****

**Lab Report**

**EE527 Spring 2020**

**Project 1**

**Name: Trieu Nguyen**

**For: Prof. Bui Hieu**

**Date Performed: 10/1/2020**

List of Contents

1.Introduction

2.Approach

3.Task 1

4.Task 2

5.Conclusion

6.Reference

7.Git hub

1.Introduction:

The Rosenblatt’s perceptron classified the inputs into either 1 or -1. It accomplished this by summing up the multiplication of weights and inputs then using the sgn function to classify the sum into either 1 and 2. Fig 1 represent the perceptron signal flow chart.

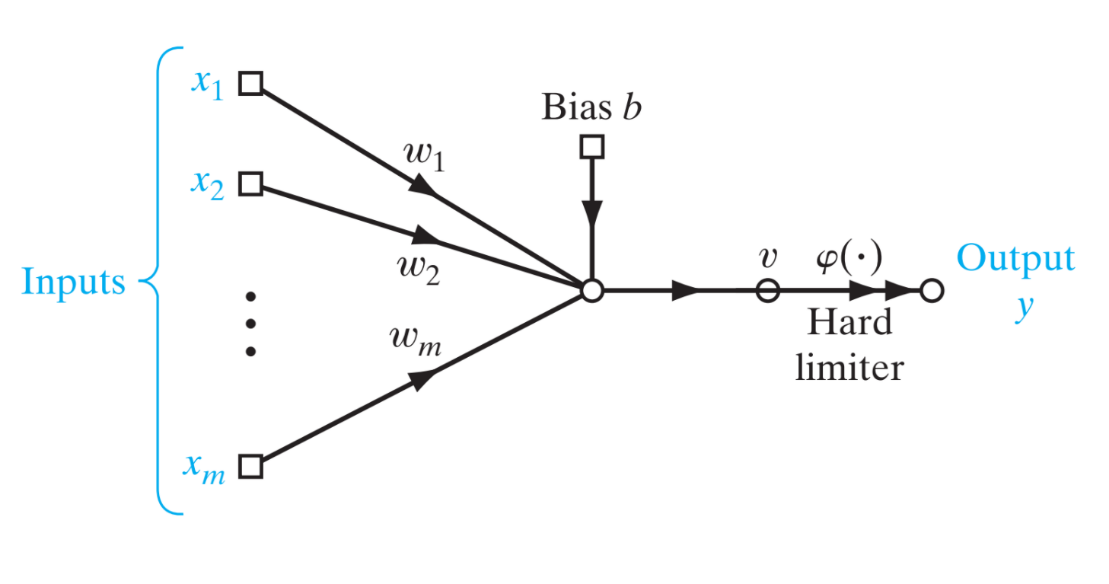


Fig 1. Signal-flow graph of the perceptron.

In particular:

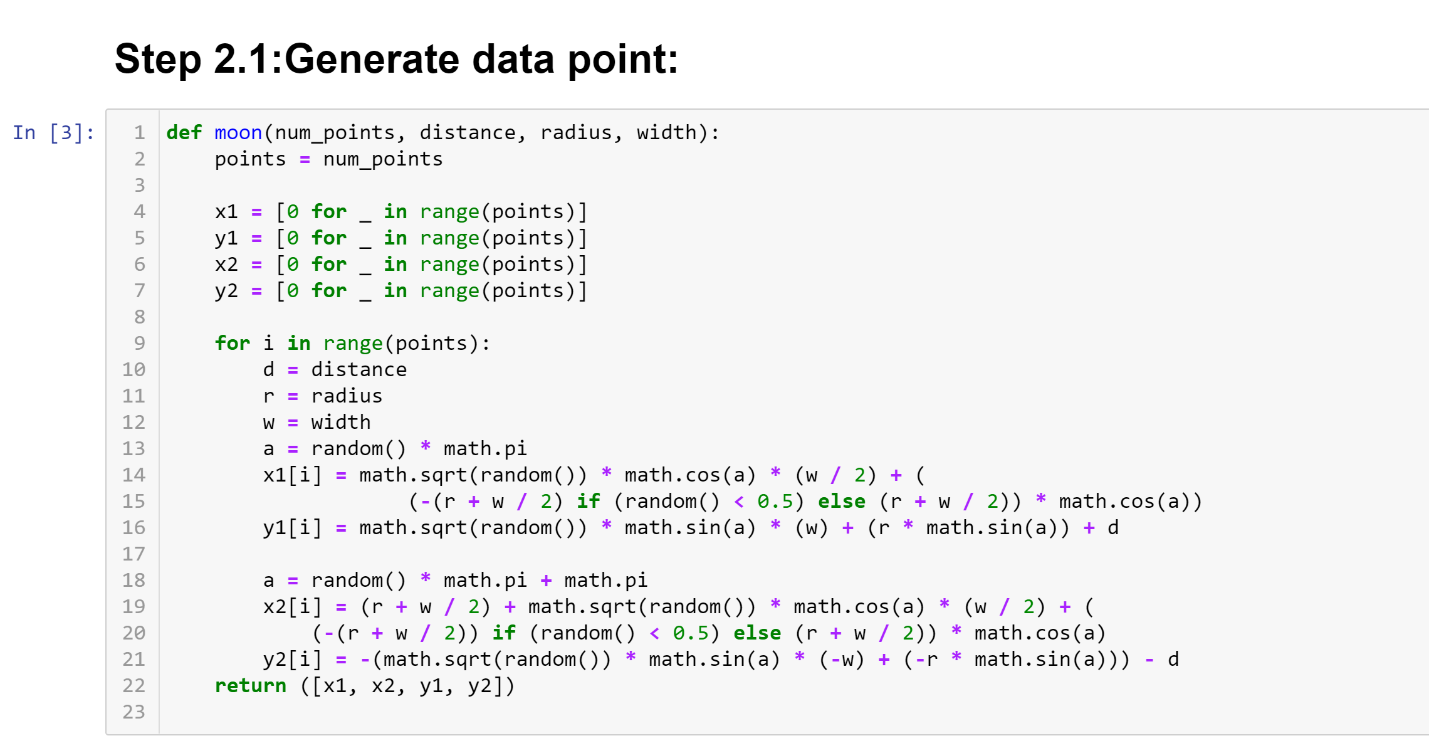
v=sum(w,x)+b

y=1 if v>=0 or -1 if v<0

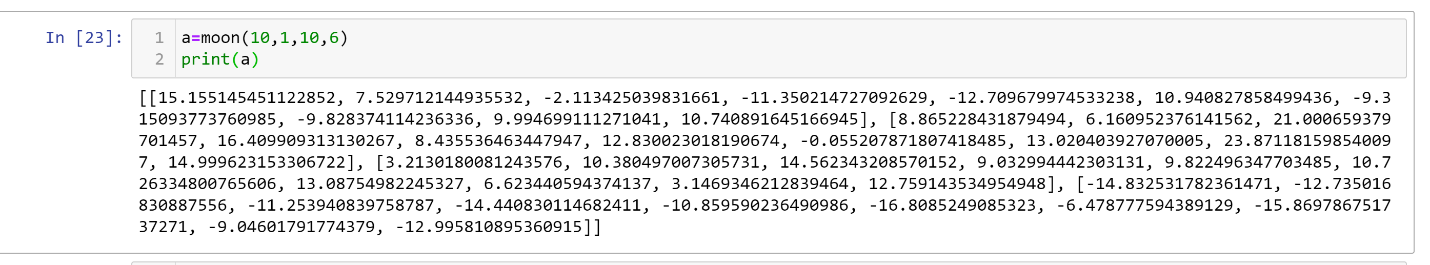
2.Approach:

2.1. Data point generation:

Using the provided moon.py code, I was able to generate the input needed.



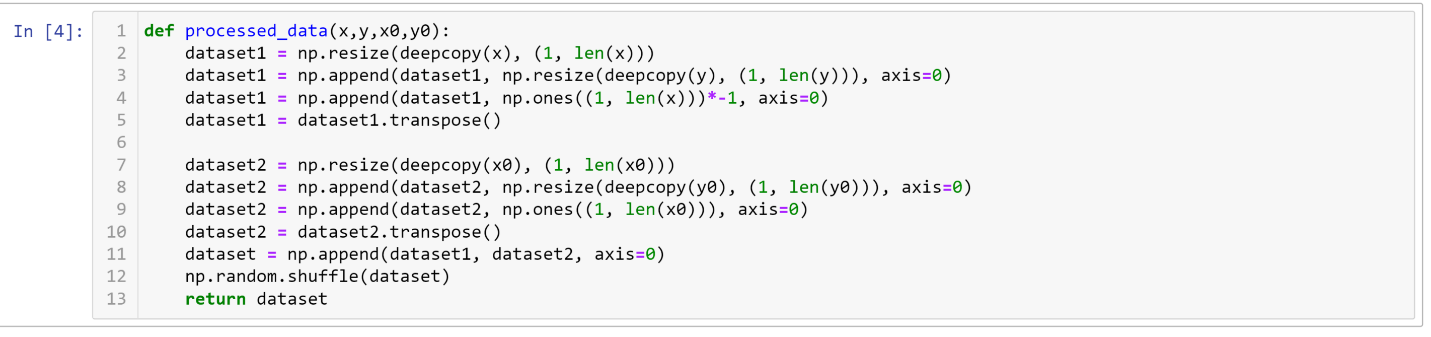
I then try to generate some points by calling out the function.



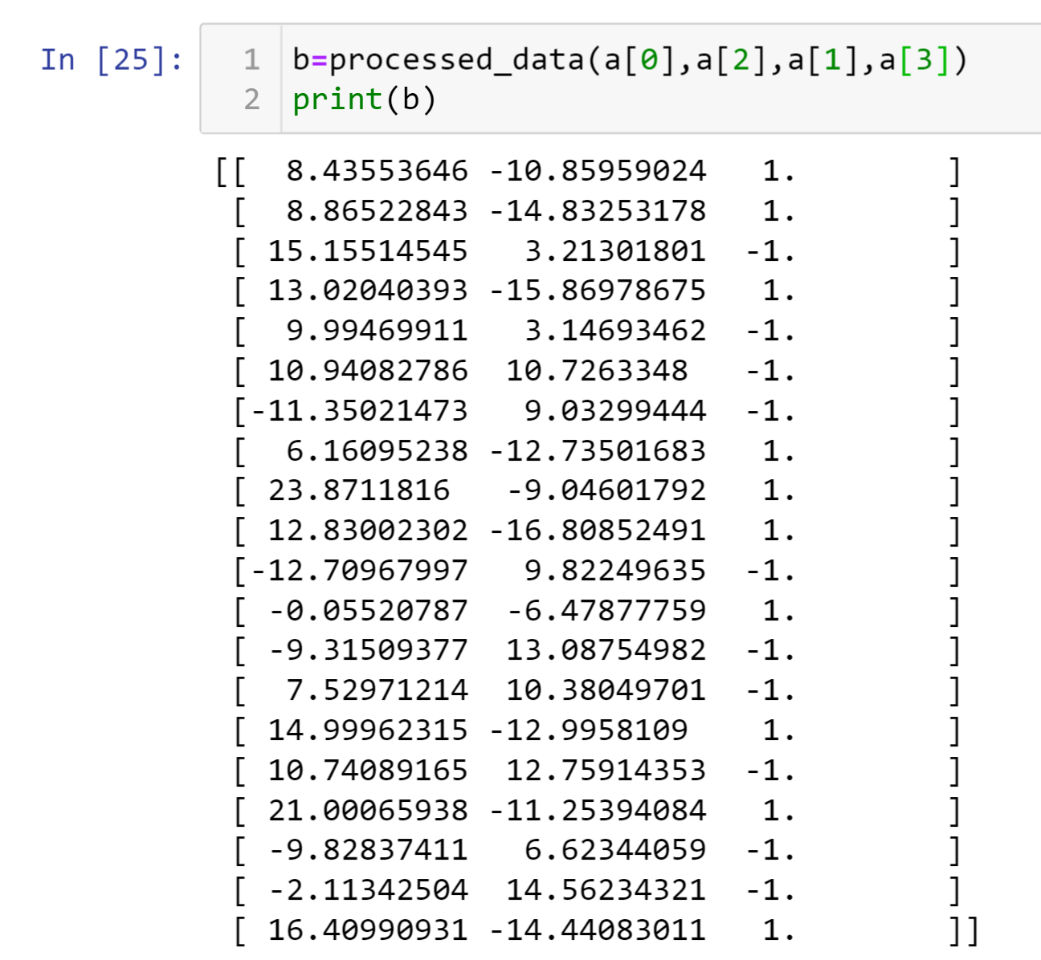
=>Obviously the data is still need processing in order to draw the moon plot.

2.2. Data processing:

The data needed to be pair up correctly as current the moon.py and put in the correct classification (1 or -1). Below is the function I used for this task:

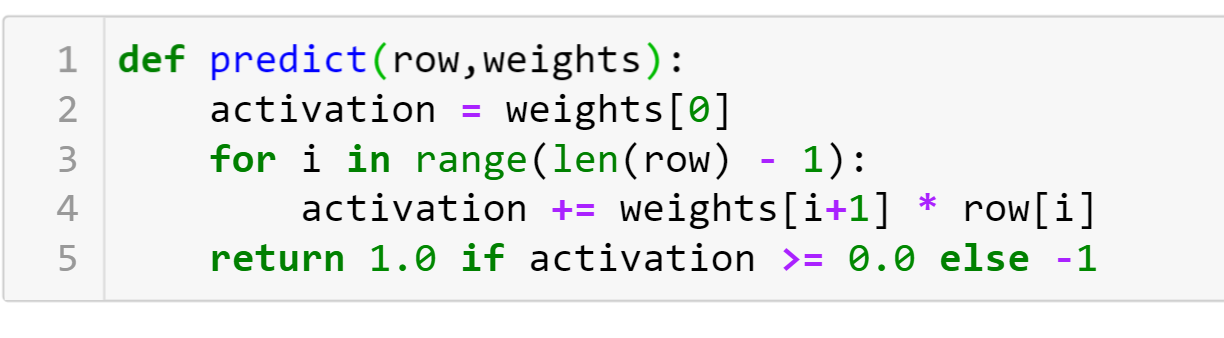


Here is test sample of the function:



2.3 Constructing the perceptron:

2.3.1 Building the perceptron:

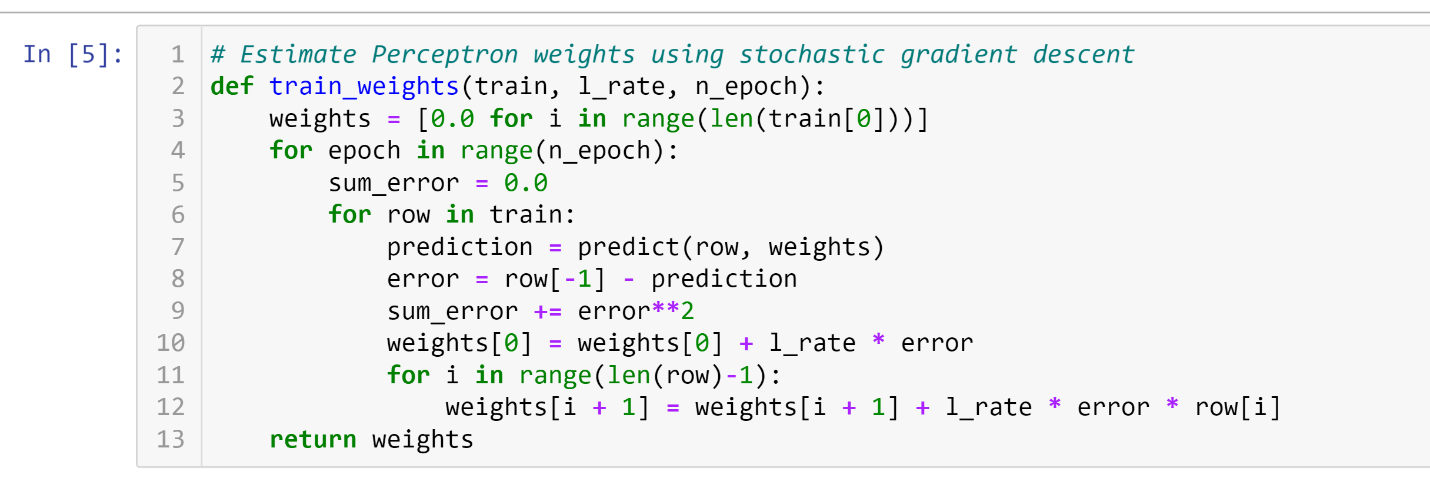


This function used sgn function as it’s activation function(1 if v>=0 and -1 if v<0)

2.3.2 Training weight:

Initially, weight is set to be 0. Then by using gradient descent (Eq. 1), we will be able to change weights in order to separate the two moons.

w(t+1)= w(t) + learning\_rate \* (expected(t) - predicted(t)) \* x(t) (Eq.1)



2.4 Ploting:



I picked min and max value based on the Fig 1.9 and Fig 1.10 in the test book (as x range from -20 to 20). The function of line that separate the two moons is:

y = -(weights[1]\*x/weights[2]) - weights[0]/weights[2] (Eq. 2)

As it is linear, I only need the 2 endpoints of x to graph the line.

The dataset needed to classify into two class before we could draw the two moons. This is done by calculating the sum of weight\*input then compare them to 0. Though, I could have use the predict function to separate them into 2 class, but as there are only 3 value of weights I though it would be off to type it out.

3.Task 1:

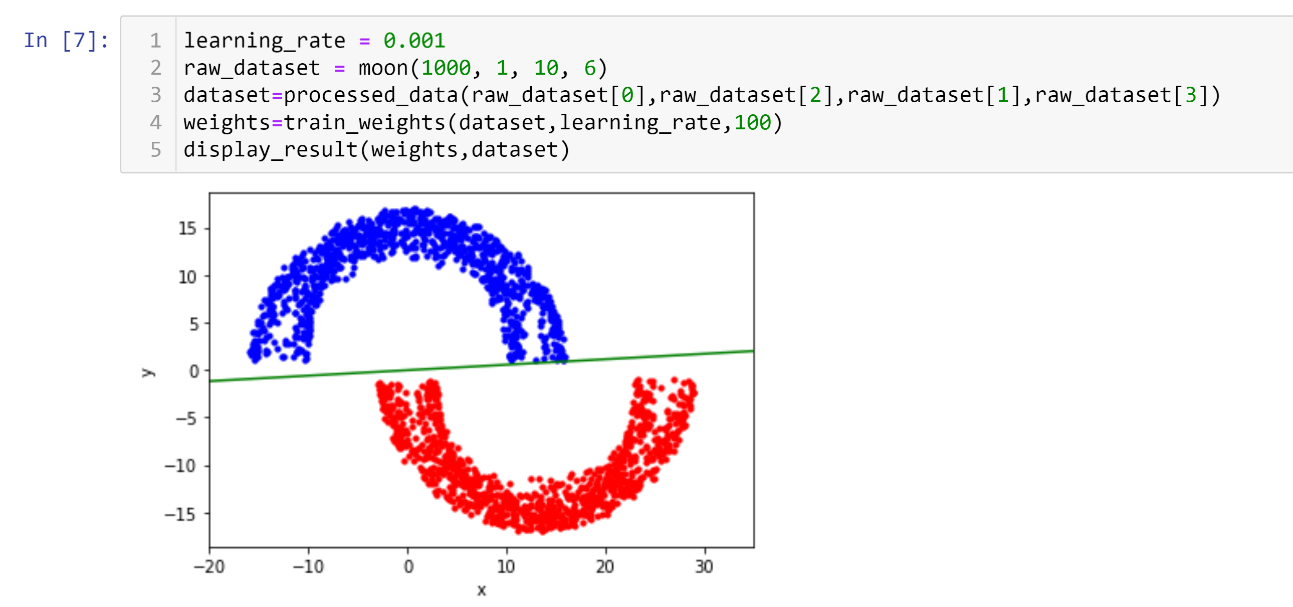
3.1. When d=1:

Number of points= 1000

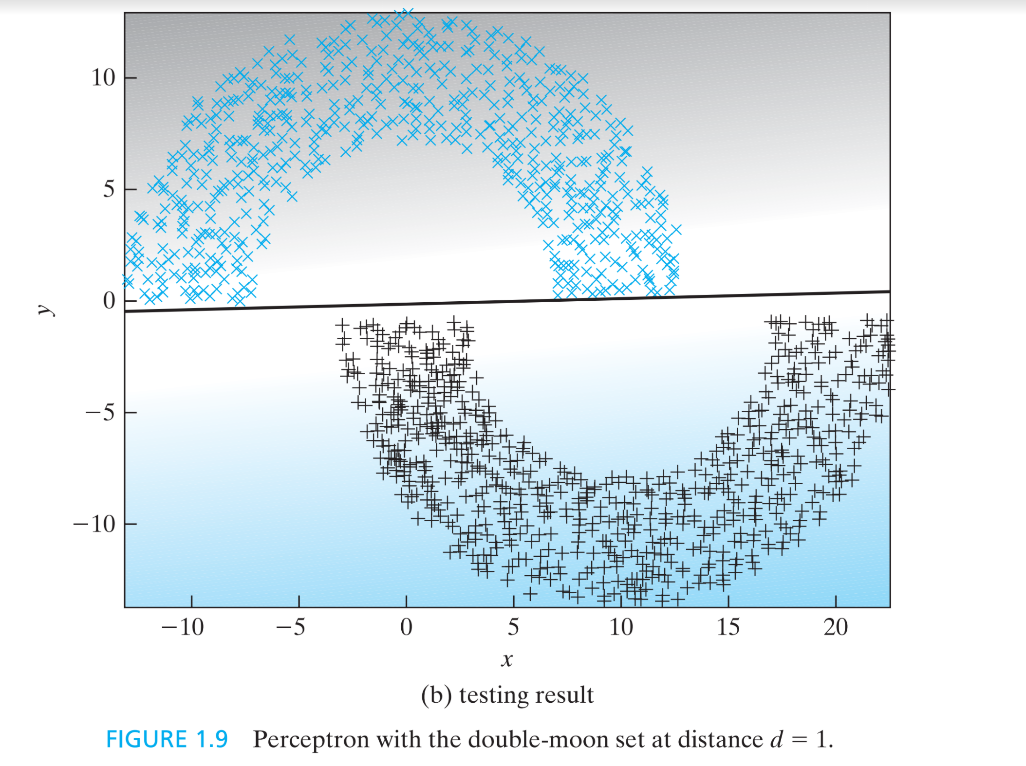
Learning\_rate=0.001

Radius=10

Width=6



=>this is exactly the same as fig 1.9 in the text book.

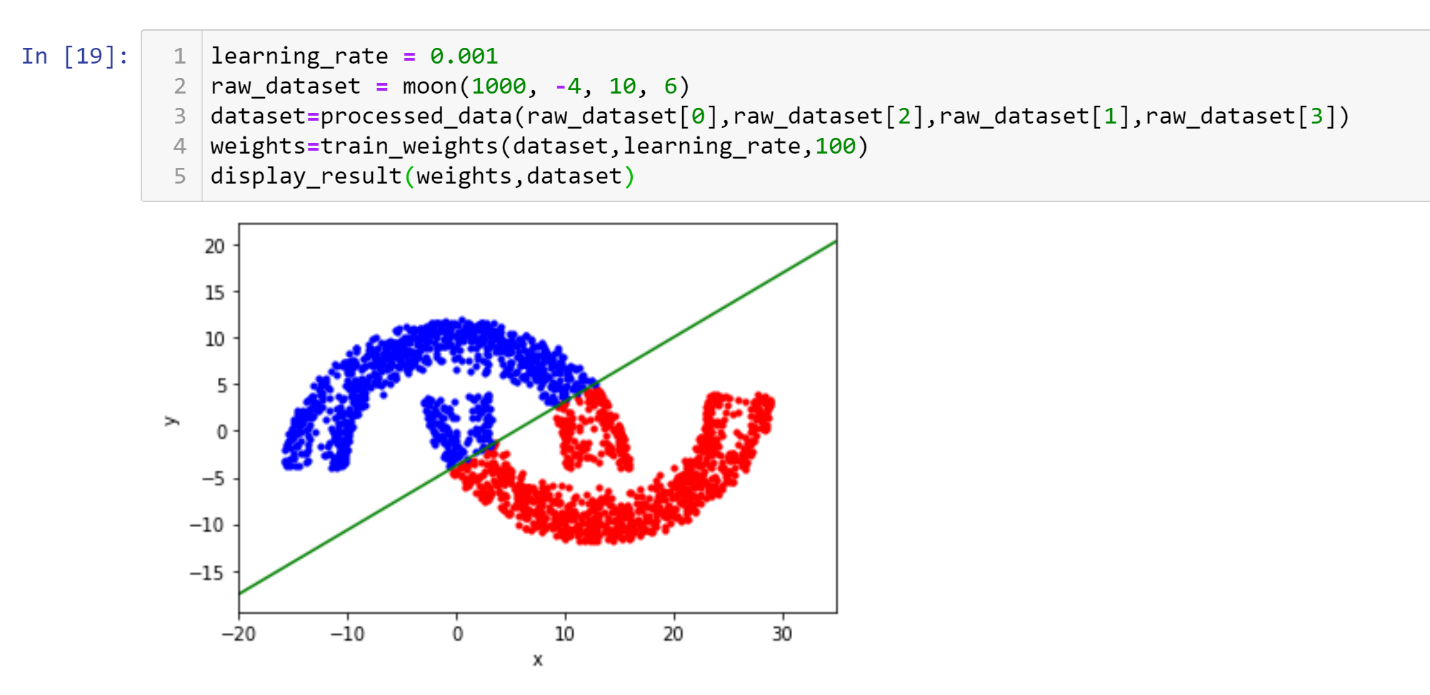


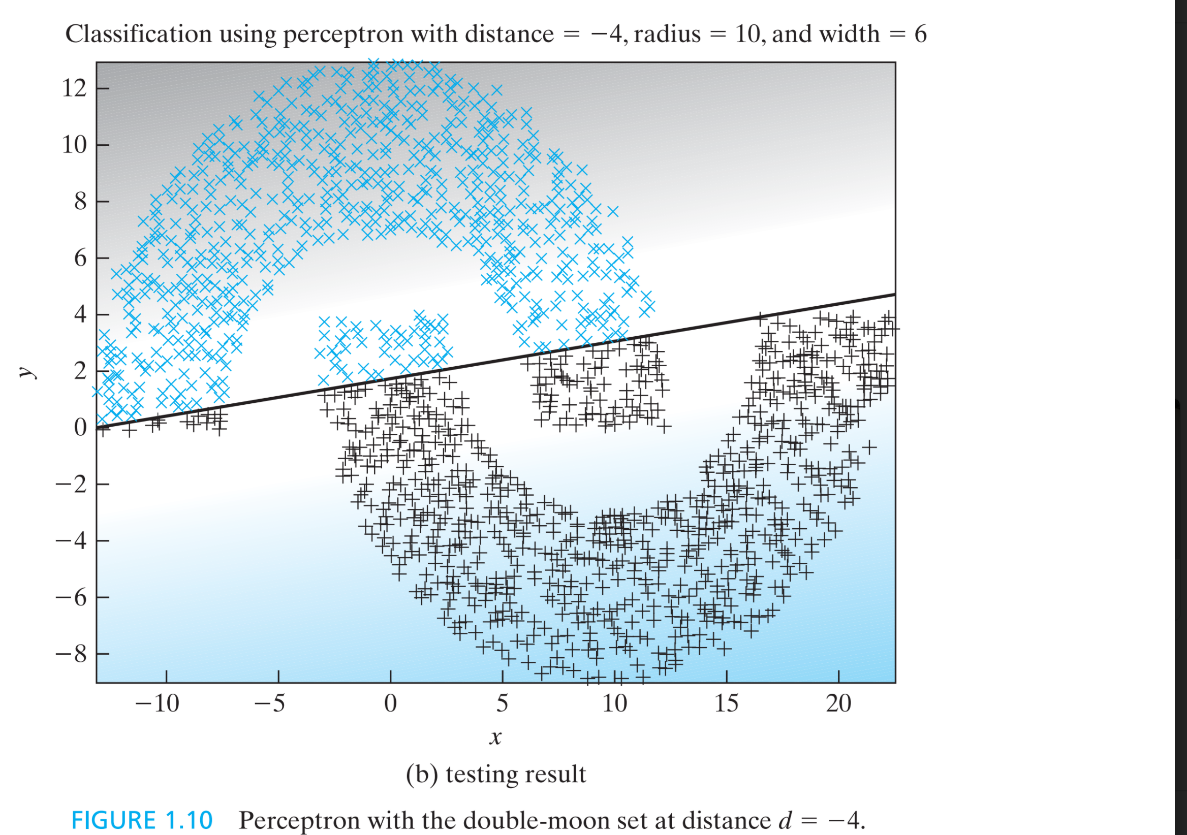
3.2.When d=-4

Number of points=1000

Learning\_rate=0.001

Radius=10

Width=6



4. Task 2:

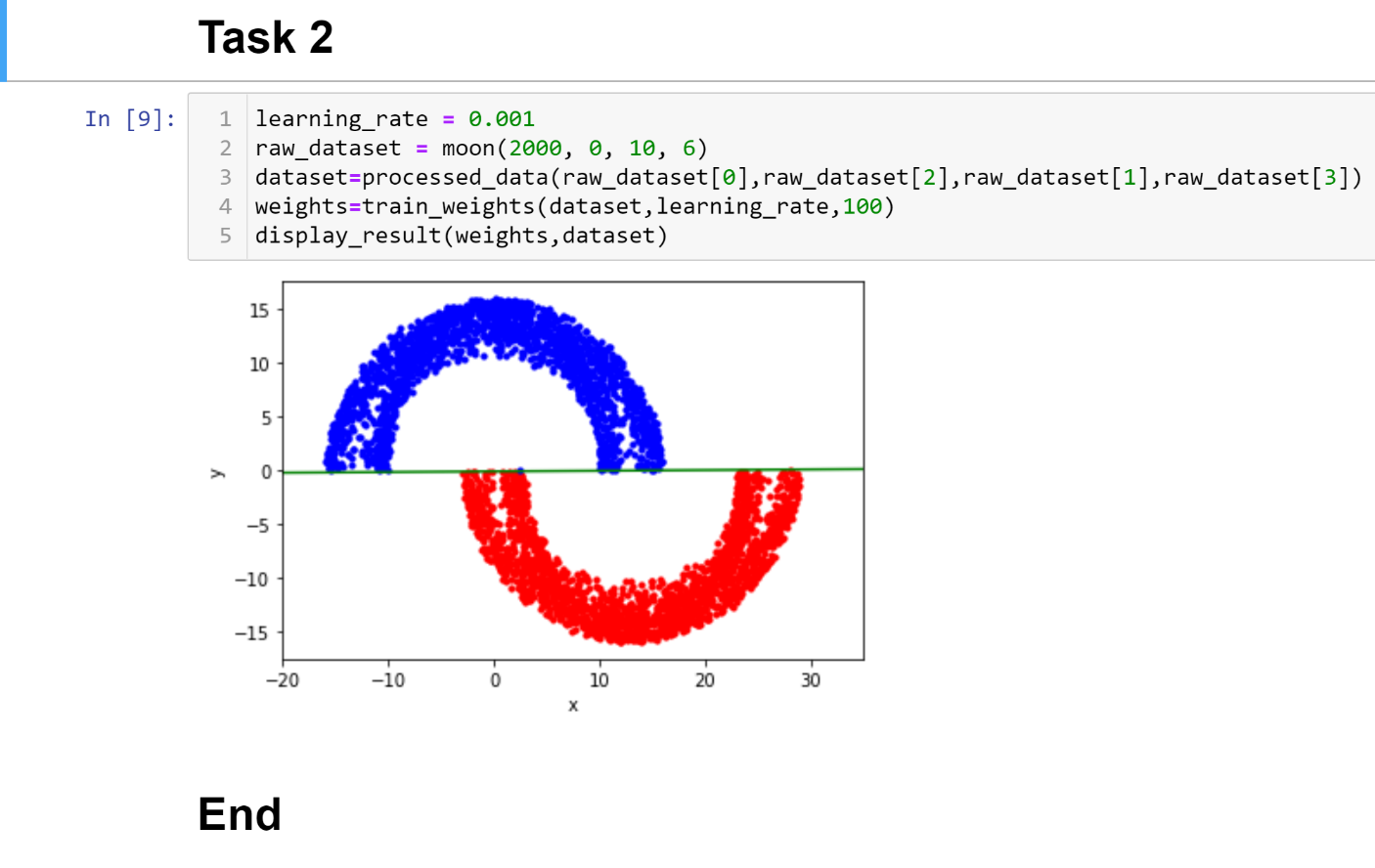
Number of points=1000

Learning\_rate=0.001

Radius=10

Width=6

d=0



5. Conclusion:

Base on the figure we could see that by using gradient descent we could successfully separate the two class perfectly when d>0. When d<=0, there will be overlap between the two datasets, and this method does not work any more as it is only linear in nature.

6.Reference

How To Implement The Perceptron Algorithm From Scratch In Python by Jason Brownlee (link: <https://machinelearningmastery.com/implement-perceptron-algorithm-scratch-python/>)

7.Github:

https://github.com/Trieu-Hoang-Nguyen/csc-527