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

Software for Python: Jupyter Notebook

Theory:

An activation function is a mathematical function that controls the output of a neural network. Activation functions help in determining whether a neuron is to be fired or not as shown in Fig.1.

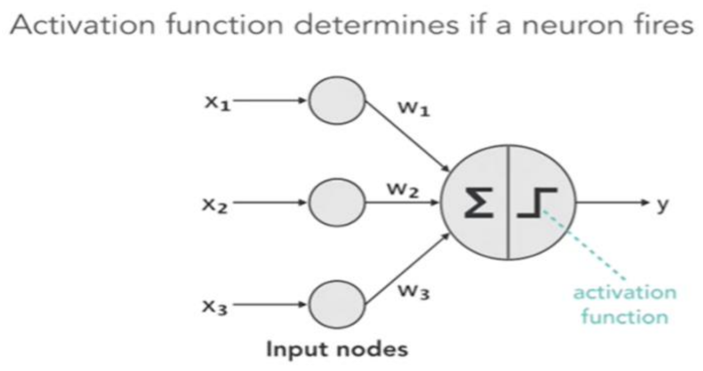


Fig.1.Activation function

Activation is responsible for adding non-linearity to the output of a neural network model. Without an activation function, a neural network is simply a linear regression.

The mathematical equation for calculating the output of a neural network is:

$$Y = \text{Activation}(\Sigma(weight * input) + bias)$$

Some of the popular activation functions are:

1. Linear
2. Sigmoid
3. ReLU
4. Tanh
5. Softmax
6. Leaky ReLU

1.Linear activation function-Linear functions are simple. It returns what it gets as input.

It is defined as f(x)=x, for all x

Here input=output

2.Sigmoid activation function-Sigmoid function returns the value between 0 and 1.

For activation function in deep learning network, Sigmoid function is considered not good since near the boundaries the network doesn't learn quickly.

This is because gradient is almost zero near the boundaries.

Sigmoid functions are so-called because their graphs are “S-shaped”.

3.RELU activation function-RELU is more well known activation function which is used in the deep learning networks. RELU is less computational expensive than the other non-linear activation functions.

RELU returns 0 if the x (input) is less than 0

RELU returns x if the x (input) is greater than 0

4.Tanh activation function-Tanh is another nonlinear activation function.

Tanh outputs between -1 and 1.

Tanh also suffers from gradient problem near the boundaries just as Sigmoid activation function does.

$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

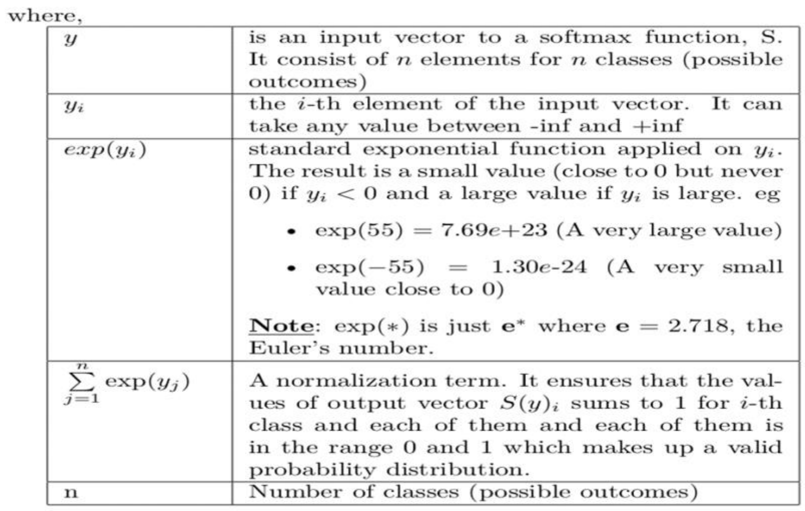
5.Softmax activation function: Softmax is an activation function that scales numbers/logits into probabilities.

The output of a Softmax is a vector (say v) with probabilities of each possible outcome.

The probabilities in vector v sums to one for all possible outcomes or classes.

Mathematically, Softmax is defined as,

$$S(y)_i = \frac{\exp(y_i)}{\sum\limits_{j=1}^{n} \exp(y_j)}$$



where,

y	is an input vector to a softmax function, S. It consist of n elements for n classes (possible outcomes)
y_i	the i-th element of the input vector. It can take any value between -inf and +inf
$exp(y_i)$	standard exponential function applied on y_i. The result is a small value (close to 0 but never 0) if $y_i < 0$ and a large value if y_i is large. eg • $\exp(55) = 7.69e{+}23$ (A very large value) • $\exp(-55) = 1.30e{-}24$ (A very small value close to 0) **Note**: $\exp(*)$ is just $\mathbf{e}^{*}$ where $\mathbf{e} = 2.718$, the Euler's number.
$\sum\limits_{j=1}^{n} \exp(y_j)$	A normalization term. It ensures that the values of output vector $S(y)_i$ sums to 1 for i-th class and each of them and each of them is in the range 0 and 1 which makes up a valid probability distribution.
n	Number of classes (possible outcomes)

6.Leaky relu activation function-The Leaky ReLu function is an improvisation of the regular ReLu function.

To address the problem of zero gradient for negative value, Leaky ReLu gives an extremely small linear component of x to negative inputs.

Mathematically we can express Leaky ReLu as:

f(x)= 0.01x, x<0

= x, x>=0

Mathematically:

f(x)=1 (x<0)

(αx)+1 (x>=0)(x)

Here a is a small constant like the 0.01 we've taken above.

Procedure:

1. First import the libraries matplotlib.pyplot as plt and numpy as np.

   For Linear Activation function
2. Define the linear function as linear(x) and return x.
3. Define the x in the range of -10 to +10
4. Set y equal to linear (x)
5. Plot x and y
6. Label x-axis as Input and label y-axis as Output
7. Give the title as Linear Activation function.
8. Plot the grid function with the command plt.grid()
9. Display the graph of Linear activation function with the command plt.show().

   For Sigmoid Activation Function
10. Define the sigmoid function as sigmoid(x) and return 1/(1+np.exp(-x)).
11. Define the x in the range of -10 to +10
12. Set y equal to sigmoid (x)
13. Plot x and y
14. Label x-axis as Input and Label y-axis as Output
15. Give the title as Sigmoid Activation function.
16. Plot the grid function with the command plt.grid()
17. Display the graph of Sigmoid activation function with the command plt.show().

   For RELU activation function

18.Define the RELU function as relu(x) and return np.maximum(0,x).
19.Define the x in the range of -10 to +10

20.Set y equal to relu(x)
21.Plot x and y
22.Label x-axis as Input and label y-axis as Output
23.Give the title as ReLU Activation function.
24.Plot the grid function with the command plt.grid()
25.Display the graph of ReLU activation function with the command plt.show().

For Hyperbolic Tangent function(tanh)

18.Define the hyperbolic tangent function as tanh(x) and return np.tanh(x)
19.Define the x in the range of -10 to +10
20.Set y equal to tanh(x)
21.Plot x and y
22.Label x-axis as Input and Label y-axis as Output
23.Give the title as Tanh Activation function.
24.Plot the grid function with the command plt.grid()
25.Display the graph of tanh activation function with the command plt.show().

For Softmax Activation function

26.Define the softmax function as softmax(x) and return np.exp(x)/np.sum(np.exp(x), axis=0)
27.Define the x in the range of -10 to +10
28.Set y equal to softmax(x)
29.Plot x and y
30.Label x-axis as Input and Label y-axis as Output
31.Give the title as Softmax Activation function.
32.Plot the grid function with the command plt.grid()
33.Display the graph of softmax activation function with the command plt.show().

For Leaky Relu Activation function

34.Define the leaky relu function as leaky_relu(x.alpha=0.01) and return np.maxima(alpha*x,x)
35.Define the x in the range of -10 to +10
36.Set y equal to leaky_relu(x)
37.Plot x and y
38.Label x-axis as Input and Label y-axis as Output
39.Give the title as Leaky ReLU Activation function.
40.Plot the grid function with the command plt.grid()
41.Display the graph of Leaky ReLU activation function with the command plt.show().

Conclusion: Thus, the various activation functions that are being used in neural network are studied.

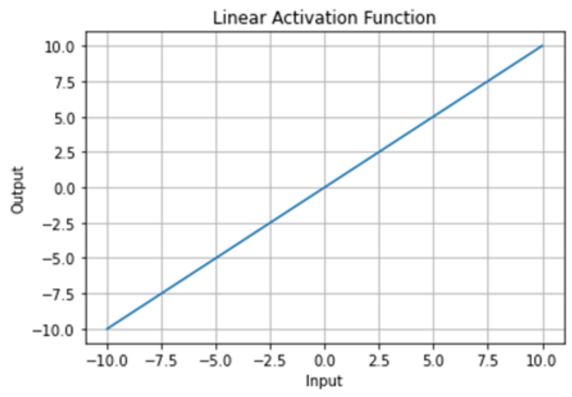
Program Code and Output

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np

        def linear(x):
          return x

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

        plt.plot(x, y)
        plt.xlabel("Input")
        plt.ylabel("Output")
        plt.title("Linear Activation Function")
        plt.grid()
        plt.show()
```

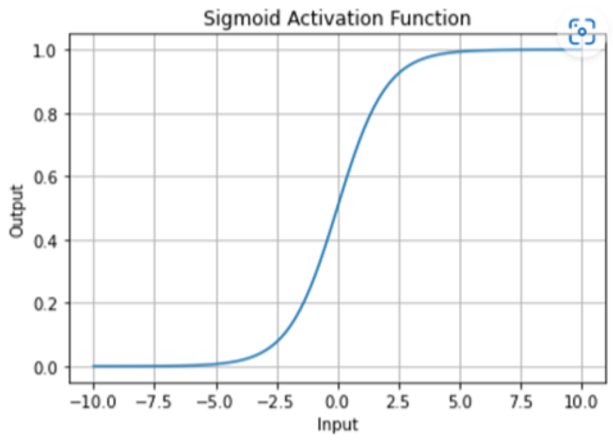


```
In [2]: import matplotlib.pyplot as plt
        import numpy as np

        def sigmoid(x):
          return 1 / (1 + np.exp(-x))

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

        plt.plot(x, y)
        plt.xlabel("Input")
        plt.ylabel("Output")
        plt.title("Sigmoid Activation Function")
        plt.grid()
        plt.show()
```

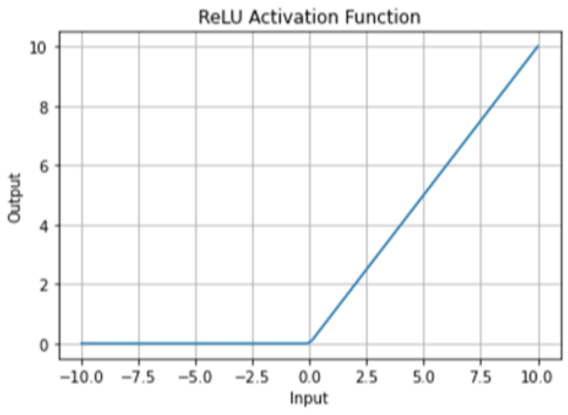


```
In [3]: import matplotlib.pyplot as plt
        import numpy as np

        def relu(x):
          return np.maximum(0, x)

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

        plt.plot(x, y)
        plt.xlabel("Input")
        plt.ylabel("Output")
        plt.title("ReLU Activation Function")
        plt.grid()
        plt.show()
```

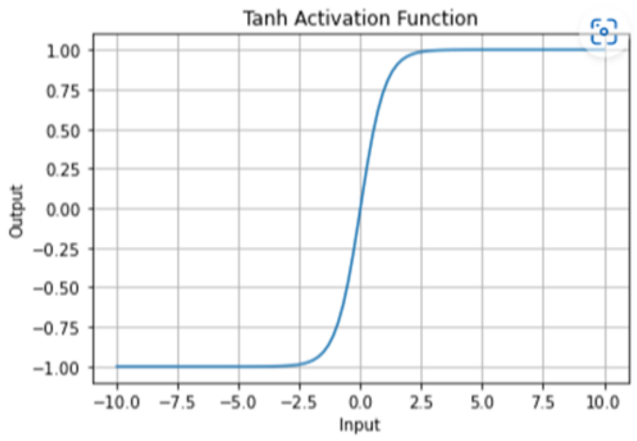


```
In [4]: import matplotlib.pyplot as plt
        import numpy as np

        def tanh(x):
          return np.tanh(x)

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

        plt.plot(x, y)
        plt.xlabel("Input")
        plt.ylabel("Output")
        plt.title("Tanh Activation Function")
        plt.grid()
        plt.show()
```

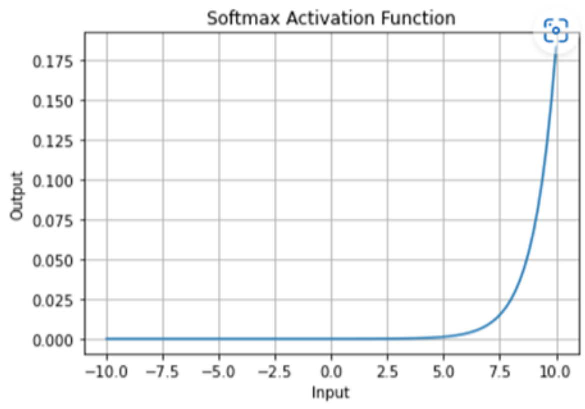


In [5]:
```python
import matplotlib.pyplot as plt
import numpy as np

def softmax(x):
  return np.exp(x) / np.sum(np.exp(x), axis=0)

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

plt.plot(x, y)
plt.xlabel("Input")
plt.ylabel("Output")
plt.title("Softmax Activation Function")
plt.grid()
plt.show()
```

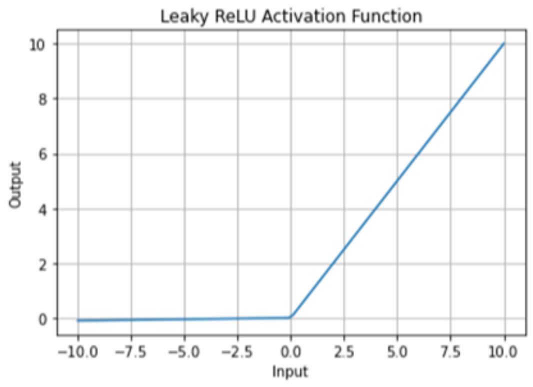


In [6]:
```python
import matplotlib.pyplot as plt
import numpy as np

def leaky_relu(x, alpha=0.01):
  return np.maximum(alpha * x, x)

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

plt.plot(x, y)
plt.xlabel("Input")
plt.ylabel("Output")
plt.title("Leaky ReLU Activation Function")
plt.grid()
plt.show()
```


Leaky ReLU Activation Function
Output
Input