Project 1: Artificial Neural Networks

Ian Cox and Tanner Mengel

February 10, 2023

1 Introduction

Neural networks use linear algebra to predict an output depending on a given input. Using a desired output, an error can be calculated from the output of the neural network. This error is then used to adjust the weights in the network through the back-propagation algorithm. The goal of this project is to build a neural network which takes in multiple different types of inputs and predicts specific outputs. This network is then trained by calculating the error and back-propagating over a number of iterations, called epochs. Specifically, we wanted to train an example problem along with the AND and XOR gates. Below is a table of the inputs and desired outputs for the problems.

Problem	Input	Output
example	[0.5,0.1]	[0.01,0.99]
and	$[0, 0]$	0
	$[0, 1]$	0
	$[1, 0]$	0
	$[1, 1]$	1
xor	$[0, 0]$	0
	$[0, 1]$	1
	$[1, 0]$	1
	$[1, 1]$	0

Table 1: Inputs and Outputs.

2 Assumptions

In this project we allow the user to specify the learning rate they want to use and the problem they want to run. Due to this, we assume that the ”example” problem will only be ran for 1 epoch and will print out the weights. Also, the ”and” and ”xor” problems run for 100000 epochs before giving their loss.

3 Issues

Handling the bias throughout the network was difficult between the layers and neutrons. This was handled by allowing the class NeuralNetwork to append 1 onto the input vector and define the randomly assigned weight matrices with the bias included. Maintaining conversion between numpy arrays and lists of numpy arrays throughout the training process was a challenge. No problems remain unsolved in the final version.

4 How to run

The code can be ran using python from the command line using,

```
python3 ANN.py [learning_rate] [example]
```

Where “[ANN.py]” is the name of the file being used, “[learning_rate]” is given as a float number, defaulted to 0.1, and “[example]” is the name of the problem which is desired to be ran, “example”, “and”,“xor”.

Then “ANN.py” file can be accessed from the zipped tar file using,

```
tar -xzf project1_code.tar.gz
```

5 Results

Below are the results showing how changing the learning rate effects the change in loss over a set number of epochs for the “And” gate problem.

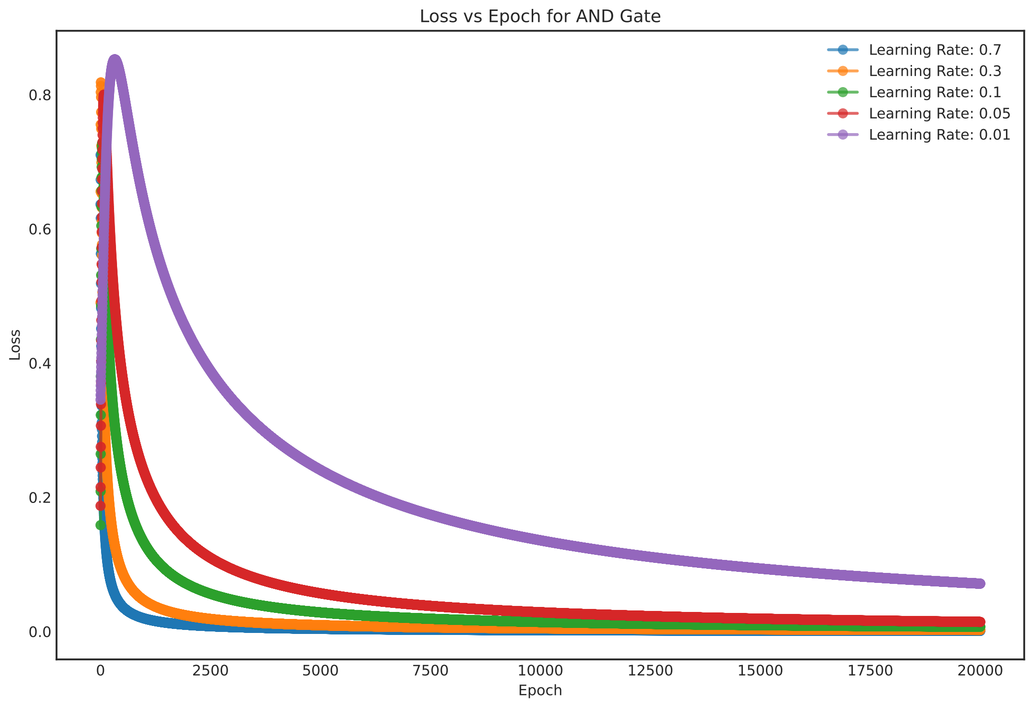

Figure 1: AND gate training for various learning rates.