EXPERIMENT 1

ACTIVATION FUNCTIONS

Dikshita Kambri

118A2044

BE EXTC A

A2

EXPERIMENT 1

ACTIVATION FUNCTIONS

AIM: Write a program to generate the following activation functions:

a) Identity function
b) Binary Step function
c) Bipolar Step Function
d) Binary Sigmoidal Function
e) Bipolar Sigmoidal function
f) Ramp function
And evaluate the value of the activation function at input value of a)zero and b)0.5

APPARATUS: PC with Python software.

THEORY:

Neural networks are a biologically inspired form of distributed computation. The basic structure of neural networks consists of a number of interconnected processing units, called *neurons*. There are a large number of such interconnected nodes that perform summation and thresholding, in loose analogy with the neurons of the brain.

A system of simple processing elements, neurons, that are connected into a network by a set of (synaptic) weights.

The Neuron

The *neuron* or *node* or *unit*, as it is also called, is a processing element that takes a number of inputs, weights them, sums them up, and uses the result as the argument for a singular valued function, the *activation function*.

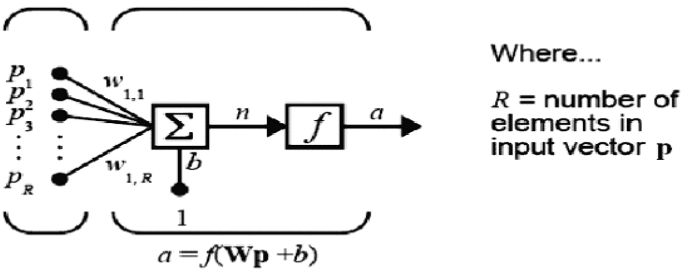


Fig. 1 Structure of a multiple input neuron

The basic structure of a multi-input neuron with "*R*" inputs is shown in Fig. 1. Let p_1, p_2, … , p_R be the inputs of the system, $w_{1,1}$, $w_{1,2}$…$w_{1,R}$ be the corresponding weights and "*b*" the bias. Let *f* be the activation function. (The indices in $w_{1,R}$ indicates that this weight represents the connection *to* the first (and only) neuron *from* the R_{th} source. This convention is useful if there is more than one neuron. The weighted input *n* to the net is given by

$$n = \sum_{i=1}^{R} w_{1,i} p_i + b$$

This expression can be written in matrix form:

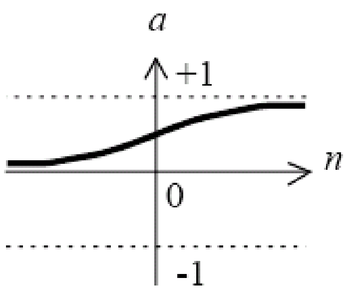
$$n = \mathbf{W}\mathbf{p} + b$$

where the matrix **W** for the single neuron case has only one row.

Now the processing done by a neuron is described by the output of neuron as:

$$a = f(\mathbf{Wp} + b)$$

Essentially, *f* can take any form but most often it is monotonic. The sigmoid function is by far the most frequently used in neural networks. It is strictly increasing function that exhibits smoothness and has the desired asymptotic properties. The important feature of these functions is that they are differentiable and this helps in the justification of derivation of gradient descent algorithm for the training of Neural Networks. It takes the input, which may have any value between plus and minus infinity, and squashes the output into the range 0 to 1. The standard sigmoid function is the *logistic* function is shown in Fig. 2.



$$a = \frac{1}{1+e^{-n}}$$

$a = logsig(n)$

Fig. 2 Sigmoid transfer function

Activation functions:

a) Identity function

$$f(x) = x \quad \text{for all } x$$

b) Binary Step function

$$f(x) = \begin{cases} 1 & \text{if } x \geqslant \theta \\ 0 & \text{if } x < \theta \end{cases}$$

c) Bipolar Step Function

$$f(x) = \begin{cases} 1 & \text{if } x \geqslant \theta \\ -1 & \text{if } x < \theta \end{cases}$$

d) Binary Sigmoidal Function

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

e) Bipolar Sigmoidal function

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

f) Ramp function

$$f(x) = \begin{cases} 1 & \text{if } x > 1 \\ x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{if } x < 0 \end{cases}$$

CONCLUSION:

We performed various activation functions such as Identity function, Binary step function, Bipolar step function, Binary sigmoidal function, bipolar sigmoidal function with the help of google Colab within the range -5 to +5.

Imported required libraries

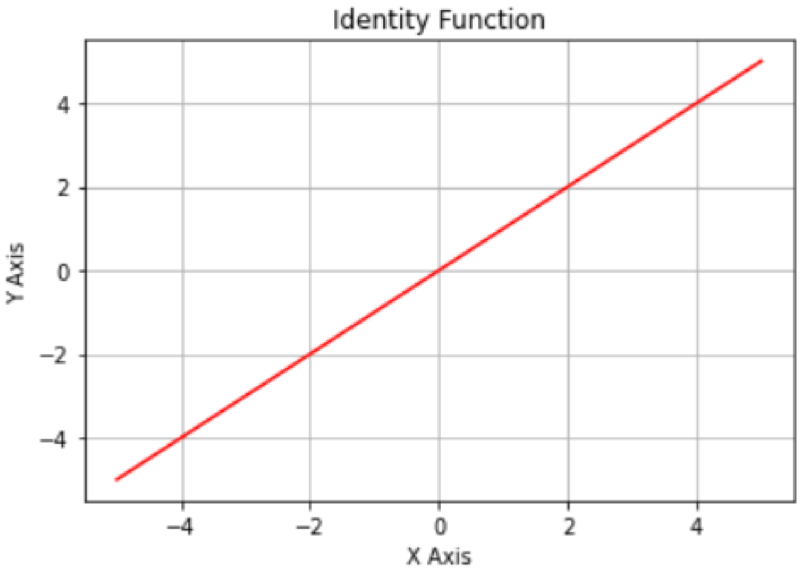
```
import numpy as np
import matplotlib.pyplot as plt
```

Defined activation function

```
[ ] def identity(x):
        return x

    def bin_step(x):
        return 1 * (x >= 2)

    def bipolar_step(x):
      return 1 * (x >= 0) - 1 * (x < 0)

    def bin_sigmoidal(x,l):
        return 1 / (1 + np.exp(-l*x))

    def bip_sigmoidal(x,l):
        return (1 - np.exp(-l*x)) / (1 + np.exp(-l*x))
```

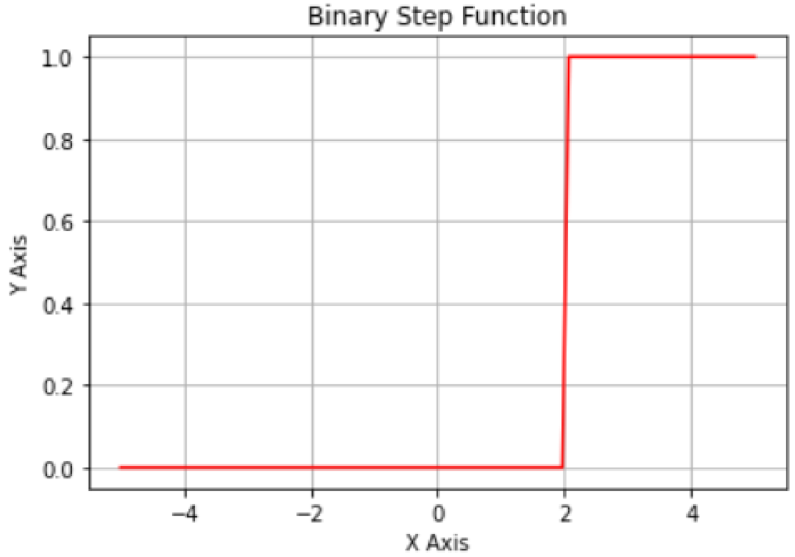
Plotted Identity function

```
[ ]  X = np.linspace(-5, 5, 100)
     plt.plot(X, identity(X),'r')
     plt.xlabel('X Axis')
     plt.ylabel('Y Axis')
     plt.title('Identity Function')
     plt.grid()
     plt.show()
```



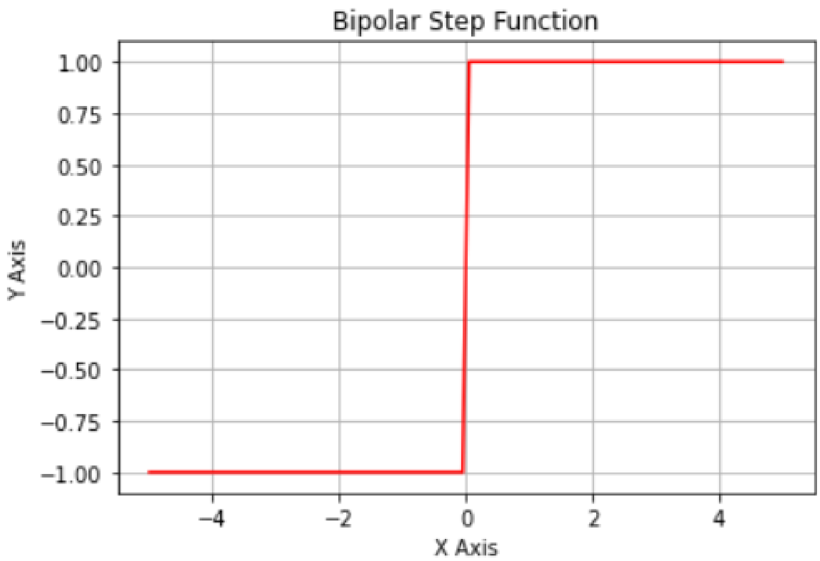
Plotted Binary step function

```
[ ]  plt.plot(X, bin_step(X),'r')
     plt.xlabel('X Axis')
     plt.ylabel('Y Axis')
     plt.title('Binary Step Function')
     plt.grid()
     plt.show()
```



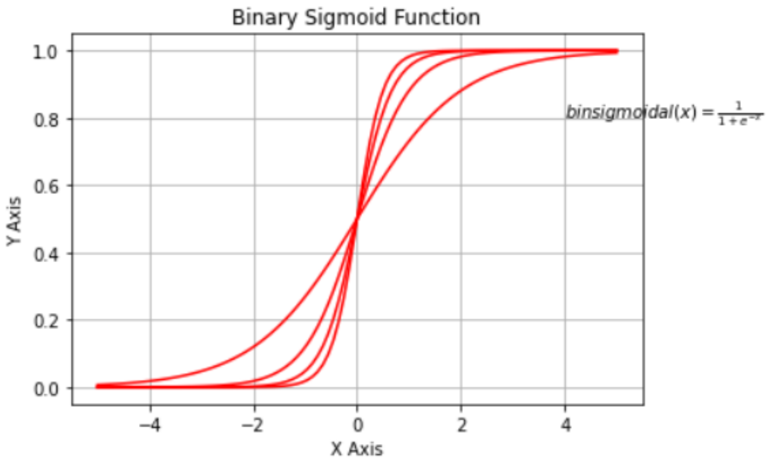
Plotted Bipolar step function

```
[ ]  plt.plot(X, bipolar_step(X),'r')
     plt.xlabel('X Axis')
     plt.ylabel('Y Axis')
     plt.title('Bipolar Step Function')
     plt.grid()
     plt.show()
```



Plotted Binary sigmoid function

```
[ ]  for l in range(1, 5):
       plt.plot(X, bin_sigmoidal(X,l),'r')
       plt.xlabel('X Axis')
       plt.ylabel('Y Axis')
       plt.title('Binary Sigmoid Function')
     plt.grid()
     plt.text(4, 0.8, r'$binsigmoidal(x)=\frac{1}{1+e^{-x}}$', fontsize=10)
     plt.show()
```



```
[ ]  for l in range(1, 5):
       plt.plot(X, bip_sigmoidal(X,l),'r')
       plt.xlabel('X Axis')
       plt.ylabel('Y Axis')
       plt.title('Bipolar Sigmoid Function')
     plt.grid()
     plt.text(4, 0.8, r'$binsigmoidal(x)=\frac{1-e^{-x}}{1+e^{-x}}$', fontsize=10)
     plt.show()
```