University at Buffalo, State University of New York

Computer Science and Engineering

Introduction to Machine Learning

# Project Assignment 2

*Author:*
Srinivasan Rajappa

*Supervisor:*
Prof. Sargur N Srihari

December 16, 2014

**Abstract**

The goal of classification is to take input vector $\mathbf{x}$ and assign it to $K$ discrete classes $C_k$ where $k = 1,2....K$. Usually the classification involves a line seperating the set of points pertaining to certain class. For classifying there are various techniques that can be used. The efficacy of the techniques can be differentiated by the type of classes in analysis, number of data and number of features.

For example in anlysis involving medical patients one can observe there will be lot of features (**D**) in the order of 20000 or more (genetic data) while the total number of patients (**N**) will be finite order of 20. For other study where vehicular traffic needs to observed, the number of features may be finite in the range of 0-100 while the data will be large of the range 20000 or more.

So, it is essential to understand how classification is applied on the data and more importantly the ground work needs to be prepared to how the problem of classification needs to be apporoached.

# 1 Introduction

Building on the abstract in the previous section, the ***Project 2*** is aimed at classifying hand written numerals. There are ten classes, each representing a numeral from 0-9. The feature set and data is derived from the GSC feature extractor developed at CEDAR/UB. The G(*gradient*), S(*structural*) and C(*concavity*) are the features that seek the unique property of the handwritten numeral. The GSC feature extractor is capble of extracting such features from an image. This is feature extractor is highly effective and is developed at CEDAR/UB.

The objective of this project is use these features and train the machine to identify the numeral and report the misclassifications which were made during the training. The classification is done using two popular classification algorithms viz. Logistic Regression and using Nueral Networks.

## 2 Logistic Regression

This classification algorithm estimates the likelihood of occurance of a class $C_k$ so the estimation function looks as follows.

$$p(C_k|\phi) = y(\phi) = \sigma(\mathbf{w}^T\phi)$$

For multi-class regression we must use the

$$p(C_k|\phi) = y_k(\phi) = exp(a_k)/\sum_j exp(a_j)$$

The formula is key to finding the weight parameters essential to model the correct classification. In multi-class classification which is the case over here the *basis* function is the equivalent data which is $\mathbf{x}$ with a bias parameters. Here $\mathbf{a}$ is called activation function and it is represented as follows.

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

It is important to understand how effective our weight parameters. Thus one needs to take steps that will ensure that the classification results are in perfect alignment. We would like to have an error function that will help us finding the weights. This process must be brought about iteratively. The Error function that is used in multi-class classification is called *cross-entropy* function, which is represented as follows:

$$E(w_1, w_2, ..., w_k) = -\ln p(T|w_1, w_2, ..., w_k) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} ln(y_n k)$$

On deriving this function with respect to w we get the gradient which we shall use in the phase of iterative gradient descent. This is also called ***Newton-Raphson*** method which uses series of steps with the gradient function to reach the local minima. On deriving we retrieve the following function.

$$\bigtriangledown_{wk} E(w_1, ..., w_k) = \sum_{n=1}^{N}(y_{nk} - t_n k)\phi(x_n)$$

Now for iterative calculation of the weights we need to use the above error function in coming up with the updated weights.

$$w_k^{\tau+1} = w_k^{\tau} - \eta \bigtriangledown_{w_k} E_n$$

After a set of iterations one can can find the correct weight parameters. These values can then be used to find the correct predictions on a set of input data and it's associated features. In this project the program is run using the **main.m** which calls the training phase and testing phase of the multiclass logistic regression classification.

## 2.1 Training-Phase

1. First we create the PHI matrix that contains the first column is always value 1 as it acts as a bias.
2. Create the T matrix which represents each class. For example class 1 will be represented by the vector.

$$(0100000000)^T$$

3. Finding the activation function $a_k$.
4. Calculating the cross-entropy error function.
5. Performing the Newton-Raphson to find the new weight vectors.
6. After a series of steps we obtain the weight values tthat will be used in the testing phase.

## 2.2 Testing-Phase

1. Use the updated W vectors and multiply with PHI matrix allocated for the testing.
2. The resultant vector has real values, the entry with maximum value in a row is to be considered 1 and the rest 0.
3. This can then be compared with the ground truth to obtain to how accurate is our implementation.

## 2.3 Output

For the present scenario. It is run for 2000 iterations which should considerably provide an outlook for the remainder of the iterations till 45000.
Total training data-set is ($N$)- 20000
Total testing data-set is - 1500
The total number of feature is ($M$)- 512
The total number of classes is ($K$)- 10
The Error Rate is ***2.566***
The Reciprocal Rank is ***1***

# 3 Neural Networks

Neural Network Model is based out of how nuerons behave. The first study that helped us to understand the potential, the experiments of Hubel and Wiesel[1] exhibits the same. Here receptors and sensors were connected to the hind side of the brain of a cat. The cat was shown a flouroscent bars in various orientations about a 2D-axis. The results that was obtained were maximum in only a specific orientation of the bar. It concluded that the nerves were behaving like a special filters that were only responding to a particular orientation of light. This concluded the fact that each neuron has the capability to use the inputs and relay the information to the other neurons and so on. This brought up the conclusion that the model was effective because the error-rates are neglible as biological animals tend to provide an accurate perception using the neurons. The mathematical expression of the same is represented below:

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

Here The hidden neural network is represented as a suprscript in the above functions. (1) represents hidden layer 1. In this layer the function applied *h(.)* is a sigmoid function.

The function *f(.)* is a softmax function which is represented by the following formula. Where $a_j$ are activations.

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

Cross entropy error function is calculated and it is used to perfrom gradient descend. After calculated and converging upon a correct minima Error *Backpropogation* is performed. This ensures a global optimal minima is reached for the myriad cost function which we have. The same is performed using the following function.

$$\delta_k = y_k - t_k$$

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

[1]Hubel and Wiesel- The neural basis of visual perception

This will find the hidden units and we use these above values to perform the gradient descent. The derivative with respect to weights in the respective units will help us converge to a relavent minima. *Note:* The amount of hidden nodes in a nueral is taken to be 2/3rd the number of features. The more the number of hidden nodes the better they will yield results.

## 3.1 Training Phase

1. Create the $w_{ji}$ and $w_{kj}$ with random values.
2. Calculate the activation units $a_j$.
3. Calculate hidden units using a sigmoid function, here it exp is used.
4. Calculate the activation units $a_k$ and $a_j$.
5. Calculate the soft-max i.e. $y_k(\phi)$.
6. Calculate the cross entropy followed by the deltas and then the derivates of the same.
7. Perform the gradient descent.
8. Derive the weight parameters.

## 3.2 Testing Phase

1. Utilize the weight paramteres viz. $w_1$ and $w_2$.
2. Calculate activation units.
3. Calculate the hidden unit values.
4. Calculate the activation units.
5. Calculate output predictions i.e. $y_k$.

### 3.3 Output

For the present scenario. It is run for 1000 iterations which should considerably provide an outlook for the remainder of the iterations till 45000.
Total training data-set is ($N$)- 20000
Total testing data-set is - 1500
The total number of feature is ($M$)- 512
The total number of classes is ($K$)- 10
The Error Rate is ***2.233***
The Reciprocal Rank is ***0.000666***

## 4 Neural Network Toolbox

Pattern recognition networks are feedforward networks that can be trained to classify inputs according to target classes. The target data for pattern recognition networks should consist of vectors of all zero values except for $a_1$ in element i, where i is the class they are to represent. Here the use of this technique helps us to visualize the results. The values are fed explicitly and we have output representing error values etc.

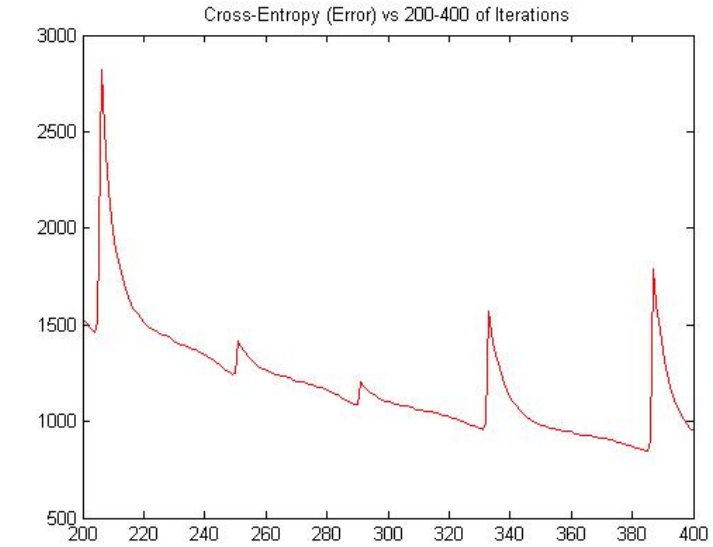
### 4.1 Output

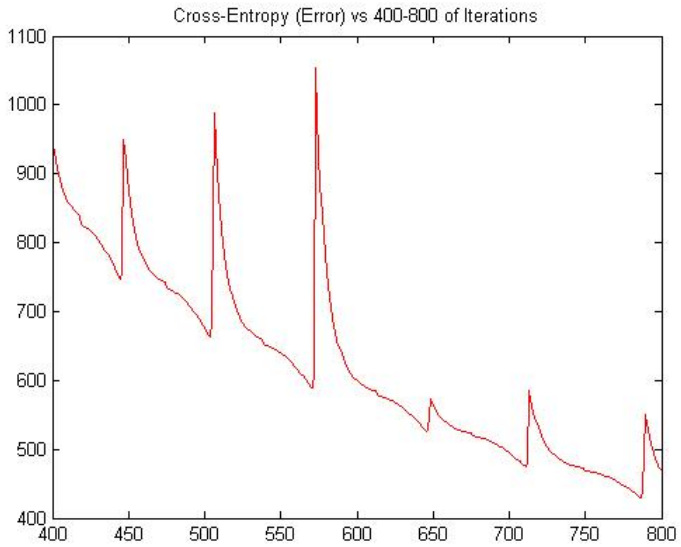
The following scenario under study is mentioned.
Total training data-set is ($N$)- 20000
Total testing data-set is - 1500
The total number of feature is ($M$)- 512
The total number of classes is ($K$)- 10
Validation Data - 20% of the total data
Testing Data - 10% of the total data
Results-2.3% for Validation and 2.9% for Training.

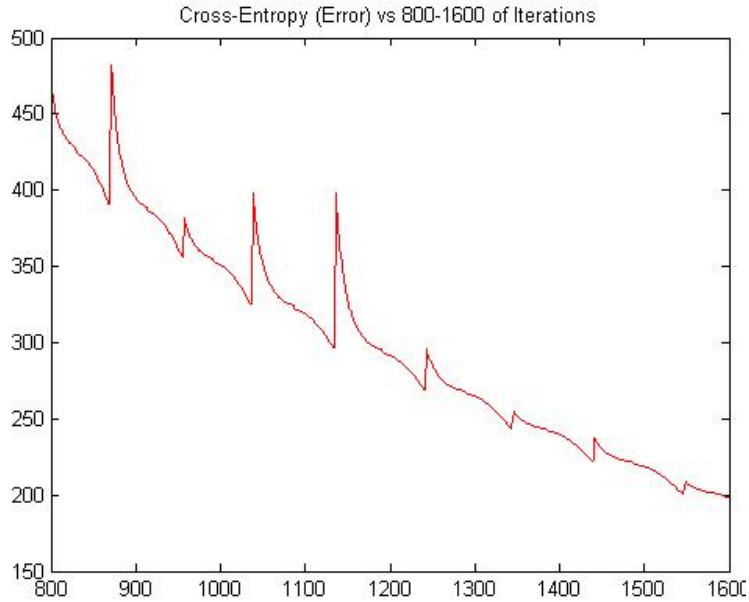
# 5 Conclusion

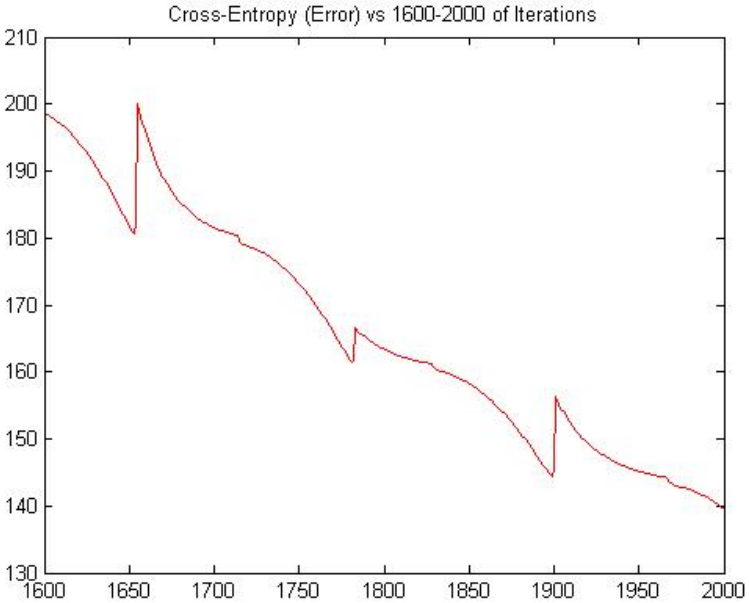
The performance is comparable when it comes to error rates. I found all the three viable to perform the operations. But the running time taken by the three is different. The nueral network toolbox was the quickets because it was already coded to provide an optimizable performance. The second best is of Neural Network which was created by me. Thus, if there is a need to go through many entries of data then we should perhaps use the Neural networks program.

# Output and Results

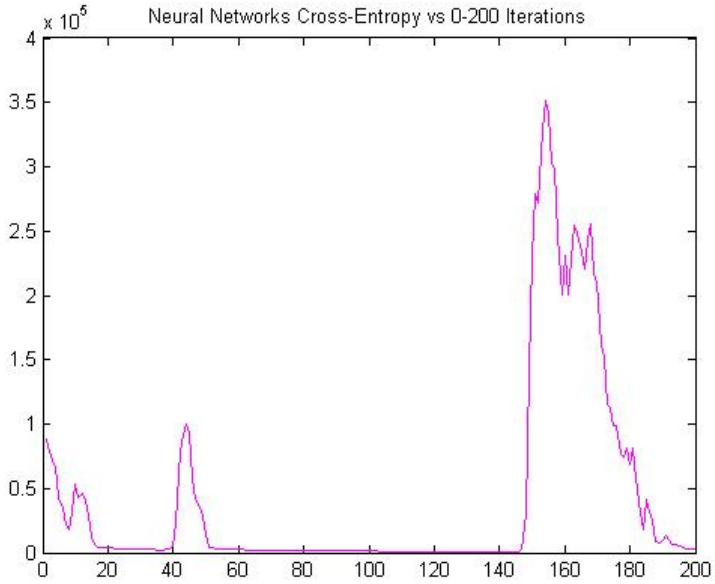


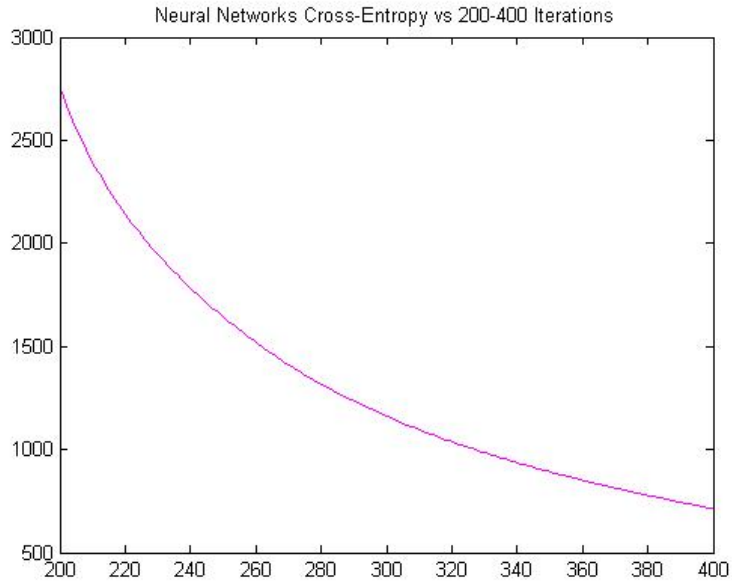



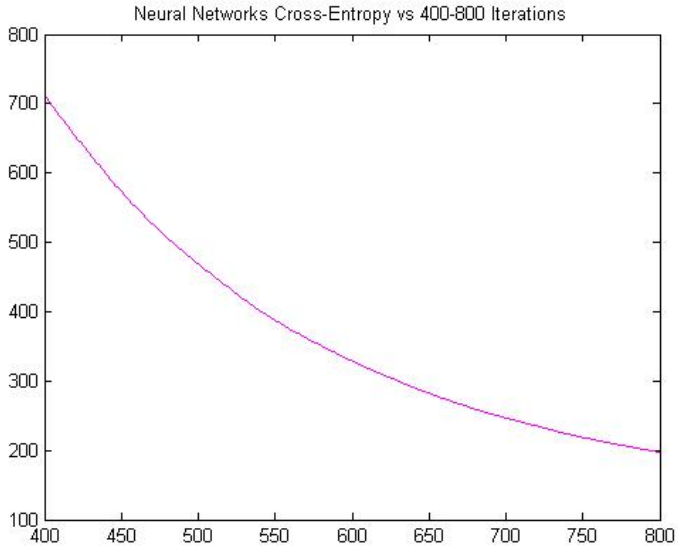



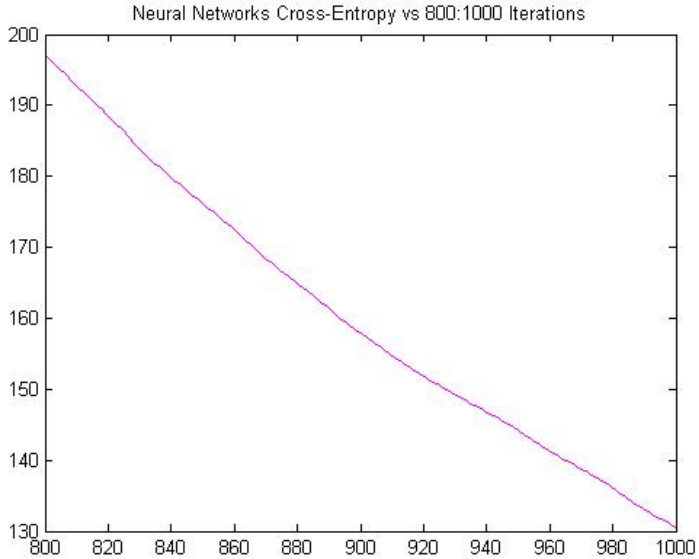



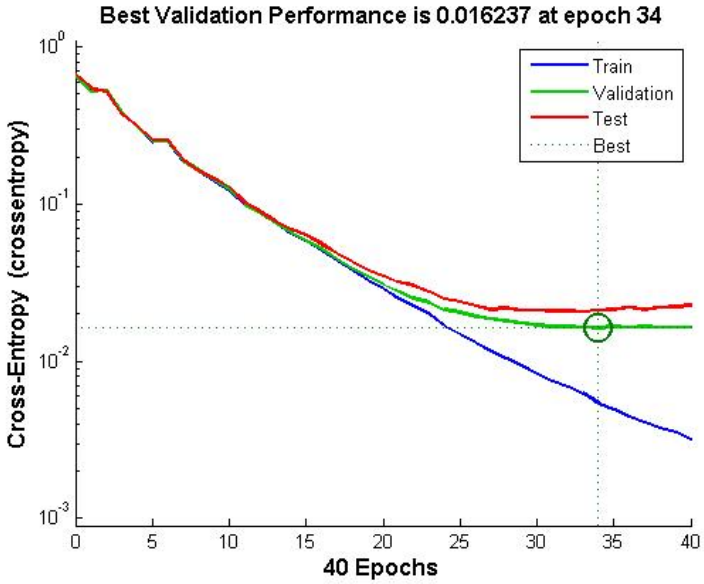
# Neural Networks Output

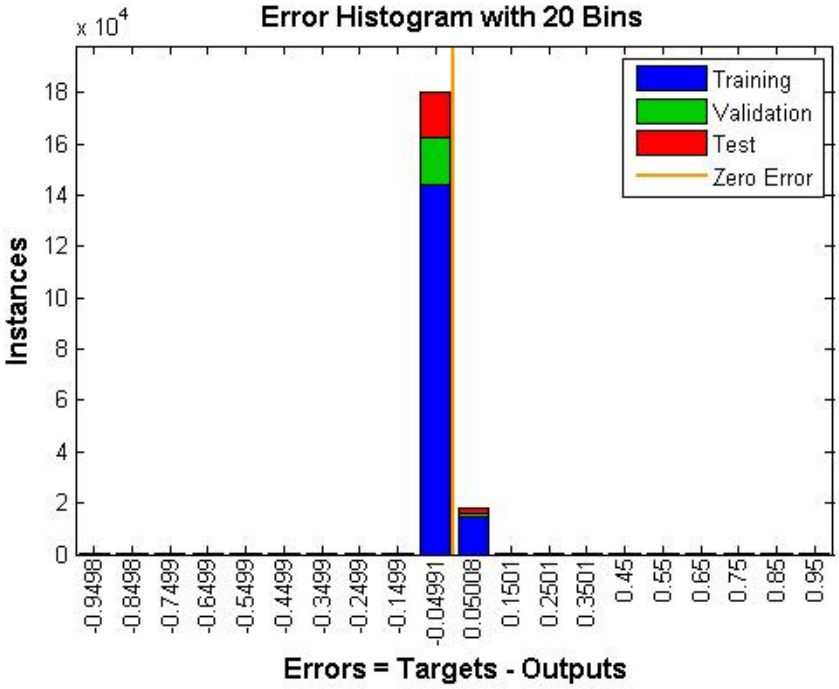



Best Validation Performance is 0.016237 at epoch 34
Train
Validation
Test
Best
Cross-Entropy (crossentropy)
40 Epochs


Error Histogram with 20 Bins
x 10^4
Training
Validation
Test
Zero Error
Instances
Errors = Targets - Outputs