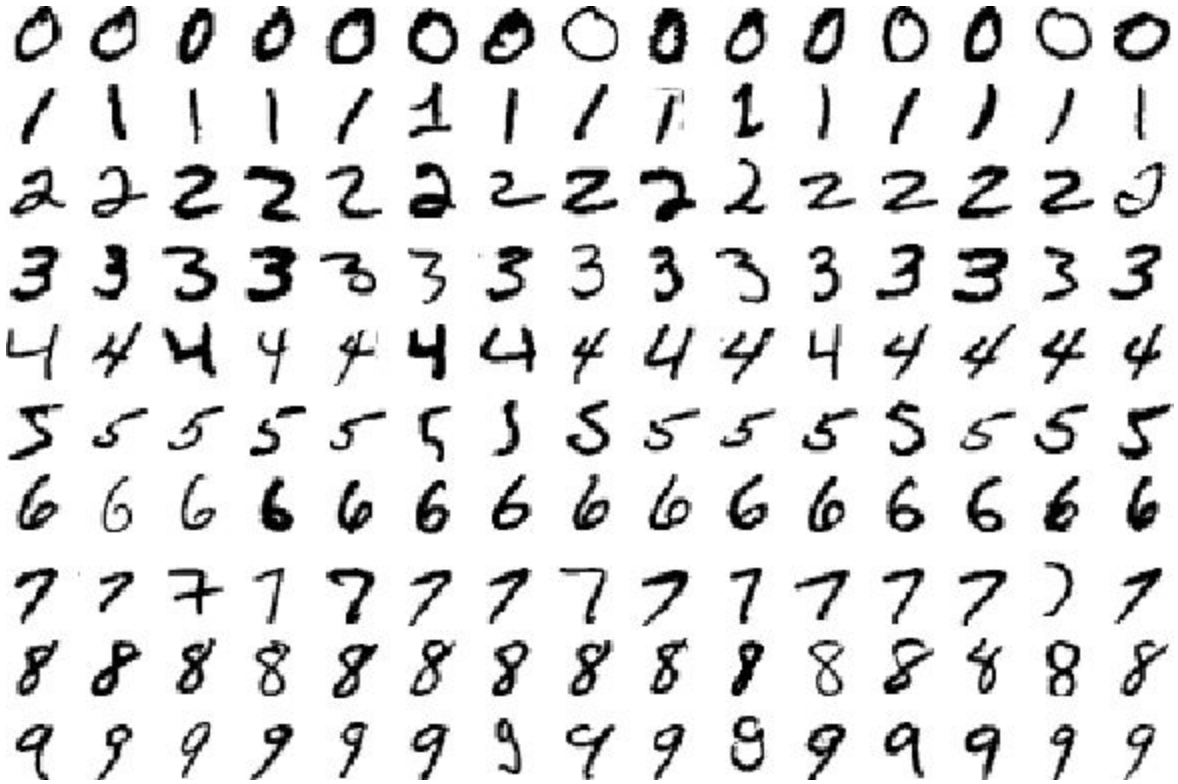
PROJECT-3 REPORT

Introduction to Machine Learning - CSE574



Jayant Solanki

UBIT Name: jayantso
Person Number: 50246821
Fall 2017 CSE

Swati Nair

UBIT Name: swatishr
Person Number: 50246994
Fall 2017 CSE

INTRODUCTION

This report contains the implementation details of Project 3 i.e., classifying handwritten digit image using different classification algorithms such as logistic regression, single hidden layer neural network, convolutional neural network.

OBJECTIVES ACHIEVED

We have trained our classification model on MNIST data using logistic regression, single hidden layer neural network, convolutional neural network and predicted the labels of the digit images in both MNIST and USPS digit dataset.

- Logistic Regression:
    - We trained the model and tuned the hyperparameter i.e. learning rate, by using our own implementation of Logistic regression, we achieved an accuracy of **91.56%** on MNIST test images and **45.15%** on USPS test images at learning rate of **0.14** and lambda (regulariser) value of **0**.
    - Using tensorflow, we have achieved an accuracy of **92.41%** on MNIST test images and **48.32%** on USPS test images
- Single hidden layer neural network:
    - We trained the model and tuned the hyperparameter i.e. learning rate and number of units in hidden layer, we achieved an accuracy of **97.76%** on MNIST test images and **64.6%** on USPS test images.
- Convolutional Neural Network:
    - After training the model using CNN, we achieved an accuracy of **99.18%** on MNIST test images and **75%** on USPS test images.
- No Free Lunch theorem remains valid:
    - We can see that the accuracies of the models on the USPS data were way lower than the accuracies they gave for the MNIST dataset. After seeing such performance of all the three training models on the USPS data set which were trained on the MNIST data set we can conclude that our model is not the best in the general way but performs well on the dataset in which it was trained. Our model needs to have the training knowledge of the USPS data in order to perform well on the USPS data set.

IMPLEMENTATION

1. **Logistic Regression:**
    a. MNIST dataset files were downloaded from the website mentioned in the main.pdf have been read using the **gzip** library function of the python with the help of direction mentioned in this website .
    b. Train and Test Images were then flattened into 2D numpy array, N x 784 size.
    c. Train image data and Test image data have been then standardised with zero mean and unit standard deviation.
    d. Step **c** and Step **d** was performed similarly on the USPS test data.
    e. Weight numpy array was generated valus of 1s and dimension of K x D, where K is the number of classes i.e., 10 and D is the number of data features plus 1 bias feature i.e., (784 + 1).
    f. Column of 1s was added as the first column to both Train set and the Test sets of MNIST and USPS data making the total feature dimension of 785
    g. One hot vectors of MNIST data labels and USPS data labels were created.
    h. Separate functions for calculating the **Cross Entropy error**, **Gradient descent**, **Softmax**, and **Predicting** the image labels were created in the **LRlibs.py** file.
        i. **Cross Entropy error** function was calculated as per the formula mentioned in the Machine Learning slides of week 5.
        ii. **Gradient descent** function was done using the pseudo code mentioned in the document provided along with the Project 3
        iii. **Softmax** function was implemented using the $exp(x - max(x))/\Sigma exp(x)$ formula.
        iv. **Predict** function was implemented by returning the column which had max of the **W.dot(X)** in each row.
    i. Model was trained on the Training data images with **epoch** count of **200** and varying **learning rate** of **0.01 to 0.14.** The accuracies and the cross entropy error have been tabulated in the **tables 1.1 and 1.2** for both MNIST and USPS data sets.
2. **Single Hidden Layer Neural Network:**
    a. This has been implemented using TensorFlow
    b. The model consists of one hidden layer and one output layer
    c. In hidden layer, we have initialized 784x1024 weights and 1024 bias for

1024 units by random values. Then, multiplied the input with the weights and added bias to it. On this value, applied ReLU activation function.

d. The output of hidden layer is fed to the output layer. The output layer consists of 1024x10 weights and 10 bias for 10 units in this layer and are randomly initialized.
e. The output of the model is obtained by multiplying these weights with output from hidden layer and adding bias of output layer. Output will be a one-hot representation of the predicted label
f. We have trained the above model using AdamOptimizer with number of epochs = 20000 and input batch size = 50 in each epoch and chose that model which has the minimum cross-entropy error
g. We have tuned the hyperparameters such as number of units in hidden layer (784, 864, 944, 1024) and learning rate (0.01 to 0.05) and chose the model which has maximum accuracy on validation set
h. Then, ran the model on MNIST and USPS test data to get test accuracy.

3. **Convolutional Neural Network:**
    a. This is also implemented using TensorFlow
    b. The model consists of 4 layers: Convolution layer 1 (convolution with 32 features applied on 5x5 patch of image + 2D max pooling), convolution layer 2 (convolution with 64 features applied on 5x5 patch of image + 2D max pooling), fully connected layer with 1024 neurons and ReLU activation function, logit layer with 10 neurons corresponding to 10 labels
    c. In fully connected layer, few neuron outputs are dropped out to prevent overfitting. The no_drop_prob placeholder makes sure that dropout occurs only during training and not during testing
    d. Trained the model using **AdamOptimizer** with learning rate set to **1e-4** and number of epochs= 20000 on input batches of size 50 and chose that model which has the minimum cross-entropy error
    e. Then, ran the model on MNIST and USPS test data to get test accuracy. USPS test set was divided
    f. No need of tuning of hyperparameter in CNN
4. **USPS data extraction:**
    a. While extracting USPS image features, we have to make it resemble to the MNIST image features as close as possible so that our trained model can classify the images correctly.
    b. In order to do that, we followed below steps:
        i. Resized the image to square shape i.e. width = height = max(width,

height) it was done in such a way that the aspect ratio of the digits was not skewed.

ii. Converted the image into grayscale.

iii. Inverted the image pixels value, that is 255 - Image, black became white and vice-versa so has to follow same pixels values that of the MNIST images.

iv. Resized the each image to 28x28

v. Normalized the image pixels with ([value-min]/[max-min]) formula

vi. Flattened each image into 1x784 numpy array

CODE LAYOUT

There are 2 folders proj3code.zip and proj3code_bp.zip

proj3code.zip :

1. **logistic_main.py**: Run this file for execution of logistic regression model without using tensorflow
2. **logistic_tensorflow_main.py**: Run this file for execution of logistic regression model using tensorflow. It creates the model, trains it, tests it on MNIST validation, test set and USPS test set
3. **single_layer_NN_main.py**: Run this file for execution of single hidden layer NN model using tensorflow. It creates the model, trains it, tests it on MNIST validation, test set and USPS test set
4. **cnn_main2.py**: Run this file for execution of CNN model using tensorflow. It creates the model, trains it, tests it on MNIST validation, test set and USPS test set
5. **libs.py**:
    a. read_gz(images,labels): To read MNIST gz data
    b. view_image(image, label=""): to view single image from the MNSIT data
    c. yDash(trains_images, W): for performing the W.dot(X)
    d. softmax(x) : for calculating the softmax of each row in W.dot(X)
    e. sgd(W, train_images, T, L2_lambda, epochNo, learning_rate): gradient descend for optimising the weights
    f. cross_entropy(W, X, T, L2_lambda): cacluating the loss in the model
    g. predict(W, X): predicting the labels from the output of the model
6. **single_layer_NN_lib.py**:
    a. create_single_hidden_layer_nn(number_hidden_units): create input layer, one hidden layer with specified number of neurons and output layer

7. **cnn_lib.py**:
    a. weight_init(shape): Initialize weight variables
    b. bias_init(shape): Initialize bias variables
    c. convolution(x, W): convolves input with given weights and stride 1 and with zero padding
    d. maxpool(x): performs max pooling on window size of 2x2 and stride of 2 with zero padding
8. **USPS_data_extraction.py**:
    a. make_square(im): To make the image with equal height and width
    b. extract_usps_data(): Get USPS test images and labels. Usps_test_images is a Nx784 numpy array and Usps_test_labels is Nx10 numpy array (one-hot representation)

proj3code_bp.zip : This zip folder consists of implementation of backpropagation

IMPROVEMENTS

1. MNIST Data Set: The training and test dataset after reading from the gzip file were stored in the file systems as numpy files and were later reload every time whenever needed. This saved us from unnecessary load time.
2. Similarly, for USPS data, the test data was stored in file system as numpy files and were later reloaded whenever needed
3. After every training on the training set our Logistical Regression model saved the optimised weight arrays into the file system with names appended by the learning rate values. These weights can be be loaded later in the memory for performing the prediction analysis.

HYPERPARAMETER TUNING:

Hyperparameter tuning was done for Logistic Regression (with and without TensorFlow) and Single Hidden Layer Neural Network.

Logistic Regression

Hyperparameters = learning_rate

Table 1.1

	At epoch count = 200		MNIST			USPS
Sr.	Learning_Rate	Regulariser	Training (%)	Val (%)	Test (%)	Test (%)
1	0.01	1	84.87	88.48	85.94	40.33
2	0.02	1	84.925	88.54	85.98	40.39
3	0.01	0	87.16	90.06	87.85	41.26
4	0.02	0	88.87	91.36	89.3	41.95
5	0.03	0	89.47	91.96	89.94	42.56
6	0.04	0	89.94	92.32	90.39	42.96
7	0.05	0	90.2	92.58	90.59	43.35
8	0.06	0	90.49	92.68	90.82	43.6
9	0.07	0	90.7	92.7	90.88	43.94
10	0.08	0	90.88	92.78	90.98	44.18
11	0.09	0	91.02	92.86	91.12	44.4
12	0.1	0	91.15	92.9	91.22	44.51
13	0.11	0	91.25	92.98	91.35	44.65
14	0.12	0	91.37	93.1	91.4	44.84
15	0.13	0	91.48	93.22	91.5	45.05
16	0.14	0	91.54	93.3	91.56	45.15
17	0.15	0	91.62	93.38	91.55	45.32

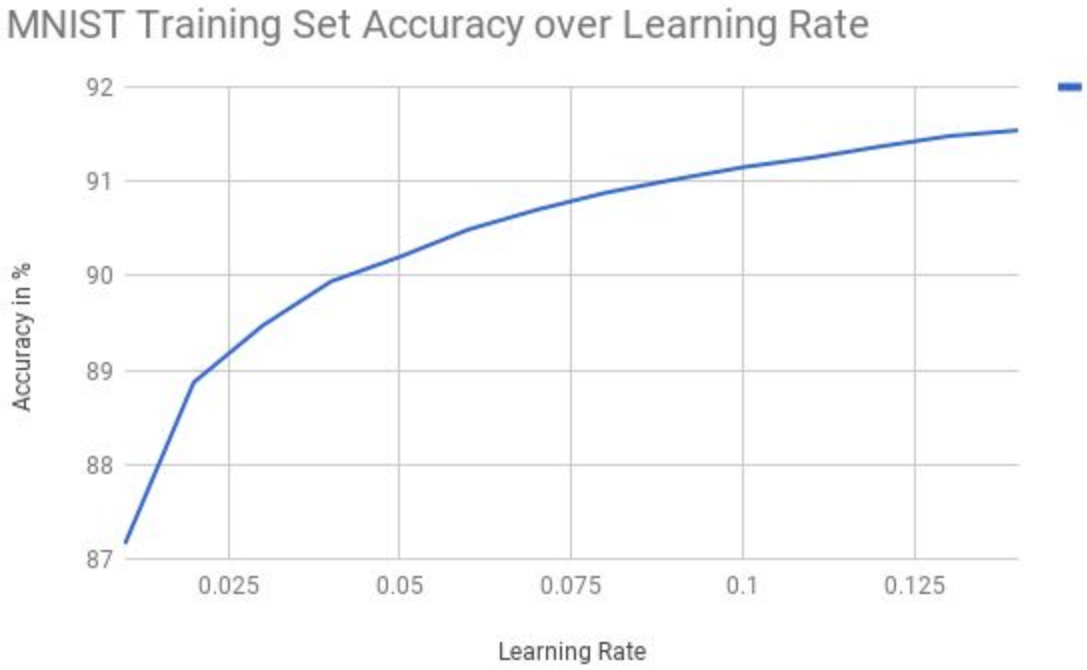


Figure 1.1

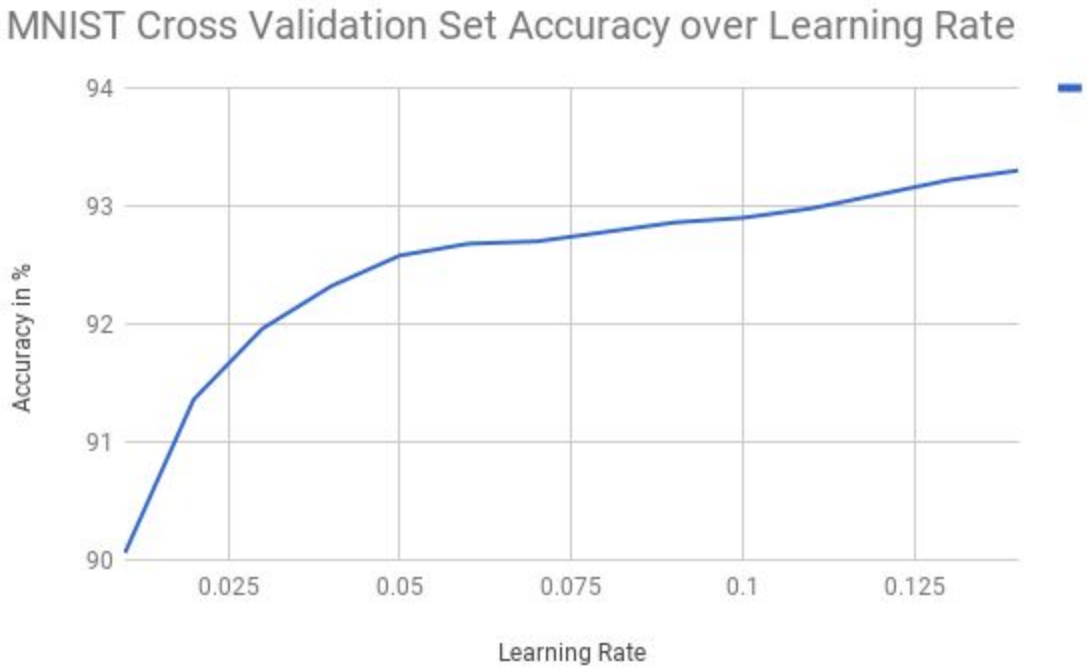


Figure 1.2

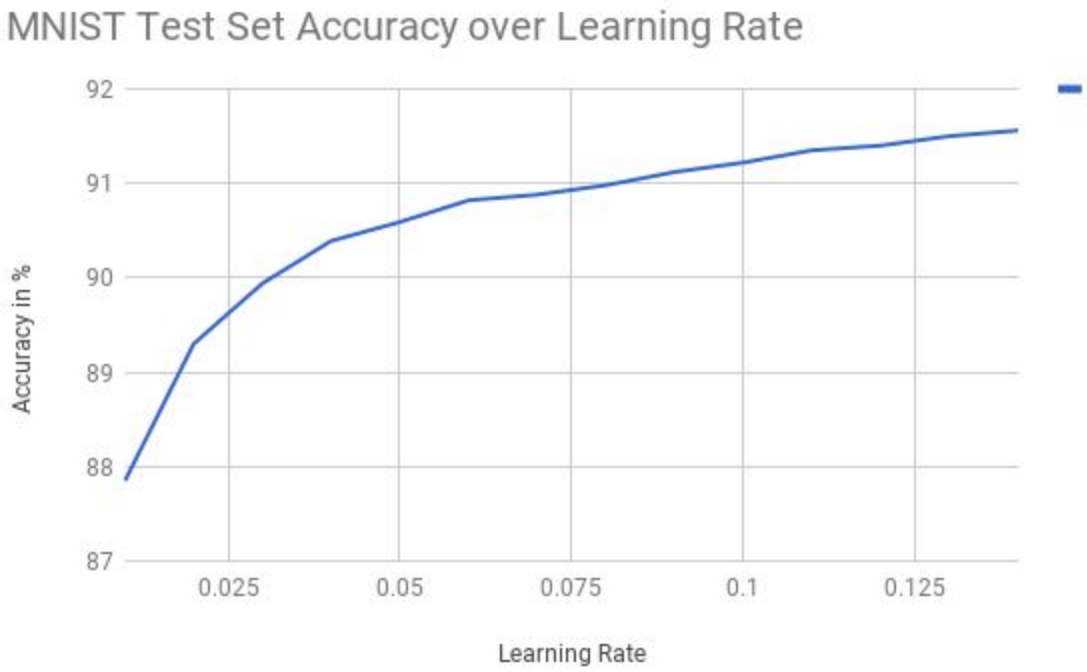


Figure 1.3

Table 1.2

	At epoch count = 200					
Epoch	Learning _Rate	Training Cross_entropy error	Learning _Rate	Training Cross_entropy error	Learning _Rate	Training Cross_entropy error
1	0.04	2.303	0.07	2.303	0.14	2.303
10	0.04	0.971	0.07	0.759	0.14	0.575
20	0.04	0.726	0.07	0.583	0.14	0.465
30	0.04	0.62	0.07	0.509	0.14	0.419
40	0.04	0.559	0.07	0.468	0.14	0.393
50	0.04	0.519	0.07	0.44	0.14	0.375
60	0.04	0.491	0.07	0.421	0.14	0.362
70	0.04	0.469	0.07	0.406	0.14	0.352
80	0.04	0.452	0.07	0.394	0.14	0.344
90	0.04	0.438	0.07	0.384	0.14	0.337
100	0.04	0.426	0.07	0.376	0.14	0.332
110	0.04	0.417	0.07	0.369	0.14	0.327

120	0.04	0.408	0.07	0.362	0.14	0.323
130	0.04	0.401	0.07	0.357	0.14	0.319
140	0.04	0.394	0.07	0.352	0.14	0.316
150	0.04	0.388	0.07	0.348	0.14	0.313
160	0.04	0.383	0.07	0.344	0.14	0.31
170	0.04	0.378	0.07	0.341	0.14	0.308
180	0.04	0.374	0.07	0.338	0.14	0.306
190	0.04	0.37	0.07	0.335	0.14	0.304

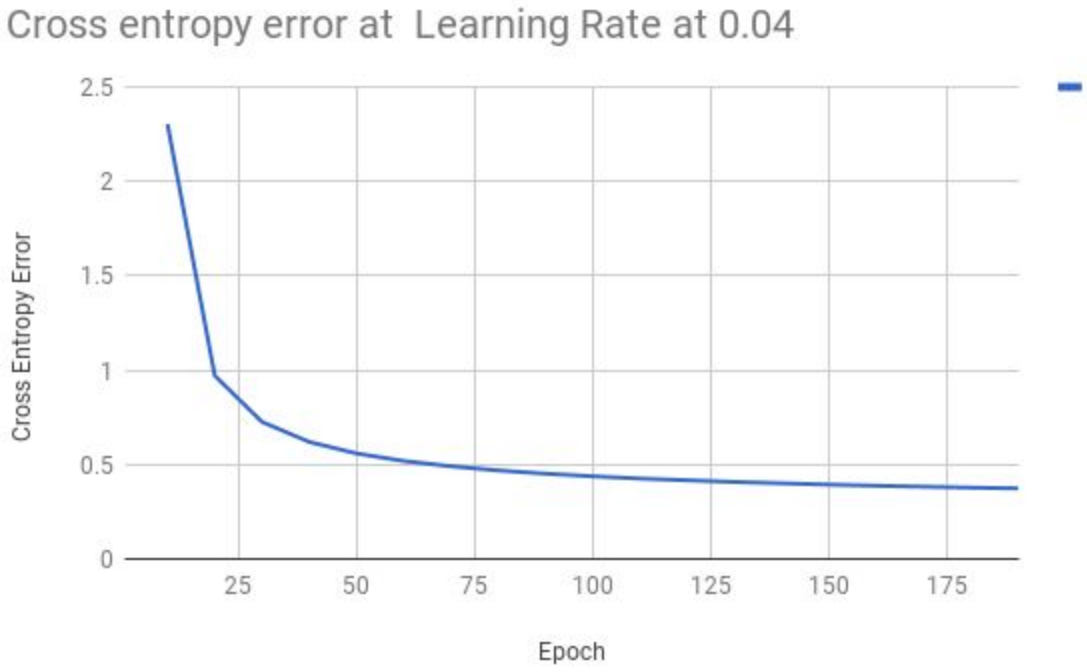


Figure 1.4

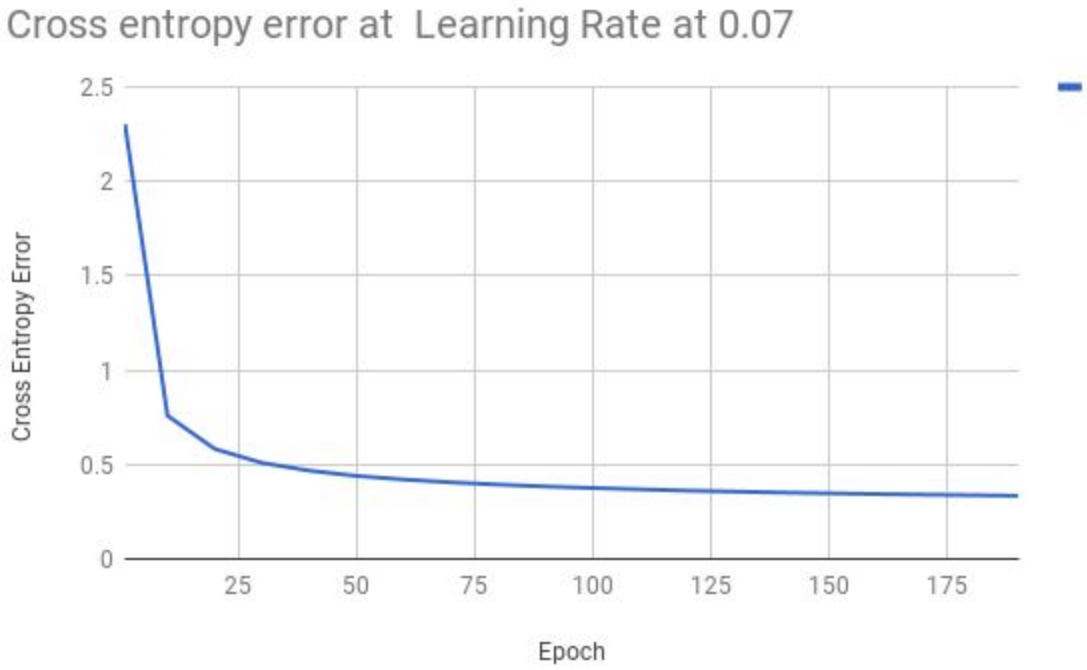


Figure 1.5

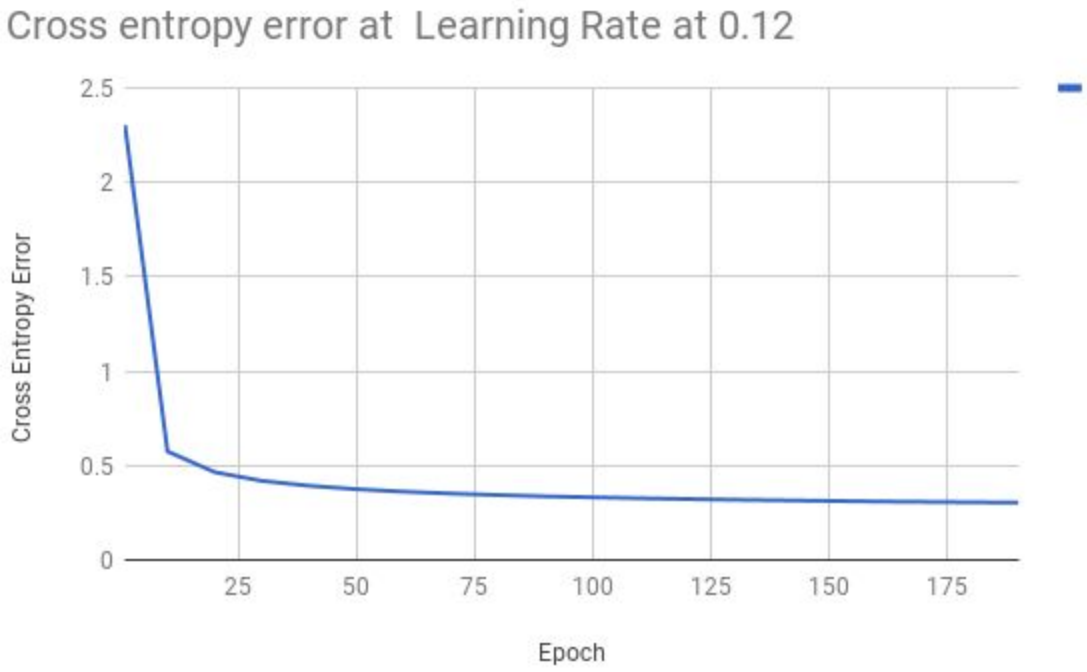


Figure 1.6

Single Hidden Layer Neural Network

Hyperparameters = number of units in hidden layer, learning_rate

	With epoch_count = 20000 and batch_size = 50		MNIST			USPS (Numerals)
Sr.	Number of units in hidden layer	Learning Rate	Training (%)	Val (%)	Test (%)	Test (%)
1	784	0.01	99.27	97.42	97.35	63.18
2		0.02	97.93	96.58	96.01	61.63
3		0.03	95.08	94.32	93.87	56.88
4		0.04	93.37	93.08	92.4	54.64
5		0.05	88.87	88.42	88.37	47.09
6	864	0.01	99.33	97.6	97.46	65.25
7		0.02	97.36	95.32	95.77	59.63
8		0.03	95.72	94.82	94.09	56.85
9		0.04	92.08	91.16	90.94	52.24
10		0.05	91.07	90.2	90.46	49.24
11	944	0.01	99.13	97.74	97.47	64.2
12		0.02	97.9	96.74	96.12	61.74
13		0.03	95	94.16	93.8	56.89
14		0.04	92.39	92.12	91.07	50.29
15		0.05	91.67	91.3	91.09	50.27
16	1024	0.01	99.25	97.52	97.81	64.6
17		0.02	97.6	96.2	95.57	59.59
18		0.03	95.46	94.2	94.16	54.95
19		0.04	90.61	90.38	89.97	52.33
20		0.05	89.56	89.08	88.83	49.46

RESULTS

1. **Logistic Regression (Implementation from scratch)**
   Output for Learning rate 0.07

   Current learning rate is 0.070000
   iteration 0/200: loss 2.303
   iteration 10/200: loss 0.759
   iteration 20/200: loss 0.583
   iteration 30/200: loss 0.509
   iteration 40/200: loss 0.468
   iteration 50/200: loss 0.440
   iteration 60/200: loss 0.421
   iteration 70/200: loss 0.406
   iteration 80/200: loss 0.394
   iteration 90/200: loss 0.384
   iteration 100/200: loss 0.376
   iteration 110/200: loss 0.369
   iteration 120/200: loss 0.362
   iteration 130/200: loss 0.357
   iteration 140/200: loss 0.352
   iteration 150/200: loss 0.348
   iteration 160/200: loss 0.344
   iteration 170/200: loss 0.341
   iteration 180/200: loss 0.338
   iteration 190/200: loss 0.335
   training set Accuracy is 0.907055
   validation set Accuracy is 0.927000
   Test set Accuracy is 0.908800
   USPS set Accuracy is 0.439422

2. **Logistic Regression (using TensorFlow):**
   Output for learning rate 0.5, number of epochs: 10000

   The accuracy on MNIST test set: 92.41
   The accuracy on USPS test set: 48.32

3. **Single Hidden Layer Neural Network:**
   Output for learning rate 0.01, number of epochs: 20000, number of units in hidden layer: 784

   MNIST validation accuracy: 97.42
   MNIST test accuracy: 97.35
   The accuracy on USPS test set: 63.18

4. **Convolutional Neural Network:**

Output for learning rate 1e-4, number of epochs: 20000

MNIST test accuracy: 99.18
The accuracy on USPS test set: 75.13

SOFTWARE/HARDWARE USED

- Sublime Text 3,
- Python 3 Environment based on Anaconda,
- TensorFlow,
- Ubuntu 16 System, Intel core i3 processor.
- Windows 10, Intel core i5 processor.
- Python libraries: numpy, matplotlib, scipy, sklearn
- Timberlake

REFERENCES

1. Ublearns
2. Stackoverflow.com
3. Python, Numpy and TensorFlow documentations
4. https://cs231n.github.io/convolutional-networks/
5. http://yann.lecun.com/exdb/mnist/
6. https://martin-thoma.com/classify-mnist-with-pybrain/