**Computational Intelligence First assignment**

**Mohammad ali farahat**

**97521423**

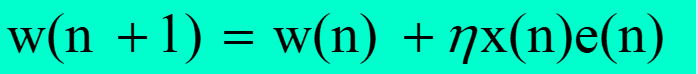
1. In this question, I created a Perceptron class that name is NOR.

This class has a method for training the perceptron, the method gets two inputs, one of them is training data and another one is the label of that data.

I used gradient descent algorithm to train my perceptron.

First, it calculates the sum of multiply of weights and inputs (dots) with NumPy. (the first weight is for bios)

After that calculates the error (difference of dots and label). Then update the weights with this formula :



We had four inputs to train a NOR perceptron:

[-1, -1] 🡪 1

[-1, 1] 🡪 -1

[1, -1] 🡪 -1

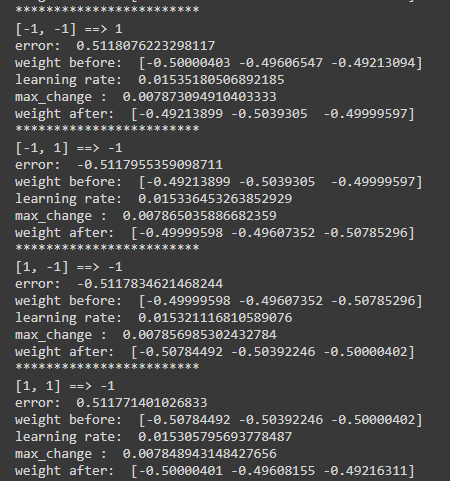
[1, 1] 🡪 -1

I use -1 instead of 0, because weights won’t update if they multiply by zero.

In each iteration, we pass all of this data to perceptron and its weights become more accurate each time.

This algorithm works until the largest weight change in one iteration becomes less than a specific limit (in this case I set the limit to 0. 000000001)

* One problem that I had was the learning rate, when learning rate was constant, the algorithm never ends. Actually, it jumps over it and never ends. So I decrease it every time learn method calls. And problem solved!
* Comments in codes shows everything clearly
* To test the program, I print some information during running, and to increase performance I commented them now , but they print something like this in each iteration:



* Finally, after 4430 iterations, the output is like below and if you run the program, you’ll see this:



1. In this question we have two class of points and we want to classify them by perceptron algorithm.

Like Q1, I design a perceptron class (I said details in last part).

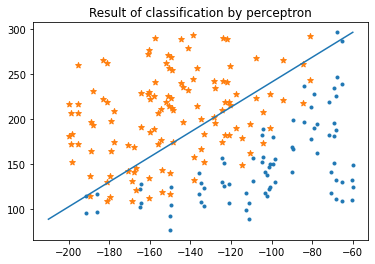
First, I open data file and read it (probably you cannot run program because you don’t have data file in correct directory), and then parse it.

I changed all 0 labels to -1.

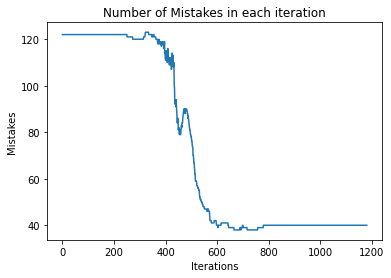
There are 200 input data and we train perceptron with them in each iteration.

In my case, after 1182 iteration algorithm stopped (because it reached maximum weight change limit) and after that I print number of iterations and two plots:

* Result of classification that have points and separator Line

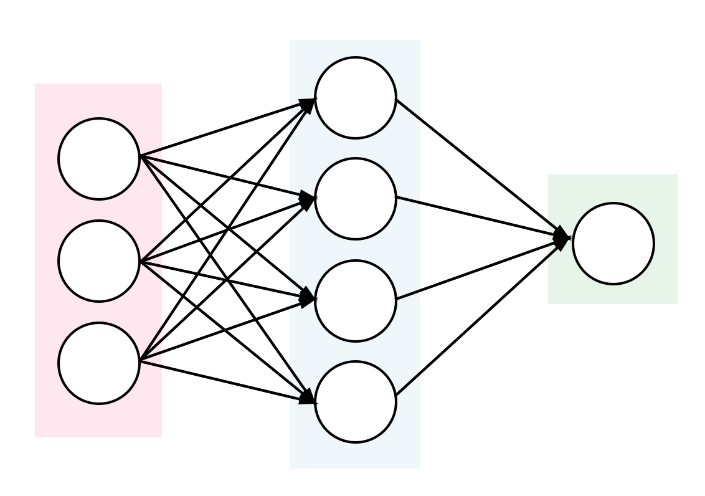


We see that perceptron works fine and separated points in 2 class

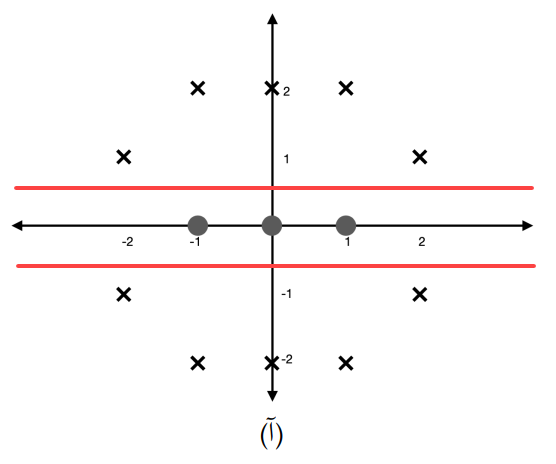
* plot of mistakes for each iteration

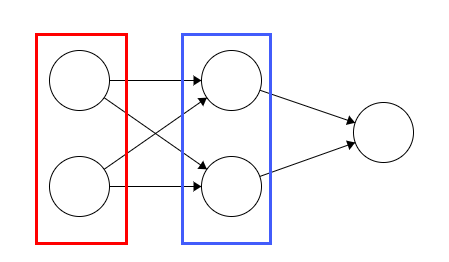
At first mistakes are 120, but after 300 iteration mistakes decreased and after about 600 iterations, they become about 40. And after that, they are stabled there.

1. Madaline is several Adaline in parallel. Madaline has one hidden layer and actually it separates one convex shape from others. Madaline has one output (like Adaline). So, if our data points are separatable by one convex shape (the lines does not have to be closed, they can be two line) , Madaline can do it.



1. **✓** this example can be solved by Madaline. neural network has two input neurons in first layer, two neurons in hidden layer (for each red line in left plot) and one final neuron for combine these two lines.





1. **✘** This example cannot solve by Madaline, because we need one convex for three inside point and one more line for three points on the right, so we need more than one hidden layer. So, Madaline cannot work here.

