Project 1 - Artificial Neural Networks

Anna-Maria Nau (anau), Christoph Metzner (cmetzner)
COSC 525 - Deep Learning
University of Tennessee, Knoxville

January 2020

1 Introduction

The aim of this project is to implement an artificial neural network (ANN) in an object oriented manner. The library contains three classes - one to create the neuron objects ("Neuron"), one to create the fully connected layer objects ("FullyConnectedLayer"), and one to create the neural network object ("NeuralNetwork"). The library supports both the logistic and linear activation function, as well as the mean-squared error and binary cross entropy loss function.

2 Assumptions and Choices

The neural network designed to solve the XOR logic has the following structure: one input layer with two neurons, one hidden layer with two neurons, and one output layer with one neuron. Both the hidden and output layer use the logistic activation function to transform input to output accordingly.

3 Problems and Issues

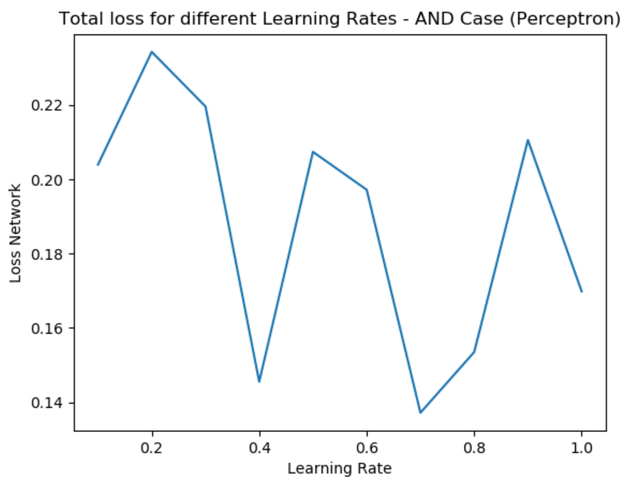
The main issue encountered throughout this project was implementing the ANN in an object-oriented manner since both of us had hardly any experience with this programming paradigm. The hardest part was knowing where to implement certain methods and understanding how the individual objects could interact with one another to produce the desired results. For example, the implementation of the back-propagation algorithm required the interaction of objects from different classes and different objects of the same class.

4 Running the Code

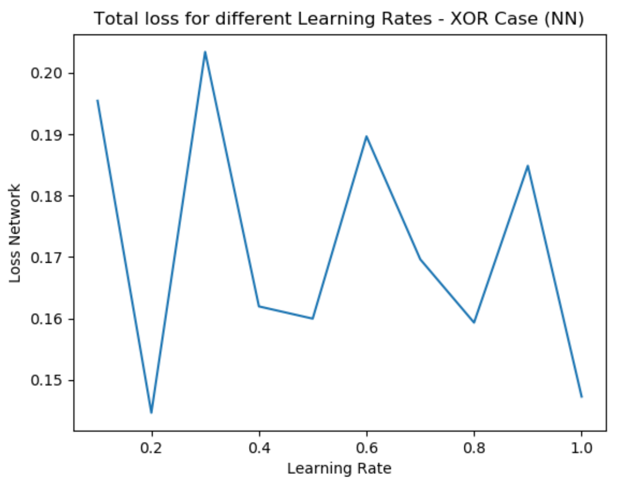
The `main` method takes the command line variables: `example`, `and`, and `xor` to train different networks. Example command to call the script with one argument:

```
python3 dl_525_proj1.py "example"
```

5 Plots: Loss vs. Learning Rate



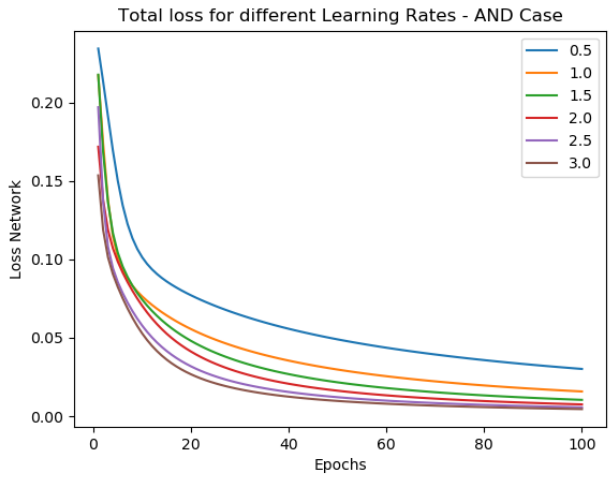
(a) Perceptron with AND-Gate



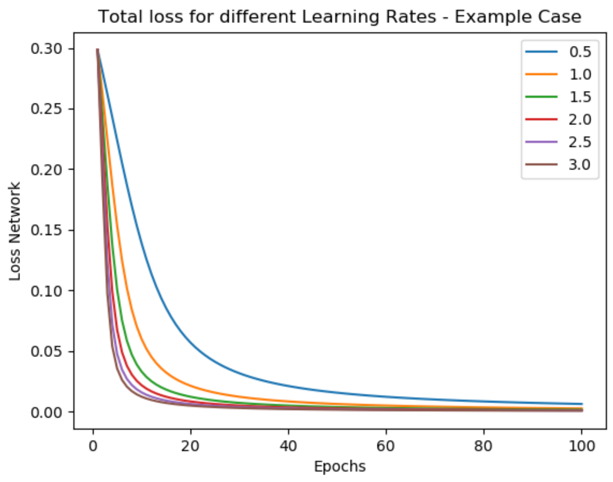
(b) Neural Network with XOR-Gate

Figure 1: Loss vs the learning rate for the perceptron using the logistic activation function applying the a) AND-Gate and b) a neural network with a XOR-Gate.

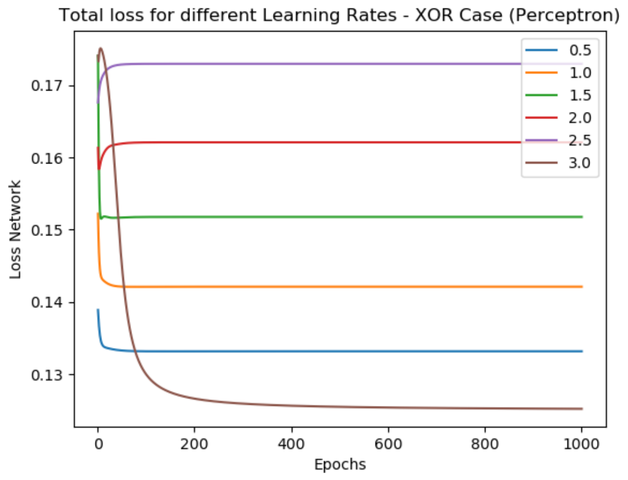
Figure 1. shows the total loss of two models for varying learning rates. The single perceptron (plot (a)) trained to produce the AND logic gate achieved the lowest loss value with a learning rate of around 0.7 and the highest loss value, around 0.2. In this case, 0.2 was too small not allowing the model to swiftly converge and requiring more training epochs. The ANN trained to represent the XOR logic gate achieved its lowest loss value around 0.2 and its highest, around 0.3. Plot (b) also shows that a learning rate above 0.2 may have been too large causing the gradient descent to overshoot the minimum and converge to a sub-optimal solution instead.



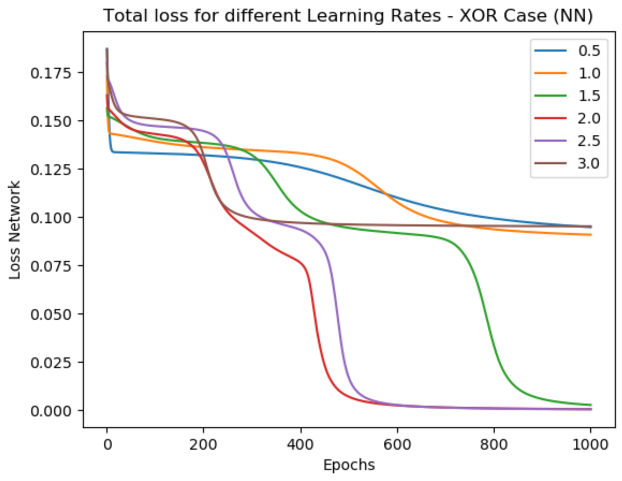
(a) Perceptron with AND logic gate: Loss vs epochs (n=100) for different learning rates.



(b) Neural network with architecture 2 input neurons, 2 hidden layer neurons, 2 output layer neurons: Loss vs epochs (n=100) for different learning rates.



(c) Perceptron with XOR logic gate: Loss vs epochs (n=1000) for different learning rates.



(d) Neural network with architecture 2 input neurons, 2 hidden layer neurons, 1 output layer neurons: Loss vs epochs (n=1000) for different learning rates.

Figure 2: Loss vs epochs for different learning rates (0.5 to 3.0) for different setups; a) Perceptron with AND logic gate, b) neural network for the example case, c) perceptron for XOR logic gate, and d) neural network for XOR logic gate.

Figure 2. shows the accrued loss over n-epochs for four different network architectures. Figures (a) and (b) show that the accrued loss decreases faster, the higher the learning rate becomes. Plot (a) and (b) show how a model's loss decreases with increasing number of training epochs for varying learning rates. Plot (c) shows how a single perceptron is not able to learn the XOR logic as indicated by the different curves. These curves stagnate at a specific loss value after roughly 50 epochs. Adding a hidden layer overcomes the problem of a single perceptron not being able to learn the XOR logic gate problem, as shown by plot (d), where the total loss is able to decrease (to 0 for certain learning rates) as the number of training epochs increases.