# Purdue University Fort Wayne

ECE 595
Machine Learning

Project # 2

**Afroza Haque**

Instructor: Bin Chen
Submitted: 07 October 2022

**Introduction:**

Perceptron is a machine learning algorithm that functions like neurons in our brain. It is also called a single layer neural network consisting of a single neuron. The output of this neural network is decided based on the outcome of just one activation function associated with the single neuron. It is a binary classifier. That means it classifies dataset into two classes.

The perceptron algorithm consists of four parts:

1. Input values/One input layer
2. Weights and Bias
3. Net Sum
4. Activation Function

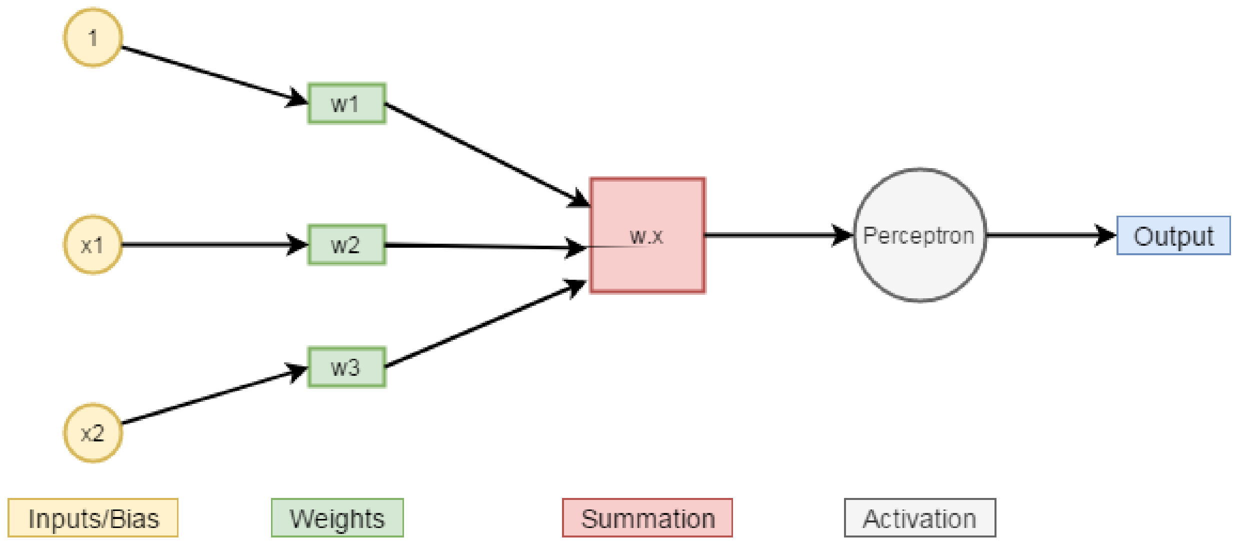

Figure-1: Perceptron Algorithm

The perceptron takes in a vector *x* as the input, multiplies it by the corresponding weight vector, *w*, then adds it to the bias, *b*. The result is then passed through an activation function.

In our case the weighted sum is:

$$X1*W1 + X2*W2$$

Then bias is added with the weighted sum and passed through an activation function. I have used unit step function as an activation function.

**Activation function = {x <= threshold: -1, x > threshold: 1}**

**Methodology:**

Activation function divides the dataset into two classes. To draw the decision boundary (perceptron line) we need slope of the perceptron line and y intercept of the line.

Now, I will derive the equation of slope and y intercept from the perceptron equation.

$$X*W1 + Y*W2 + b = 0$$

When X=0, $Y*W2 + b = 0$

$$Y = -(b/W2)$$

So, one point in the perceptron line is (0, -(b/W2))

Again,

$$X*W1 + Y*W2 + b = 0$$

When Y=0, $X*W1 + b = 0$

$$X = -(b/W1)$$

So, another point in the perceptron line is (-(b/W1), 0)

Now using these two points in the perceptron line slope (m) can be calculated.

$$\text{Slope (m)} = (y2 - y1) / (x2 - x1)$$

$$= \{0 - (-(b/W2)\} / \{-(b/W1) - 0\}$$

$$\text{Slope (m)} = W1 / W2$$

$$\boxed{\begin{array}{c}\textbf{Slope (m) = W1 / W2} \\ \textbf{Y intercept = -(b/W2)}\end{array}}$$

We get weights and bias from the perceptron algorithm. In my program I have used weights and bias to calculate the slope and y intercept of the perceptron line.

**Code Explanation:**

1. **Libraries:**

   - Numpy
   - Random
   - Matplotlib.pyplot

2. **Calculating slope and y intercept of the given equation**

   Given equation: 3x1 + 4x2 – 10 = 0
   4x2 = -3x1 + 10
   x2 = (-3/4)x1 + (10/4)
   x2 = -0.75x1 + 2.5
   Therefore, m= -0.73, b = 2.5

3. **Generating random positive and negative samples above and below the line**

   I have added and subtracted gaussian noise from the line equation to generate random points above and below the line.

4. **Plotting the line and data points**

   I have plotted the positive data points with '+' and negative data points with 'o' as instructed in task 2. I have also plotted the original line in green color.

5. **Data set preparation:**

   For dataset preparation I have created two arrays of 1 and -1 as label data set. Then I concatenated the label dataset with the feature dataset.

6. **Perceptron algorithm:**

   First, I have defined the values of initial weights, initial bias, threshold, learning rate. Then the training iterations start. During each iteration, a weighted sum is calculated, then bias is added. Then activation function determines the class (positive or negative). After that the predicted result is compared with the actual class. And if the result is miss classified then weights and bias gets updated.

   The new weight is : w = w + {learning rate * (actual class – predicted class) * input}

7. **Perceptron line (Decision boundary):**

   The slope and y intercept are calculated using the weights and bias returned from the perceptron algorithm. Then the perceptron line is calculated.

8. **Plotting perceptron line:**

   I have plotted the initial perceptron line in dotted red and actual line in solid green. In my first iteration two samples are misclassified.

```
--------------------------Epoch1------------------------------

 Updated weight: [0.0, 0.24054418767043 32] Updated bias: 0.99
```

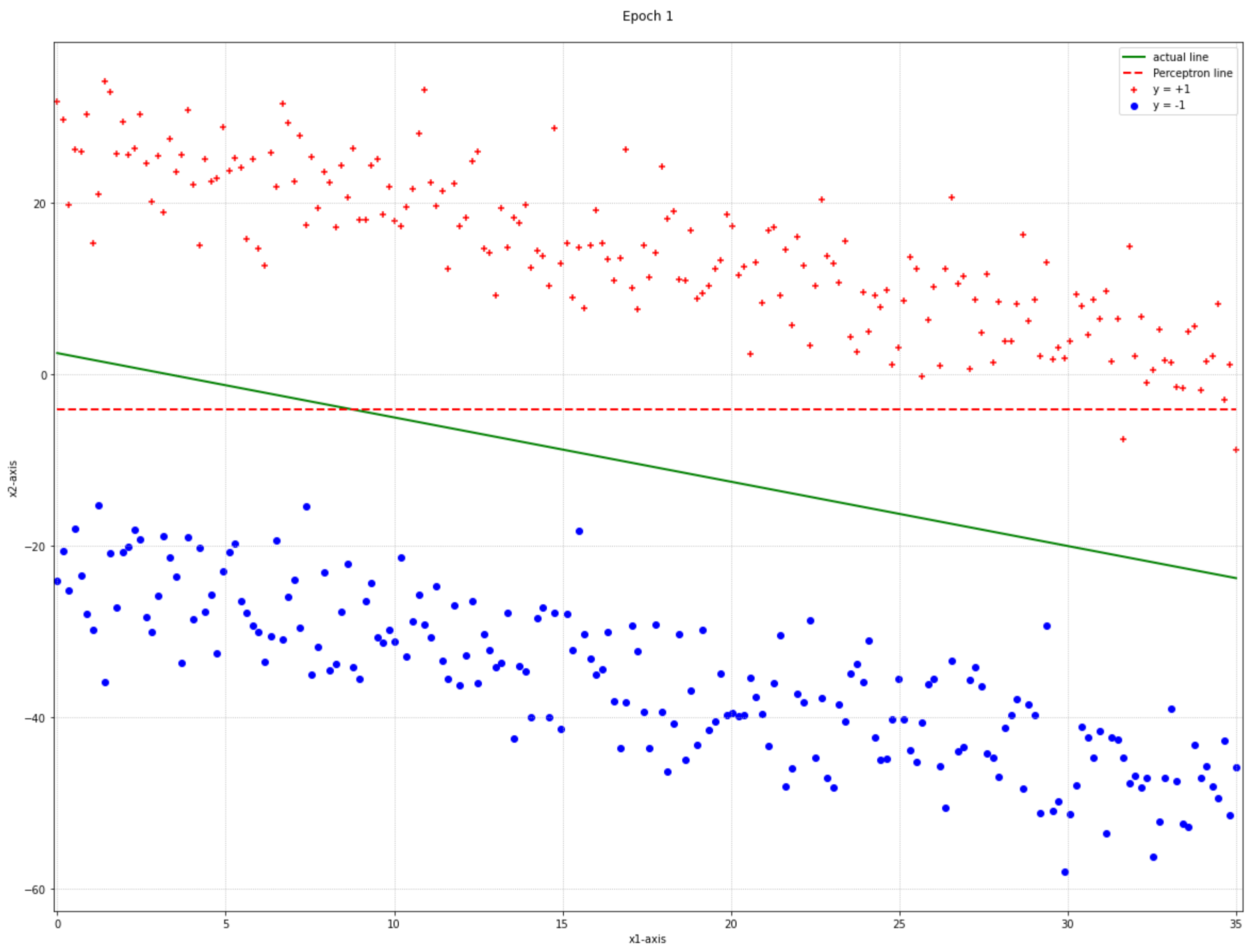

Figure-2: Perceptron line for epoch-1

Then I ran for two epochs. Within two epochs my data set was correctly classified.

```
--------------------------Epoch2------------------------------
Updated weight: [0.242713567839196, 0.3177877965434691] Updated bias: 0.99
```

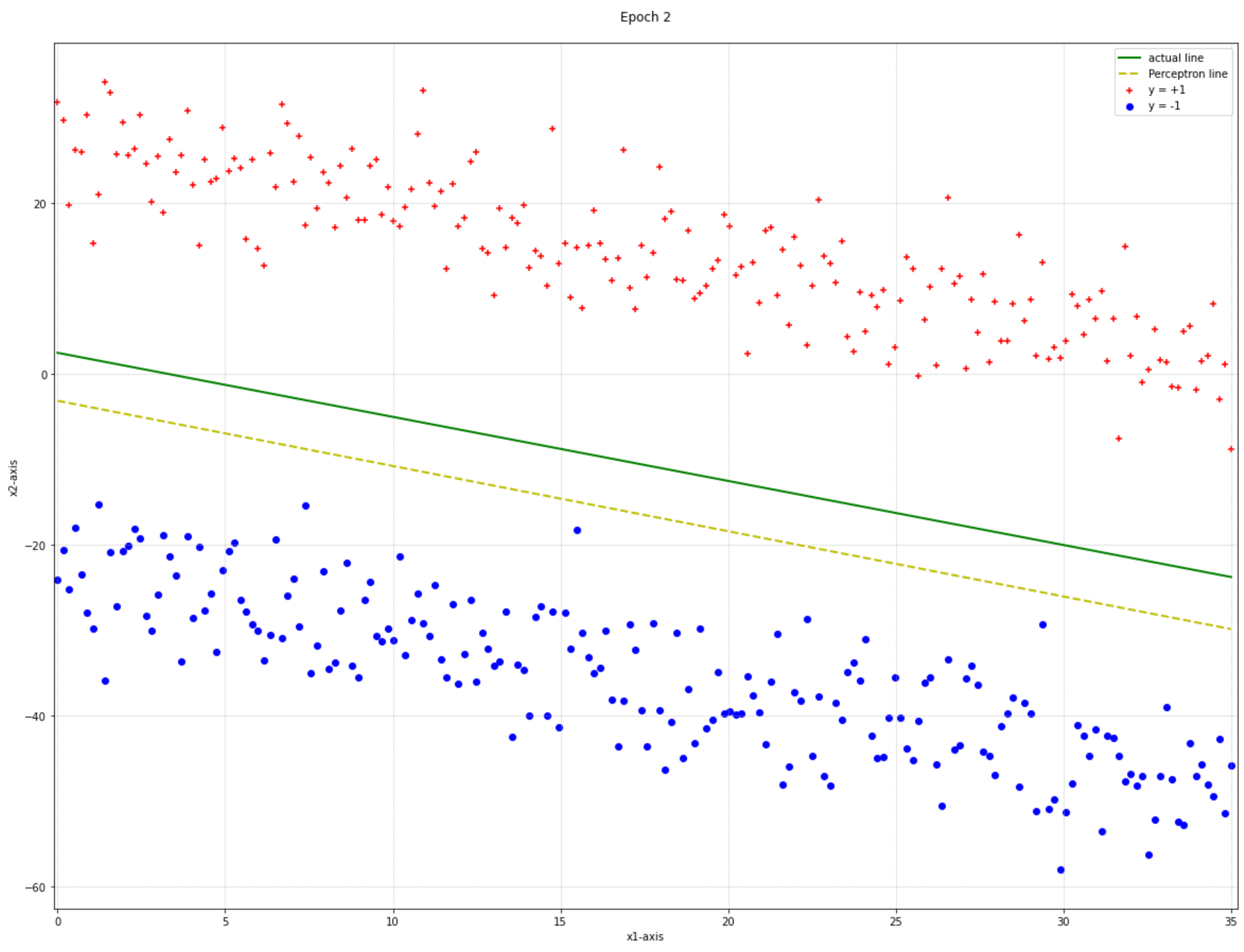

Figure-3: Perceptron line for epoch-2

Then I ran for up to 10 epochs. But the weights didn’t change.

```
--------------------------Epoch3------------------------------
Updated weight: [0.242713567839196, 0.3177877965434691] Updated bias: 0.99
```

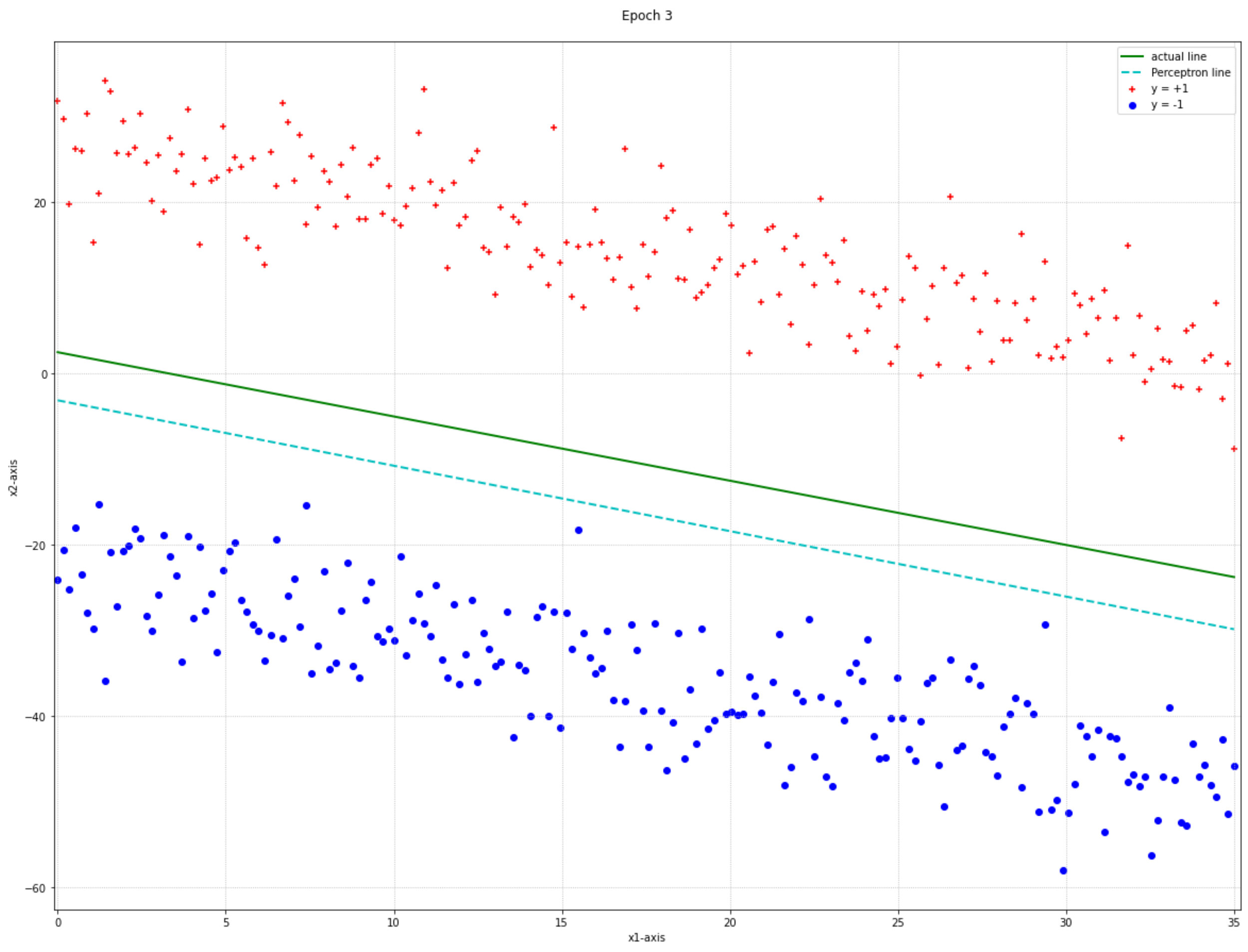

Figure-4: Perceptron line for epoch-3

```
--------------------------Epoch8------------------------------

 Updated weight: [0.242713567839196, 0.3177877965434691] Updated bias: 0.9
9

--------------------------Epoch9------------------------------

 Updated weight: [0.242713567839196, 0.3177877965434691] Updated bias: 0.9
9

--------------------------Epoch10------------------------------

 Updated weight: [0.242713567839196, 0.3177877965434691] Updated bias: 0.9
9
```

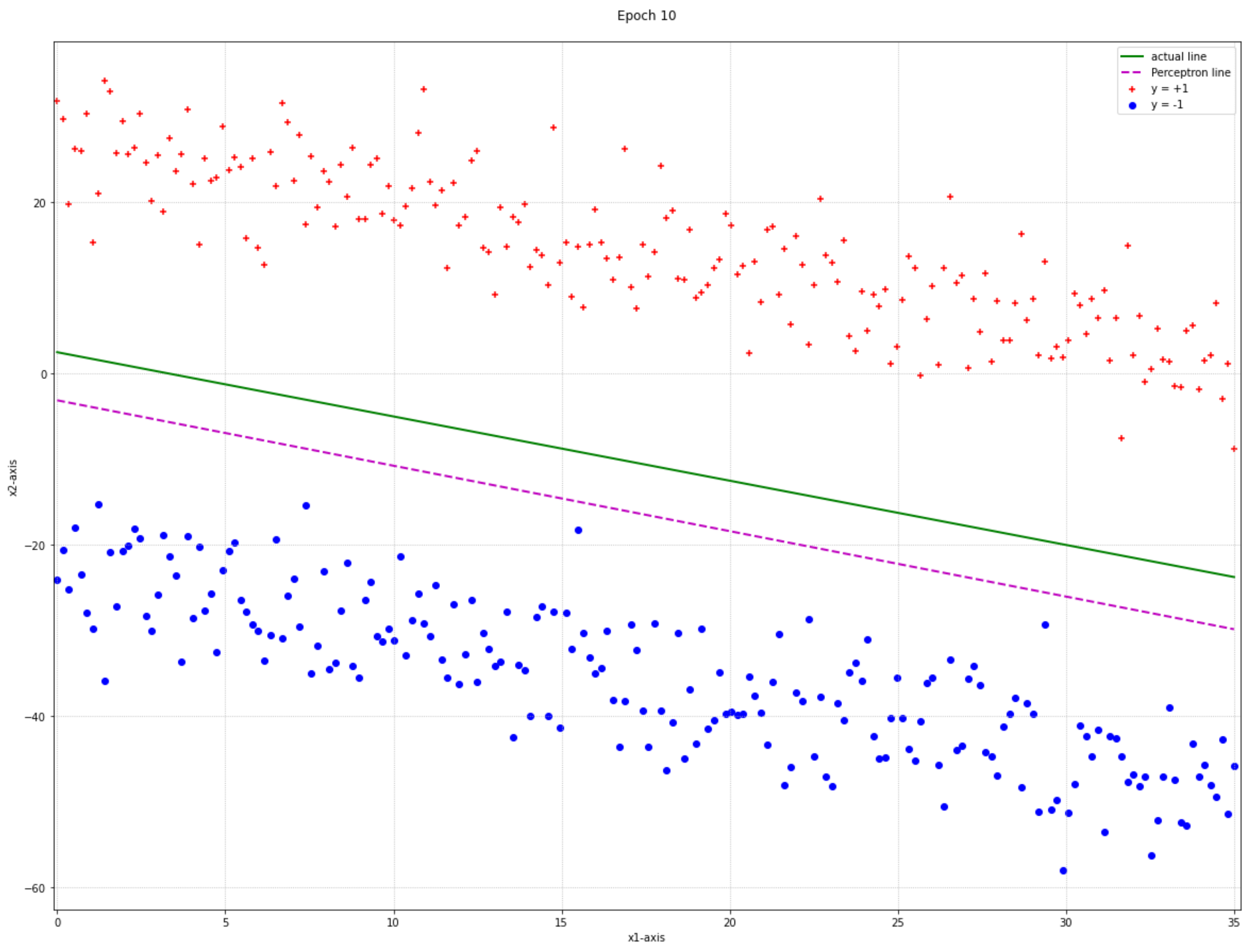

Figure-5: Perceptron line for epoch-10

## Conclusion:

Perceptron algorithm is the simplest form of neural networks. There is a limitation of perceptron algorithms. It is a linear classifier. So, the algorithm can be used to classify only linearly separable data.

## Reference:

1. https://www.thomascountz.com/2018/04/13/calculate-decision-boundary-of-perceptron
2. https://www.youtube.com/watch?v=R2AXVUwBh7A
3. https://towardsdatascience.com/perceptron-algorithm-in-python-f3ac89d2e537