CSE 574

Introduction to Machine Learning

Project Report on Multiclassification on LR and Neural Network

Submited by:

Munish Mehra

50097875

PROBLEM OVERVIEW

This problem is about multi class classification of numerical image data which consitutes of digits 0,1,2,3,4,5,6,7,8 and 9. We are provided with features of these data in indivitual files of classes. For this we need to create our own target vectors as we already know which data set belongs to which image.

Our main target is to come up with Logistic Regression model and Neural Network multi classification model which can classify between these classes.

And besides these 2 methods we need to implement one extra package and compare these 3 on basis of 2 following Evaluation Metrics:

1. Error rate
2. Reciprocal Rank

LOGISTIC REGRESSION:

Well in Logistic regresion we input our training data in Hypothesis function which actually calculates the probability that whether that image belongs to a particular class or not. We are going to train our model by minimizing error by Error fucntion and parallelly implementing the gradient desent formula to find optimum values of Weight vector.

Choise of Model:

In this project one vs All method is applied in which w assume that even if their are 10 classes that only one class as 1 (for which we want to train our weight vector)and rest all classes as 0. We repeatedly train out model by taking target vector for that class as 1 and rest as 0.

e.g : for class 0 target vector was [1 0 0 0 0 0 0 0 0 0]

Hypothesis Function:

The Hypothesis function is taken as

$$y(\boldsymbol{\phi}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}\right)$$

in which W is for particular class for which we want to train our model and phi is whole data set.

Where

$$0 \leq h_\theta(x) \leq 1$$

and

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

NOTE: here h is same as y in above equation, x is same as phi and theta is same as W.

Error function used for Logistic regression:

Following is Error function which is being used for LR, Where y is used as target vector and h is used as sigmoid function.

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Second term will be zero if our target is 1 else it will be calculated. Which is shown as follows:

$$= \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

this would be computed for each image and and following this we need to compute gradient which is basically derivative of error fucntion calculated as below to reach min of error :

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

(simultaneously update all θ_j)

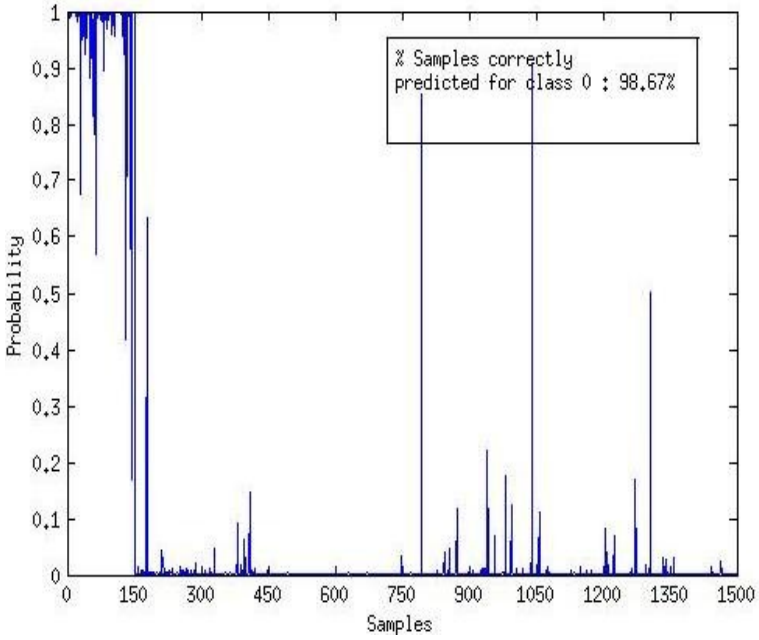
$\}$

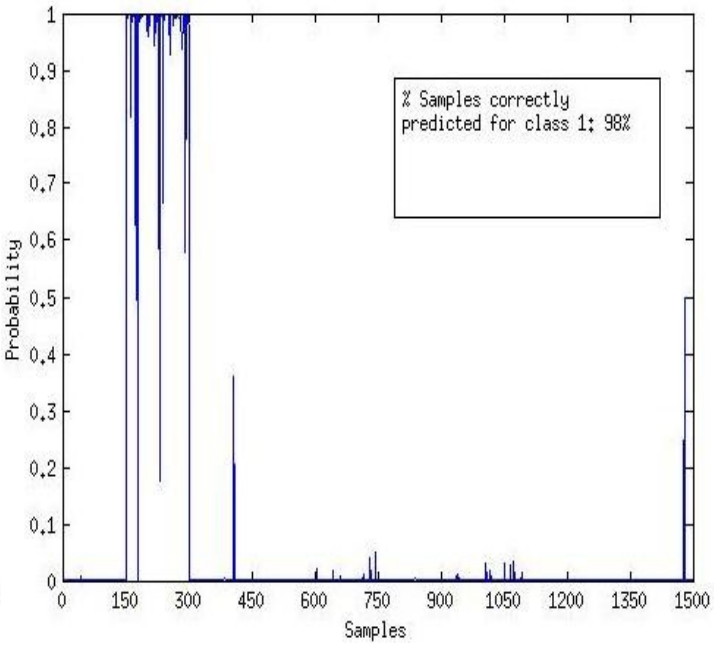
Where alpha is learning rate which i have taken as 0.2.

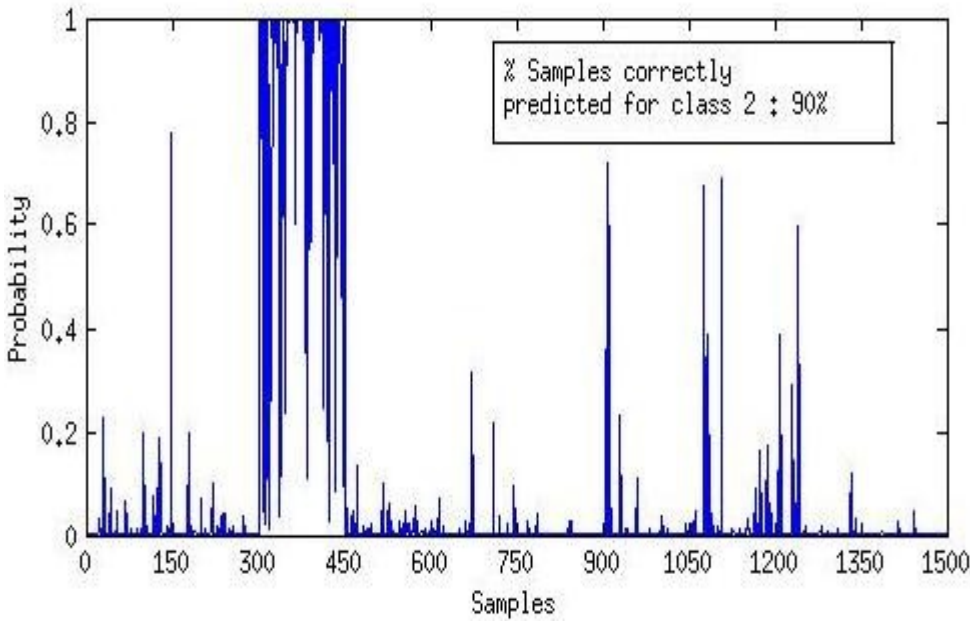
this is basic process that i have followed.

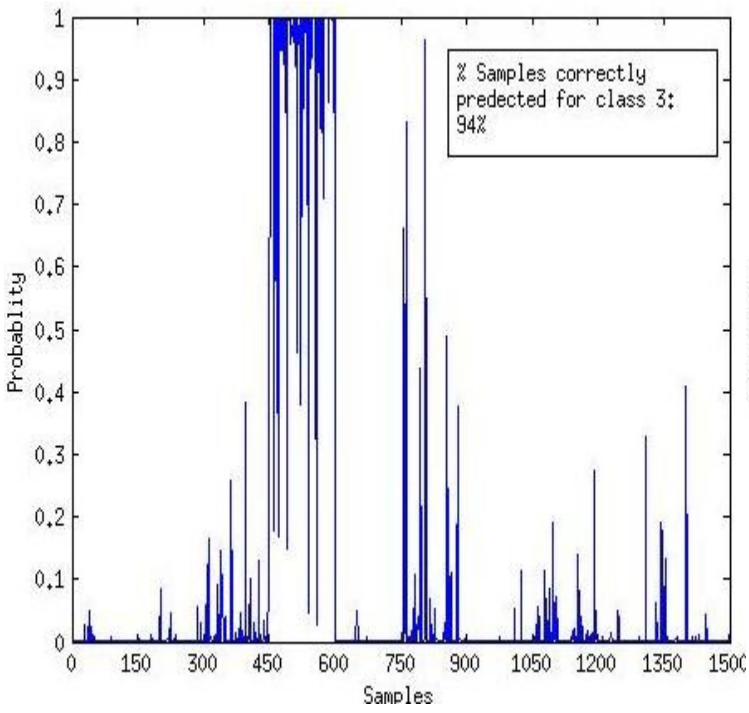
One last understanding is that i have computed 10 different wieght matrix for 10 classes and these are used on test set and results of which is shown below.

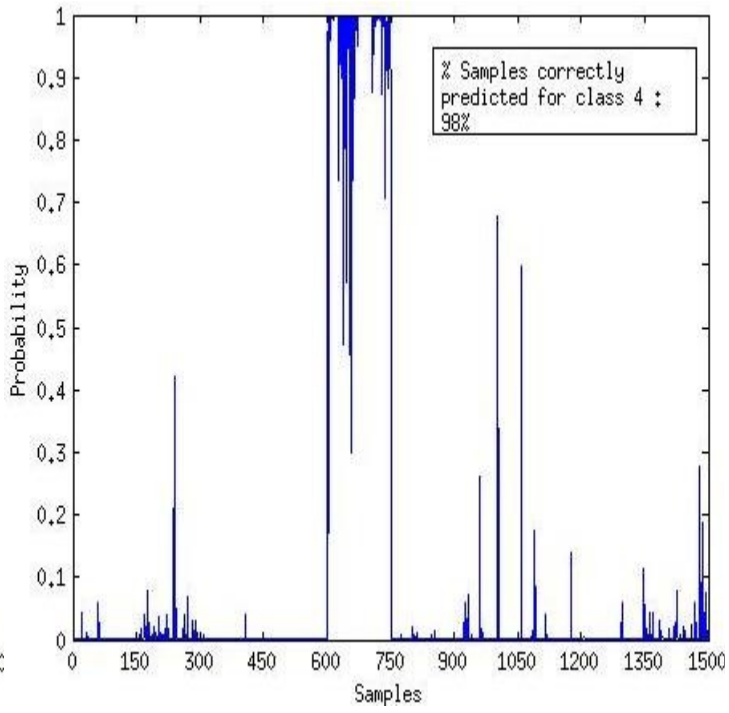
These are basillay graph for probability vs test data size.
And % computed is how many samples are computed correctly in that particular class.

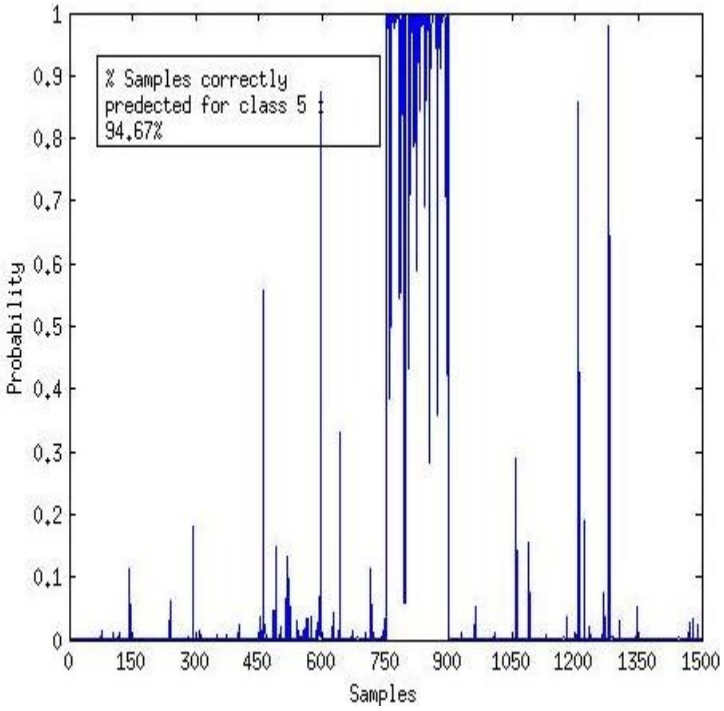


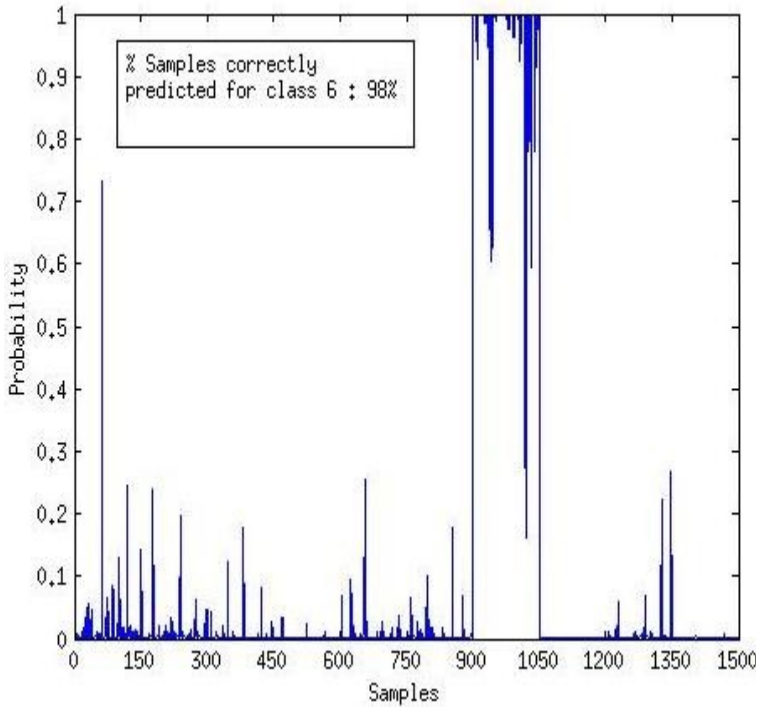


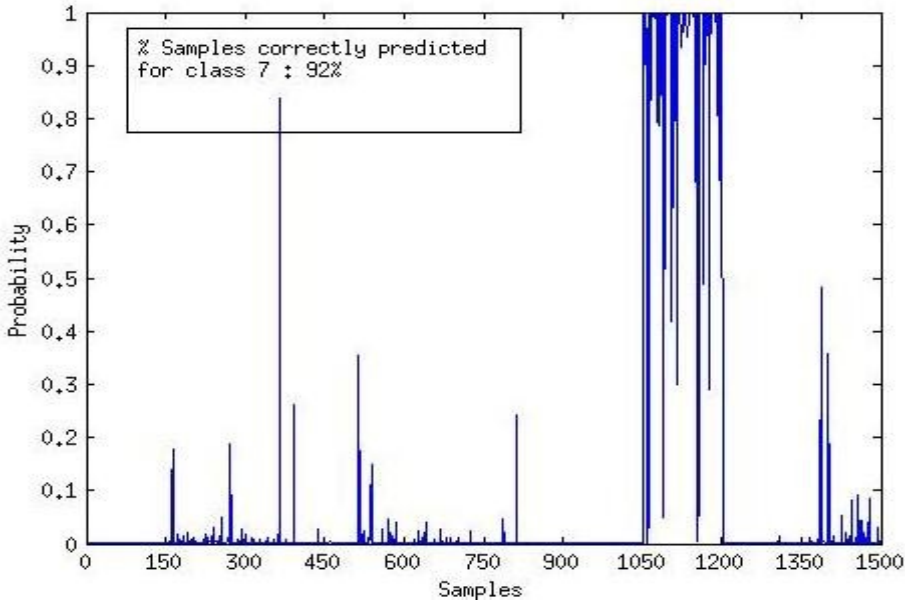


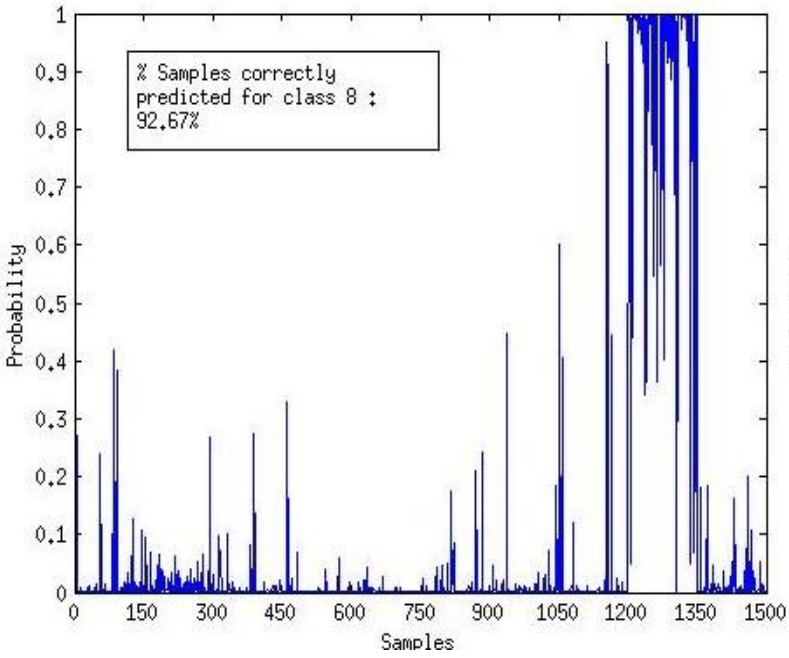

% Samples correctly predected for class 3: 94%
Probablity
Samples

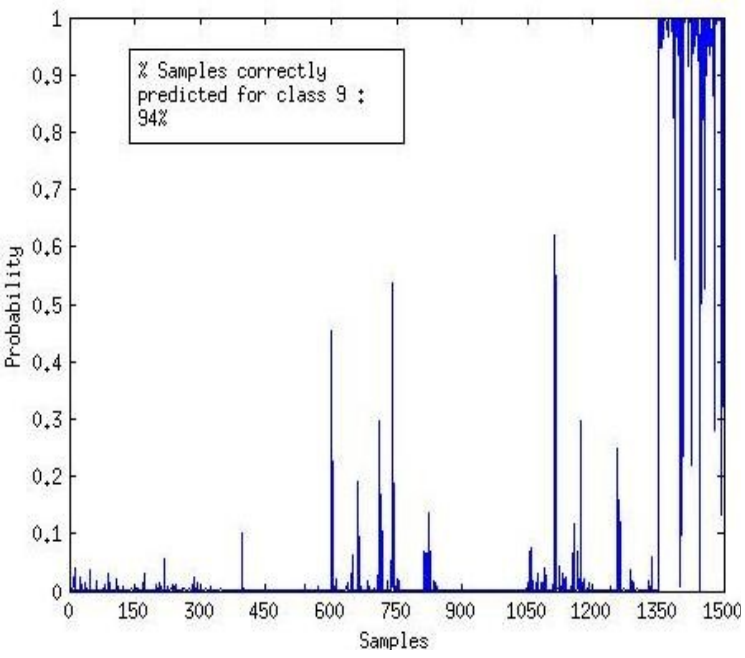

% Samples correctly predicted for class 4 : 98%
Probability
Samples


% Samples correctly predected for class 5 : 94.67%
Probability
Samples


% Samples correctly predicted for class 6 : 98%
Probability
Samples

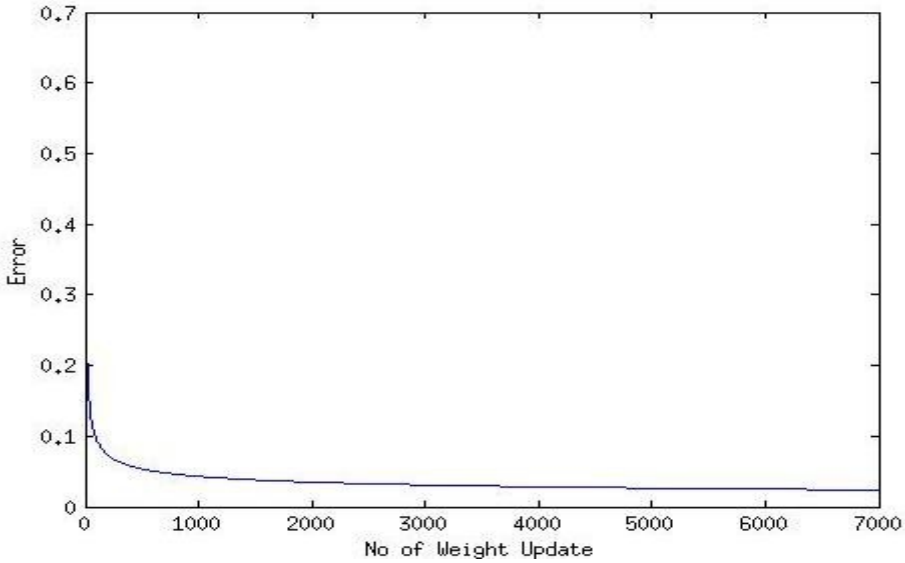






a

The Error Graph for same is shown below:



Evaluation Metrics for Logistic Regression :

1. Error rate: 5.0667%

2. Reciprocal Rank

	Reciprocal Rank
Class 0	1/1
Class 1	1/2
Class 2	1/1
Class 3	1/1
Class 4	1/1
Class 5	1/1
Class 6	1/1
Class 7	1/1
Class 8	1/1
Class 9	1/1

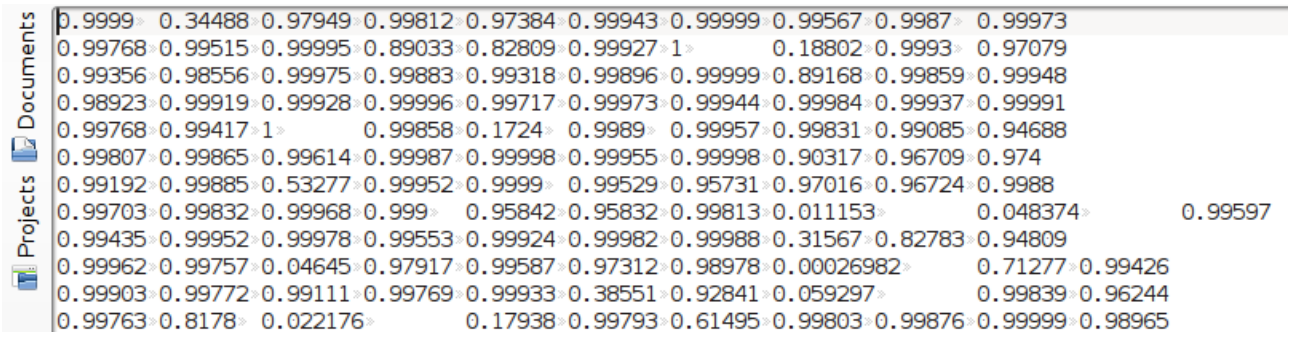
Parameter choice for LR:

1.Lambda : This was taken as 0.01 for LR as changing this would not make any much diff so this was not varied much to check error.

2.Alpha : Step size Alpha was taken as 0.2 .

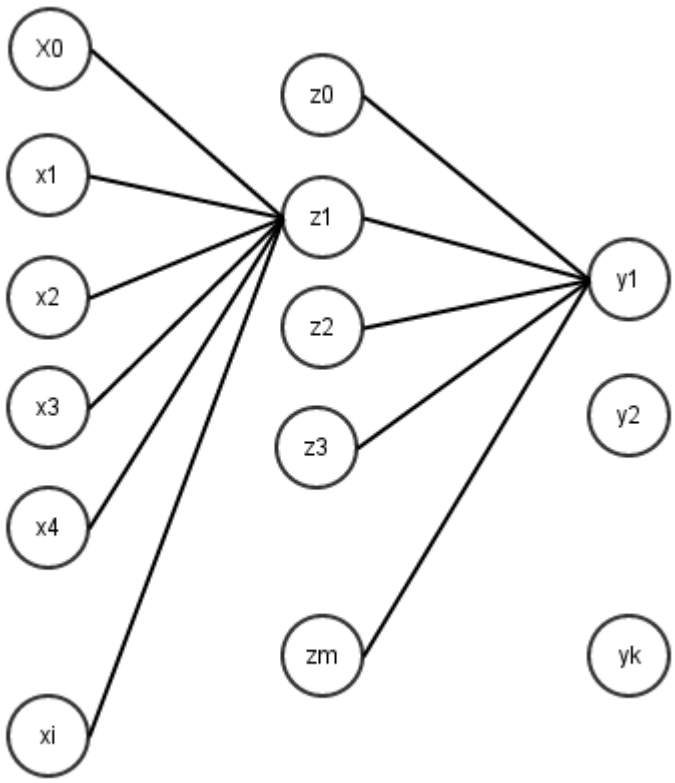
Probability shown in output(classes_lr.txt):

Each row repesent each class and coloum represent probability for each image:



Neural network :

For Neural Network i have taken 3 layer network and no of hidden neurons are taken as 25.



Above archietecture is adopted to compute NN classification model where i=513, m =25 and k =10.
z in above is not sigmodal output, a=sigmoid(z).

Following steps are applied to train NN:

1. Feed forward :

$$
\begin{aligned}
a^{(1)} &= x \\
z^{(2)} &= \Theta^{(1)} a^{(1)} \\
a^{(2)} &= g(z^{(2)}) \quad (\text{add } a_0^{(2)}) \\
z^{(3)} &= \Theta^{(2)} a^{(2)} \\
a^{(3)} &= g(z^{(3)}) \quad (\text{add } a_0^{(3)}) \\
z^{(4)} &= \Theta^{(3)} a^{(3)} \\
a^{(4)} &= h_\Theta(x) = g(z^{(4)})
\end{aligned}
$$

This is for 4 layer network applying same technique we can compute for 3 layer

2. Back propagation .

To compute error at each layer i applyed following:

$$\delta_j^{(4)} = a_j^{(4)} - y_j$$

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$$

algorithm for back propagation:

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$

$\triangle_{ij}^{(l)} := \triangle_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

$D_{ij}^{(l)} := \frac{1}{m} \triangle_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ if $j \neq 0$

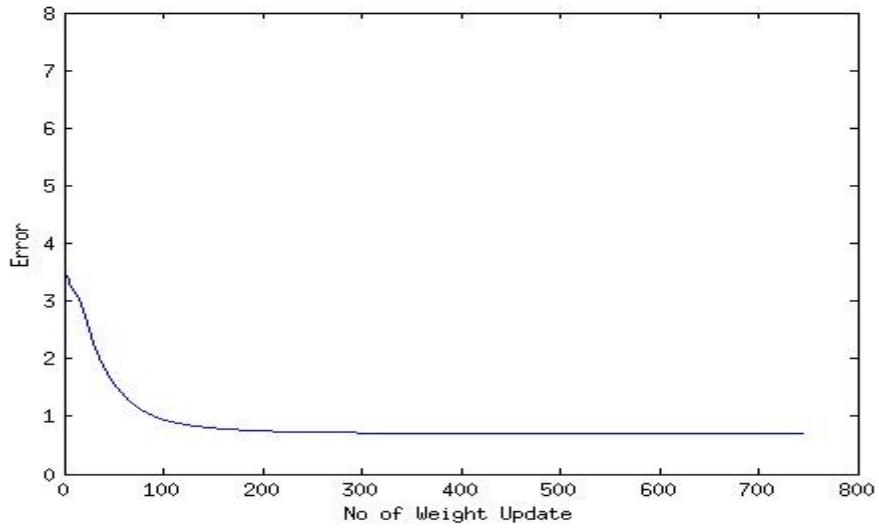
$D_{ij}^{(l)} := \frac{1}{m} \triangle_{ij}^{(l)}$ if $j = 0$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

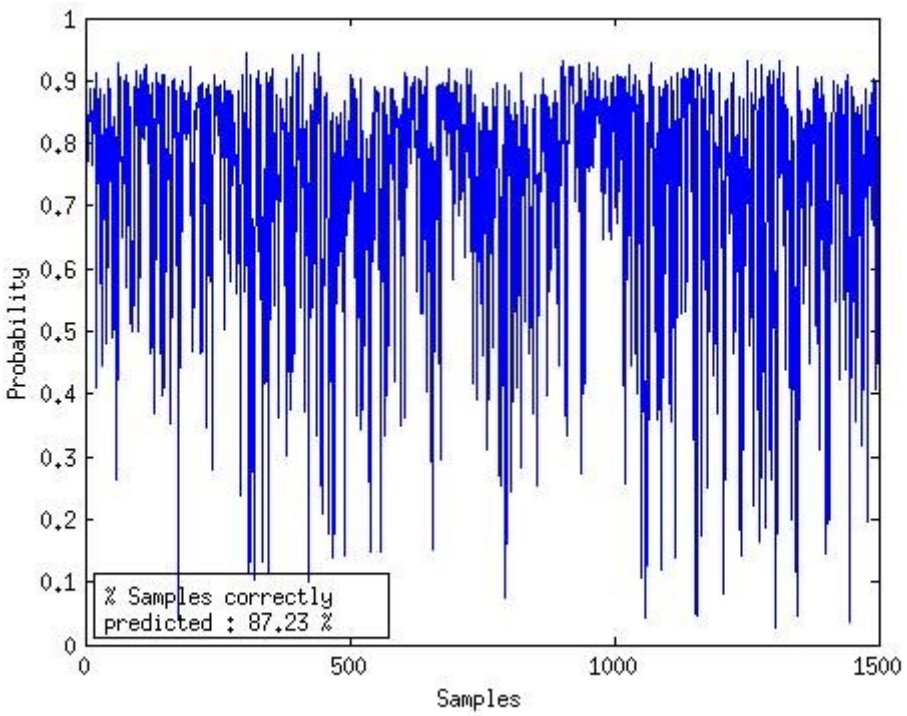
Finally when we have derivative if particular layer then i apply following formula for weight update:

Wnew = Wold – alpha*(derivative of gradeint at that layer).

Above steps are repeated for some iterations and results are ploted below:



Probability graph for whole test set shown below which image has what probability



Output probability is shown as follows for 8 image which belongs to class 0:

First row higher prob predicted that it belongs to class 0:

1	0.8709	0.0138	0.0314	0.0262	0.0068	0.0304	0.0295	0.0119	0.0385	0.0297
2	0.6401	0.0362	0.0186	0.0114	0.0083	0.0213	0.0435	0.0118	0.1233	0.0412
3	0.7733	0.0262	0.0269	0.0096	0.0101	0.0408	0.0499	0.0120	0.0449	0.0131
4	0.8445	0.0084	0.0301	0.0207	0.0071	0.0333	0.0510	0.0201	0.0337	0.0343
5	0.8342	0.0175	0.0272	0.0123	0.0030	0.0226	0.0466	0.0133	0.0917	0.0251
6	0.7702	0.0106	0.0199	0.0285	0.0024	0.0321	0.0317	0.0435	0.0412	0.0204
7	0.8156	0.0109	0.0182	0.0192	0.0025	0.0421	0.0422	0.0274	0.0520	0.0222
8	0.8519	0.0091	0.0435	0.0312	0.0033	0.0223	0.0578	0.0188	0.0317	0.0195

Evaluation Metrics for Neural Network :

1. Error rate: 12.8667

2. Reciprocal Rank

	Reciprocal Rank
Class 0	1/1
Class 1	1/1
Class 2	1/1
Class 3	1/1
Class 4	1/1
Class 5	1/1
Class 6	1/1
Class 7	1/1
Class 8	1/1
Class 9	1/1

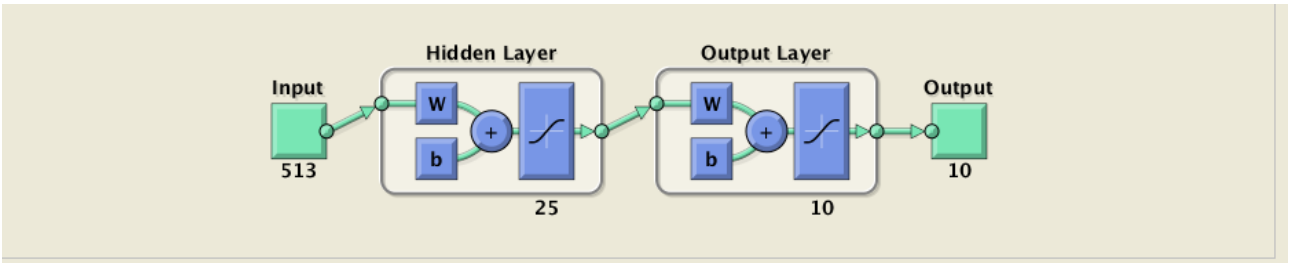
Parameter choice for Neural Network:

1.Lambda : This was taken as 0.01 for LR as changing this would not make any much diff so this was not varied much to check error.

2.Alpha : Step size Alpha was taken as 1 as this give significant results as expected so i did not decrease or increase it.

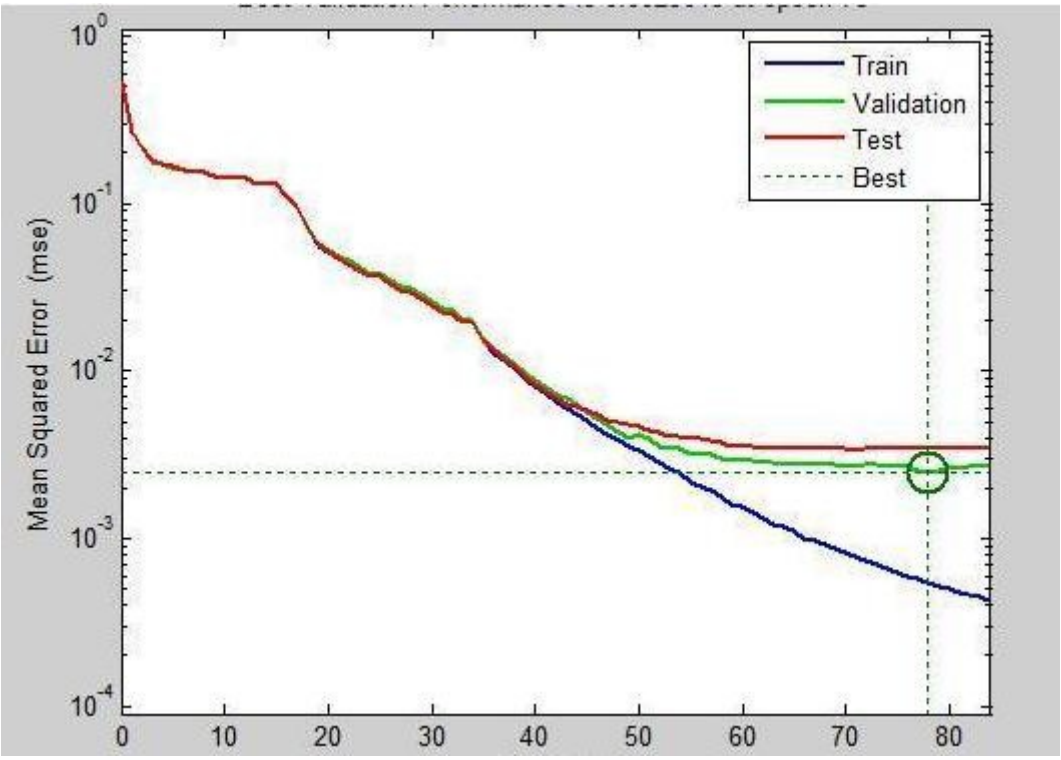
Neural Network Package:

The 3 layer network is choosen as below in which 513 are features in input layer and in hidden layer their are 25 nodes and in output layer their are 10 output units.



%Error calculated for this network is 10%.

Error for NN package:



Comparision:

For LR the error predicted is 5 around and for both NN imlemented and NN package the error is about 12 and 10 percent.

But if we compare LR and NN with reciprocal rate the first correct image prediction is good in NN than LR but in case of %correct prediction LR is far ahead from NN.