Project 3: Classification

CS-574 – Introduction to Machine Learning

Name : Dhiren Patel
UBitName : dhirenbh
UBit number : 50170084

1 Problem Introduction

In this project, We are required to use linear regression, Neural network with 1 hidden layer and Convolutional neural network to classify handwritten digits provided by MNIST datasets. This dataset contains 60000 training images and 10000 testing images.

2 Reading data

All data is imported using below code :

```
imageTrain = loadMNISTImages('train-images.idx3-ubyte');
labelTrain = loadMNISTLabels('train-labels.idx1-ubyte');
imageTest  = loadMNISTImages('t10k-images.idx3-ubyte');
labelTest  = loadMNISTLabels('t10k-labels.idx1-ubyte');
```

I have used 60000 training images and 10000 testing images.

3 Logistic Regression Method

How have I got hyper parameter values

I have rearranged input image 28 x 28 in array 784 x 1 and included 1 as first element in this array for bias parameter.
Output will be 10 x 1. Each value for each class.

So, Bias **blr** will have size 10 x 1. (1 x 1 for each class)
Weights **Wlr** will have size 784 x 10. (784 x 1 for each class)

I have used mini batch gradient method with different mini batch sizes (ranges in 1 to 30) to achieve these parameter values.

I have used adaptive **learning rate (eta)** with initial value as 0.01.

If current error is less compared to previous error (means it's converging in right direction) then increase eta as below.

```
eta = eta + (0.05*eta);
```

If current error is more compared to previous error (means it missed optimal point with minimum error) then decrease eta as below.

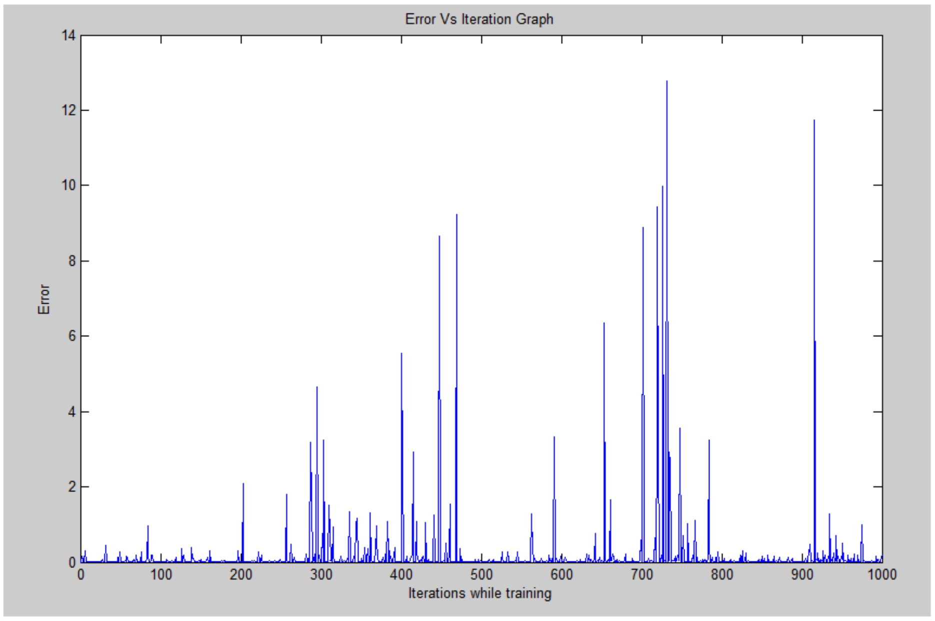
```
eta = eta*0.5;
```

I kept iterating over complete training sets (60000 images) many number of times to update weights and bias until it stopped converging further (Error stopped reducing).
Weights and bias are updated as described in project document.

Misclassification rate : (percentage of images wrongly classified)

On Training set (60000 images) : 6.3283 %
On Testing set (10000 images) : 7.5 %

How Error is converging while being trained :



I have plotted it for last 1000 training images on first iteration. As we can see most of the time error stays near 0. Even if sometimes it goes high then it again converges to 0.

4 Neural Network

I have used 700 hidden units in hidden layer after trying different numbers between 300 to 1000. Misclassification ratio is more in case of less number of hidden units.

Learning rate eta is adaptive same as described in above method.
I have included bias in weights by adding 1 in inputs and hidden layers.

Bias **bnn1** will have size of 1 x 700.
Weights **Wnn1** will have size of 785 x 700.
Bias **bnn2** will have size of 1 x 10.
Weights **Wnn2** will have size of 701 x 10.

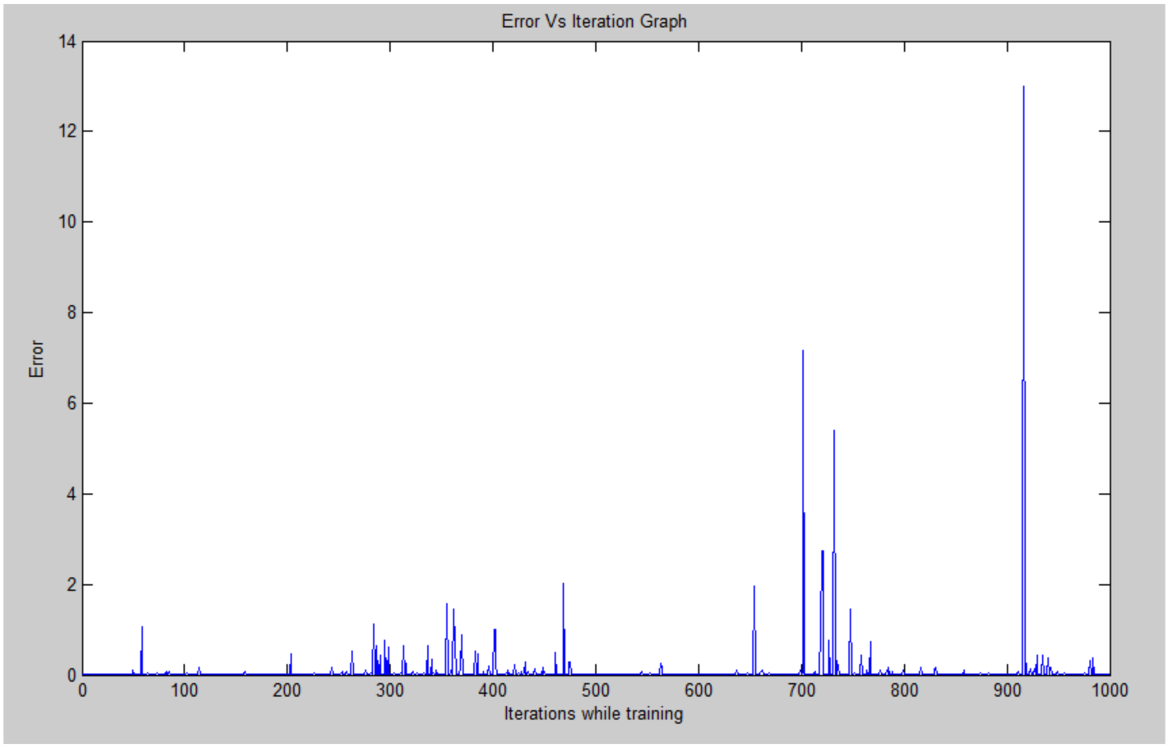
All weights and bias are updated using back propagation method as described in project document.

Misclassification rate : (percentage of images wrongly classified)

On Training set (60000 images) : 0.36 %
On Testing set (10000 images) : 4.24 %

These values can be reduced further by more iterations. (In my program it keeps reducing as many number of iterations)

How Error is converging while being trained :



I have plotted it for last 1000 training images on first iteration. As we can see most of the time error stays near 0. Even if sometimes it goes high then it again converges to 0. As we can see error is more converging towards 0 compared to logistic regression method.

5 Convolutional Neural Network

I used deeplearningtoolbox for this purpose.
I defined cnn layers as below :

```
cnn.layers = {
 struct('type', 'i') %input layer
 struct('type', 'c', 'outputmaps', 10, 'kernelsize', 7) %convolution layer
 struct('type', 's', 'scale', 2) %sub sampling layer
};
```

Input layer for 28 x 28 input images.
10 layers of Convolutional layers getting 7 x 7 different pixels from input images.
Size of convolutional layers is 22 x 22 x 10.
Finally Sub sampling layers for each convolutional layer pooling 2 x 2 unit region from convolutional layers.
Size of sub sampling layers is 11 x 11 x 10.

How Many Weights are defined ?

1) **Weights for convolutional layers**

   Same weights will be used for 1 convolutional layer. Here we are getting 7 x 7 pixels. So, total 49 weights plus 1 bias, so total of 50 parameters for each convolutional layer.

   So, Total number of parameters for this layers are 50 * 10 = 500 parameters

2) **Weights for subsampling layers**

   Here 2 x 2 unit region is fetched from each convolutional layer resulting into 11 x 11 sampling layer for each convolutional layer.
   2 x 2 input is averaged and multiplied by weight and bias is added and then transformed by sigmoid.

   So, Number of weights (if common for 1 sub sampling layer) = 2 then resulting into total of 2 * 10 = 20 parameters

3) **Weights for output layers**

   Here we have 10 classes so, total number of output units are 10. Each output layer is connected to each subsampling layer.
   Number of parameters(weights and bias) for transforming 1 sub sampling layer will be

11 * 11 .
We have total of 10 sub sampling layers. So, no. of weights will be 11 * 11 * 10 = 1210.
And total of 10 output units are there.
So, total number of parameters for this will be 1210 x 10.

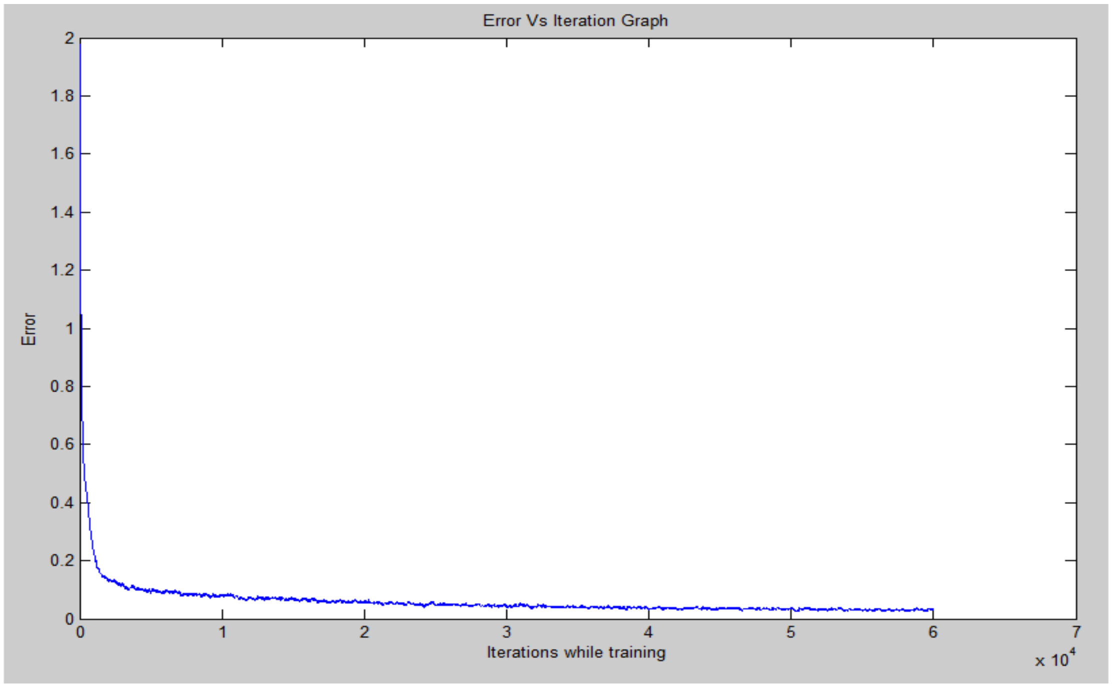
<u>What are the values for other input parameters ?</u>

Learning parameter = 0.1
No. of epochs = 10
Mini batch size = 10

Misclassification rate : (percentage of images wrongly classified)

On testing data (10000 images) : 2.63 % (which can be reduced further by more iterations)

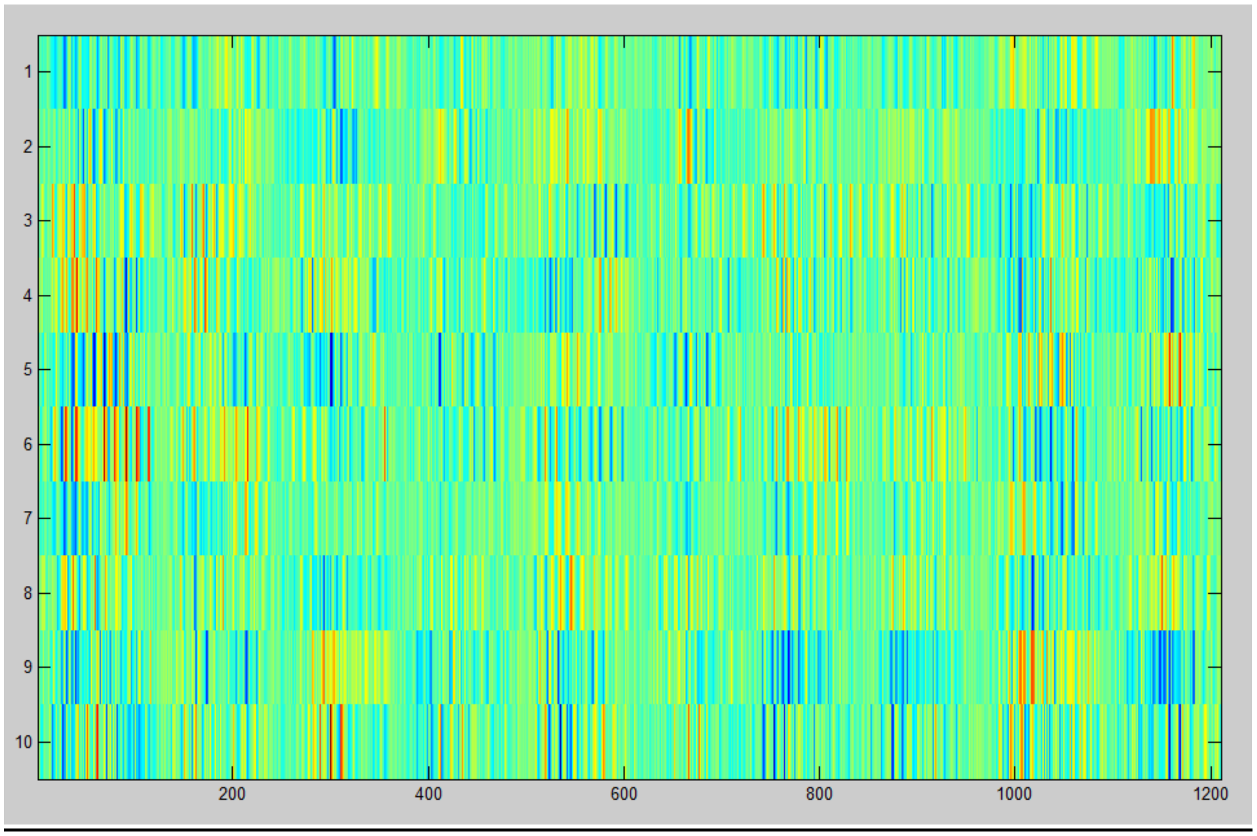
How Error is converging while being trained :



I have plotted error graph for 1 complete iteration on training set (1 to 60000). So, As we can see in this method error is constantly reducing. So, Training time will be less.

Visualization of weights :

Feed forward weights are populated in **cnn.ffW** in **DeepLearnToolbox** package.
Size of cnn.ffW is 10 x 1210. Whose visualization graph is as below.



6 Comparison of different methods

Convolutional neural network can produce better predictions with less training. Clearly it's best candidate among all here.

As we can see misclassification ratio and error graph (as given earlier in this document), error is less for neural network than logistic regression method. So, among these 2, neural network is better method.

Convolution Neural network	
On Testing Data (10000 images)	2.63 %
Simple Neural Network	
On Testing Data (10000 images)	4.24 %
On Training Data (60000 images)	0.36 %
Logistic regression	
On Testing Data (10000 images)	7.5 %
On Training Data (60000 images)	6.3283 %