Neural Net Assignment

(Due Tue Sun 11/29)

ISOM 674, Fall 2020

In this assignment we are going to explore the behavior of stepwise logistic regression, random forests, and neural nets using the spam dataset. We will use the original data (in the file "spambasedata-Orig.csv"). In this file, all of the word and character frequency variables for each email are coded as percents (of the words or characters, respectively) and the run-length variables are coded as counts. Also, we are going to be using all 57 features in the data set. The documentation for the data set can be found at the follow link: http://archive.ics.uci.edu/ml/datasets/Spambase.

To get you started, I have provided a "helper" R script (Assignment-Template-Due-2020-11-29.r) that:

- Reads in the data
- Splits the data into a training, validation samples, and test samples.

Questions

1. Fit a stepwise logistic regression to the spam data using all 57 features. To get you started, in addition to reading in the data and creating the training and validation samples, the R helper script also computes the "small" and "big" formulas you will need to run the stepwise regression and runs the glm() function to estimate the logistic regression for the "small" model.
    - You can use the code in the file "StepwiseDemo.r" from class to remind yourself exactly how stepwise regression works.
    - Use direction = "both" so that the procedure will consider both adding and removing variable at each step.
    - Don't worry about any warnings from the stepwise regression telling you that some predictions are numerically 0 or 1.
    - Note: The stepwise regression may take some time to run, so be patient.
    - The predict() function works with the output of the stepwise regression function, so you can make predictions in a manner quite similar to what you have done before. For the output from the gml() function, the prediction will be the log odds rather than the prediction probability. To get predict() to give you the predicted probability, add the argument type="response" to the arguments of the predict function. The predict function for objects returned from gml() actually calls the function predict.glm(), so you can look at the help for that function for more details if you need to.
    - Make predictions for both the validation data and the test data.
    - I have provided an R script ROCPlot.r which defines a function ROCPlot() that you can use to make the requested ROC plots and to compute the AUC.

Answer the questions on the Google Form and submit the ROC plot.

2. Fit a random forest to the data. Note that the file I used in class (BaggingRandomForests.r) basically provides a template. We need to make a few changes, however:
   - In order to do classification, randomForest() needs the Y-variable to be of the factor data type. Again, this was taken care of early in the template file.
   - Just as in the stepwise logistic regression, use all 57 features in the model specification. (Also as in the logistic regression, the ultimate model may not use all 57, but we want there to be an opportunity for each of the 57 features to be used.)
   - Use 1000 trees and the default values for mtry and nodesize.
   - Add the argument type="prob" to the arguments of the predict() function when you make the predictions for the validation data set. This causes predict() to return the probabilities of each class. Thus, a matrix is returned.

     Note: When used on the output from randomForest(), the predict function actually called is predict.randomForest(). Thus, you can find the documentation by using the help for predict.randomForest.

   Answer the questions on the Google form and submit the ROC plot

3. Next explore fitting a couple of different neural nets to the spam dataset. To answer this question, you can refer to the SpiralNN.py example from class. Following the approach I used for the spiral data in class, the R template file I have provided will write out the data as csv files so that it can be read into the python program. I have provided a python template file (SpamNNTemplate.py).

   We are going to explore the behavior of both a wide but shallow neural net and also a narrow but deep neural net. To this end, suppose we have a "budget" of 20 nodes (neurons). The first model is going to have one layer of 20 neurons. In addition, to cause the network to do binary classification, we will have one additional layer of one neuron using a "sigmoid" activation function. Also, classification requires a loss function that is appropriate for classification, so use binary cross-entropy.

   In addition to the above, use the following settings.

   - Allow a bias term for all nodes.
   - Use the relu activation function for all nodes except for the last node in the output layer of one neural which should be sigmoid as indicated above.
   - Limit the number of epochs to 10000.
   - Use a batchsize of 250.
   - Use RMSprop as the optimizer with a learning rate of 0.001.
   - In the neural net fitting procedure, use an early stopping rule based on the validation data loss. Use a patience setting of 100.

    - Make probability predictions for both the validation and test data.

   Once you have fit the neural net model, read the data back into R and use the ROCPlot function to make the ROC plot and calculate the AUC. Answer the questions on the Google Form.

4. Repeat question 3, but use a narrow but deep neural net. Specifically, use 5 layers of 4 nodes instead of the one layer with 20 nodes. Answer the questions on the Google Form and submit the ROC plot.

Instructions for Submitting the Plots for Question 1 to 4

In addition to answering the questions on the Google form, you need to turn in the plots requested above. Note that there are 4 plots requested. Turn these plots in as labeled figures in a pdf file named:

EmoryNetID.pdf

Please substitute your Emory Net ID for “EmoryNetID” in the file name above.

As I have indicated before, you can make the required pdf file by first making a Word document containing the figures. An easy way to get a plot into Word from RStudio is to pop the plot out of the RStudio IDE into its own window by using the Zoom button. The window can then be copied using Control-Alt-PrtScr and pasted into the Word document with Control-V.