Shrinkage Methods

2020-04-22

Contents

Chapter 1

Motivation

1.1 The ordinary least squared model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon$$

is the most commonly used method to describe the relationship between the response variable Y and a set of variables X. However, the OLS model is not perfect and it faces two major criticism and one major drawback:

- First, the OLS model usually has low prediction accuracy because it often has large variance and low bias.
- The second weakness is efficient interpretation. If we have a dataset with large number of predictors, we would usually choose a small subset of predictors which are strongly associated with the response variable; thus, it seems to be a loss that we still include the weak predictors in our model.
- Also, OLS does not work when there are more predictors than data points. That being said, it does not generate a unique solution($p > n$).

However, the linear regression model is easy to regress and interpret; therefore, we would still use linear regression method but we want to modify it in a way such that we could have higher prediction accuracy and stronger interpretation.

1.2 Several solutions proposed by statisticians

The first method is subset selection:

- Best subset selection algorithm: This is a sort of brute force technique that in real life no actual statistician would use for subset selection. In order to perform best subset selection, we generate the power set of all p variables in a dataset, and then we select the one with the least error.
- backward stepwise selection: This is a more feasible version of subset selection. We first generate the full model of the dataset. Then among all the features, we test the deletion of each variable using a chosen model fit criterion, deleting the variable (if any) whose loss gives the most statistically insignificant deterioration of the model fit, and repeating this process until no further variables can be deleted without a statistically insignificant loss of fit.
- More stepwise selection: forward stepwise selection, bidirectional elimination$\cdots$.

Problems with subset selection algorithms:

- They are mostly computationally expensive.
- They are discreet selection processes(1 variable at a time).
- The algorithms only do locally(each step) best choices. Once a variable is deleted, it is never going back in.

1.3 Shrinkage

There are two different kinds of shrinkage algorithms: * L1 shrinkage methods: only shrinks the coefficients but does not perform variable selection. * L2 shrinkage methods: performs both shrinking coefficients and variable selection(shrink some variables to 0). We are going to start by looking at an L1 shrinkage method, and then look at the most classic L2 shrinkage method: Lasso.

Chapter 2

Ridge Regression

2.1 Definition

Recall that the ordinary least squares model estimates coefficients with the goal of minimizing the sum of the squared residuals:

$$RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$$

Since the shrinkage approach takes into account of variable selection, the ridge regression estimates the coefficients that minimizes the combination of RSS and sum of the squared coefficients:

2.2 Ridge Regression:

$$\beta^{\hat{ridge}} = arg \min_{\beta}\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2\} \quad (1)$$

Which is the same as:

$$\beta^{\hat{ridge}} = arg \min_{\beta}\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2\}\text{subject to}\sum_{j=1}^{p} \beta_j^2 \leq t$$

where there is a one-to-one corresondence between the parameters λ and t.

λ is the tuning parameter, and $\lambda \sum_{j=1}^{p} \beta_j^2$ is the skrinkage penalty. If λ is zero, then the penalty term equals to zero, and we would minimize the RSS, which is equivalent as ordinary least squared model; if λ is large, then the penalty term would be larger as well, the coefficients would be smaller; therefore, if λ approaches infinity, then the coefficients would approach zero. The selection of the optimal tuninig parameter is based on the evidence which is the MSE, and we would discuss the choice of tuning parameter later in the application.

Chapter 3

How Ridge Works

Recall from Linear Algebra that if we want to solve for $\boldsymbol{A}x = b$, we can do:

$$\boldsymbol{A}^T\boldsymbol{A}x = \boldsymbol{A}^T y$$

We then take the inverse of $\boldsymbol{A}^T\boldsymbol{A}$ from the LHS, and multiply both sides by the inverse:

$$x = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T y$$

Then we have come up with a least squares solution for x in $Ax = b$. From here then we derive that: $\hat{\beta}_{OLS} = (X^T X)^{-1} X^T y$. We will use a very similar approach to show how the shrinkage work for ridge regression.

3.0.1 Univariate example

Let's consider a very simple model: $y = \beta x + \epsilon$, with an L2(ridge regression) penalty on $\hat{\beta}$ and a least-squares loss function on $\hat{\epsilon}$. We can then expand the expression for sum of squared residuals to be minimized as:

$$\hat{\epsilon} = arg\min_{\beta}\{\sum_{i=1}^{N}(y_i - x_i\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\}$$

Which if we transform into a Matrix form, gives:

$$\hat{\epsilon} = arg\min_{\beta}\{(\vec{y} - \vec{x}\hat{\beta})^T(\vec{y} - \vec{x}\hat{\beta}) + \lambda\hat{\beta}^2\}$$

Which we can further expand into:

$$\hat{\epsilon} = arg\min_{\beta}\{\vec{y}^T\vec{y} - 2\vec{y}^T\vec{x}\hat{\beta} + \hat{\beta}\vec{x}^T\vec{x}\hat{\beta} + \lambda\hat{\beta}^2\}$$

Now if we take the derivative w.r.t $\hat{\beta}$ and set equal to 0, we can calculate the the estimator for ridge regression coefficients $\hat{\beta}$:

$$-2\vec{y}^T\vec{x} + 2\vec{x}^T\vec{x}\hat{\beta} + 2\lambda\hat{\beta} =_{set} 0$$

Where we can obtain:

$$\hat{\beta} = \vec{y}^T\vec{x}(\vec{x}^T\vec{x} + \lambda)^{-1}$$

3.0.2 Higher dimension ridge regression

Note that here we are using an univariate example(1-dimensional y and 1-dimensional x), in a more complicated case, we calculate the sum of squared residual in the same manner:

$$RSS(\lambda) = (\boldsymbol{y} - \boldsymbol{X}\beta)^T(\boldsymbol{y} - \boldsymbol{X}\beta) + \lambda\beta^T\beta$$

Where we can get:

$$\hat{\beta} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_{pp})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

Which is very similar to the result from the univariate example.

3.0.3 The tuning parameter

To further understand why λ helps shrink the coefficients, we are going to use singular value decomposition(SVD) to examine the effect of λ. Let the singular value decomposition of the $(n \times p)$-dimensional design matrix X be:

$$\boldsymbol{X} = \boldsymbol{U}_x\boldsymbol{D}_x\boldsymbol{V}_x^T$$

where $\boldsymbol{D}_x$ is an $(n \times n)$-dimensional diagonal matrix with the singular values, $\boldsymbol{U}_x$ is an $(n \times n)$-dimensional matrix with columns containing the left singular vectors (denoted u_i), and $\boldsymbol{V}_x$ is a $(p \times n)$-dimensional matrix with columns containing the right singular vectors (denoted V_i). The columns of $\boldsymbol{U}_x$ and $\boldsymbol{V}_x$ are orthogonal:

$$\boldsymbol{U}_x^T\boldsymbol{U}_x = \boldsymbol{I}_{nn} = \boldsymbol{V}_x^T\boldsymbol{V}_x$$

Then we can rewrite $\hat{\beta}$ as:

$$\begin{aligned}
\hat{\beta} &= (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_{pp})^{-1}\boldsymbol{X}^T\boldsymbol{Y} \\
&= (\boldsymbol{V}_x\boldsymbol{D}_x\boldsymbol{U}_x^T\boldsymbol{U}_x\boldsymbol{D}_x\boldsymbol{V}_x^T + \lambda\boldsymbol{I}_{pp})^{-1}\boldsymbol{V}_x\boldsymbol{D}_x\boldsymbol{U}_x^T\boldsymbol{Y} \\
&= (\boldsymbol{V}_x\boldsymbol{D}_x^2\boldsymbol{V}_x^T + \lambda\boldsymbol{V}_x\boldsymbol{V}_x^T)^{-1}\boldsymbol{V}_x\boldsymbol{D}_x\boldsymbol{U}_x^T\boldsymbol{Y} \\
&= \boldsymbol{V}_x(\boldsymbol{D}_x^2 + \lambda\boldsymbol{I}_{nn})^{-1}\boldsymbol{V}_x^T\boldsymbol{V}_x\boldsymbol{D}_x\boldsymbol{U}_x^T\boldsymbol{Y} \\
&= \boldsymbol{V}_x(\boldsymbol{D}_x^2 + \lambda\boldsymbol{I}_{nn})^{-1}\boldsymbol{D}_x\boldsymbol{U}_x^T\boldsymbol{Y}
\end{aligned}$$

Then the ridge solutions becomes

$$\begin{aligned}\boldsymbol{X}\hat{\beta} &= \boldsymbol{U}_x\boldsymbol{D}_x\boldsymbol{V}_x^T\boldsymbol{V}_x(\boldsymbol{D}_x^2+\lambda\boldsymbol{I}_{nn})^{-1}\boldsymbol{D}_x\boldsymbol{U}_x^T\boldsymbol{Y}\\ &= \boldsymbol{U}_x\boldsymbol{D}_x(\boldsymbol{D}_x^2+\lambda\boldsymbol{I}_{nn})^{-1}\boldsymbol{D}_x\boldsymbol{U}_x^T\boldsymbol{Y}\\ &= \sum_{j=1}^{p}\boldsymbol{u}_j\frac{d_j^2}{d_j^2+\lambda}\boldsymbol{u}_j^T\boldsymbol{y}\end{aligned}$$

Note that since $\lambda \geq 0$, we have $\frac{d_j^2}{d_j^2+\lambda} \leq 1$. Ridge regression computes the coordinates of $\boldsymbol{y}$ with respect to the orthonormal basis U. Then it shrinks these coordinates by the factor $\frac{d_j^2}{d_j^2+\lambda}$. This means that variable with smaller singular value is shrunk more than those with larger singular values.

Chapter 4

Lasso

Least absolute shrinkage and selection operator, aka Lasso, is similar to the ridge regression but its constraint is estimated with the sum of the absolute value of the coefficient:

$$\beta^{\hat{Lasso}} = arg\min_{\beta}\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\} \quad (1)$$

Which is the same as:

$$\beta^{\hat{Lasso}} = arg\min_{\beta}\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2\}\text{subject to}\sum_{j=1}^{p}|\beta_j| \leq t$$

Notice that in Lasso, the penalty term now becomes the sum of absolute value of the coefficients instead of sum of squares of the coefficients.

Chapter 5

How Lasso performs variable selection

Again, just as we discussed how shrinkage worked, we'll use the univariate model to illustrate.

5.0.1 univariate example

Again we have a simple model: $y = \beta x + \epsilon$, with an L2 penalty(Lasso regression) on $\hat{\beta}$ and a least squares loss function on $\hat{\epsilon}$. We can then expand the expression for sum of squared residuals to be minimized as:

$$\hat{\epsilon} = arg \min_{\beta} \{ \sum_{i=1}^{N} (y_i - x_i \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \}$$

For convenience, we will now substitute $2\lambda^* = \lambda$, since $2\lambda^* = \lambda$ has a one-to-one relationship, this change does not affect the result. We can then rewrite $\hat{\epsilon}$ in the matrix form:

$$\hat{\epsilon} = arg \min_{\beta} \{ (\vec{y} - \vec{x}\hat{\beta})^T (\vec{y} - \vec{x}\hat{\beta}) + 2\lambda^* |\hat{\beta}| \}$$

Which we can further expand into:

$$\hat{\epsilon} = arg \min_{\beta} \{ \vec{y}^T \vec{y} - 2\vec{y}^T \vec{x}\hat{\beta} + \hat{\beta}\vec{x}^T \vec{x}\hat{\beta} + 2\lambda^* |\hat{\beta}| \}$$

Unlike the ridge regression case where we can take a derivative directly and set it equal to zero so that we can minimize the residual, here we have a $|\hat{\beta}|$ term which makes such procedure painful. Thus, we will instead compare different cases w.r.t. $\hat{\beta}$. + Case1: $\hat{\beta} \geq 0$, $|\hat{\beta}| = \hat{\beta}$ - Since we are assuming that $\hat{\beta} \geq 0$, it is the same as assuming $\vec{y}^T\vec{x} \geq 0$(x's and y's have a positive relationship). - In this case, we can rewrite the above expression as:

$$\hat{\epsilon} = arg\min_{\beta}\{\vec{y}^T\vec{y} - 2\vec{y}^T\vec{x}\hat{\beta} + \hat{\beta}\vec{x}^T\vec{x}\hat{\beta} + 2\lambda^*\hat{\beta}\}$$

- Then we can take the derivative w.r.t. $\hat{\beta}$ and set it equal to zero:

$$-2\vec{y}^T\vec{x} + 2\vec{x}^T\vec{x}\hat{\beta} + 2\lambda^* =_{set} 0$$

- Where we can obtain a solution for $\hat{\beta}$:

$$\hat{\beta} = (\vec{y}^T\vec{x} - \lambda^*)(\vec{x}^T\vec{x})^{-1}$$

- Obviously by increasing λ^*, we can eventually achieve $\hat{\beta} = 0$ at $\lambda^* = \vec{y}^T\vec{x}$. However, it is tricky to think about what happens when we increase λ^* once $\vec{y}^T\vec{x} = 0$. The thing is that increasing λ^* at this point will not drive $\hat{\beta}$ negative, because once the estimator $\hat{\beta}$ becomes negative, the derivative of the penalty term estimator becomes:

$$-2\vec{y}^T\vec{x} + 2\vec{x}^T\vec{x}\hat{\beta} + 2\lambda^* =_{set} 0$$

- where the flip in the sign of λ^* is due to the absolute value function before taking the derivative. Thus, we have a new solution for $\hat{\beta}$:

$$\hat{\beta} = (\vec{y}^T\vec{x} + \lambda^*)(\vec{x}^T\vec{x})^{-1}$$

- This solution, however, is inconsistent with our premises $\hat{\beta} < 0$, since we have assumed that the least squares solution is greater than or equal to zero($\vec{y}^T\vec{x} \geq 0$), and $\lambda^* \geq 0$. For this solution, the sum of squared residual does not have an minimum anymore. Thus, we will just stick at $\hat{\beta} = 0$, even if $\lambda^* > \vec{y}^T\vec{x}$. + Intuitively, when we assume that the least squares solution is negative with $\hat{\beta} < 0$, the logic is the same that we stick with $\hat{\beta} = 0$ once it reaches zero. Note that so far we've only talked about a univariate Lasso example. When having a dataset that has multiple dimensions, as we keep increasing the value of λ(or λ^*), some of the features or variables will be zeroed out just as $\hat{\beta}$ shown above while some other features are shrinked toward zero but not yet reduced to zero. Therefore, the Lasso does variable selection in this manner, by shrinking some of the coefficients to zero while some non-zero.

5.0.2 Shrinkage explaination from geometric perspective

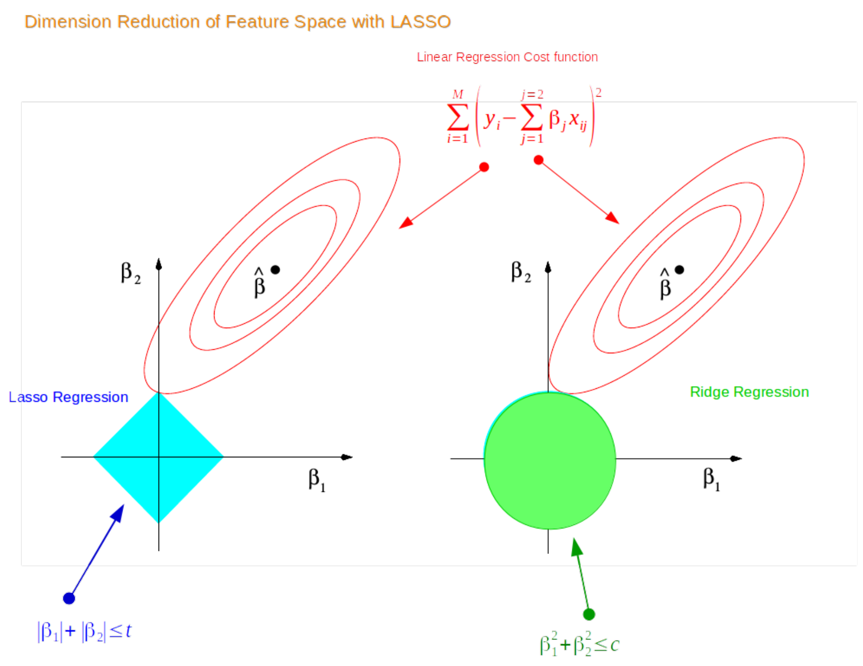
Now suppose that we have a dataset with 2 features. We first train a liner model for the data, and suppose β_1 and β_2 are the coefficients for the two features. Thus by the definition of the ridge regression constraint, we have:

$$\beta_1^2 + \beta_2^2 \leq t$$

Similarly, for the Lasso regression contraint, we have:

$$|\beta_1|+|\beta_2| \leq t$$

Thus, if we plot the two constraints in a 2-dimensional coordinate system, we


Dimension Reduction of Feature Space with LASSO
Linear Regression Cost function
$\sum_{i=1}^{M}\left(y_i-\sum_{j=1}^{j=2}\beta_j x_{ij}\right)^2$
β_2
$\hat{\beta}$
β_1
Lasso Regression
Ridge Regression
$|\beta_1|+|\beta_2|\leq t$
$\beta_1^2+\beta_2^2\leq c$

have:

Note that the red contours are the vector space for the coefficient estimator $\hat{\beta}$. Originally, the constraint and the vector space are separate; as we increase the constraint limit t, eventually the two constraints will hit the vector space, and thus achieving optimization for both the ridge regression model and the Lasso regression model. Since the 2-dimensional Lasso constraint has corners(diamond shape), if the constraint hit the vector space on one such corner, one of the two features gets dropped and shrinked to zero. In higher dimensional spaces, the diamond shape becomes rhomboid where there are more corners and more possibility for dropping one or more variables. In contrast, the constraint for ridge regression is a circle, where all points on or within the circle resemble a linear combination of the two features. Thus, technically speaking, when the ridge regression constraint hit the vector space and achieve optimization, one variable could be shrunk very close to zero while the other remain slightly shrunk, it is impossible for ridge regression to achieve variable selection in reality. Similarly, in higher dimensional space, the constraint for ridge regression becomes a shpere and the situation is very similar to what we have in the two dimensional space.

Chapter 6

A Real Example

6.1 Cross validation

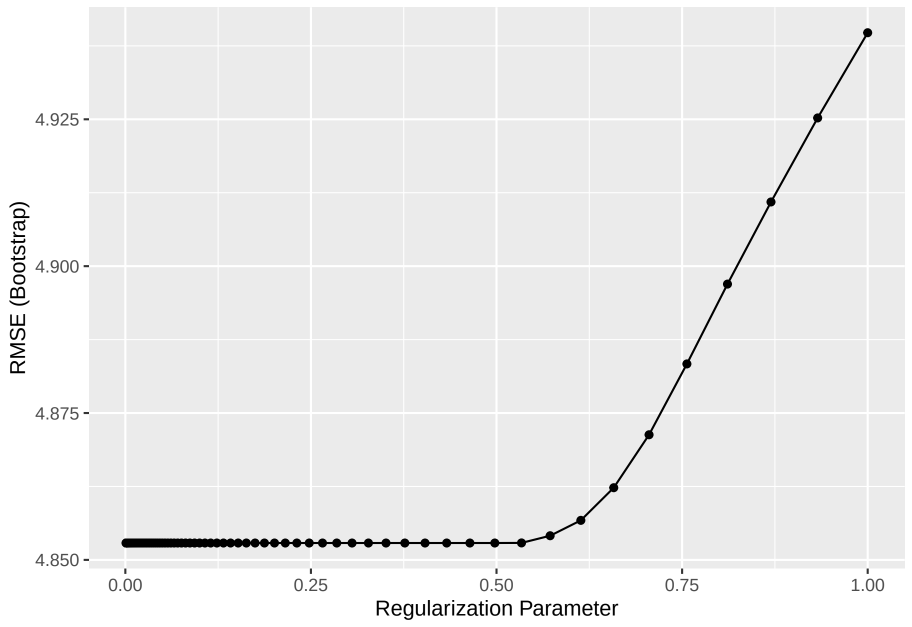
Cross validation randomly divides the data set into ten folds and in each fold the data is divided into ten groups, nine training groups to which models will be fitted and one testing group to test the model.

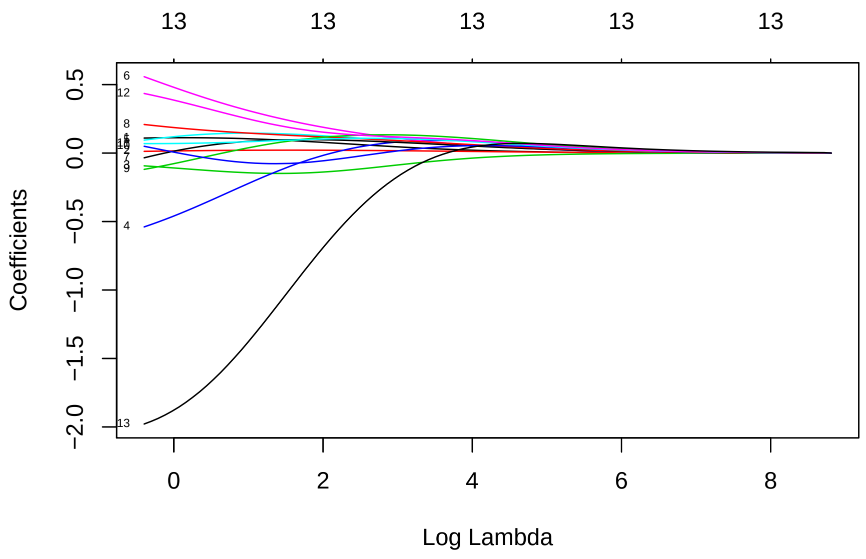
6.2 WITHOUT CV

6.2.1 OLS

```
## (Intercept)         Age      Weight      Height        Neck       Chest
## 10.53771759  0.07344948  0.01129956 -0.05372288 -0.80411627 -0.11789624
##     Abdomen         Hip       Thigh        Knee       Ankle      Biceps
##  0.98528673 -0.43320085  0.37243770 -0.25791914  0.24878673  0.07757529
##     Forearm       Wrist
##  0.58521403 -1.97978709
## [1] 4.712247
```

6.2.2 RIDGE

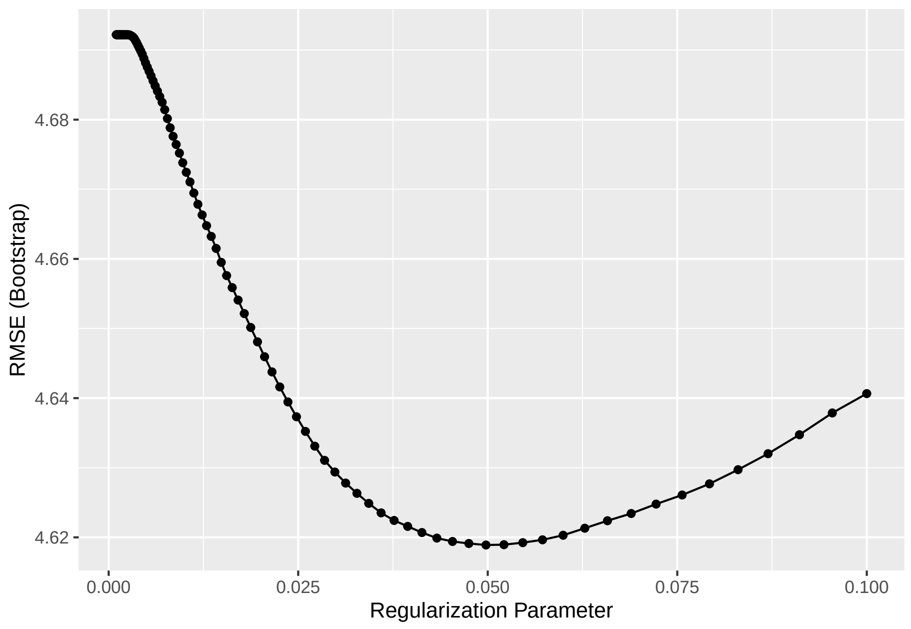


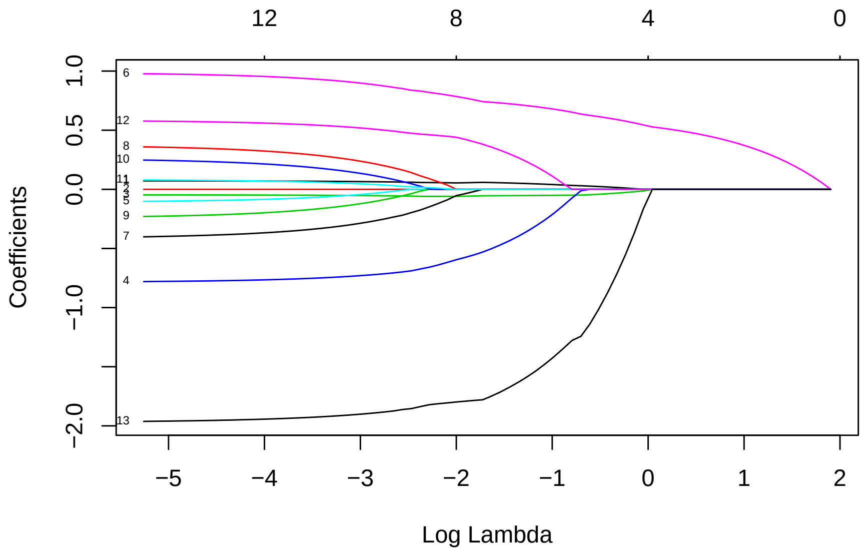


```
## [1] 0.4977024
```

```
## [1] 4.856731
```

6.2.3 LASSO





```
## [1] 0.04977024
```

```
## [1] 4.618946
##       RMSE
## 1 5.023873
##       RMSE
## 1 4.885821
##       RMSE
## 1 4.901642
```

6.3 WITH CV

6.3.1 OLS

```
set.seed(455)
bf_ols <- train(
  bodyfat ~ .,
  data = bf_train,
  method = "lm",
  trControl = trainControl(method = "cv",
                           number = 10),
  na.action = na.omit
)

coefficients(bf_ols$finalModel)
```

```
## (Intercept)         Age      Weight      Height        Neck       Chest
## 10.53771759  0.07344948  0.01129956 -0.05372288 -0.80411627 -0.11789624
##     Abdomen         Hip       Thigh        Knee       Ankle      Biceps
##  0.98528673 -0.43320085  0.37243770 -0.25791914  0.24878673  0.07757529
##     Forearm       Wrist
##  0.58521403 -1.97978709
```

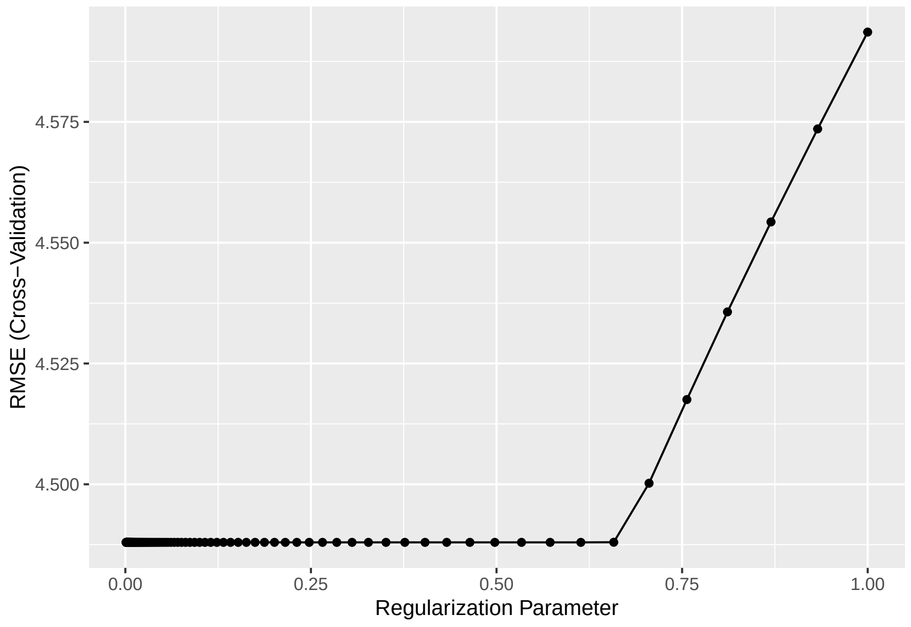
```
bf_ols$results$RMSE
```

```
## [1] 4.319528
```

Chapter 7

RIDGE

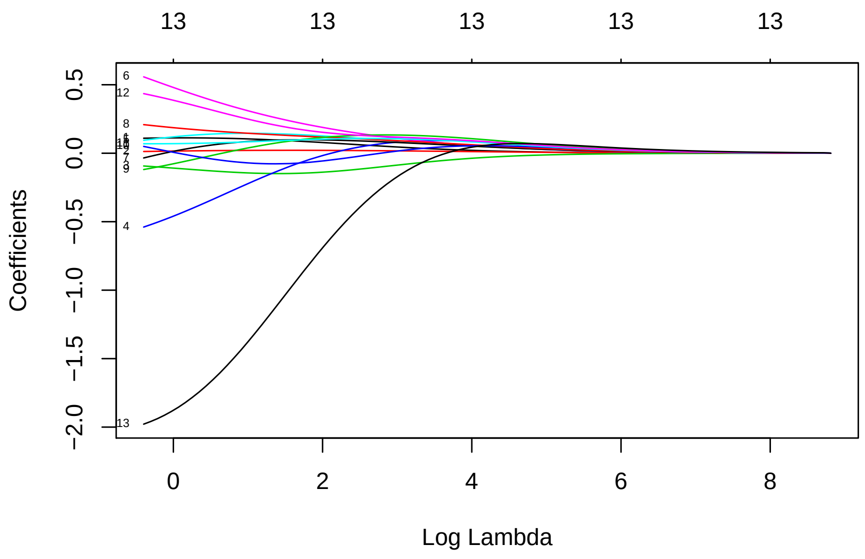
```r
set.seed(455)
bf_ridge <- train(
  bodyfat ~ .,
  data = bf_train,
  method = "glmnet",
  trControl = trainControl(method = "cv",
                           number = 10),
  tuneGrid = data.frame(alpha = 0,
                        lambda = 10^seq(-3,0, length = 100)),
  na.action = na.omit
)

bf_ridge%>%
  ggplot(aes(x=lambda,y=RMSE))+
  geom_point(size = .5, alpha = .5)
```



```
bf_ridge$bestTune$lambda
```

```
## [1] 0.6135907
```

```
plot(bf_ridge$finalModel,xvar="lambda",label=TRUE)
```

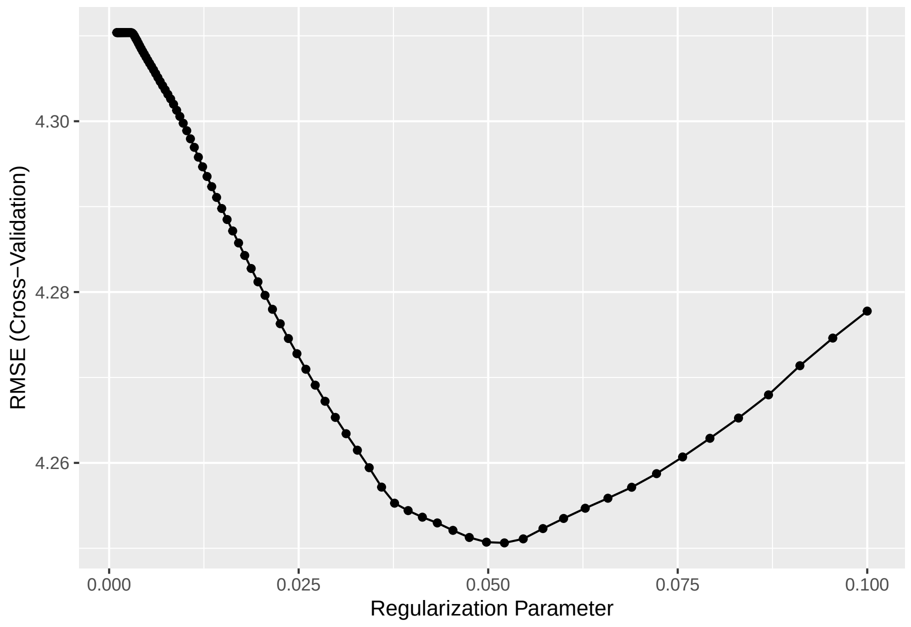


```
set.seed(455)
bf_ridge_best <- train(
  bodyfat ~ .,
  data = bf_train,
  method = "glmnet",
  trControl = trainControl(method = "cv",
                           number = 10),
  tuneGrid = data.frame(alpha = 0,
                        lambda = 0.6135907),
  na.action = na.omit
)

bf_ridge_best$results$RMSE
```

```
## [1] 4.487984
```

7.0.1 LASSO

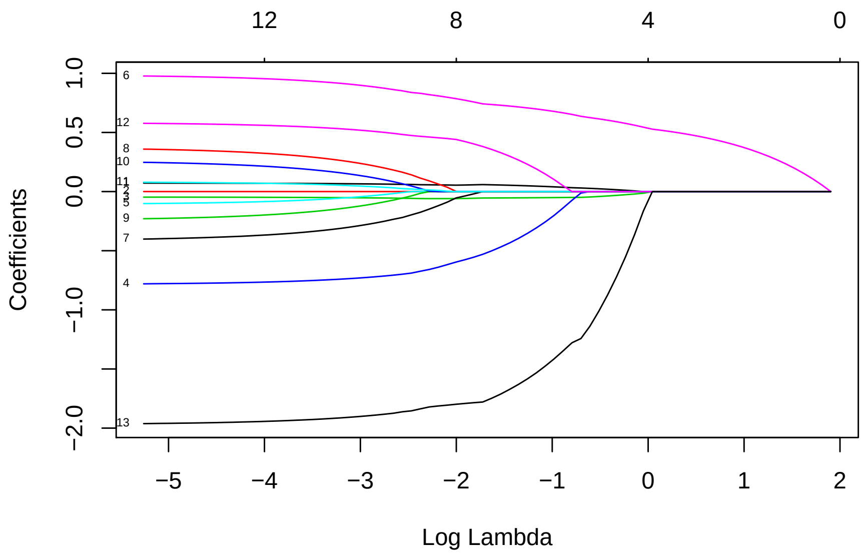
```
set.seed(455)
bf_lasso <- train(
  bodyfat ~ .,
  data = bf_train,
  method = "glmnet",
  trControl = trainControl(method = "cv",
                           number = 10),
  tuneGrid = data.frame(alpha = 1,
                        lambda = 10^seq(-3, -1, length = 100)),
  na.action = na.omit
)

bf_lasso%>%
  ggplot(aes(x=lambda,y=RMSE))+
  geom_point(size = .5, alpha = .5)
```



```
bf_lasso$bestTune$lambda
```

```
## [1] 0.05214008
```

```
plot(bf_lasso$finalModel,xvar='lambda',label=TRUE)
```



```
set.seed(455)
bf_lasso_best <- train(
  bodyfat ~ .,
  data = bf_train,
  method = "glmnet",
  trControl = trainControl(method = "cv",
                           number = 10),
  tuneGrid = data.frame(alpha = 1,
                        lambda = 0.05214008),
  na.action = na.omit
)

bf_lasso_best$results$RMSE
```

```
## [1] 4.250632
```

```
bf_test%>%
  mutate(pred_bf = predict(bf_ols, newdata = bf_test)) %>%
  summarize(RMSE = sqrt(mean((bodyfat - pred_bf)^2)))
```

```
##       RMSE
## 1 5.023873
```

```
bf_test%>%
  mutate(pred_bf = predict(bf_ridge_best, newdata = bf_test)) %>%
  summarize(RMSE = sqrt(mean((bodyfat - pred_bf)^2)))
```

```
##       RMSE
## 1 4.885821
```

```
bf_test%>%
  mutate(pred_bf = predict(bf_lasso_best, newdata = bf_test)) %>%
  summarize(RMSE = sqrt(mean((bodyfat - pred_bf)^2)))
```

```
##       RMSE
## 1 4.901642
```

```
set.seed(455)
cars_split <- initial_split(cars2018, prop = .7)
cars_train <- training(cars_split)
cars_test <- testing(cars_split)
```

7.0.2 OLS

```
set.seed(455)
cars_ols <- train(
  MPG ~ .,
  data = cars_train,
  method = "lm",
  trControl = trainControl(method = "cv",
                           number = 10),
  na.action = na.omit
)

coefficients(cars_ols$finalModel)
```

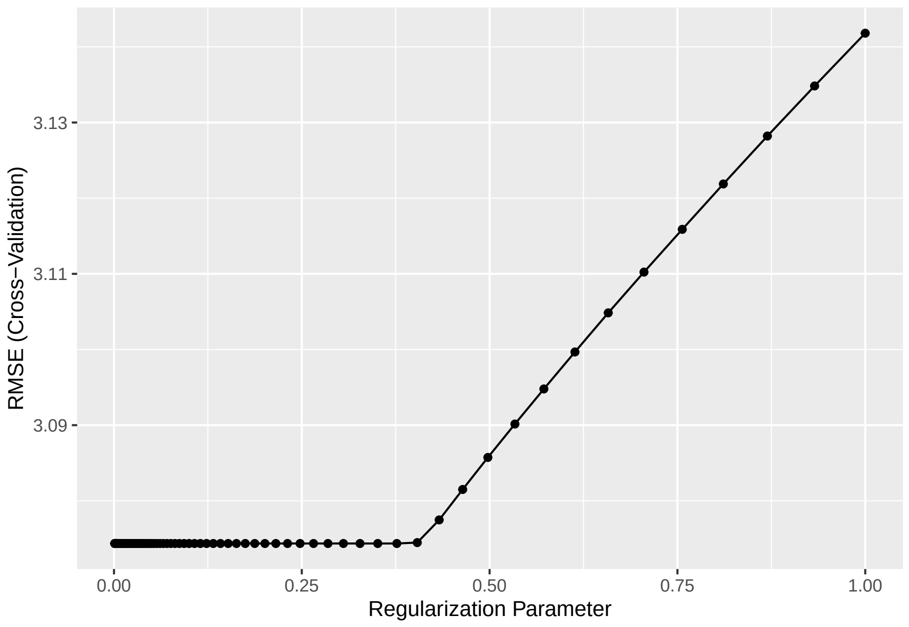
```
##                                            (Intercept)
##                                           45.795137024
##                                           Displacement
##                                           -3.897041594
##                                              Cylinders
##                                            0.591798369
##                                                  Gears
##                                            0.133841817
##                                           TransmissionCVT
##                                               4.864490054
##                                        TransmissionManual
##                                              -1.582604120
##                 `AspirationTurbocharged/Supercharged`
##                                              -2.533164224
##                      `\\`Lockup Torque Converter\\`Y`
##                                              -3.224584126
##                            `Drive2-Wheel Drive, Rear`
##                                              -2.865060504
##                                  `Drive4-Wheel Drive`
##                                              -3.542661160
##                                `DriveAll Wheel Drive`
##                                              -2.945736451
##                                   `\\`Max Ethanol\\``
##                                              -0.004295138
##    `\\`Recommended Fuel\\`Premium Unleaded Required`
##                                              -0.322194856
## `\\`Recommended Fuel\\`Regular Unleaded Recommended`
##                                              -1.199460113
##                           `\\`Intake Valves Per Cyl\\``
##                                              -1.682415911
##                          `\\`Exhaust Valves Per Cyl\\``
##                                              -2.339357182
## `\\`Fuel injection\\`Multipoint/sequential ignition`
##                                              -0.866682965
```

```
cars_ols$results$RMSE
```

```
## [1] 3.027263
```

7.0.3 RIDGE

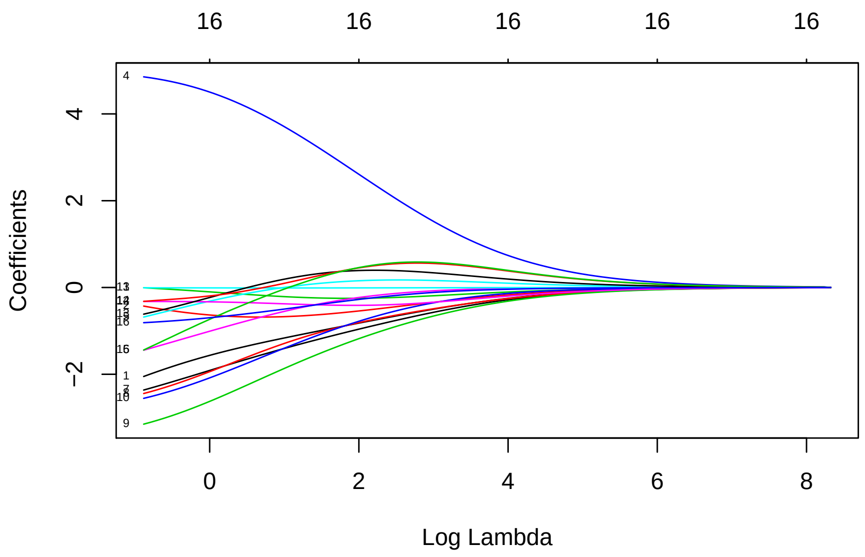
```
set.seed(455)
cars_ridge <- train(
  MPG ~ .,
  data = cars_train,
  method = "glmnet",
  trControl = trainControl(method = "cv",
                           number = 10),
  tuneGrid = data.frame(alpha = 0,
                        lambda = 10^seq(-3,0, length = 100)),
  na.action = na.omit
)

cars_ridge%>%
  ggplot(aes(x=lambda,y=RMSE))+
  geom_point(size = .5, alpha = .5)
```



```
cars_ridge$bestTune$lambda
```

```
## [1] 0.3764936
```

```
plot(cars_ridge$finalModel,xvar="lambda",label=TRUE)
```

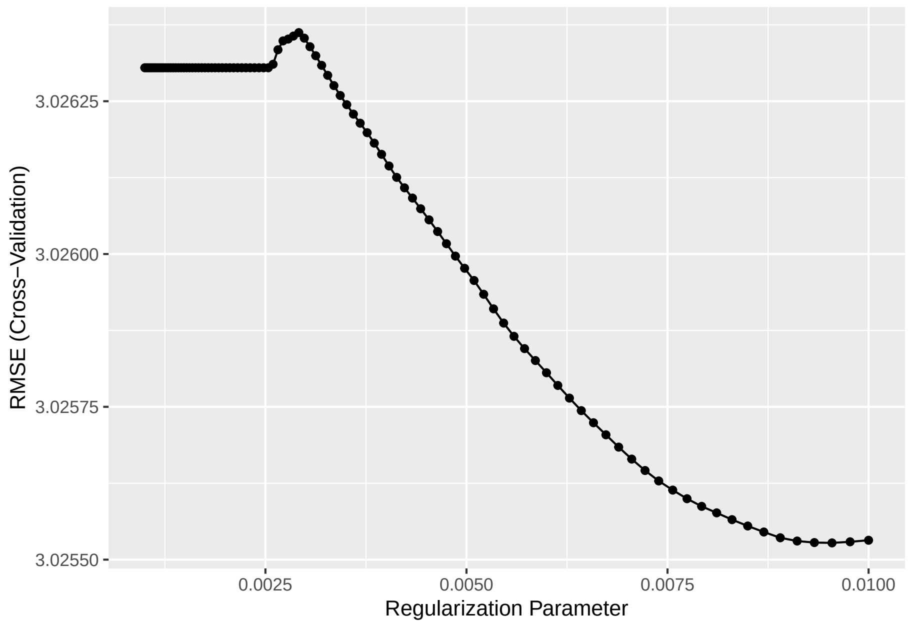


```r
set.seed(455)
cars_ridge_best <- train(
  MPG ~ .,
  data = cars_train,
  method = "glmnet",
  trControl = trainControl(method = "cv",
                           number = 10),
  tuneGrid = data.frame(alpha = 0,
                        lambda = 0.3764936),
  na.action = na.omit
)

cars_ridge_best$results$RMSE
```

```
## [1] 3.074365
```

7.0.4 LASSO

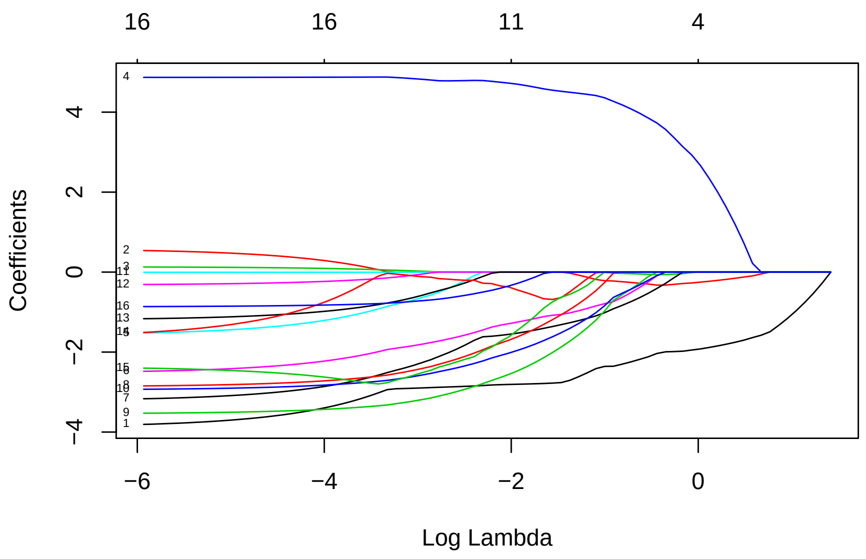
```r
set.seed(455)
cars_lasso <- train(
  MPG ~ .,
  data = cars_train,
  method = "glmnet",
  trControl = trainControl(method = "cv",
                           number = 10),
  tuneGrid = data.frame(alpha = 1,
                        lambda = 10^seq(-3, -2, length = 100)),
  na.action = na.omit
)

cars_lasso%>%
  ggplot(aes(x=lambda,y=RMSE))+
  geom_point(size = .5, alpha = .5)
```



```
cars_lasso$bestTune$lambda
```

```
## [1] 0.009545485
```

```
plot(cars_lasso$finalModel,xvar='lambda',label=TRUE)
```



```
set.seed(455)
cars_lasso_best <- train(
  MPG ~ .,
  data = cars_train,
  method = "glmnet",
  trControl = trainControl(method = "cv",
                           number = 10),
  tuneGrid = data.frame(alpha = 1,
                        lambda = 0.009545485),
  na.action = na.omit
)

cars_lasso_best$results$RMSE
```

```
## [1] 3.025527
```

```
cars_test%>%
  mutate(pred_mpg = predict(cars_ols, newdata = cars_test)) %>%
  summarize(RMSE = sqrt(mean((MPG - pred_mpg)^2)))
```

```
## # A tibble: 1 x 1
##    RMSE
##   <dbl>
## 1  2.54
```

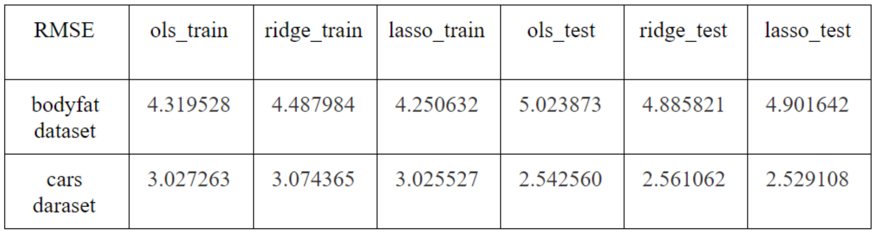
```
cars_test%>%
  mutate(pred_mpg = predict(cars_ridge_best, newdata = cars_test)) %>%
  summarize(RMSE = sqrt(mean((MPG - pred_mpg)^2)))
```

```
## # A tibble: 1 x 1
##    RMSE
##   <dbl>
## 1  2.56
```

```
cars_test%>%
  mutate(pred_mpg = predict(cars_lasso_best, newdata = cars_test)) %>%
  summarize(RMSE = sqrt(mean((MPG - pred_mpg)^2)))
```

```
## # A tibble: 1 x 1
##    RMSE
##   <dbl>
## 1  2.53
```

7.1 Data Summary



RMSE	ols_train	ridge_train	lasso_train	ols_test	ridge_test	lasso_test
bodyfat dataset	4.319528	4.487984	4.250632	5.023873	4.885821	4.901642
cars daraset	3.027263	3.074365	3.025527	2.542560	2.561062	2.529108

From the summary table above, we can see that for the highly corrlated numerical dataset bodyfat, lasso gives the smallest training RMSE while ridge gives the smallest testing RMSE. For the less correlated dataset cars with categorical variables included, lasso always gives the smallest RMSE.

Chapter 8

Bayesian Lasso

8.1 Generalization of shrinkage methods from Bayes perspective

Bayesian estimation assumes the coefficients have prior distribution, $p(\beta)$, where $\beta = (\beta_0, \beta_1 ... \beta_p)^T$. The likelihood of data is $f(Y|X, \beta)$, where $X = (X_1 ... X_p)$. Therefore, we can derive the formula for posterior distribution of β:

$$p(\beta|Y, X) \propto f(Y|X, \beta)p(\beta|X) = f(Y|X, \beta)p(\beta)$$

8.2 Bayesian Interpretation

We can show that $\hat{\boldsymbol{\beta}}_{LASSO}$ has a Bayesian interpretation. In particular, we can show that is a Bayes estimator for $\boldsymbol{\beta}$ assuming a multivariate Normal likelihood for $\mathbf{y}$:

$$f(y \mid \boldsymbol{\beta}, \sigma^2) \sim N(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}),$$

and an independent double exponential (aka Laplace) prior for $\boldsymbol{\beta}$:

$$f(\beta_j) = \left(\frac{\lambda}{2}\right) \exp(-\lambda|\beta_j|)$$

8.2.1 Posterior

The posterior distribution for $\boldsymbol{\beta}$ (assuming $\sigma^2 = 1$ for simplicity):

$$
\begin{aligned}
g(\boldsymbol{\beta} \mid y) &\propto f(y \mid \boldsymbol{\beta}, \sigma^2) f(\boldsymbol{\beta}) \\
&= f(y \mid \boldsymbol{\beta}) \prod_{i=1}^{p} f(\beta_j), \text{ by independence assumption} \\
&= (2\pi)^{-n/2} \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right) \left(\frac{\lambda}{2}\right)^p \exp\left(-\lambda \sum_{j=1}^{p} |\beta_j|)\right) \\
&\propto \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right) \exp\left(-\lambda \sum_{j=1}^{p} |\beta_j|)\right) \\
&= \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - \lambda \sum_{j=1}^{p} |\beta_j|)\right)
\end{aligned}
$$

8.2.2 Posterior Mode

Maximize the posterior distribution, or minimize the -log posterior

$$
\begin{aligned}
&\max \;\; \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - \lambda \sum_{j=1}^{p} |\beta_j|)\right) \\
&= \max \;\; -\frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - \lambda \sum_{j=1}^{p} |\beta_j| \\
&= \min \;\; \frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} |\beta_j| \\
&= \min \;\; \text{-log posterior}
\end{aligned}
$$

Chapter 9

Limitations of Ridge and Lasso

- One important drawback of Lasso and ridge is that it's very hard/impossible to do proper inference (i.e., get confidence intervals and p-values)—you really just get estimates and that's it.
- The other thing is that we usually cannot give a good explanation for the coefficients and the entire model, because the coefficients are shrunk. Some might say we could use Lasso simply as a variable selection tool and we can use the Lasso result to generate an OLS which is easier to explain. But note that there is possibility that Lasso would give us wrong information, based on assumptions we have made about the data.
- In real examples, when dealing with categorical variables(those with more than 10 sub-categories), Lasso often does weird thing as it treats each of the sub-category as an individual variable, that is, it selects some of the sub-categories while think others to be irrelevant. This also makes it hard to interpret.