Alphabet Soup: Neural Network Modelling Report

Laura Faulds

Purpose

The purpose of this analysis is to provide the nonprofit foundation Alphabet Soup with a tool to help it predict whether their selected grant applicants will be successful in their future ventures. We will use machine learning to train and test our neural network model on past Alphabet Soup data, hoping that we can use the same model on future data as well.

Data Preprocessing

*What variable(s) are the target(s) for your model?*

The target variable for our model is “IS\_SUCCESSFUL,” a binary variable telling us whether a given organization succeeded or not after receiving a grant, using 1 to represent success and 0 to represent failure.

*What variable(s) are the features for your model?*

The features for our model are: APPLICATION\_TYPE, AFFILIATION, CLASSIFICATION, USE\_CASE, ORGANIZATION, STATUS, INCOME\_AMT, SPECIAL\_CONSIDERATIONS, and ASK\_AMT. These are the variables that we will train the neural network on analyzing in order to help it define a target.

*What variable(s) should be removed from the input data because they are neither targets nor features?*

The EIN (Employer Identification Number) and NAME categories have been removed from our dataset as they have no bearing on the potential success of a given grant applicant.

Compiling, Training and Evaluating the Model

*How many neurons, layers, and activation functions did you select for your neural network model, and why?*

* **Model #1**: 3 layers, with 80, 30 and 1 neurons, a combination of relu and sigmoid activation functions
* **Model #2**: 3 layers, with 100, 50 and 1 neurons, a combination of relu and sigmoid activation functions
* **Model #3:** 3 layers, with 100, 50 and 1 neurons, a combination of leaky\_relu and sigmoid activation functions
* **Model #4:** 4 layers, with 80, 30, 30, and 1 neurons, a combination of relu and sigmoid activation functions

For Model 1, I used the basic specifications provided in my Starter Code as a neutral jumping off point. With this model, I achieved an accuracy rate of **72.58%.** In my next attempt, I upped the number of neurons in my model to help it learn more complex patterns, resulting in an accuracy rate of **72.72%.** In my third attempt, I decided to keep the neurons and layers consistent since they had worked well for me in the last attempt, but changed my first two activation functions to leaky\_relu from relu, as relu is a generally solid and reliable activation function, but suffers from a dying neurons problem that leads certain neurons to not contribute to training. Frustratingly, I saw no improvement, with my lowest accuracy rate yet: **72.54%.** For my final attempt I went back to the reliable relu, but added another hidden layer to see if that would help. It did not. My final attempt gave me the accuracy rate of **72.52%.**

*Were you able to achieve the target model performance?*

I was not able to achieve the target model performance of 75%. The closest I came was 72.72% with my second attempt overall/first attempt at optimization.

Summary

Overall, this deep learning model did an adequate job of predicting applicant success, but could use some work. I did some independent research on different models and how I could use them to my advantage. While I looked into the possibility of using a Random Forest model instead, it seems like a Neural Network model is much likelier to create something effective and accurate. One source suggested that I train my model using a smaller dataset before I unleash it on the larger dataset, which is something I would like to try if I had more time. Otherwise, I would just continue playing around with different combinations of neurons, optimizers and activation functions.