Biased Estimation: LASSO and Elastic Net

Andrew Zieffler
November 06, 2020

The data in *equal-educational-opportunity.csv* consist of data taken from a random sample of 70 schools in 1965 (Chatterjee & Hadi, 2012; Coleman et al., 1966; Mosteller & Moynihan, 1972). We will use these data to mimic one of the original regression analyses performed; examining whether the level of school facilities was an important predictor of student achievement after accounting for the variation in faculty credentials and peer influence.

```
# Load libraries
library(broom)
library(glmnet)
library(MASS)
library(tidyverse)


# Read in data
eeo = read_csv("../data/equal-education-opportunity.csv")
head(eeo)
```

```
## # A tibble: 6 x 4
##   achievement faculty    peer school
##         <dbl>   <dbl>   <dbl>  <dbl>
## 1      -0.431   0.608  0.0351  0.166
## 2       0.800   0.794  0.479   0.534
## 3      -0.925  -0.826 -0.620  -0.786
## 4      -2.19   -1.25  -1.22   -1.04
## 5      -2.85    0.174 -0.185   0.142
## 6      -0.662   0.202  0.128   0.273
```

To examine the RQ, the following model was posited:

$$\text{Achievement}_i = \beta_0 + \beta_1(\text{Faculty}_i) + \beta_2(\text{Peer}_i) + \beta_3(\text{School}_i) + \epsilon_i$$

Recall that, unfortunately, the model suffered from strong collinearity resulting in poor estimates of the coefficients and very large SEs. This means we cannot effectively control for peer influence and faculty credentials, which means we cannot answer the RQ.

Ridge Regression: Revisited

One method of dealing with collinearity is to use a biased estimation method. Previously we introduced ridge regression as one such biased estimation method. One problem with fitting a ridge regression model is choosing the d value, or ridge penalty. One good way of choosing d is via cross-validation.

The `glmnet()` function can uses k-fold cross-validation to select a d value. The algorithm to determine the d value is:

- Split the dataset randomly into 10 groups
- For each unique group:
    - Hold the group out as test data set
    - Take the remaining 9 groups and use them as a training data set
    - Fit a ridge model on the training set with a specific value of d
    - Evaluate the model using the test set by computing the error (deviance)
    - Repeat for different values of d
- Summarize the performance for the various d values by averaging across the 10 different tests sets
- Choose the d value that gives the lowest average amount of error

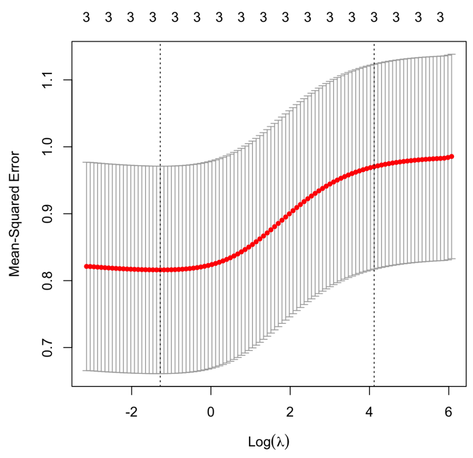
To use cross-validation for d selection, we perform the cross-validation using the `cv.glmnet()` function. Then, we can plot the results and extract the “best” d value.

```
# Get matrices for use in glmnet
X = scale(eeo)[ , 2:4]
Y = scale(eeo)[ , 1]

# Make results reproducible
set.seed(42)

# Carry out the cross-validation
ridge_cv = cv.glmnet(
  x = X,
  y = Y,
  alpha = 0,
  intercept = FALSE,
  standardize = FALSE
  )

# Extract the lambda value with the lowest mean error
ridge_cv$lambda.min
```

```
## [1] 0.2786974
```

Here the d_{Modified} value with the lowest mean error from the cross-validation results is 0.2786974. This modified λ value could be used directly in the `lambda=` argument of the `glmnet()` function, or could be transformed (by multiplying it by n) for use in the matrix algebra equation. We can also see a plot of the CV-MSE values plotted against $\ln(d)$.

```
plot(ridge_cv)
```



Cross-validated MSE (with confidence bands), as a function of ln(d) for several d values.

The *d* value of 0.2786974 value is also associated with the $\ln \lambda$ value that corresponds to the lowest MSE from the plot.

```
# Fit ridge regression
ridge_best = glmnet(
  x = X,
  y = Y,
  alpha = 0,
  lambda = ridge_cv$lambda.min,
  intercept = FALSE,
  standardize = FALSE
  )

# Show coefficients
tidy(ridge_best)
```

```
## # A tibble: 3 x 5
##   term     step estimate lambda dev.ratio
##   <chr>   <dbl>    <dbl>  <dbl>     <dbl>
## 1 faculty     1   0.116   0.279     0.185
## 2 peer        1   0.180   0.279     0.185
## 3 school      1   0.0987  0.279     0.185
```

Based on the *d* value chosen from the cross-validation, the fitted equation is:

$$\hat{\text{Achievement}}_i = 0.116(\text{Faculty}_i) + 0.180(\text{Peer}_i) + 0.100(\text{School}_i)$$

where all of the variables are standardized.

Using the lm.ridge() Function with Cross-Validation

We can also use cross-validation to choose the *d* value using the `lm.ridge()` function. To do so, we fit a sequence of potential *d* values into the `lambda=` argument. Then we can use the `select()` function from the **MASS** package to obtain the optimal *d* value.

```
# Standardize all the variables and turn into a data frame
z_eeo = scale(eeo) %>%
  data.frame()

# Carry out the cross-validation
ridge_cv_2 = lm.ridge(achievement ~ -1 + faculty + peer + school, data =
        z_eeo,
                      lambda = seq(from = 0, to = 50, by = 0.001))

# Obtain d
MASS::select(ridge_cv_2)
```

```
## modified HKB estimator is 0.3729338
## modified L-W estimator is 4.020601
## smallest value of GCV  at 22.436
```

The `select()` function optimized *d* based on the generalized cross-validation (GCV) statistic. (The `MASS::` before the `select()` function forces R to use the `select()` fuction from the **MASS** package rather than the `select()` function from the **tidyverse** package.) Based on this metric, the optimal *d* value is 22.436. This is different from the value chosen by the `cv.glmnet()` function, which was $0.279 \times 70 = 19.53$. The primary reason for this is the cross-validation method used. The modified *d* value obtained by the `cv.glmnet()` function is based on k-fold cross-validation, while the GCV is more akin to LOOCV.

Once we have identified *d*, e can fit the ridge model:

```
# Fit ridge model
ridge_best_2 = lm.ridge(achievement ~ -1 + faculty + peer + school, data =
        z_eeo, lambda = 4.02)

# Obtain coefficients
tidy(ridge_best_2)
```

```
## # A tibble: 3 x 5
##   lambda    GCV term    estimate scale
##    <dbl>  <dbl> <chr>      <dbl> <dbl>
## 1   4.02 0.0119 faculty  0.111   0.993
## 2   4.02 0.0119 peer     0.318   0.993
## 3   4.02 0.0119 school  -0.00616 0.993
```

Based on the *d* value chosen from the GCV, the fitted equation is:

$$\hat{\text{Achievement}}_i = 0.111(\text{Faculty}_i) + 0.318(\text{Peer}_i) - 0.006(\text{School}_i)$$

Least Absolute Shrinkage and Selection Operator (LASSO)

One disadvantage to ridge regression is that although the coefficients are “shrunk”, they are not set to 0 (unless $d = \infty$). In some cases this is fine, but in many applications it may be of interest to find the subset of correlated predictors that are useful. This is especially true when your data have a large number of predictors and a limited number of cases (the so-called "large p, small n problems). In such cases, the LASSO (Tibshirani, 1996) is an alternative to ridge regression. Similar to ridge regression, the LASSO finds the coefficients that minimize a penalized sum of squares, namely:

$$\text{SSE}_{\text{Penalized}} = \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2 + d \sum_{j=1}^{p} |\beta_j|$$

If we compare this to ridge regression, the only difference is in the penalty term. Namely, the penalty term for ridge regression was based on the sum of the squared beta terms while that for the LASSO is based on the sum of the absolute value for the beta terms.

In statistical nomenclature, the LASSO uses an L1 penalty and ridge regression uses an L2 penalty. This terminology is based on the norm (magnitude) of the vector of beta coefficients. The L1 norm is the sum of the absolute values (Manhattan distance) and the L2 norm is the sum of the squared values (Euclidean distance).

Similar to ridge regression, the penalty term has the effect of biasing some coefficients by shrinking them toward zero. The LASSO, however, will shrink some coefficients to 0; effectively removing them from the model. We can also use the `glmnet()` function from the **glmnet** package to fit a LASSO. To fit a ridge regression we provide the following arguments to `glmnet()` :

- `x=` : A matrix of the predictors
- `y=` : A vector of the outcome
- `alpha=1` : This fits a LASSO
- `lambda=` : This sets a modified d value which is equal to $\frac{d}{n}$
- `intercept=FALSE` : We will include this argument to drop the intercept term; appropriate for standardized regression.

- `standardize=` : Here we set this to `FALSE` since we pre-standardized the predictors and outcome. The default is `standardize=TRUE` .

Choosing the LASSO Penalty

Again, we have to choose the modified *d* value to put it the `glmnet()` function. To select *d*, we will again use cross-validation.

```
# Make reproducible
set.seed(100)

# Fit cross-validated LASSO
lasso_1 = cv.glmnet(
  x = X,
  y = Y,
  alpha = 1,
  intercept = FALSE,
  standardize = FALSE
  )

# Extract d value with lowest CV-MSE
lasso_1$lambda.min
```

```
## [1] 0.02423968
```

Using this value in the LASSO:

```
# Fit LASSO
best_lasso = glmnet(
  x = X,
  y = Y,
  alpha = 1,
  lambda = lasso_1$lambda.min,
  intercept = FALSE,
  standardize = FALSE
  )

# Show coefficients
tidy(best_lasso)
```

```
## # A tibble: 1 x 5
##   term   step estimate lambda dev.ratio
##   <chr> <dbl>    <dbl>  <dbl>     <dbl>
## 1 peer      1    0.415 0.0242     0.193
```

Based on using the modified *d* value of 0.042, the fitted LASSO model is:

$$\hat{\text{Achievement}}_i = 0.397(\text{Peer}_i)$$

where the variables are all standardized.

Note that the coefficients for faculty credentials and school facilities were both shrunk to zero! Computing standard errors for LASSO coefficients are problematic in the same way that computing SEs are problematic for the ridge regression; namely we have no good estimate of the true amount of bias. Lastly, we note that, as in ridge regression, selecting a good value of *d* for the LASSO is critical.

Although better than ridge regression, the LASSO is not a panacea for model selection under collinearity. As we saw in our example, when the predictors are highly correlated, the LASSO will tend to select a single variable from the set of predictors and ignore the others. It also will only choose at most *n* variables when there are a large number of predictors and small number of cases (large *p*; small *n*).

Elastic Net

A generalization of the ridge and lasso penalties, introduced by Zou & Hastie (2005) is called the *elastic net*. This combines the two penalties and subsequently minimizes a double-penalized sum of squares, namely:

$$\text{SSE}_{\text{Penalized}} = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 + \left(d_1 \sum_{j=1}^{p} \beta_j^2 + d_2 \sum_{j=1}^{p} |\beta_j|\right)$$

The elastic net has two penalty parameters, namely d_1 and d_2. As can be seen by this equation, the ridge regression and LASSO are special cases of the elastic net. This reduces to the ridge regression model when $d_2 = 0$ and to the LASSO when $d_1 = 0$. When both penalty parameters are non-zero, this results in a mixture of both the L1 and L2 penalties.

Recall that this is simply,

$$\text{SSE}_{\text{Penalized}} = \text{SSE} + \text{Penalty}$$

In the `glmnet()` function we set an argument `alpha=` . This is actually a mixture parameter that defines the penalty term:

$$\text{Penalty} = \frac{1-\alpha}{2}\left(\beta_j^2\right) + \alpha\left(|\beta_j|\right)$$

When $\alpha = 0$ the penalty is a scaled version of the L1 penalty, β_j^2; namely ridge regression. (Dividing the ridge penalty by 2 is a mathematical convenience for optimization.) When $\alpha = 1$ the penalty is the L2 penalty; namely LASSO. To include both the L1 and L2 penalties we set `alpha=0.5` .

```
# Make reproducible
set.seed(1000000)

# Fit cross-validated elastic net
elastic_net_1 = cv.glmnet(
  x = X,
  y = Y,
  alpha = 0.5,
  intercept = FALSE,
  standardize = FALSE
  )

# Extract d value with lowest CV-MSE
elastic_net_1$lambda.min
```

```
## [1] 0.03044645
```

We re-fit the elastic net using this modified *d* value in the `lambda=` argument.

```
# Fit elastic net
best_elastic_net = glmnet(
  x = X,
  y = Y,
  alpha = 0.5,
  lambda = elastic_net_1$lambda.min,
  intercept = FALSE,
  standardize = FALSE
  )

# Get coefficients
tidy(best_elastic_net)
```

```
## # A tibble: 2 x 5
##   term     step estimate lambda dev.ratio
##   <chr>   <dbl>    <dbl>  <dbl>     <dbl>
## 1 faculty     1   0.0283 0.0304     0.193
## 2 peer        1   0.391  0.0304     0.193
```

the fitted elastic net model is:

$$\hat{\text{Achievement}}_i = 0.391(\text{Peer}_i) + 0.028(\text{Faculty}_i)$$

where the variables are all standardized. Here the model still finds a moderate positive effect of peer influence, but there is also a small effect of faculty. Although shrunken toward zero relative to the ridge regression model, it is not 0 as in the LASSO.

Non-Collinear Predictors

These methods can also be used when the predictors are not collinear. The elastic net or LASSO is a good option in this case as both essentially do automatic variable selection (by shrinking some of the coefficients to zero) and perform coefficient estimation, simultaneaously. Here we use the elastic net, as the degree of shrinkage is less than the LASSO, to select a model for our life expectancy model using the *states-2019.csv* data.

```
usa = read_csv("../data/states-2019.csv")

X = usa[ , 3:9] %>% scale()
Y = usa[ , 2] %>% scale()

# Make reproducible
set.seed(3)

# Fit cross-validated elastic net
en_results = cv.glmnet(
  x = X,
  y = Y,
  alpha = 0.5,
  intercept = FALSE,
  standardize = FALSE
  )

# Extract d value with lowest CV-MSE
en_results$lambda.min
```

```
## [1] 0.5087431
```

```
# Re-fit elastic net
en_model = glmnet(
  x = X,
  y = Y,
  alpha = 0.5,
  lambda = en_results$lambda.min,
  intercept = FALSE,
  standardize = FALSE
  )

# Get coefficients
tidy(en_model)
```

```
## # A tibble: 2 x 5
##   term    step estimate lambda dev.ratio
##   <chr>  <dbl>    <dbl>  <dbl>     <dbl>
## 1 income     1   0.178   0.509     0.162
## 2 murder     1  -0.0290  0.509     0.162
```

The model slected is:

$$\text{Life Exp}\hat{\text{e}}\text{ctancy}_i = 0.195(\text{Income}_i) - 0.042(\text{Murder Rate}_i)$$

The model selected the predictors of income, and murder rate. The problem with using biased regression methods when there is not a collinearity issue is that it unnecessarily introduces bias into the coefficient estimates—the effects are smaller than when estimated with OLS. So, while it selected a small subset of predictors, we may not have reasonable estimates of the effects. It is not recommended to use shrinkage methods when there is little/no collinearity problems.

References

Chatterjee, S., & Hadi, A. S. (2012). *Regression analysis by example*. Wiley.

Coleman, J. S., Cambell, E. Q., Hobson, C. J., McPartland, J., Mood, A. M., Weinfield, F. D., & York, R. L. (1966). *Equality of educational opportunity*. U.S. Government Printing Office.

Mosteller, F., & Moynihan, D. F. (1972). *On equality of educational opportunity*. Random House.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267–288.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(2), 301–320.