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

Stat 5100: Dr. Bean

Matrix Specification

Up to this point in the course, we have specified models using formula notation such as:

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

However, there are some functions in R that require a matrix specification. This requires us to organize the explanatory variables as a matrix (X) (performing variable transformations beforehand) and the response variable as a vector (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 i^{th} row of X and the j^{th} column of X) would refer to the recorded value of x_j (the j^{th} predictor variable for the the i^{th} observation in our data.

The matrix vs formula model specification is a product of open source software: each R package software contributor requires different model inputs in order for their functions to run properly. This in mind, always be sure to study the function documentation before you start trying to use a new R package. While there are formula-based version of the penalized regression techniques, these notes use a package that requires matrix style inputs due to its ease of implementation and consistency across the three main penalized regression approaches.

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))

# Ridge regression requires that we first standarize our explanatory variables.
# However, the glmnet implementation does this automatically for you
# within the function so we do not need to standardize prior to use.

# Rather than select an optimal value of lambda using the trace plot
# (see 4.1 notes), we will select an optimal value for lambda by minimizing
# the 5-fold cross validated error.
set.seed(123) # Set seed for reproducibility.
bodyfat_test_ridge_lm <- glmnet::cv.glmnet(X, y, alpha = 0, nfolds = 5)
bodyfat_test_ridge_lm

##
## Call:  glmnet::cv.glmnet(x = X, y = y, nfolds = 5, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  0.437   100   7.247 1.195       3
## 1se  2.125    83   8.434 2.337       3
```

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

```r
# 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)
# Look at coefficients
bodyfat_ridge_lm$beta
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##                 s0
## triceps  0.4379510
## thigh    0.4369632
## midarm  -0.1193229
```

```r
# Store our coefficients. You could do this by manually entering the numbers,
# but we 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:

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

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

```
## [1] 20.195
```

```r
mean(bodyfat$triceps)
```

```
## [1] 25.305
```

```r
mean(bodyfat$thigh)
```

```
## [1] 51.17
```

```r
mean(bodyfat$midarm)
```

```
## [1] 27.62
```

```r
# 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] -9.95106
```

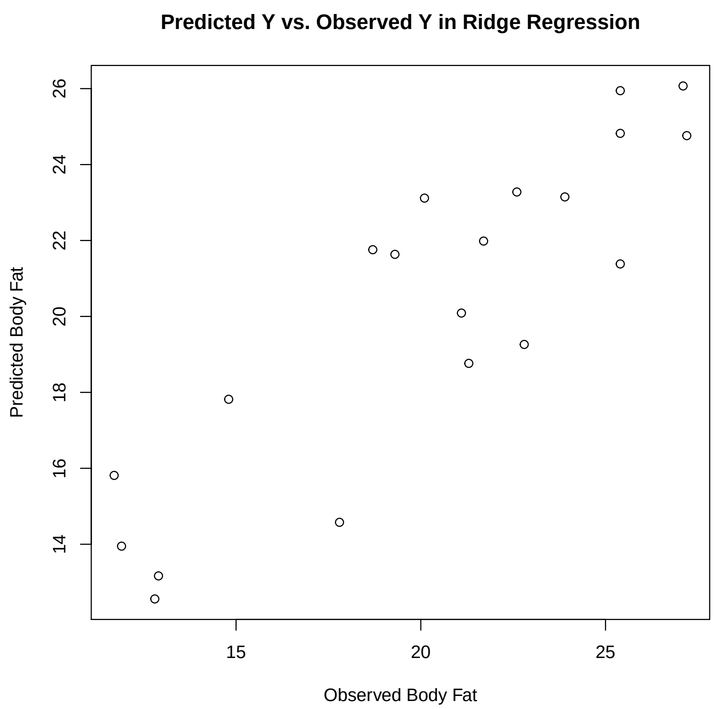
```r
# Confirm that our "by hand" approach matches the intercept calculated
# automatically by glmnet.
bodyfat_ridge_lm$a0
```

```
##       s0
## -9.95106
```

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

# Subset the dataset to retain only variables that are relevant for prediction
# and remove all NAs prior to model input (algorithm fails if NAs are retained)
baseball_sub <- subset(baseball, select = c(logSalary, nAtBat, nHits,
                                            nHome, nRuns, nRBI, nBB, YrMajor,
                                            CrAtBat, CrHits, CrHome, CrRuns,
                                            CrRbi, CrBB, nOuts, nAssts, nError,
                                            League, Division))

baseball_sub <- na.omit(baseball_sub)

# Take log-transformation of salary and create design matrix.
# Note that this function only retains observations with no missing values
# for the variables specified in the formula. We removed missing values
# prior to this step in order to keep X (baseball_design) and y
# (baseball_sub£logSalary) aligned.
# The [, -1] omits the intercept column since glm will fit one automatically
baseball_design <- model.matrix(object = logSalary ~ nAtBat + nHits + nHome +
                                  nRuns + nRBI + nBB + YrMajor + CrAtBat +
                                  CrHits + CrHome + CrRuns + CrRbi + CrBB +
                                  nOuts + nAssts + nError + League + Division,
                                data = baseball_sub)[, -1]
```

First, let's use Lasso regression:

```
baseball_lasso_optimal <- glmnet::cv.glmnet(baseball_design,
                                            baseball_sub$logSalary,
                                            alpha = 1)
baseball_lasso_optimal$lambda.min

## [1] 0.01525366

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

## 18 x 1 sparse Matrix of class "dgCMatrix"
##                           s0
## nAtBat          .
## nHits           0.0067888515
## nHome           0.0025273020
## nRuns           .
## nRBI            .
## nBB             0.0057078209
## YrMajor         0.0653074206
## CrAtBat         .
## CrHits          0.0002562012
## CrHome          .
```

```
## CrRuns           .
## CrRbi            .
## CrBB             .
## nOuts            0.0001707651
## nAssts           .
## nError          -0.0060889931
## LeagueNational  0.0645659173
## DivisionWest    -0.1255002632

baseball_lasso$a0

##       s0
## 4.283545
```

Now, let's show an example with elastic net regression. This is specified by using some alpha between 0 and 1. For simplicity, we we will pick $\alpha = 0.5$.

```
baseball_elnet_optimal <- glmnet::cv.glmnet(baseball_design,
                                            baseball_sub$logSalary,
                                            alpha = 0.5)
baseball_elnet_optimal$lambda.min

## [1] 0.02532772

# Pick optimal lambda from the above
baseball_elnet <- glmnet::glmnet(baseball_design,
                                 baseball_sub$logSalary,
                                 alpha = 0.5,
                                 lambda = baseball_elnet_optimal$lambda.min)

# Obtain beta estimates (including intercept)
baseball_elnet$beta

## 18 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## nAtBat           .
## nHits            6.406212e-03
## nHome            2.546061e-03
## nRuns            5.455873e-04
## nRBI             6.043526e-05
## nBB              5.554989e-03
## YrMajor          6.114898e-02
## CrAtBat          1.438093e-05
## CrHits           2.344433e-04
## CrHome           .
## CrRuns           3.177752e-06
## CrRbi            .
## CrBB             .
## nOuts            1.798556e-04
## nAssts           .
## nError          -6.376027e-03
## LeagueNational  6.945671e-02
## DivisionWest    -1.287318e-01

baseball_elnet$a0

##       s0
## 4.303893
```