**PRACTICAL – 10**

# # Python program to implement a single neuron neural network

from numpy import exp; from numpy import array; from numpy import random; from numpy import dot; from numpy import tanh;

class NeuralNetwork(): def init (self):

random.seed(1)

self.weight\_matrix = 2 \* random.random((3, 1)) – 1 return tanh(x)

def tanh\_derivative(self, x): return 1.0 - tanh(x) \*\* 2

def forward\_propagation(self, inputs):

return self.tanh(dot(inputs, self.weight\_matrix)) def train(self, train\_inputs, train\_outputs,

num\_train\_iterations):

for iteration in range(num\_train\_iterations): output = self.forward\_propagation(train\_inputs) error = train\_outputs - output

adjustment = dot(train\_inputs.T, error \* self.tanh\_derivative(output))

self.weight\_matrix += adjustment if name == " main ":

neural\_network = NeuralNetwork()

print ('Random weights at the start of training') print (neural\_network.weight\_matrix)

train\_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])

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

neural\_network.train(train\_inputs, train\_outputs, 10000) print ('New weights after training')

print (neural\_network.weight\_matrix)

# Test the neural network with a new situation. print ("Testing network on new examples ->")

print (neural\_network.forward\_propagation(array([1, 0, 0])))

# OUTPUT:-

