Deep Learning Assignment
Title: Charity Funding Predictor

Vilaysack Khonsavanh
01/19/2022

Background

This project is to create a deep learning algorithm to predict whether funding will be successful or denied for a non-profit foundation called ‘Alphabet Soup’. Dataset is a CSV file that includes historical data of 34,000 organizations that have received funding from Alphabet Soup. To successfully create the best model for this project, data preprocessing, model building and optimization were carried out. The desired accuracy rate is any rate above 0.75 or 75%.

Data Preprocessing

The data processing started off by dropping columns that include non-beneficial columns, such as ‘EIN ‘ and ‘NAME’. After checking on these data, ‘EIN’ is too individual to be considered as categorical data, however, ‘NAME’ column contains categorical data which could potentially be included as a feature for model training and testing.

After that, the remaining columns were checked for unique values. From those columns, ‘APPLICATION_TYPE’ and ‘CLASSIFICATION’ have more than ten unique values. These values will be converted to columns when applying the ‘pd.get_dummies()’ function. Hence the more unique values we have the more columns will be generated and they need to be reduced. The reduction was done by replacing the value that has minor or rare occurrence to ‘Other’ and binning them up as a new value. After reducing and binning were successfully completed, all categorical data were converted to numerical using pandas get dummies method.

Before creating and fitting the model, the dataset was further splitted into features and target. Features are arrays in the data frame which are not included ‘IS_SUCCESSFUL’ and hence, ‘IS_SUCCESSFUL’ is our target array.

Deep Learning Model

The model was created using a Neural Network in each layer. Firstly, three layers were applied using different numbers of nodes and activation. The model came up with toal parameter of 7,101 and an accuracy rate of 0.73. To improve the accuracy rate for our model, several attempts to optimize our data were carried out.

```python
# Define the model - deep neural net, i.e., the number of input features

nn = tf.keras.models.Sequential()

# First hidden layer

nn.add(tf.keras.layers.Dense(units=80, activation='relu', input_dim=57))
# Second hidden layer

nn.add(tf.keras.layers.Dense(units=30, activation='relu'))
# Output layer

nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
# Check the structure of the model
nn.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 80)                4640
_________________________________________________________________
dense_1 (Dense)              (None, 30)                2430
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 31
=================================================================
Total params: 7,101
Trainable params: 7,101
Non-trainable params: 0
_________________________________________________________________
```

```
8575/1 - 1s - loss: 0.5246 - accuracy: 0.7307
Loss: 0.5577262964922902, Accuracy: 0.7307288646697998
```

Model Optimization

First, Sklearn tuner was used to find the best hyperparameter for the model, using the same dataset as from the step above. The best parameters showed an accuracy rate of 0.73 as the same as from the initial model.

```python
# Get top 3 model hyperparameters and print the values
top_hyper = tuner.get_best_hyperparameters(3)
for param in top_hyper:
    print(param.values)
```

0.4s Python

```
{'activation': 'relu', 'first_units': 57, 'num_layers': 6, 'units_0': 77, 'units_1': 79, 'units_2': 39, 'units_3': 71, 'units_4': 21, 'units_5':
17, 'tuner/epochs': 4, 'tuner/initial_epoch': 2, 'tuner/bracket': 4, 'tuner/round': 1, 'tuner/trial_id': '557671f163e8248cf7af1074d6efa055'}
{'activation': 'tanh', 'first_units': 23, 'num_layers': 5, 'units_0': 57, 'units_1': 9, 'units_2': 5, 'units_3': 55, 'units_4': 77, 'units_5':
61, 'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 4, 'tuner/round': 0}
{'activation': 'relu', 'first_units': 51, 'num_layers': 3, 'units_0': 69, 'units_1': 63, 'units_2': 75, 'units_3': 37, 'units_4': 79,
'tuner/epochs': 2, 'tuner/initial_epoch': 0, 'tuner/bracket': 4, 'tuner/round': 0}
```

```python
# Evaluate the top 3 models against the test dataset
top_model = tuner.get_best_models(3)
for model in top_model:
    model_loss, model_accuracy = model.evaluate(X_test_scaled,y_test,verbose=2)
    print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

9.9s Python

```
8575/1 - 2s - loss: 0.5400 - accuracy: 0.7320
Loss: 0.5540125988871294, Accuracy: 0.7320116758346558
8575/1 - 2s - loss: 0.5616 - accuracy: 0.7318
Loss: 0.5574850544290014, Accuracy: 0.7317784428596497
8575/1 - 2s - loss: 0.5439 - accuracy: 0.7317
Loss: 0.555207087590465, Accuracy: 0.7316617965698242
```

Second attempt was to adjust the data binning, including more data distributions in the columns and use the suggested hyperparameter received from the first attempt. The results didn't show any improvement of model accuracy rate.

Third attempt was to bring back the 'NAME' column, The unique values in this column which are names of organizations who applied for funding. Then all names were put into the bins. This time the model generated a total of 24,301 parameters because of the large number of features in the dataset.

The result showed an improvement of the model accuracy rate to 0.77 and loss rate at 0.49.

```
# Define the model - deep neural net, i.e., the number of input features and

nn = tf.keras.models.Sequential()

# First hidden layer

nn.add(tf.keras.layers.Dense(units=80, activation='relu', input_dim=272))
# Second hidden layer

nn.add(tf.keras.layers.Dense(units=30, activation='relu'))
# Output layer

nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
# Check the structure of the model
nn.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 80)                21840
_________________________________________________________________
dense_1 (Dense)              (None, 30)                2430
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 31
=================================================================
Total params: 24,301
Trainable params: 24,301
Non-trainable params: 0
_________________________________________________________________
```