Matthew Walters

Steven Koprowicz

Neural Networks

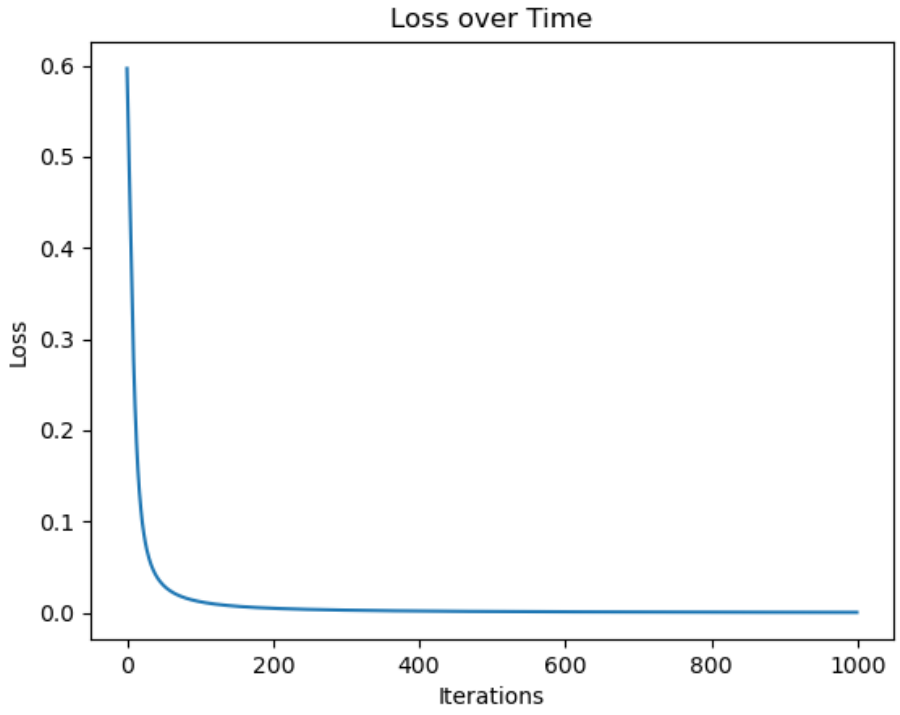
1. In this lab, we were tasked with writing the code for a neural network, including calculating an output based on the weights, which could be given or randomly generated, and perform back propagation in order to update the weights and better approximate the problem.

2. At first, we assumed that the network would have layers that each had the same number of neurons, which made our programming easier, but when it came to the xor problem, we could not build a network with only one hidden layer that could solve it. We had to go back and fix our function that calculated the deltas for each neuron in a layer, so that it returned a list the size of the previous layer, not a list the size of the current layer.

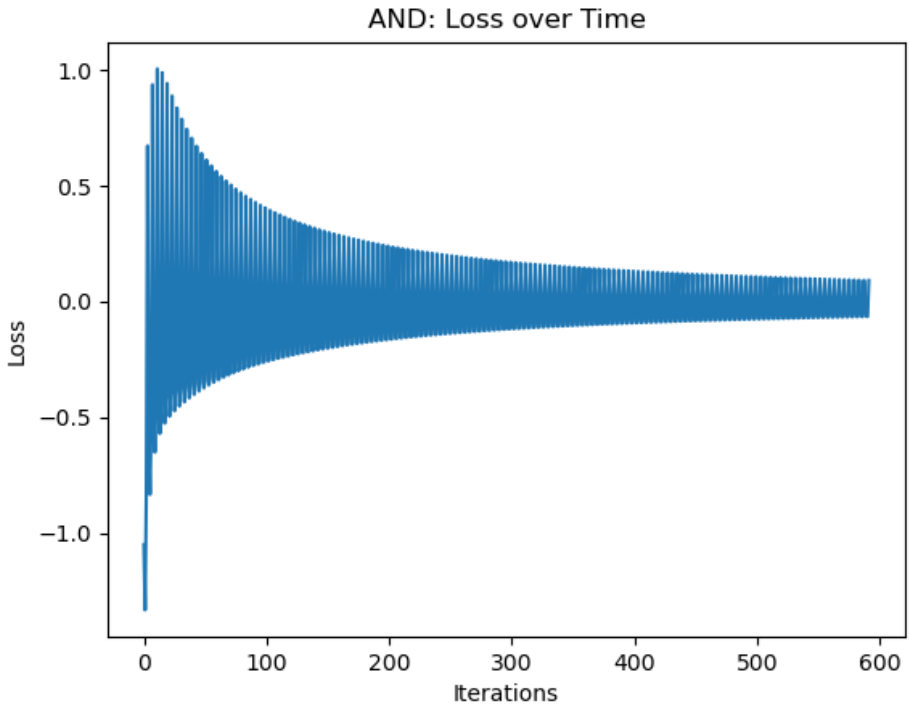
3. In the end, we were able to implement everything.

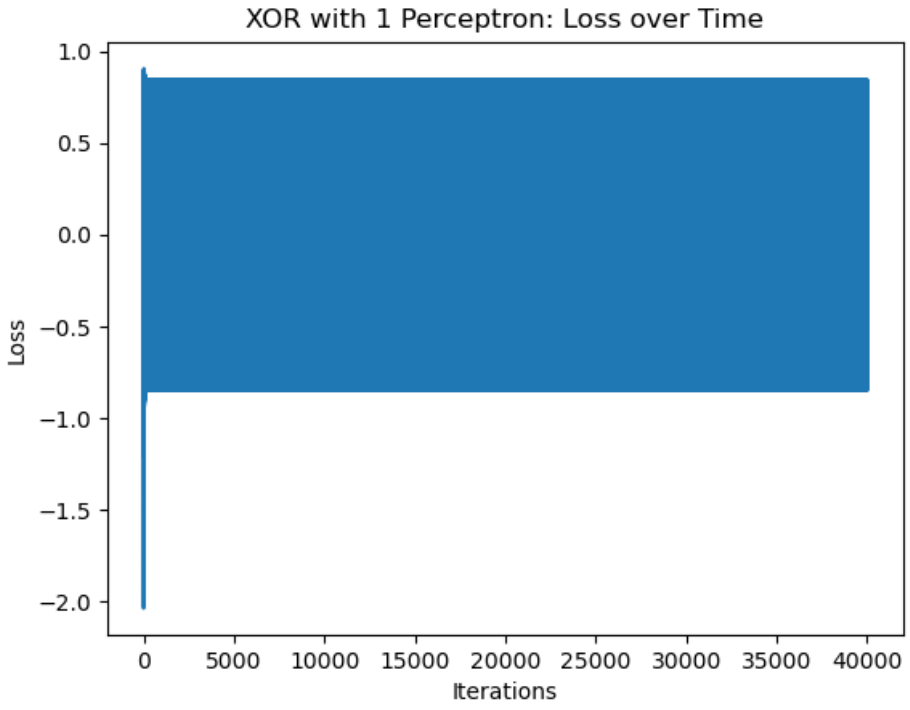
4. Our code runs as the original skeleton intended, all you have to do to learn AND or XOR or the example problems is “python lab1.py [and/xor/example]”.

5. I plotted the example problem as the loss decreased over 1,000 iterations, here are the results.

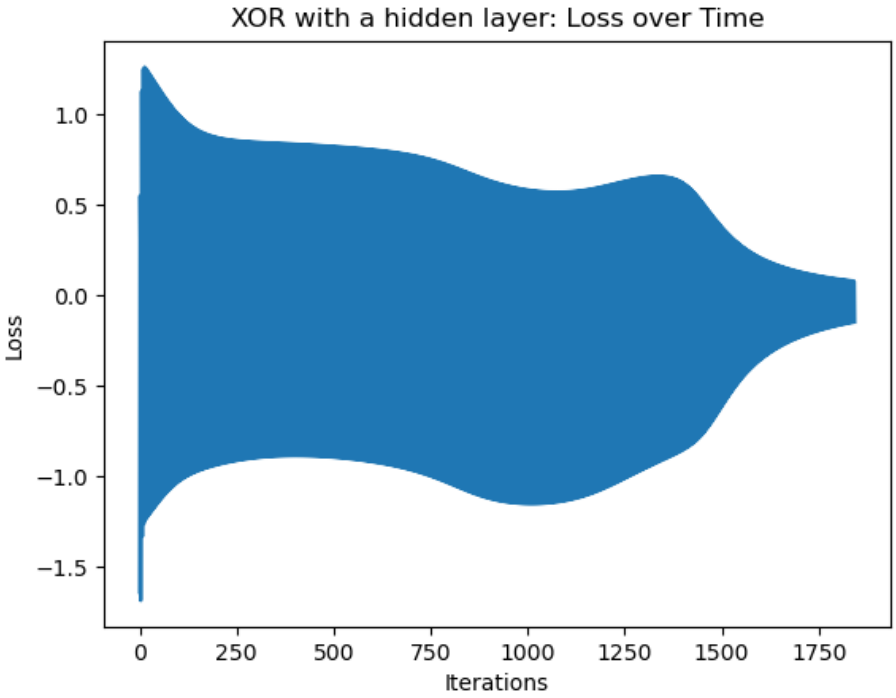


Here's a graph of the AND and XOR functions as well





Clearly with 1 perceptron this will not converge.


XOR with a hidden layer: Loss over Time
1.0
0.5
0.0
−0.5
−1.0
−1.5
Loss
0
250
500
750
1000
1250
1500
1750
Iterations