Lecture 3

Shiva Chelluri

20-01-2025

Shrinkage Estimators

Loading Data set

```r
# glmnet for the conducting ridge and lasso
# install.packages("glmnet")
library(glmnet)
```

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

```
## Loaded glmnet 4.1-8
```

```r
# loading dataset
cars <- mtcars
```

Ridge Regression

Splitting the data into train-test

```r
# setting seed for reproducibility
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

```r
# making a ridge model with an arbitrary lambda = 1 as an example
ridge.mod <- glmnet(x, y, alpha = 0, lambda = 1)
```

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

```r
# getting the coefficients of the model
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

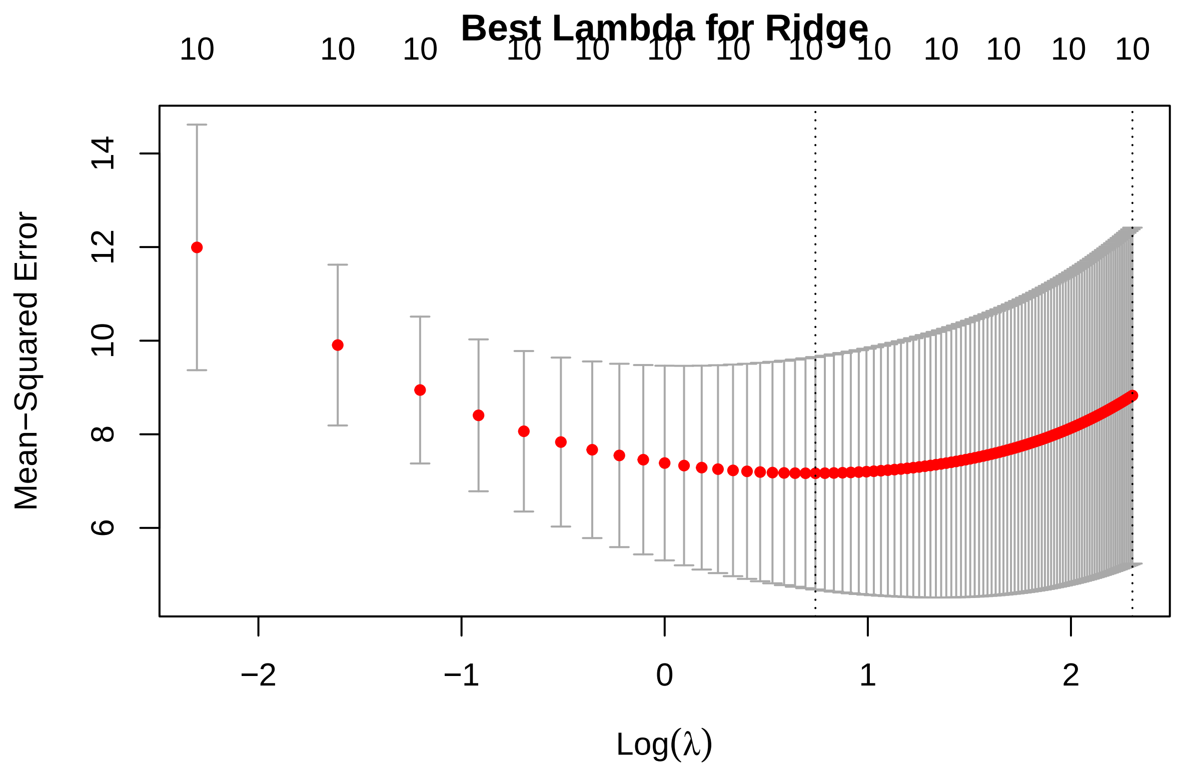
We can conduct k-Fold cross validation in using the in-built method called `nfolds=` to declare how many folds do we want in the K-Fold cross validation. I will choose `nfolds=3` as it will randomly split the data into folds of 10 (approximately given that the data set has only 30 observations).

```
lambda_grid <- seq(0.1, 10, 0.1)
ridge.cv <- cv.glmnet(x, y, type.measure = "mse", nfolds = 3,
                      lambda = lambda_grid, alpha = 0)
ridge.cv
```

```
##
## Call:  cv.glmnet(x = x, y = y, lambda = lambda_grid, type.measure = "mse",      nfolds = 3, alpha = 
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min    2.1    80   7.167 2.483      10
## 1se   10.0     1   8.827 3.588      10
```

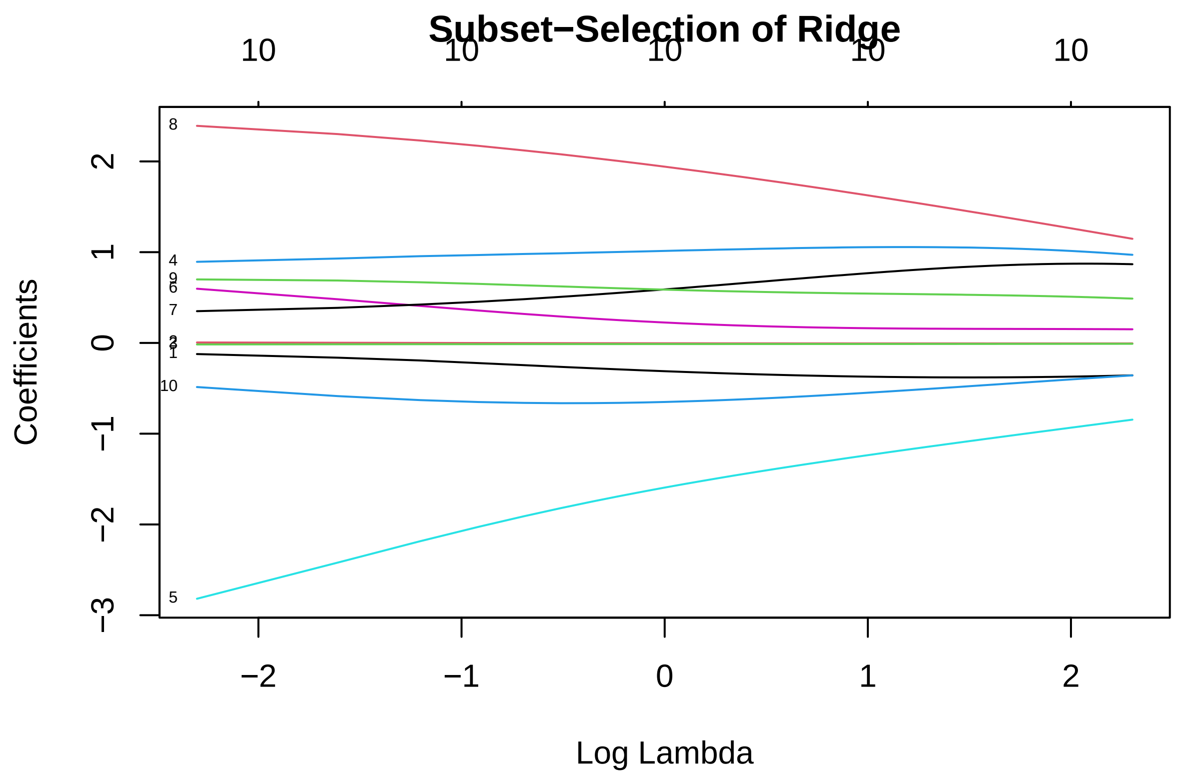
So we know the best $\lambda = 2.1$ (it might different in the LaTeX published version) for the given data. We can plot it using the in-built plot as:

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



Then, we can also see how ridge performs shrinkage on the coefficients.

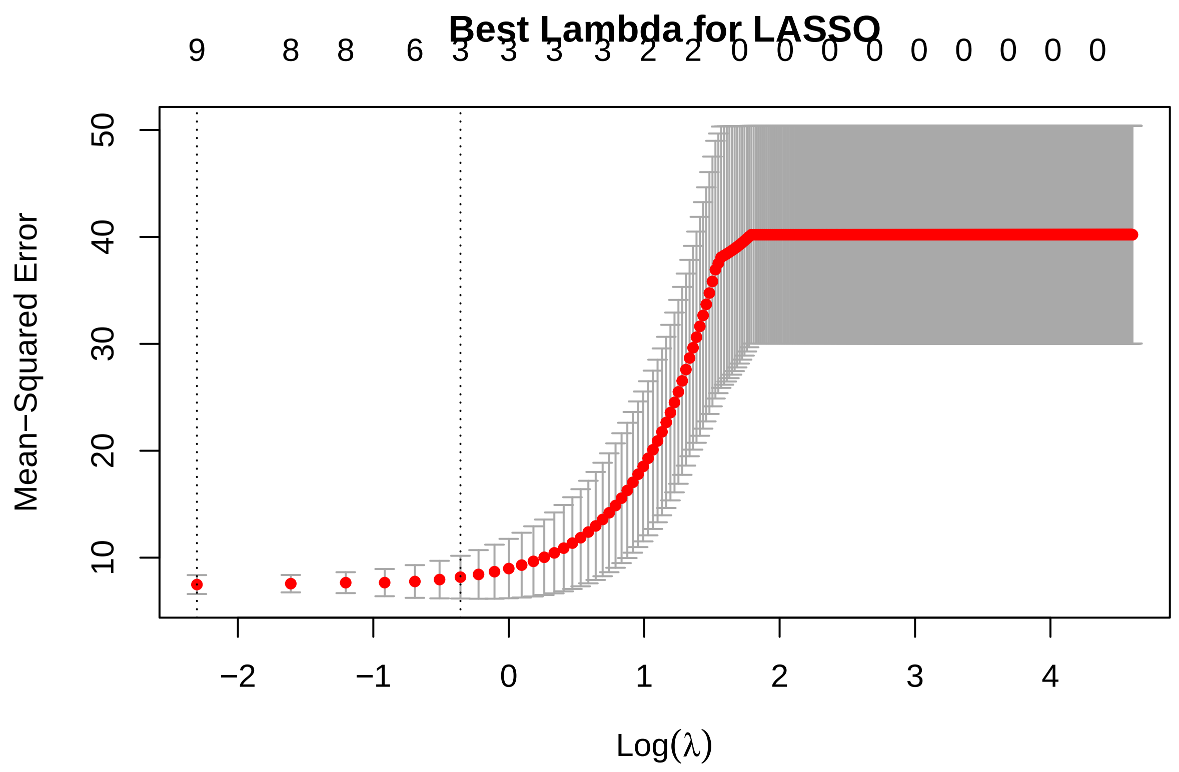
```r
plot(ridge.cv$glmnet.fit, xvar="lambda",label = T, main="Subset-Selection of Ridge")
```



LASSO Regression

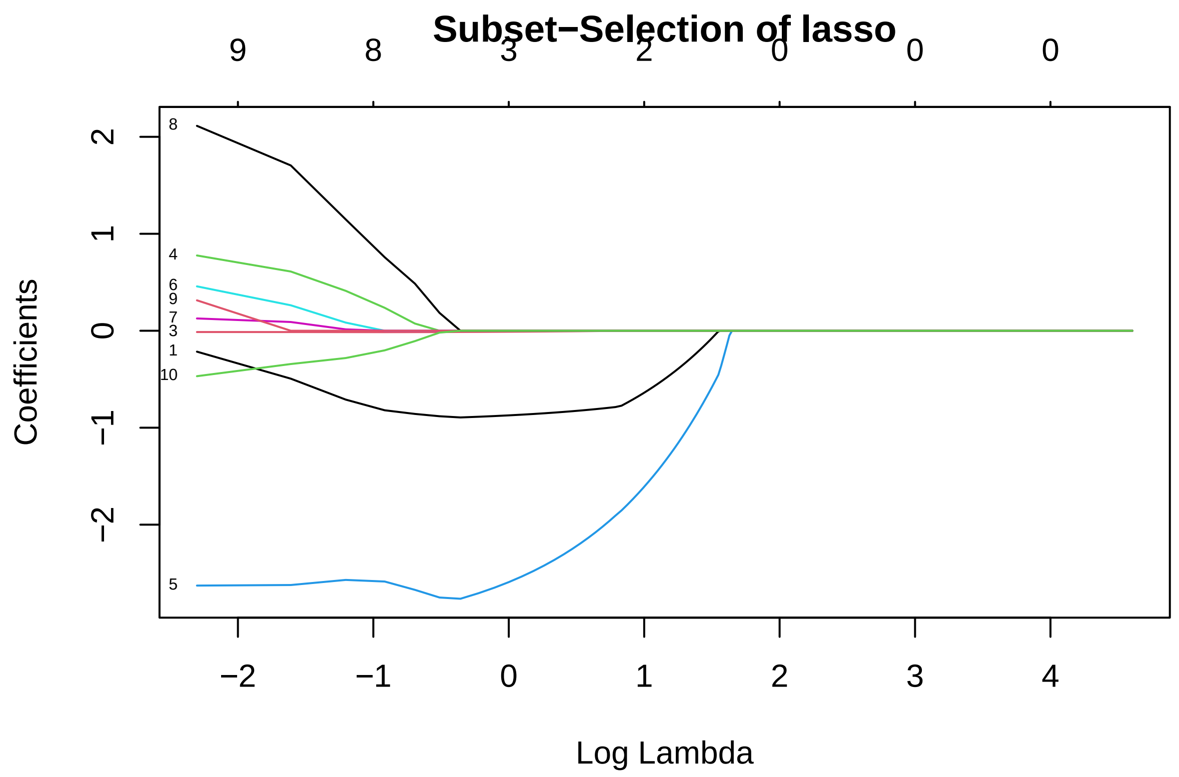
Will be using the same methodology as before but set $\alpha = 1$ and as the following:

```r
lambda_grid <- seq(0.1, 100, by = 0.1)
lasso.cv <- cv.glmnet(x, y, type.measure = "mse", nfolds = 3,
                      lambda = lambda_grid, 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 few plots above but LASSO shrinks "more roughly" with certain β 's to 0 as seen here. (This is due to the modified least-angle regression algorithm but it is not in syllabus)

NOTE: The numbers on the top of the plots represent the number of non-zero β values. So with high-enough λ, all the βs are set to 0. And we can see the main disadvantage of Ridge, that it does not drop any variables and requires $\lambda \to \infty$.

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)))
```

```
## [1] "MSE of Ridge: 7.16739027468535"
```

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

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

so Ridge performs better in terms of MSE compared to the LASSO. (in the iteration I am running but it might not be in the case in your run because of the small sample and exact random sample used.)

Additionally, we can see from the training the exact λ that performs the best for both by the following code:

```
print(paste0("Optimal Lambda of Ridge: ", ridge.cv$lambda.min))
```

```
## [1] "Optimal Lambda of Ridge: 2.1"
```

```r
print(paste0("Optimal Lambda of LASSO: ", lasso.cv$lambda.min))
```

```
## [1] "Optimal Lambda of LASSO: 0.1"
```

So, ridge regression penalizes the coefficient more than the LASSO.