TRIBHUVAN UNIVERSITY



**Sagarmatha College of Science &**

**Technology**

Lab Report On: Neural Network

Lab Report No.: 02

Date: 2077-08-02

**SUBMITTED BY SUBMITTED TO**

***Name: Ramesh Neupane CSIT Department***

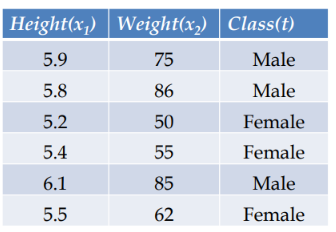
***Roll no.: 37***

**OBJECTIVE**

To implement Perceptron Learning Algorithm

**LAB QUESTIONS**

1. Train AND Gate Using Perceptron Learning Algorithm
2. Train perceptron using given training set and predict class for the input (6,82) and (5.3,52)



**SOURCE CODE AND OUTPUT**

import numpy as np

b = 0.0

alpha = 1

def train\_perceptron(w, x, t):

global b

for i in range(len(x)):

v = sum(w \* x[i]) + b

y = hard\_limiter(v)

dw = alpha \* (t[i] - y) \* x[i]

w = np.add(w, dw)

db = alpha \* (t[i] - y)

b += db

return w

def predict\_perceptron(w, x):

z = w \* x

v = sum(z) + b

y = hard\_limiter(v)

return y

def hard\_limiter(v):

if v > 0:

return 1

elif v < 0:

return -1

else:

return 0

trainx = np.array([[-1,-1], [-1, 1], [1, -1], [1, 1]])

trainy = np.array([-1, -1, -1, 1])

testx = np.array([1, 1])

wt = np.array([0.0, 0.0])

print("Training")

print("+++++++++")

for i in range(1, 5):

print(f"Epoch #{i}")

wt = train\_perceptron(wt, trainx, trainy)

print(f"Weights after epoch {i}: {wt}")

print(f"Bias: {b}")

print("\*\*\*\*\*Testing\*\*\*\*\*")

y = predict\_perceptron(wt, testx)

print(f"Test data: {testx}")

print(f"Output: {y}")

**Output:**

Training

+++++++++

Epoch #1

Weights after epoch 1: [1. 1.]

Bias: -1.0

Epoch #2

Weights after epoch 2: [1. 1.]

Bias: -1.0

Epoch #3

Weights after epoch 3: [1. 1.]

Bias: -1.0

Epoch #4

Weights after epoch 4: [1. 1.]

Bias: -1.0

\*\*\*\*\*Testing\*\*\*\*\*

Test data: [1 1]

Output: 1

import numpy as np

from sklearn.preprocessing import MinMaxScaler

bias = 0.0

alpha = 1

# Neuron for the calculation

def train\_perceptron(tx, wt, t):

global bias

for i in range(len(tx)):

n = sum(wt \* tx[i]) + bias

y = hard\_limiter(n)

wt = np.add(wt, alpha \* (t[i] - y) \* tx[i])

bias += alpha \* (t[i] - y)

return wt

# Output for the test data

def predict\_class(x, w):

n = sum(w \* x) + bias

y = hard\_limiter(n)

return y

# Hard limiter activatin function

def hard\_limiter(n):

if n > 0:

return 1

elif n < 0:

return -1

else:

return 0

# Raw input heights and weights into trainx

trainx = [[5.9, 75], [5.8, 86], [5.2, 50], [5.4, 55], [6.1, 85], [5.5, 62]]

# Output class 1 for MALE and -1 for FEMALE

trainy = [1, 1, -1, -1, 1, -1]

# Changing trainx and trainy into np.array

trainx = np.array(trainx)

trainy = np.array(trainy)

weights = np.array([0.0, 0.0])

# normalizing input data

minmax = MinMaxScaler()

trainx = minmax.fit\_transform(trainx)

print("\*\*\*\*\*\*Training\*\*\*\*\*\*")

for i in range(5):

print(f"Epoch #{i}")

weights = train\_perceptron(trainx, weights, trainy)

print(f"Weights after epoch {i}: {weights}")

print(f"Bias: {bias}")

print("\n\*\*\*\*\*\*Testing\*\*\*\*\*\*")

print("Enter test data:")

testx = []

for i in range(len(weights)):

e = float(input())

testx.append(e)

testx = np.array([testx])

testx = minmax.transform(testx)

testx = testx.flatten()

output = predict\_class(testx, weights)

if output == 1:

print("Predicted class: MALE")

else:

print("Predicted class: FEMALE")

**Output:**

\*\*\*\*\*\*Training\*\*\*\*\*\*

Epoch #0

Weights after epoch 0: [0.77777778 0.69444444]

Bias: -1.0

Epoch #1

Weights after epoch 1: [0.77777778 0.69444444]

Bias: -1.0

Epoch #2

Weights after epoch 2: [0.77777778 0.69444444]

Bias: -1.0

Epoch #3

Weights after epoch 3: [0.77777778 0.69444444]

Bias: -1.0

Epoch #4

Weights after epoch 4: [0.77777778 0.69444444]

Bias: -1.0

\*\*\*\*\*\*Testing\*\*\*\*\*\*

1. Enter test data:

6

82

Predicted class: MALE

1. Enter test data:

5.3

52

Predicted class: FEMALE

**CONCLUSION**

Hence, we are able to implement Perceptron Learning Algorithm to predict the class for the test data.