5 Penalized Regression

Immanuel Klein

```r
library(tidyverse)
library(ggplot2)
library(glmnet)
library(ISLR)
```

This task is about analyzing a dataset of baseball players to predict their salaries using ridge and lasso regression. We start by loading and cleaning the data to include only complete cases. Next, the condition number of the design matrix XtX is calculated to check for multicollinearity, with and without standardizing the variables. Standard linear regression and ridge regression with a specific value of lambda are then compared to evaluate the impact on coefficient sizes. To choose the optimal lambda, we split the data into training and test sets, and calculate mean squared prediction errors across lambda values. The optimal lambda is used to fit ridge and lasso models on the data to compare the coefficients, particularly to see whether lasso regression results in any coefficients being exactly zero.

Exercise (a)

```r
# Loading the dataset and creating a new dataset
# containing only those players for which all data is available.
data(Hitters)
hitters.clean <- na.omit(Hitters)

str(hitters.clean)
```

```
'data.frame':   263 obs. of  20 variables:
 $ AtBat    : int  315 479 496 321 594 185 298 323 401 574 ...
 $ Hits     : int  81 130 141 87 169 37 73 81 92 159 ...
 $ HmRun    : int  7 18 20 10 4 1 0 6 17 21 ...
 $ Runs     : int  24 66 65 39 74 23 24 26 49 107 ...
```

```
 $ RBI      : int  38 72 78 42 51 8 24 32 66 75 ...
 $ Walks    : int  39 76 37 30 35 21 7 8 65 59 ...
 $ Years    : int  14 3 11 2 11 2 3 2 13 10 ...
 $ CAtBat   : int  3449 1624 5628 396 4408 214 509 341 5206 4631 ...
 $ CHits    : int  835 457 1575 101 1133 42 108 86 1332 1300 ...
 $ CHmRun   : int  69 63 225 12 19 1 0 6 253 90 ...
 $ CRuns    : int  321 224 828 48 501 30 41 32 784 702 ...
 $ CRBI     : int  414 266 838 46 336 9 37 34 890 504 ...
 $ CWalks   : int  375 263 354 33 194 24 12 8 866 488 ...
 $ League   : Factor w/ 2 levels "A","N": 2 1 2 2 1 2 1 2 1 1 ...
 $ Division : Factor w/ 2 levels "E","W": 2 2 1 1 2 1 2 2 1 1 ...
 $ PutOuts  : int  632 880 200 805 282 76 121 143 0 238 ...
 $ Assists  : int  43 82 11 40 421 127 283 290 0 445 ...
 $ Errors   : int  10 14 3 4 25 7 9 19 0 22 ...
 $ Salary   : num  475 480 500 91.5 750 ...
 $ NewLeague: Factor w/ 2 levels "A","N": 2 1 2 2 1 1 1 2 1 1 ...
 - attr(*, "na.action")= 'omit' Named int [1:59] 1 16 19 23 31 33 37 39 40 42 ...
  ..- attr(*, "names")= chr [1:59] "-Andy Allanson" "-Billy Beane" "-Bruce Bochte" "-Bob Boon
```

```r
head(hitters.clean)
```

```
                  AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun
-Alan Ashby         315   81     7   24  38    39    14   3449   835     69
-Alvin Davis        479  130    18   66  72    76     3   1624   457     63
-Andre Dawson       496  141    20   65  78    37    11   5628  1575    225
-Andres Galarraga   321   87    10   39  42    30     2    396   101     12
-Alfredo Griffin    594  169     4   74  51    35    11   4408  1133     19
-Al Newman          185   37     1   23   8    21     2    214    42      1
                  CRuns CRBI CWalks League Division PutOuts Assists Errors
-Alan Ashby         321  414    375      N        W     632      43     10
-Alvin Davis        224  266    263      A        W     880      82     14
-Andre Dawson       828  838    354      N        E     200      11      3
-Andres Galarraga    48   46     33      N        E     805      40      4
-Alfredo Griffin    501  336    194      A        W     282     421     25
-Al Newman           30    9     24      N        E      76     127      7
                  Salary NewLeague
-Alan Ashby        475.0         N
-Alvin Davis       480.0         A
-Andre Dawson      500.0         N
-Andres Galarraga   91.5         N
-Alfredo Griffin   750.0         A
-Al Newman          70.0         A
```

Exercise (b)

```r
# Build a model matrix
X <- model.matrix(Salary ~ ., data = hitters.clean)[, -1]

# Compute XtX
XtX <- t(X) %*% X

# Compute eigenvalues of XtX
eigenvalues <- eigen(XtX)$values

# Calculate condition number
condition.number <- max(eigenvalues) / min(eigenvalues)
condition.number
```

```
[1] 424299885
```

```r
# Standardize design matrix
X.standardized <- scale(X)

# Compute XtX
XtX.standardized <- t(X.standardized) %*% X.standardized

# Compute eigenvalues of standardized XtX
eigenvalues.standardized <- eigen(XtX.standardized)$values

# Calculate condition number for standardized matrix
condition.number.standardized <-
  max(eigenvalues.standardized) / min(eigenvalues.standardized)
condition.number.standardized
```

```
[1] 6131.339
```

- With 424299885, the condition number of the original matrix is relatively high (especially compared to 6131.339). A high condition number indicates that the columns of X are nearly linearly dependent, which could lead to numerical instability in regression analysis.
- After standardizing the design matrix such that each column (except the intercept) has a mean of zero and variance of one, the condition number has decreased immensely, making the matrix XTX better conditioned. This can help mitigate multicollinearity and improve numerical stability.

Exercise (c)

```r
# Separate salary and predictors
y <- hitters.clean$Salary
# Remove intercept
X <- model.matrix(Salary ~ ., data = hitters.clean)[, -1]

# Standard Linear Regression Model
linear.models <- lm(Salary ~ ., data=hitters.clean)
linear.coefficients <- coef(linear.models)
print("Standard Linear Model Coefficients:")
```

```
[1] "Standard Linear Model Coefficients:"
```

```r
print(linear.coefficients)
```

```
 (Intercept)        AtBat         Hits        HmRun         Runs          RBI
 163.1035878   -1.9798729    7.5007675    4.3308829   -2.3762100   -1.0449620
       Walks        Years       CAtBat        CHits       CHmRun        CRuns
   6.2312863   -3.4890543   -0.1713405    0.1339910   -0.1728611    1.4543049
        CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
   0.8077088   -0.8115709   62.5994230 -116.8492456    0.2818925    0.3710692
      Errors   NewLeagueN
  -3.3607605  -24.7623251
```

```r
# Ridge Regression Model with   = 70
ridge.model <- glmnet(X,
                      hitters.clean$Salary,
                      alpha = 0,
                      lambda = 70,
                      standardize = TRUE)
ridge.coefficients <- coef(ridge.model)
print("Ridge Regression Model Coefficients with   = 70:")
```

```
[1] "Ridge Regression Model Coefficients with   = 70:"
```

```r
print(ridge.coefficients)
```

```
20 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept)  3.364118e+01
AtBat       -2.242666e-01
Hits         1.644728e+00
HmRun       -1.057644e+00
Runs         1.173291e+00
RBI          8.334793e-01
Walks        2.435300e+00
Years       -4.566521e+00
CAtBat       8.250831e-03
CHits        9.457427e-02
CHmRun       5.887674e-01
CRuns        1.879741e-01
CRBI         1.939934e-01
CWalks      -9.289755e-02
LeagueN      4.151885e+01
DivisionW   -1.142969e+02
PutOuts      2.404133e-01
Assists      9.828542e-02
Errors      -2.986010e+00
NewLeagueN  -4.722875e+00
```

```r
comparison <- data.frame(
  Variable = rownames(ridge.coefficients),
  Linear = as.vector(linear.coefficients),
  Ridge = as.vector(ridge.coefficients)
)
print("Comparison of Coefficients:")
```

```
[1] "Comparison of Coefficients:"
```

```r
print(comparison)
```

```
      Variable       Linear         Ridge
1  (Intercept)  163.1035878  3.364118e+01
2        AtBat   -1.9798729 -2.242666e-01
3         Hits    7.5007675  1.644728e+00
4        HmRun    4.3308829 -1.057644e+00
5         Runs   -2.3762100  1.173291e+00
6          RBI   -1.0449620  8.334793e-01
```

```
7          Walks    6.2312863  2.435300e+00
8          Years   -3.4890543 -4.566521e+00
9         CAtBat   -0.1713405  8.250831e-03
10         CHits    0.1339910  9.457427e-02
11        CHmRun   -0.1728611  5.887674e-01
12         CRuns    1.4543049  1.879741e-01
13          CRBI    0.8077088  1.939934e-01
14        CWalks   -0.8115709 -9.289755e-02
15       LeagueN   62.5994230  4.151885e+01
16     DivisionW -116.8492456 -1.142969e+02
17       PutOuts    0.2818925  2.404133e-01
18       Assists    0.3710692  9.828542e-02
19        Errors   -3.3607605 -2.986010e+00
20    NewLeagueN  -24.7623251 -4.722875e+00
```

- The intercept in the ridge regression model is smaller (around 33.64) compared to the linear model (163.10). Ridge regression shrinks the intercept due to the regularization effect.
- The coefficients in the ridge regression model are generally smaller than to those in the linear model. This is because ridge regression penalizes the size of the coefficients, which leads to shrinking.
- Some coefficients in the ridge regression model have a different sign than in the linear model.
- Ridge regression tends to reduce the coefficients towards zero, which leads to more stable coefficients that are less sensitive to multicollinearity.

Exercise (d)

```r
set.seed(1122)
train.indices <- sample(1:nrow(hitters.clean),
                        size = 0.7 * nrow(hitters.clean))
X.train <- X[train.indices, ]
y.train <- hitters.clean$Salary[train.indices]
X.test <- X[-train.indices, ]
y.test <- hitters.clean$Salary[-train.indices]

# Fit ridge regression models
lambda.seq <- 10^seq(10, -2, length = 100)
ridge.cv <- cv.glmnet(X.train,
```

```r
                      y.train,
                      alpha = 0,
                      lambda = lambda.seq,
                      standardize = TRUE)

# Find best lambda with cross-validation & use it
best.lambda <- ridge.cv$lambda.min
paste("Best lambda:", best.lambda)
```

```
[1] "Best lambda: 18.7381742286039"
```

```r
ridge.best <- glmnet(X.train,
                     y.train,
                     alpha = 0,
                     lambda = best.lambda,
                     standardize = TRUE)

# Predict on test set & MSE
predictions <- predict(ridge.best,
                       s = best.lambda,
                       newx = X.test)

mse <- mean((y.test - predictions)^2)

# Coefficients in model with best lambda
ridge.coefficients.best <- coef(ridge.best)

# Standard linear regression model for comparison
linear.model <- lm(Salary ~ ., data = hitters.clean[train.indices, ])
linear.coefficients <- coef(linear.model)

comparison <- data.frame(
  Variable = rownames(ridge.coefficients.best),
  Linear = as.vector(linear.coefficients),
  Ridge = as.vector(ridge.coefficients.best)
)
comparison
```

```
      Variable        Linear         Ridge
1  (Intercept)   25.45064830 -4.993090e+01
2        AtBat   -1.11916015 -7.011377e-01
```

```
3          Hits    2.23709751  2.901755e+00
4         HmRun   -3.41770857 -2.287678e+00
5          Runs    2.04551459  1.361425e+00
6           RBI    3.41665843  1.975079e+00
7         Walks    4.49599953  3.479701e+00
8         Years  -11.87281948 -1.224104e+01
9        CAtBat   -0.32908168  8.271724e-03
10        CHits    1.98152498  2.568917e-01
11       CHmRun    2.01548871  1.380512e-01
12        CRuns   -0.08869553  4.140786e-01
13         CRBI   -0.74866098  2.450340e-01
14       CWalks   -0.35894073 -4.233952e-01
15      LeagueN   65.89643098  4.422723e+01
16    DivisionW -107.80186143 -1.201734e+02
17      PutOuts    0.29846424  2.567713e-01
18      Assists    0.17850621  6.614518e-03
19       Errors   -1.64180697 -1.541995e+00
20   NewLeagueN  -26.96274441 -4.090620e+00
```

Exercise (e)

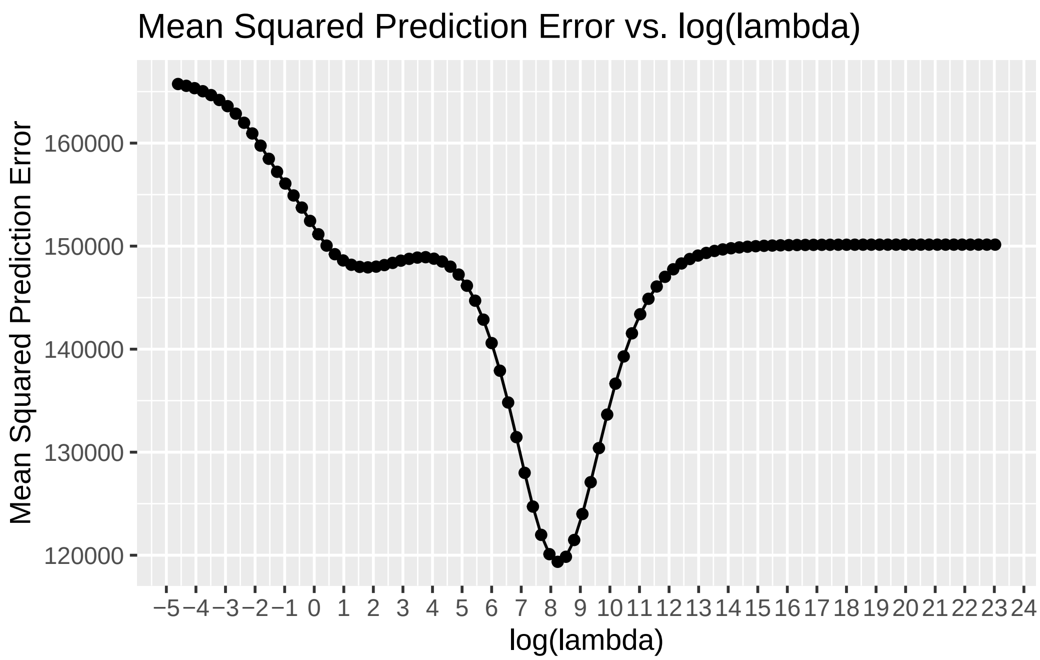
```r
ridge.mse <- function(lambda) {
  ridge.model <- glmnet(X.train,
                        y.train,
                        alpha = 0,
                        lambda = lambda,
                        standardize = TRUE)
  predictions <- predict(ridge.model, s = lambda, newx = X.test)
  mse <- mean((y.test - predictions)^2)
  return(mse)
}

lambda.seq <- 10^seq(10, -2, length = 100)

# MSE for each lambda
mse.values <- sapply(lambda.seq, ridge.mse)

ggplot(data.frame(log.lambda = log(lambda.seq), mse = mse.values),
       aes(x = log.lambda, y = mse)) +
  geom_line() +
  geom_point() +
```

```
  scale_x_continuous(breaks = scales::pretty_breaks(n = 40)) +
  labs(x = "log(lambda)", y = "Mean Squared Prediction Error", title = "Mean Squared Predicti
```




```
# Identify lambda that minimizes MSE
lambda.opt <- lambda.seq[which.min(mse.values)]
lambda.opt
```

```
[1] 3764.936
```

At the lowest point of the plot, log(lambda) is around 8.2, which means that the optimal lambda is somewhere around 3700. In fact, the optimal value is 3764.936.

Exercise (f)

```
# Fit a ridge regression with ptimal lambda on all data
ridge.final <- glmnet(X,
                      y,
                      alpha = 0,
                      lambda = lambda.opt,
```

```r
                      standardize = TRUE)

final.coefficients <- coef(ridge.final)

print("Ridge Regression Coefficients with lambda_opt:")
```

```
[1] "Ridge Regression Coefficients with lambda_opt:"
```

```r
print(final.coefficients)
```

```
20 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept) 263.849510443
AtBat         0.076921773
Hits          0.305659199
HmRun         1.036529179
Runs          0.497947263
RBI           0.503662054
Walks         0.637765722
Years         2.171800162
CAtBat        0.006469684
CHits         0.024607395
CHmRun        0.183467094
CRuns         0.049357002
CRBI          0.051010830
CWalks        0.050636680
LeagueN       1.899688442
DivisionW   -16.595081836
PutOuts       0.041066474
Assists       0.006047685
Errors       -0.091286211
NewLeagueN    1.914755755
```

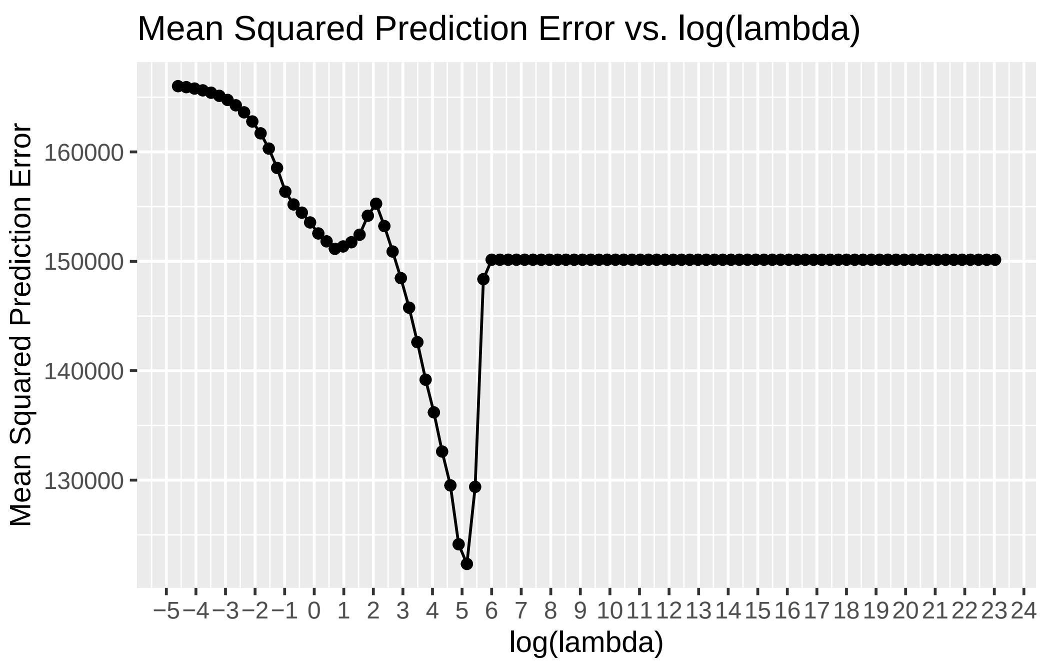
- The most important variables are (in order of their magnitude): `DivisionW`, `Years`, `NewLeagueN`, `LeagueN`, `HmRun`. They all have values above 1.
- Although there are coefficients with values very close to 0 (e. g. Assists), there are no coefficients that equal zero exactly.

Exercise (g)

```r
lasso.mse <- function(lambda) {
  lasso.model <- glmnet(X.train,
                        y.train,
                        alpha = 1,
                        lambda = lambda,
                        standardize = TRUE)
  predictions <- predict(lasso.model, s = lambda, newx = X.test)
  mse <- mean((y.test - predictions)^2)
  return(mse)
}
lambda.seq <- 10^seq(10, -2, length = 100)

# Calculate MSE for each lambda
mse.values <- sapply(lambda.seq, lasso.mse)

ggplot(data.frame(log.lambda = log(lambda.seq), mse = mse.values),
       aes(x = log.lambda, y = mse)) +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 40)) +
  labs(x = "log(lambda)",
       y = "Mean Squared Prediction Error",
       title = "Mean Squared Prediction Error vs. log(lambda)")
```




```r
# Identify lambda that minimizes MSE
lambda.opt.lasso <- lambda.seq[which.min(mse.values)]
lambda.opt.lasso
```

```
[1] 174.7528
```

```r
# Fit lasso regression with optimal lambda on all data
lasso.final <- glmnet(X,
                      y,
                      alpha = 1,
                      lambda = lambda.opt.lasso,
                      standardize = TRUE)

final.coefficients.lasso <- coef(lasso.final)

print("Lasso Regression Coefficients with lambda_opt:")
```

```
[1] "Lasso Regression Coefficients with lambda_opt:"
```

```r
print(final.coefficients.lasso)
```

```
20 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept) 440.66993436
AtBat         .
Hits          0.10216936
HmRun         .
Runs          .
RBI           .
Walks         .
Years         .
CAtBat        .
CHits         .
CHmRun        .
CRuns         0.06873144
CRBI          0.17980826
CWalks        .
LeagueN       .
DivisionW     .
PutOuts       .
Assists       .
Errors        .
NewLeagueN    .
```

- At the lowest point of the plot, log(lambda) is around 5.2, which means that the optimal lambda is somewhere around 180. In fact, the optimal value is 174.7528.
- Using lasso regression regression, now all coefficients except `Hits`, `CRuns`, and `CRBI` are equal to 0.