COSC424 Project 1

Eli Carter and Gabe Stowe

1/30/23

1 Introduction

Artificial neural networks (ANNs) have many use cases today. When people code ANNs, they often use pre-built libraries. This allows the use of very complicated ANNs without the hassle of having to build the functions yourself. Unfortunately, since people have not created their own functions, they may not understand exactly how ANNs work since they just deal with abstractions.

This project implements a simple ANN library with the purpose of increasing our understanding of key deep learning concepts like feed-forward, gradient descent, and back propagation. This will be done by implementing a few classes to organize the ANN and functions that allow it to perform learning. Once completed arbitrary networks should be able to be formed and simple learning tasks solved.

2 Code Assumptions and Choices

The code is split up into three classes per the instructions. A neuron, fully connected layer, and neural network class are implemented. Some small assumptions were made when implementing the code. They are 1) only a linear and logistic activation functions will be used 2) only the sum of square errors loss function will be used.

3 Problems and Issues

For the undergrad portion of the project, there are no problems to report. The example code runs smoothly and gets the same weights and biases as expected.

We attempted the AND and XOR as well. These mostly work, though XOR can have some troubles training depending on what the weights are initialized to.

4 How to Run the Code

The code runs the same as described in the requirements. "Example" runs the example set up and trains it for one pass through then prints the weights. To see the output of the example network for 1000 train cycles to compare it to the class example, run the code with no command line arguments.

The XOR and AND function are better general case examples of how to run the code and each of their first 5 lines show how to set up lists to feed to the network if a different architecture is wanted.

5 Results and Conclusion

The ANN is a mystical being in the world of analytics and this project has allowed us to gain a understanding of how they really work by implementing a bare-bone example ourselves. We now have a much better feel for what propagation does and how different parameters like learning rate will affect learning.

In Fig. 1, loss is shown as the network learns the example output as a function of training epoch. The higher learning rates race to a lower loss quicker than that of the lower learning rates. This fits into the intuition we have built since the updates of the weights is being being larger in the higher learning rates. Luckily for this example we are just trying to learn one point, so there is no downside to making the learning rate higher and higher. In other examples with larger data sets, this may prove to be a bad strategy if some data points lead us farther astray from the global optimal weights

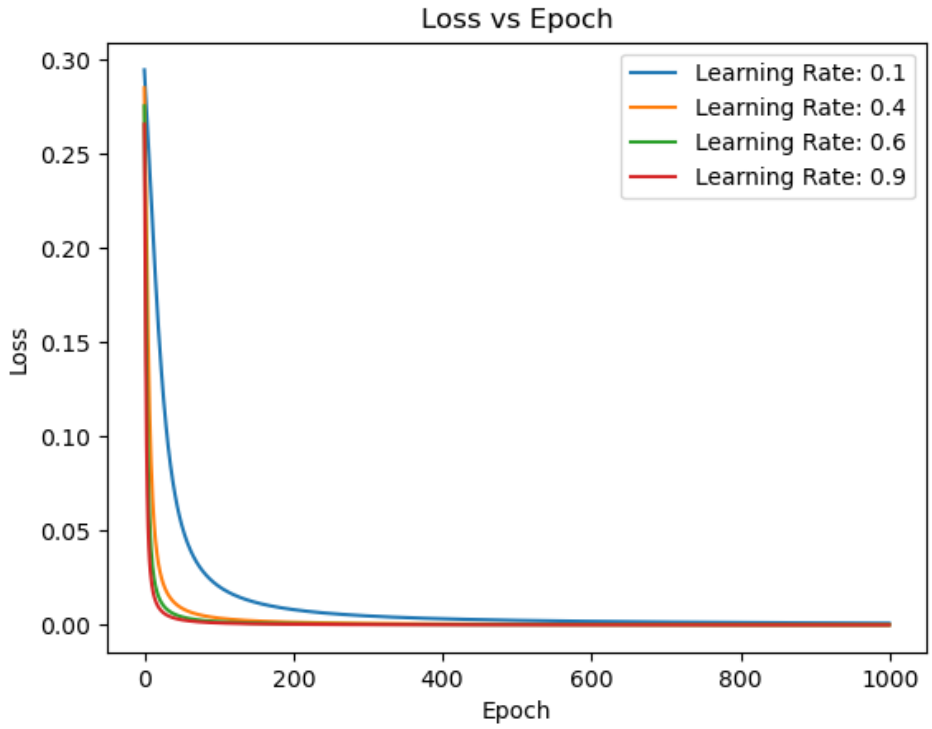

Figure 1: Loss vs Epoch for 1000 Epochs