Problem Set 3 : Neural Networks Report

Aman Dongre

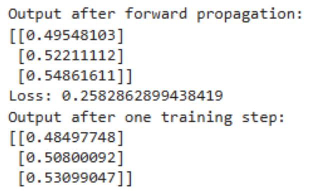
November 2024

1 Introduction

In this problem set, we will explore the foundational concepts of neural networks, covering basic architecture, activation functions, forward propagation, regularization, and adaptive learning. We will also complete the coding exercises to implement neural network components and train a basic neural network.

2 3 Layer Neural Network

Implementation of a 3-Layer Neural Network: A neural network is composed of layers of neurons: input layer, hidden layer(s), and an output layer. Each neuron receives inputs, processes them, and passes the result to the next layer. With the help of numpy on VScode, the neural network is built and the outputs are shown below:



```
Output after forward propagation:
[[0.49548103]
 [0.52211112]
 [0.54861611]]
Loss: 0.2582862899438419
Output after one training step:
[[0.48497748]
 [0.50800092]
 [0.53099047]]
```

Figure 1: 3 Layer Neural Network

This is a basic, minimal implementation to give you a foundational understanding of how a neural network is structured and how it performs forward propagation and back propagation.

3 Activation Functions (Plots)

Outputs for the Activation functions are given below:

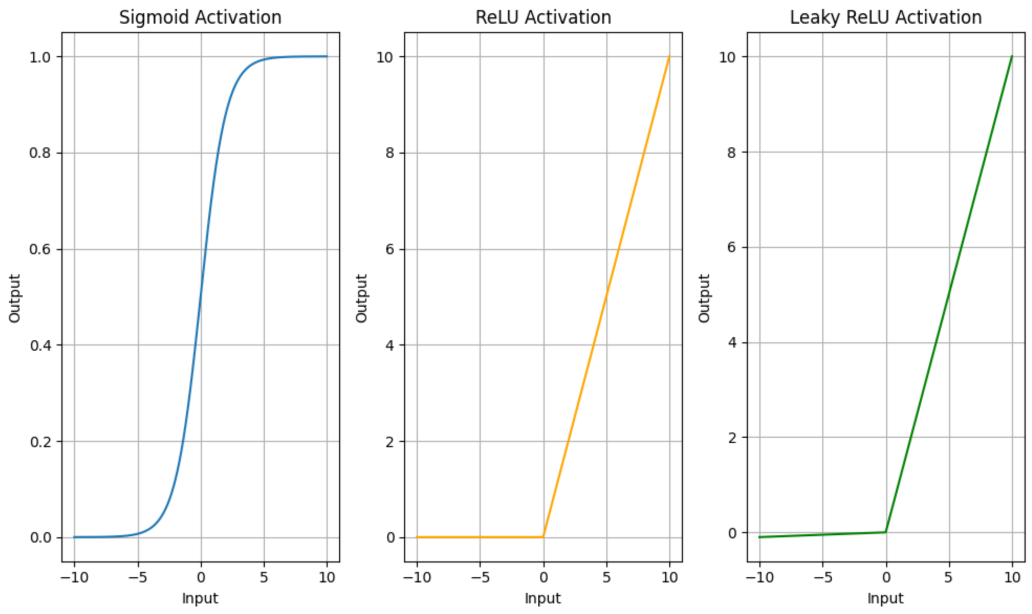
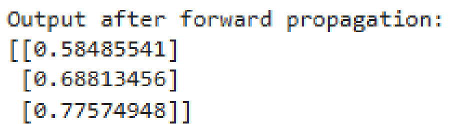


Figure 2: Activation Functions Plots

The Sigmoid, ReLU, and Leaky ReLU activation functions, showing their respective behaviors over the input range from -10 to 10.

4 Forward propagation

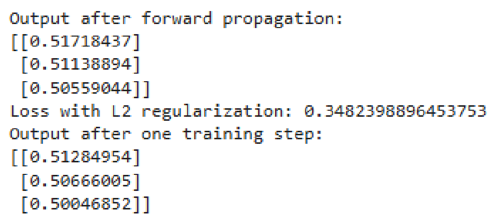
Implemented forward propagation for 3 layer neural network and the Output is shown below :



```
Output after forward propagation:
[[0.58485541]
 [0.68813456]
 [0.77574948]]
```

5 Overfitting and Regularization

Overfitting occurs when a model learns the noise in the training data rather than the actual pattern and we used Regularization to help control this in this problem set.
The Output is given below using L2 regularization and dropout :

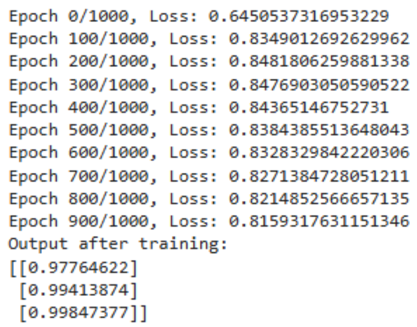


```
Output after forward propagation:
[[0.51718437]
 [0.51138894]
 [0.50559044]]
Loss with L2 regularization: 0.3482398896453753
Output after one training step:
[[0.51284954]
 [0.50666005]
 [0.50046852]]
```

- Sigmoid with Overflow Prevention: The sigmoid function now uses np.clip(x, -20, 20) to clamp input values to a range that prevents overflow when calculating the exponential term.
- Sigmoid Derivative: We compute the derivative of the sigmoid using the output from the sigmoid function itself, making the back propagation calculations more efficient.

6 Training a Neural Network

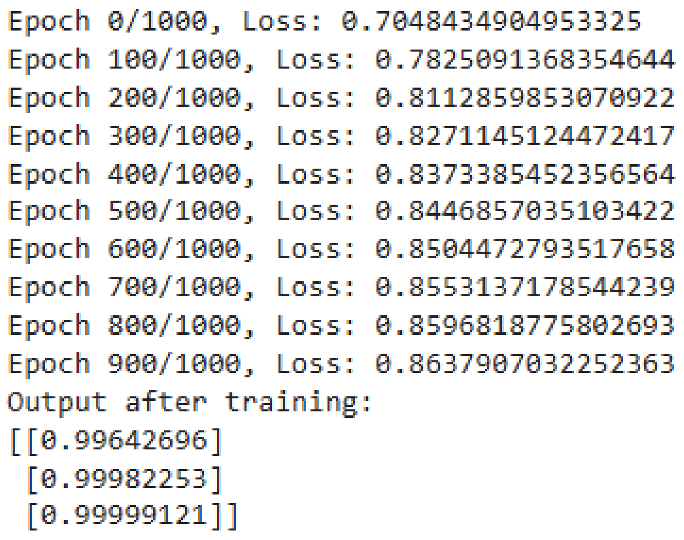
Implemented a simple gradient descent loop to train the neural network using Epoch tests. The output is shown below :



```
Epoch 0/1000, Loss: 0.6450537316953229
Epoch 100/1000, Loss: 0.8349012692629962
Epoch 200/1000, Loss: 0.8481806259881338
Epoch 300/1000, Loss: 0.8476903050590522
Epoch 400/1000, Loss: 0.84365146752731
Epoch 500/1000, Loss: 0.8384385513648043
Epoch 600/1000, Loss: 0.8328329842220306
Epoch 700/1000, Loss: 0.8271384728051211
Epoch 800/1000, Loss: 0.8214852566657135
Epoch 900/1000, Loss: 0.8159317631151346
Output after training:
[[0.97764622]
 [0.99413874]
 [0.99847377]]
```

7 Adaptive Learning Rates

We used Adaptive learning rate methods, like Adam, adjust the learning rate based on past gradients, improving the training stability, And the output is shown Below :



```
Epoch 0/1000, Loss: 0.7048434904953325
Epoch 100/1000, Loss: 0.7825091368354644
Epoch 200/1000, Loss: 0.8112859853070922
Epoch 300/1000, Loss: 0.8271145124472417
Epoch 400/1000, Loss: 0.8373385452356564
Epoch 500/1000, Loss: 0.8446857035103422
Epoch 600/1000, Loss: 0.8504472793517658
Epoch 700/1000, Loss: 0.8553137178544239
Epoch 800/1000, Loss: 0.8596818775802693
Epoch 900/1000, Loss: 0.8637907032252363
Output after training:
[[0.99642696]
 [0.99982253]
 [0.99999121]]
```

8 Summary

This report explores the foundational concepts of neural networks, focusing on the architecture, activation functions, forward propagation, regularization, and adaptive learning. It includes the implementation of a 3-layer neural network using Python, demonstrating how layers interact and how forward propagation and backpropagation are performed. Key techniques like L2 regularization and dropout are discussed to mitigate overfitting. The report also covers the application of gradient descent for training the network, as well as visualizations of activation functions and training results.