DeepLearning Class (ANN_Viz)

(Github.com): TarekARashed/ANN_VIZ (github.com)

The purpose of DeepLearning class is to generate the image diagram of a feedforward neural network structure and show results at the output layer for a given data sample. It can be used as a simulator to verify the deep learning results, for education purposes, or as a way to simplify communication between experts and non-expert people.

Description

The DeepLearning class consists of the following public and private interfaces:

The public interfaces:

1- class DeepLearning().

2- Add_Layer(self, No_Neurons, ActivationF, Threshold_Value="None").

3- compile(self,File_Name=None, Inputs=None, Random_Values=None).

4- ANNToolBox(self, Action="Draw", Sample_Data=[], Digram_Title="ANN Visulization").

5- Create_JSON_Structure(self, File_Name).

The most important private interfaces are:

1- __Draw(self, Digram_Title="ANN Visulization").

2- __predict(self, Sample_Data).

3- __Weights_Settings(self, Min, Max):

4- __Activation_Function(self, Neuron_Output, Activation_Fun, Threshold_Value)

Dependencies

The class uses Python 3.x and Windows 10. The following packages are required:

1. tkinter import *
2. json
3. numpy
4. random
5. sys

Executing program

Step 1: Create an Object from class DeepLearning(). For example, model=DeepLearning().

Step 2: Use Add_Layer(No_Neurons, ActivationF, Threshold_Value="None") to create a layer and its associated parameters. No_Neurons is an integer value, referring to the total number of neurons at the layer n (required), ActivationF is a string, referring to a layer Activation function(recommended) and Threshold_Value is a float value, referring to a layer Threshold value (optional).

Example: Create a feedforward neural network with 3 hidden layers and output layer

1- To create the first hidden layer of 5 neurons, no threshold value, and with Relu activation function: use model.Add_Layer(5, "Relu")

2- To create the second hidden layer of 6 neurons, no threshold value, and with Relu activation function: use model.Add_Layer(6, "Relu")

3- To create the third hidden layer of 3 neurons, no threshold value, and with Tanh activation function: model.Add_Layer(3, "Tanh")

4- To create the output layer of 1 neurons, threshold value is 0.5, and with Sigmoid activation function: use model.Add_Layer(1, "Sigmoid", Threshold_Value=0.5)

Step 3: Use compile(File_Name=None, Inputs=None, Random_Values=None) to setup the input layer and generate random values across all Artificial Neural Network (ANN) structure.

Where, File_Name=None is a string referring to a JSON file structure (optional). It is a file name which holds an ANN structure. File_Name is optional if you do not have a JSON ANN file structure. If you have one and want to load the ANN parameters, write its name. If you do not have a JSON ANN file and you want to create one for your current ANN model use the method Create_JSON_Structure(File_Name).

, Inputs=None is an integer value, referring to the number of inputs (neurons) of the input layer. It is required if you do not have a JSON file structure.

, Random_Values=None is a list of two values [min, max] used to generate random numbers between min and max values across all the current ANN. It is required if you use the Inputs argument.

Example: Generate a ANN structure using the file "ANN.JSON".

1- Use compile(File_Name="ANN.JSON") with only one argument.

Example: Generate a ANN structure using 10 inputs (Neurons) and initially generate random values between 0, 1 across all the current ANN

1- Use compile(Inputs=10, Random_Values=[0,1]) with two arguments.

Note: For intercept value (b) of each neuron, the Linear equation wx+b, b values is generated with the value 1 or -1.

Step 4: To generate an ANN diagram or show the output results at the output layer of our current ANN structure, use ANNToolBox(Action="Draw", Sample_Data=[], Digram_Title="ANN Visulization")

Where, Action="Draw" is a string with two values "draw" and "predict. The default value is "Draw".

Action="Draw" is used to generate the ANN diagram of our current ANN model. It does show any results at the output layer.

Action="predict" is used to generate an ANN diagram and show ANN results at the output layer.

Sample_Data=[] refers to our input data. It is a list of our input items. It is required if you use Action="predict".

Digram_Title="ANN Visualization" is a string, referring to our diagram title. It is optional and the default value is "ANN Visualization".

Step 5 (optional): Use create_JSON_Structure(File_Name) If you need to save the current ANN structure and use it in the future.

Where, File_Name is a string, referring to an name of ANN structure file.

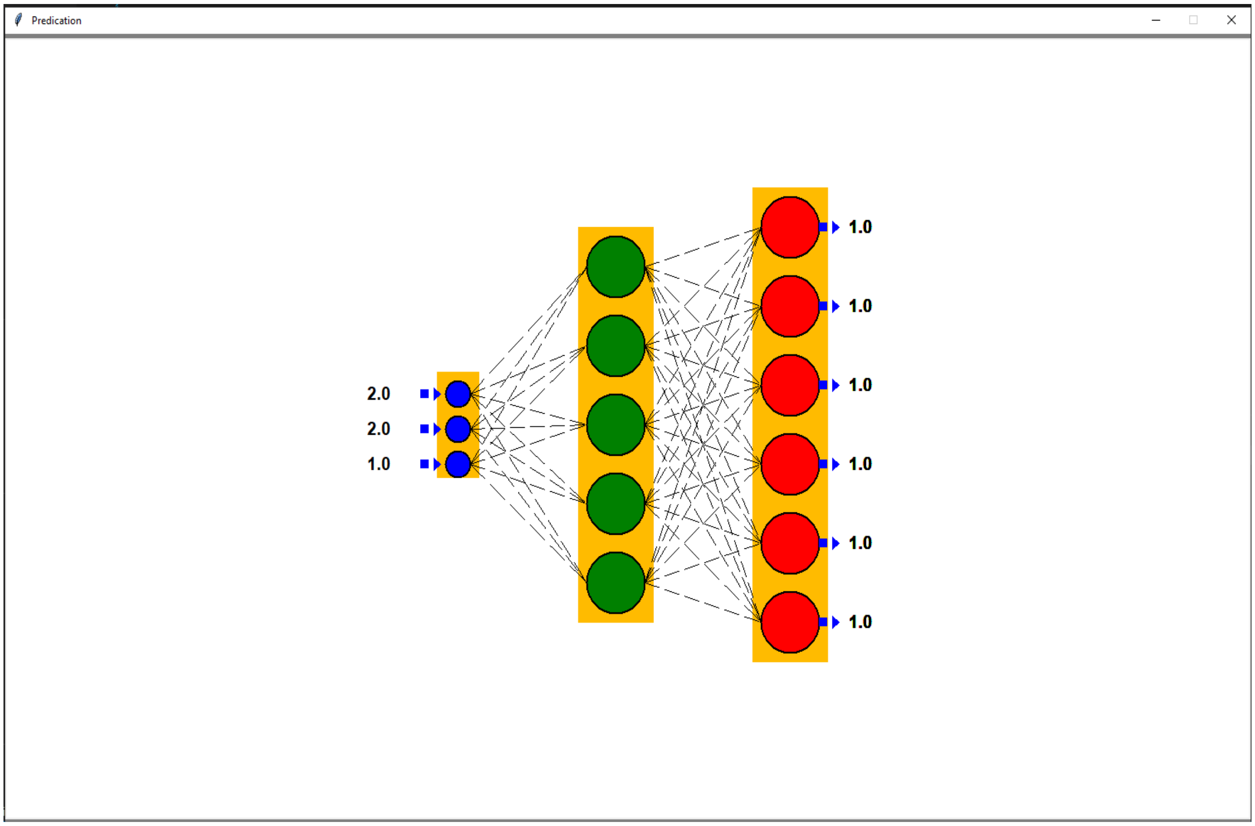
Private methods:

1- __Draw(self, Digram_Title="ANN Visulization") is used to generate a diagram of our current model.

2- __predict(self, Sample_Data) is used to generate the ANN diagram and predict the outputs at the output layer of our current model. Sample_Data is a list, referring to a sample data.

3- __Weights_Settings(self, Min, Max) is used to generate ANN weights between min, max values.

4- __Activation_Function(self, Neuron_Output, Activation_Fun, Threshold_Value) is used by each single neuron. It applies activation function and then computes activation and threshold value at each neuron.


Predication
2.0
2.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0