4.1.1: R - Penalized Regression Methods (Ridge Regression, LASSO, and Elastic Net)

Stat 5100: Dr. Bean

Matrix Specification

Previously, when we have created linear models in this class, you have always seen the model be created in the form:

```
linear_model <- lm(response ~ x1 + x2, data = mydata)
```

However, when we create models in R, we don't always do it like this. Another way to specify models in R is with "matrix specification." In matrix specification, instead of using the "∼" syntax (this is called a formula in R), instead we pass in a matrix of the predictor variables and a vector of the response variable y. As an example:

```
# Either:
model <- somemodel(y, X)
# Or in some packages:
model <- somemodel(X, y)
```

In the above, the variable X is a matrix in R where each column would contain our predictor variables x_1, x_2, etc. and our rows would contain the different observations. For example, $X_{i,j}$ (meaning the ith row of X and the jth column of X) would refer to the recorded value of x_j (the jth predictor variable for the the ith observation in our data.

The reason that this syntax is not always exactly synchronized with the formula notation you are used to, is because many model types in R are only available in community-created packages where it is up to the author to specify the form of their function's interface. The takeaway here is that when you learn how to use a new package, do not assume that everything will always be the same. Always look up a new package's documentation and learn how to use it.

In R, the most commonly used implementations of ridge regression and other penalized regression methods use matrix specification. It is important to keep in mind these implementations do not exist natively in R and are thus created by community users who later upload their package to CRAN (R's official repository). It is a good skill to learn how to use many types of functions as you will encounter many different types in your R career.

Example: (Ridge Regression; recall Handout 2.6.1 example) A study seeks to relate (in females) amount of body fat (Y) to triceps skinfold thickness (X_1), thigh circumference (X_2), and midarm circumference (X_3). Amount of body fat is expensive to measure, requiring immersion of person in water. This expense motivates the desire for a predictive model based on these inexpensive predictors.

```
# Load the data
library(stat5100)
data(bodyfat)

# Look at the original fit along with VIF:
bodyfat_lm <- lm(body ~ triceps + thigh + midarm, data = bodyfat)

summary(bodyfat_lm)
```

```
##
## Call:
## lm(formula = body ~ triceps + thigh + midarm, data = bodyfat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7263 -1.6111  0.3923  1.4656  4.1277
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  117.085     99.782   1.173    0.258
## triceps        4.334      3.016   1.437    0.170
## thigh         -2.857      2.582  -1.106    0.285
## midarm        -2.186      1.595  -1.370    0.190
##
## Residual standard error: 2.48 on 16 degrees of freedom
## Multiple R-squared:  0.8014,Adjusted R-squared:  0.7641
## F-statistic: 21.52 on 3 and 16 DF,  p-value: 7.343e-06

# VIF:
olsrr::ols_vif_tol(bodyfat_lm)

##   Variables   Tolerance      VIF
## 1   triceps 0.001410750 708.8429
## 2     thigh 0.001771971 564.3434
## 3    midarm 0.009559681 104.6060

# Try ridge regression as a remedial measure
# -----------------------------------------
# We use the glmnet() function inside the glmnet package to do this. Note that
# instead of specifying our model using a formula (formulas in R are of the
# form Y ~ X1 + X2 + X3), we create a dataframe of just our predictor variables
# and a vector of our response variable.
y <- bodyfat$body

# Our X must come in the form of a matrix. First we take out the "body" column
# from the dataframe, and then we convert it to a matrix.
X <- as.matrix(subset(bodyfat, select = -body))

# It is standard to standardize our variables in ridge regression. Note however
# that if you forget to do this, you should be completely fine because
# the implementations of ridge regression should standardize it for you
# if you pass in an unstandardized X matrix.
X <- scale(X)

# Select an optimal value for lambda. In glmnet, set alpha = 0 to do ridge regression.
# This cv function here will selecte a value of lambda for you.
# (Ignore the warning below)
bodyfat_test_ridge_lm <- glmnet::cv.glmnet(X, y, alpha = 0)

## Warning:  Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold

bodyfat_test_ridge_lm

##
## Call:  glmnet::cv.glmnet(x = X, y = y, alpha = 0)
##
```

```
## Measure: Mean-Squared Error
##
##     Lambda Measure   SE Nonzero
## min  0.437   8.102 1.93       3
## 1se  4.473   9.882 2.07       3
```

Here let's pick $\lambda = 0.437$ based upon the above output.

```
# Use the non-cv version to actually create a model that we can use and predict with.
bodyfat_ridge_lm <- glmnet::glmnet(X, y, alpha = 0, lambda = 0.437)
# Looka at coefficients
bodyfat_ridge_lm$beta

## 3 x 1 sparse Matrix of class "dgCMatrix"
##                  s0
## triceps  2.1999416
## thigh    2.2873325
## midarm  -0.4351881

# Store our coefficients. You could do this by manually entering the numbers,
# but I index them here for better automation.
triceps_coef <- bodyfat_ridge_lm$beta[1]
thigh_coef <- bodyfat_ridge_lm$beta[2]
midarm_coef <- bodyfat_ridge_lm$beta[3]
```

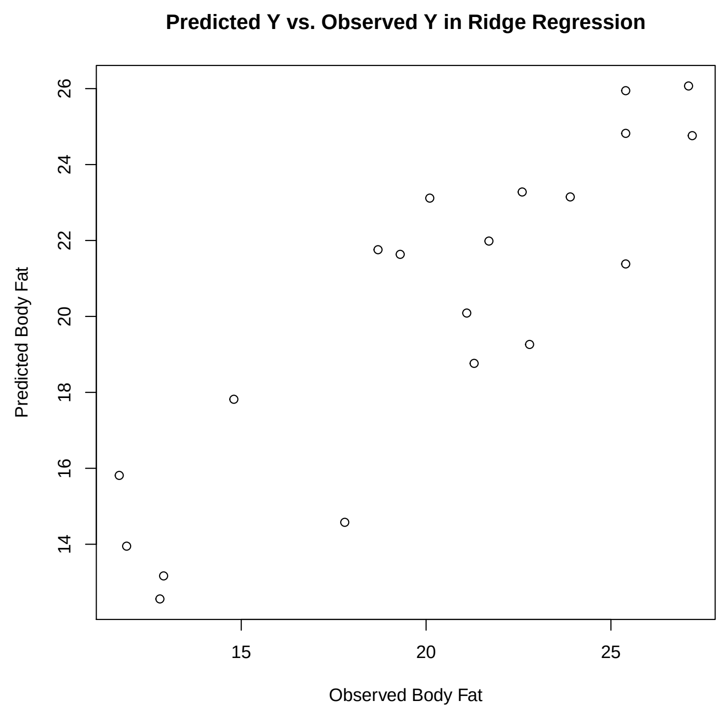
In order to get b_0 for the *unstandardized* coefficients, we use the formula:

$$\beta_0 = \bar{Y} - \beta_1 \bar{X}_1 - \beta_2 \bar{X}_2 - \beta_3 \bar{X}_3$$

```
# Means of various variables
mean(bodyfat$body)

## [1] 20.195

mean(bodyfat$triceps)

## [1] 25.305

mean(bodyfat$thigh)

## [1] 51.17

mean(bodyfat$midarm)

## [1] 27.62

# Crunch our b0 formula:
b0_estimate <- mean(bodyfat$body) - (triceps_coef * mean(bodyfat$triceps)) -
  (thigh_coef * mean(bodyfat$thigh)) - (midarm_coef * mean(bodyfat$midarm))
b0_estimate

## [1] -140.4974
```

Get predicted values in ridge regression

```r
predicted_y <- predict(bodyfat_ridge_lm, X)

# Plot the predicted values vs observed
plot(y, predicted_y, xlab = "Observed Body Fat", ylab = "Predicted Body Fat",
     main = "Predicted Y vs. Observed Y in Ridge Regression")
```



Example 2: Baseball

This data set (from SAS Help: the dataset has been imported into this R package) contains salary (for 1987) and performance (1986 and some career) data for 322 MLB players who played at least one game in both 1986 and 1987 seasons, excluding pitchers. How can salary be predicted from performance?

```r
# Load and take a look at the baseball dataset
data(baseball)
head(baseball)

##                 Name      Team nAtBat nHits nHome nRuns nRBI nBB YrMajor CrAtBat
## 1    Allanson, Andy Cleveland    293    66     1    30   29  14       1     293
## 2       Ashby, Alan   Houston    315    81     7    24   38  39      14    3449
## 3       Davis, Alan   Seattle    479   130    18    66   72  76       3    1624
## 4      Dawson, Andre  Montreal    496   141    20    65   78  37      11    5628
## 5 Galarraga, Andres  Montreal    321    87    10    39   42  30       2     396
## 6   Griffin, Alfredo   Oakland    594   169     4    74   51  35      11    4408
##   CrHits CrHome CrRuns CrRbi CrBB   League Division Position nOuts nAssts
## 1     66      1     30    29   14 American     East        C   446     33
## 2    835     69    321   414  375 National     West        C   632     43
## 3    457     63    224   266  263 American     West       1B   880     82
## 4   1575    225    828   838  354 National     East       RF   200     11
## 5    101     12     48    46   33 National     East       1B   805     40
## 6   1133     19    501   336  194 American     West       SS   282    421
##   nError Salary Div logSalary
```

```
## 1     20     NA  AE        NA
## 2     10  475.0  NW  6.163315
## 3     14  480.0  AW  6.173786
## 4      3  500.0  NE  6.214608
## 5      4   91.5  NE  4.516339
## 6     25  750.0  AW  6.620073

# First, we need to remove all NAs otherwise the algorithm will not work.
baseball <- na.omit(baseball)

# Fit all this into matrix specification
X_baseball <- as.matrix(subset(baseball,
                               select = c(nAtBat, nHits, nHome, nRuns, nRBI,
                                          nBB, YrMajor, CrAtBat, CrHits, CrHome,
                                          CrRuns, CrRbi, CrBB, nOuts, nAssts, nError)))
y_baseball <- baseball$logSalary
```

First, let's use Lasso regression:

```
baseball_lasso_optimal <- glmnet::cv.glmnet(X_baseball, y_baseball, alpha = 1)
baseball_lasso_optimal

##
## Call:  glmnet::cv.glmnet(x = X_baseball, y = y_baseball, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Measure      SE Nonzero
## min 0.01390  0.3508 0.02621       8
## 1se 0.09805  0.3734 0.02771       7

# Pick optimal lambda from the above
baseball_lasso <- glmnet::glmnet(X_baseball, y_baseball,
                                 alpha = 1, lambda = baseball_lasso_optimal$lambda.min)
baseball_lasso$beta

## 16 x 1 sparse Matrix of class "dgCMatrix"
##                    s0
## nAtBat   .
## nHits    6.823840e-03
## nHome    1.638456e-03
## nRuns    .
## nRBI     5.689562e-05
## nBB      5.994047e-03
## YrMajor  6.571314e-02
## CrAtBat  .
## CrHits   2.571496e-04
## CrHome   .
## CrRuns   .
## CrRbi    .
## CrBB     .
## nOuts    1.830800e-04
## nAssts   .
## nError  -6.013340e-03
```

Now, let's show an example with elastic net regression. Here we will pick $\alpha = 0.5$.

```r
baseball_elnet_optimal <- glmnet::cv.glmnet(X_baseball, y_baseball, alpha = 0.5)
baseball_elnet_optimal
```

```
##
## Call:  glmnet::cv.glmnet(x = X_baseball, y = y_baseball, alpha = 0.5)
##
## Measure: Mean-Squared Error
##
##      Lambda Measure      SE Nonzero
## min 0.02308  0.3536 0.03368      11
## 1se 0.21522  0.3822 0.03269       9
```

```r
# Pick optimal lambda from the above
baseball_elnet <- glmnet::glmnet(X_baseball, y_baseball,
                                 alpha = 0.5, lambda = baseball_elnet_optimal$lambda.min)
baseball_elnet$beta
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                    s0
## nAtBat   .
## nHits    6.248992e-03
## nHome    3.635330e-04
## nRuns    6.724745e-04
## nRBI     7.479326e-04
## nBB      5.710352e-03
## YrMajor  6.223543e-02
## CrAtBat  3.613491e-06
## CrHits   2.369305e-04
## CrHome   .
## CrRuns   6.582984e-05
## CrRbi    .
## CrBB     .
## nOuts    1.933099e-04
## nAssts   .
## nError  -6.201626e-03
```