Name: Tania Rajabally

Roll No: 43

UID: 2017130047

Batch: C

Course Code: CEL71 (AI and Soft Computing Lab)

Date:19/08/2020

**Experiment No 2**

**To implement Transfer/Activation Functions**

**Aim:** To implement Transfer/Activation Functions.

i)A symmetric hard limit transfer function.

ii)A Binary step activation function.

iii)A Bipolar step activation function.

iv)A saturating linear transfer function.

v)A hyperbolic tangent sigmoid (tansig) transfer function.

vi)A log-sigmoid transfer function

**Theory:**

Neural network activation functions are a crucial component of deep learning. Activation functions determine the output of a deep learning model, its accuracy, and also the computational efficiency of training a model—which can make or break a large scale neural network. Activation functions also have a major effect on the neural network’s ability to converge and the convergence speed, or in some cases, activation functions might prevent neural networks from converging in the first place.

Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated (“fired”) or not, based on whether each neuron’s input is relevant for the model’s prediction. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

An additional aspect of activation functions is that they must be computationally efficient because they are calculated across thousands or even millions of neurons for each data sample. Modern neural networks use a technique called backpropagation to train the model, which places an increased computational strain on the activation function, and its derivative function.

In a neural network, numeric data points, called inputs, are fed into the neurons in the input layer. Each neuron has a weight, and multiplying the input number with the weight gives the output of the neuron, which is transferred to the next layer.

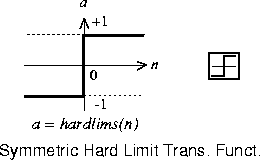
The activation function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer. It can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold. Or it can be a transformation that maps the input signals into output signals that are needed for the neural network to function.

Increasingly, neural networks use non-linear activation functions, which can help the network learn complex data, compute and learn almost any function representing a question, and provide accurate predictions.

**Procedure:**

Symmetric Hard Limit Transfer Function:

The symmetric hard limit transfer function forces a neuron to output a 1 if its net input reaches a threshold. Otherwise it outputs -1. Like the regular hard limit function, this allows a neuron to make a decision or classification. It can say yes or no.



Binary Step Function:

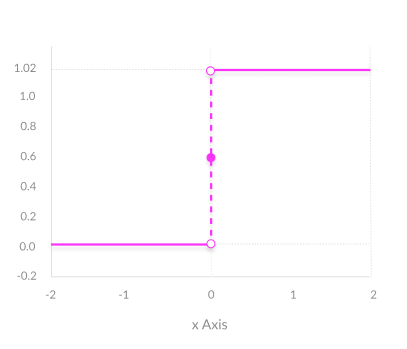
A binary step function is a threshold-based activation function. If the input value is above or below a certain threshold, the neuron is activated and sends exactly the same signal to the next layer.

This function can be defined as,

f(x)={1, if x≥θ

0, if x<θ

where θ represents the threshold value. This function is most widely used in single layer nets to convert the net input to an output that is binary (1 or 0).



The problem with a step function is that it does not allow multi-value outputs—for example, it cannot support classifying the inputs into one of several categories.

Bipolar Step Activation Function:

This function can be defined as,

f(x)={1, if x≥θ

−1, if x<θ

where θ represents the threshold value. This function is also used in a single layer net to convert the net input to an output that is bipolar (+1 or -1).

Saturating Linear Transfer Function:

satlins is a transfer function. Transfer functions calculate a layer's output from its net input.

satlins(N) takes one input,

N - S x Q matrix of net input (column) vectors.

and returns values of N truncated into the interval [-1, 1].

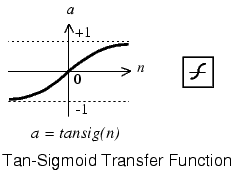
Hyperbolic Tangent Sigmoid Transfer Function:

tansig is a transfer function. Transfer functions calculate a layer's output from its net input.

tansig(N) takes one input,

N -- S x Q matrix of net input (column) vectors

and returns each element of N squashed between -1 and 1.



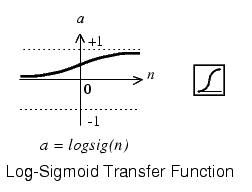
Log Sigmoid Transfer Function:

logsig is a transfer function. Transfer functions calculate a layer's output from its net input.

logsig(N) takes one input,

N -- S x Q matrix of net input (column) vectors

and returns each element of N squashed between 0 and 1.



**Code:**

**import math**

**import numpy as np**

**from matplotlib import pyplot as plt**

**i1 = 0.8**

**i2 = 0.6**

**i3 = 0.4**

**w1 = 0.1**

**w2 = 0.3**

**w3 = -0.2**

**bias = 0.35**

**X = w1\*i1 + w2\*i2 + w3\*i3 + bias**

**def hard\_limit(y):**

**if y>=0:**

**return 1**

**else:**

**return 0**

**Y=hard\_limit(X)**

**print(Y)**

**def bin\_step(x):**

**threshold = 0**

**if(x>=threshold):**

**return(1)**

**else:**

**return(0)**

**Y=bin\_step(X)**

**print(Y)**

**def bip\_step(x):**

**threshold = 0**

**if(x>=threshold):**

**return(1)**

**else:**

**return(-1)**

**Y=bip\_step(X)**

**print(Y)**

**def bin\_sigmoid(x):**

**return 1/(1+np.exp(-x))**

**Y=bin\_sigmoid(X)**

**print(Y)**

**def saturating\_linear(x):**

**if x<=0:**

**return 0**

**elif x>0 and x<1:**

**return x**

**else:**

**return 1**

**Y=saturating\_linear(X)**

**print(Y)**

**def tansig(x):**

**return 2/(1+np.exp(-2\*x))-1**

**Y=tansig(X)**

**print(Y)**

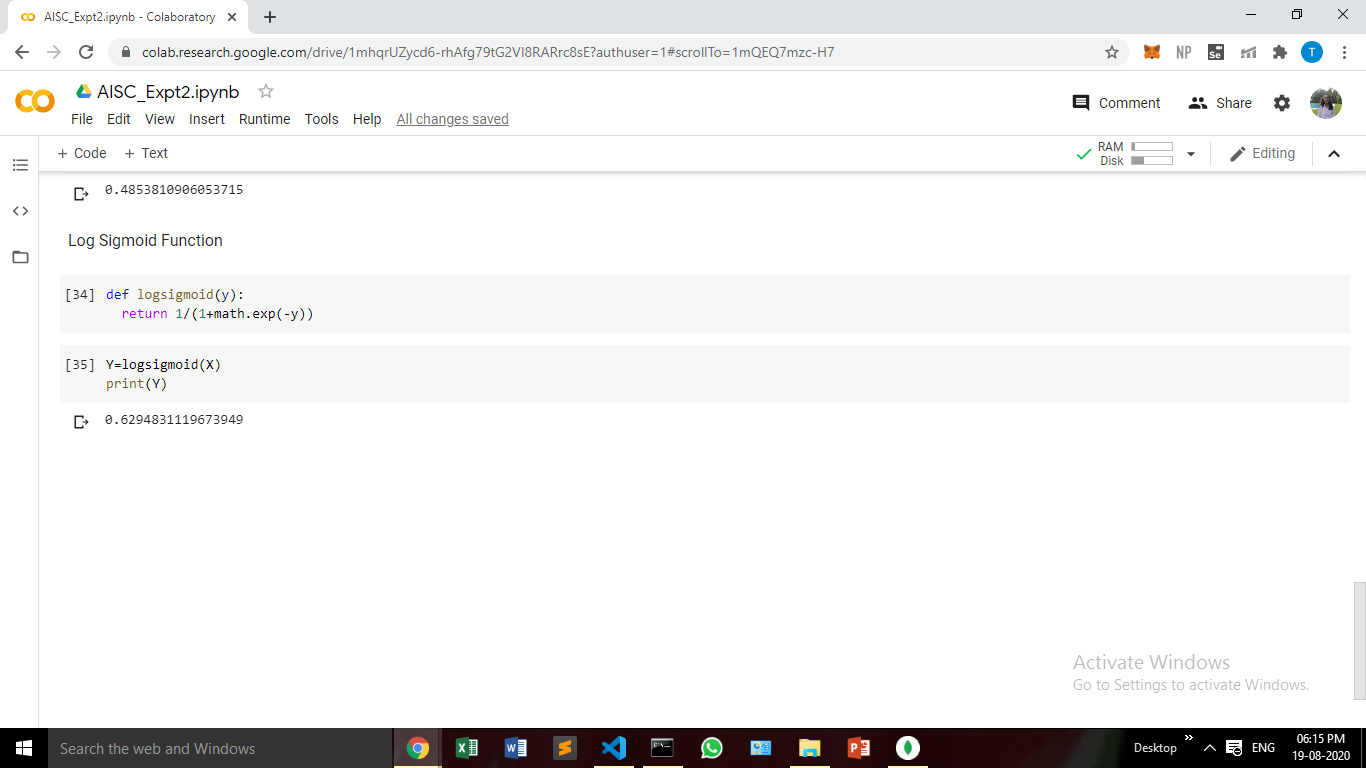
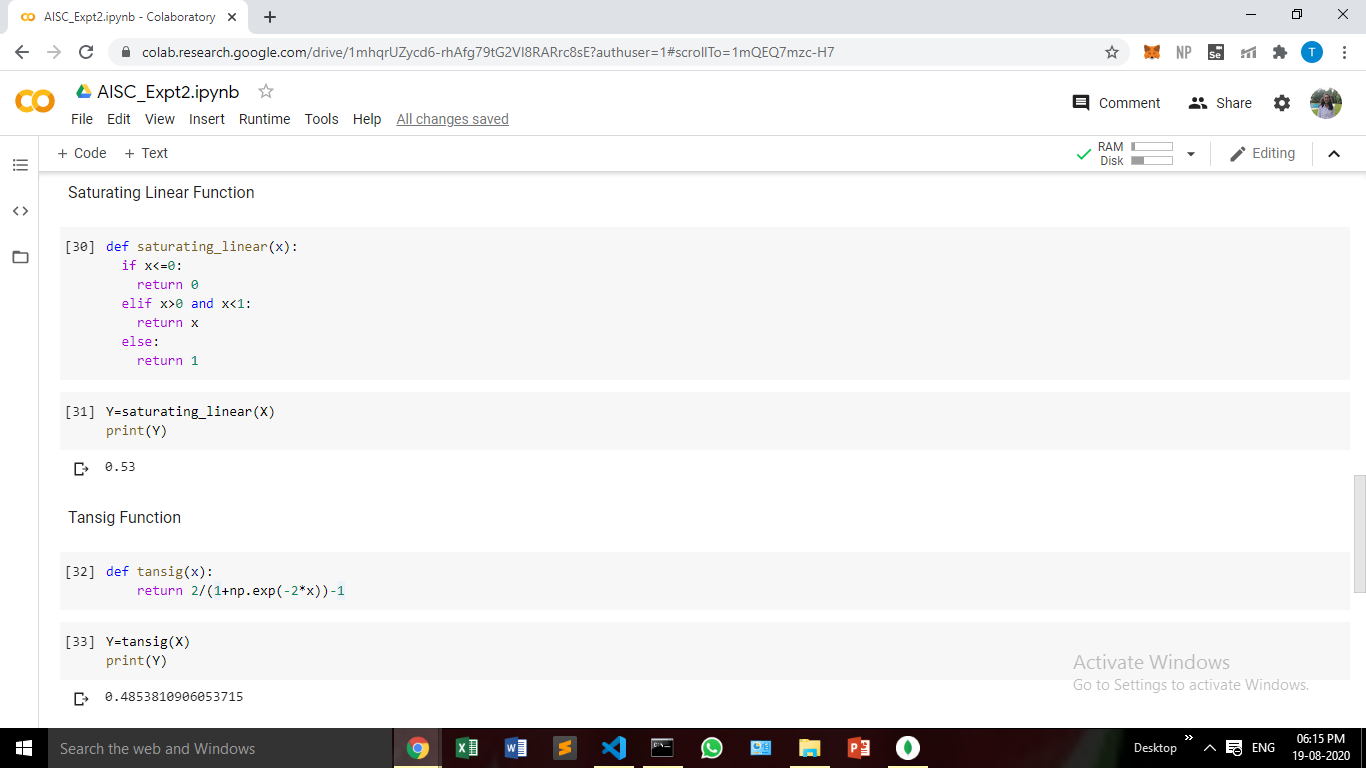
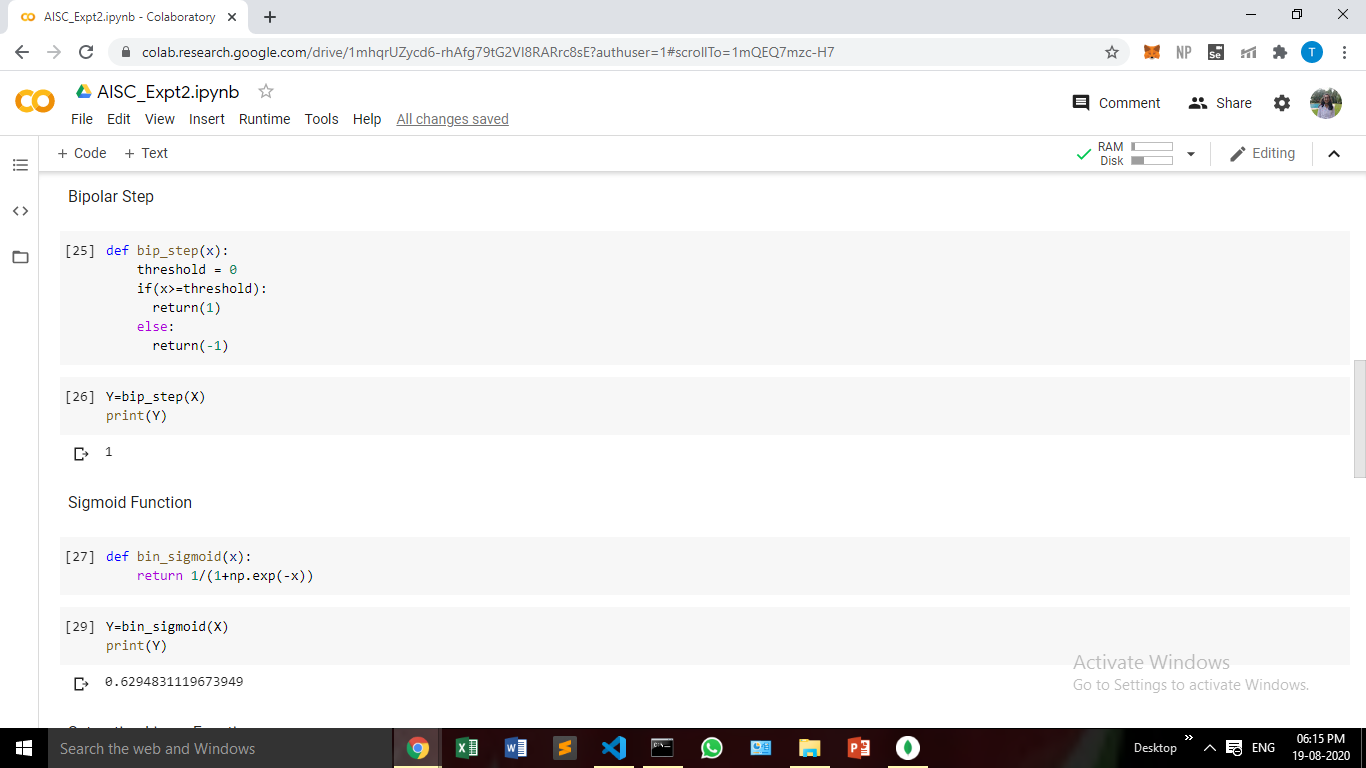
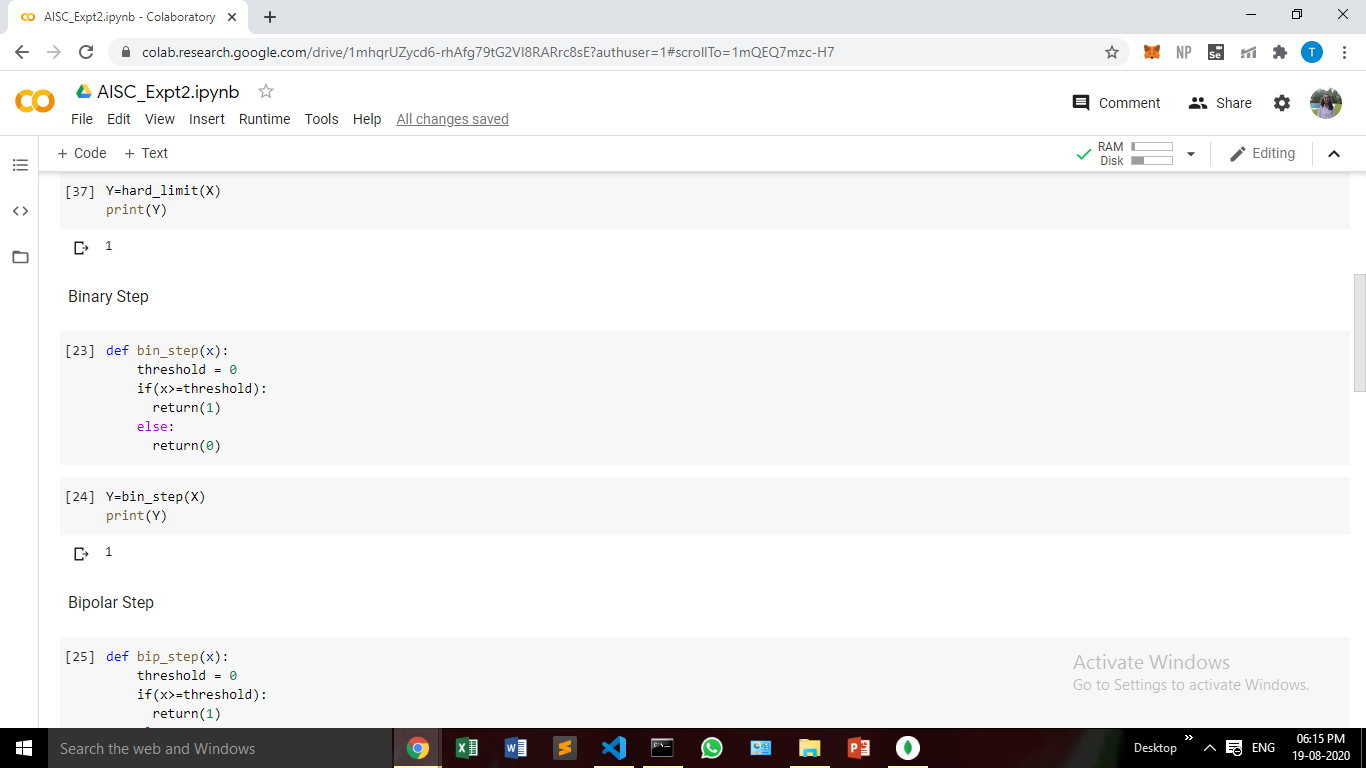
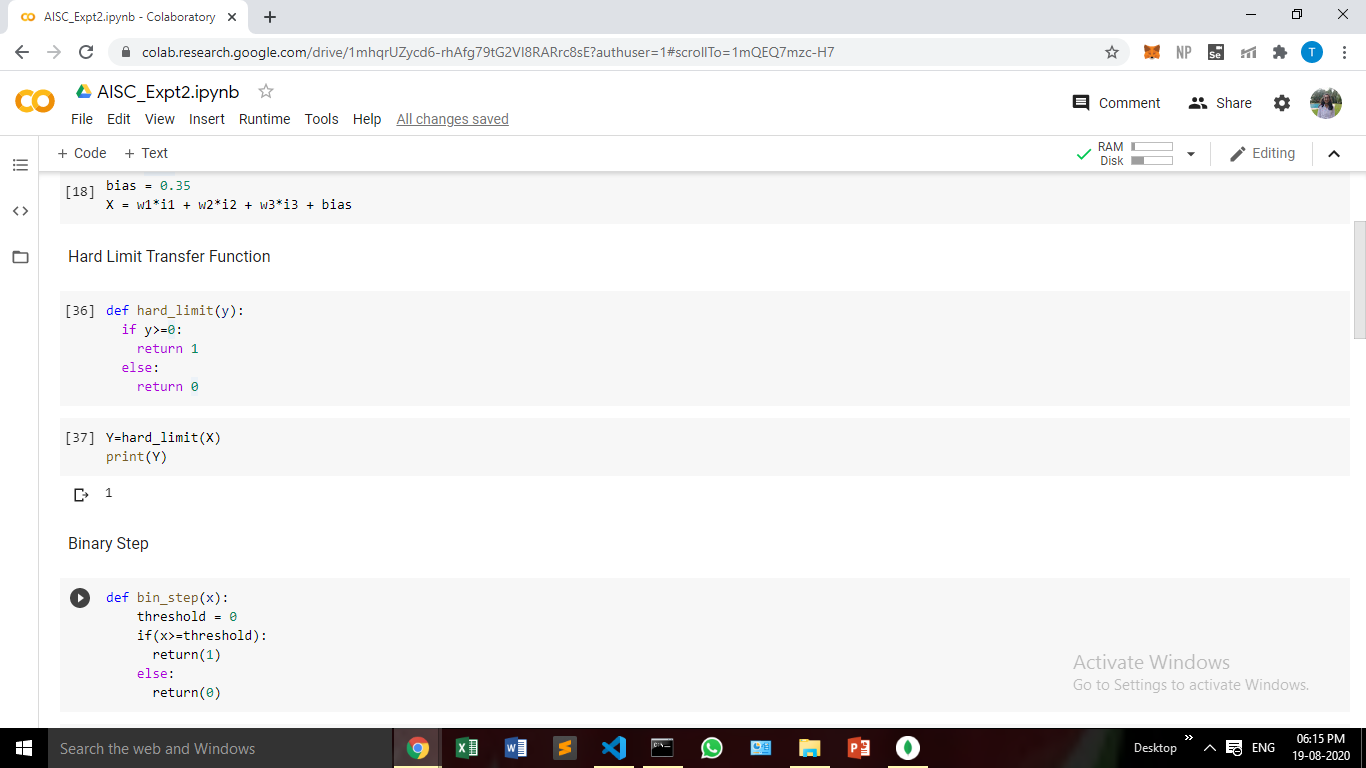
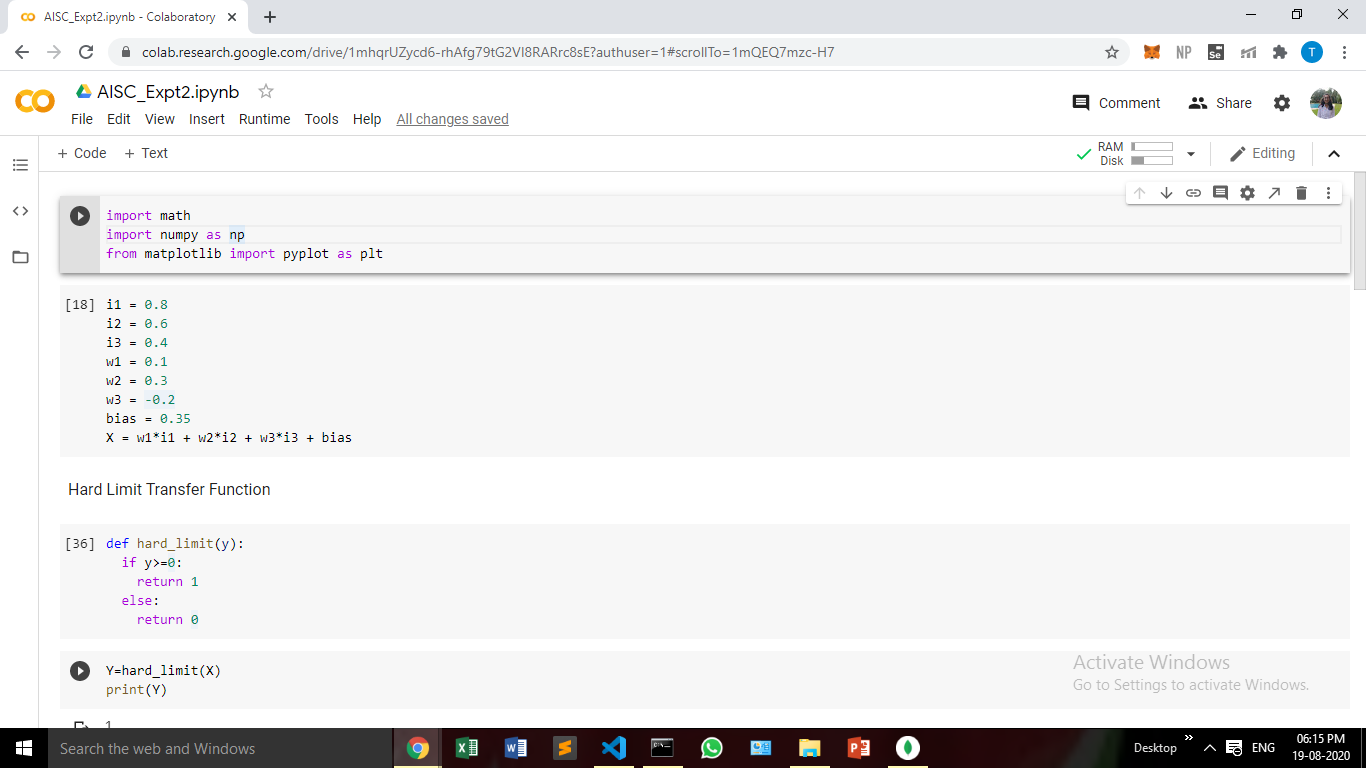
**def logsigmoid(y):**

**return 1/(1+math.exp(-y))**

**Y=logsigmoid(X)**

**print(Y)**

**Output:**

****

**Conclusion:**

In this experiment, we have implemented various activation functions and displayed the output for a given set of weights and bias.