**Home Work 4**

**Intelligent Systems**

# 

# **Problem One**

**System Description:**

* The **learning rate** for the program was 0.05. If the learning rate was taken to be a small value than the neural network will be very slow. The value I choose wasn’t too high or too low. It was the ideal value for my network; if I had chosen a large learning rate than it could have caused the weights to move around from positive region to negative region but never providing an optimum result.
* The **β value** that was chosen was 2 & **λ value** was chosen to be 0.005
* The **Momentum** is also defined as alpha value which was set at 0.5 in the code. This momentum value was applied to equation of delta weights.
* I considered only **one hidden layer** in the neural network while the output neurons we have the range of 0 to 9 values which is defined as output threshold.
* The **dropout rate** for the neural network was 0.5.
* The initial weights were chosen such a way that they range from -0.1 to +0.1 this helps in good learning.
* The **criterion for deciding when to stop the training for our network**: For the first 3500 randomized data points out of 4000 data points was used for training. This was selected because the question mentioned that only a subset of the data should be used for training.

**Note**:

I have repeated the learning process by using a smaller learning value as well (ηh = 0.001 instead of 0.05) for the hidden layer weights. These learning weights are usually initialized after some learning has already been done on them using the weights from “auto-encoder” simulation.

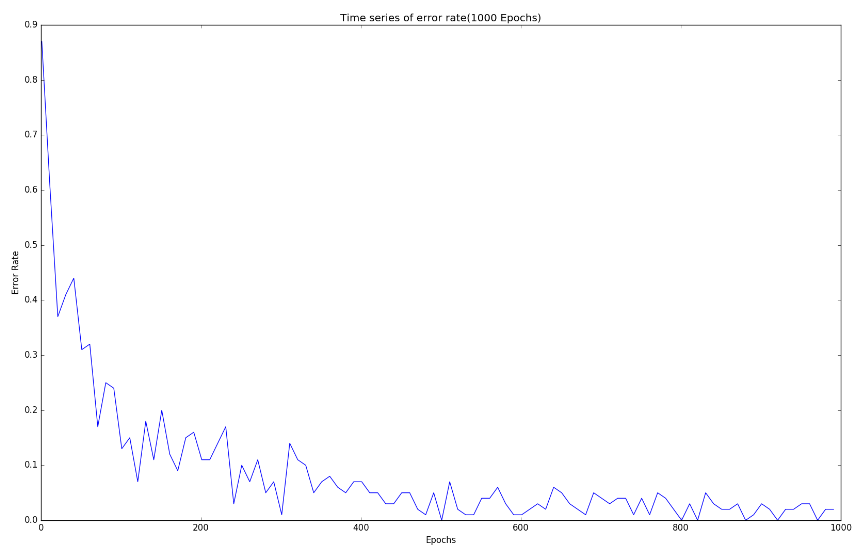
**Results:**

* In this assignment we are training only the outer layer hence the classification error decreases when compared to the previous assignment. If I went ahead & trained all the layers or backpropagate than classification improves. Here mainly we do is transfer learning hence we get better features when compared to previous assignment.
* Classification error would decrease significantly if I back propagated.
* Without regularization loss function can be higher.
* If the classification was using the softmax than there could be better classification’ in this assignment, we are using root mean square the classifier is not that efficient.

**Note**:

1. Sparseness is determined by row it can range from 1 – 10%. It helps in obtaining better features. One of the deciding factors.

Using the above parameters, the final network was trained only for the outer layers. Later when I ran it longer for over (Thousand) **1000** epochs (similar number of epochs as in **problem 2 from Home Work 3)** with hundred (100) random points in each epoch. Below is Figure 1 which depicts the error rates when plotted for every (Ten) 10 epochs as shown…..



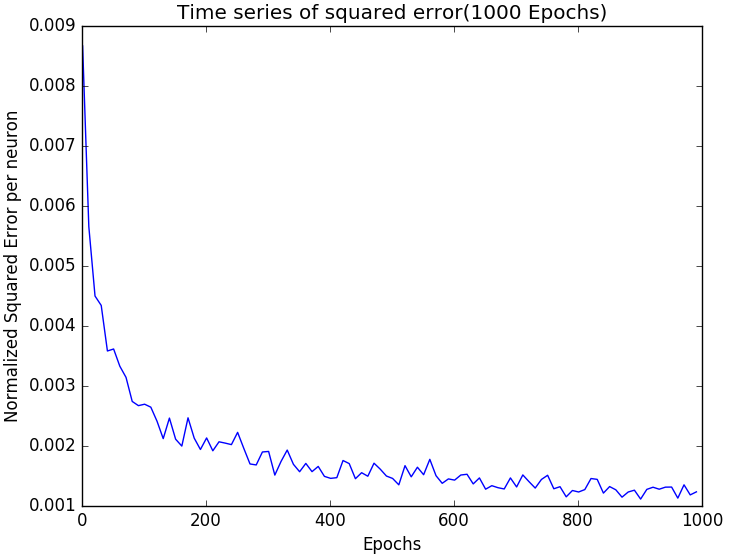
**Figure 1**: Error rates for Time series data set for every (ten) 10 epochs over (Thousand) 1000 epochs

By (Two Fifty) 250 epochs the error rate as shown above decreases sharply.

From the 1st epoch itself & completely flat-lines by (Two fifty) 250 epochs which is a lot faster than in previous case **(Problem 2 from HOMEWORK3**) in Figure 2 (shown below) which flat-lines after 400 epochs.

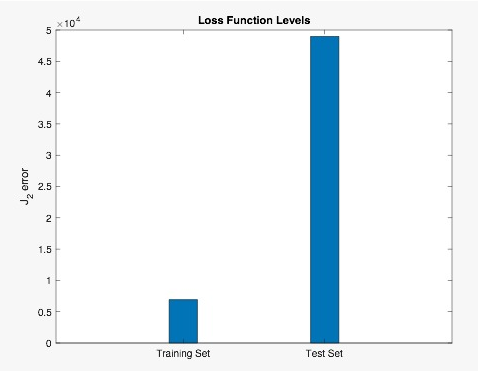
This was possible due to **Fine-tuning** process followed by using hidden layer weights from auto-encoding. The speed of learning was improved due to auto-encoding; it had a positive effect.

The hit-rate was **91.9%** in the end which is very to the best-case simulation rate of **93.1 %.** I also made sure to generate the **Confusion matrices. The confusion matrices** of for both testing & training sets to further evaluate the overall performance. Training set of 4000 points confusion matrix is depicted in Table 1



**Figure 2**: Time series plot of squared (J2) error rates for every 10 epochs over 1000 epochs of training **(Homework 3 problem 2)**

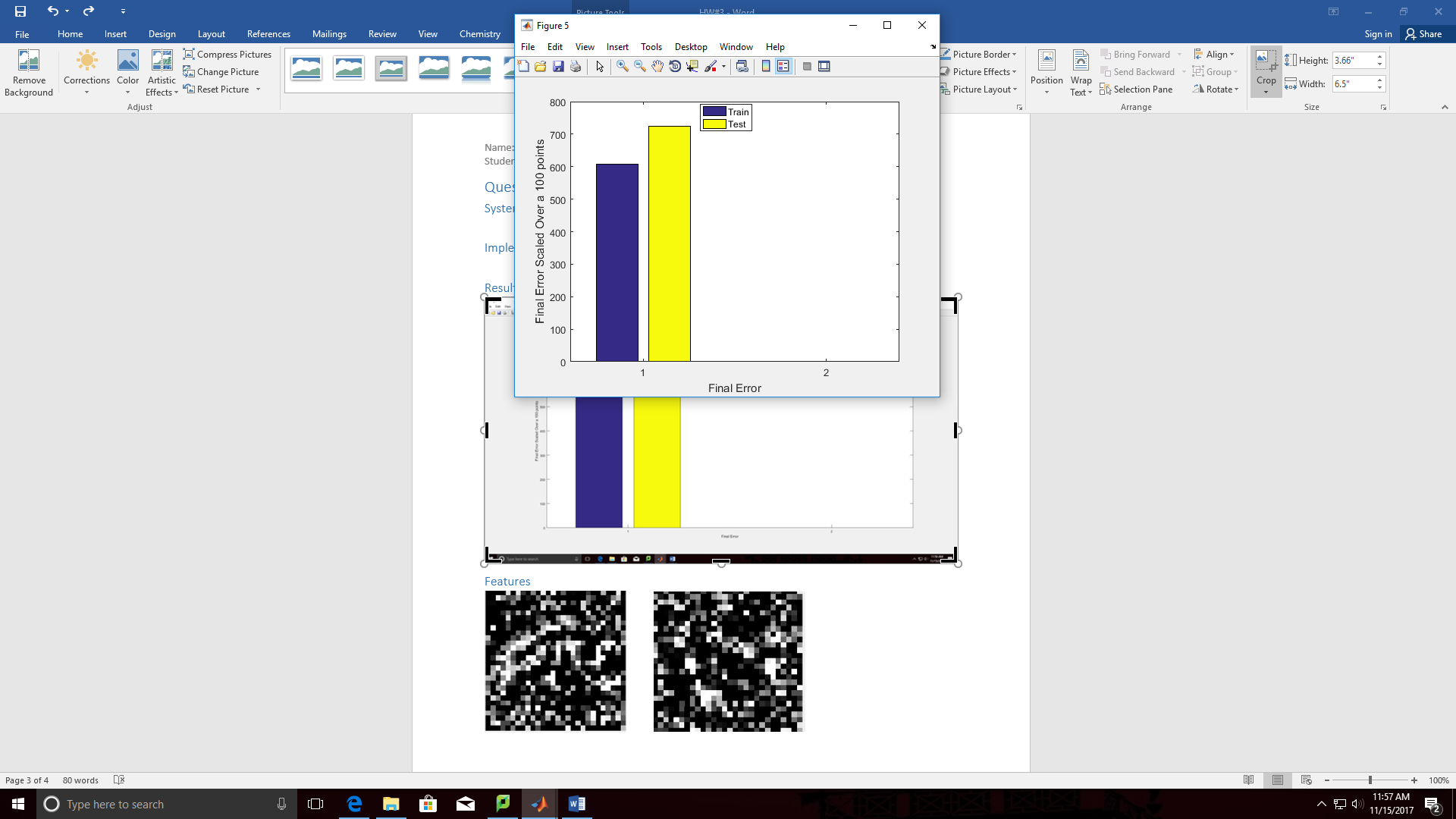
Program is run over series of epochs for the entire training data set. The loss function observed in below figure was high for the test data while the loss was barely anything for the training data.



**Figure 3**: shows the J2 loss function was tabulated for the testing & training dataset

Figure 4 depicted below shows the result of **Problem 2 Homework 3**.

It shows the total final training loss value (2.4348\*10^4) over Four thousand (4000) points & the total final test loss value (7.2541\*10^3) are shown scaled over a hundred points by multiplying by hundred & dividing by the training or test size. As can be seen, the final testing loss value over hundred points (725.41) is higher than the training loss value over hundred (100) points (606.48). When the training loss value over hundred (100) points is divided by (100\*784), the error squared/dimension is approximately 0.0076 which is acceptable considering that the range is 1.00 (it is approximately 0.7% of the range).

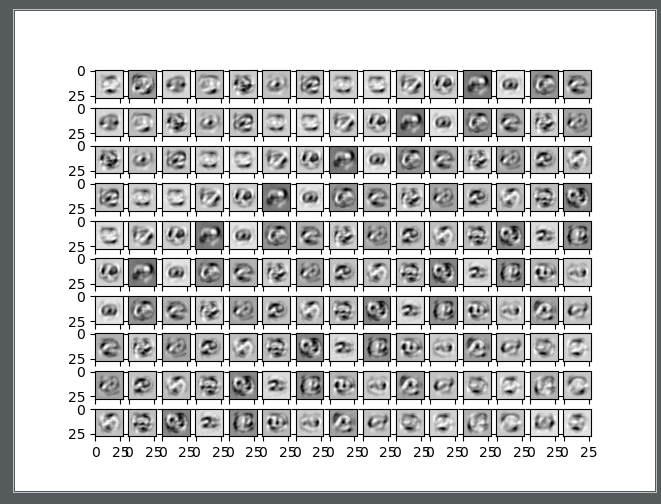


**Figure 4**: Final Error Squared for Testing & Training Data Scaled Over 100 Points ***(Homework 3 problem 2)***

Next, Figure 6 is basically plotting of the weights between the input & hidden layer. Figure 6 clearly depicts the transformation; **we can clearly see the features.**



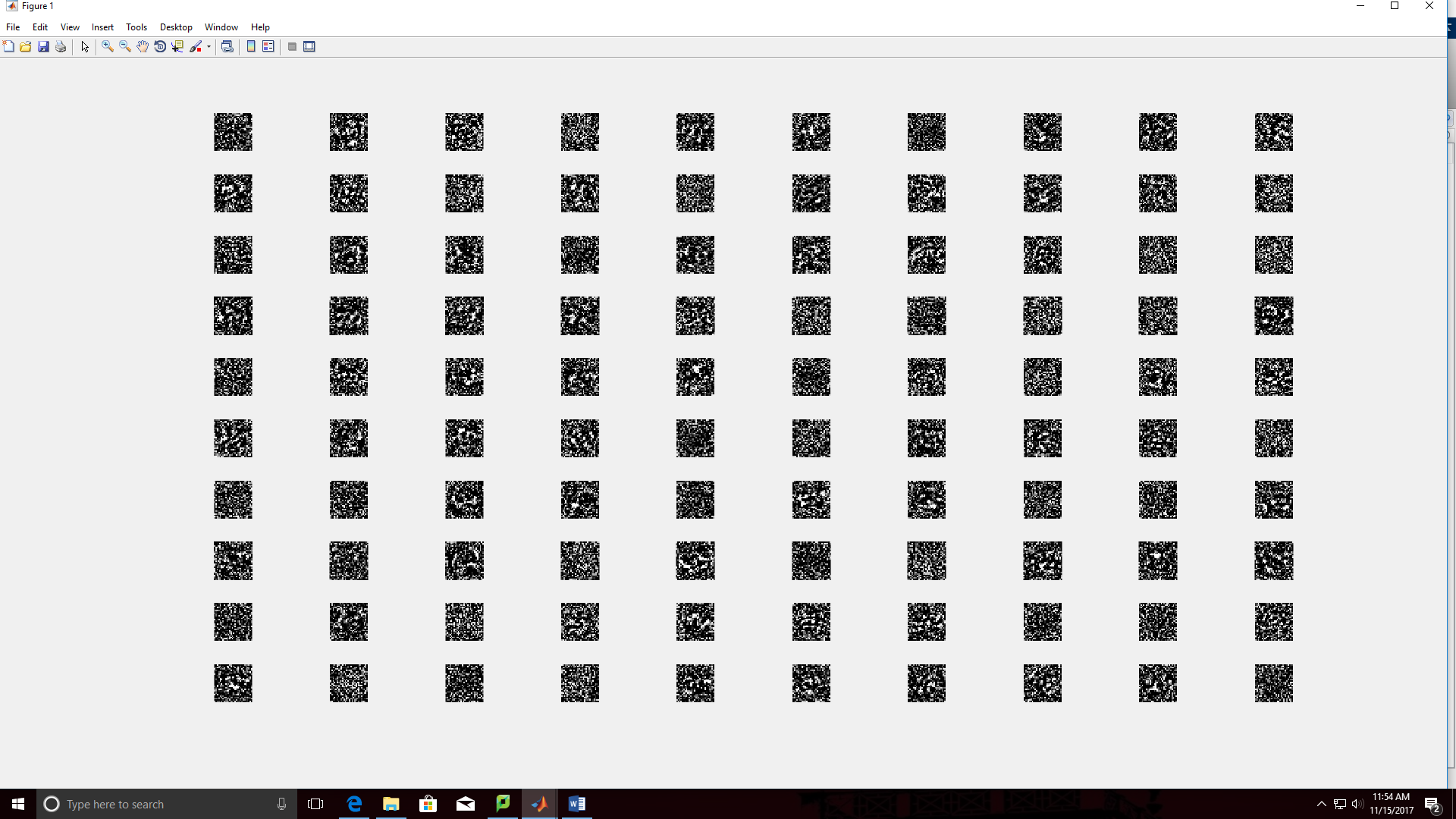
Figure 5: Average Error vs Label of MNIST Data



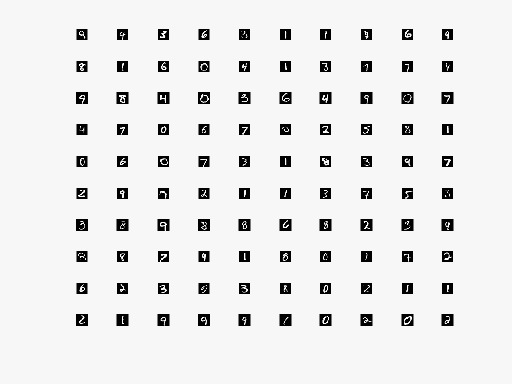
**Figure 6**: Weights between Input layer & hidden layer

**In figure 7,** we can observe that the numbers five & Six are visible. That means that for those 2 numbers the neurons associated with them generate a strong response hence its features are captured. For most numbers we will observe blurriness the reason is because most hidden neurons do not specialize in identifying the correct features for 1 digit. Most features for the hidden neurons when plotted are not too distinct/sharp.

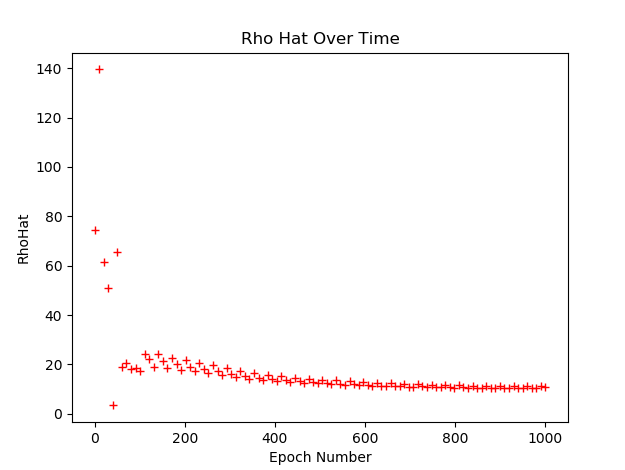
While **in Figure 6 we can clearly see the features for more digits.**



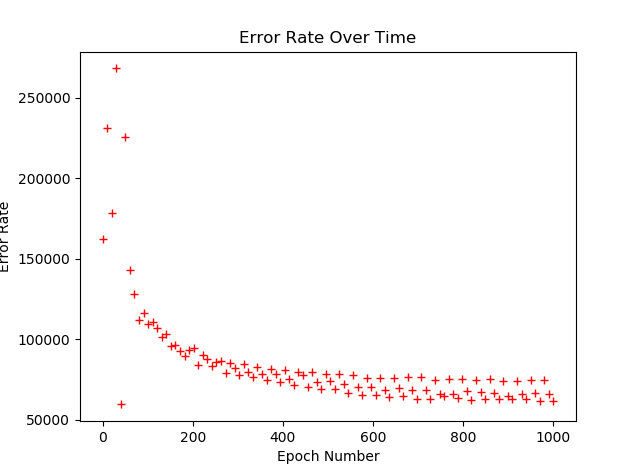
**Figure 7**: Features for All of the Hundred Hidden Neurons ***(Homework 3 problem 2)***



**Figure 8**: Input images

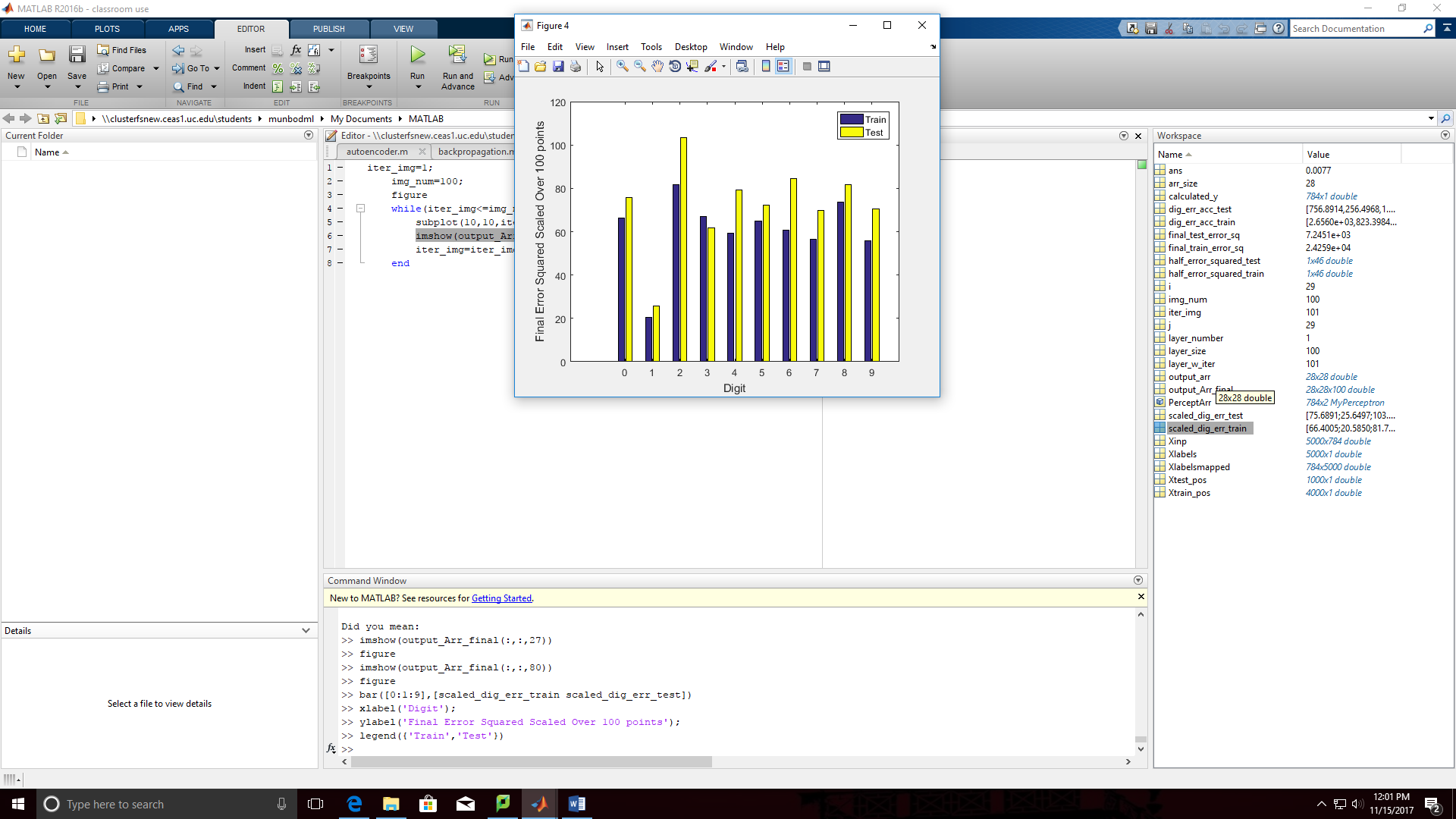


**Figure 10**: Rho Hat over Time



**Figure 11**: Error rate vs Epoch Number

I plotted the Rho hat over time in figure 10 to show the performance of my neural network. While Figure 11 is similar to Figure 1. I fine tuned the



**Figure 12**: Final Error Squared for Testing & Training Data Scaled Over 100 Points v/s Digit Label ***(Homework 3 problem 2)***



**Figure 13**: Average Error vs Label of MNIST Data ***(Homework 4 problem 1)***

# **Problem Two**

A flexible multi-layer feed-forward network need to be simulated in this question which uses back propagation with momentum step enabled in it.

The expectation is to use the “auto-encoder” network which was trained in previous question. The hidden layer weights are initiated using initial weights.

## **System Description**

In this Question the **Problem 1 from Homework 3** system was repeated. Hence to test the features from the auto-encoder, the exact same setup was used. Basically, **150 hidden neurons** were used which gave us the best-case simulation. It enabled us to accurately compare between the two scenarios.

Learning rate of ηh = ηo = **0.1** & momentum parameter (α) = **0.15 was used**.

Initialization of the Input to hidden layer weights was done using the same setup from the best-case simulation obtained from earlier problem. Initially, the weights from hidden to output layer was set by a random process which is based on **Gaussian** method.

The mean (µ) was set to 0 & Standard Deviation (σ) was set at √(2/n)

where n is equal to the number of neurons in the layer same as the best-case simulation.

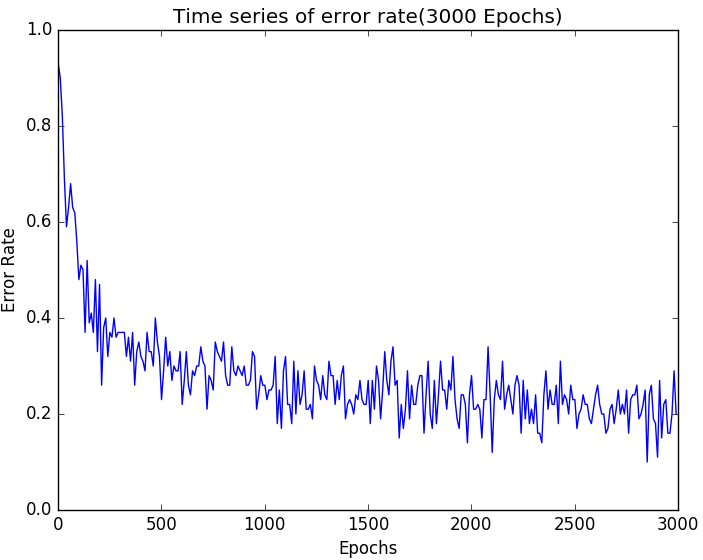
* The **learning rate** for the program was 0.06. If the learning rate was taken to be a small value than the neural network will be very slow. The value I choose wasn’t too high or too low. It was the ideal value for my network; if I had chosen a large learning rate than it could have caused the weights to move around from positive region to negative region but never providing an optimum result.
* The **Momentum** is also defined as alpha value which was set at 0.6 in the code. This momentum value was applied to equation of delta weights. 0.6 enabled faster execution.
* I considered only **one hidden layer** in the neural network while the output neurons we have the range of 0 to 9 values which is defined as output threshold.
* The initial weights were chosen such a way that they range from -0.1 to +0.1 this helps in good learning.
* The **criterion for deciding when to stop the training for our network**: For the first 3500 randomized data points out of 4000 data points was used for training. This was selected because the question mentioned that only a subset of the data should be used for training.

|  |  |
| --- | --- |
| **Parameters** | **Architecture 2** |
| **Learning Rate** | 0.06 |
| **Momentum parameter** | 0.6 |
| **Hidden neurons** | 100 |
| **Hidden Layer** | 1 |
| **Output neurons** | 10 |
| **Computation time** | 2 hours |
| **Coding style** | Matrix multiplication |
| **Epoch at which error rate stabilized** | 20 |

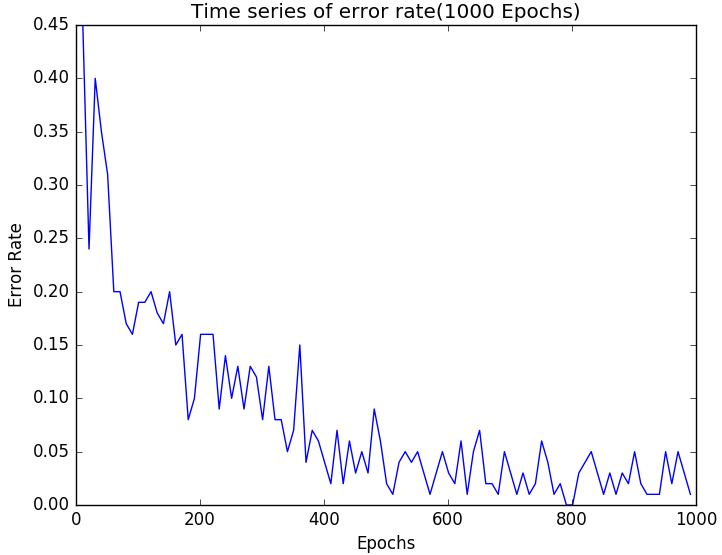
**Note**: Used Architecture 2 in the end for problem 1 of homework 3.

## **Results**

Using the above parameters, the final network was trained by Back-propagation using **stochastic gradient descent** method. Only the output layer weights were ‘learnt’ as part of this process. 100 random training points from the training set in each epoch & initially 1000 epochs were used. The error rates observed were higher when compared to the best-case simulation. Hence **3000 epochs** were used to observe the error rates over a longer simulation (300,000 points). The error rates were plotted for every 10 epochs as shown in Figure 11.



**Figure 11**: Error rates for every 10 epochs over 3000 epochs is plotted here. A Time series of data plot of error rates

****

**Figure 12**: Error rates for every 10 epochs over 1000 epochs is plotted here. A Time series data plot of error rates

(Obtained during simulation run of **Problem 1 in HomeWork3)**

I have generated the confusion matrix for both testing & training; this will help us to evaluate the performance of the architecture or the design. The table below indicates the confusion matrix for the training set of four thousand points (4000).

After 500 epochs, the error rate reduces or settles down at around 0.20. Initially, the error rates was high. Based on Max-threshold approach the final test hit rate was calculated it was 88.1 down from 93.1% when both the layers were trained.

From earlier problem; the average error rate obtained is noted to be **higher** when compared to the best-case simulation from previous problem in which - hidden & output layer weights were both trained. Please refer to Figure 12.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **A**  **C**  **T**  **U**  **A**  **L** |  | **PREDICTED** | | | | | | | | | |
| Digits | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **0** | 361 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 3 | 1 |
| **1** | 0 | 435 | 4 | 1 | 0 | 1 | 2 | 1 | 0 | 1 |
| **2** | 2 | 0 | 390 | 3 | 10 | 1 | 4 | 5 | 13 | 3 |
| **3** | 1 | 1 | 6 | 371 | 2 | 3 | 1 | 4 | 7 | 6 |
| **4** | 0 | 0 | 1 | 0 | 362 | 2 | 3 | 2 | 3 | 14 |
| **5** | 12 | 1 | 3 | 10 | 4 | 313 | 12 | 3 | 5 | 7 |
| **6** | 0 | 3 | 2 | 0 | 2 | 4 | 350 | 1 | 0 | 0 |
| **7** | 1 | 6 | 7 | 1 | 8 | 2 | 0 | 373 | 1 | 7 |
| **8** | 2 | 5 | 4 | 10 | 5 | 10 | 9 | 4 | 351 | 4 |
| **9** | 4 | 6 | 1 | 6 | 16 | 1 | 4 | 4 | 6 | 381 |

**Table 1**: the actual & predicted labels for training set (4000) points Confusion matrix

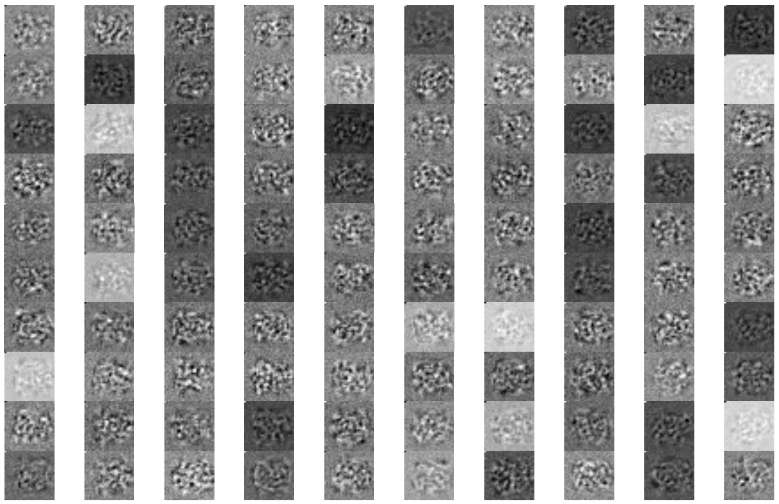
It’s evident that there is a significant increase in the mis-classification counts in this problem. Now please observe the table above focus on the mis-classifications (denoted by red), Digit 8 has the **most** mis-classifications which is around 53 & it is followed by digit 9 which has around 48 which is consistent with the earlier problem. It is important to note that most individual mis-classifications was seen/observed between the numbers/digits 9 to 4 & 2 to 8.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **A**  **C**  **T**  **U**  **A**  **L** |  | **PREDICTED** | | | | | | | | | |
| Digits | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **0** | 89 | 0 | 1 | 0 | 0 | 0 | 3 | 1 | 0 | 0 |
| **1** | 0 | 123 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 0 |
| **2** | 1 | 0 | 87 | 2 | 2 | 0 | 2 | 1 | 5 | 1 |
| **3** | 0 | 0 | 0 | 85 | 1 | 5 | 1 | 4 | 1 | 3 |
| **4** | 0 | 1 | 0 | 0 | 103 | 1 | 1 | 0 | 0 | 7 |
| **5** | 0 | 2 | 2 | 3 | 3 | 68 | 2 | 1 | 5 | 0 |
| **6** | 2 | 0 | 2 | 0 | 0 | 0 | 97 | 0 | 1 | 0 |
| **7** | 1 | 1 | 2 | 1 | 4 | 1 | 3 | 90 | 0 | 3 |
| **8** | 4 | 4 | 6 | 3 | 1 | 4 | 1 | 1 | 57 | 4 |
| **9** | 1 | 1 | 0 | 1 | 2 | 3 | 0 | 3 | 0 | 79 |

**Table 2**: the actual & predicted labels for testing set (1000) point’s Confusion matrix

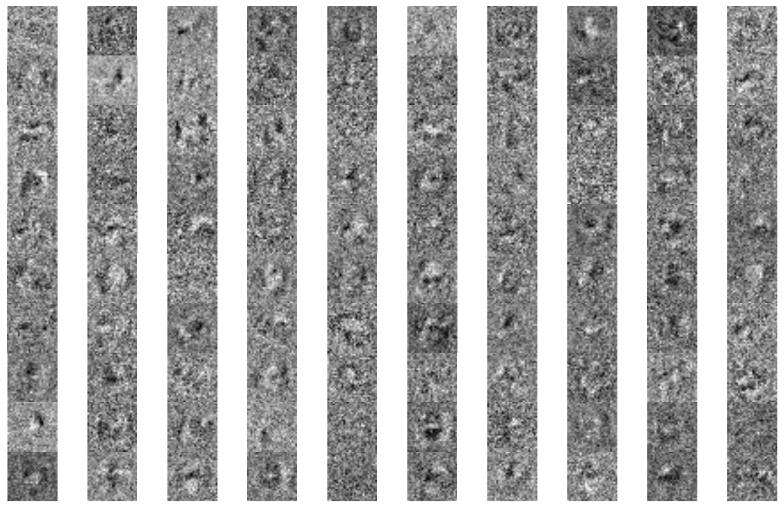
In the Table 2 the confusion matrix for the testing set is provided for (1000) points. Again, we can observe that most mis-classifications was seen for number/digit 8; the number of misclassifications was (twenty-eight) 28 with the most individual mis-classifications was observed between the digits (nine) 9 to (four) 4 & 2 (two) to (eight) 8.

From this we can conclude that the weights trained by the “auto-encoder” were probably not of high quality & did not necessarily identify distinct features. Features from” auto-encoder” simulation in Figure 13 & figure 14 still look more distinctive & clearer when compared to those from the best-case simulation **(problem 1 from HOMEWORK 3)** in Figure 14, but still resulted in higher error rates.



**Figure 13**: the hidden layer neurons from “auto-encoder” simulation via which the features was learnt

One possible explanation for the above observation is that the features obtained was learnt together from both the layers which lead to a best-case simulation **(Problem1 from HOMEWORK 3**) which was missed in this problem. In this question only the hidden layer features were initialized while the weights in output layer was chosen at random value.



**Figure 14**: Features learnt by hidden layer neurons from best-case simulation **(Problem 1 of Home Work 3)**

Table 3 (below): Final Configuration Matrix between the actual & predicted labels for Training Data (Over All 4000 Training Points) **(problem 1 from HOMEWORK 3)**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **A**  **C**  **T**  **U**  **A**  **L** |  | **PREDICTED** | | | | | | | | | | |
| **Digits** | **Zero** | **One** | **Two** | **Three** | **Four** | **Five** | **Six** | **Seven** | **Eight** | **Nine** |
| **Zero** | 357 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **One** | 0 | 453 | 2 | 3 | 0 | 0 | 2 | 1 | 0 | 0 |
| **Two** | 2 | 1 | 391 | 2 | 3 | 0 | 2 | 2 | 4 | 1 |
| **Three** | 0 | 0 | 2 | 397 | 2 | 0 | 0 | 1 | 1 | 2 |
| **Four** | 0 | 0 | 2 | 1 | 391 | 1 | 0 | 1 | 1 | 2 |
| **Five** | 0 | 1 | 0 | 1 | 3 | 343 | 2 | 1 | 2 | 2 |
| **Six** | 1 | 0 | 0 | 0 | 1 | 1 | 371 | 0 | 0 | 0 |
| **Seven** | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 418 | 0 | 0 |
| **Eight** | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 397 | 0 |
| **Nine** | 1 | 3 | 0 | 1 | 2 | 4 | 2 | 3 | 4 | 395 |

Table 4 (below): Final Configuration Matrix between the actual & predicted labels for Testing Data (Over All 1000 Testing Points) **(problem 1 from HOMEWORK 3)**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **A**  **C**  **T**  **U**  **A**  **L** |  | **PREDICTED** | | | | | | | | | |
| **Digits** | **Zero** | **One** | **Two** | **Three** | **Four** | **Five** | **Six** | **Seven** | **Eight** | **Nine** |
| **Zero** | 96 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| **One** | 1 | 108 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Two** | 1 | 2 | 105 | 1 | 3 | 1 | 2 | 5 | 2 | 1 |
| **Three** | 0 | 0 | 2 | 91 | 0 | 0 | 0 | 3 | 0 | 0 |
| **Four** | 0 | 0 | 0 | 0 | 102 | 0 | 0 | 0 | 0 | 3 |
| **Five** | 1 | 0 | 0 | 2 | 1 | 91 | 0 | 1 | 3 | 0 |
| **Six** | 1 | 1 | 0 | 0 | 1 | 1 | 80 | 0 | 1 | 0 |
| **Seven** | 0 | 2 | 1 | 0 | 4 | 0 | 0 | 80 | 0 | 1 |
| **Eight** | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 0 | 76 | 0 |
| **Nine** | 0 | 2 | 0 | 2 | 3 | 0 | 0 | 1 | 2 | 97 |