**R: Ridge and Lasso Regression**

*Goal*: Perform regularized regression using LASSO and Ridge and select the best

Lambda using k-fold Cross Validation.

*Data*: Credit.csv available under Week 4 in MyCourses.

**Table of Contents**

Part A: Setup ----------------------------- 1

Part B: Ridge Model -------------------- 2

* K-fold cross-validation -------- 3

Part C: Lasso Model -------------------- 4

* K-fold cross-validation -------- 5

Result Summary ------------------------- 7

**Part 1: Setup**

First, we must load all necessary packages. *Don’t forget to uncomment the first few lines to install the packages* if you are running the code for the first time! These libraries include:

1. **glmnet**: Implements Lasso and ElasticNet generalized linear models.
2. **plotmo**: Implements functions for plotting model residuals and other diagnostics for linear models.

#install.packages("glmnet")

library(glmnet)

#install.packages('plotmo')

library(plotmo)

Next, we load and clean the credit score data included in the assignment to prepare for input into the linear model. This involves separating the input (independent variables) from the output (dependent variable) – in this case, we want to predict **Balance**, so we assign this as output y.

# Read in data via a path

credit<- read.csv("/Users/hfang/Downloads/CIS490\_2020Spring/Credit.csv", sep = ',', header = TRUE)

credit <- credit[, 2:12] # Remove first column (unneeded)

# Create input matrix of data with text variables changed to numeric, e.g. Gender Male/Female changed to 1/0

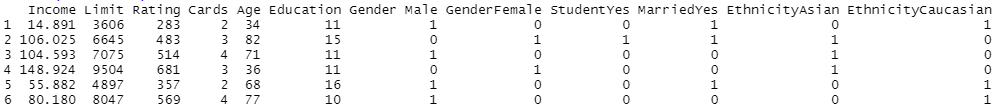
credit.mat <- model.matrix(Balance ~ .-1, data=credit)

# Delete unnecessary info at the bottom

credit.mat <- credit.mat[,-8]

head(credit.mat)

##



##

set.seed(1) # Set seed for reproducibility

# Separate the features (independent) from the target (dependent) variables

x <- credit.mat

y <- credit[, 'Balance']

**Part B: Ridge Model and K-fold CV for choosing lambda**

Creating a ridge regression is as simple as calling the glmnet function. Evaluating its performance, however, requires using the cv.glmnet function to perform 10-fold cross-validation, as well as optimizing the **lambda** hyperparameter. This process is as follows:

1. Create a list of possible lambda values at which to evaluate the ridge model
2. Run the ridge model at each lambda values
3. Plot the coefficient results at different values of lambda
4. Output numerical cross-validated results
5. choose λ∗ as the minimizer of CV error: Plot MSE performance at each lambda and calculate at which lambda (ie. finding the optimal lambda, λ∗ ) the MSE has the minimum value, or the error is within 1 standard error of the minimum MSE.
6. Use the optimal λ∗ to refit the model on the entire data and evaluate its predictive performance using MSE.

# Create a grid of possible values of lambda to test

grid <- 10^seq(6, -3, length=10)

# Construct linear model with the following parameters:

# scale function centers mean of each column around the same value

# alpha = 0 specifies ridge regression

# lambda = grid specifies to run the model for each lambda value in grid

# thresh = 1e-2 defines the stopping criteria for optimization

# standardize = TRUE converts X into same range (mean = 0, sd = 1)

ridge.mod <- glmnet(scale(x), y, alpha=0, lambda=grid, thresh=1e-2, standardize = TRUE)

# Plot results over different values of lambda

# If you modify the function so that label = 2, what happens?

plot\_glmnet(ridge.mod, xvar = "lambda", label = 4)

Chart, line chart

Description automatically generated

# Perform cross-validation

cv.out <- cv.glmnet(scale(x), y, alpha=0, nfolds = 10)

cv.out

# Results below yield lambda with minimum MSE as well as

# Lambda.1se is the largest value of lambda such that error is within 1 standard error of the minimum mse

# In this case (ridge regression) they are the same

##

Call: cv.glmnet(x = scale(x), y = y, nfolds = 10, alpha = 0)

Measure: Mean-Squared Error

Lambda Measure SE Nonzero

min 39.66 14054 516.6 11

1se 39.66 14054 516.6 11

##

# Plot the MSE for each lambda value

plot(cv.out)

Chart, histogram

Description automatically generated

# Determine best lamda

best.lambda <- cv.out$lambda.1se

best.lambda

[1] 39.65627

# Run ridge regression at the best lambda and evaluate using MSE

ridge.final <- glmnet(scale(x), y, alpha=0, lambda=best.lambda, thresh=1e-2, standardize = TRUE)

predict(ridge.final, type="coefficients", s=best.lambda)

12 x 1 sparse Matrix of class "dgCMatrix"

s1

(Intercept) 520.015000

Income -179.068432

Limit 386.880532

Rating 142.012771

Cards 26.339482

Age -17.675691

Education -1.690490

Gender Male 2.423412

StudentYes 115.165538

MarriedYes -5.664827

EthnicityAsian 5.948937

EthnicityCaucasian 5.011724

Hence, the best ridge model is written as the following linear equation:

*Balance = 502.02 – 179.07\*Income + 386.88\*Limit + 142.01\*Rating + 26.34\*Cards - 17.68\*Age - 1.69\*Education + 2.42\*Gender – 115.17\*Student - 5.66\*Married + 5.95\*Asian + 5.01\*Caucasian*

# Evaluate MSE and RMSE of the ridge model with the optimal lambda

ridge.pred <- predict(ridge.final, s=best.lambda, newx=scale(x))

print(paste('MSE:', mean((ridge.pred - y)^2)))

[1] "MSE: 13026.371868871"

print(paste('RMSE:', sqrt(mean((ridge.pred - y)^2))))

[1] "RMSE: 114.133132213529"

**Part C: Lasso Model and K-fold CV for choosing lambda**

Creating a lasso regression is virtually the same process – the only difference is that we specify alpha=1 instead of alpha=0 as in ridge regression to change the model to lasso. See part B for more details on this process.

Note that in the lasso model, when we plot the coefficients and MSE, some variables are reduced to 0; the number of non-zero coefficients at each lambda is denoted by the top axis “Degrees of Freedom.” Lambda.1se is the largest value of lambda such that error is within 1 standard error of the minimum mse but with a smaller number of attributes/predictors (degrees of freedom).

# Create a grid of possible values of lambda to test

grid <- 10^seq(6, -3, length=10)

# Create linear model with the following parameters:

# scale function centers mean of each column around the same value

# alpha = 1 specifies lasso regression

# lambda = grid specifies to run the model for each lambda value in grid

# thresh = 1e-2 defines the stopping criteria for optimization

# standardize = TRUE converts X into same range (mean = 0, sd = 1)

lasso.mod <- glmnet(scale(x), y, alpha=1, lambda=grid, thresh=1e-2, standardize = TRUE)

# Plot coefficient values at different lambda

plot\_glmnet(lasso.mod, xvar="lambda", label = 4)

Chart, line chart

Description automatically generated

# Perform cross-validation to determine best lambda value

lasso.cv.out <- cv.glmnet(scale(x), y, alpha=1, nfolds = 10)

lasso.cv.out

# Results below yield lambda with minimum MSE as well as

# largest value of lambda such that error is within 1 standard error of the minimum mse

##

Call: cv.glmnet(x = scale(x), y = y, nfolds = 10, alpha = 1)

Measure: Mean-Squared Error

Lambda Measure SE Nonzero

min 0.589 10045 535.4 11

1se 6.027 10506 457.2 6##

# Plot the MSE for each value of lambda

plot(lasso.cv.out)

Chart

Description automatically generated

# Create model using value of best lambda with a small number of variables and print output

lasso.best.lambda <- lasso.cv.out$lambda.1se

lasso.best.lambda

lasso.final <- glmnet(scale(x), y, alpha=1, lambda= lasso.best.lambda, thresh=1e-2, standardize = TRUE)

predict(lasso.final, type="coefficients", s=lasso.best.lambda )

##

12 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) 520.015000

Income -201.186637

Limit 547.774004

Rating 4.306531

Cards 27.936428

Age -11.171937

Education .

Gender Male .

StudentYes 120.701548

MarriedYes .

EthnicityAsian .

EthnicityCaucasian .

##

Hence, the best lasso model is written as the following linear equation with only 6 predictors/attributes:

*Balance = 520.02 – 201.19 \*Income + 547.77 \*Limit +4.31\*rating + 27.94\*Cards-11.17\*Age+120.70\*Student*

# Calculate MSE and RMSE for best model

lasso.pred <- predict(lasso.final, s= lasso.best.lambda, newx=scale(x))

print(paste('MSE:', mean((lasso.pred - y)^2)))

[1] "MSE: 11701.0165977173"

print(paste('RMSE:', sqrt(mean((lasso.pred - y)^2))))

[1] "RMSE: 108.171237386458"

In conclusion: Lasso reduces variance and performs variable selection. Below, we compare results from Ridge and Lasso regression:

|  |  |  |
| --- | --- | --- |
|  | Ridge Regression (α = 0) | Lasso Regression (α = 1) |
| min λ | 39.656 | 0.589 |
| 1se λ (optimal lambda, **λ\***)) | **39.656** | **6.027** |
| MSE (**at λ\***) | 13026.37 | 11701.02 |
| RMSE (**at λ\*** ) | 114.13 | 108.17 |

Note: for your projects, just report optimal lambda, or one of the two measures, MSE or RMSE.

Adapted from:

Gareth James, Daniela Witten, Trevor Hastie and Rob Tibshirani (2017). ITSL: Data for an Introduction to Statistical Learning with Applications in R. R package version 1.2. [https://CRAN.R-project.org/package=ISLR](https://cran.r-project.org/package%3DISLR)