Penalized Logistic Regression

Nestor Mendoza

March 28, 2025

Overview

1. Define Linear Regression, Penalized Regression and Penalized Logistic Regression.
2. Define Lasso, Ridge and Elastic Net Regression.
3. Comment on the advantages/disadvantages of each type of Penalized Regression.
4. See the implementation of Lasso, Ridge and Elastic Net Regression using R packages.

Linear Regression

Linear regression is defined as a model that summarizes data by the linear equation

$$y = \beta_0 + \sum_{i=1}^{n} x_i \beta_i + \varepsilon.$$

Where the β_i are fixed coefficients that are found by a minimizing process.

Penalized Regression

Penalized Regression is a type of Regression that adds a penalty term to the minimizing process used to find the fixed coefficients.

The purpose of this penalization is to shrink the fixed coefficients.

Types of Penalized Regression

There are 3 main types of Penalized Regression: Lasso Regression, Ridge Regression and Elastic Net Regression.

We will define these types of regression later on.

Logistic Regression

Logistic Regression is a type of Regression in which we look to fit the linear model

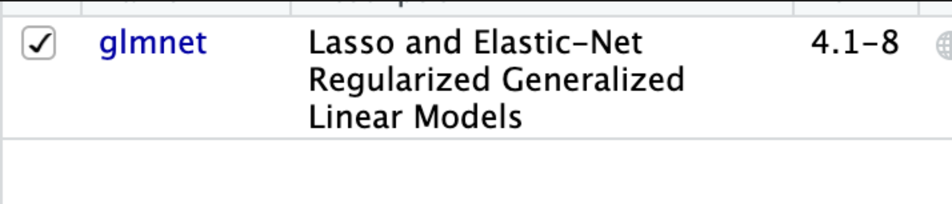
$$\mathrm{logit}(p_{x_1,\ldots,x_k}) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i.$$

Logistic Penalized Regression

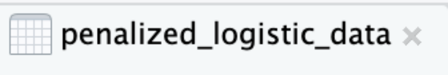
Logistic Penalized Regression is the implementation of a penalty term to the calculation of the fixed coefficients in a standard Logistic Regression.

R Packages

To implement a Lasso, Ridge or Elastic Net penalized regression we will use the package:



Given a dataset named



How to implement a Lasso Regression

To, implement a Lasso Regression we first extract all the dependent variable values and all the independent variable values into two matrices using the following code:

```
y<-penalized_logistic_data$Y
x<- as.matrix(penalized_logistic_data[, 2:21])
```

How to implement a Lasso Regression

Now, we can use the following function to implement a Lasso regression to the data set

```
lasso_model <- glmnet(x, y, alpha = 1)

# alpha=1 means that we are choosing a Lasso Regression
```

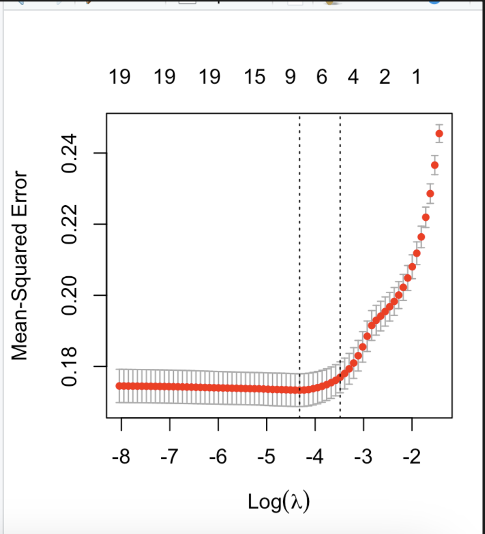
Minimizing process

Now, to find the minimum of the Likelihood function we implement the code:

```
cv_lasso <- cv.glmnet(x, y, alpha = 1)
# Creates all the lambdas to be minimized the cross-validation error by the regression

plot(cv_lasso)
```

Minimizing proccess

Which creates the plot



Picking the Optimum Lambda

From this plot we can use the following code to pick the optimum Lambda.

```
# Best lambda (minimizes cross-validation error)
best_lambda <- cv_lasso$lambda.min
best_lambda
```

Obtaining the coefficients of the optimal model

So, now that we found the optimal Lambda. We get the coefficients of the optimal model by the following code:

```
final_model <- glmnet(x, y, alpha = 1, lambda = best_lambda)

coef(final_model)  # Displays the coefficients for our model
```

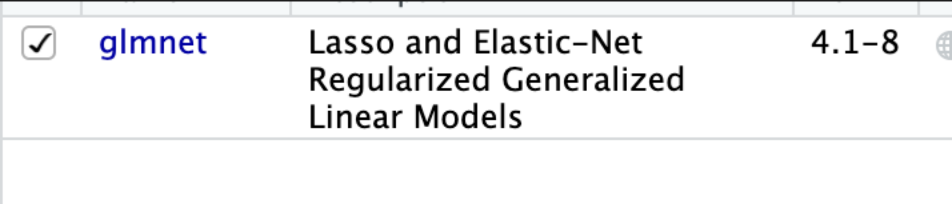
Coefficients

Which outputs the coefficients in the following table

```
> final_model <- glmnet(x, y, alpha = 1, lambda = best_lambda)
> coef(final_model)  # Displays the coefficients for our model
21 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  0.431033219
X1           0.215397752
X2          -0.138476691
X3           0.064573323
X4           0.053105612
X5          -0.006981416
X6           0.014067553
```

How to implement a Ridge Regression

To implement a ridge regression we will use the same package



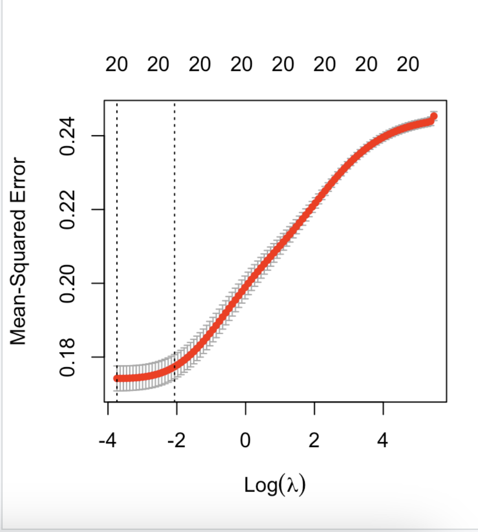
How to implement a Ridge Regression

The only difference in the implementation is changing alpha to zero:

```r
y<-penalized_logistic_data$Y
x<- as.matrix(penalized_logistic_data[, 2:21])

lasso_model <- glmnet(penalized_logistic_data, y, alpha = 0)

# alpha=0 means that we are choosing a Ridge Regression
cv_lasso <- cv.glmnet(x, y, alpha = 0)
# Creates all the lambdas to be minimized the cross-validation error by the regression
plot(cv_lasso)
# Best lambda (minimizes cross-validation error)
best_lambda <- cv_lasso$lambda.min
best_lambda
final_model <- glmnet(x, y, alpha = 0, lambda = best_lambda)
coef(final_model)  # Displays the coefficients for our model
```

Results of the implementation of the Ridge Regression

From the code we obtain the plot



Results of the implementation of the Ridge Regression

From the code we obtain the coefficients below. Note that none of these coefficients are zero.

```
> best_lambda <- cv_lasso$lambda.min
> best_lambda
[1] 0.02377139
> final_model <- glmnet(x, y, alpha = 0, lambda = best_lambda)
> coef(final_model)  # Displays the coefficients for our model
21 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept)  0.4312159068
X1           0.2060237127
X2          -0.1520492334
X3           0.0720542662
X4           0.0571590863
X5          -0.0370763266
X6           0.0246069066
```

Results of the implementation of the Ridge Regression

This code outputs the coefficients of our model in the form below. Note that none of these coefficients are zero.

```
> best_lambda <- cv_lasso$lambda.min
> best_lambda
[1] 0.02377139
> final_model <- glmnet(x, y, alpha = 0, lambda = best_lambda)
> coef(final_model)  # Displays the coefficients for our model
21 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  0.4312159068
X1           0.2060237127
X2          -0.1520492334
X3           0.0720542662
X4           0.0571590863
X5          -0.0370763266
X6           0.0246069066
```

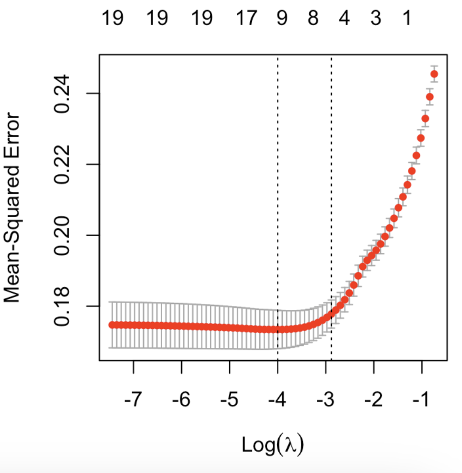
Implementation of the Elastic Net Regression

The only difference for the implementation for a Elastic Net Regression is changing the alpha to .5

```
# Elastic Net Regression

y<-penalized_logistic_data$Y
x<- as.matrix(penalized_logistic_data[, 2:21])

lasso_model <- glmnet(penalized_logistic_data, y, alpha = .5)

# alpha=.5 means that we are choosing a Elastic Net Regression
cv_lasso <- cv.glmnet(x, y, alpha = .5)
# Creates all the lambdas to be minimized the cross-validation error by the regression
plot(cv_lasso)
# Best lambda (minimizes cross-validation error)
best_lambda <- cv_lasso$lambda.min
best_lambda
final_model <- glmnet(x, y, alpha = .5, lambda = best_lambda)
coef(final_model)  # Displays the coefficients for our model
```

Results of the implementation of the Elastic Net Regression

This code outputs the plot:



Results of the implementation of the Elastic Net Regression

This code outputs the coefficients of our model in this form. Note that the coefficients may or may not be zero.

```
> # Best lambda (minimizes cross-validation error)
> best_lambda <- cv_lasso$lambda.min
> best_lambda
[1] 0.01832075
> final_model <- glmnet(x, y, alpha = .5, lambda = best_lambda)
> coef(final_model)  # Displays the coefficients for our model
21 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept)  0.4311264839
X1           0.2089072523
X2          -0.1352850284
X3           0.0644061103
X4           0.0529878254
X5          -0.0089211054
X6           0.0151415000
X7           .
```