MACHINE LEARNING PROJECT 3 REPORT

Name: Anmisha Reddy Ramidi
Ubit name: anmishar
Person #: 50208673

Machine Learning Project 3 Report:

Data Partition:

- The data used for this project was MNIST and USPS data.
- The MNIST data extracted consisted of 3 different data sets. They are training data, testing data and validation data.
- The USPS data consists of images that are hand written ranging from 0-9, which have been separated in different folders named 0, 1, 2, etc.
- The test data consists of images randomly ordered which consist of all numbers ranging from 0-9 at a single place.

Implementation:

1. Logistic Regression:
   - The logistic regression was implemented using the formulae in the pdf. The formulae so used are stated below.
   - The logistic Regression in first implemented on the training data. The X input is taken as taining_data[0] and the value of T is obtained from training_data[1].
   - The dimensions of X are 50000x784 and that of T are 50000x1.
   - Hence the variables are taken relatively so as to enable calculation using the formulae.
   - The activation function:

     activation a_k are given by $a_k = \mathbf{w}_k^\top \mathbf{x} + b_k$.

   - The Y matrix that we obtain can be calculated by:

$$p(\mathcal{C}_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

- The gradient to iterate and get new weights are:

$$\nabla_{\mathbf{w}_i} E(\mathbf{x}) = (y_j - t_j)\,\mathbf{x}$$

$$\mathbf{w}_j^{t+1} = \mathbf{w}_j^t - \eta \nabla_{\mathbf{w}_j} E(\mathbf{x})$$

- The weight can be calculated for many iterations until we get the maximum value for the matches.
- During the calculation of the matrix Y we need to perform the one – hot encoding of the T dataset (i.e.) mapping the single valued matrix T similar to Y such that it is of the size 50000xM. Here the value for M is taken to be 10.
- The error function is calculated using the T matrix and the Y matrix and the similarity is recognized.
- The parameters that I assumed for this are learning rate= 1 for which I got the maximum similarity of 0.91.
- As the number of iterations increase the value for the similarity increases thus, giving less number of Nwrongs and hence less Nwrong /Nv ratio.

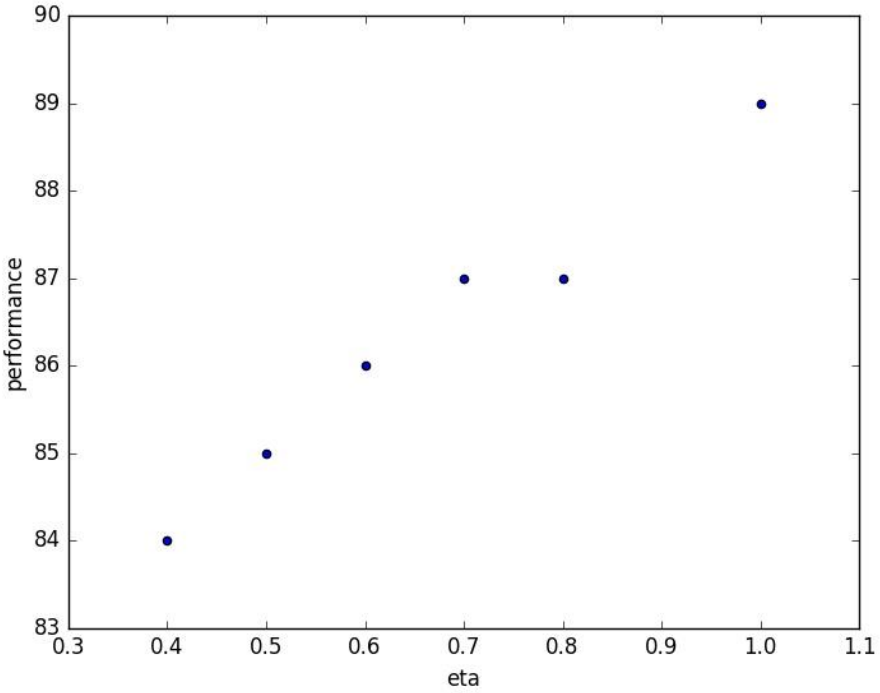
$$E = \frac{N_{\text{wrong}}}{N_{\text{V}}},$$

- We use the 1-k coding scheme to calculate the Nwrong . The function udes for that is argmax in numpy.

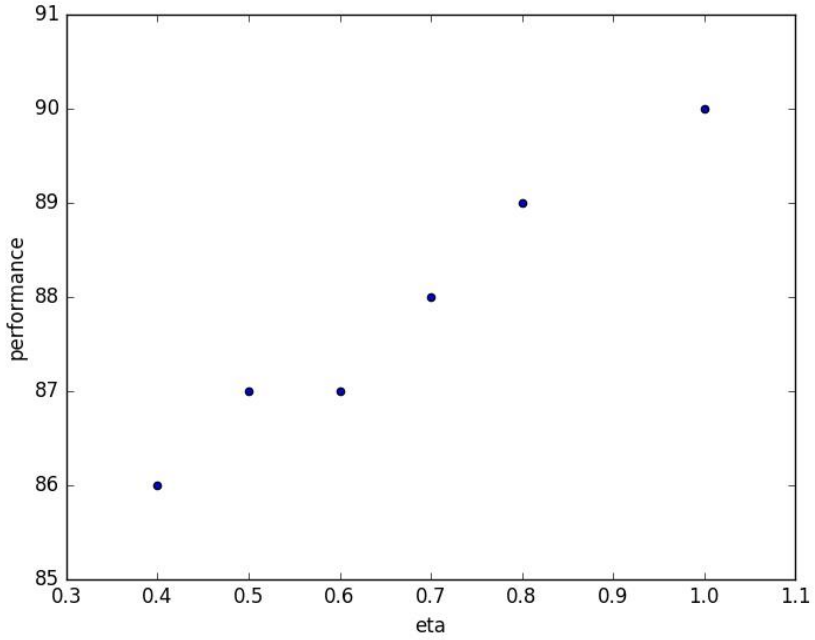
$$C = \arg\max_i y_i.$$

- The number of iterations that I applied in this implementation is 100, for better results we could use 500, but the computational time for this is high and may take very long to process.

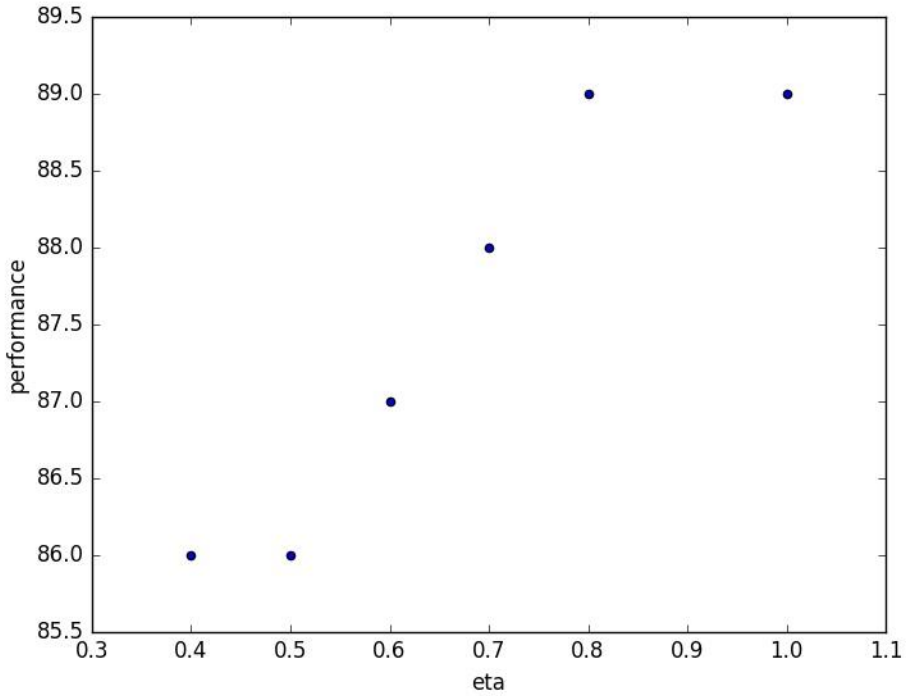
- The graphs for the learning rate vs performance can be found below.



Graph for the training data with different learning rate values and their respective performance.



Graph of learning rate vs performance for the validation data set



Graph for the learning rate vs performance for the test data set

2. Single Layer Neural Network:
    - The single layer Neural Network was also implemented similar to the logistic regression but with few changes in the activation function, calculating weights and the sigmoid and cross-entropy functions used.
    - The Single Layer Neural Network was implemented by using formulae given in the project description.
    - First we calculate the values required to feed forward propagation.
    - The X is taken as the input and the Y is the obtained output, the B matrix is a buffer added to that.

$$z_j = h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + b_j^{(1)}\right)$$

    - The so obtained value of Z is used in the activation function,

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + b_k^{(2)}$$

A.

- This is used further to calculate the value of Y.

$$y_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

- For the function h(.) the logistic sigmoid is used, the value for function h is 1/(1+exp(-x)).
- For the generation of new weights the gradient functions used are:

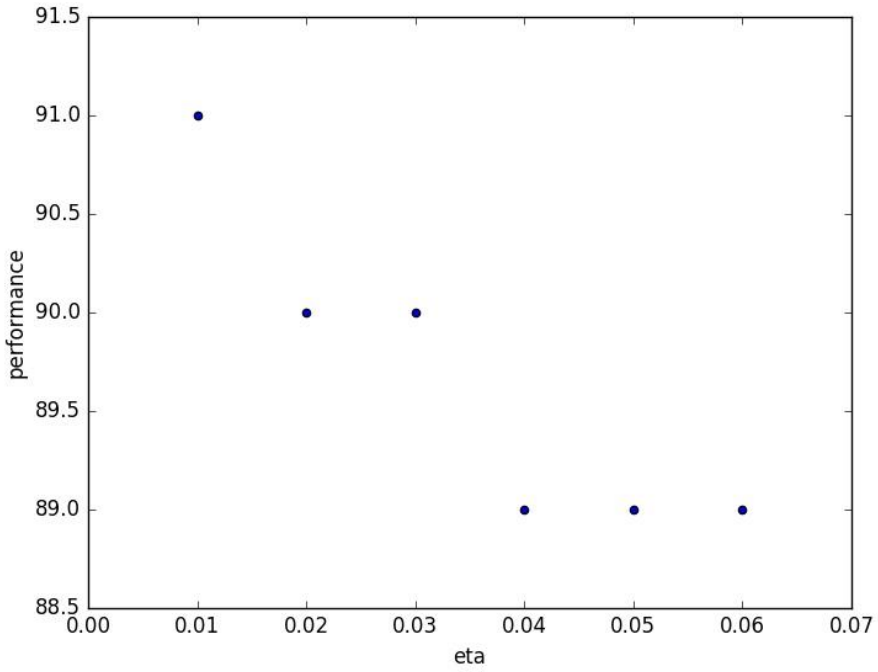
$$\delta_k = y_k - t_k$$

$$\delta_j = h'(z_j) \sum_{k=1}^{K} w_{kj} \delta_k$$

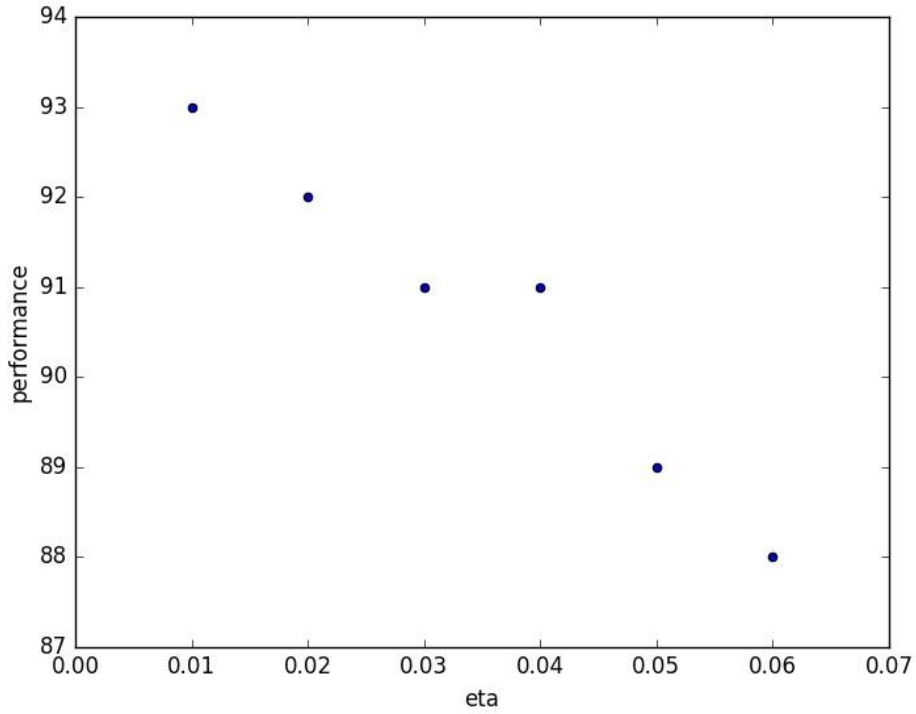
$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \delta_j x_i, \qquad \frac{\partial E}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla_{\mathbf{w}} E(\mathbf{x})$$

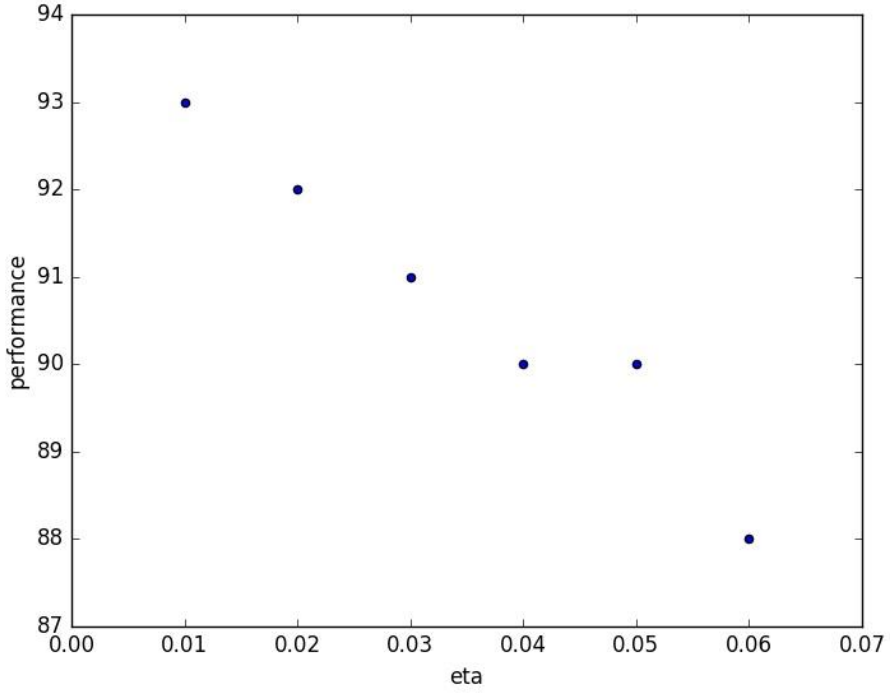
- The learning rate for which the maximum performance measure is obtained is 0.01.
- The graphs related to the performance dependence on various values of learning rate is as follows.



Graph for the learning rate vs performance for training data



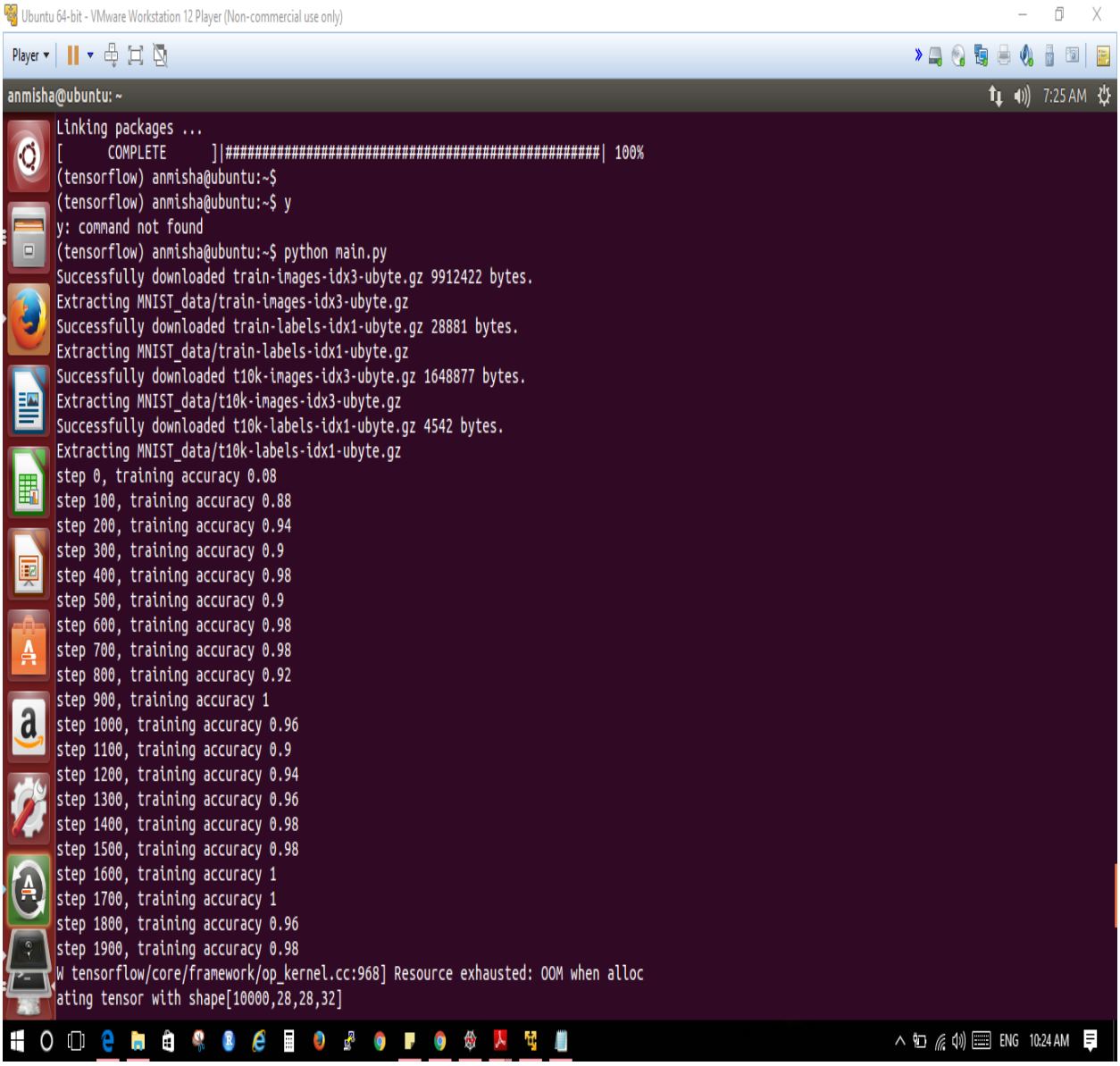
Graph for validation data set where the learning rate and performance vary



Graph for Test dataset of MNIST

3. Convolution Neural Network:

- The convolution neural network was implemented using the tensor flow.
- Tensor flow is an interface that is generally used for the implementation and execution of machine learning algorithms.
- The following are the screenshots of what was implemented
- The data is assumed to be tensors.
- This was implemented in Ubuntu by installing Miniconda and using conda creating a virtual environment.
- For this purpose the slides given by the TA, MLG_Tensor have been used.


Ubuntu 64-bit - VMware Workstation 12 Player (Non-commercial use only)
Player
anmisha@ubuntu: ~
7:25 AM
Linking packages ...
[COMPLETE]|##| 100%
(tensorflow) anmisha@ubuntu:~$
(tensorflow) anmisha@ubuntu:~$ y
y: command not found
(tensorflow) anmisha@ubuntu:~$ python main.py
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting MNIST_data/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
step 0, training accuracy 0.08
step 100, training accuracy 0.88
step 200, training accuracy 0.94
step 300, training accuracy 0.9
step 400, training accuracy 0.98
step 500, training accuracy 0.9
step 600, training accuracy 0.98
step 700, training accuracy 0.98
step 800, training accuracy 0.92
step 900, training accuracy 1
step 1000, training accuracy 0.96
step 1100, training accuracy 0.9
step 1200, training accuracy 0.94
step 1300, training accuracy 0.96
step 1400, training accuracy 0.98
step 1500, training accuracy 0.98
step 1600, training accuracy 1
step 1700, training accuracy 1
step 1800, training accuracy 0.96
step 1900, training accuracy 0.98
W tensorflow/core/framework/op_kernel.cc:968] Resource exhausted: OOM when alloc
ating tensor with shape[10000,28,28,32]
ENG
10:24 AM

The output:

- For learning rate=1 and loop iterations =500
- Performance for logistic=91
- But for this the time taken to execute is too high.
- Hence the number of iterations is reduced to 100 and the result so obtained is:

----------M=100----------

Training Data
eta=1
Performance for Logistic Regression 88
Performance for Single Neural Network 91
Validation Data
Performance for Logistic Regression 90
Performance for Single Neural Network 93
Test Data
Performance for Logistic Regression 89
Performance for Single Neural Network 92

USPS Data:
The USPS data used is in the form of images.
This image data is converted to the required format by resizing them into 28x28 sized pixels.
28x28 so as to match with the algorithms data set with which we performed the training.
This data set is mainly used for testing purpose.