Lab 3 - LASSO and Ridge Regularized Regressions

Collee McCamy

2023-01-25

Lab 3: Predicting the age of abalone

Abalones are marine snails. Their flesh is widely considered to be a desirable food, and is consumed raw or cooked by a variety of cultures. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age.

The data set provided includes variables related to the sex, physical dimensions of the shell, and various weight measurements, along with the number of rings in the shell. Number of rings is the stand-in here for age.

Data Exploration

Pull the abalone data from Github and take a look at it.

```
abdat <- dat <- read_csv(file = "https://raw.githubusercontent.com/MaRo406/eds-232-machine-learning/mai
    janitor::clean_names()
```

```
## New names:
## Rows: 4177 Columns: 10
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): Sex dbl (9): ...1, Length, Diameter, Height, Whole_weight, Shucked_weight,
## Visce...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
glimpse(abdat)
```

```
## Rows: 4,177
## Columns: 10
## $ x1             <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
## $ sex            <chr> "M", "M", "F", "M", "I", "I", "F", "F", "M", "F", "F", ~
## $ length         <dbl> 0.455, 0.350, 0.530, 0.440, 0.330, 0.425, 0.530, 0.545,~
## $ diameter       <dbl> 0.365, 0.265, 0.420, 0.365, 0.255, 0.300, 0.415, 0.425,~
## $ height         <dbl> 0.095, 0.090, 0.135, 0.125, 0.080, 0.095, 0.150, 0.125,~
## $ whole_weight   <dbl> 0.5140, 0.2255, 0.6770, 0.5160, 0.2050, 0.3515, 0.7775,~
## $ shucked_weight <dbl> 0.2245, 0.0995, 0.2565, 0.2155, 0.0895, 0.1410, 0.2370,~
## $ viscera_weight <dbl> 0.1010, 0.0485, 0.1415, 0.1140, 0.0395, 0.0775, 0.1415,~
## $ shell_weight   <dbl> 0.150, 0.070, 0.210, 0.155, 0.055, 0.120, 0.330, 0.260,~
## $ rings          <dbl> 15, 7, 9, 10, 7, 8, 20, 16, 9, 19, 14, 10, 11, 10, 10, ~
```

Data Splitting

- ***Question 1***. Split the data into training and test sets. Use a 70/30 training/test split.

```r
# setting the seed for reproducibility
set.seed(123)

# creating a split training set of 70/30 training/test
split <- initial_split(data = abdat, prop = 0.7, strata = rings)

# establishing this split as datasets to use in training & testing
abdat_train <- training(split)
abdat_test <- testing(split)
```

We'll follow our text book's lead and use the caret package in our approach to this task. We will use the glmnet package in order to perform ridge regression and the lasso. The main function in this package is glmnet(), which can be used to fit ridge regression models, lasso models, and more. In particular, we must pass in an x matrix of predictors as well as a y outcome vector , and we do not use the y x syntax.

Fit a ridge regression model

- ***Question 2***. Use the model.matrix() function to create a predictor matrix, x, and assign the Rings variable to an outcome vector, y.

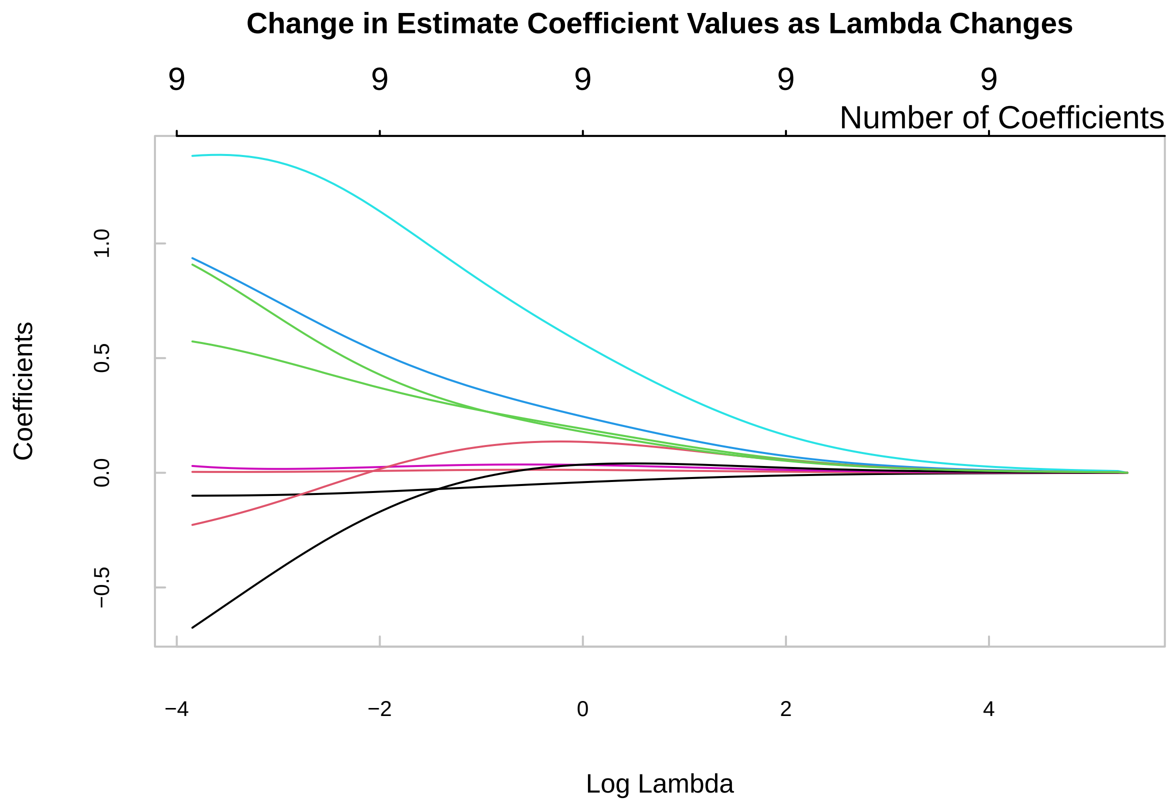
```r
# creating a predictor matrix with rings as the outcome variable and all
# predictors included, removing the intercept
X <- model.matrix(rings ~ ., data = abdat)[, -c(1, 2)]

# taking the log of the outcome
Y <- log(abdat$rings)
```

- ***Question 3***. Fit a ridge model (controlled by the alpha parameter) using the glmnet() function. Make a plot showing how the estimated coefficients change with lambda. (Hint: You can call plot() directly on the glmnet() objects).

```r
# fitting a ridge model with alpha equal to zero
ridge <- glmnet(x = X, # specify the data (the model matrix created above)
                y = Y, # specify the outcome (log of number of rings)
                alpha = 0)

# plotting the changes in the coefficients as lambda increases
plot_ridge <- plot(ridge, xvar = "lambda",
     tck = 0.02,
     cex.main = 0.9,
     cex.lab = 0.8,
     cex.axis = 0.7,
     family = "sans",
     fg = "#c4c4c4",
     main = "Change in Estimate Coefficient Values as Lambda Changes\n\n")
mtext("Number of Coefficients", side = 3,adj = 1)
```



Using *k*-fold cross validation resampling and tuning our models

In lecture we learned about two methods of estimating our model's generalization error by resampling, cross validation and bootstrapping. We'll use the *k*-fold cross validation method in this lab. Recall that lambda is a tuning parameter that helps keep our model from over-fitting to the training data. Tuning is the process of finding the optima value of lamba.

- ***Question 4***. This time fit a ridge regression model and a lasso model, both with using cross validation. The glmnet package kindly provides a cv.glmnet() function to do this (similar to the glmnet() function that we just used). Use the alpha argument to control which type of model you are running. Plot the results.

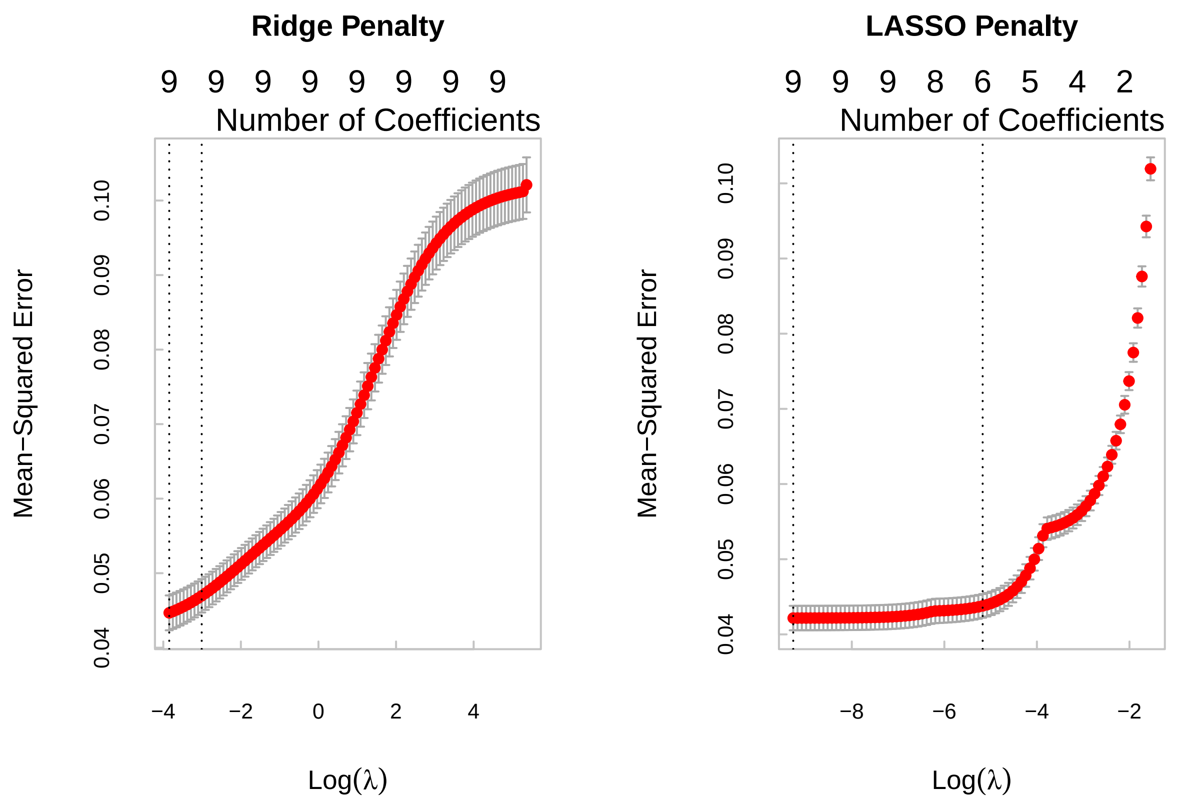
```r
# fitting a ridge model with a k-fold cross validation
cv_ridge <- cv.glmnet(x = X, # specify the data
                y = Y, # specify the outcome - log of number of rings
                alpha = 0) # alpha of 0 is equal to a true ridge

# fitting a lasso model with a k-fold cross validation
cv_lasso <- cv.glmnet(
  x = X, # specify the data
  y = Y, # specify the outcome - log of number of rings
  alpha = 1) # alpha of 1 is equal to a true LASSO

# setting up a container for the plots
par(mfrow = c(1, 2))
```

```r
# plotting the cross validation ridge
plot(cv_ridge,
     tck = 0.02,
     cex.main = 0.9,
     cex.lab = 0.8,
     cex.axis = 0.7,
     family = "sans",
     fg = "#c4c4c4",
     main = "Ridge Penalty\n\n")
mtext("Number of Coefficients", side = 3,adj = 1)

# plotting the cross validation LASSO
plot(cv_lasso,
     tck = 0.02,
     cex.main = 0.9,
     cex.lab = 0.8,
     cex.axis = 0.7,
     family = "sans",
     fg = "#c4c4c4",
     main = "LASSO Penalty\n\n",
     col = "#0081a7")
mtext("Number of Coefficients", side = 3,adj = 1)
```



- ***Question 5***. Interpret the graphs. What is being show on the axes here? How does the performance of the models change with the value of lambda?

  - The two graphs compare the Mean-Standard Error (MSE) and log of lambda for ridge and LASSO regression analysis. To break this down, the MSE is the difference between model's predictions and the measured true values. The best regression model is on the minimizes the MSE. Thus, plotting the MSE against the log of lambda on the x-axis can indicate the optimal log lambda values to minimize the MSE.
  - The dotted line closest to the y-axis indicates the log lambda value that minimizes the MSE on both the Ridge and LASSO penalty graphs. The second dotted line indicates the largest log lambda value that falls within one standard error away from the log lambda value that minimizes the MSE.
  - Interpreting these graphs, we can see for the ridge penalty, as lambda increases the MSE also increases. We can also see that for the LASSO penalty, as lambda increases the MSE remains mostly constant until log lambda increases past -5. Then the MSE increases rapidly.
  - In addition, we can see that the number of coefficients decrease as lambda increases for the LASSO penalty. Using the rule of one standard error and wanting the most parsimonious model, selecting a lambda value of -5 could provide optimal predictive results for the LASSO regression.
  - These graphs also suggest that a standard OLS model would likely overfit the training data, as we see MSE decrease as we constrain the model.

- ***Question 6***. Inspect the ridge model object you created with cv.glmnet(). The $cvm column shows the MSEs for each cv fold. What is the minimum MSE? What is the value of lambda associated with this MSE minimum?

```r
# looking at all values first
summary(cv_ridge)
```

```
##            Length Class  Mode
## lambda     100    -none- numeric
## cvm        100    -none- numeric
## cvsd       100    -none- numeric
## cvup       100    -none- numeric
## cvlo       100    -none- numeric
## nzero      100    -none- numeric
## call         4    -none- call
## name         1    -none- character
## glmnet.fit  12    elnet  list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric
```

```r
# selecting the minimum MSE value
min_mse_r <- min(cv_ridge$cvm)
# printing the answer
print(paste0("The minimum MSE value is ", round(min_mse_r, 3), " for the ridge k-fold cross-validation
```

```
## [1] "The minimum MSE value is 0.045 for the ridge k-fold cross-validation regression."
```

```r
# selecting the minimum lambda value at the minimum MSE value
min_lam_r <- cv_ridge$lambda.min
# printing the answer
print(paste0("The minimum lambda value for when the MSE is the lowest is ", round(min_lam_r,
    3), " for the ridge k-fold cross-validation regression."))
```

```
## [1] "The minimum lambda value for when the MSE is the lowest is 0.021 for the ridge k-fold cross-vali
```

Response:

- **The minimum MSE value is 0.045 rings-squared for the ridge k-fold cross-validation regression.**
- **The minimum lambda value for when the MSE is the lowest is 0.021 for the ridge k-fold cross-validation regression.**

Question 7. Do the same for the lasso model. What is the minimum MSE? What is the value of lambda associated with this MSE minimum?

```r
# looking at all values first
summary(cv_lasso)
```

```
##            Length Class  Mode
## lambda     84     -none- numeric
## cvm        84     -none- numeric
## cvsd       84     -none- numeric
## cvup       84     -none- numeric
## cvlo       84     -none- numeric
## nzero      84     -none- numeric
## call        4     -none- call
## name        1     -none- character
## glmnet.fit 12     elnet  list
## lambda.min  1     -none- numeric
## lambda.1se  1     -none- numeric
## index       2     -none- numeric
```

```r
# selecting the minimum MSE value
min_mse_l <- min(cv_lasso$cvm)
# printing the answer
print(paste0("The minimum MSE value is ", round(min_mse_l, 3), " for the LASSO k-fold cross-validation r
```

```
## [1] "The minimum MSE value is 0.042 for the LASSO k-fold cross-validation regression."
```

```r
# selecting the minimum lambda value at the minimum MSE value
min_lam_l <- cv_lasso$lambda.min
# printing the answer
options(scipen = 999)  # changing from scientific notation to standard form
print(paste0("The minimum lambda value for when the MSE is the lowest is ", round(min_lam_l,
    4), " for the LASSO k-fold cross-validation regression."))
```

```
## [1] "The minimum lambda value for when the MSE is the lowest is 0.0001 for the LASSO k-fold cross-val
```

Response:

- **The minimum MSE value is 0.042 rings-squared for the LASSO k-fold cross-validation regression.**

- **The minimum lambda value for when the MSE is the lowest is 0.0001 for the LASSO k-fold cross-validation regression.**

Data scientists often use the “one-standard-error” rule when tuning lambda to select the best model. This rule tells us to pick the most parsimonious model (fewest number of predictors) while still remaining within one standard error of the overall minimum cross validation error. The cv.glmnet() model object has a column that automatically finds the value of lambda associated with the model that produces an MSE that is one standard error from the MSE minimum ($lambda.1se).

- ***Question 8.*** Find the number of predictors associated with this model (hint: the $nzero is the # of predictors column).

```r
summary(cv_lasso)
```

```
##            Length Class  Mode
## lambda     84     -none- numeric
## cvm        84     -none- numeric
## cvsd       84     -none- numeric
## cvup       84     -none- numeric
## cvlo       84     -none- numeric
## nzero      84     -none- numeric
## call        4     -none- call
## name        1     -none- character
## glmnet.fit 12     elnet  list
## lambda.min  1     -none- numeric
## lambda.1se  1     -none- numeric
## index       2     -none- numeric
```

```r
# finding the value of lambda at 1-SE from minimum MSE lambda value
mse_1se <- cv_lasso$cvm[cv_lasso$lambda == cv_lasso$lambda.1se]

# finding the value of lambda at 1-SE from minimum MSE lambda value
lambda_1se <- cv_lasso$lambda.1se

# finding the number of predictors when lambda is at 1-SE value
n_coef_1se <- cv_lasso$nzero[cv_lasso$lambda == cv_lasso$lambda.1se]

# printing the results
print(paste0("There are ", n_coef_1se, " number of predictors when tuning the model so that lambda is a
```

```
## [1] "There are 6 number of predictors when tuning the model so that lambda is at the maximum value t
```

Response: There are 6 number of predictors when tuning the model so that lambda is at the maximum value that falls within one standard error from the value when MSE is minimized.

- **Question 9.** Which regularized regression worked better for this task, ridge or lasso? Explain your answer.
  - **Response: The LASSO regularized regression works best for this task as we were able to minimize the MSE and number of coefficients to make a parsimonious model, given that interpreting all predictors are not important to this task. This is important because a simple model can better create predictions and can require less computing needs to run. The MSE is also a little bit lower for the LASSO cross validation regression at 0.042 rings-squared compared to the Ridge cross validation regression at 0.045 rings-squared. Thus, we can get approximately the same accuracy with only using 6 predictors, in the LASSO regression, compared to 9 predictors in the Ridge regression.**