Eric Wu, Jacob O'Leary, Caden Summers, Matthew Brady

STAT 380

Professor Yin Tang

4/11/2023

Part One Report of Single Layer Network on Hitters Data

Introduction

The Single-layer neural network is a type of network that consists of one single hidden layer between the input and the output layer. We picked this example because we believe single-layer networks served as a foundation for exploring much more complex neural networks in deep learning. On the other hand, according to Kaggle.com, the hitter's data set is a major league baseball player data set that contains a variety of information about each player regarding their baseball careers, and their performance in 1986. The data set included 322 observations with 20 attributes, some of which are:

1. AtBat - Number of times at bat in 1986.
2. Hits - Number of hits in 1986.
3. HmRun - Number of home runs in 1986.
4. Run - Number of runs in 1986.
5. RBI - Number of runs batted in 1986.
6. Walks - Number of Walks in 1986
7. Anything that starts with a "C" represents career statistics.
8. Etc.

The Single layer neural network is the simplest variation of a neural network. The network takes in the predictor variables as *P* number units, that form what is known as the input layers, then each unit in the input layer feeds into each of the *K* hidden units (K being a number greater than the number of input layers). These *K* units form the hidden layer of the network, these K hidden units transform the input units P with a chosen nonlinear activation function. The outputs of the activation function are then fed into the output layer of the network.

Methodology

After installing the Keras scheme into R studio, follow the handout steps. we decided to run the code found in the HTML document linked below the Part One description. The description for the single-layer network on hitters' data indicates that the author had built a multiple linear regression, a LASSO regression, and a single-layer network to predict the salary using all the other variables.

Here are the general steps that the author takes in the Lab: Deep Learning under Single Layer Network on Hitters Data:

1. First, load the libraries (ISLR2, glmnet, and keras) into the R environment.
2. Create and name a new data frame "Gitters" to store data points from the "Hitters" data set with no null values.
3. Randomly selected 66.92% of the "Gitters" data as the training set and 33.08% of the data as the testing set using the seed "13".
4. Create a multiple linear regression to predict salary using all the other variables by using the lm() function. And calculated the mean absolute error of the multiple linear regression.
5. Create a data frame named "lpred" of predictions on the test data selected previously with the model and return the mean absolute prediction error from the original data.
6. Scale the dataset and reserve the first column for the response variable (y).
7. Fit a LASSO model to the training data and use it to make a data frame of predictions on the dataset, then return the mean absolute prediction error.
8. Create a single-layer neural network named "modnn" starting with a 50-node hidden layer and ReLU as the activation function. Followed by a dropout layer and then finishing with a single output layer.
9. Turn the input data into a matrix and then scale the data.
10. Pass the model information to the Python instance of the model that has been created with compile()
11. Create a variable that stores the performance metrics of the data returned from fitting "modnn" to the training data and tests it with the validation data. Plot the data. (The number of observations taken per epoch is 32)

12. Use the model to make a prediction matrix on the test data and return the mean absolute prediction error.

Discussion

In this code, linear regression is easy and simple to carry out, but is often rigid in how accurate it can be and isn't as flexible to changes in the data. In addition, variable selection must be carried out manually for multiple linear regression, when compared to Lasso and Neural Network regression, which both do so automatically.

Lasso regression takes a bit more adjustment of the data to carry out in R but is still relatively simple to do with the proper packages. A noticeable difference from linear regression is that Lasso regression has a parameter that performs variable selection, and this R function will find the optimal parameter for you based on the direction it is given. This makes variable selection easier and makes the model more flexible when compared to linear regression. This does come at the cost of some understandability, however, as the shrinkage parameter used in Lasso regression makes it harder to understand exactly what's going on, especially when compared to linear regression's very simple slope values.

Performing regression with a Neural Network takes this a step further, as it is the most flexible yet least understandable model of these three. Neural Networks are essentially a "black box", in that the specifics of what the network is doing is unknown. This means it is impossible to completely understand what the network is doing, but because of the way this model works, it is very flexible and can seamlessly adjust to changes in the data, as well as able to determine what variables are worth using, all on its own. This makes it very flexible.

In terms of R code, Neural Networks are also the hardest to implement, as there are many more steps that need to be done before the data is ready to be put into the network for training. However, the nice thing about this is that the data can be repeatedly fit into the Neural Network, which allows it to become more accurate by performing multiple fittings. When doing this, however, one should be careful not to do this too much as it could lead to overfitting the data, which is not ideal as it would lower the overall performance of the model.

Reference

- https://www.kaggle.com/datasets/floser/hitters
- https://hastie.su.domains/ISLR2/Labs/Rmarkdown_Notebooks/Ch10-deeplearning-lab-keras.html