Stat 427/627 Statistical Machine Learning

In-class Lab (Class 7, Topic 2): Collinearity and Shrinkage Methods in Regression

Contents

Recall the `Auto` data set in the `ISLR2` package. This data frame has 392 observations on the following 9 variables.

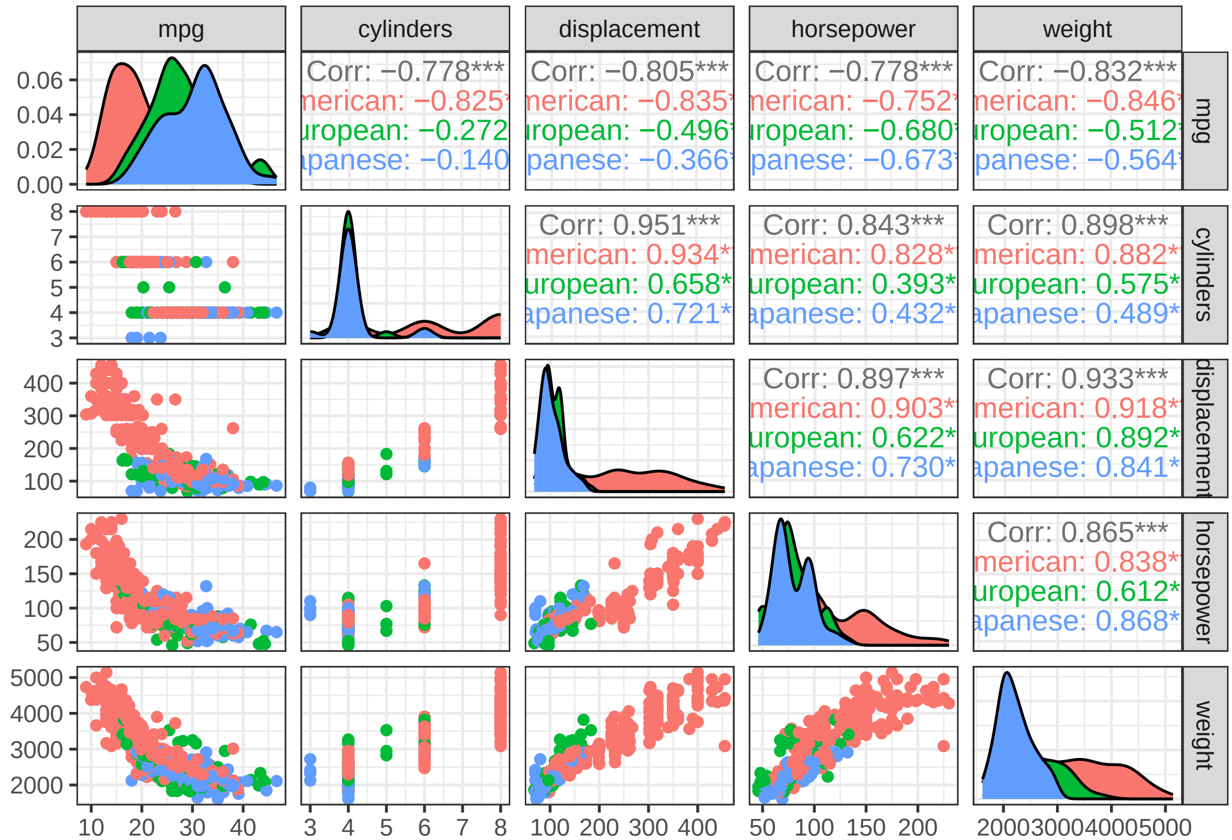
- `mpg`: miles per gallon
- `cylinders`: Number of cylinders between 4 and 8
- `displacement`: Engine displacement (cu. inches)
- `horsepower`: Engine horsepower
- `weight`: Vehicle weight (lbs.)
- `acceleration`: Time to accelerate from 0 to 60 mph (sec.)
- `year`: Model year (modulo 100)
- `origin`: Origin of car (1. American, 2. European, 3. Japanese)
- `name`: Vehicle name

```
library(ISLR2)
colnames(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

```
auto.data <- Auto
auto.data$country <- factor(auto.data$origin, labels = c("American", "European",
                                                          "Japanese"))
library("ggplot2")
library("GGally")

ggpairs(auto.data, columns = 1:5, ggplot2::aes(colour=country)) + theme_bw()
```



1 Collinearity: VIF

```r
# install.package(car)
library(car)
```

```
## Loading required package: carData
```

```r
mpg.lm <- lm(mpg ~ cylinders + displacement + horsepower + weight +
               acceleration + year, data=auto.data )

vif(mpg.lm)
```

```
##    cylinders displacement   horsepower       weight acceleration         year
##    10.633049    19.641683     9.398043    10.731681     2.625581     1.244829
```

```r
mpg.lm2 <- lm(mpg ~ cylinders + displacement + horsepower + weight +
                acceleration + year + country, data=auto.data )

vif(mpg.lm2)  # With categorical predictors
```

```
##                   GVIF Df GVIF^(1/(2*Df))
## cylinders    10.737771  1        3.276854
## displacement 22.937950  1        4.789358
## horsepower    9.957265  1        3.155513
## weight       11.074349  1        3.327814
## acceleration  2.625906  1        1.620465
```

```
## year          1.301373  1        1.140777
## country       2.096060  2        1.203236
```

2 Automatic Variable Selection Algorithms (review)

2.1 Stepwise Selection

Function `step()` conducts stepwise for linear model and generalized linear models using AIC or BIC.

```r
mpg.lm2
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + year + country, data = auto.data)
##
## Coefficients:
##     (Intercept)        cylinders     displacement       horsepower
##       -17.95460         -0.48971          0.02398         -0.01818
##          weight     acceleration             year  countryEuropean
##        -0.00671          0.07910          0.77703          2.63000
## countryJapanese
##         2.85323
```

```r
step(mpg.lm2, direction="both") # AIC
```

```
## Start:  AIC=946.48
## mpg ~ cylinders + displacement + horsepower + weight + acceleration +
##     year + country
##
##                Df Sum of Sq    RSS     AIC
## - acceleration  1      7.09 4194.5  945.14
## - horsepower    1     19.24 4206.6  946.28
## <none>                      4187.4  946.48
## - cylinders     1     25.41 4212.8  946.85
## - displacement  1    107.32 4294.7  954.40
## - country       2    355.96 4543.3  974.46
## - weight        1   1147.04 5334.4 1039.39
## - year          1   2461.64 6649.0 1125.74
##
## Step:  AIC=945.14
## mpg ~ cylinders + displacement + horsepower + weight + year +
##     country
##
##                Df Sum of Sq    RSS     AIC
## <none>                      4194.5  945.14
## - cylinders     1     26.85 4221.3  945.64
## + acceleration  1      7.09 4187.4  946.48
## - horsepower    1     58.80 4253.3  948.60
## - displacement  1    102.96 4297.4  952.65
## - country       2    357.11 4551.6  973.17
## - weight        1   1372.52 5567.0 1054.11
## - year          1   2455.71 6650.2 1123.81
```

```
##
## Call:
```

```
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     year + country, data = auto.data)
##
## Coefficients:
##     (Intercept)        cylinders     displacement       horsepower
##       -16.33231         -0.50277          0.02337         -0.02500
##          weight             year  countryEuropean  countryJapanese
##        -0.00646          0.77388          2.63452          2.85736
```

To use BIC as the selection criteria, set argument `k = (sample size)`. Though the output still lists "AIC =", it is actually the BIC value.

```r
step(mpg.lm2, direction="both", k = log(length(mpg.lm2$fitted.values)))
```

```
## Start:  AIC=982.22
## mpg ~ cylinders + displacement + horsepower + weight + acceleration +
##     year + country
##
##                Df Sum of Sq    RSS     AIC
## - acceleration  1      7.09 4194.5  976.91
## - horsepower    1     19.24 4206.6  978.05
## - cylinders     1     25.41 4212.8  978.62
## <none>                      4187.4  982.22
## - displacement  1    107.32 4294.7  986.17
## - country       2    355.96 4543.3 1002.26
## - weight        1   1147.04 5334.4 1071.16
## - year          1   2461.64 6649.0 1157.51
##
## Step:  AIC=976.91
## mpg ~ cylinders + displacement + horsepower + weight + year +
##     country
##
##                Df Sum of Sq    RSS     AIC
## - cylinders     1     26.85 4221.3  973.44
## - horsepower    1     58.80 4253.3  976.40
## <none>                      4194.5  976.91
## - displacement  1    102.96 4297.4  980.45
## + acceleration  1      7.09 4187.4  982.22
## - country       2    357.11 4551.6  997.00
## - weight        1   1372.52 5567.0 1081.91
## - year          1   2455.71 6650.2 1151.61
##
## Step:  AIC=973.44
## mpg ~ displacement + horsepower + weight + year + country
##
##                Df Sum of Sq    RSS     AIC
## - horsepower    1     50.63 4272.0  972.15
## <none>                      4221.3  973.44
## - displacement  1     79.90 4301.2  974.82
## + cylinders     1     26.85 4194.5  976.91
## + acceleration  1      8.53 4212.8  978.62
## - country       2    342.93 4564.3  992.12
## - weight        1   1437.29 5658.6 1082.34
## - year          1   2462.30 6683.6 1147.60
##
```

```
## Step:  AIC=972.15
## mpg ~ displacement + weight + year + country
##
##                Df Sum of Sq    RSS     AIC
## - displacement  1     38.44 4310.4  969.69
## <none>                      4272.0  972.15
## + horsepower    1     50.63 4221.3  973.44
## + acceleration  1     44.74 4227.2  973.99
## + cylinders     1     18.68 4253.3  976.40
## - country       2    296.95 4568.9  986.55
## - weight        1   1620.18 5892.1 1092.22
## - year          1   2729.74 7001.7 1159.85
##
## Step:  AIC=969.69
## mpg ~ weight + year + country
##
##                Df Sum of Sq     RSS     AIC
## <none>                       4310.4  969.69
## + displacement  1      38.4  4272.0  972.15
## + acceleration  1       9.5  4300.9  974.79
## + horsepower    1       9.2  4301.2  974.82
## + cylinders     1       1.1  4309.3  975.56
## - country       2     258.5  4569.0  980.58
## - year          1    2786.7  7097.1 1159.19
## - weight        1    5712.8 10023.2 1294.51
```

```
##
## Call:
## lm(formula = mpg ~ weight + year + country, data = auto.data)
##
## Coefficients:
##     (Intercept)           weight             year  countryEuropean
##      -18.306944        -0.005887         0.769849         1.976306
## countryJapanese
##        2.214534
```

Remarks:

- See the help file for other arguments in `step()`. For example, you can specify the `scope` of the selection.
- Other packages may have their own stepwise selection function. E.g., `stepAIC()` in pacakge `MASS`.

2.2 Best Subset Algorithem

Use `regsubsets()` from package {`leaps`}.

```r
# Install.package(leaps)
library(leaps)
```

```r
mpg.sub <- regsubsets(mpg ~ cylinders + displacement + horsepower +
                        weight + acceleration + year + country, auto.data)

summary(mpg.sub)
```

```
## Subset selection object
## Call: regsubsets.formula(mpg ~ cylinders + displacement + horsepower +
##     weight + acceleration + year + country, auto.data)
```

```
## 8 Variables  (and intercept)
##                  Forced in Forced out
## cylinders            FALSE      FALSE
## displacement         FALSE      FALSE
## horsepower           FALSE      FALSE
## weight               FALSE      FALSE
## acceleration         FALSE      FALSE
## year                 FALSE      FALSE
## countryEuropean      FALSE      FALSE
## countryJapanese      FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          cylinders displacement horsepower weight acceleration year
## 1  ( 1 ) " "       " "          " "        "*"    " "          " "
## 2  ( 1 ) " "       " "          " "        "*"    " "          "*"
## 3  ( 1 ) " "       " "          " "        "*"    " "          "*"
## 4  ( 1 ) " "       " "          " "        "*"    " "          "*"
## 5  ( 1 ) " "       "*"          " "        "*"    " "          "*"
## 6  ( 1 ) " "       "*"          "*"        "*"    " "          "*"
## 7  ( 1 ) "*"       "*"          "*"        "*"    " "          "*"
## 8  ( 1 ) "*"       "*"          "*"        "*"    "*"          "*"
##          countryEuropean countryJapanese
## 1  ( 1 ) " "             " "
## 2  ( 1 ) " "             " "
## 3  ( 1 ) " "             "*"
## 4  ( 1 ) "*"             "*"
## 5  ( 1 ) "*"             "*"
## 6  ( 1 ) "*"             "*"
## 7  ( 1 ) "*"             "*"
## 8  ( 1 ) "*"             "*"
```

For models with the same number of parameters (p), the "best" model is selected based on the the lowest Sum of Squared Residuals (aka. SSE, Residual Sum of Squares, RSS).

Use desired selection criteria to determine the optimal number of parameters (p).

```r
mpg.subsum <- summary(mpg.sub)
mpg.subsum$adjr2
```

```
## [1] 0.6918423 0.8071941 0.8107755 0.8171643 0.8183256 0.8200125 0.8206916
## [8] 0.8205274
```

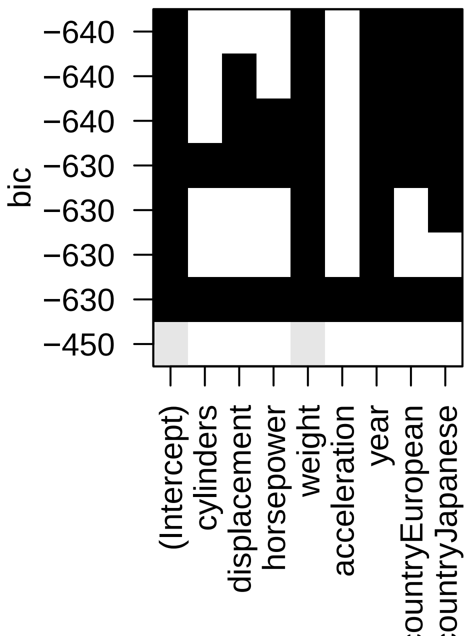
```r
mpg.subsum$bic
```

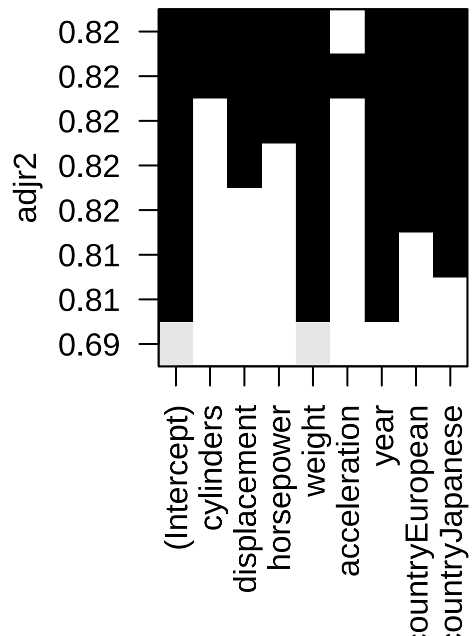
```
## [1] -450.5016 -629.3564 -631.7442 -640.2482 -637.7889 -636.4914 -633.0215
## [8] -627.7135
```

```r
mpg.subsum$cp
```

```
## [1] 281.637026  31.899439  25.082435  12.251831  10.735481   8.104512   7.648634
## [8]   9.000000
```

```r
par(mfrow=c(1, 2))
plot(mpg.sub)
plot(mpg.sub, scale = "adjr2")
```





3 Shrinkage Methods: Ridge regression and Lasso with `glmnet`

Function `glmnet()` in package `glmnet` can fit both Ridge regression and LASSO (least absolute shrinkage and selection operator). The package also has a function `cv.glmnet()` that conducts K-fold cross-validation.

```
# install.packages("glmnet")
library(glmnet)
```

For Ridge regression and LASSO, `glmnet()` and `cv.glmnet()` need, at least, the following arguments:

- x: the matrix of predictors terms. (The design matrix of the linear model, but without the column of 1s.)
- y: the vector of response.
- alpha: 0 for Ridge regression, 1 for LASSO.
- lambda: often a decreasing sequence of positive numbers.

3.1 Ridge Regression

3.1.1 Fit the model with `glmnet(x, y, alpha=0, lambda)`

```
mpg.lm2
```

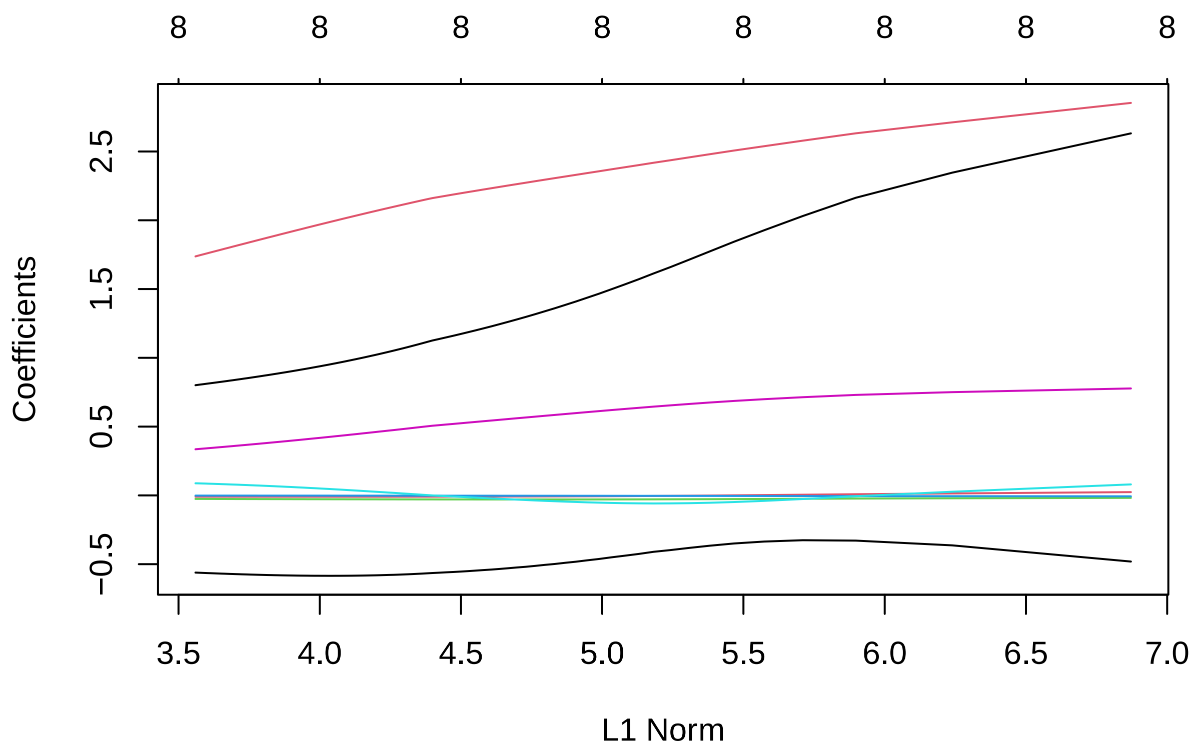
```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
```

```
##     acceleration + year + country, data = auto.data)
##
## Coefficients:
##     (Intercept)        cylinders     displacement       horsepower
##       -17.95460         -0.48971          0.02398         -0.01818
##          weight     acceleration             year  countryEuropean
##        -0.00671          0.07910          0.77703          2.63000
## countryJapanese
##         2.85323
```

```r
auto.X <- model.matrix(mpg.lm2)[, -1]  # Remove the first 1 column of 1's
dim(auto.X)
```

```
## [1] 392   8
```

```r
auto.ridge <- glmnet(x = auto.X, y=auto.data$mpg, alpha=0, lambda = seq(10, 0, by= -0.1))
plot(auto.ridge)
```



- What's in the above plot?
- Is it really L1 Norm (as shown in the plot)?
  - NO! Since we set `alpha=0` in the `glmnet()` function. The horizontal axis is the L2 Norm $\sum_{j=1}^{p}(\beta_j)^2$.

3.1.2 K-fold cross-validation (default $K = 10$) with `cv.glmnet()`

```r
set.seed(2023)
auto.ridgeCV <- cv.glmnet(x=auto.X, y = auto.data$mpg, alpha=0,
```

```r
                          lambda=seq(10, 0, by=-0.1))
auto.ridgeCV
```

```
##
## Call:  cv.glmnet(x = auto.X, y = auto.data$mpg, lambda = seq(10, 0,      by = -0.1), alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min      0   101   11.35 0.9068       8
## 1se      1    91   12.19 1.1264       8
```

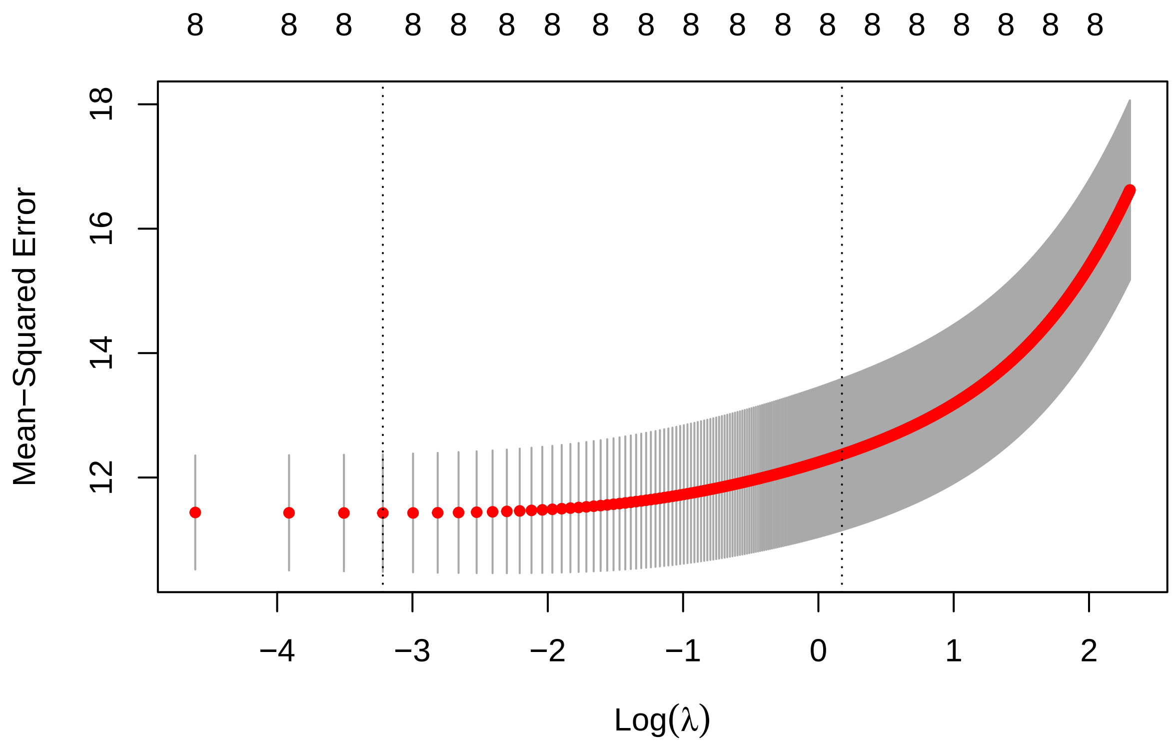
```r
auto.ridgeCV <- cv.glmnet(x=auto.X, y = auto.data$mpg, alpha=0,
                          lambda=seq(10, 0, by=-0.01))
auto.ridgeCV
```

```
##
## Call:  cv.glmnet(x = auto.X, y = auto.data$mpg, lambda = seq(10, 0,      by = -0.01), alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min   0.04   997   11.43 0.9461       8
## 1se   1.19   882   12.37 1.2189       8
```

```r
names(auto.ridgeCV)
```

```
##  [1] "lambda"     "cvm"        "cvsd"       "cvup"       "cvlo"
##  [6] "nzero"      "call"       "name"       "glmnet.fit" "lambda.min"
## [11] "lambda.1se" "index"
```

```r
plot(auto.ridgeCV)
```



3.1.3 Estimate the coefficients using `predict(..., type="coefficients")`

```
lambda.opt <- auto.ridgeCV$lambda.min
lambda.opt
```

```
## [1] 0.04
```

```
predict(auto.ridge, s = lambda.opt, type="coefficients")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                            s1
## (Intercept)     -17.306347741
## cylinders        -0.433900198
## displacement      0.019984317
## horsepower       -0.019202445
## weight           -0.006350692
## acceleration      0.058553699
## year              0.766611471
## countryEuropean   2.517553514
## countryJapanese   2.796552339
```

3.1.4 Predict the response using `predict(..., type="response")`

```
# Generate a "new" set of X-value for the sample.
auto.new <- auto.X[sample(nrow(auto.X), 3), ]
dim(auto.new)
```

```
## [1] 3 8
```

```
lambda.opt
```

```
## [1] 0.04
```

```
predict(auto.ridge, newx=auto.new, s=lambda.opt, type="response")
```

```
##           s1
## 231 16.19851
## 258 23.41549
## 336 30.56023
```

3.2 LASSO Regression

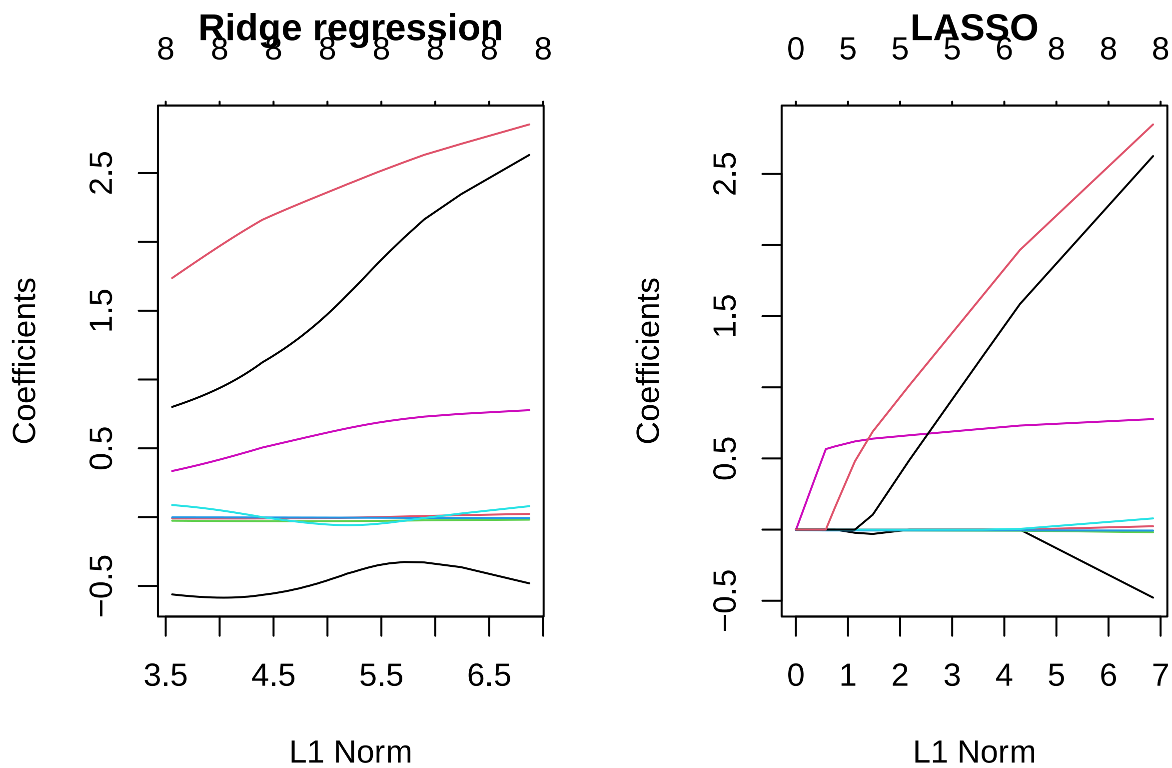
Set `alpha=1` in `glmnet()` and `cv.glmnet()` to fit LASSO.

```
head(auto.X, 2)
```

```
##   cylinders displacement horsepower weight acceleration year countryEuropean
## 1         8          307        130   3504         12.0   70               0
## 2         8          350        165   3693         11.5   70               0
##   countryJapanese
## 1               0
## 2               0
```

```
auto.lasso <- glmnet(x = auto.X, y=auto.data$mpg, alpha=1, lambda = seq(10, 0, by= -0.1))
par(mfrow=c(1, 2))
plot(auto.ridge, main="Ridge regression")
plot(auto.lasso, main="LASSO")
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



- Compare the above plots:

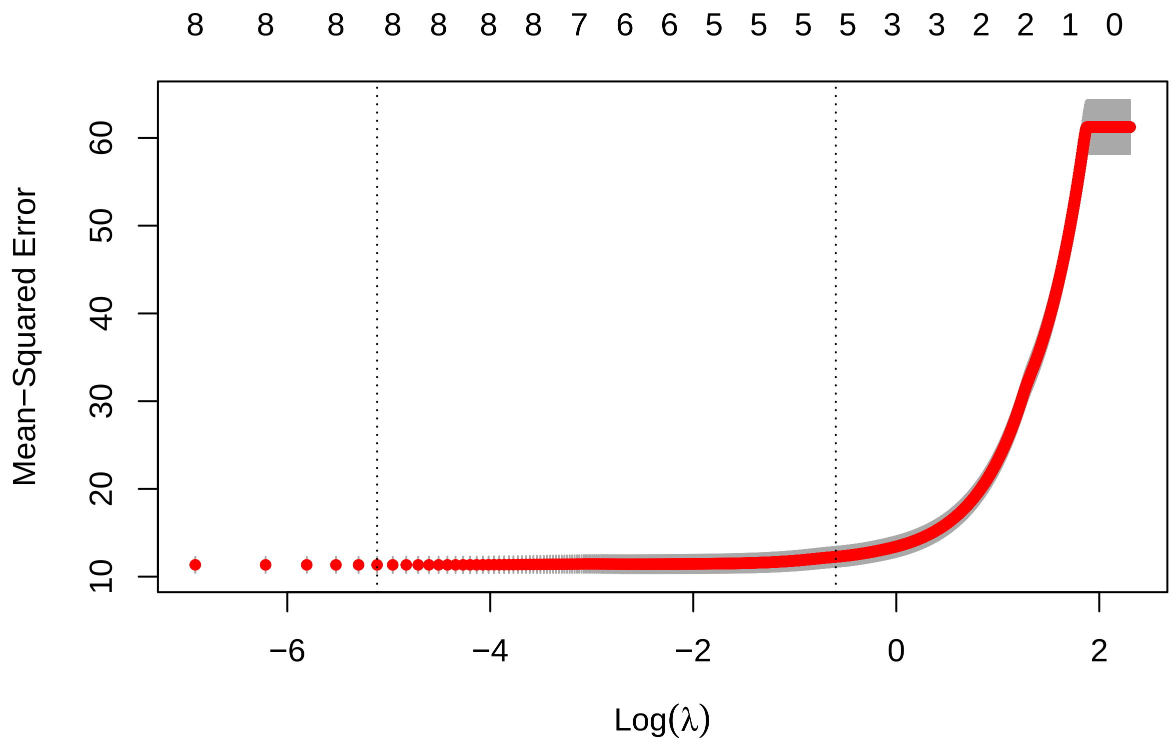
- Which one is really using L1 Norm?
- What are the similarities?
- What is the main difference?

Fitting the model, obtaining the regression coefficients and predicting the response.

```
set.seed(2023)
auto.lassoCV <- cv.glmnet(x=auto.X, y = auto.data$mpg, alpha=1,
                          lambda=seq(10, 0, by=-0.001))
auto.lassoCV
```

```
##
## Call:  cv.glmnet(x = auto.X, y = auto.data$mpg, lambda = seq(10, 0,      by = -0.001), alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min  0.006  9995   11.34 0.9191       8
## 1se  0.551  9450   12.26 1.1264       5
```

```
plot(auto.lassoCV)
```




```
lambda.opt <- auto.lassoCV$lambda.min
lambda.opt
```

```
## [1] 0.006
```

```
predict(auto.lasso, s = lambda.opt, type="coefficients")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                           s1
## (Intercept)     -17.800644562
## cylinders        -0.449440589
## displacement      0.022357124
## horsepower       -0.017541460
## weight           -0.006647528
## acceleration      0.074342410
## year              0.774185494
## countryEuropean   2.562802913
## countryJapanese   2.794885759
```

```
auto.new
```

```
##     cylinders displacement horsepower weight acceleration year countryEuropean
## 231         8          350        170   4165         11.4   77               0
## 258         6          232         90   3210         17.2   78               0
## 336         4          122         88   2500         15.1   80               1
##     countryJapanese
## 231               0
## 258               0
```

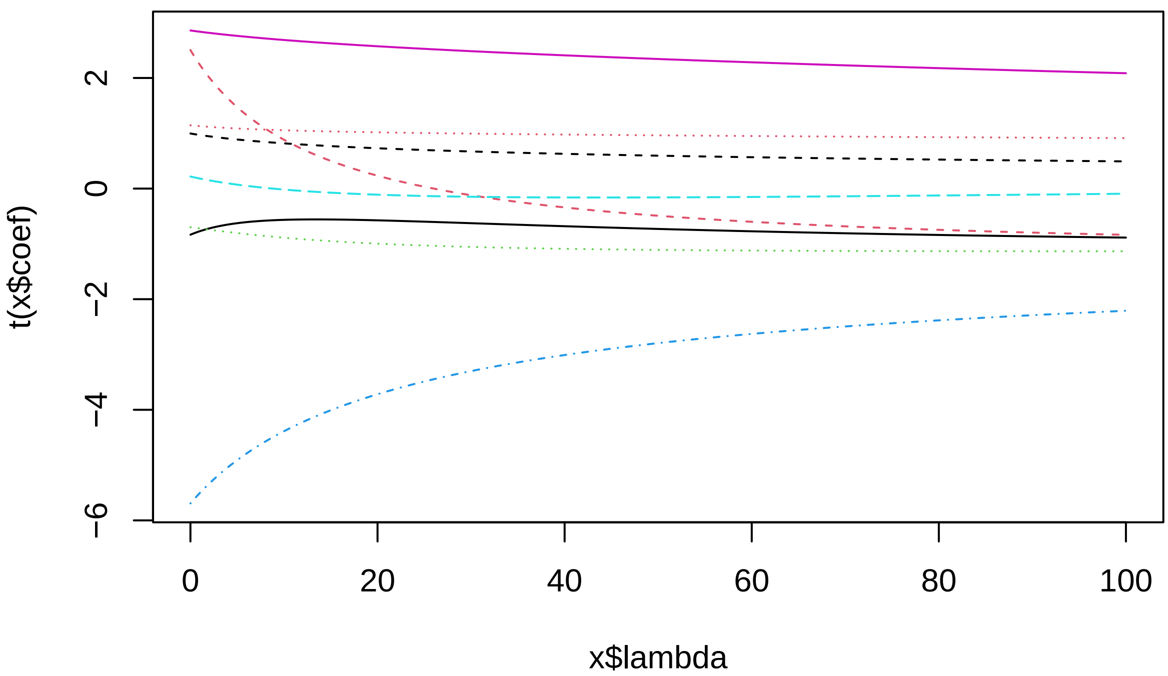
```
## 336              0
```

```r
predict(auto.lasso, newx=auto.new, s=lambda.opt, type="response")
```

```
##           s1
## 231 16.21961
## 258 23.43743
## 336 30.58691
```

3.3 (Extra) Another Ridge Regression function: `lm.ridge()` in package `MASS`

```r
library(MASS)
auto.ridge2 <- lm.ridge(mpg ~ cylinders + displacement + horsepower + weight +
                          acceleration + year + country, data = auto.data,
                        lambda = seq(0, 100, 0.01))
plot(auto.ridge2)
```




```r
auto.ridge2 <- lm.ridge(mpg ~ cylinders + displacement + horsepower + weight +
                          acceleration + year + country, data = auto.data,
                        lambda = seq(0, 4, 0.001))

select(auto.ridge2)
```

```
## modified HKB estimator is 1.30179
## modified L-W estimator is 1.309865
## smallest value of GCV  at 0.929
```

```r
plot(auto.ridge2$lambda, auto.ridge2$GCV)
```