Neural Network

Project 1

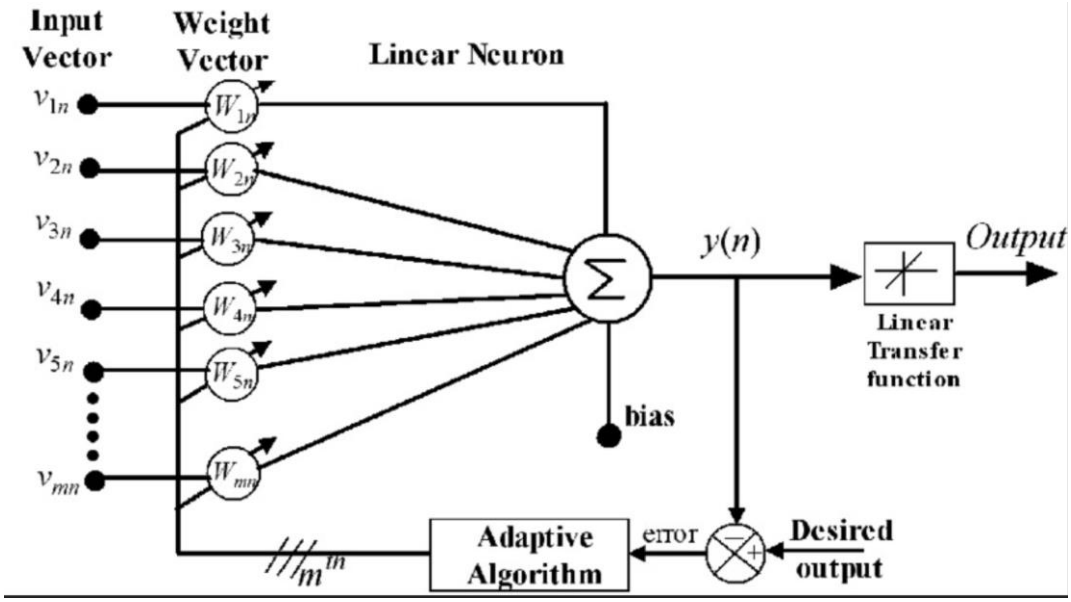
206604027 – זהראא זייד

205946221 – נטלי אלמוג

At this part of the project ,We generated 1,000-point two-dimensional random data in the dimensions< x, y> (-1, 1). Then each sample was assigned a value of "1" or "-1," with "1" being assigned if x and y are both greater than 0.5, and "-1" otherwise.

First of all what is Adaline:

ADALINE (**Adaptive Linear Neuron** or later **Adaptive Linear Element**) is an early single-layer artificial neural network and the name of the physical device that implemented this network.



Part A:

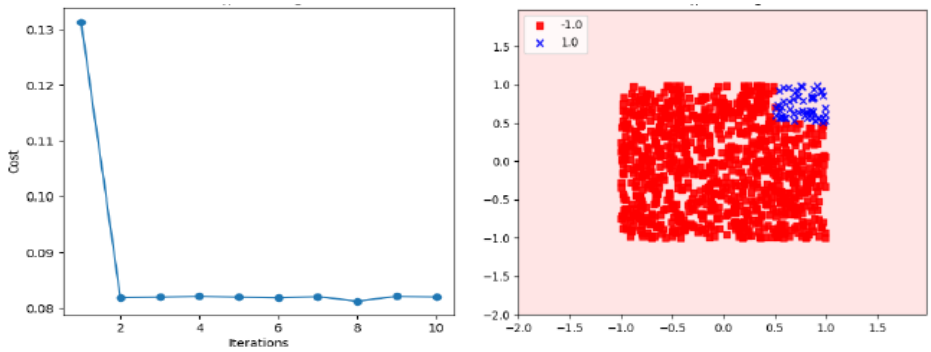
Here we compared different learning rates using data from 1000 samples:
From these figure ,We can see how important it is to choose the right learning rate.
And We'll need further repetition to determine the cost (error^2 /2) as the learning rate decreases.

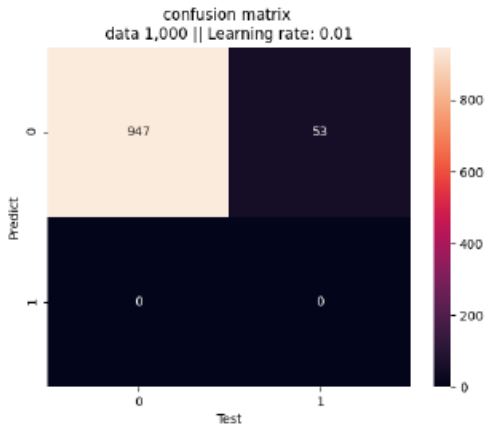
Comparison between the performance of real data and the predictions of our model : We notices that by classifying all of the data as "-1" with a learning rate of - 0.01, our model was able to estimate 96.4 percent of the time correctly. At the same time, our model got 89.1 percent correct predictions with a learning rate of –0.001 by classifying 85.6 percent as "-1" and the others as "1."

Part A : Adaline algorithm

Data :1000

Learning Rate:0.01

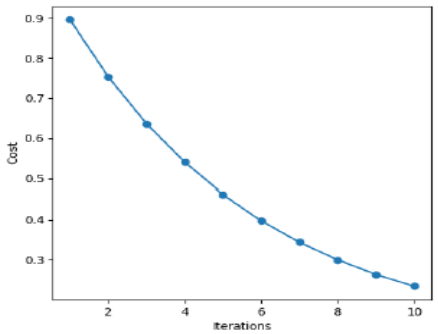


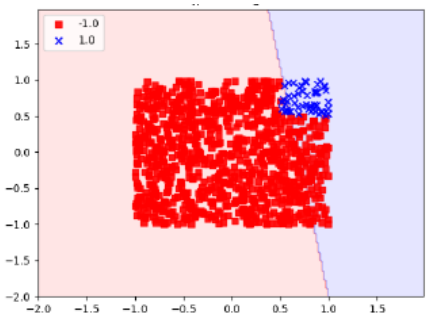


Part A : Adaline algorithm

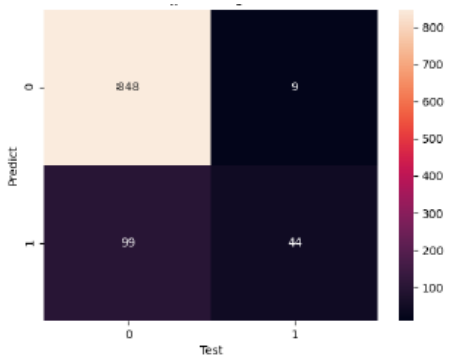
Data :1000

Learning Rate:0.0001





Confusion matrix:



* We would select a larger data set and a higher learning rate for the same number of iterations if the data is n/10,000 (with -10000 <= n <= 10,000).
For eg, with 10 iterations, if we increase our data to 10,000 instead of 1,000 and our learning rate to 0.1 instead of 0.01. then we will notice that as our distribution grows, and needs more data to approach the typical distribution. To practice on a wider variety of potential values, we'll need to improve our data and learning rate.

The score of this model is: 97.3%.

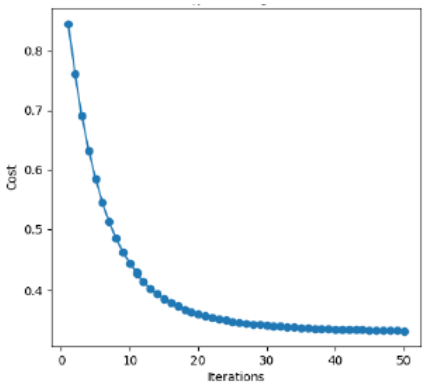
Part B:

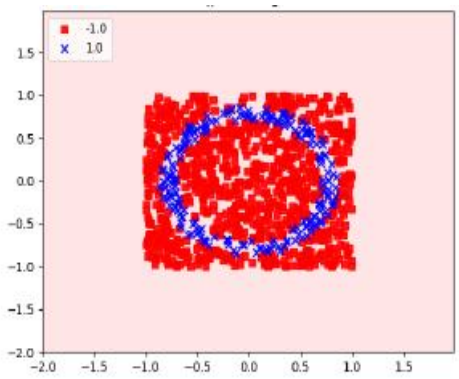
In this part we changed the problem so that points such that has value 1 only if 1/2 <= x**2 + y**2 <= 3/4. And we get the best results for an Adaline when we use data size = 100,000 instead of data size = 1,000, with an 80.16 percent performance rate. Since our model categorized all data as "-1," the findings improved as we used more data. And Since there is no normal distribution, if more data is collected, the number of values in the "-1" range increases.
Here we used a learning rate of 0.0001 and a variable data size :

Part B : Adaline algorithm

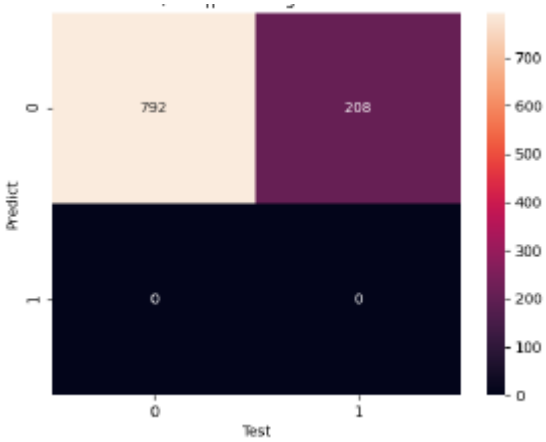
Data :1000

Learning Rate:0.0001





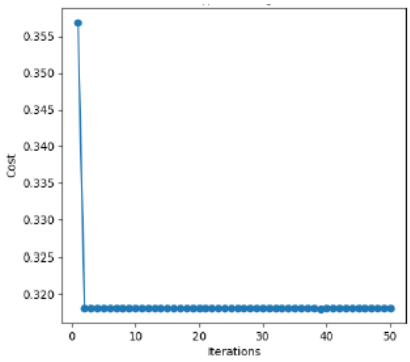
Confusion matrix:

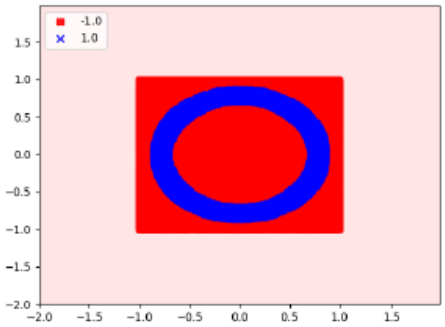


Part B : Adaline algorithm

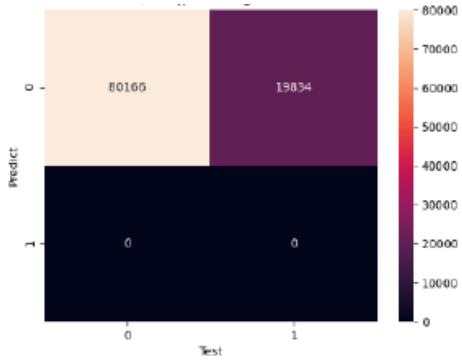
Data :100000

Learning Rate:0.0001





Confusion matrix:



At the end we concluded that adding more data to our model would not improve it. Since Adaline is a linear model, there is no way to classify values of high performance rates that are distributed non - linearly.

Finally …

I wrote the code using python language and pytorch, and here is an explanation of the functions used to implement the Adaline algo:

1- Fit():We use this function to train our model, and we update the weights for each iteration.
2- shuffle(): to adjust the training data.
3- net_input(): here we used Matrix multiplication to calculate the net input.
4- activation() : Linear activation for a static function.
5- predict() : this one is to predict the correct answer using the model , By multiplying Wi with Xi.
6- score(): gives us the success rate of our model.
7- update_weight():this is the function that we used to update the vector using LSM algo : Wi (new) = Wi (old) + a * (y – output) * Xi (when y is the desired output, a is the Learning rate and the output is sum Xi, Wi.

Note: I have to annotate that tohave a concept of how to write the code I used these repositories from github : https://github.com/Natsu6767/Adaline
https://github.com/camilo-v/Neural-Networks/blob/master/single-layer-networks/adaline/adaline.py