STA 712 Challenge Assignment 3: Neural Networks and Logistic Regression

Due: Friday, September 30, 12:00pm (noon) on Canvas

Instructions:

- Submit your work as a single PDF. All work should be typed, with math and equations typeset using LaTeX or similar software. Include all R code needed to reproduce your results in your submission.
- You are welcome to work with others on this assignment, but you must submit your own work.
- The goal of this assignment is for you to learn about a topic beyond the core material covered in class. This may require more work than a normal homework assignment, so I recommend starting early. If you get stuck, I am happy to chat over email or in office hours.
- You can probably find the answers to many of these questions online. It is ok to use online resources! But make sure to show all your work in your final submission.

Introduction

Logistic regression is one of the most widely used tools for predicting a binary response. However, logistic regression is limited by the assumption that the log odds are a *linear* function of the predictors. What if we want to model nonlinear relationships? Transformations are one option, but it can be hard to choose the right transformations with high-dimensional data.

Another option is *neural networks*, a popular prediction/classification method which can be used to fit more complex relationships. It also turns out that logistic regression can be represented as a special case of a *feedforward* neural network (the simplest type), so neural networks provide a nice generalization of logistic regression.

The goal of this assignment is to introduce you to neural networks, show how logistic regression can be viewed as a simple network, and experiment with fitting more complicated models.

Background reading:

This assignment requires you to learn quite a bit of new information that isn't covered in class. Here I provide some references to get started, but you may need to do some research beyond these references.

- To get started and see an overview of neural networks, skim Chapter 1 of *Neural Networks and Deep Learning*, a free online book by Michael Nielsen available here: `http://neuralnetworksanddeeplearning.com/index.html`
- I also recommend the 3Blue1Brown YouTube videos on neural networks: `https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi`
- You may also find these slides from a CMU course on machine learning helpful, in particular lecture 11 and lecture 14: `http://www.cs.cmu.edu/~mgormley/courses/10601-s18/slides/`

Fitting a neural network:

There are many options for fitting a neural network. If you have never fit a network before, I suggest using one of the following:

- The `neuralnet` package in R (`https://cran.r-project.org/web/packages/neuralnet/`)
- TensorFlow for R, using Keras (`https://tensorflow.rstudio.com/`)
- PyTorch, in Python (`https://pytorch.org/tutorials/`)

Data

In this challenge activity, you will work with data from a website called ScienceForums.Net (SFN), which has been open since 2002 and hosts conversations on a range of topics from biological and physical science to religion and philosophy. Each row in the data represents one 'thread', which is comprised of a series of posts stemming from an initial post. For each thread, we have some information that SFN collects such as the number of views and the number of authors. The threads present in the data are a random sample of threads from 2002-2014, with the data collected in 2014. SFN moderators are interested in using this data to determine which threads warrant the most attention.

You can load the SFN data into R by

```
sfn <- read.csv("https://sta712-f22.github.io/class_activities/sfn.csv")
```

The `sfn` dataset contains the following columns:

- *Age*: the age of the thread (in days) when the data was collected in 2014, measured from the first post in the thread
- *State*: sometimes moderators close threads if they are inappropriate. closed indicates the thread has been closed, otherwise State is open
- *Posts*: the number of posts in the thread
- *Views*: the total number of views of the thread
- *Duration*: the number of days between the first and last posts in the thread
- *Authors*: the number of distinct authors posting in the thread
- *AuthorExperience*: the number of days the author of the first post in the thread had been registered on SFN when the thread began (0 indicates they registered that day)
- *DeletedPosts*: the number of posts in the thread that have been deleted by a moderator
- *Forum*: the forum in which the thread was posted (e.g., Science)
- *AuthorBanned*: whether the original author of the thread is currently banned from posting on SFN (at the time of data collection, not when the thread was first posted)

Questions

Moderators are interested in predicting whether a thread will have any posts which need to be deleted.

1. Create a new variable, *HasDeleted*, which is $= 1$ when the number of deleted posts is > 0, and 0 otherwise.

2. Randomly split the data into a *training* set, which contains 60% of the observations, and a *test* set which contains the remaining 40% of the observations. (We will fit our models on the training set, then evaluate their performance on the test set. This helps to avoid issues with overfitting).

3. Fit a logistic regression model on the *training set* to predict whether any posts have been deleted, using the number of views, the number of posts, and the number of authors as your explanatory variables. Report the equation of the fitted model.

4. Assess the performance of your fitted logistic regression model with an ROC curve. Calculate the ROC curve on the *test set*, show a plot of the curve, and report the area under the curve (AUC).

5. Draw a network diagram (that is, draw the nodes in each layer of the network, and the connections between the nodes), showing how the logistic regression model from 3. can be represented as a feedforward neural network. Specify the input layer, output layer, the weights, any activation functions, and the loss function used in training.

6. Fit your neural network from 5. (on the training data), and report the fitted weights. These fitted weights should be similar to the estimated coefficients from 3.

7. Now add a hidden layer to your network, and fit the new model on the training data. Does the performance of your model improve when you add the hidden layer?

8. In class, we discussed gradient ascent as an alternative to Fisher scoring. Neural networks are usually trained with a variation of gradient ascent called *stochastic gradient descent* (in this case it is *descent* rather than *ascent*, because we are minimizing a loss function rather than maximizing a likelihood). Explain the difference between stochastic gradient descent and gradient descent, and why *stochastic* gradient descent is useful for fitting complicated models.