Implementation of Backpropagation

Shubham Miglani
Department of ECE
North Carolina State University
smiglan@ncsu.edu

***Abstract*—The implementation of backpropagation algorithm for multilayer perceptron on MNIST dataset for handwritten digit recognition is discussed in this report. Final accuracy of the model on the test data was found to be 92.51% with hyperparameters as (80 neurons in hidden layer,6e-4 as learning rate, mini batch size of 16)**

I. Introduction

The aim of the homework was to implement backpropagation in the given program and use it to train the model on the MNIST dataset. Using the validation data, the hyperparameters were tuned. The learning curves were recorded for loss and accuracy for the final network. Finally, the network was tested on the test data.

II. Derivation of Loss Function

A. Derivation of Loss function

The derivation of Loss function (Cross Entropy) is as follows:

$$\hat{y} = \sigma(W^T x + B)$$

$$L_y(\hat{y}) = \sum_{k=1}^{10} -(y_k \ln \hat{y}_k + (1 - y_k) \ln (1 - \hat{y}_k))$$

$$\hat{y}_k = \sigma(a_k)$$

$$L_y(\sigma(a)) = -(y \ln \sigma(a) + (1 - y) \ln (1 - \sigma(a)))$$

$$\frac{\partial L_y(\sigma(a))}{\partial a_k} = -\left(\frac{y_k}{\sigma(a)} - \frac{1 - y_k}{1 - \sigma(a)}\right)\left(\sigma(a_k)(1 - \sigma(a_k))\right)$$

$$\frac{\partial L_y(\sigma(a))}{\partial a_k} = -\left(y_k(1 - \sigma(a_k)) - (1 - y_k)(\sigma(a_k))\right)$$

$$\frac{\partial L_y(\sigma(a))}{\partial a_k} = -(y_k - y_k(\sigma(a_k) - \sigma(a_k) + y_k(\sigma(a_k)))$$

$$\frac{\partial L_y(\sigma(a))}{\partial a_k} = \sigma(a_k) - y_k = \hat{y}_k - y_k$$

$$\Delta_a L_y(y) = \begin{bmatrix} \frac{\partial L_y(\hat{y})}{\partial a_1} \\ \frac{\partial L_y(\hat{y})}{\partial a_2} \\ \cdot \\ \cdot \\ \frac{\partial L_y(\hat{y})}{\partial a_{10}} \end{bmatrix} = \begin{bmatrix} \hat{y}_1 - y_1 \\ \hat{y}_2 - y_2 \\ \cdot \\ \cdot \\ \hat{y}_{10} - y_{10} \end{bmatrix} = \hat{y} - y$$

III. Hyperparameter Tuning

Various Hyperparameters were varied and their effects were studied. As training takes time, these hyperparameters were varied individually and selected based on the highest validation accuracy and the least loss. After one parameter was selected, the next one was varied, and the process was repeated to attain the final model.

A. Number of neurons in the hidden layer

Firstly, number of neurons in the hidden layer were varied from the default configuration of 20 neurons to 40,60,80,100 neurons. The rest of the parameters were kept the same as the default configuration.

Table 1
Validation Accuracy for change in neurons in hidden layer

Number of neurons	Validation accuracy
20(default)	0.9094
40	0.9141
60	0.9157
80	**0.9182**
100	0.9157

Based on this, the layer with the highest validation accuracy with 80 neurons in the hidden layer was selected. The increase in the number of neurons in the hidden layers also leads to an increase in the computation time.

B. Batch size for SGD

The mini batch size for SGD was varied keeping the number of neurons as 80 and the other parameters with the default configuration.

Table 2
Validation Accuracy for change in mini batch size

Mini batch size	Validation accuracy
16	**0.9226**
32	0.9199
64	0.9181
128	0.9182

As can be seen from Table 2, when the mini batch size decreases, the validation accuracy increases but this also leads to significant increase in the computation time. The increase in validation accuracy could be attributed to the fact that since the

batch size is decreased, the training is being done on more amount of data per batch. So, the mini batch size of 16 is selected based on the above data.

C. *Learning Rate(eta)*

Then the learning rate was varied as shown in Table 3. The rate was decreased, and the results were observed. The decrease in learning rate will cause the gradient descent to work slowly. When the learning rate was decreased, it would for some values give a Runtime warning about overflow being encountered in exp.

Table 3
Validation Accuracy for change in learning rate

Learning Rate	Validation accuracy
1.00E-03	0.9226
6.00E-04	**0.9251**
3.00E-04	0.9226
1.00E-04	0.9132
5.00E-05	0.9067

Based on this, the learning rate with the highest validation accuracy of 6e-4 was selected.

IV. FINAL NETWORK ARCHTIECTURE AND PARAMETERS

The hyperparameter tuning is summarized in the table below. The final values selected for the network are shown in the Selected Value column

Table 4
Parameters checked and final values

Parameters	Values checked	Final Value
Neurons	20,40,60,80,100	80
Batch size	16,32,64,128	16
Learning Rate	1e-3,6e-4,3e-4,1e-4,5e-5	6e-4

The final structure for the network is as shown in the following figure:

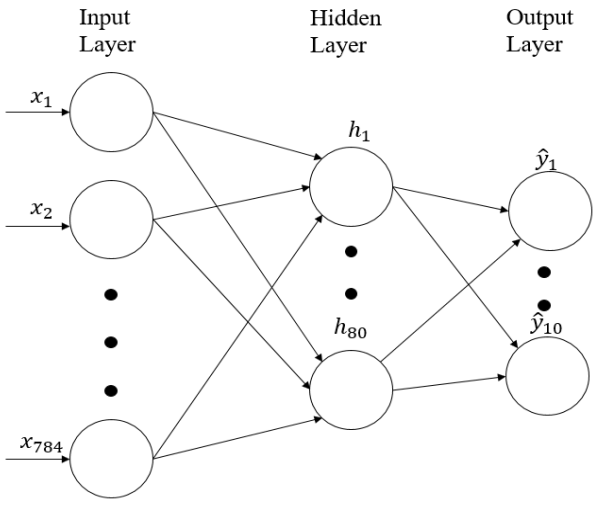


Fig. 1: Network Architecture

V. LEARNING CURVES

The learning curve for the final model for accuracy is as shown in Fig2. The accuracy of the training data is very high (more than 99%) while the validation accuracy is around 92.5%. There is some gap observed between the training and validation data, but the gap interval is very small (around 6%)

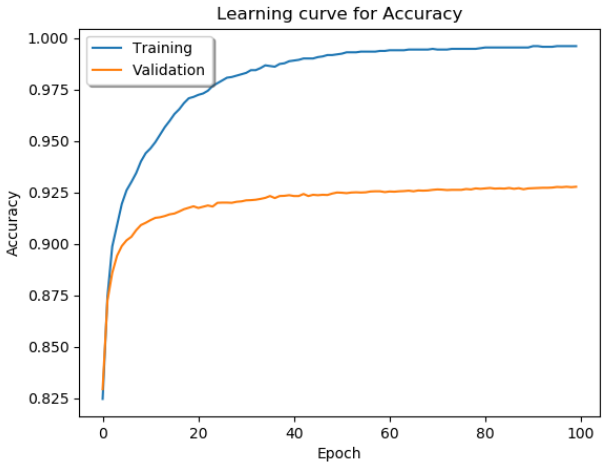
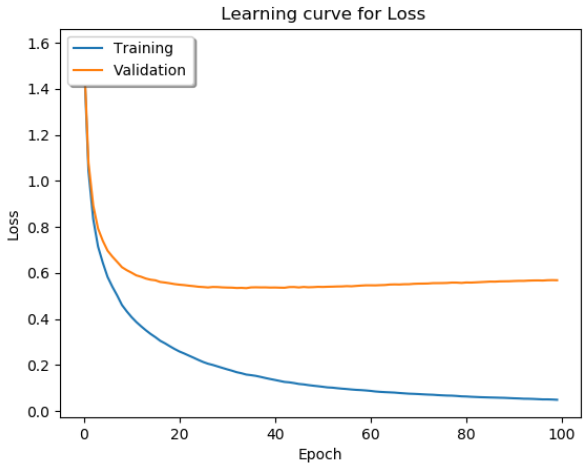


Fig2. Learning curve for Accuracy

The learning curve for the final model for loss is as shown in Fig3. The loss on training data keeps decreasing continuously. This can be attributed to the low batch size. The loss on validation data becomes approximately a straight line after around 20 Epochs. So rather than training for 100 epochs, number of epochs could also have been considered an hyperparameter for tuning to achieve less computation time with simultaneously ensuring high accuracy on the testing data.



The final model trained was tested on the test data and gave an accuracy of **92.15%**

VI. CONCLUSION

In this homework, the backpropagation algorithm was implemented. Gradient check was performed on various layers

and weight values to confirm the calculated gradient was correct. The relative error for gradient check was around e^{-10}confirming the gradient implementation. After that, various hyperparameters including number of neurons, batch size, learning rate were varied and selected for our model. Finally, the loss curves for training and validation data were plotted and the model was tested on the test data to give a result of 92.15%