Claudia Canamas
HW 21
Deep Learning

Overview

The purpose of this analysis is to examine the effectiveness of using neural networks and deep learning models to accurately predict the success of various applicants that have been funded by Alphabet Soup foundation.

Modeling & Results

Preprocessing

The raw data from Alphabet Soup, in the form of a .csv, contained the following columns:

EIN and **NAME**—Identification columns

APPLICATION_TYPE—Alphabet Soup application type

AFFILIATION—Affiliated sector of industry

CLASSIFICATION—Government organization classification

USE_CASE—Use case for funding

ORGANIZATION—Organization type

STATUS—Active status

INCOME_AMT—Income classification

SPECIAL_CONSIDERATIONS—Special consideration for application

ASK_AMT—Funding amount requested

IS_SUCCESSFUL—Was the money used effectively

- Columns EIN and NAME are used purely for identification and do not play a relevant role in the model. They can be removed from the data.
- Columns APPLICATION_TYPE, AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION, STATUS, INCOME_AMT, SPECIAL_CONSIDERATIONS, ASK_AMT are all either categorical or numeric features that describe the applicant in some way; I

believe all of them are relevant to the outcome and should thus be included as features of the model.
- The final column, IS_SUCCESSFUL, is a binary outcome specifying whether or not the applicant was ultimately successful after being funded. This is the target.

Compiling, Training, and Model Evaluation

- The initial attempt of the model used three layers total (input, output, and one hidden layer). The first two layers each had 6 neurons and used ReLu as the activation function, while the output layer just had one neuron and used sigmoid function. After training for 200 epochs, the accuracy score on the test data was 71.92%. These initial parameters were chosen somewhat arbitrarily to try to establish a baseline accuracy score from which to improve upon. ReLu was chosen as the activation function for its simplicity. Sigmoid function was chosen for the output layer due to the binary nature of the outcome.
- After this initial model, three iterative attempts were made to continue optimizing the model, aiming for 75% accuracy on the test set.
    - The first iteration changed the activation function of the first two layers from ReLu to Tanh and reduced the number of epochs to 125. This resulted in an accuracy score of 72.23% on the test set - marginally better than the first attempt. I chose to change the activation function as pure experimentation. The increase in accuracy was enough justification for me to keep it as the activation function moving forward.
    - The second iteration kept all of the changes implemented by the first, but also added 2 additional neurons to the first two layers of the model, increasing neuron count from 6 to 8. The test set had an accuracy of 72.93%, again a very slight improvement. I chose to increase the number of neurons because I hypothesized that the relationship between all of the features and output was more complex than could be captured by only 6 neurons.
    - The third and final iteration of optimization added another hidden layer to the model, for 4 total layers, and increased the number of neurons in all layers (except the output layer) to 10. I increased the number of neurons again for the same reason stated above, and increased the number of hidden layers thinking that more abstraction would allow the model to better capture the complexity as well. The number of epochs in training stayed at 125. The test data scored an accuracy of 73.16% on this version of the model - the best score yet.
- Ultimately, I was not able to achieve the goal accuracy of 75%, but was pleased with the improvement seen from iteration to iteration.

Summary

Overall, the deep learning model was not totally successful but was on the right course. With additional resources, I would continue experimenting with the number of neurons and the number of hidden layers, as well as with different activation functions, keeping in mind that increased complexity can possibly lead to overfitting.
It could also be interesting to try a totally different model on this data for classification, such as random forest. Random forests do not require as much training data as neural networks to achieve good performance and are more easy to interpret, i.e. are less of a black box than neural networks. This could allow the user to further refine the input data and only keep features that are contributing significantly to the outcome.