Problem Set 4

Justin Dang
CS 3113 - Artificial Intelligence
University of Arkansas — Fort Smith

May 19, 2023

1 Background

This following is my implementation of a neural network that classifies handwritten digits. I will be using the following foundations to implement the network.

1. The handwritten digits will be 28×28 pixels and represented as a 1×784 row of float values in a plain text file.
2. Initial weights for layers and true $\mathbf{y}$ values are provided [1]
3. There are 30 neurons in the first hidden layer and 10 in the output layer representing digits 0-9.
4. We will run a fixed number of 700 epochs to simplify the training as opposed to going until the convergence criteria has been met.

The network training will consist of two parts: forward propagation and backward propagation. Forward propagation takes the input data and is run through the network in order to generate outputs. [2] These outputs were run through the logistic activation function, and feeds forward based on the given weights[3]. The back propagation step then computes the gradient of the loss function with respect to the weights. The algorithm uses these gradients to adjust the weights and the process repeats in another epoch. This model uses L2 regularization to prevent over fitting. [4]

2 Implementation

A brief summary of the implemented steps in the network:

[1] Mackey, A CS3113 Course Material 2023
[2] Artificial Intelligence Modern Approach Third Edition
[3] c/o *Funderburk, T* – UAFS
[4] Massachusetts Institute of Technology 9.520 Class 02 September 2015

1. The input data $\mathbf{X}$ is constructed with 10000 records. With the bias column added, we get $\underset{10000\times 785}{\mathbf{X}}$.

2. I then calculated the activation of the first hidden layer as $\underset{10000\times 30}{\mathbf{H}_1} = \sigma\left(\mathbf{X} \times (\mathbf{W}^1)^T\right)$.

3. Then I calculated the activation of the output layer in a similar fashion but using the previous activated outputs. $\underset{10000\times 10}{\mathbf{H}_1} = \sigma\left(\mathbf{H}_1 \times (\mathbf{W}^2)^T\right)$.

4. The predictions will be the largest number out of each row of $\hat{Y}$.

5. To begin the backprop step, we calculate the deltas of the output layer by calculating $\hat{Y}^{(i)} - Y^{(i)}$.

6. We then take those deltas and multiply them by the their respective weights. Then we will take the Hadamard product of that and the activated dot product of that x record and its weights: $\Delta^s = (\Delta^2 \times W^{2*}) \cdot \sigma'(x^{(i)} \times (W^{1*})^T)$.

7. The gradient of the output layer takes the deltas of the output layers and matrix multiplies it with the outputs from the hidden layer: $\nabla_{W^2} J^{(i)} = (\Delta^2)^T \times \mathbf{h}^{(i)}$.

8. Then the gradient of the first hidden layer is taken in similar fashion. $\nabla_{W^1} J^{(i)} = (\Delta^1)^T \times \mathbf{x}^{(i)}$

9. We perform the weight adjustments through the layers with these gradients.

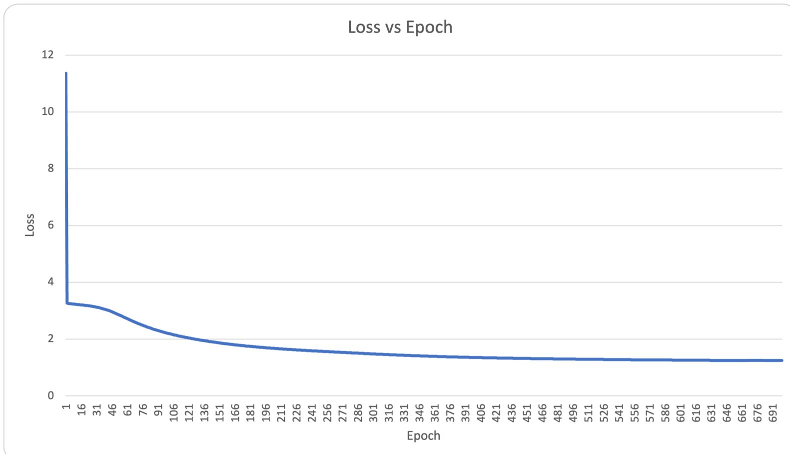
10. With the new weights, the process repeats with a new epoch using the new weights and predictions in the feed forward step.

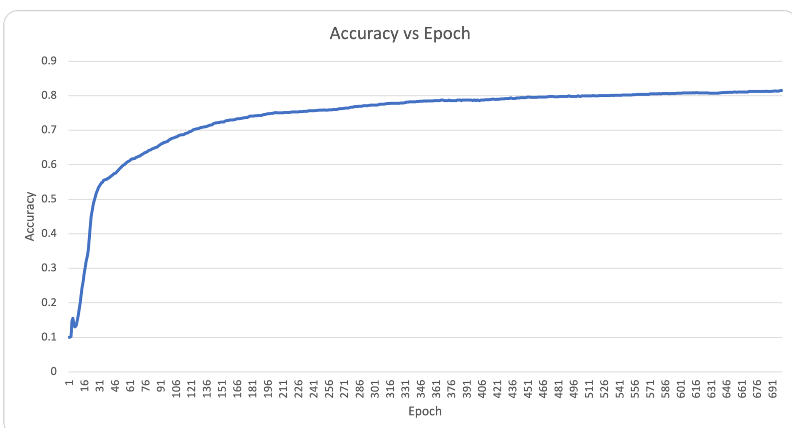
The console outputs show the accuracy of the model after training by testing the first 10 records.

3 Outputs

```
**********************************************************
Problem Set:  Problem Set 4:  Neural Network
Name:         Justin Dang
Synax:        java PS4  w1.txt  w2.txt  xdata.txt  ydata.txt
**********************************************************
Training Phase:  PS4/xdata.txt
--------------------------------------------------------------
    => Number of Entries (n):             10000
    => Number of Features (p):            784

Starting Gradient Descent:
--------------------------------------------------------------


Epoch 1   :     Loss of 11.362     Delta = N/A
Epoch 2   :     Loss of 3.265      Delta = 24801%
Epoch 3   :     Loss of 3.257      Delta = 23%
Epoch 4   :     Loss of 3.252      Delta = 17%
Epoch 5   :     Loss of 3.247      Delta = 14%
Epoch 6   :     Loss of 3.243      Delta = 13%
Epoch 7   :     Loss of 3.239      Delta = 12%
Epoch 8   :     Loss of 3.235      Delta = 12%
Epoch 9   :     Loss of 3.232      Delta = 12%
Epoch 10  :     Loss of 3.228      Delta = 12%
---
Epoch 690 :     Loss of 1.248      Delta = 0%
Epoch 691 :     Loss of 1.248      Delta = 0%
Epoch 692 :     Loss of 1.248      Delta = 0%
Epoch 693 :     Loss of 1.248      Delta = 0%
Epoch 694 :     Loss of 1.248      Delta = 0%
Epoch 695 :     Loss of 1.248      Delta = 0%
Epoch 696 :     Loss of 1.248      Delta = 0%
Epoch 697 :     Loss of 1.248      Delta = 0%
Epoch 698 :     Loss of 1.248      Delta = 0%
Epoch 699 :     Loss of 1.248      Delta = 0%
Epoch 700 :     Loss of 1.248      Delta = 0%

Epochs Required: 700

Testing Phase (first 10 records):
--------------------------------------------------------------
Test Record 1: 4 Prediction: 4 Correct: TRUE
Test Record 2: 1 Prediction: 1 Correct: TRUE
Test Record 3: 4 Prediction: 4 Correct: TRUE
Test Record 4: 6 Prediction: 6 Correct: TRUE
```

```
Test Record 5: 4 Prediction: 4 Correct: TRUE
Test Record 6: 9 Prediction: 9 Correct: TRUE
Test Record 7: 1 Prediction: 2 Correct: FALSE
Test Record 8: 2 Prediction: 2 Correct: TRUE
Test Record 9: 2 Prediction: 2 Correct: TRUE
Test Record 10: 9 Prediction: 9 Correct: TRUE

=> Number of Test Entries (n):         10
=> Accuracy: 90.00%
```


Loss vs Epoch
Loss
Epoch
0
2
4
6
8
10
12


Accuracy vs Epoch
Accuracy
Epoch
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9