Challenge 21: Deep Learning

Overview

The purpose of this analysis was to use machine learning and neural networks to create an algorithm that could predict if applicants for funding, from the nonprofit foundation Alphabet Soup, will be successful in their ventures.

Results

To preprocess the data, I removed the irrelevant information within the data set. This consisted of dropping the "EIN" and "NAME" columns as they didn't provide any relevance. The remaining columns were considered the model's features. Next, the number of unique values was determined for each feature and if the value was greater than 10, a cutoff point was chosen to bin the rare categorical values in a combined value deemed "Other". This was done for the APPLICATION_TYPE and CLASSIFICATION values with bins of 500 and 100 respectively. After, the data was split into training and testing variables, with the target variable being IS_SUCCESSFUL. The categorical data was encoded using the get_dummies() function and the model was ready to be trained.

Compiling, Training, and Evaluating the Model:

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
#  YOUR CODE GOES HERE
input_features = len(X_train_scaled[0])
hidden_node1 = 16
hidden_node2 = 16

nn = tf.keras.models.Sequential()

# First hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node1, activation='relu', input_dim=input_features))

# Second hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node2, activation='relu'))

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

To begin, three layers were applied to the model. The first two having 16 nodes each and the relu activation function, while the third had one node and the sigmoid function. The number of nodes was randomly selected to act as a benchmark for further optimization. The results from this model are as follows:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 16)                800
_________________________________________________________________
dense_1 (Dense)              (None, 16)                272
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 17
=================================================================
Total params: 1,089
Trainable params: 1,089
Non-trainable params: 0
_________________________________________________________________
```

```python
# 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}")
```

```
8575/8575 - 0s - loss: 0.5595 - acc: 0.7248
Loss: 0.559534022467477, Accuracy: 0.724781334400177
```

The model created 1,089 parameters and resulted in an accuracy of 72.4%, falling short of the goal at 75%.

Optimization:

Five attempts were made after the initial model to reach the target of 75%. The first three changed individual parameters while the last two changed combinations.

Attempt 1:

```python
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
#  YOUR CODE GOES HERE
input_features = len(X_train_scaled[0])
hidden_node1 = 16
hidden_node2 = 16
hidden_node3 = 32

nn = tf.keras.models.Sequential()

# First hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node1, activation='relu', input_dim=input_features))

# Second hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node2, activation='relu'))

#Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_node3, activation='relu'))

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 16)                800
_________________________________________________________________
dense_1 (Dense)              (None, 16)                272
_________________________________________________________________
dense_2 (Dense)              (None, 32)                544
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,649
Trainable params: 1,649
Non-trainable params: 0
_________________________________________________________________
```

```python
# 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}")
```

```
8575/8575 - 0s - loss: 0.5627 - acc: 0.7241
Loss: 0.5627252310809867, Accuracy: 0.7240816354751587
```

Attempt 2:

```python
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
#  YOUR CODE GOES HERE
input_features = len(X_train_scaled[0])
hidden_node1 = 16
hidden_node2 = 16

nn = tf.keras.models.Sequential()

# First hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node1, activation='tanh', input_dim=input_features))

# Second hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node2, activation='tanh'))


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

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 16)                800
_________________________________________________________________
dense_5 (Dense)              (None, 16)                272
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 17
=================================================================
Total params: 1,089
Trainable params: 1,089
Non-trainable params: 0
_________________________________________________________________
```

```python
# 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}")
```

```
8575/8575 - 0s - loss: 0.5548 - acc: 0.7209
Loss: 0.5548409486373034, Accuracy: 0.7209329605102539
```

Attempt 3:

```python
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
#  YOUR CODE GOES HERE
input_features = len(X_train_scaled[0])
hidden_node1 = 16
hidden_node2 = 16

nn = tf.keras.models.Sequential()

# First hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node1, activation='relu', input_dim=input_features))

# Second hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node2, activation='relu'))


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

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 16)                800
_________________________________________________________________
dense_8 (Dense)              (None, 16)                272
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 17
=================================================================
Total params: 1,089
Trainable params: 1,089
Non-trainable params: 0
_________________________________________________________________
```

```python
# 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}")
```

```
8575/8575 - 0s - loss: 0.5643 - acc: 0.7240
Loss: 0.5643033004085107, Accuracy: 0.7239649891853333
```

Attempt 4:

```python
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
#  YOUR CODE GOES HERE
input_features = len(X_train_scaled[0])
hidden_node1 = 6
hidden_node2 = 12
hidden_node3 = 6
hidden_node4 = 12

nn = tf.keras.models.Sequential()

# First hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node1, activation='relu', input_dim=input_features))

# Second hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node2, activation='relu'))

#Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_node3, activation='tanh'))

#Fourth hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_node4, activation='tanh'))

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

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
dense_10 (Dense)            (None, 6)                 300
_________________________________________________________________
dense_11 (Dense)            (None, 12)                84
_________________________________________________________________
dense_12 (Dense)            (None, 6)                 78
_________________________________________________________________
dense_13 (Dense)            (None, 12)                84
_________________________________________________________________
dense_14 (Dense)            (None, 1)                 13
=================================================================
Total params: 559
Trainable params: 559
Non-trainable params: 0
_________________________________________________________________
```

```python
# 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}")
```

```
8575/8575 - 0s - loss: 0.5583 - acc: 0.7213
Loss: 0.5583434527121897, Accuracy: 0.7212827801704407
```

Attempt 5:

```python
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
#  YOUR CODE GOES HERE
input_features = len(X_train_scaled[0])
hidden_node1 = 25
hidden_node2 = 25
hidden_node3 = 35
hidden_node4 = 50

nn = tf.keras.models.Sequential()

# First hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node1, activation='relu', input_dim=input_features))

# Second hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_node2, activation='relu'))

#Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_node3, activation='relu'))

#Fourth hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_node4, activation='relu'))

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

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
dense_15 (Dense)            (None, 25)                1250
_________________________________________________________________
dense_16 (Dense)            (None, 25)                650
_________________________________________________________________
dense_17 (Dense)            (None, 35)                910
_________________________________________________________________
dense_18 (Dense)            (None, 50)                1800
_________________________________________________________________
dense_19 (Dense)            (None, 1)                 51
=================================================================
Total params: 4,661
Trainable params: 4,661
Non-trainable params: 0
_________________________________________________________________
```

```python
# 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}")
```

```
8575/8575 - 0s - loss: 0.6488 - acc: 0.7247
Loss: 0.6487750527879587, Accuracy: 0.7246647477149963
```

Each optimization failed to reach the 75% accuracy goal.

Summary:

With each attempt have a lower accuracy score than the original, I don't believe that adjusting the parameters would be the solution to reaching a score greater than or equal to 75%. I would recommend adjusting the input data or changing the target features of the model and optimizing from there.