ISOM 674

Machine Learning I

EXAM Part 2
Long Answer Questions

Fall 2020

Instructions:

The rules with respect to resources also apply to this part of the exam. To reiterate, this exam is open book, open notes, and open computer and open internet. That is, you may access the internet as a resource for documentation, to look at responses to questions on technical websites, and so on. You may not have any synchronous communication with another person during the exam or post questions to forums in hopes of getting a fast response, or any other behavior contrary to the idea that the exam is your own individual work.

There are a total of 48 points on this part of the exam.

I reserve the right to adjust the point values in response to analysis of how students have responded to the questions as it is difficult or impossible to anticipate all of the issues that might occasionally come up.

When you have complete the exam, upload your R script file using the link provided.

Question 1 – Best Tree for the Spam E-Mail Data

You will recall that in the last assignment (the Neural Net assignment), we worked with the spam e-mail data. Specifically, we fit a stepwise logistic regression, a random forest, and two neural net models to the original spam data (which has the feature frequencies coded as continuous variables). We used AUC on the validation data to evaluate the model performance in order to select the best model among those we examined.To keep that assignment to a reasonable length, I skipped having you fit basic tree models to the data set and asking you to find the best tree.

This question asks you to fill in that gap. So in this problem I would like you to find the best tree model for the spam data. Instead of using AUC as the performance criterion, I would like you to use the log likelihood as the performance criteria.

As usual, I have provided an R template file that reads in the data and created the training, validation, and test samples.

(1a) Write an R function to compute the log likelihood. The function should have the form

```
loglk(Phat,Y)
```

where Phat is the vector of probabilities predicted by the model and Y is the vector of 0's and 1's indicating whether or not the e-mail is spam. The function should return the log likelihood and you will use this function in evaluating the performance.

Note 1: Because the predicted probabilities could be 0 or 1 exactly, I have included a couple of lines of code that rescale the Phat so that it is slightly bounded away from 0 and 1.

NOTE: If you are unable to write this function, you can proceed with solving the problem using the AUC (as we did in the Neural Net Assignment) as a substitute, but you will lose points for this part of the question. If you do this, then substitute AUC for log likelihood as the criterion in the questions below.

(1b) Compute the large tree that basically fits the data down to the individual observation. Use the following parameters values: minsize=2, mincut=1, mindev=0. Also use "deviance" as the splitting criteria as this means that the splits are performed essentially using the log likelihood.

Hint 1: Regression trees were introduced in Class 6 (9-17-2020) in the file IntroToRegressionTrees.r. You may want to refer to this file and use the code as a template.

Hint 2: In order to get the tree() function to perform classification, the Y variable must be a factors. This was taken care of for you in the template file as IsSpam was converted to a factor.

Hint 3: Recall that the formula IsSpam ~ . sets IsSpam as the Y variable and then uses all of the other variable in the data frame as the x-variables (features). This notation means that you do not have to list all of the x-variables out in the formula used in the tree function.

(1c) We are going to vary the complexity of the tree model by constraining the number of end nodes of the tree. Recall that we do this by pruning back the tree from the “big tree” model as shown in the IntroToRegressionTrees.r example. Specifically, using the parameter “best” in the R function prune.tree() find the best tree model of specified size.

Specifically, find the best tree models for sizes ranging from 2 to 20 and evaluate the performance on the validation sample using the log likelihood function.

Hint: You will need to make the predict() function return the predicted probabilities. Do this using the type=”vector” arguments. Documentation can be found by using the R help with the predict.tree search term.

(1d) Make a plot of the log likelihood (Y-axis) against the size of the tree.

(1e) What is the size of the best tree? What is the value of the log likelihood for this model?

(1f) Make an ROC plot for your best model and report the AUC.

(1g) Evaluate the log likelihood for the test data

(1h) Make an ROC plot for the test data and report the AUC.

Be sure to label your answers to each of these clearly in your R code file.

Question 2 -Face Recognition Neural Net

This question may be a bit of a stretch question, but basically I want you to make as much progress on it as you can in the time remaining after you answer Question 1.

In the face recognition project we have worked on in class (and which I summarized during the last class), I used random forests to create the model used to assess whether images of two faces were the same person or not.

The goal of this question is to use a neural net as the model instead of random forests.

As a reminder, the data used to fit the model was based on pairs of images. The dimension of the images had been reduced to the first 300 principle components. The Y variable was whether or not the two images were of the same person. The features were the absolute difference of the 300 principle component values for the two images.

I have created a template file (Exam Part2 Q2 Template.r) that reads in the training and validation data from the binary R file SamePersonMatchingDataFiles.RData. It them write out the csv files that can be read into a python file to run Keras to fit the neural net.

In the template file, I only write out the first 100 or the 300 features in order to make things run a little faster. This is very suitable for debugging. Once everything works, it would be easy to then user all 300 features (beyond the scope of this question).

The template file also reads back into R the output from the python neural net program.

Note: What must be done here closely parallels the neural net assignment. As a template for answering this question, I suggest that you use one of the python program used for that assignment and modify it to work in this case.

Finally, when you are debugging the program, I would set the number of epochs run to some small number such as 20 or 50 so the program runs quickly.

(2a) Using Keras in Python (as in the Neural Net Assignment), fit a neural net model to the face recognition data. As a start and for fairly quick debugging runs, use a model that has 2 layers with 5 Relu neurons each. (of course, you will also need a 1 neural layer for the sigmoid function). Use the same approach are as was used in the Neural Net assignment – that is, implement a stopping rule based on the validation performance using binary cross-entropy.

(2b) Read the output from your neural net model back into R and make an ROC plot and compute the AUC.

For these questions, upload you R file and your Python file. In the R template file, change NetID to your Emory network ID. Name your python file "Exam Part2 Q2 – NetID.py" and also change NetID to your Emory Network ID.