FIN 510 Big Data Analytics in Finance

Lab 12: Lasso Regression

Due on 10/09/2021

Predicting annual salaries

The file Hitters.csv contains salaries and career statistics concerning 322 major league baseball players from the 1986 and 1987 seasons. The salary data was originally form Sports Illustrated, and career statistics were obtained from the Baseball Encyclopedia Update. The dataset contains 19 predictors, and the response is the 1987 annual salary on opening day in thousands of dollars (Salary). The goal is to fit lasso regression models to predict baseball players annual salaries based on career statistics. The following table describes each of the predictors and the response.

DESCRIPTION OF VARIABLES FOR BASEBALL SALARY DATA EXAMPLE	
Salary	1987 annual salary on opening day in thousands of dollars
AtBat	Number of times at bat in 1986
Hits	Number of hits in 1986
HmRun	Number of home runs in 1986
Runs	Number of runs in 1986
RBI	Number of runs batted in in 1986
Walks	Number of walks in 1986
Years	Number of years in the major leagues
CAtBat	Number of times at bat during his career
CHits	Number of hits during his career
CHmRun	Number of home runs during his career
CRuns	Number of runs during his career
CRBI	Number of runs batted in during his career
CWalks	Number of walks during his career
League	A factor with levels A and N indicating player's league at the end of 1986
Division	A factor with levels E and W indicating player's division at the end of 1986

PutOuts	Number of put outs in 1986
Assists	Number of assists in 1986
Errors	Number of errors in 1986
NewLeague	A factor with levels A and N indicating player's league at the beginning of 1987

0) Load the package

Use library() to load glmnet.

1) Create a data frame

Load the data with read.csv(). Save the result in a data frame named df. Return the first six rows and column names using head() and names(), respectively.

Use dim(df) to return the total number of rows and columns. Return the number of missing values using sum() and is.na().

Hint: is.na() returns TRUE if a value is missing and FLASE, otherwise.

2) Remove rows that have missing values in any variables

Modify df in place by removing rows with any missing values using na.omit().

After updating df, use dim(df) to return the total number of rows and columns. Return the number of missing values using sum() and is.na(). Make sure that the number of missing values is now equal to zero.

3) Create a matrix of predictors and a vector of the response

Lasso regression uses library glmnet, which requires a matrix of predictors. Use model.matrix() to convert predictors to a matrix named x. Inside model.matrix(), specify Salary~. as the model formular and df as the data. Use [,-1] to exclude the intercept in the resulting matrix. Return the first six rows of x with head().

Notice: model.matrix() creates dummy variables for categorical or character variables. For example, it creates a dummy variable for each of the binary predictors: League, Division, and NewLeague.

Select Salary from df and save it in a vector named y.

4) Data partition

Partition the data into training (50%) and test (50%) sets. Use set.seed(1) to set the random seed and sample() to take a sample of row numbers of the training set. Save the row numbers of the training set as train.index.

Hint: dim(x)[1] returns the length of rows in the matrix, 0.5* dim(x)[1] specifies the number of rows to select for the training set, and c(1:dim(x)[1]) represents row numbers.

Subset matrix x by train.index to return the predictors in the training set. Use head() to list the first six rows.

Subset vector y by train.index to return the outcome in the training set. Use head() to list the first six values.

Use setdiff() to return the row numbers of the test set and save the result as test.index.

Subset matrix x by test.index to return the predictors in the test set. Use head() to list the first six rows.

Subset vector y by test.index to return the outcome in the test set, and save the result as y.test. Use head() to list the first six values.

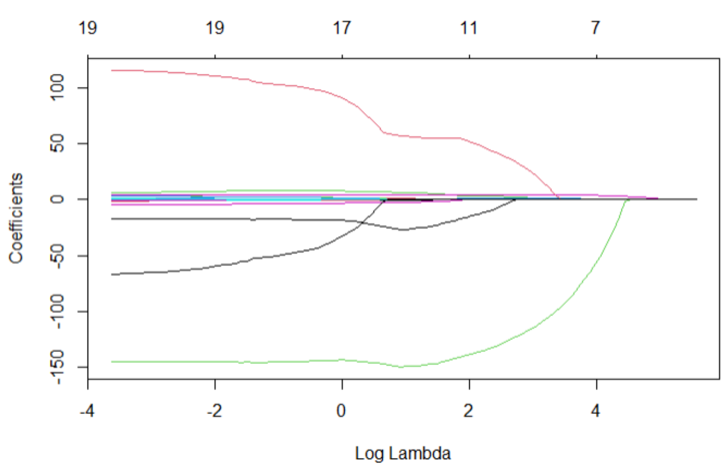
5) Fit a lasso regression model on the training set

To predict annual salaries of baseball players, fit a lasso regression with all predictors using glmnet(). The first parameter is a matrix of predictors in the training set and the second parameter is a vector of the outcome in the training set. Specify alpha=1 to fit a lasso regression. Save the model as fit.

Use fit$lambda to return an automatically selected range of lambda values. Save the 1st, 50th, and 100th elements in fit$lambda as lambda.large in question 8, lambda.medium in question 7, and lambda.small in question 6.

Use dim() to return the dimension of the coefficient estimates. It is a 20 by 100 matrix, which has 20 rows (one for each predictor, plus an intercept) and 100 columns (one of each value of lambda).

Visualize the change of coefficient estimates with respect to log of lambda values using plot() with xvar="lambda".



According to the fitted lasso regression model in question 5, answer questions 6, 7, and 8.

6) Lasso regression with a small lambda value

Save the 100th element in fit$lambda as lambda.small and return its value.

Use predict() with type="coefficients" to return the coefficient estimates of the lasso regression model where s=lambda.small. Select 20 coefficient estimates from the sparse matrix with [1:20,], and save them in a vector named coef.lambda.small. Return non-zero coefficient estimates in coef.lambda.small. Notice that none of the 20 coefficient estimates are exactly zero.

Use predict() to predict Salary for records in the test set based on the lasso regression with the smallest lambda. Save the predicted salaries in the test set as pred.lambda.small and return the first six values with head(). Evaluate the model performance by computing the mean squared error (MSE) in the test set.

Hint: use mean((y.test-pred.lambda.small)^2) to compute the mean squared error (MSE) in the test set.

7) Lasso regression with a medium-sized lambda value

Save the 50th element in fit$lambda as lambda.medium and return its value.

Use predict() with type="coefficients" to return the coefficient estimates of the lasso regression model where s=lambda.medium. Select 20 coefficient estimates from the sparse matrix with [1:20,], and save them in a vector named coef.lambda.medium. Return non-zero coefficient estimates in coef.lambda.medium. Notice that 6 of the 20 coefficient estimates are exactly zero.

Use predict() to predict Salary for records in the test set based on the lasso regression with the medium-sized lambda. Save the predicted salaries in the test set as pred.lambda.medium and return the first six values with head(). Evaluate the model performance by computing the mean squared error (MSE) in the test set.

8) Lasso regression with a large lambda value

Save the 1^{s} element in fit$lambda as lambda.large and return its value.

Use predict() with type="coefficients" to return the coefficient estimates of the lasso regression model where s=lambda.large. Select 20 coefficient estimates from the sparse matrix with [1:20,], and save them in a vector named coef.lambda.large. Return non-zero coefficient estimates in coef.lambda.large. Notice that 19 of the 20 coefficient estimates are exactly zero.

Use predict() to predict Salary for records in the test set based on the lasso regression with the largest lambda. Save the predicted salaries in the test set as pred.lambda.large and return the first six values with head(). Evaluate the model performance by computing the mean squared error (MSE) in the test set.

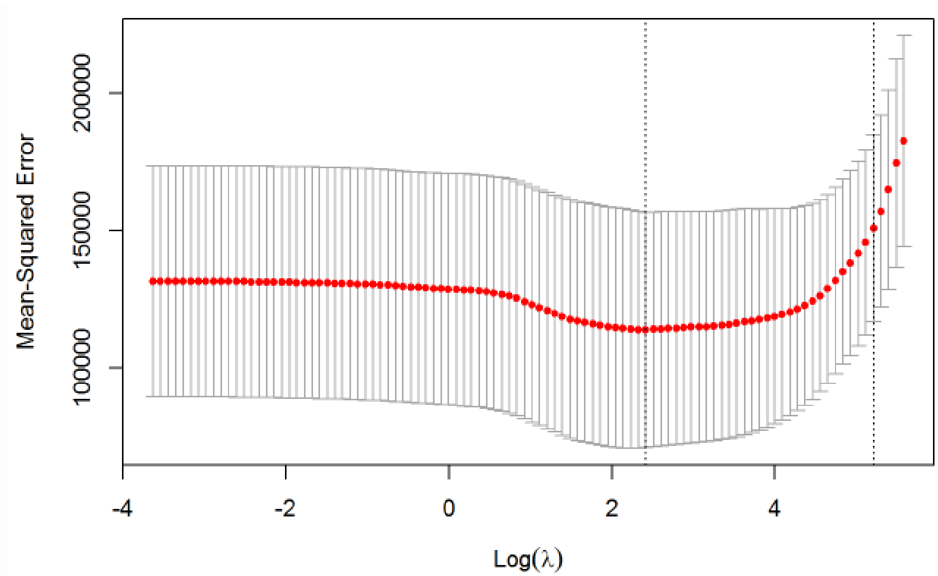
9) Using cross-validation to choose lambda

Use set.seed(1) to set the random seed.

To identify the lambda at which the lowest cross-validated MSE is achieved, use cv.glmnet() to perform a lasso regression model with 5-fold cross validation on the training set. The first parameter is a matrix of predictors in the training set and the second parameter is a vector of the outcome in the training set. Specify alpha=1 to fit a lasso regression. To specify the cross-validation criterion as the mean squared error, set type.measure to mse. Use nfold=5 to perform 5-fold cross-validation. Save the result as cv.fit.

Plot the cross-validated MSE for each lambda using plot().

Save the lambda value that corresponds to the lowest 5-fold cross-validated MSE as lambda.best using cv.fit$lambda.min, and return its value.



10) Lasso regression with the best lambda value

According to cv.fit, use predict() with type="coefficients" to return the coefficient estimates of the lasso regression model where s=lambda.best. Select 20 coefficient estimates from the sparse matrix with [1:20,], and save them in a vector named coef.lambda.best. Return non-zero coefficient estimates in coef.lambda.best. Notice that 8 of the 20 coefficient estimates are exactly zero.

Use predict() to predict Salary for records in the test set based on the lasso regression with the best lambda. Save the predicted salaries in the test set as pred.lambda.best and return the first six values with head(). Evaluate the model performance by computing the mean squared error (MSE) in the test set.

11) Compare lasso regression models

Compare the four lasso regression models in questions 6, 7, 8, and 10 and answer the following questions.

Which model has the lowest test MSE? Which model selects the greatest number of predictors? Which model selects the least number of predictors?