Devorah Zweig

Professor Lavelley

05/24/23

Neural Model Network

Introduction

A ‘neural network’ is a network that “...stimulates large amounts of interconnected processing units…” in a way that resembles the way that neurons in a human brain would function. It is a way to teach computers to think in a way similar to the human brain. In this project, a neural model network is used to analyze multiple applicants, and correctly determine if they would receive funding. First, the data is processed. Next, the model is compiled, trained and evaluated. Lastly, the model becomes optimized.

Data Processing

In order to process the data, we begin by filtering out the data that is considered useless. In the original model, both the [‘EIN’] and [‘NAME’] columns were dropped from the dataset. After analyzing the remaining data, the [‘APPLICATION_TYPE’] and [‘CLASSIFICATION’] column were renamed to “Other”. Once the data was fully cleaned, it was split into training and testing datasets. In order to train it, the column [‘IS_SUCCESSFUL’] was used as a target variable, in which a value of 1 means successful, while a value of 0 means not successful.

Compile, Train, and Evaluate the Model

The first model created was applied to three layers: Two hidden layers and one output layer. This model generated 477 parameters in total.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 350

 dense_1 (Dense)             (None, 14)                112

 dense_2 (Dense)             (None, 1)                 15

=================================================================
Total params: 477
Trainable params: 477
Non-trainable params: 0
_________________________________________________________________
```

However, after evaluating the model with the test data, it was found that the accuracy came out to be about 0.72, which is less than the ideal, 0.75. Because of this, we later optimized the model with an adjusted dataset in order to yield more accurate results.

In [15]:
```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - loss: 0.5524 - accuracy: 0.7275 - 320ms/epoch - 1ms/step
Loss: 0.5523685812950134, Accuracy: 0.7274635434150696
```

Optimization

In order to optimize the results of our model, the dataset was slightly adjusted from the original model. In this case, only the [‘EIN’] column was dropped, while the [‘NAME’] column remained in the dataset. Again, this model was applied to three layers, two hidden and one output layer. In this case, the results came back different. The amount of parameters jumped to 3,298.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 7)                 3171

 dense_4 (Dense)             (None, 14)                112

 dense_5 (Dense)             (None, 1)                 15

=================================================================
Total params: 3,298
Trainable params: 3,298
Non-trainable params: 0
_________________________________________________________________
```

After evaluating the test data a second time, the results yielded a significantly higher testing accuracy, 0.78. This surpassed the ideal 0.75 and thus deemed the model a better fit for the calculations.

```
✓ 0s
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")

268/268 - 0s - loss: 0.4664 - accuracy: 0.7897 - 453ms/epoch - 2ms/step
Loss: 0.4663858115673065, Accuracy: 0.7897375822067261
```