

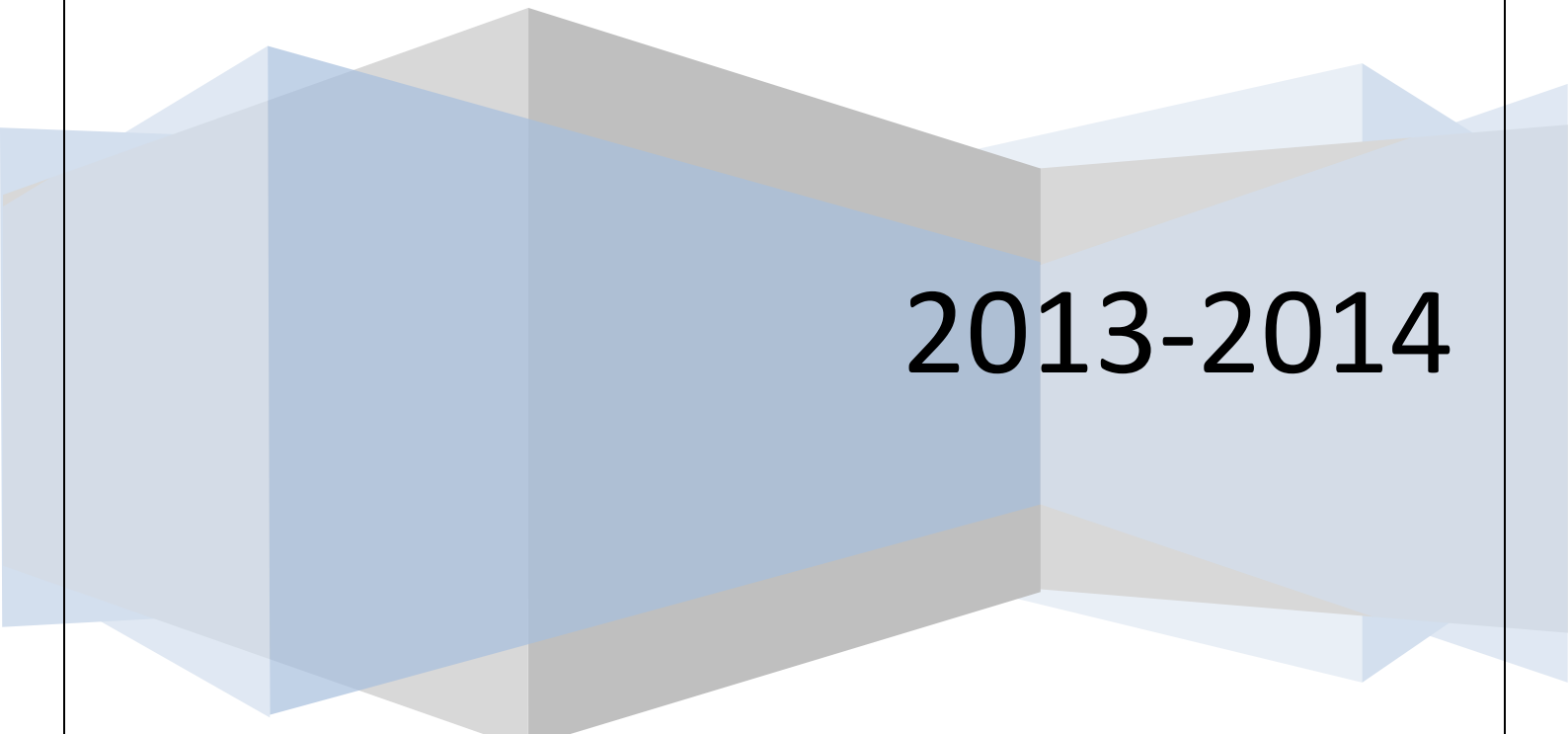
Machine Learning

Project 2 Report

By-

SIDDHARTH

50097583



2013-2014

1. Project Objectives:

Project objective is about multi-class classification of handwritten digit recognition. There are ten digits (0-9). Examples of each of the digits are given below.

0123456789

1.1 Data Set provided:

- 1.1.1 GSC features extracted from each of the image: each image is represented by a 512-bit Vector, the first 192 are G (gradient), the next 192 are S (structural) and the last 128 are C (concavity).
- 1.1.2 Handwritten digit images: in case you want to experiment with other feature extractors, you are given handwritten images (image type .png). These are segmented images scanned at a resolution of 100ppi and cropped.

1

Self Implemented Algorithms:

Logistic Regression Model:

For a large class of distributions, the posterior probabilities are given by a softmax transformation of linear functions of the feature variables, so that

$$p(C_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

with $p(C_2|\phi) = 1 - p(C_1|\phi)$. Here $\sigma(\cdot)$ is the logistic sigmoid function defined by (4.59). In the terminology of statistics, this model is known as logistic regression, Although it should be emphasized that this is a model for classification rather than Regression.

For a data set $\{\phi_n, t_n\}$, where $t_n \in \{0, 1\}$ and $\phi_n = \phi(x_n)$, with $n = 1, \dots, N$, the likelihood function can be written

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{1-t_n}$$

where the 'activations' a_k are given by:

$$a_k = \mathbf{w}_k^T \phi.$$

The likelihood function is then given by

$$p(\mathbf{T}|\mathbf{w}_1, \dots, \mathbf{w}_K) = \prod_{n=1}^N \prod_{k=1}^K p(C_k|\phi_n)^{t_{nk}} = \prod_{n=1}^N \prod_{k=1}^K y_{nk}^{t_{nk}}$$

Taking the negative logarithm then gives:

$$E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^N \sum_{k=1}^K t_{nk} \ln y_{nk}$$

which is known as the *cross-entropy* error function for the multiclass classification problem.

We now take the gradient of the error function with respect to one of the parameter vectors \mathbf{w}_j .

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = \sum_{n=1}^N (y_{nj} - t_{nj}) \phi_n$$

Input feature matrix contain 512 features of the image of all 10 digits in each input vector and have one associated target value as a value of digit. The target is to compute Yk relevancy label with the help of feature vectors.

Training Algorithm:

Input feature matrix contain 512 features of the image of all 10 digits in each input vector and have one associated target value as a value of digit. The target is to compute Yk relevancy label with the help of feature vectors.

1. Take the training set of every digit having 512 Dimension and append it vertically then you will have a matrix of 19978*512
2. Create one more dimension by appending the target label as 513 dimension.
3. In train_Ir strip of the Label from 513 dimension and make it into 1-K class encoding
4. Apply a bias parameter to the matrix making it again dimension of 513.
5. Create a random weight parameter by rand function of dimension =No of classification * number of dimension +1 = which is here 10*513.
6. Taking the product of Weight parameter and input matrix with bias parameter for calculating a_k :

$$a_k = \mathbf{w}_k^T \phi.$$

7. Taking the softmax of the a_k as:

$$p(C_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

8. We then compute the Error gradient and cross entropy error as:

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = \sum_{n=1}^N (y_{nj} - t_{nj}) \phi_n$$

$$p(\mathbf{T}|\mathbf{w}_1, \dots, \mathbf{w}_K) = \prod_{n=1}^N \prod_{k=1}^K p(C_k|\phi_n)^{t_{nk}} = \prod_{n=1}^N \prod_{k=1}^K y_{nk}^{t_{nk}}$$

9. Then adjust the weight parameter by gradual descent as:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n$$

10. Repeat the step 7 to 9 till the error comes into permissible limits. (8000 iterations).

11. Calculate the Weight parameter for minimum error.

Testing Algorithm:

1. Calculate the Y_k with the testing data:

$$a_k = \mathbf{w}_k^T \phi.$$

$$p(C_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

2. After Calculating the classifier change it to the respective digit as If 1 is at first position then digit will be "0";
3. Check the error with the actual result and calculate Error and Reciprocal rank.

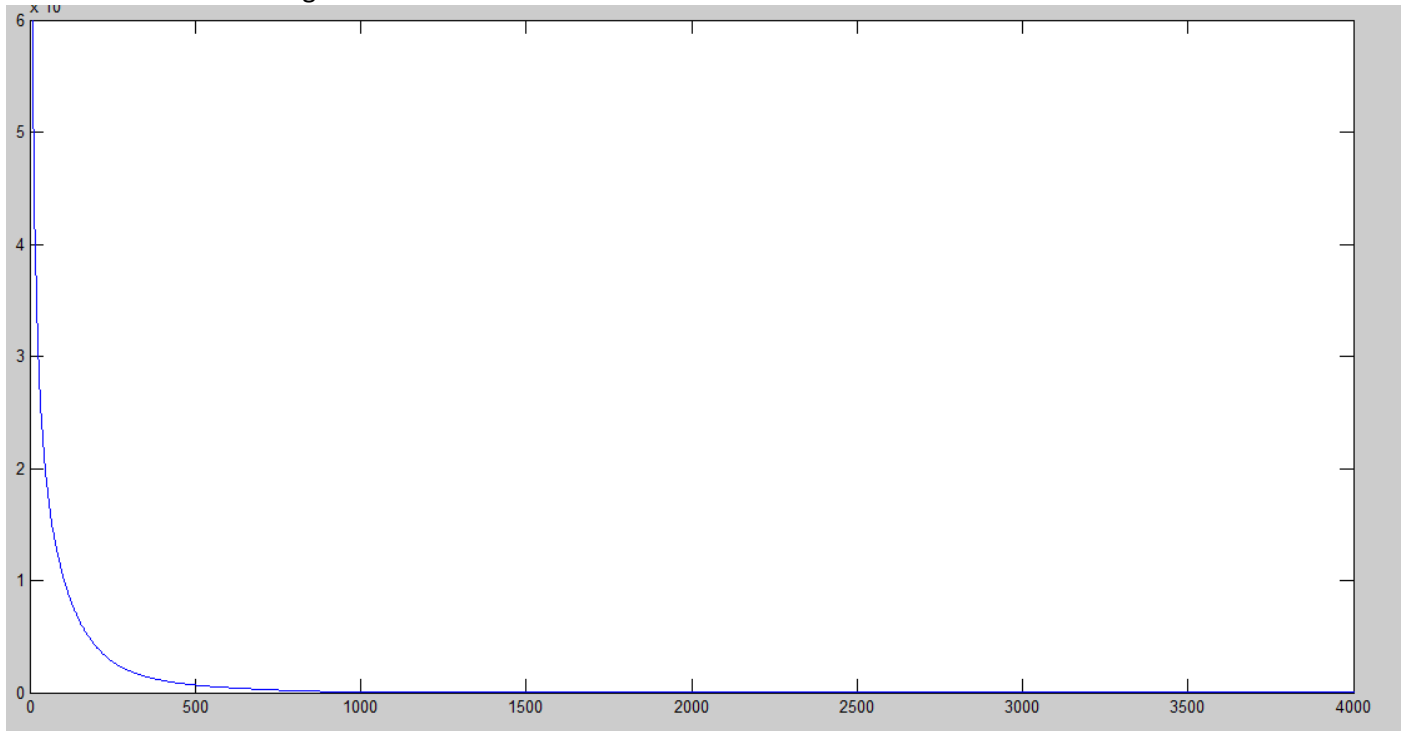
Analysis:

Various values are itta are tested with interation of 8000 to calculate lowest error value, and it was anlysed that with incresing value of itta error decresed faster and found a better lower value **but** also after a point in incresing value of itta the error always comes as NaN.

With itta= 0.0018 and interation =8000:

Cross entropy error= 3.7383

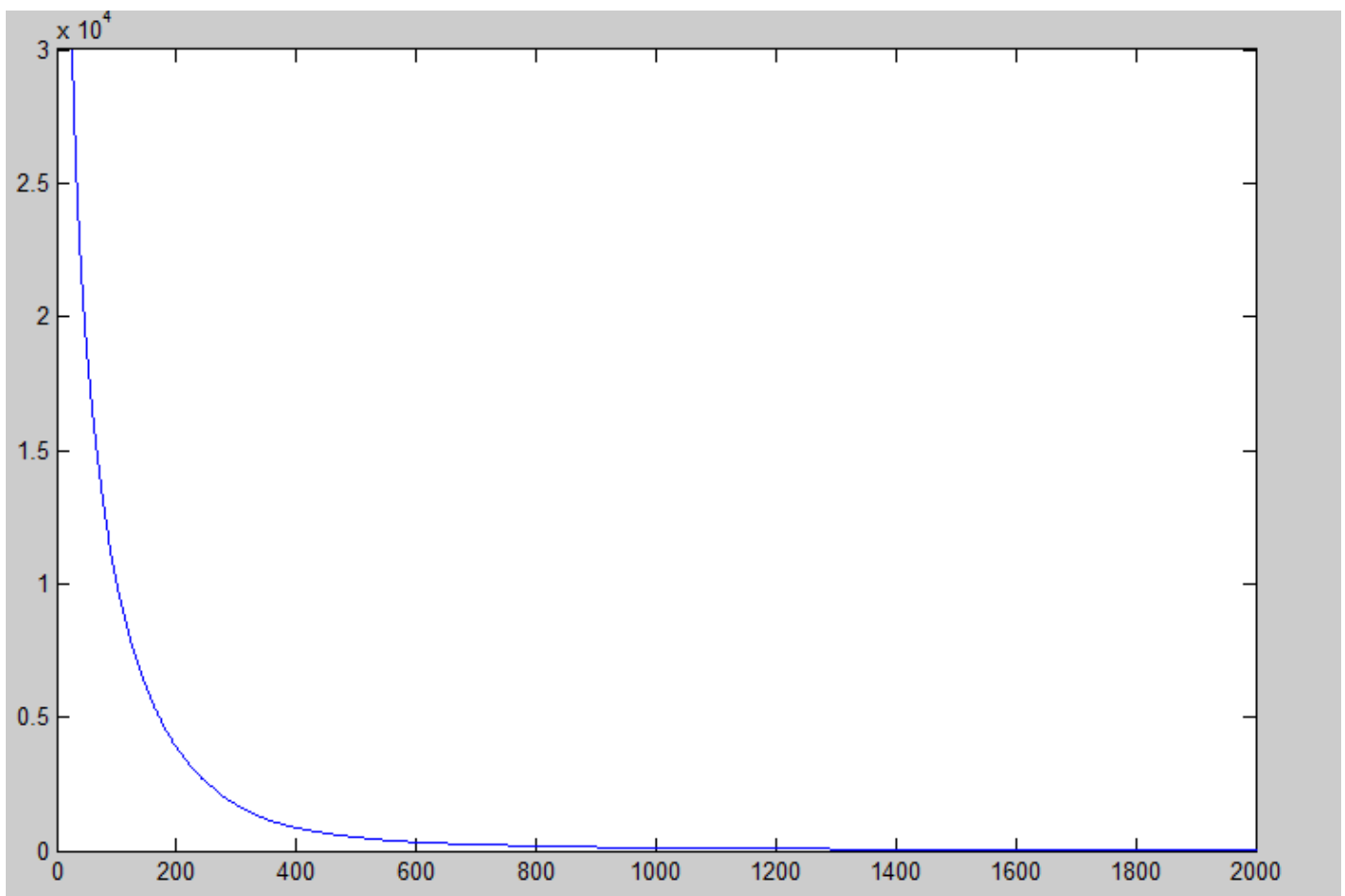
CSE 574 Machine Learning



(zoomed figure so only result for 4000 iterations are showing)

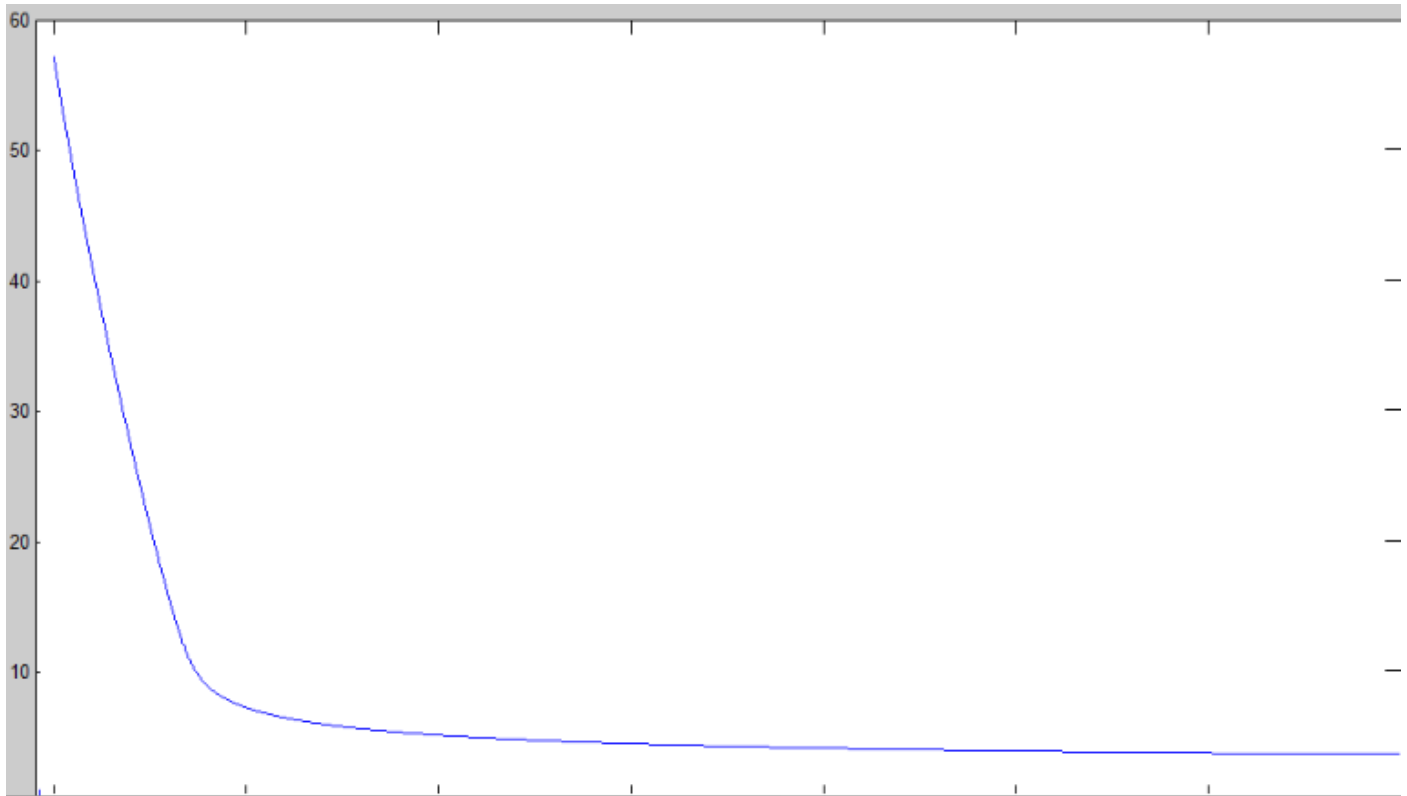
With itta=0.0019 and iteration=8000:

Cross entropy error= 3.6616



(zoomed figure so only result for 2000 iterations are showing)

Cross entropy error= 3.578977201286208



Result (On testing Set):

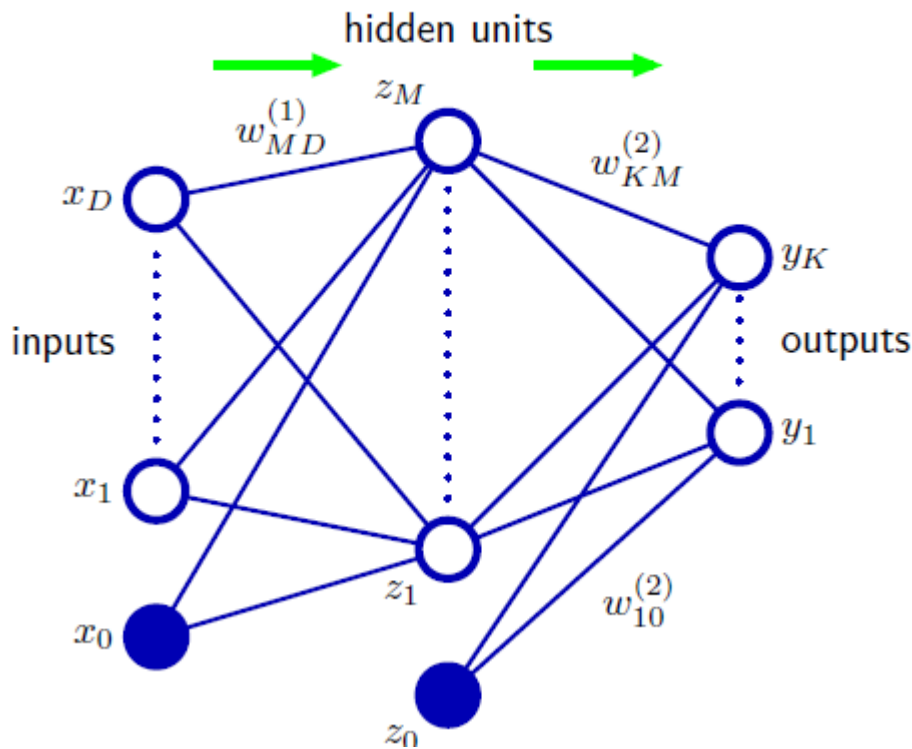
Tested with weight calculated with $\text{it_ta}=0.0020$

Error Rate= $\frac{1500-1461}{1500}$

=0.026=> 2.6%

Reciprocal rank=0.97

1. Neural Network Model:



Neural Network can be described a series of functional transformations. First we construct M linear combinations of the input variables x_1, \dots, x_D in the form.

$$a_j = \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

where $j = 1, \dots, M$, and the superscript (1) indicates that the corresponding parameters are in the first 'layer' of the network. The quantities a_j are known as activations. Each of them is then transformed using a differentiable, nonlinear activation function $h(\cdot)$ to give

$$z_j = h(a_j).$$

We can combine these various stages to give the overall network function that, for sigmoidal output unit activation functions, takes the form:

$$y_k | (\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=1}^M w_{kj}^{(2)} h \left(\sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

Training Algorithm:

1. Repeat the steps 1-5 from earlier algorithms.
2. Calculate the Winput and Woutput with rand funtion.
3. Calculate a_j with Winput:

$$a_j = \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

4. Calculate z_j and y_k :

$$z_j = \tanh(a_j)$$
$$y_k = \sum_{j=0}^M w_{kj}^{(2)} z_j.$$

5. Calculate Error gradient :

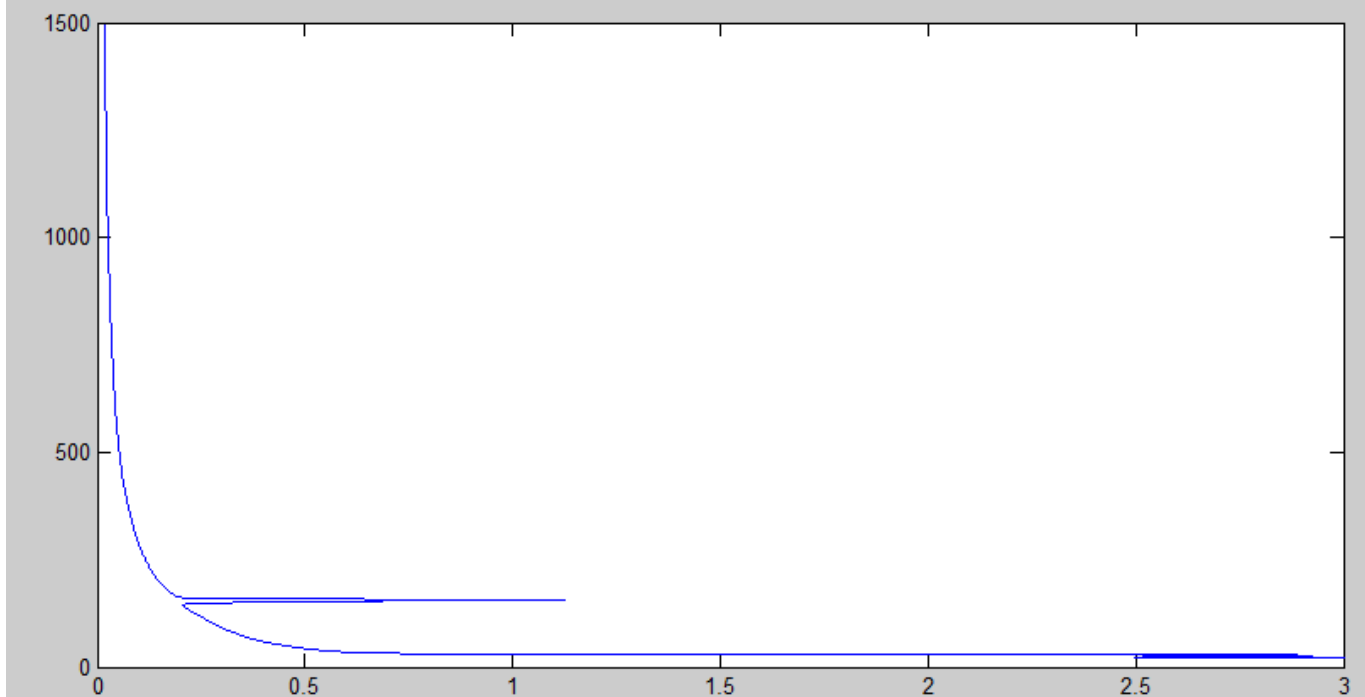
$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j x_i, \quad \frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k z_j.$$

6. Use gradient descent to recalculate weights.

Analysis:

We use different value of M and iteration of value=4000 to calculate the weights weights:

Using M= 20:



Results:

Using M=20:

Error Rate:

$$=1500-1454/1500= 0.0306$$

$$\Rightarrow 3.06\%$$

Reciprocal rank=0.96

External Package Used:

SVM Model:

In order to compare the performance SVM model, the libsvm has been used for training and testing the classification for digit recognition. We train one-vs-all classifiers using LIBSVM [5], one for each digit. A test example is assigned the class with the highest posterior probability which is estimated based on the margin of the test example.

SVM was able to classify 1476 test samples successfully with an error rate of 1.6%.

Procedure :

CSE 574 Machine Learning

1. Compile libsvm package using make file for svm_train.c and svm_predict.c

2. Use svm_train to train the model using the command “svm-train svminput”

3. Use svm_predict to classify using the learned model. Use the command

“svm-predict svmtestinput svminput.model out.txt”

Analysis :

```
optimization finished, #iter = 403
nu = 0.075372
obj = -194.346402, rho = -1.306645
nSV = 361, nBSV = 251
Total nSV = 4954
mata@mata-Lenovo-Z580:/media/xabi/Dropbox/ML/Project_2/libsvm-3.17$ ./svm-predict svmtestinput svminput.model svm_pred.txt
Accuracy = 98.4% (1476/1500) (classification)
```

Accuray : 98.4%

Error Rate: 1.6%