Sanjay Dabra
08/17/2022

Assignment 22: Deep Learning
Alphabet Soup Charity Funding Success Predictor

The purpose is to design a tool to assist Alphabet Soup to select candidates with the best chance of success. Deep learning and neural networks were used. Google Colaboratory, keras, and t-SNE are the primary tools used.

Preprocessing Data:

After importing charity_data.csv, the data was cleaned. This was accomplished by:

1. Removing columns “EIN” and “NAME”.
2. Binning the “APPLICATION_TYPE” based on the count of number applications, then choosing the “APPLICATION_TYPE” based on those with greater than 500.
3. Binning the “CLASSIFICATION” based on the count of number applications, then choosing the “CLASSIFICATION” based on those with greater than 100.
4. Convert categorial data to numeric with ‘pd.get_dummies’.
5. Split the data into features and target arrays. Our target array is the column “IS_SUCCESSFUL”.
6. Splitting the dataset into training and testing datasets.
7. Using StandardScaler to scale the data.

Compiling, Training, and Evaluating the Model:

Four models were created. The first model had 2 layers, 30 neurons in the first layer and 10 in the second.

```
In [25]:  # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 = 30
hidden_nodes_layer2 = 10

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))

# Output layer
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_10 (Dense)            (None, 30)                1500

 dense_11 (Dense)            (None, 10)                310

 dense_12 (Dense)            (None, 1)                 11

=================================================================
Total params: 1,821
Trainable params: 1,821
Non-trainable params: 0
_________________________________________________________________
```

The accuracy of this model was 73.0%, slightly below the 75% threshold.

The next model also had 2 layers, but the number of neurons was increased to 80 for the first layer and 30 for the second.

```
In [31]:
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 = 80
hidden_nodes_layer2 = 30

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))

# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_13 (Dense)            (None, 80)                4000

 dense_14 (Dense)            (None, 30)                2430

 dense_15 (Dense)            (None, 1)                 31

=================================================================
Total params: 6,461
Trainable params: 6,461
Non-trainable params: 0
_________________________________________________________________
```

This model did not perform any better or worse, it also had an accuracy of 73.0%.

The number of layers was increased for the third model. The model consisted of 3 layers, 10 neurons for the first layer, 20 for the second, and 30 for the third.

```
In [38]:
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 = 10
hidden_nodes_layer2 = 20
hidden_nodes_layer3 = 30

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))

nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer3, activation='relu'))

# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_18 (Dense)            (None, 10)                500

 dense_19 (Dense)            (None, 20)                220

 dense_20 (Dense)            (None, 30)                630

 dense_21 (Dense)            (None, 1)                 31

=================================================================
Total params: 1,381
Trainable params: 1,381
Non-trainable params: 0
_________________________________________________________________
```

Again, the accuracy was the same at 73.0%

The fourth model went back to 2 layers, but the neurons were dramatically increased to 300 neurons for the first layer and 100 for the second.

```
In [43]: # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 = 300
hidden_nodes_layer2 = 100

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))


# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_22 (Dense)            (None, 300)               15000

 dense_23 (Dense)            (None, 100)               30100

 dense_24 (Dense)            (None, 1)                 101

=================================================================
Total params: 45,201
Trainable params: 45,201
Non-trainable params: 0
_________________________________________________________________
```

This showed a minuscule amount of improvement in the accuracy, 73.1%.

The last model increased the number of layers to 4, 10 neurons for the first, 20 for the second and 30 for both the third and fourth layers.

```
In [15]: # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 = 10
hidden_nodes_layer2 = 20
hidden_nodes_layer3 = 30
hidden_nodes_layer4 = 30

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))

# Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer3, activation='relu'))

# Fourth hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer4, 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, 10)                500

 dense_1 (Dense)             (None, 20)                220

 dense_2 (Dense)             (None, 30)                630

 dense_3 (Dense)             (None, 30)                930

 dense_4 (Dense)             (None, 1)                 31

=================================================================
Total params: 2,311
Trainable params: 2,311
Non-trainable params: 0
_________________________________________________________________
```

This model was slightly worse than the other models with an accuracy of 72.7%.

Summary:

The results of the models are summarized in the table below.

IPYNB File	Layer 1 (Neurons)	Layer 2 (Neurons)	Layer 3 (Neurons)	Layer 4 (Neurons)	Parameters	Accuracy
Starter Code	30	10	N/A	N/A	1821	73.0%
Optimization_1	80	30	N/A	N/A	6461	73.0%
Optimization_2	10	20	30	N/A	1381	73.0%
Optimization_3	300	100	N/A	N/A	45201	73.1%
Optimization_4	10	20	30	30	2311	72.7%

Increasing the layers and adjusting the neurons had little effect in obtaining the target accuracy of 75%. It is recommended to use keras tuner to find the optimal number of layers and neurons, however additional programming may need to be required. For this dataset, the shape of the data did not allow the tuner to run.