**Name :- prajwal santosh mankar**

**Roll no :-96**

**Practical No.: 13.1**

**Practical Name: Construction Of simple Neural Network using Python**

**Code:**

import numpy as np

from scipy.special import expit as activation\_function

from scipy.stats import truncnorm

# define the network

# generate numbers within a truncated (bounded)

# normal Distribution

def truncated\_normal(mean=0, sd=1, low=0, upp=10):

return truncnorm((low - mean) / sd, (upp - mean) / sd, loc=mean, scale=sd)

# creat the Network class and define the arguments:

# set the no. of neurons/nodes for each layer

# and initialize the weight matrices

class Nnetwork:

def \_\_init\_\_(self, no\_of\_in\_nodes, no\_of\_out\_nodes, no\_of\_hidden\_nodes, learning\_rate):

self.no\_of\_in\_nodes = no\_of\_in\_nodes

self.no\_of\_out\_nodes = no\_of\_out\_nodes

self.no\_of\_hidden\_nodes = no\_of\_hidden\_nodes

self.learning\_rate = learning\_rate

self.create\_weight\_matrices()

def create\_weight\_matrices(self):

"""A method to initialize the weight matrices of the neural network"""

rad = 1 / np.sqrt(self.no\_of\_in\_nodes) # rad = 0.2707

x = truncated\_normal(mean=0, sd=1, low=-rad, upp=rad)

self.weight\_in\_hidden = x.rvs((self.no\_of\_hidden\_nodes, self.no\_of\_in\_nodes))

print("weights\_in\_hidden = ", self.weight\_in\_hidden)

rad = 1/np.sqrt(self.no\_of\_hidden\_nodes)

x = truncated\_normal(mean=0, sd=1, low=-rad, upp=rad)

self.weight\_in\_hidden\_out = x.rvs((self.no\_of\_out\_nodes, self.no\_of\_hidden\_nodes))

print("weights\_in\_hidden\_out = ", self.weight\_in\_hidden\_out)

def train(self, input\_vector, target\_vector):

pass

def run(self, input\_vector):

input\_vector = np.array(input\_vector, ndmin=2).T

print("Input = ", input\_vector)

input\_hidden = activation\_function(self.weight\_in\_hidden @ input\_vector)

print("Hidden = ", input\_hidden)

output\_vector = activation\_function(self.weight\_in\_hidden\_out @ input\_hidden)

print("Output = ", output\_vector)

return output\_vector

simple\_network = Nnetwork(no\_of\_in\_nodes=2, no\_of\_out\_nodes=2, no\_of\_hidden\_nodes=4, learning\_rate=0.6)

#run simple network for arrays, lists and tuples with shape (2):

y = simple\_network.run([2,3])

print("Y = ", y)

**OUTPUT”:**

weights\_in\_hidden = [[-0.68798443 0.29428266]

[ 0.57363879 -0.64646032]

[-0.38809421 0.07104818]

[-0.23288421 0.26427463]]

weights\_in\_hidden\_out = [[ 0.12718945 -0.15067287 -0.36574728 0.3725497 ]

[-0.09102931 -0.22077172 0.40025881 -0.32163589]]

Input = [[2]

[3]]

Hidden = [[0.37915865]

[0.31171721]

[0.36284346]

[0.58104275]]

Output = [[0.52124119]

[0.46381691]]

Y = [[0.52124119]

[0.46381691]]