Ridge and LASSO

Jongrak

2022-12-30

```
library(ISLR)
data(Hitters)

Hitters <- na.omit(Hitters)
dim(Hitters)

## [1] 263  20
```

#NA check

```
sum(is.na(Hitters))

## [1] 0

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-4

attach(Hitters) # $없이도 variables를 사용가능.

x <- model.matrix(Salary ~. , Hitters)[, -1] #0,1로
y <- Salary

#Ridge
model <- glmnet(x, y, alpha = 0, lambda = 0.01)
model <- glmnet(x, y, alpha = 0, lambda = 0) #linear model

#alpha? 0이면 Ridge, 1이면 Lasso. 각각의 비율
#lambda? 가중치 Rigde or Lasso에 얼마나 가중치를 둘지. (Regularization term)
보통은 0.01

summary(model)

##           Length Class     Mode
## a0         1     -none-    numeric
## beta      19     dgCMatrix S4
## df         1     -none-    numeric
## dim        2     -none-    numeric
## lambda     1     -none-    numeric
## dev.ratio  1     -none-    numeric
## nulldev    1     -none-    numeric
## npasses    1     -none-    numeric
```

```
## jerr       1     -none-    numeric
## offset     1     -none-    logical
## call       5     -none-    call
## nobs       1     -none-    numeric
```

```
coef_ridge <- coef(model) #Beta
coef_ridge
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  161.96353833
## AtBat         -1.94821630
## Hits           7.34916171
## HmRun          4.17526359
## Runs          -2.25373139
## RBI           -1.00612103
## Walks          6.17357807
## Years         -2.89425617
## CAtBat        -0.18782143
## CHits          0.21589198
## CHmRun        -0.05278157
## CRuns          1.40391488
## CRBI           0.76220232
## CWalks        -0.79112619
## LeagueN       63.51769593
## DivisionW   -116.76116069
## PutOuts        0.28137031
## Assists        0.37672854
## Errors        -3.41815042
## NewLeagueN   -25.65608789
```

```
grid <- 10 ^ seq(10, -2, length = 100) #10에서 -2까지 100개의 수열 # 다양한 parameter를 만들기 가능
```

```
ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid) # 계산을 100개... 100개의 모델에 대해 coefficient를 잡는 것이 가능.
```

```
coef(ridge.mod)[, 20] # 20번째
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  5.358880e+02  1.093664e-05  3.967221e-05  1.598556e-04  6.708833e-05
##           RBI         Walks         Years        CAtBat         CHits
##  7.086606e-05  8.340541e-05  3.410894e-04  9.390097e-07  3.455823e-06
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##  2.606160e-05  6.933126e-06  7.155123e-06  7.570013e-06 -1.164983e-04
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -1.568625e-03  4.380543e-06  7.154972e-07 -3.336588e-06 -2.312257e-05
```

```
ridge.mod$lambda[20]
```

```
## [1] 49770236
```

```r
sqrt(sum(coef(ridge.mod)[-1, 20] ^ 2)) # 지나치게 단순한 모델로 바꿨다는 것을 확인 가능 > Bias가 지나치게 커진다
```

```
## [1] 0.001623465
```

```r
set.seed(2022)
train <- sample(1:nrow(x), 180)
test <- (-train)

y.test <- y[test]

ridge.mod <- glmnet(x[train,], y[train], alpha = 0, lambda = grid) # 모델 100개
ridge.pred <- predict(ridge.mod, newx = x[test,]) # 100가지의 예측
dim(ridge.pred)
```

```
## [1]  83 100
```

```r
#lambda_index : 몇 번째 모델?
lambda_checker <- function(lambda_index) {
  coef_mse <- sqrt(sum(coef(ridge.mod)[-1, lambda_index] ^ 2))
  mse <- mean((ridge.pred[, lambda_index] - y.test) ^ 2)

  return(c(ridge.mod$lambda[lambda_index], coef_mse,mse))
}

no_reg <- glmnet(x[train,], y[train], alpha = 0, lambda = 0) # Without regularization
sqrt(sum(coef(no_reg) ^ 2)) # 261.5556
```

```
## [1] 261.5556
```

```r
no_reg_pred <- predict(no_reg, newx = x[test,])
mean((no_reg_pred - y.test) ^ 2)  # test error 90333.63
```

```
## [1] 90333.63
```

```r
lambda_checker(2)
```

```
## [1] 7.564633e+09 7.808214e-06 2.051833e+05
```

```r
lambda_checker(20)
```

```
## [1] 4.977024e+07 1.186767e-03 2.051708e+05
```

```r
lambda_checker(70)
```

```
## [1]    43.28761   124.72430 85294.95381
```