Due: 13.12.2019, 22:59.

END3971 Artificial Intelligence & Expert Systems

Assignment #3

Group Member 1 NUMBER

Group Member 2 NUMBER

Group Member 3 NUMBER

Group Member 4 NUMBER

1 Binary Classification with Artificial Neural Networks.

1.1 Artificial Neural Network

We used keras library for Artificial Neural Network training, split input and output feature and scale input features with scikitlearn.StandardScaler() function. Set a threshold to build binary classified prediction of test data. After that we configured a confusion matrix because we need matrix values to calculate accuracy as given.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

In following lines, you can find fully-commented version of our piece of code for Assignment #3.

 # Import Libraries

```
#
================================================================================
import pandas
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
# Just disables the warning, doesn't enable AVX/FMA
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
#
================================================================================
```

 # Configs

Choosed 2 #of Neurons and 2 Activation Number as assigned.

```
#
================================================================================
num_neuron1 = 8
num_neuron2 = 16
act_function1 = 'relu'
act_function2 = 'sigmoid'
#
================================================================================
```

This is the function that complete training process for specified model with 7 inputs (X_train, X_test, y_train, y_test, num_neurons, act_function, model_num):

 # model function

```
def binary_classification(X_train, X_test, y_train, y_test, num_neurons, act_function, model_num):


  print('\n')
  print('>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
MODEL',model_num,'<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<')
  print('Number of Neurons in Model',model_num,'is ...:',num_neurons)
  print('Activation Function in Model',model_num,'is ...:',act_function)
  classifier = Sequential()
  #First Hidden Layer
  classifier.add(Dense(num_neurons, activation= act_function, input_dim=4))
  #Output Layer
  classifier.add(Dense(1, activation='sigmoid'))
```

For output layer, sigmoid function choosed. 1 neuron, because, we want a binary output (0 or 1)

```
  #Compiling the neural network
  classifier.compile(optimizer ='rmsprop',loss='binary_crossentropy', metrics =['accuracy'])
```

For optimizer rmsprop selected from keras library and for loss function binary crossentropy because of binary classification model.

```
  #Fitting the data to the training dataset
  classifier.fit(X_train,y_train, batch_size=10, epochs=100, verbose=0)
```

Fit model with these specs (10 Batch size and repeating 100 epochs for optimization)

```
  # evaluate model accuracy
  # find predicted value for test data
  y_pred=classifier.predict(X_test)
  # Threshold = 0.6 for binary classification
  y_pred =(y_pred>0.5)
  # Construct confusion matrix
  cm = confusion_matrix(y_test, y_pred)
  # Calculate accuracy with given formula ==> (TP+TN)/(TP+TN+FP+FN)
  acc = (cm[0][0]+cm[1][1]) / (cm[0][0]+cm[1][0]+cm[0][1]+cm[1][1])
  print('Model',model_num,'Accuracy is ..:', acc)
  print('>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> END <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<')
```

```
# load dataset
dataframe = pandas.read_excel("PerturbedBanknoteAuthentication.xlsx", header=None)
dataset = dataframe.values

# split input and output columns
X = dataset[:,0:4].astype(float)
Y = dataset[:,4]

# SCALE INPUT FEATURE
#
================================================================================
#standardizing the input feature
sc = StandardScaler()
X = sc.fit_transform(X)
#
================================================================================
# split train test dataset
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3)
```

Split dataset with 30% test data and 70% training data.

print('>>>>>>>>>>>> Program is starting... <<<<<<<<<<<<<<')

 # model 1

# of Neurons = 8

Activation Function = Relu

binary_classification(X_train, X_test, y_train, y_test, num_neuron1, act_function1, 1)

 # model 2

# of Neurons = 8

Activation Function = sigmoid

binary_classification(X_train, X_test, y_train, y_test, num_neuron1, act_function2, 2)

 # model 3

# of Neurons = 16

Activation Function = Relu

binary_classification(X_train, X_test, y_train, y_test, num_neuron2, act_function1, 3)

 # model 4

# of Neurons = 16

Activation Function = sigmoid

binary_classification(X_train, X_test, y_train, y_test, num_neuron2, act_function2, 4)

print('>>>>>>>>>>>> Program finished! <<<<<<<<<<<<<<')

1.2 Accuracy

Find the accuracy for all your logistic regression models by using the following formula:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

Fill in the following table:

Accuracy	Activation Function (1)	Activation Function (2)
#Neurons (1)	0.723	0.679
#Neurons (2)	0.75	0.684

You can check the result by running our .py file on your python compiler or terminal with python 3.