**NAME: AKSHATA NAIK**

**USN: 4NM17IS011**

**IMPLEMENTATION OF OR GATE USING ARTIFICIAL NEURAL NETWORK**

import numpy

import scipy.special

import glob

import scipy.misc

class neuralNetwork:

def \_\_init\_\_(self, inputNodes, hiddenOneNodes, hiddenTwoNodes, hiddenThreeNodes, finalNodes, alpha):

self.inputNodes = inputNodes

self.hiddenOneNodes = hiddenOneNodes

self.hiddenTwoNodes = hiddenTwoNodes

self.hiddenThreeNodes = hiddenThreeNodes

self.finalNodes = finalNodes

self.alpha = alpha

self.weightsInputHidden = numpy.random.normal(0.0, pow(self.hiddenOneNodes, -0.5),(self.hiddenOneNodes,self.inputNodes))

self.weightsHiddenOneHiddenTwo = numpy.random.normal(0.0, pow(self.hiddenTwoNodes,-0.5),(self.hiddenTwoNodes,self.hiddenOneNodes))

self.weightsHiddenTwoHiddenThree = numpy.random.normal(0.0, pow(self.hiddenThreeNodes,-0.5),(self.hiddenThreeNodes,self.hiddenTwoNodes))

self.weightsHiddenOutput = numpy.random.normal(0.0, pow(self.hiddenOneNodes,-0.5),(self.finalNodes, self.hiddenThreeNodes))

pass

def train(self, inputs, target):

inputs = numpy.array(inputs, ndmin=2).T

target = numpy.array(target, ndmin=2).T

hiddenInput = numpy.dot(self.weightsInputHidden,inputs)

hiddenOneOutput = self.sigmoid(hiddenInput)

hiddenTwoInput = numpy.dot(self.weightsHiddenOneHiddenTwo,hiddenOneOutput)

hiddenTwoOutput = self.sigmoid(hiddenTwoInput)

hiddenThreeInput = numpy.dot(self.weightsHiddenTwoHiddenThree,hiddenTwoOutput)

hiddenThreeOutput = self.sigmoid(hiddenThreeInput)

finalInput = numpy.dot(self.weightsHiddenOutput,hiddenThreeOutput)

finalOutput = self.sigmoid(finalInput)

outputError = target - finalOutput

hiddenOutputError = numpy.dot(self.weightsHiddenOutput.T, outputError)

hiddenThreeHiddenTwoError = numpy.dot(self.weightsHiddenTwoHiddenThree.T, hiddenOutputError)

hiddenTwoHiddenOneError = numpy.dot(self.weightsHiddenOneHiddenTwo.T, hiddenThreeHiddenTwoError)

hiddenInputError = numpy.dot(self.weightsInputHidden.T, hiddenTwoHiddenOneError)

self.weightsHiddenOutput += self.alpha \* numpy.dot((outputError \* finalOutput \* (1.0 - finalOutput)),numpy.transpose(hiddenThreeOutput))

self.weightsHiddenTwoHiddenThree += self.alpha \* numpy.dot((hiddenOutputError \* hiddenThreeOutput \* (1.0 - hiddenThreeOutput)),numpy.transpose(hiddenTwoOutput))

self.weightsHiddenOneHiddenTwo += self.alpha \* numpy.dot((hiddenThreeHiddenTwoError \* hiddenTwoOutput \* (1.0 - hiddenTwoOutput)),numpy.transpose(hiddenOneOutput))

self.weightsInputHidden += self.alpha \* numpy.dot((hiddenTwoHiddenOneError \* hiddenOneOutput \* (1.0 - hiddenOneOutput)),numpy.transpose(inputs))

pass

def query(self, inputs):

inputs = numpy.array(inputs, ndmin=2).T

hiddenInput = numpy.dot(self.weightsInputHidden,inputs)

hiddenOneOutput = self.sigmoid(hiddenInput)

hiddenTwoInput = numpy.dot(self.weightsHiddenOneHiddenTwo,hiddenOneOutput)

hiddenTwoOutput = self.sigmoid(hiddenTwoInput)

hiddenThreeInput = numpy.dot(self.weightsHiddenTwoHiddenThree,hiddenTwoOutput)

hiddenThreeOutput = self.sigmoid(hiddenThreeInput)

finalInput = numpy.dot(self.weightsHiddenOutput,hiddenThreeOutput)

finalOutput = self.sigmoid(finalInput)

return finalOutput

pass

def sigmoid(self, x):

return scipy.special.expit(x)

pass

n = neuralNetwork(2,12,36,12,1,0.1)

print('Before training')

print(n.query([0,0]))

print(n.query([0,1]))

print(n.query([1,0]))

print(n.query([1,1]))

print("Training...")

for i in range(0, 10000):

n.train([0,0],[0])

n.train([0,1],[1])

n.train([1,0],[1])

n.train([1,1],[1])

print("Done")

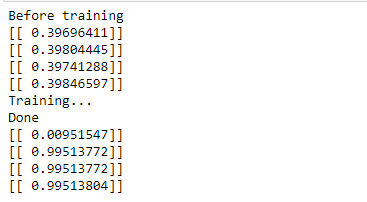
print(n.query([0,0]))

print(n.query([0,1]))

print(n.query([1,0]))

print(n.query([1,1]))

**OUTPUT:**

****