Data Analytics for Business

Homework 5 – Part 2

Neural Networks

In this homework exercise, we will apply neural networks for prediction. We will use the S&P 500 stock index data to see if past stock performance can be used to predict future stock performance. The data set consists of daily percentage returns for the S&P 500 stock index between 2001 and 2005, 5 years with 1,250 trading days in total. For each trading day, we have the previous five days' percentage returns, the current day's trading volume, and the current day's percentage return. We create a binary variable to indicate whether the stock market goes up or down in each trading day. We would like to use these available data to predict the direction of stock market performance every day. We are interested to see if the neural networks, a potentially more powerful tool for prediction, can help us do any better than traditional models such as logistic regression.

Download the data file "**Smarket.csv**," which contains the following fields. Complete the tasks outlined below.

Data Description

Column Name	Variable Description
Year	The year that the observation was recorded
Lag1	Percentage return for previous day
Lag2	Percentage return for 2 days previous
Lag3	Percentage return for 3 days previous
Lag4	Percentage return for 4 days previous
Lag5	Percentage return for 5 days previous
Volume	Volume of shares traded (number of daily shares traded in billions)
Today	Percentage return for today
Up	Whether or not the market had a positive return in the current day

Task 1: Data Preparation

1. Install and load the **"neuralnet"** package that will be needed. Import the data into R.

2. Note that neural networks perform the best when variables are scaled. To **scale** the data, **remove Columns 1 and 8** (which are "Year" and "Today") because we won't incorporate them into the analysis. Also **exclude the "Up" column when scaling** because it is binary and meant to enter the analysis as is. Use the **scale()** function to scale the data.

3. Separate the scaled data into a training set and a testing set. Use **80%** of all records as the training set (i.e., 1,000 records) and leave the rest **20%** as the testing set (i.e., 250 records). Do random sampling in creating the training/testing sets. As usual, to ensure replicability, **set the seed to 1000**.

Task 2: Single-Layer Neural Network

4. Let us first fit a simple neural network to the training data set, with only **1 layer** and **2 activation nodes**. Use **Up** as the output variable, and only **Lag1** and **Lag2** as the input variables. Use the **neuralnet()** function in the **"neuralnet"** package. Set the **linear.output** argument to **FALSE**, meaning that we are applying the same logistic activation function to the output node as well.

   Note: Because the estimation of neural networks involves randomness, the fitted neural network will be different each time it's run. **To ensure uniformity of the answers, please use the neural network I have pre-fitted for you and provided on Canvas**. Use the following statement to load the pre-fitted neural network and **complete the rest of Task 2 based on this provided neural network**.

   load("Smarket_nn1.Rda")

5. Plot the neural network using the **plot()** function. Be sure to understand what all the information on the plot means.

6. Retrieve and display the **weights** of the neural network. Make sense of these values. You will need to use these weights in the calculations next.

7. Take the first record in the testing data set. You should have the following values:

	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Up
4	-0.5516457	0.9047775	0.8406793	0.3331338	-0.1722013	-0.5614015	TRUE

Manually calculate the values of all activation nodes and the output node for this record. Use the formula we introduced in class with the **logistic** (a.k.a. **Sigmoid**) activation function. Be sure to include the bias terms (a.k.a. intercepts). You need to use the weights retrieved in Step 6.

8. Verify the results of your manual calculation by calling the **compute()** function to calculate the values of all nodes. Your calculation should be the same as the values of the **neurons** and the **net.result** displayed in the **compute()** function output.

Task 3: Multi-Layer Neural Network

9. Now let us fit a more complex neural network to the training data set, with **2 layers**, **4 activation nodes** for the first layer, and **2 activation nodes** for the second layer. Now include all available variables as the predictors: **Lag1** through **Lag5**, and **Volume.** Same as Step 4, use the **neuralnet()** function, and set the **linear.output** argument to **FALSE**.

   Note: Again, because the estimation of neural networks involves randomness, the fitted neural network will be different each time it's run. **To ensure uniformity of the answers, please use the neural network I have pre-fitted for you and provided on Canvas**. Use the following statement to load the pre-fitted neural network and **complete the rest of Task 3 based on this provided neural network**.

   load("Smarket_nn2.Rda")

10. Plot the neural network using the **plot()** function. Retrieve and display the **weights** of the neural network.

11. Use the given neural network to predict the probabilities of "Up" for all the trading days in the testing set. Use the **compute()** function.

12. Based on the predicted probabilities of "**Up**" in the previous step, determine the predicted market performance (i.e., whether "**Up**" is TRUE or FALSE) for each trading day in the testing set. Use **0.5** as the cutoff.

13. Compare the predicted outcomes with the actual outcomes by constructing the **confusion matrix**. Calculate the out-of-sample **prediction accuracy** of this multi-layer neural network.

14. As a benchmark for performance comparison, fit a usual **logistic regression** model to the training data set.

    - As we did previously, use the **glm()** function, and include all available predictors.
    - Predict the probabilities of "Up" for all the trading days in the testing set. Use the **predict()** function, and set **type="response"**.
    - Determine the predicted market performance (i.e., whether "**Up**" is TRUE or FALSE) for each trading day in the testing set by using **0.5** as the cutoff for the predicted probabilities.
    - Construct the **confusion matrix**, and calculate the out-of-sample **prediction accuracy** of the logistic regression model.

15. Comparing the prediction accuracy in Steps 13 and 14, how would you comment on the performance of a multi-layer neural network and the accuracy of predicting future stock market movement based on past performance?