CMSC 422 - PS4

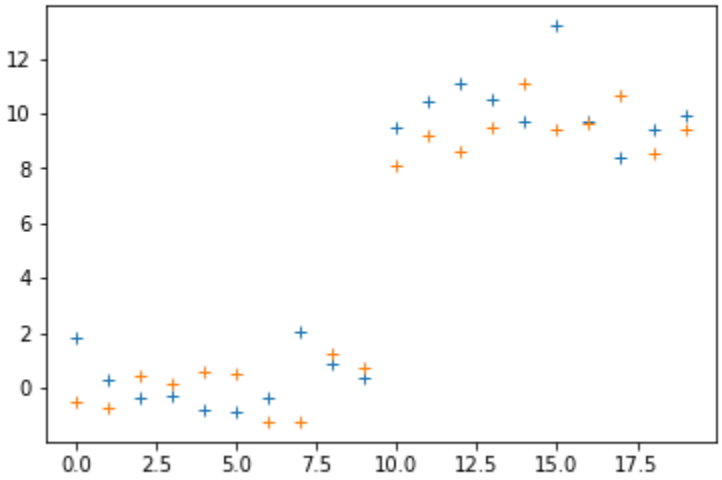
Submitted by – Arpit Maclay

1.) For the first problem I have trained my neural network for k=10000, eta=0.02 and tested for x=[0,1,2,3,4,5,6,7,8,9] and the results are- [array([2.]), array([5.]), array([8.]), array([11.]), array([14.]), array([17.]), array([20.]), array([23.]), array([26.])]

2.) As per my training data the labels inverted i.e. for (0,0) the output is 1 because all the points are trained according to that and for points around (10,10) the label is 0 so the output is 0.

Results-

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

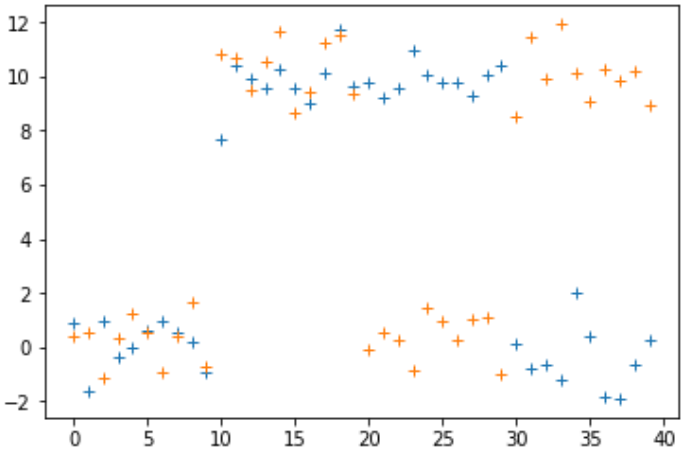



```
[array([0.99745479]), array([0.00015326])]
```

For training set =1

For points (0,0) and (10,10)

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



```
[array([0.76996375]), array([0.05303045])]
```

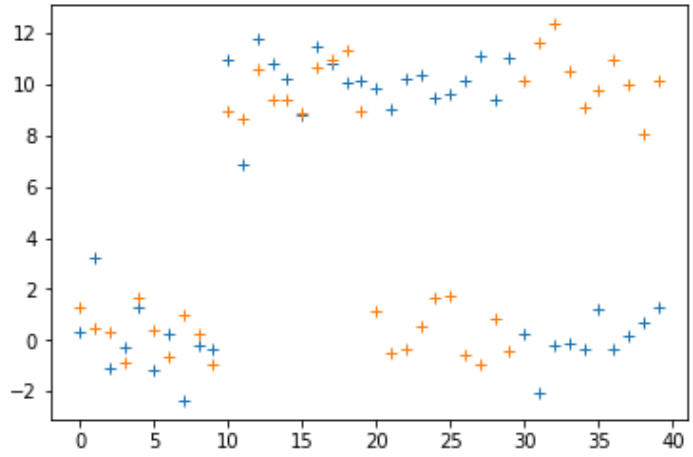
For trainset =2

For point (0,0) and (10,10)

In this case the probability prediction is little altered because of the non-linearity.

Now we consider the training set 2 and points [0,0] and [0,10]

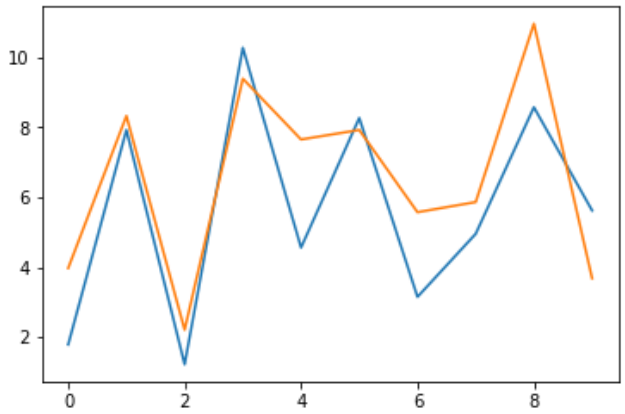
```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



```
[array([0.73081139]), array([0.04986913])]
```

We can see that for the point [0,0] the probability is high towards one while for the points [0,10] the probability is almost 0.

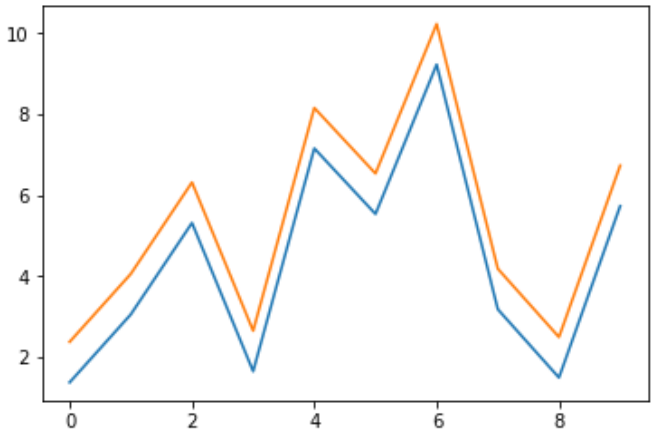
3.) For k =2000, sigma=1 and eta =0.03 my network is showing these results




```
[[0.9487914360841899, 2.0256042819579045], [4.183896573055937, 4.90805171347203],
[4.902808825716325, 5.548595587141836]]
```

Which are accurate but for the input of (10,3) it is resulting [4.902808825716325, 5.548595587141836]
Because the training data is trained according to the line y=x+1 and in testing the network is trying to achieve that relationship hence the unexpected values.

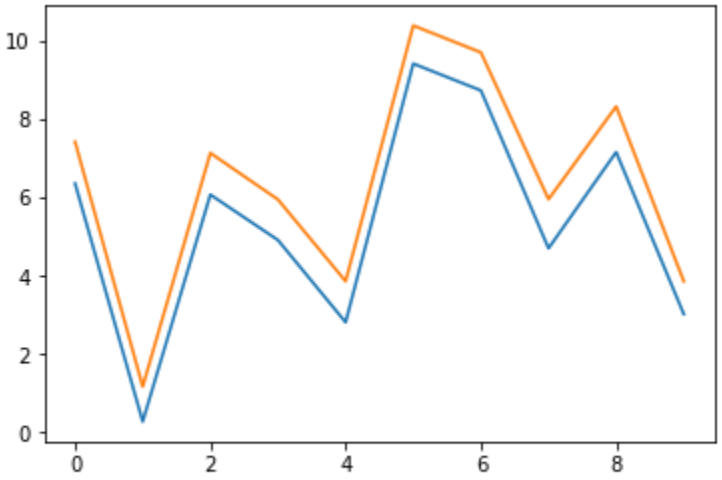
For sigma=0




```
[[1.0000000000000004, 1.9999999999999993], [4.0, 5.0], [6.038309658633295,
7.038309658633295]]
```

The results are even more accurate because of no noise.

For sigma=0.1




```
[[1.0282027225367538, 1.951049772365542], [4.004036251530454,
4.9697517653935614], [6.024706326316277, 7.0195306668601125]]
```

The results deviate by very tiny bit because of low noise.