|  |  |
| --- | --- |
| **Name:** | Bodhisatya Ghosh |
| **Branch:** | CSE – Data Science |
| **Batch:** | B |
| **Course:** | Soft Computing |
| **Experiment no:** | 1 |

**Aim:** To implement activation functions

**Theory:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Graph** | **Formula** | **Significance** | **Description** | **Usage** |
| Identity | Figure B.7: Plot of the Linear / Identity activation function. For a... |  Download Scientific Diagram | f(x) = x | A linear function | The identity function is a special case of an activation function where the output signal is equal to the input signal. In other words, the identity function simply passes the input signal through unchanged. | Used when there is no need for any activation function but consistency is required across all layers |
| Binary Step | Getting to know Activation Functions in Neural Networks. | by Hasara Samson  | Towards Data Science | f(x) = 1,x>=0  = 0, x<0 | Most common activation function in neural networks | Binary step function is one of the simplest activation functions. The function produces binary output and thus the name binary step funtion. The function produces 1 (or true) when input passes a threshold limit whereas it produces 0 (or false) when input does not pass threshold. | Used in single-layer nets to convert the net input to an output that is a binary (1 or 0) |
| Bipolar Step | Activation Functions – Machine Learning Geek |  |  | In the **Bipolar Step Function**, if the value of Y is above a certain value known as the threshold, the output is +1 and if it’s less than the threshold then the output is -1. | It has bipolar outputs (+1 to -1). It can be utilized in single-layer networks. |
| Binary Sigmoid | Binary sigmoid activation function The limited numeric response range,... |  Download Scientific Diagram | f(x) = 1/(1+e^(-x)) | It is differentiable, non-linear, and produces non-binary activations But the problem with Sigmoid is the vanishing gradients. | Binary Sigmoid Function or **Sigmoid function** is a logistic function where the output values are either binary or vary from 0 to 1. | The sigmoid function extracts a bounded absolute value from the model's output. Can be used in logistic problems. |
| Bipolar Sigmoid | 3: Bipolar sigmoid function. | Download Scientific Diagram |  |  | This is the function from where the Hyperbolic Tan Function was derived from. Here (lambda) represents the steepness factor. The range of this function is between -1 and 1. For the hyperbolic tangent function, the value of the steepness factor is 2. | If the network uses the binary data, then it is better to convert it to bipolar form and use the bipolar sigmoidal activation function or hyperbolic tangent function. |
| Ramp | Ramp function - Wikipedia |  |  | 0 for negative inputs, output equals input for non-negative inputs |  |
| ReLu | 6: Graph of ReLu activation function | Download Scientific Diagram | f(x) = max(0,x) |  | It is a piecewise linear function that is defined to be 0 for all negative values of x and equal to a × x otherwise, where a is a learnable parameter. |  |

**Program:**

from math import exp as e

import numpy as np

def identity(x):

    return x

def binary\_step(x):

    if(x>=0):

        return 1

    return 0

def bipolar\_step(x):

    if(x>=0):

        return 1

    return -1

def binary\_sigmoid(x):

    val = 1/(1+(e(x\*-1)))

    return val

def bipolar\_sigmoid(x):

    val = (1-e(x \* -1))/(1+e(x \* -1))

    return val

def ramp(x):

    if(x >= 1):

        return 1

    elif (0 <= x <= 1):

        return x

    return 0

def relu(x):

    return max(0,x)

def main():

    n = int(input("Enter number of input: "))

    x = []

    w = []

    yin = 0

    for i  in range(n):

        xn = float(input("Enter value of x{}: ".format(i+1)))

        wn = float(input("Enter weight of x{}: ".format(i+1)))

        x.append(xn)

        w.append(wn)

    x = np.array(x)

    w = np.array(w)

    for i in range(len(x)):

        yin = yin + (x[i]\*w[i])

    print("Identity: {}".format(identity(yin)))

    print("Binary step: {}".format(binary\_step(yin)))

    print("Bipolar step: {}".format(bipolar\_step(yin)))

    print("Binary sigmoid: {}".format(binary\_sigmoid(yin)))

    print("Bipolar sigmoid: {}".format(bipolar\_sigmoid(yin)))

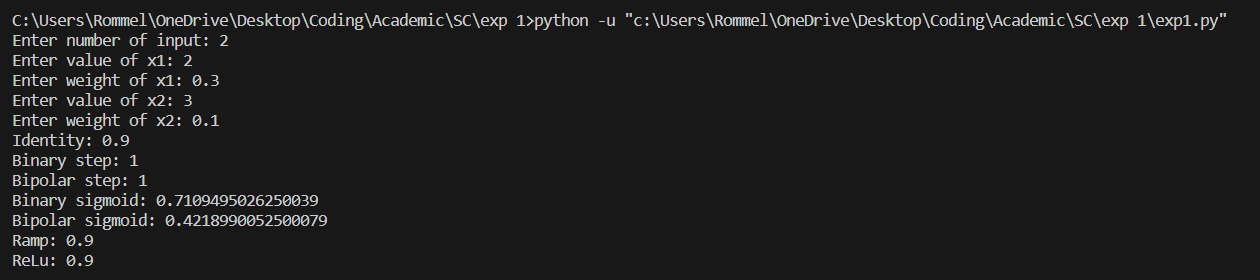
    print("Ramp: {}".format(ramp(yin)))

    print("ReLu: {}".format(relu(yin)))

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**Results:**



**Conclusion:** In this experiment we have learnt how to implement activation functions