Report – Assignment 2

Akshatha Thonnuru Swamygowda, Sarveshwaran Sampathkumar

The assignment is to create a Keras model for the function (y1,y2) = f(x1,x2) using randomized sets of numbers. The assignment also directs us to train and test the model along with the final report explaining the assignment.

We have implemented this assignment on Spyder (Python). The assignment requires us to use two values of X (i.e. x1 and x2) and Y (i.e. y1 and y2). To implement this in python, we make use of the dataset for x and y value and we have used keras model.

```
from keras.models import Sequential
import matplotlib.pyplot as plt
from keras.layers import Dense
from keras.optimizers import Adam
import numpy as np

#Training Data
#Training Data
x1 = np.multiply(np.random.random([1000]),10)
x2 = np.multiply(np.random.random([1000]),10)
y1 = np.add(np.add(pow(x1,2),x2),5)
y2 = np.add((np.add(pow(x1,3),x2)),6)
X_train=np.stack((x1,x2)).T
Y_train=np.stack((y1,y2)).T
```

Our model has three-layer, input layer, output layer and one hidden layer. We have two neurons for input layer, two neurons for output layer and four neurons in hidden layer and all the layer are dense. We are using relu activation function for our model.

```
#creating layers for the model
model=Sequential()
model.add(Dense(8,input_shape=(2,),activation='relu'))
model.add(Dense(4,activation='relu'))
model.add(Dense(2))
model.summary()
```

For compiling and training the model we have used model.compile and model.fit. We are using the adam optimizer for our model.

```
#for compiling the model
model.compile(loss='mean_squared_error',optimizer=Adam(lr=0.003), metrics=['accuracy'])
#for training the model
history=model.fit(Y_train,X_train,epochs=10)
```

We are using predict function for predicting the values of y1 and y2 and testing the model by randomizing the value of x1 and x2.

```
#testing
#
#Training Data
xt_1 = np.multiply(np.random.random([1000]),10)
xt_2 = np.multiply(np.random.random([1000]),10)
yt_1 = np.add(np.add(pow(x1,2),x2),5)
yt_2 = np.add((np.add(pow(x1,3),x2)),6)

Y_test=np.stack((yt_1,yt_2)).T
print(Y_test)


tes=model.predict(Y_test)
print(tes)
```

After the model is trained we are showing the accuracy for each set and we are plotting the graph for loss and epochs.

```
def plot_loss_accuracy(history):
    fig = plt.figure(figsize=(12, 6))
    ax = fig.add_subplot(1, 2, 1)
    ax.plot(history.history["loss"],'r-x', label="Train Loss")
    ax.legend()
    ax.set_title('Training loss')
    ax.grid(True)

plot_loss_accuracy(history)
```

Output of Our model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_36 (Dense)             (None, 8)                 24
_________________________________________________________________
dense_37 (Dense)             (None, 4)                 36
_________________________________________________________________
dense_38 (Dense)             (None, 2)                 10
=================================================================
Total params: 70
Trainable params: 70
Non-trainable params: 0
```

Epoch of 10 sets

```
_________________________________________________________________
Epoch 1/10
1000/1000 [==============================] - 1s 1ms/step - loss: 2404366.2688 - acc:
0.3560
Epoch 2/10
1000/1000 [==============================] - 0s 108us/step - loss: 196359.5567 - acc:
0.8900
Epoch 3/10
1000/1000 [==============================] - 0s 107us/step - loss: 86094.1333 - acc:
0.9230
Epoch 4/10
1000/1000 [==============================] - 0s 94us/step - loss: 51319.2144 - acc:
0.9410
Epoch 5/10
1000/1000 [==============================] - 0s 94us/step - loss: 35010.2574 - acc:
0.9490
Epoch 6/10
1000/1000 [==============================] - 0s 109us/step - loss: 26399.7962 - acc:
0.9560
Epoch 7/10
1000/1000 [==============================] - 0s 109us/step - loss: 21480.1628 - acc:
0.9670
Epoch 8/10
1000/1000 [==============================] - 0s 94us/step - loss: 18772.7418 - acc:
0.9700
Epoch 9/10
1000/1000 [==============================] - 0s 94us/step - loss: 17209.7596 - acc:
0.9750
Epoch 10/10
1000/1000 [==============================] - 0s 94us/step - loss: 16166.7606 - acc:
0.9750
[17 24 10 14 27 29 12 25 10 20]
[40 38 41 59 36 32 46 44 31 42]
[[1889 1311]
 [2020  868]
 [1781 1581]
 [3677 3285]
 [2025  567]
 [1865  183]
 [2260 1972]
 [2561 1311]
 [1061  861]
 [2164 1364]]
```

random value of x1 and x2

Y1 and Y2 value of the model

The predicted values of the model

```
[[-6.8002507e-02  6.9370264e-01]
 [-6.8002507e-02  6.9370264e-01]
 [-6.8002507e-02  6.9370264e-01]
 [-6.8002507e-02  6.9370264e-01]
 [-6.8002507e-02  6.9370264e-01]
 [ 6.8304817e+01  3.5301712e+00]
 [-6.8002507e-02  6.9370264e-01]
 [-6.8002507e-02  6.9370264e-01]
 [-6.8002507e-02  6.9370264e-01]
 [-6.8002507e-02  6.9370264e-01]]
```


Training loss
Train Loss
2500000
2000000
1500000
1000000
500000
0
0
2
4
6
8