Analysis of the Impact of Learning Rate and Time Dependence on a Linear Neural Network

Powers-Luhn, J.R.

I. Introduction

An investigation was conducted into the ability of a single-layer neural network to model a linear equation.

II. Methods

A. Training Rate

A MATLAB script was written to iteratively train a neural network. The network had an input layer of two neurons, x_1 and x_2, a bias neuron, and a single output neuron. The network was trained with labeled data (as appropriate for a regression problem). One hundred input vectors (of two values each) were generated in the range $[0, 1]$ using the MATLAB `rand` command with a seed of 10. Output values were set to $y = 2x_1 - 3x_2 - 1$. The network was trained for 100 epochs, with error and weights recorded after each epoch. The learning rate was varied from 0.1 to 0.9.

B. Time Dependence

In order to test the impact of a time-dependent factor on the training, a loop was added to the code and the network output was trainined to match the function $y = 2x_1 - 3x_2 - 1 + 0.02i$ with i varying from 1 to 100 in steps of 1. The same learning rates were applied as in the Training Rate section above. As before, the value of the weights, bias, and error were recorded at each iteration.

III. Results

A. Training Rate

It was expected that the weights should converge on the coefficients of the input terms x_1 and x_2. In the cases where the learning rate was low (see figs 1, 2, 3, 4)

With higher learning rates in place, the weights converged as expected (see figs 5, 6, 7, 8). It was theorized that the weights would have converged even with slower learning rates had more training epochs been run.

B. Time Dependence

The introduction of a time dependent factor (see figs 9, 10, 11, 12) complicated the learning process for the network, as expected. Networks with low learning rates were unable to converge on the solution and oscillated about the correct value.

With higher learning rates in place, the weights converged (see figs 13, 14, 15, 16), though not as closely as they had in the case of the time-independent function. The convergence time was similar in each case, around 30 epochs. It is theorized that after this time the network was sufficiently well trained to allow for only small corrections, thereby giving the appearance of a well-trained network.

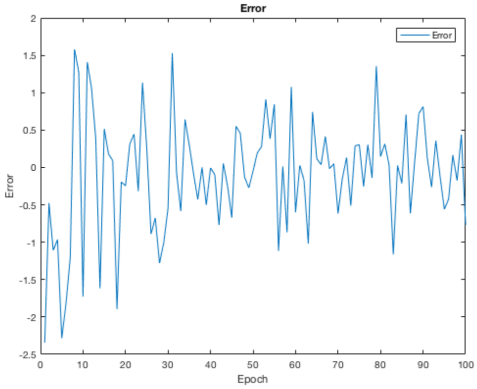

Fig. 1. Error for each training epoch, learning rate = 0.1

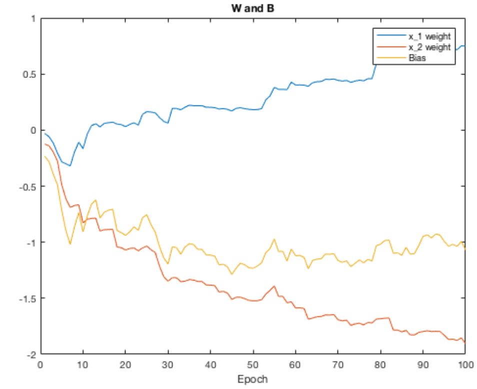

Fig. 2. Weights for each training epoch, learning rate = 0.1

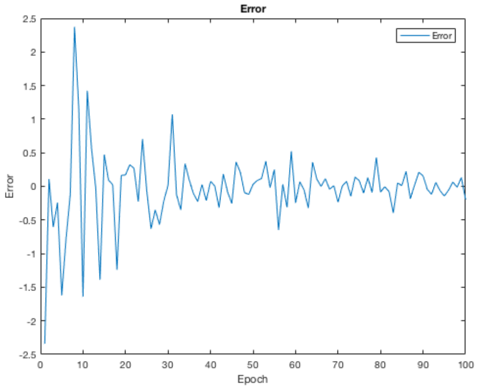

Fig. 3. Error for each training epoch, learning rate = 0.3

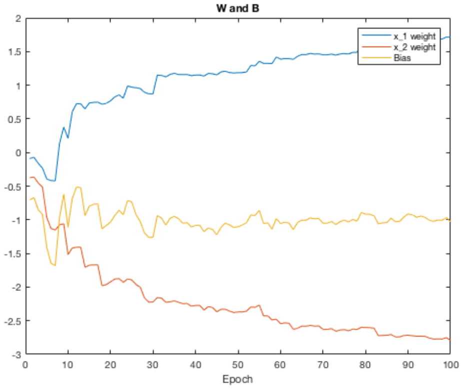

Fig. 4. Weights for each training epoch, learning rate = 0.3

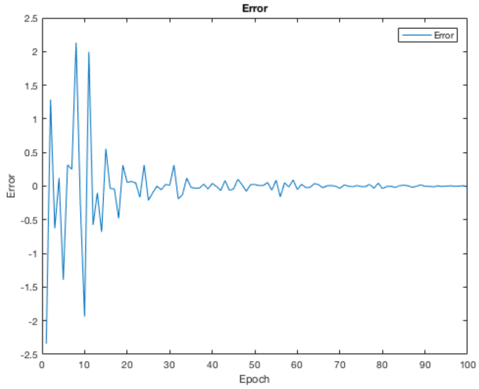

Fig. 5. Error for each training epoch, learning rate = 0.7

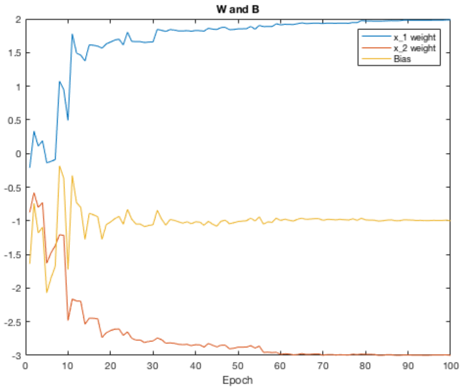

Fig. 6. Weights for each training epoch, learning rate = 0.7

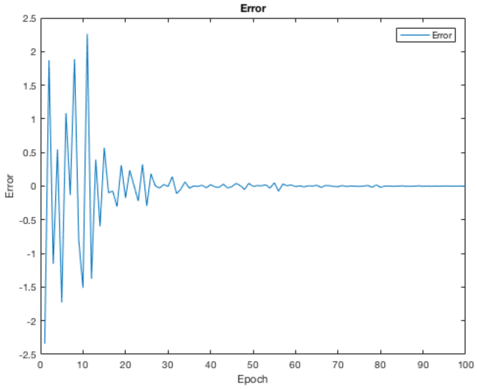

Fig. 7. Error for each training epoch, learning rate = 0.9

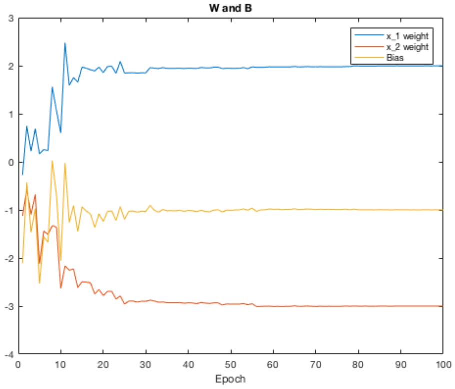

Fig. 8. Weights for each training epoch, learning rate = 0.9

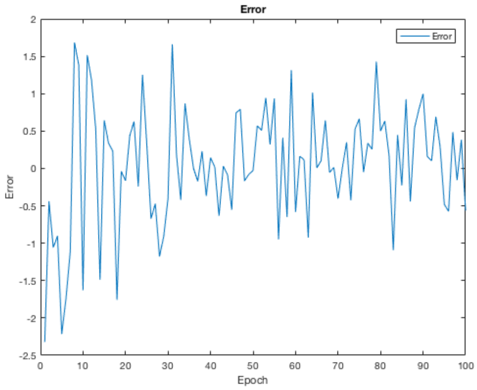

Fig. 9. Error for each training epoch (time-dependent), learning rate = 0.1

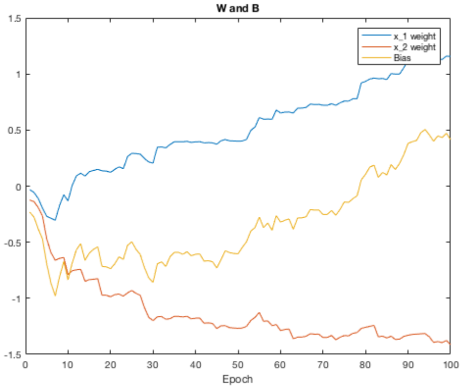

Fig. 10. Weights for each training epoch (time-dependent), learning rate = 0.1

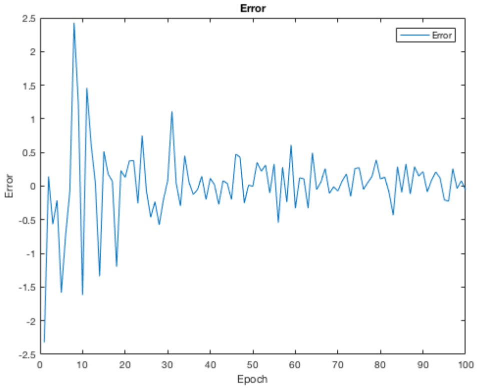

Fig. 11. Error for each training epoch (time-dependent), learning rate = 0.3

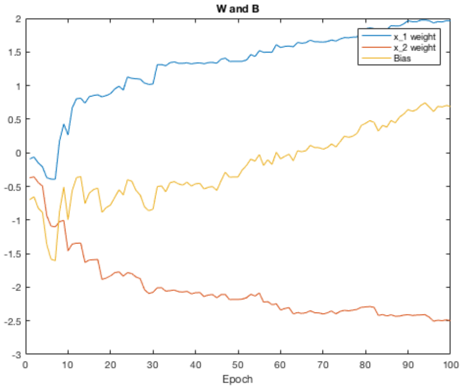

Fig. 12. Weights for each training epoch (time-dependent), learning rate = 0.3

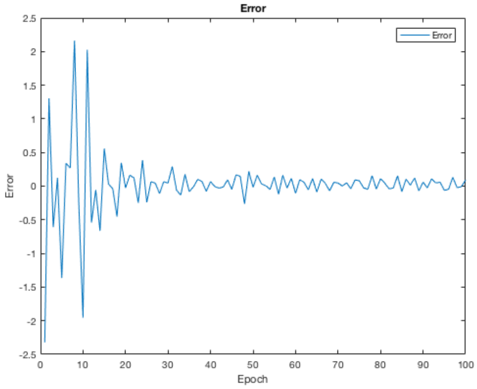

Fig. 13. Error for each training epoch (time-dependent), learning rate = 0.7

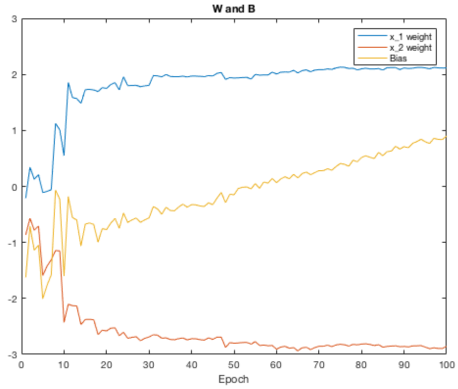

Fig. 14. Weights for each training epoch (time-dependent), learning rate = 0.7

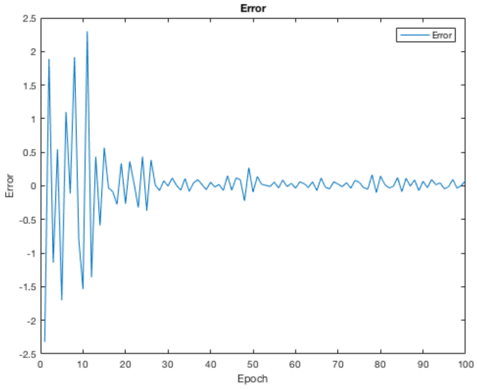

Fig. 15. Error for each training epoch (time-dependent), learning rate = 0.9

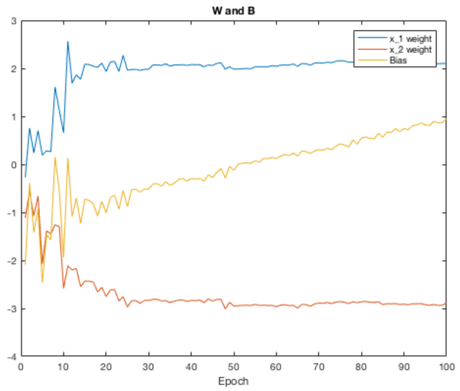

Fig. 16. Weights for each training epoch (time-dependent), learning rate = 0.9

IV. Conclusions

An analysis of the impact of learning rates on the ability of a neural network to model a linear function was performed. For short training times, learning rates of 0.3 or less were insufficient to accurately model the function studied. Higher learning rates converged more quickly, and modeled the function well.

The network was less capable of modeling a time-dependent function. The change in the equation with each time step meant that the low learning rate networks suffered even more, and were not able to converge on the correct weights. Higher learning rates helped this problem, though the errors never fell to the levels observed in the time-independent problem.