ANN Practical 1

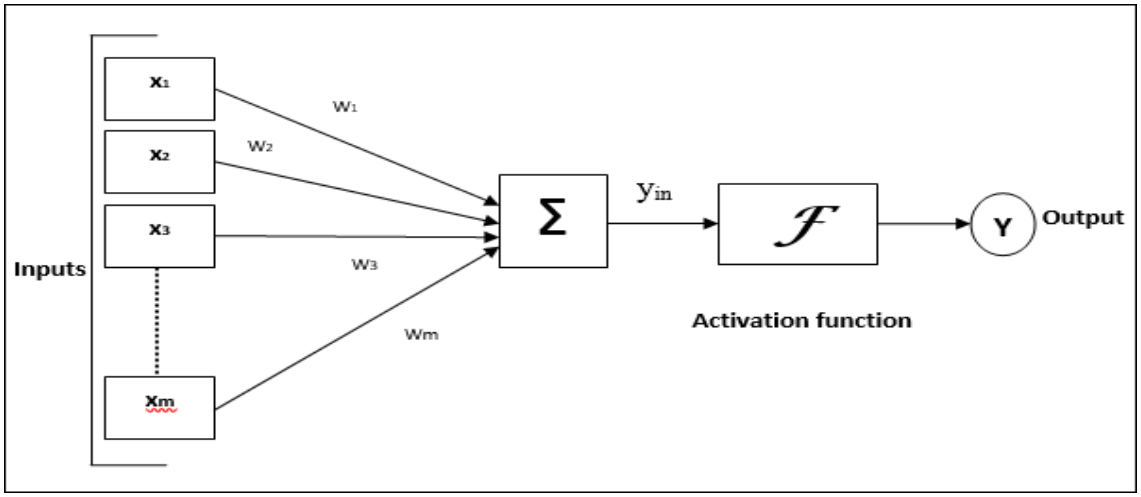
Title – Write a Python program to plot a few activation functions that are being used on neural networks.

In Artificial Neural Networks (ANN), activation functions are essential components used to introduce non-linearity into the network. They are applied to the output of each neuron in a neural network layer, except for the input layer. Activation functions determine the output of a neural network, enabling the network to learn complex patterns and relationships in data.

-Activation function is an internal state of neuron ,its used to convert input signal on node of ann to an output signal

- they introduce non linear properties to network

-Used to map result values in between 0 to 1 or -1 to 1 depending upon type of function used



Weighted sum of input become input signal to activation function to give one output signal.

$$yin=x1.w1+x2.w2+x3.w3\ldots xm.wm$$

$$\text{i.e., Net input } yin=\sum xi.wi$$

Here are some common activation functions used in ANNs:

1. **Sigmoid Function**: The sigmoid function squashes the input into the range [0, 1]. It is defined as: $\text{sigmoid}(x)=1+e-x1$

It is useful in binary classification problems where the output needs to be interpreted as probabilities.

2. **ReLU (Rectified Linear Unit)**: ReLU is a simple and popular activation function. It returns 0 for negative inputs and the input value for positive inputs. It is defined as: $\text{ReLU}(x)=\max(0,x)$

ReLU is computationally efficient and helps alleviate the vanishing gradient problem during training.

3. **Tanh (Hyperbolic Tangent)**: Tanh function squashes the input into the range [-1, 1]. It is defined as: $\tanh(x)=ex+e−xex−e−x$

Tanh is similar to the sigmoid function but centered at 0. It is often used in hidden layers of neural networks.

4. **Softmax Function**: The softmax function is used in the output layer for multi-class classification problems. It converts raw scores (logits) into probabilities. It is defined as: $\text{softmax}(xi)=\sum j=1Nexjexi$

Where N is the number of classes.

These activation functions introduce non-linearities into the neural network, enabling it to learn complex mappings between inputs and outputs. The choice of activation function depends on the nature of the problem and the characteristics of the data. Experimentation and empirical testing often help determine the most suitable activation function for a particular task.

//Import necessary libraries: numpy as np and matplotlib.pyplot as plt. These libraries are commonly used for numerical computing and creating plots in Python.

```
import numpy as np

import matplotlib.pyplot as plt
```

//Define activation functions: Four activation functions are defined - sigmoid, ReLU, tanh, and softmax. These functions are commonly used in neural networks and machine learning models

```
def sigmoid(x):

return 1 / (1 + np.exp(-x))

def relu(x):

return np.maximum(0, x)
```

def tanh(x):

return np.tanh(x)

def softmax(x):

return np.exp(x) / np.sum(np.exp(x))

//Create input values (**x**): Generate 100 equally spaced values between -10 and 10 using **numpy.linspace()** function. These values will be used to plot the activation functions.

Create x values

x = np.linspace(-10, 10, 100)

- **-10**: This is the start value of the sequence.
- **10**: This is the end value of the sequence.
- **100**: This is the number of points you want to generate between the start and end values, evenly spaced.

//np.linspace() generates an array of evenly spaced numbers over a specified interval. In this case, it generates 100 equally spaced numbers between -10 and 10.

//Create subplots for each activation function: Use matplotlib.pyplot.subplots() to create a 2x2 grid of subplots with a specified figure size.

Create plots for each activation function

fig, axs = plt.subplots(2, 2, figsize=(8, 8))

1. //**plt.subplots()**: This is a function provided by the matplotlib.pyplot library, which was imported at the beginning of the script. It is used to create a figure and a set of subplots.
2. **Parameters**:
   - **2, 2**: These two integer values specify the number of rows and columns of subplots you want in the figure. In this case, it creates a 2x2 grid, meaning there will be 2 rows and 2 columns of subplots.
   - **figsize=(8, 8)**: This parameter specifies the width and height of the figure in inches. Here, (8, 8) indicates that the width and height of the figure will be 8 inches each.
3. **Return Values**:

- **fig**: This is the variable that stores the generated figure object. You can use this variable to customize properties of the entire figure.
- **axs**: This is a NumPy array of axes objects. Each element in the array corresponds to a subplot. You can use these axes objects to customize properties of individual subplots.

//Plot each activation function on its respective subplot:

axs[0, 0].plot(x, sigmoid(x))

axs[0, 0].set_title(‘Sigmoid’)

axs[0, 1].plot(x, relu(x))

axs[0, 1].set_title(‘ReLU’)

axs[1, 0].plot(x, tanh(x))

axs[1, 0].set_title(‘Tanh’)

axs[1, 1].plot(x, softmax(x))

axs[1, 1].set_title(‘Softmax’)

1. **//axs[0, 0].plot(x, sigmoid(x))**: This line plots the sigmoid activation function using the data x. The **sigmoid()** function likely computes the sigmoid function for each value of x, and it is plotted on the subplot located at the first row and first column ([0, 0]) of the grid specified by axs.
2. **axs[0, 0].set_title('Sigmoid')**: This line sets the title of the subplot at position [0, 0] to 'Sigmoid'.
3. **axs[0, 1].plot(x, relu(x))**: This line plots the ReLU (Rectified Linear Unit) activation function using the data x. Similar to the previous line, it is plotted on the subplot located at the first row and second column ([0, 1]) of the grid.
4. **axs[0, 1].set_title('ReLU')**: This line sets the title of the subplot at position [0, 1] to 'ReLU'.
5. **axs[1, 0].plot(x, tanh(x))**: This line plots the hyperbolic tangent (tanh) activation function using the data x. It is plotted on the subplot located at the second row and first column ([1, 0]) of the grid.
6. **axs[1, 0].set_title('Tanh')**: This line sets the title of the subplot at position [1, 0] to 'Tanh'.
7. **axs[1, 1].plot(x, softmax(x))**: This line plots the softmax activation function using the data x. It is plotted on the subplot located at the second row and second column ([1, 1]) of the grid.
8. **axs[1, 1].set_title('Softmax')**: This line sets the title of the subplot at position [1, 1] to 'Softmax'.

Add common axis labels and titles

fig.suptitle(Common Activation Functions)

//Set common axis labels for all subplots:

for ax in axs.flat:

ax.set(xlabel='x', ylabel=;'y')

//Adjust spacing between subplots using plt.subplots_adjust():

Adjust spacing between subplots

plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9,

wspace=0.4, hspace=0.4)

Show the plot

plt.show()

OUTPUT –