Implementation of Backpropagation Algorithm

Anushka Gupta[1], Eshaan V. Kirpal[1], Siddhesh Gotad[1]
[1]Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC
Email: {agupta35, evkirpal, svgotad}@ncsu.edu,

***Abstract*—Several state of the art machine learning algorithms have been used to attain near perfect accuracy for the image classification tasks. For performing image classification on the MNIST dataset, we implemented the multilayer perceptron using the sigmoid activation function. In this report, we present our implementation of the back propagation algorithm and share the accuracy results on the training, validation, and testing datasets.**

***Index Terms*—Backpropagation, MNIST, hyperparamater, multilayer perceptron.**

I. Introduction

The goal of this project is to understand the backpropagation algorithm. Backpropagation is a method used in artificial neural networks to calculate a gradient that is needed in the calculation of the weights to be used in the network. In the backpropagation algorithm we adjust the weights by calculating the gradient of the loss function.

The rest of this report is organized as follows. Section II describes the derivation of the back propagation for the designed multilayer perceptron network. Subsequently, Section III presents the results of our hyperparameter tuning while Section IV presents a comparison on the accuracy and loss plots on the validation and training sets. Finally Section V provide the test set accuracy result.

II. Implementation

A. Dataset

The dataset used is the MNIST dataset. It is a database of handwritten digits with each image a gray scale image of dimension 28*28. We used 3000 images in the training set, 10000 images in the validation set, and 10000 images in the test set.

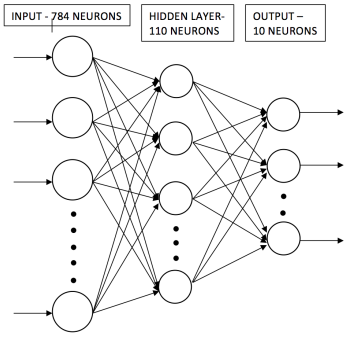

Fig. 1: Designed MLP Network Architecture

B. Derivation of backpropagation for the network

$u_1 = w_1 * x$
$u_2 = w_1 * x + b_1$
$u_3 = \sigma(u_2)$
$u_4 = u_3 * w_3$
$u_5 = u_4 + b_2$
$\hat{y} = \sigma(u_5)$

We now find gradient values with respect to the four parameters: w_1, b_1, w_2, b_2.

$\nabla_{w1} L(\hat{y}) = L'(\hat{y}) * \frac{d\hat{y}}{dw_1}$

$=L'(\hat{y}) * \nabla_{w1} * \sigma(u_5)$
$=L'(\hat{y}) * \sigma'(u_5) * \nabla_{w1}(u_5)$
$=L'(\hat{y}) * \sigma'(u_5) * \nabla_{w1}(u_4 + b_2)$
Since $\nabla_{w1} b_2 = 0, weget$
$=L'(\hat{y}) * \sigma'(u_5) * \nabla_{w1}(u_3 * w_2)$
$=L'(\hat{y}) * \sigma'(u_5) * [\nabla_{w1} w_2 * u_3 + w_2 * \nabla_{w1} u_3]$
$=L'(\hat{y}) * \sigma'(u_5) * [w_2 * \nabla_{w1}(\sigma(u_2))]$
$=L'(\hat{y}) * \sigma'(u_5) * [w_2 * \sigma'(u_2) * \nabla_{w1} u2]$
$=L'(\hat{y}) * \sigma'(u_5) * w_2 * \sigma'(u_2) * x$
Therefore,$\nabla_{w1} L(\hat{y}) = L'(\hat{y}) * w_2 * x * \sigma'(u_5) * \sigma'(u_2)$

$\frac{dL(\hat{y})}{db_1} = L'(\hat{y}) * \frac{d(\hat{y})}{db_1}$
$= L'(\hat{y}) * \sigma'(u_5) * \frac{du_5}{db_1}$
$= L'(\hat{y}) * \sigma'(u_5) * \frac{du_4}{db_1}$
$= L'(\hat{y}) * \sigma'(u_5) * \frac{d(\sigma(u_2) * w_2)}{db_1}$
$= L'(\hat{y}) * \sigma'(u_5) * w_2 * \frac{d\sigma(u_2)}{db_1}$
$= L'(\hat{y}) * \sigma'(u_5) * w_2 * \sigma'(u_2) * \frac{du_2}{db_1}$
Since $\frac{du_2}{db_1} = 1$ we get,
$\frac{dL(\hat{y})}{db_1} = L'(\hat{y}) * \sigma'(u_5) * w_2 * \sigma'(u_2)$

$\nabla_{w2} L(\hat{y}) = L'(\hat{y}) * \nabla_{w2}(\hat{y})$
$= L'(\hat{y}) * \nabla_{w2}(\sigma_{u_5}))$
$= L'(\hat{y}) * \nabla_{w2}(u_5) * \sigma'_{u_5}$
$= L'(\hat{y}) * \nabla_{w2}(u_4 + b_2) * \sigma'_{u_5}$
$= L'(\hat{y}) * \nabla_{w2}(u_3 * w_2) * \sigma'_{u_5}$
$= L'(\hat{y}) * \nabla_{w2}(w_2 * \nabla_{w2} u_3 + u_2 * \nabla_{w2} w_2) * \sigma'_{u_5}$
Therefore, $\nabla_{w2} L(\hat{y}) = L'(\hat{y}) * u_3 * \sigma'_{u_5}$

$\frac{dL(\hat{y})}{db_2} = L'(\hat{y}) * \frac{d(\hat{y})}{db_2}$
$= L'(\hat{y}) * \sigma'(u_5) * \frac{du_5}{db_2}$
$= L'(\hat{y}) * \sigma'(u_5) * 1$
Therefore, $\frac{dL(\hat{y})}{db_2} = L'(\hat{y}) * \sigma'(u_5)$

[h!]

TABLE I: *Accuracy Performance wrt No. of Hidden Units*

Number of Hidden Units	30	60	90	110	150	200
Training Acc.(percent)	86	90.4	91.06	91.56	91.3	91.3
Validation Acc. (percent)	84.6	88.79	89.53	89.53	89.53	89.5

[h!]

TABLE II: *Accuracy Performance wrt Learning Rate*

Learning Rate	5e-4	1e-3	2.5e-3	5e-3	1e-2
Training Acc.(percent)	88.6	91.23	93.96	96.26	97.6
Validation Acc. (percent)	87.68	89.61	90.29	89.72	89.39

C. Sigmoid and Derivative of Sigmoid

The following functions are implemented to perform the sigmoid and the derivative of the sigmoid function:

```
def sigmoid(z):
    return 1/(1+np.exp(-z))

def sigmoid_prime(z):
    return sigmoid(z)*(1 - sigmoid(z))
```

III. Hyperparamater Tuning Results

For training our model, we first tuned the number of hidden units in hidden layer 1 of the network and found the optimal value where the training loss is the minimum and the model isn't overfitting.

The accuracy increases and training loss decreases as we increase the number of hidden units till 110, though increasing beyond that wasn't improving the accuracy or the loss, as can be seen in figure 2.

The final architecture includes the following number of activation units= [784,110,10]

After finalizing the architecture, we tuned the learning rate for the model and observed that increase learning rate beyond 1e-1, led to sharp decline in the accuracy. Also, learning rates smaller than 1e-1 led to a slow learning process.

Learning rate selected 0.001.

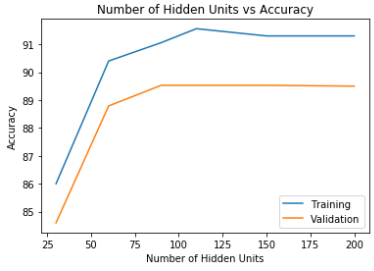

Fig. 2: Tuning number of hidden units in layer 2

1) Table 2 and the graph show the accuracy performance of the model wrt the Learning rate used.

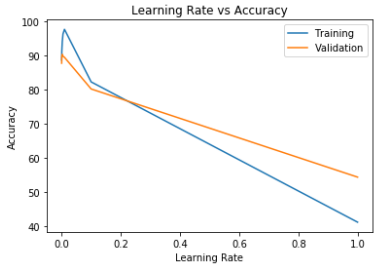

Fig. 3: Tuning of Learning Rate for better Accuracy

IV. Plot of Learning Curves

1) For Training Datatset:
   Loss=0.693
2) For Validation Dataset:
   Loss=0.776

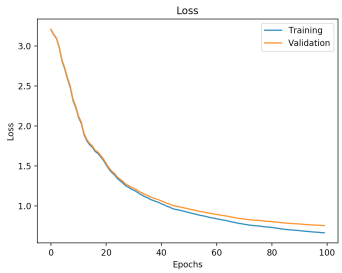

Fig. 4: Loss curves for Training and Validation dataset

1) For Training Datatset:
   Accuracy=90.866
2) For Validation Dataset:
   Accuracy=89.49

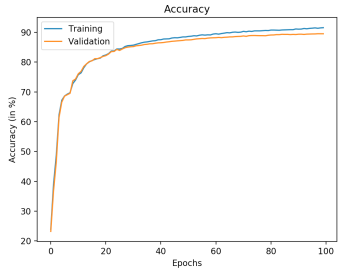

Fig. 5: Accuracy curves for Training and Validation dataset

V. Accuracy on Testing Set

With our final architecture, we obtained an accuracy percent of 88.48 on our testing dataset as seen in figure 6.

```
Epoch 99 training complete
[training loss]: 0.6931256262531553
[training accuracy]: 2726 / 3000
[Validation loss]: 0.776438560045812
[Validation accuracy]: 8949 / 10000
(tensorflow) Eshaans-MacBook-Pro:experiment MyReservoir$ python mlp.py --test
Number of training: 3000
Number of validation: 10000
Number of testing: 10000
Testing accuracy : 88.48 %
```

Fig. 6: Results of testing our model