1.1.Experiment No. 1

Aim: Write a Python program to plot a few activation functions that are being used in neural networks.

Objective: To learn about activation functions and perform its code in python.

Theory:

What are Activation Functions?

A neural network activation function is a function that introduces nonlinearity into the model. A neural network has multiple nodes in each layer, and in a fully connected network, every node in one layer is connected to every node in the next layer.

Why use activation functions?

1. Activation functions' main objective is to add non-linearity into the network so that it can model more intricate and varied interactions between inputs and outputs. In the absence of activation functions, the network would only be capable of performing linear transformations, which cannot adequately represent the complexity and nuances of real-world data. Since neural networks need to implement complex mapping functions, non-linear activation functions must be used to introduce the much-needed nonlinearity property that allows approximating any function.
2. Normalizing each neuron in the network's output is a key benefit of utilizing activation functions. Depending on the inputs it gets and the weights associated to those inputs, a neuron's output can range from extremely high to extremely low. Activation functions make ensuring that each neuron's output falls inside a defined range, which makes it simpler to optimise the network during training.

Types of Activation Function:

Linear Activation Function: The linear activation function, also known as "no activation," or "identity function" (multiplied x 1.0), is where the activation is proportional to the input. The function doesn't do anything to the weighted sum of the input, it simply spits out the value as it is

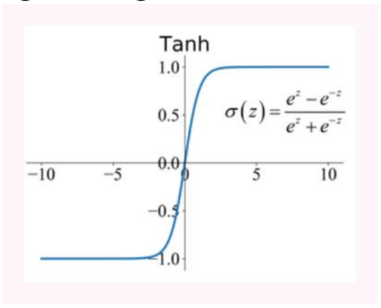
Linear $f(x) = x$

Sigmoid Activation Function: This function takes any real value as input and outputs values in the range of 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0,

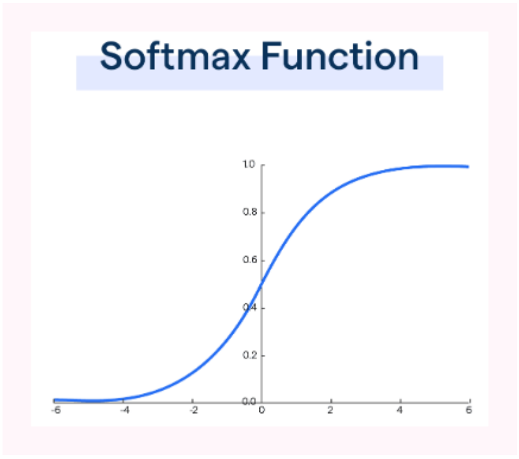
Sigmoid / Logistic → $f(x) = 1/(1 + e^{-x})$

Tanh Activation Function: Tanh function is very similar to the sigmoid/logistic activation function, and even has the same S-shape with the difference in output range of -1 to 1. In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$



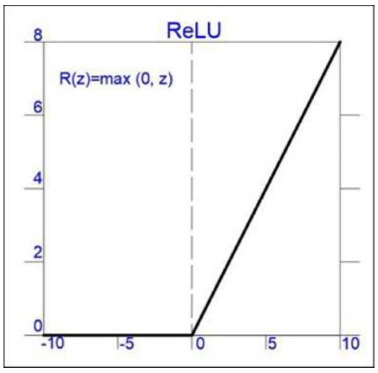
Softmax Activation Function: Softmax functions are often written as a combination of multiple sigmoid. We know that Sigmoid returns a value between 0 and 1. This can be treated as the probability of a data point belonging to a particular class. Therefore, sigmoid are often used for binary classification problems. The softmax function can be used for multiclass classification problems. This function returns the probability of a data point belonging to each unique class. Here is the formula for the same –



$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

σ	= softmax
$\vec{z}$	= input vector
e^{z_i}	= standard exponential function for input vector
K	= number of classes in the multi-class classifier
e^{z_j}	= standard exponential function for output vector
e^{z_j}	= standard exponential function for output vector

ReLU Function: ReLU stands for Rectified Linear Unit. ReLU has a derivative function and allows for backpropagation while simultaneously making it computationally efficient. The main catch here is that the ReLU function does not activate all the neurons at the same time. The neurons will only be deactivated if the output of the linear transformation is less than 0.



f(x) = max(0, x)

Applications:

1. Activation function decides whether a neuron should be activated or not.
2. The role of the Activation Function is to derive output from a set of input values fed to a node (or a layer).

Input: Array

e.g. np.array([-1, 0, 1])

Output: Activation Function

```
print(sigmoid(x)) # [0.26894142 0.5 0.73105858]
print(tanh(x)) # [-0.76159416 0. 0.76159416]
print(relu(x)) # [0 0 1]
print(softmax(x)) # [0.09003057 0.24472847 0.66524096]
```

Conclusion:

We have successfully implemented program to plot a few activation functions that are being used in neural networks.

Outcome:

Upon completion of this experiment, students will be able to:

Experiment level outcome (ELO1): Perform activation function on element-wise operations on arrays.

Questions:

1. What is Activation Function?
2. What are the three activation functions?
3. Why do we use the ReLU activation function**?**
4. What are activation functions and their examples?
5. What is the use of Activation Function?