University Institute of Engineering

Department of Electronics & Communication Engineering

Experiment No. :- 1

Student Name: Priyanshu Mathur
UID: 20BEC1073
Branch: Electronics and Communication
Section/Group: A
Semester: 7th
Date of Performance: 08/08/23
Subject Name: artificial intelligence & machine learning
Subject Code: 20ECA-445

1. Aim of the practical: Write a Program to implement a perceptron.

2. Tool Used: Google Collab

3. Theory:

Artificial Neural Networks

Artificial Neural Network (ANN) is a computational model inspired by the way biological neural networks (such as the human brain) function. ANNs are used in machine learning and deep learning to process and analyze complex data. ANN is Composed of layers of interconnected nodes (neurons), ANNs process data through weighted connections and activation functions. During forward propagation, data flows through the network, and backpropagation adjusts connection weights based on error to minimize a loss function.

Biological Neural Network

Biological Neural Network (BNN) is a structure that consists of Synapse, dendrites, cell body, and axon. In this neural network, the processing is carried out by neurons. Dendrites receive signals from other neurons, Soma sums all the incoming signals and axon transmits the signals to other cells.

Difference between ANN and BNN

Parameters	ANN	BNN
Structure	Not so much complex	Complex
Process Speed	High Speed	Low Speed
Computing	Stored Program	Self-learning
Oriented	Well-Defined	Poorly Defined

4. Steps for experiment/practical:

Step 1: - Open Google Collab

Step 2: - Create a new notebook

Step 3: - Write the code given below and run it.

5. Program Code and Simulation Output:

Code:-

```
import numpy
import matplotlib.pyplot

def sigmoid(sop):
    return 1.0/(1+numpy.exp(-1*sop))

def error(predicted, target):
    return numpy.power(predicted-target, 2)

def error_predicted_deriv(predicted, target):
    return 2*(predicted-target)

def activation_sop_deriv(sop):
    return sigmoid(sop)*(1.0-sigmoid(sop))

def sop_w_deriv(x):
    return x

def update_w(w, grad, learning_rate):
    return w - learning_rate*grad

x=0.1
target = 0.3
```

```
learning_rate = 0.5
w=numpy.random.rand()
print("Initial W : ", w)

network_error = []
predicted_output = []

old_err = 0
for k in range(80000):
    # Forward Pass

    y = w*x
    predicted = sigmoid(y)
    err = error(predicted, target)

    predicted_output.append(predicted)
    network_error.append(err)

    # Backward Pass
    g1 = error_predicted_deriv(predicted, target)

    g2 = activation_sop_deriv(predicted)

    g3 = sop_w_deriv(x)

    grad = g3*g2*g1
    print(predicted)

    w = update_w(w, grad, learning_rate)

    old_err = err

matplotlib.pyplot.figure()
matplotlib.pyplot.plot(network_error)
matplotlib.pyplot.title("Iteration Number vs Error")
matplotlib.pyplot.xlabel("Iteration Number")
matplotlib.pyplot.ylabel("Error")
```

```
matplotlib.pyplot.figure()
matplotlib.pyplot.plot(predicted_output)
matplotlib.pyplot.title("Iteration Number vs Prediction")
matplotlib.pyplot.xlabel("Iteration Number")
matplotlib.pyplot.ylabel("Prediction")
```

Plot

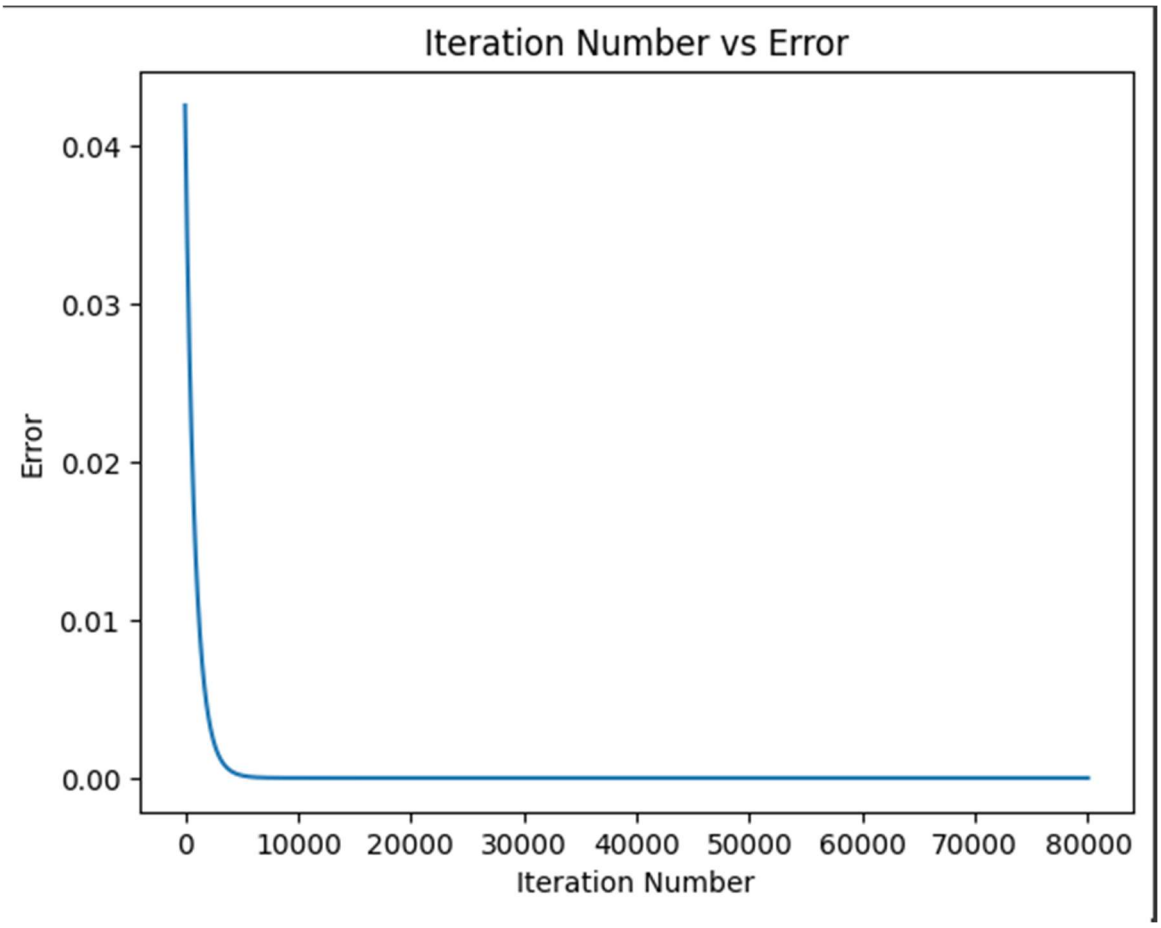

Figure 2: - Plot of iteration number Vs Error.

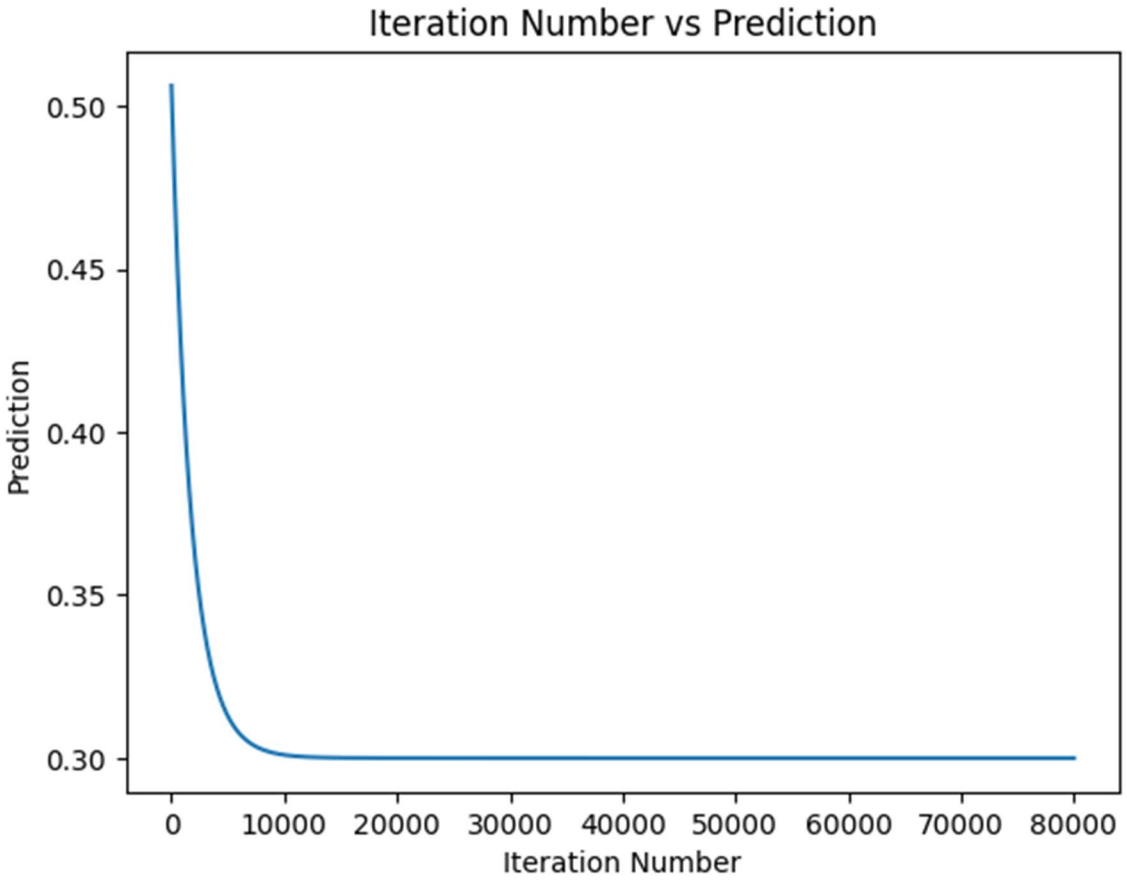

Figure 3: - Plot of Iteration Number vs Prediction

Result and Discussion: - In this experiment we calculated error by the predicted value and target value. We have iterated 80,000 times to get close to our predicted value. We have defined sigmoid function which will give us predicted value.

Learning outcomes (What I have learnt):

- Learnt about errors measurement and predicted value.
- Learn about Google collab.
- Learn to about Python.
- Learn to simulate the robot in Simulink.