Abstract

Incorporating all the resources learned in Stat 159, we will dive deeper into data analytics by applying more advanced statistical learning methods to model a data set listing credit information of 400 randomly selected individuals. This project will reproduce the methods performed in Chapter 5 and 6 from the textbook, "An Introduction to Statistical Learning" by James et al. These sections cover cross-validation techniques and linear model selection/regularization methods, both of which are necessary in locating the optimal model for a given data set. The credit data set is sourced from the following url: `http://www-bcf.usc.edu/~gareth/ISL/Credit.csv`. Collaborating with a partner was mandatory and allowed us to become more comfortable with Github and Git processes.

Introduction

The motivation for this project is to perform a predictive modeling process to the Credit data set. The first step in almost all data analyses is to understand and search for unique characteristics in the data set we will be working with. Gaining insight into the relationships among the multiple variables is essential in subset selection and prior knowledge into which variables are insignificant. After completing the section covering exploratory data analysis, the Credit data set will be processed to be prepared for model fitting, which includes centering/standardizing each variable column and dealing with categorical variables. The processed Credit data set is then split into training and test sets and thereafter, five different regression models are fitted to select the best model in accurately predicting the response variable Balance.# Data

The Credit data set presents information on elevent different values of 400 bank clients. Of these eleven variables, seven of them are quantitative and the remaining four are qualitative. Below contains brief descriptions of each variable.

Quantitative

- Income: customer's income
- Limit: customer's credit limit
- Rating: customer's credit rating
- Cards: number of credit cards
- Age: customer's age
- Education: number of years in education
- Balance: current balance in the customer's bank

Qualitative

- Gender: customer's gender (factor with two levels - Male/Female)
- Student: customer's current student status (factor with 2 levels - Yes/No)
- Married: customer's current marital status (factor with 2 levels - Yes/No)
- Ethnicity: customer's ethnicity (factor with 3 levels - Asian/Caucasian/AfricanAmerican)

For this project, we will see which model we fit will most closely predict with the quantitative variable `Balance`. Thus, we can consider the Balance variable as a response and the other variables as predictors. Since there are 11 variables and 400 customers, the Credit data set has a 400 x 11 dimension. In the next section, in order to assess the accuracy of each regression fit, the Credit data set must be split into two, a training set and a test set.

Methods

Train and Test Sets

To split the Credit data set, the `test_split_script.R` was used. In the source code, 300 rows were randomly selected and stored into a csv file called `train_credit.csv`. The remaining 100 rows were written into a csv file called `test_credit.csv`. The regression models will be trained onto the 300-row data set and their fitted predictions will be measured by how much they deviate compared to the test set.

Pre-modeling Data Processing

Since the splitted Data sets were still raw, pre-processing techniques were done to properly prepare for modeling. The four categorical variables cannot be interpreted by the model functions in R, so it is better to convert them to numerical values. The categorical variables with 2 levels can be converted to a dummy variable while the variable with 3 levels such as Ethnicity needs two binary columns or two dummy variables. Using the function `model.matrix` does this automatically, so our data set(s) is converted to a 400 x 12 data frame.

In addition, the coefficients after fitting each model may be unexpectedly altered by the scaling of each variable. Thus, it is proper to standardize and center each variable columns so that comparisons among each variable are done fairly and that the beta coefficients would not blow up.

Regression Models

The following lists the 5 regressions that will be applied to the Credit data set:

- Ordinary Least Squares Regression
- Ridge Regression
- Lasso Regression
- Principal Components Regression
- Partial Least Squares Regression

We start with the ordinary least model to fit onto the Credit data set. After computing the test mean standard error, this statistic will serve as a basis to compare the accuracies of the other models. The OLS regression will be performed by the `ols_script.R`, using the `lm()` function to model Balance over the 10 predictors. From this model object, we then use the `predict()` function on the test predictors and calculate the mean squared error.

Another set of more complex models deals with shrinkage methods. These models, ridge regression and lasso regression, shrink the coeffcients to `0` by setting a constraint to the `beta` coefficients, having lambda as the tuning parameter. The difference in ridge and lasso is basically the formula inside of the sum of the constraint. The following provides a step-by-step instruction on how these methods work:

- Load the library called `glmnet` and read in the training and test sets
- Split the the training set by subsetting every columns except Balance into x and Balance into y
- Perform the regression using the function `cv.glmnet()`. A 10-fold cross validation technique is applied by setting `nfold = 10`. Since we already standardized and centered the variables, the argument `intercept` and `standardize` are set to `FALSE`. By default, `cv.glmnet()` incorporates all lambdas, but in this project, we will assume that ‘lambda = 10ˆseq(10, -2, length = 100). The alpha argument differentiates between the two models: ridge regression is performed when alpha = 0 and lasso regression is performed when alpha = 1.
- The model object after fitting one of these shrinkage regression models displays a model for each lambda. We use this object and retrieve the lowest lambda by `model_object$min.lambda`.

- A plot of the cross-validation errors can be done easily by using the `plot` function. The lowest lambda is saved onto the text file and the plot is saved as a png image.
- Use the model object and the `predict()` function to make predictions on the predictor variables of the test set. The mean squared error is then calculated by comparing the predictions and the actual response values. The MSEs are stored into the text file as well.
- Using the lowest lambda, the optimal model is refitted onto the whole data set `scaled_credit.csv` and the model's coefficients are then saved into the same text file.

The last two regression models deal with dimension reduction by locating which predictors explain the most variance of Balance. Knowing how many predictors or components to use depends on which set has the lowest predictors. Principal Components Regression and Partial Least Squares Regression are run by the `pcr_script.R` and `plsr_script.R`.

- Load the library called `pls` and read in the training and test sets
- Use the model fitting functions `pcr()` and `plsr()` to run the regressions, setting `formula = Balance ~.` and `validation = 'CV'`. The data should only cover the training set
- The best model in the cross-validations is found using `which.min(model_object$validation$PRESS)`. This value is stored into a text file
- Plot the model using the `plot()` function and save it as a png image
- Use the test set to predict the response Balance and compare it to the real Balance observation, thereafter calculating the test mean squared error of the model. The MSE is stored in a text file.
- Fit the model onto the `scaled_credit.csv` with `ncomp` equal to the minimum principal components since this model will lead to the lowest MSE. The coefficients of this official model is then saved into a text file.

These five regression methods are then compared by their MSEs, and the model with the lowest MSE is considered the best. Other model statistics can be used for comparisons, but for this project, the MSE is the most convenient.

Data

Data

Data