Batch: A2 **Roll No.: 16010121045**

Experiment No. 01

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Activation functions.

Objective: To implement activation functions of Neural Network.

Expected Outcome of Experiment:

CO1 : Identify and describe soft computing techniques and their roles

Books/ Journals/ Websites referred:

- J.S.R.Jang, C.T.Sun and E.Mizutani, "Neuro-Fuzzy and Soft Computing", PHI, 2004, Pearson Education 2004.
- Davis E.Goldberg, "Genetic Algorithms: Search, Optimization and Machine Learning", Addison Wesley, N.Y., 1989.

Pre Lab/ Prior Concepts:

Neural networks, sometimes referred to as connectionist models, are parallel-distributed models that have several distinguishing features-

1) A set of processing units;
2) An activation state for each unit, which is equivalent to the output of the unit;
3) Connections between the units. Generally each connection is defined by a weight w_{jk} that determines the effect that the signal of unit j has on unit k;
4) A propagation rule, which determines the effective input of the unit from its external inputs;
5) An activation function, which determines the new level of activation based on the effective input and the current activation;
6) An external input (bias, offset) for each unit;
7) A method for information gathering (learning rule);

8) An environment within which the system can operate, provide input signals and, if necessary, error signals.

Implementation Details:

Most units in neural network transform their net inputs by using a scalar-to-scalar function called an *activation function*, yielding a value called the unit's activation. Except possibly for output units, the activation value is fed to one or more other units. Activation functions with a bounded range are often called squashing functions. Some of the most commonly used activation functions are :

1) Identity function

Identity or Linear activation function is the simplest activation function of all. It applies identity operation on your data and output data is proportional to the input data.

2) Binary step function:

Binary step function is a threshold-based activation function which means after a certain threshold neuron is activated and below the said threshold neuron is deactivated.

3) Sigmoid function:

The sigmoid activation function is **also called the logistic function**. It is the same function used in

the logistic regression classification algorithm. The function takes any real value as input and outputs values in the range 0 to 1.

4) **Bipolar sigmoid function**:

The Bipolar activation function **used to convert the activation level of a unit (neuron) into an output signal**. It is also known as transfer function or squashing function due to the capability to squeeze the amplitude range of output signal to some finite value.

5) **Ramp function**:

The rectified linear activation function or ReLU is a **non-linear** function or **piecewise linear** function that will output the input directly if it is positive, otherwise, it will output zero.

Code:

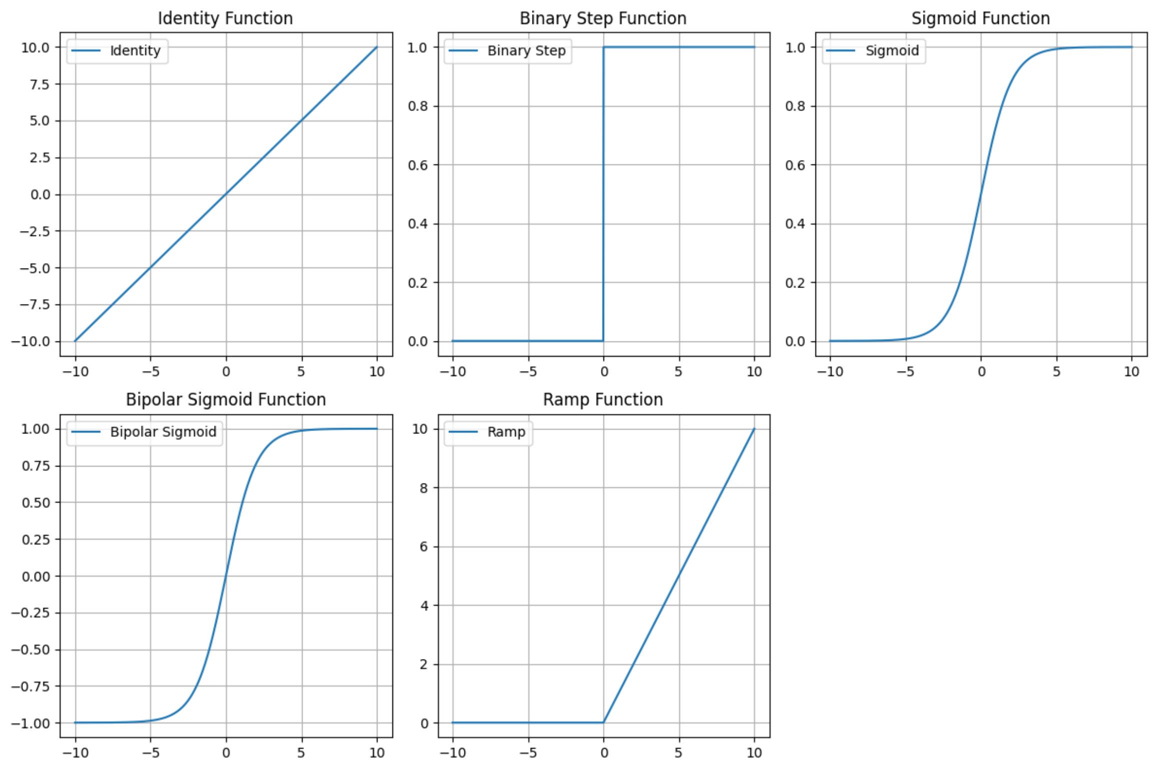
```
import numpy as np
import matplotlib.pyplot as plt

# Activation functions
def identity(x):
    return x

def binary_step(x):
    return 0 if x < 0 else 1

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

def bipolar_sigmoid(x):
    return (2 / (1 + np.exp(-x))) - 1

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

# Range of x values

n= abs(int(input("Enter the range variable: ")))
x = np.linspace(-n, n, 1000)

# Calculate activation values for each function
identity_values = identity(x)
binary_step_values = [binary_step(val) for val in x]
sigmoid_values = sigmoid(x)
bipolar_sigmoid_values = bipolar_sigmoid(x)
ramp_values = ramp(x)

# Plot the activation functions
plt.figure(figsize=(12, 8))
```

```python
plt.subplot(2, 3, 1)
plt.plot(x, identity_values, label='Identity')
plt.title('Identity Function')
plt.grid()
plt.legend()

plt.subplot(2, 3, 2)
plt.plot(x, binary_step_values, label='Binary Step')
plt.title('Binary Step Function')
plt.grid()
plt.legend()

plt.subplot(2, 3, 3)
plt.plot(x, sigmoid_values, label='Sigmoid')
plt.title('Sigmoid Function')
plt.grid()
plt.legend()

plt.subplot(2, 3, 4)
plt.plot(x, bipolar_sigmoid_values, label='Bipolar Sigmoid')
plt.title('Bipolar Sigmoid Function')
plt.grid()
plt.legend()

plt.subplot(2, 3, 5)
plt.plot(x, ramp_values, label='Ramp')
plt.title('Ramp Function')
plt.grid()
plt.legend()

plt.tight_layout()
plt.show()
```

Output:

```
❯ python3 -u "/Users/pargatsinghdhanjal/Desktop/Soft Computing/exp1.py"
Enter the range variable: 10
2023-08-03 14:48:02.276 Python[98457:2932811] +[CATransaction synchronize] called within transaction
```

Mathplot



Conclusion: Thus, we have successfully implemented 4 Activation Functions of Neural Network.

Post Lab Descriptive Questions :

1. **Explain the concept behind using Activation function.**
Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.
We know, neural network has neurons that work in correspondence of weight, bias and their

respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.
A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks. The most important feature in an activation function is its ability to add non-linearity into a neural network.

2. Explain the different properties of activation functions.

Different types of ANN are:

- *Single layer feed forward network*
- *Multilayer feed-forward network*
- *Single node with its own feedback*
- *Single-layer recurrent network*
- *Multilayer recurrent network*

Single layer feed forward network

- Layer is formed by taking processing elements and combining it with other processing elements.
- Input and output are linked with each other
- Inputs are connected to the processing nodes with various weights, resulting in series of outputs one per node.

Multilayer feed-forward network

- Formed by the interconnection of several layers.
- Input layer receives input and buffers input signal.
- Output layer generated output.
- Layer between input and output is called *hidden layer.*
- Hidden layer is internal to the network.
- Zero to several hidden layers in a network.
- More the hidden layer, more is the complexity of network, but efficient output is produced.

Feedback network

- If no neuron in the output layer is an input to a node in the same layer / proceeding layer –

  feed forward network.

- If outputs are directed back as input to the processing elements in the same layer/proceeding layer –***feedback network.***
- If the output are directed back to the input of the same layer then it is ***lateral feedback.***
- ***Recurrent networks*** are networks with feedback networks with closed loop.
- Fig 2.8 (A) –simple recurrent neural network having a single neuron with feedback to itself.

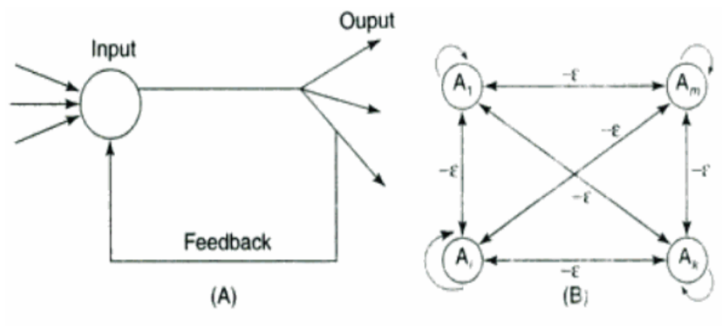


Figure 2-8 (A) Single node with own feedback. (B) Competitive nets.

- Fig 2.9 – single layer network with feedback from output can be directed to processing element itself or to other processing element/both.
- Processing element output can be directed back to the nodes in the preceding layer, forming a ***multilayer recurrent network.***
- Processing element output can be directed to processing element itself or to other processing element in the same layer.

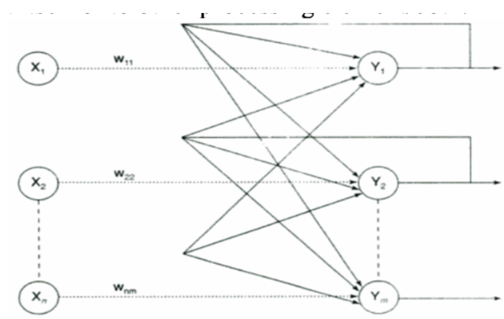


Figure 2-9 Single-layer recurrent network.

Date: _____________ **Signature of faculty in-charge**