DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF E&ME, NUST, RAWALPINDI

EC-350 Artificial Intelligence and Decision Support System

LAB MANUAL – 13

Course Instructor: Assoc Prof Dr Arsalan Shaukat

Lab Engineer: Kashaf Raheem

Student Name: ______________________________

Degree/ Syndicate: ______________________________

LAB # 13: SINGLE LAYERED FEED FORWARD NEURAL NETWORK (PERCEPTRON)

Lab Objective:

- To implement perceptron in Python

Hardware/Software required:

Hardware: Desktop/ Notebook Computer
Software Tool: Python 3.10.0

Lab Description:

The perceptron is a supervised classification algorithm with a linear decision boundary, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time. The perceptron algorithm dates back to the late 1950s. Its first implementation, in custom hardware, was one of the first artificial neural networks to be produced.

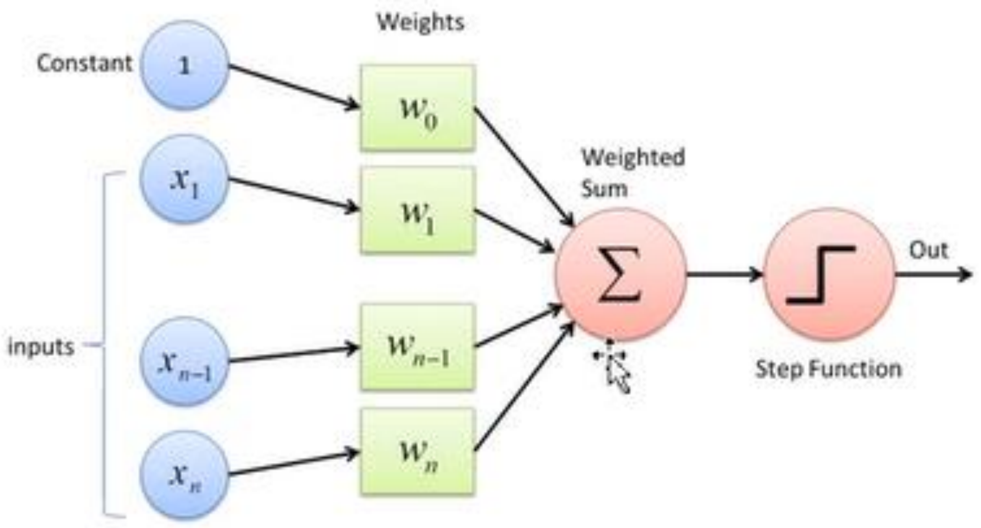
Perceptron maps a feature vector x to an output binary value (0 or 1) using the following relation:

$$f(x) = \begin{cases} 1 & if\ w.x + b > t \\ 0 & otherwise \end{cases}$$

where w is a vector of real-valued weights, $w.x$ is the dot product i.e. $\sum_{i=0}^{m} wixi$, where m is the number of inputs to the perceptron, b is the bias and t is the activation threshold. The bias shifts the decision boundary away from the origin and does not depend on any input value. If the sum of dot product and the biasing factor is greater than the activation threshold, the output of a perceptron is fired otherwise it is not.

Weights can be updates using following formula:

$$Newweight = oldweight + 0.1 \times (target - output) \times input$$



x0	x1	x2	class
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Weights can be updates using following formula:

$$Newweight = oldweight + 0.1 \times (target - output) \times input$$

The pseudo code for the perceptron is given below:

Algorithm: Perceptron

```
Split the dataset into training and testing
Initialize weights and activation threshold ‘t’ with 0
For iter = num iterations:
    For each sample ‘x’ in training
        Split the class ‘c’ and features ‘f’ of x from each other
```

```
Compute dot product of weights and f add them up and store it in ‘o’
    if o >= t
        fire output i.e. o = 1
    else
        o = 0;
    compute i.e. error = learningRate * (c-o)
    for j = number of features
        update weights i.e. weights[j] = weights[j] + f[j] * error
predict output for training data using the weights
Compute accuracy by analyzing predicted Outputs and actual Outputs
```

Lab Tasks:

1) Implement perceptron for the dataset given below with threshold=0.5 and learning rate 0.1. Update weights till all data points are correctly classified.

x0	x1	x2	class
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

2) Use the given SONAR dataset and classify it using perceptron:

a) First create a Python script and load ‘SONAR.csv’ file.
b) Identify features and classes from the loaded dataset.
c) Shuffle the dataset and Split it in such way that 80% of dataset is used for training and 20% for testing.
d) Implement the perceptron using the above algorithm and use training dataset to classify each of the sample within testing dataset. Use learning rate of 0.1, Threshold of 0.5 and 0 biasing factor with 0 initial weights.
e) Repeat the above process until the training accuracy of 75% or above is not achieved.

f) Use those weights to predict the classes for testing data and find the accuracy of prediction.
g) Plot the confusion matrix for the test data classification.