Neural Networks

Shrey Gupta

The input and output target vectors that were taken for this network :

(class 1)

$$\mathbf{s}^{(1)} = \begin{bmatrix} 1 & 1 & -1 \end{bmatrix},\ \mathbf{s}^{(2)} = \begin{bmatrix} 1 & 2 & -1 \end{bmatrix} \quad \text{with targets} \quad \begin{bmatrix} -1 & -1 \end{bmatrix},$$

(class 2)

$$\mathbf{s}^{(3)} = \begin{bmatrix} 2 & -1 & 1 \end{bmatrix},\ \mathbf{s}^{(4)} = \begin{bmatrix} 2 & 0 & 1 \end{bmatrix},\ \mathbf{s}^{(5)} = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix},\ \mathbf{s}^{(6)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \quad \text{with targets} \quad \begin{bmatrix} -1 & 1 \end{bmatrix},$$

(class 3)

$$\mathbf{s}^{(7)} = \begin{bmatrix} -1 & 2 & 1 \end{bmatrix},\ \mathbf{s}^{(8)} = \begin{bmatrix} -2 & 1 & 1 \end{bmatrix} \quad \text{with targets} \quad \begin{bmatrix} 1 & -1 \end{bmatrix},$$

(class 4)

$$\mathbf{s}^{(9)} = \begin{bmatrix} -1 & -1 & -1 \end{bmatrix},\ \mathbf{s}^{(10)} = \begin{bmatrix} -2 & -2 & -1 \end{bmatrix},\ \mathbf{s}^{(11)} = \begin{bmatrix} -2 & -1 & -1 \end{bmatrix} \quad \text{with targets} \quad \begin{bmatrix} 1 & 1 \end{bmatrix}.$$

Mean Square error defined as :

$$\frac{1}{Q}\sum_{q=1}^{Q}\sum_{j=1}^{M}\left(y_j - t_j^{(q)}\right)^2,$$

Initially weight and bias are kept to be 0. ADALINE is used to create the neural network and is trained with the help of delta(LMS) rule. According to the delta rule, one of the ways to achieve convergence is that the learning rate is kept as (1/k) where k is the step number . This helps in producing a more optimal set of weights and biases. I have used the same method to get gradually decreasing alpha and used it to get weights and bias. The accuracy for the same is calculated as well as the least mean square error. The number of steps for which the training is done in my case is 900.

The output when alpha is considered to be gradually decreasing is:

Accuracy at epoch : 0 is : 45.45454545454545 and alpha = 1.0

Accuracy at epoch : 1 is : 45.45454545454545 and alpha = 0.5

Accuracy at epoch : 2 is : 45.45454545454545 and alpha = 0.3333333333333333

Accuracy at epoch : 3 is : 36.36363636363637 and alpha = 0.25

Accuracy at epoch : 4 is : 0.0 and alpha = 0.2

Accuracy at epoch : 5 is : 27.27272727272727 and alpha = 0.16666666666666666

Accuracy at epoch : 6 is : 18.181818181818183 and alpha = 0.14285714285714285

Accuracy at epoch : 7 is : 45.45454545454545 and alpha = 0.125

Accuracy at epoch : 8 is : 0.0 and alpha = 0.1111111111111111

Accuracy at epoch : 9 is : 0.0 and alpha = 0.1

Accuracy at epoch : 10 is : 18.181818181818183 and alpha = 0.09090909090909091

Accuracy at epoch : 11 is : 36.36363636363637 and alpha = 0.08333333333333333

Accuracy at epoch : 12 is : 36.36363636363637 and alpha = 0.07692307692307693

Accuracy at epoch : 13 is : 36.36363636363637 and alpha = 0.07142857142857142

Accuracy at epoch : 14 is : 27.27272727272727 and alpha = 0.06666666666666667

Accuracy at epoch : 15 is : 27.27272727272727 and alpha = 0.0625

Accuracy at epoch : 16 is : 27.27272727272727 and alpha = 0.058823529411764705

Accuracy at epoch : 17 is : 27.27272727272727 and alpha = 0.05555555555555555

Accuracy at epoch : 18 is : 27.27272727272727 and alpha = 0.05263157894736842

Accuracy at epoch : 19 is : 27.27272727272727 and alpha = 0.05

Accuracy at epoch : 20 is : 27.27272727272727 and alpha = 0.047619047619047616

Accuracy at epoch : 21 is : 27.27272727272727 and alpha = 0.045454545454545456

Accuracy at epoch : 22 is : 27.27272727272727 and alpha = 0.043478260869565216

Accuracy at epoch : 23 is : 27.27272727272727 and alpha = 0.041666666666666664

Accuracy at epoch : 24 is : 27.27272727272727 and alpha = 0.04

Accuracy at epoch : 25 is : 27.27272727272727 and alpha = 0.038461538461538464

Accuracy at epoch : 26 is : 27.27272727272727 and alpha = 0.037037037037037035

Accuracy at epoch : 27 is : 27.27272727272727 and alpha = 0.03571428571428571

Accuracy at epoch : 28 is : 36.36363636363637 and alpha = 0.034482758620689655

Accuracy at epoch : 29 is : 63.63636363636363 and alpha = 0.03333333333333333

Accuracy at epoch : 30 is : 72.72727272727273 and alpha = 0.03225806451612903

Accuracy at epoch : 31 is : 100.0 and alpha = 0.03125

Accuracy at epoch : 32 is : 100.0 and alpha = 0.030303030303030304

Accuracy at epoch : 33 is : 100.0 and alpha = 0.029411764705882353

Accuracy at epoch : 34 is : 100.0 and alpha = 0.02857142857142857

. . . .

Accuracy at epoch : 896 is : 100.0 and alpha = 0.0011148272017837235

Accuracy at epoch : 897 is : 100.0 and alpha = 0.0011135857461024498

Accuracy at epoch : 898 is : 100.0 and alpha = 0.0011123470522803114

Accuracy at epoch : 899 is : 100.0 and **alpha =** 0.0011111111111111111

The least mean square after applying activation is : 0.0

Weight : [[-0.61014064 0.08106886]

 [0.00442896 -0.62135899]

 [0.01646772 0.08778964]]

Bias : [[-0.14799343 0.21656197]]

Least Mean Square Error at the end : 0.35238745853691705

The number of cycles after which 100% accuracy was achieved : 30

Total Cycles performed : 900

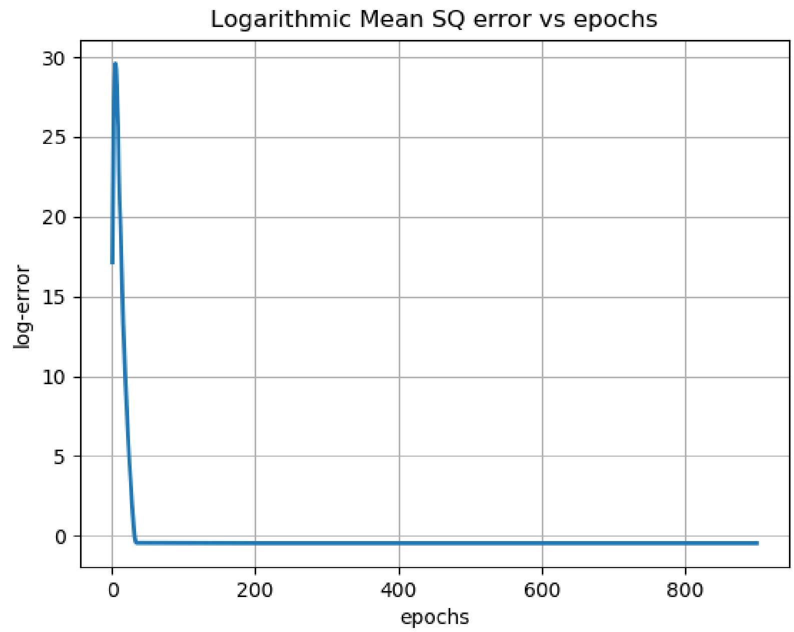
Value of alpha at end : 0.0011111111111111111

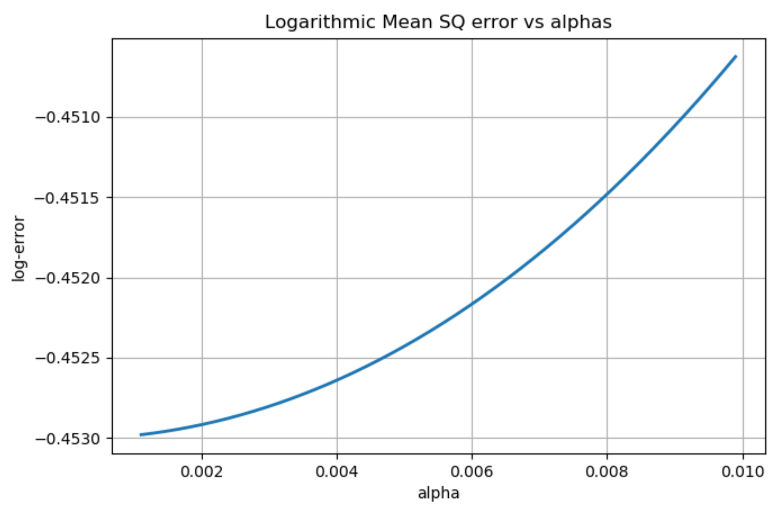
A 100% accuracy and least mean square(with activation) of 0 indicates that all the patterns were classified correctly.

Even though a 100% accuracy was achieved at cycle number 31, I still continued training the model because :

- Robustness needs to be maintained in the neural network. It should be able to classify the vectors which are outside the training set. For that more training is required to get a more optimal choice of weights and bias.
- The least mean square error is used to check whether the weights and bias are optimal or not. The cycles are continued so that the error can be reduced and the alpha value is chosen having the least least mean square error. If more training was performed , the value of error might have been lower, but the difference would be negligible and more training might lead to overfitting the model. The results of the lower least mean square error against the number of steps is shown with the help of graphs which were plotted in the code.

For convenience, I included the errors in a logarithmic function so that the plot is visually better.





Plot of Log Mean Square error against the decreasing Alphas

In the case when the 11th vector value is changed and increased , there is a huge change observed in the value of Least Mean Square error. When the training vector becomes [12 -1 -1] , that is when the value of x becomes 12. In this case the Least mean square error increases exponentially which can be shown when the second function of the code is called. The following error values are produced for the respective value of x and a graph is also shown the exponential increase in the error.

Error for [-2 -1 -1] : 0.35238745853691705

Error for [1 -1 -1] : 0. 5397048591389599

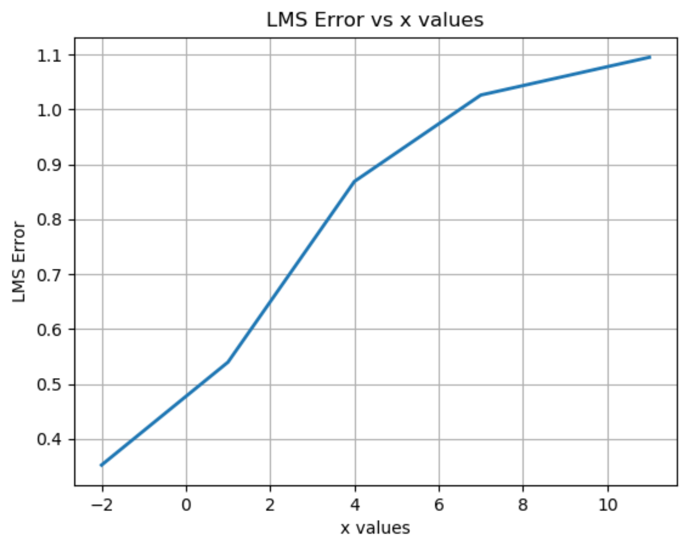
Error for [4 -1 -1] : 0. 8686685623291157

Error for [7 -1 -1] : 1. 0259765522455342

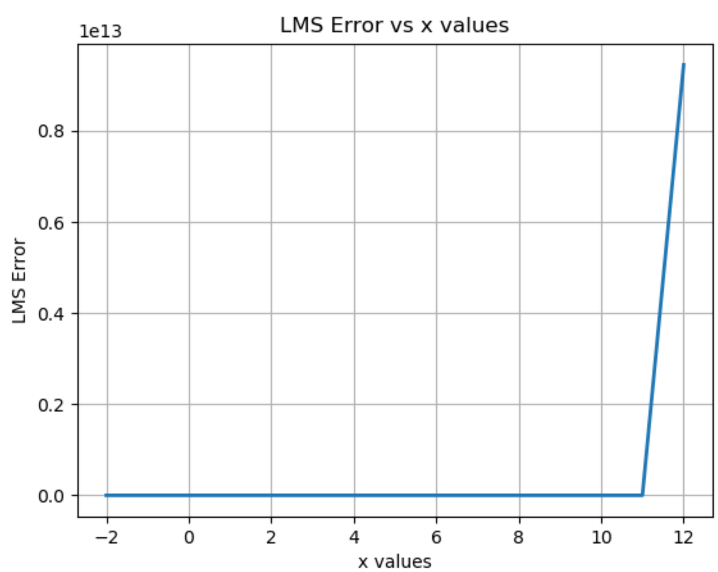
Error for [11 -1 -1] : 1. 094722310966957

Error for [12 -1 -1] : 9447085599473. 123

The change in error before [11 -1 -1] is very small and in the figures between 0 and 1. This is shown graphically as :



However, as soon as x becomes 12, the error exponentially increases . It can also be seen graphically :



Hence , one major noteworthy change is the change in the least mean square

The accuracy also starts dropping when we start increasing x. It drops with a lower value in the initial increase of x , however when the value of x reaches 12, the accuracy drops down to 18%. The accuracy for each of the values of x are :

Accuracy for [-2 -1 -1] : 100

Accuracy for [1 -1 -1] : 90.9

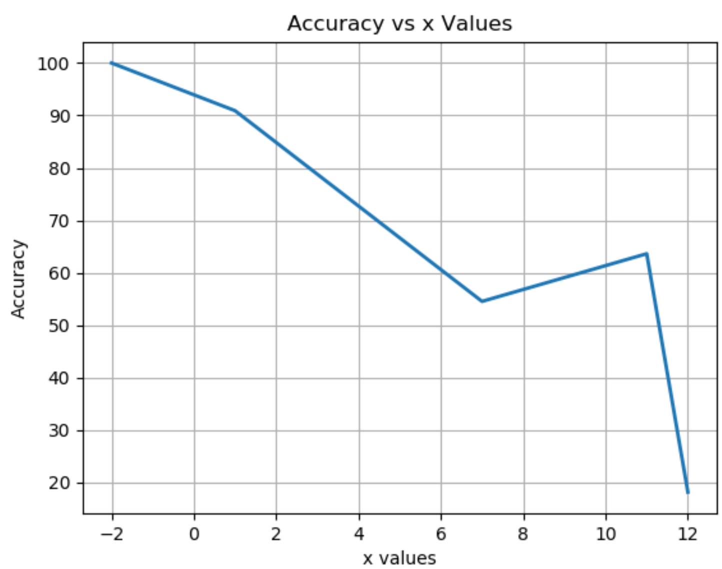
Accuracy for [4 -1 -1] : 72.72

Accuracy for [7 -1 -1] : 54.54

Accuracy for [11 -1 -1] : 63.63

Accuracy for [12 -1 -1] : 18.18

Graphically it can be shown as:



Hence this is the second noteworthy change which is observed.

These changes happen when we increase the value of x. This is because the value of x becomes too high in comparison to the features of the other vectors in the training set. This affects the weights and bias. This is because it starts getting more weightage in the calculation of weights and bias. Also, since the error calculated is the squared error rather than the absolute error, the squaring further contributes to the exponential increase of the LMS error value. If the error was absolute error rather than the squared error, the graph of error against the x value would not have shooten up this exponentially.

Also, the number of epochs are set according to a different training data. For this type of training data, maybe a different alpha value is required and more number of epochs are required to train the model as the data changes drastically from the original training data. The optimal choice of convergence of the previous training data might not be the optimal choice for this set of training data.