**Soft Computing**

**Lab – 1**

1. **Creating a Simple Neural Network**

**Source Code :**

import numpy as np

class NeuralNetwork():

    def \_\_init\_\_(self):

        # Seed the random number generator

        np.random.seed(1)

        # Set synaptic weights to a 3x1 matrix,

        # with values from -1 to 1 and mean 0

        self.synaptic\_weights = 2 \* np.random.random((3, 1)) - 1

    def sigmoid(self, x):

        """

        Takes in weighted sum of the inputs and normalizes

        them through between 0 and 1 through a sigmoid function

        """

        return 1 / (1 + np.exp(-x))

    def sigmoid\_derivative(self, x):

        """

        The derivative of the sigmoid function used to

        calculate necessary weight adjustments

        """

        return x \* (1 - x)

    def train(self, training\_inputs, training\_outputs, training\_iterations):

        """

        We train the model through trial and error, adjusting the

        synaptic weights each time to get a better result

        """

        for iteration in range(training\_iterations):

            # Pass training set through the neural network

            output = self.think(training\_inputs)

            # Calculate the error rate

            error = training\_outputs - output

            # Multiply error by input and gradient of the sigmoid function

            # Less confident weights are adjusted more through the nature of the function

            adjustments = np.dot(training\_inputs.T, error \* self.sigmoid\_derivative(output))

            # Adjust synaptic weights

            self.synaptic\_weights += adjustments

    def think(self, inputs):

        """

        Pass inputs through the neural network to get output

        """

        inputs = inputs.astype(float)

        return self.sigmoid(np.dot(inputs, self.synaptic\_weights))

if \_\_name\_\_ == "\_\_main\_\_":

    # Initialize the single neuron neural network

    neural\_network = NeuralNetwork()

    print("Random starting synaptic weights: ")

    print(neural\_network.synaptic\_weights)

    # The training set, with 4 examples consisting of 3

    # input values and 1 output value

    training\_inputs = np.array([[0,0,1],

                                [1,1,1],

                                [1,0,1],

                                [0,1,1]])

    training\_outputs = np.array([[0,1,1,0]]).T

    # Train the neural network

    neural\_network.train(training\_inputs, training\_outputs, 10000)

    print("Synaptic weights after training: ")

    print(neural\_network.synaptic\_weights)

    A = str(input("Input 1: "))

    B = str(input("Input 2: "))

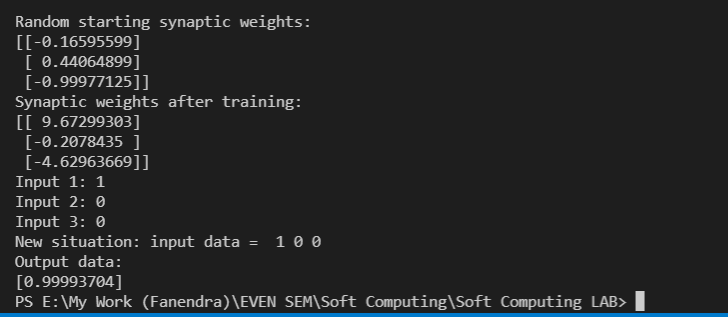
    C = str(input("Input 3: "))

    print("New situation: input data = ", A, B, C)

    print("Output data: ")

    print(neural\_network.think(np.array([A, B, C])))

**Output :**

****