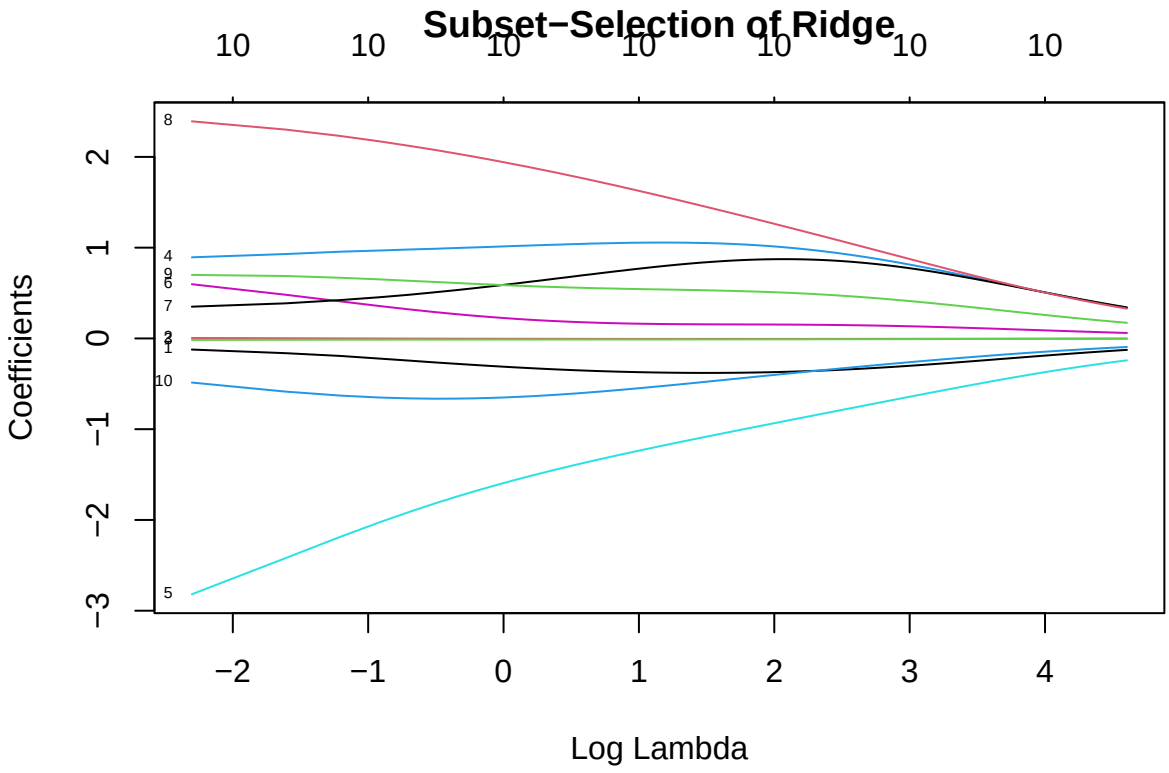
```
##
## Call:  cv.glmnet(x = x, y = y, lambda = lambdas, type.measure = "mse",      nfolds = 3, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min    2.1   980   7.167 2.483      10
## 1se   12.8   873   9.622 3.831      10
```

So we know the best $\lambda = 0.51$ for the given data. We can plot is as:

```
plot(ridge.cv, main="Best Lambda for Ridge")
```



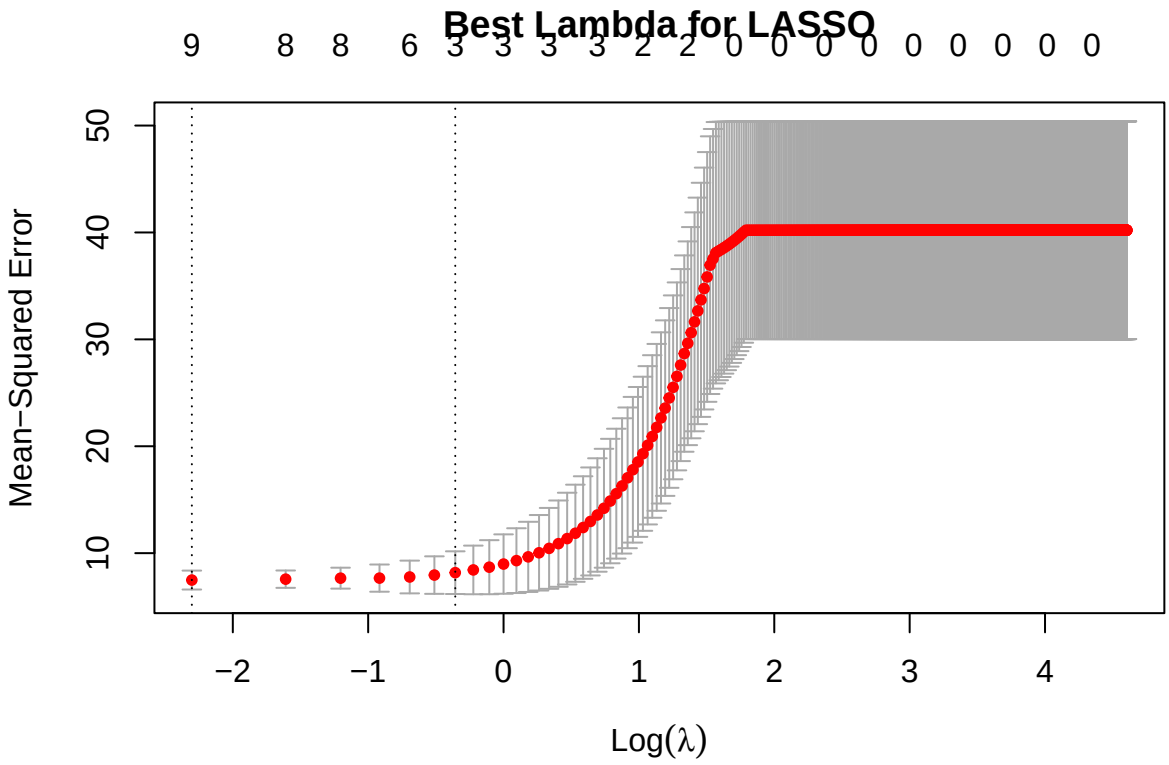

```
plot(ridge.cv$glmnet.fit, xvar="lambda",label = T, main="Subset-Selection of Ridge")
```



LASSO Regression

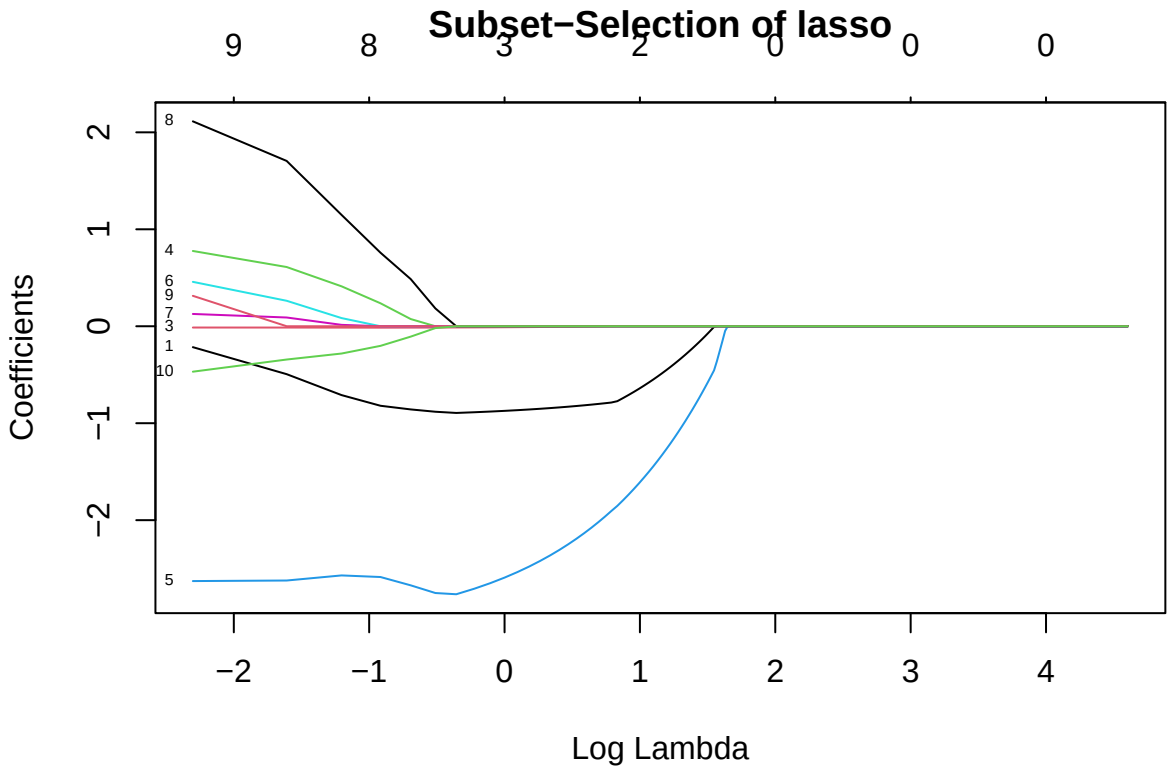
Will be using the same methodology as before but set $\alpha = 1$ and as the following:

```
lambdas <- seq(0.1, 100, by = 0.1)
lasso.cv <- cv.glmnet(x, y, type.measure = "mse", nfolds = 3, lambda = lambdas, alpha = 1)
plot(lasso.cv, main = "Best Lambda for LASSO")
```



Now we can also see by how much what parameters are shrunk:

```
plot(lasso.cv$glmnet.fit, xvar="lambda",label = T, main="Subset-Selection of lasso")
```



So we can see that Ridge shrinks the coefficients "smoothly" as seen above but LASSO shrinks certain betas to 0.

But the choice in the end of the day the choice of the models depend on the MSE and they are the following:

```
print(paste0("MSE of Ridge: ", min(ridge.cv$cvm), "  MSE of LASSO: ", min(lasso.cv$cvm)))
```

```
## [1] "MSE of Ridge: 7.1673902746854  MSE of LASSO: 7.48284055931801"
```

so Ridge performs better in terms of MSE compared to the LASSO.