# Machine Learning (CSE 574)

## Project 1

### Kaustubh Vartak

### 50098072

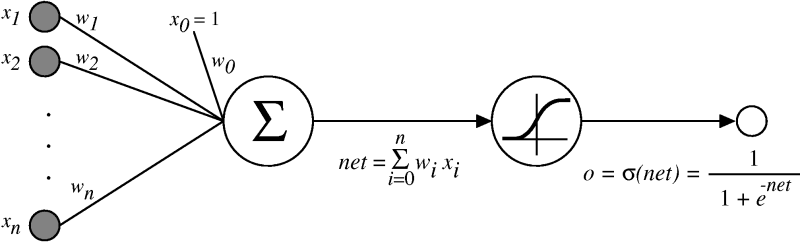
# Understanding of classifiers

Dataset:

The dataset consists of images of digits 0-9 and are 28x28 pixels wide. There are 60000 such images in the training set an 10000 more in the testing set. Hence, the matrices are 60000x784 and 10000x784 respectively. 10000 datasets from training is used for validation. Also the columns are reduced by removing the pixels which are redundant throughout the dataset to reduce the computation. The input pixel values are intensities ranging from 0-255. We normalize them to 0-1.

Neural Network classifier

Feed forward propagation: The network is established by choosing the no. of input nodes same as the no. of pixels in the image. Using a hidden layer with arbitrary no. of nodes (here 50) the first weight matrix is created with random weights. Thus each node in the hidden layer is a sigmoid unit with 28 x 28 = 784 +1 inputs as shown below.



n = 784

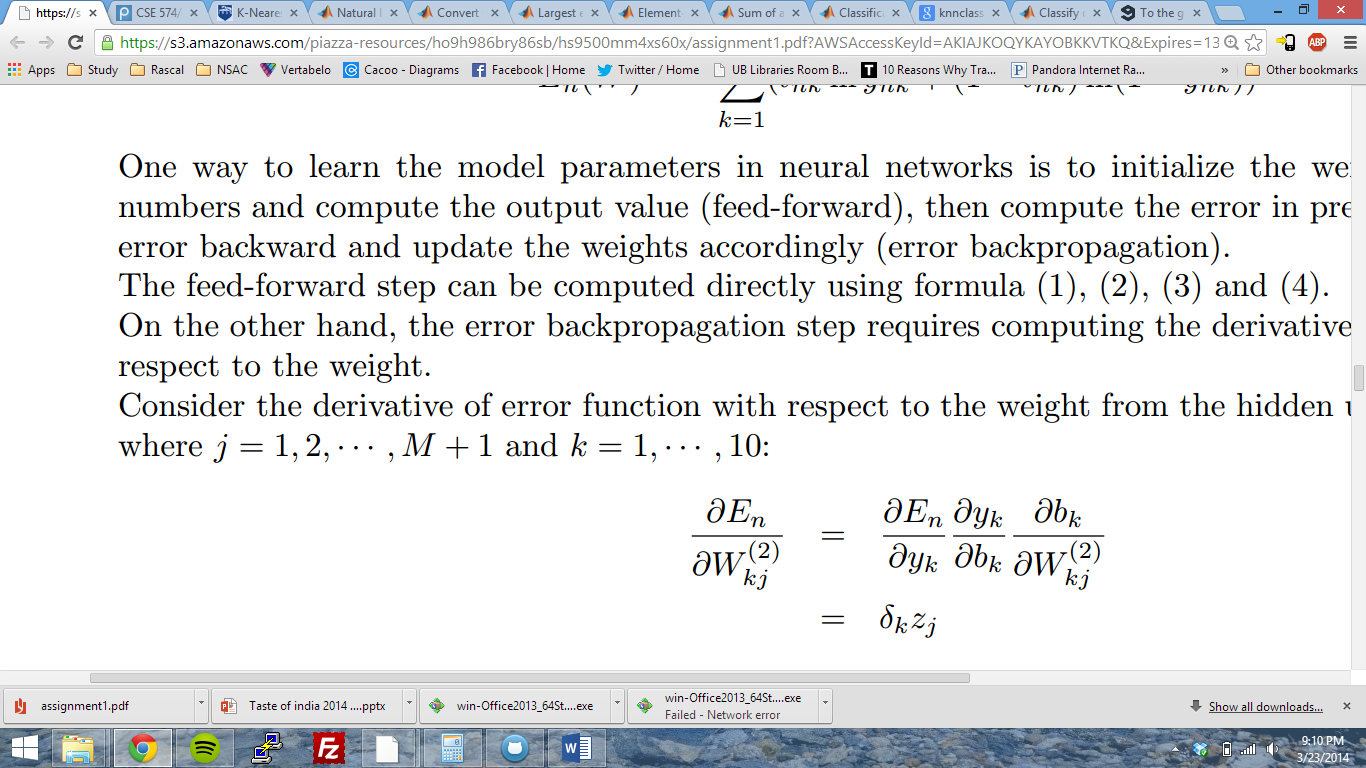
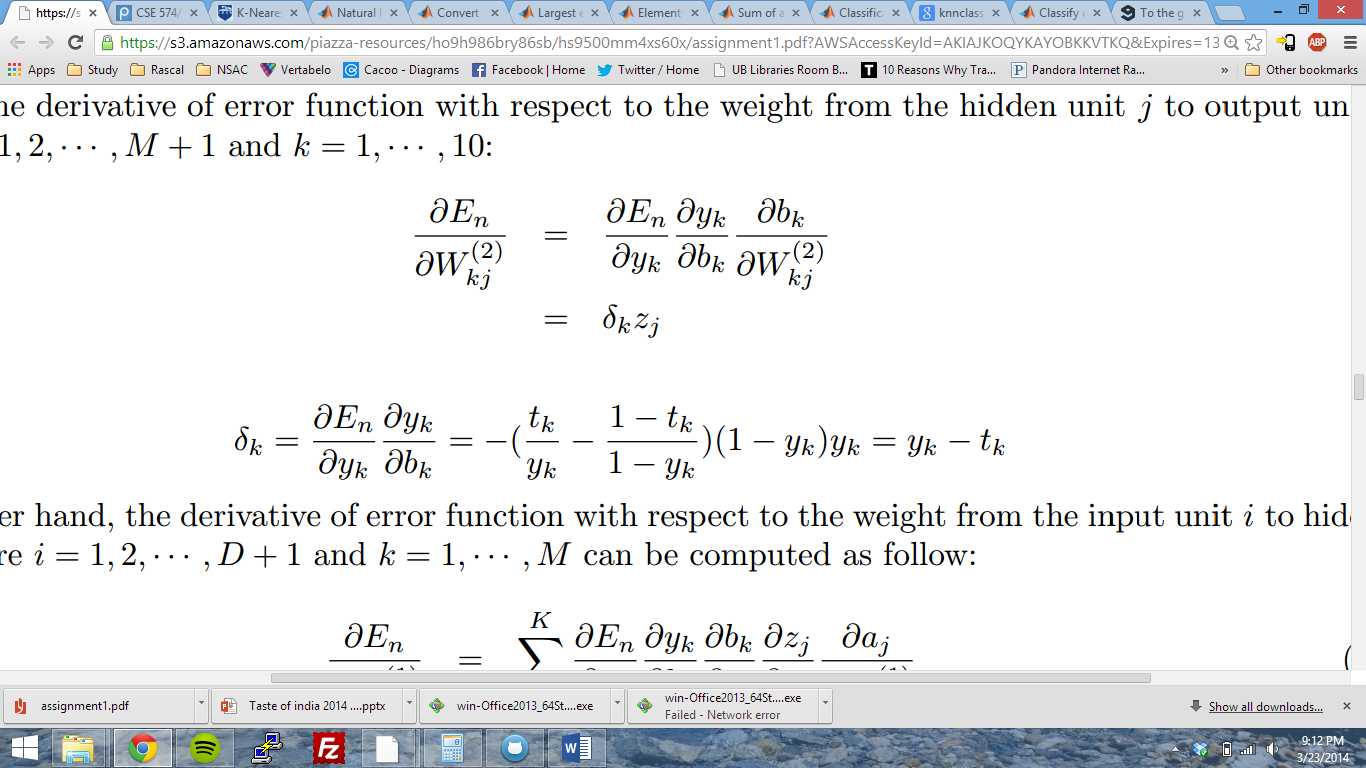
A second layer exists from hidden to output nodes with a corresponding weight matrix. Since we have 10 integers (0-9) to be classified we have 10 output nodes.

We apply the feed forward propagation using the equation shown in the figure above and training data at the two layers and using training labels get the error for each pixel at each training input. Thus the error is a 10x50000 matrix which is used in back propagation.

Back propagation:

Since we have 10 labels we convert the training label vector into a matrix using 1-of-K coding scheme wherein the kth row in the matrix for ith is 1 if the ith row in the training label is k else its 0.

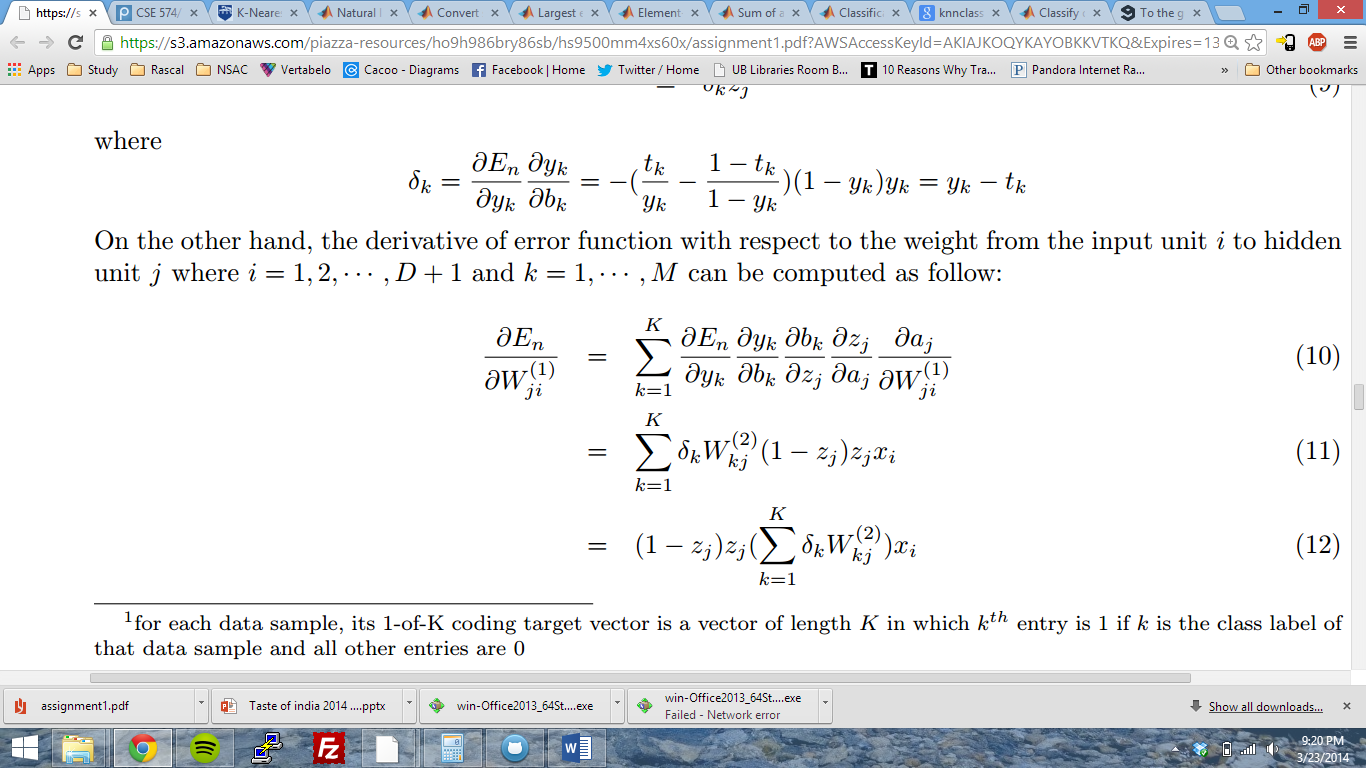
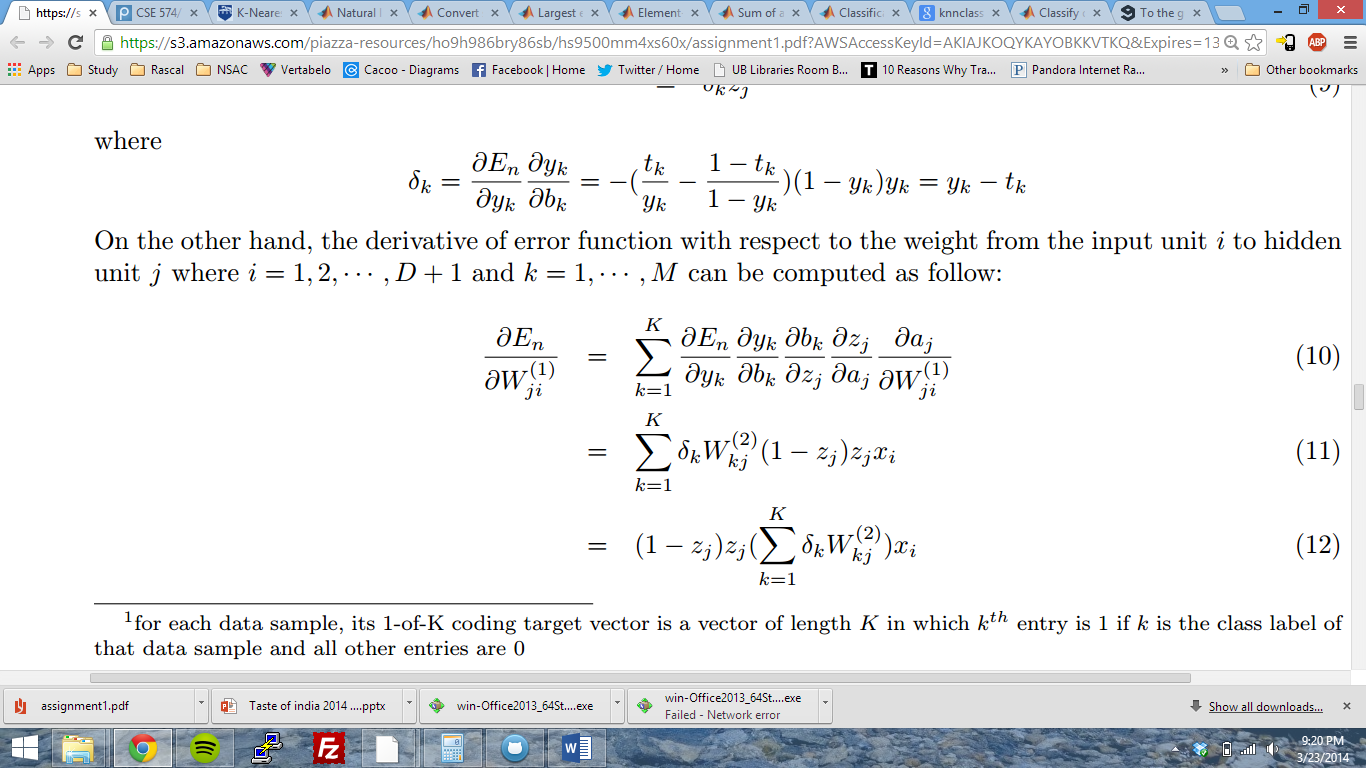
We calculate the change in output with respect to the second weight matrix using the equation



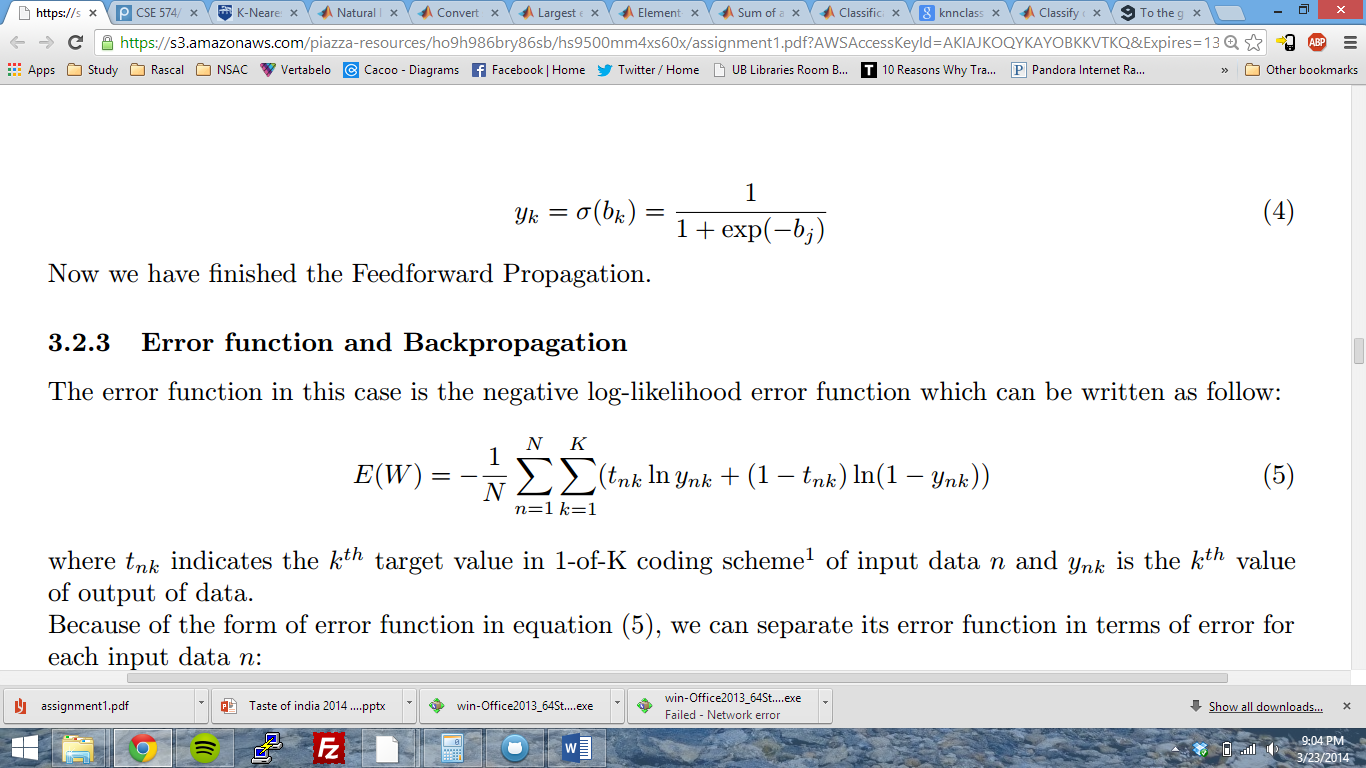
where

Here y­k and tk are the output and expected probabilities of the network for each class k for one training input.

Similarly, the change in output with respect to first matrix is given by

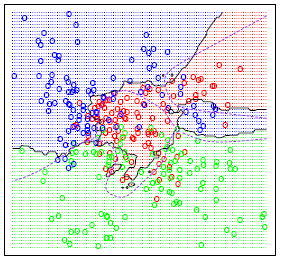


We calculate the error function of the network with the equation



­Using this we get the gradient of error function. The nnObjFunction returns the derivative matrices and the gradient to fmincg method which performs 50 iterations to calculate the conjugate gradient descent. If implemented correctly the gradient should keep reducing till a point of convergence is reached or the iterations are completed. Thus the value of gradient is low.

## K-Nearest Neighbor classification

The k-nearest neighbor plots the input training data in the hyperspace. The no. of dimensions in this hyperspace is same as the no. of inputs. Thus each input is plotted and a region of a class is decided from the training labels.

The output labels for test data is determined by its position in the hyperspace and the region of the class it belongs to.

If the test data point is on the boundary of two regions, a tie breaker rule has to be applied to

How to choose the hyper-parameters

The regularization term (lambda) improves the accuracy till a certain point after which the accuracy drops back.

The k parameter keeps improvising the accuracy of the classification however it does have a significant impact on the performance of the algorithm. Beyond a value of 4, k was too expensive to run.

# Comparison the performance

The neural network is computationally less expensive than K-nearest neighbor. This is because we calculate the conjugate gradient descent which uses matrix multiplication. On the other hand, for K-nearest neighbor, we calculate the Euclidean distances with all the points to get the k nearest points which is O(n2).

However, the accuracy of a neural network relies on effective learning of weights and their convergence. The convergence is limited by the no. of iterations and is never perfect. Also, weights can converge wrongly if input of two different classes are similar. However, KNN completely relies on the k parameter i.e. the number of nearest neighbors to be used for classification. The tie breaker rule has less impact when the training data is large (which in our case is). Hence, accuracy of KNN is higher than neural network.

# Advantages and disadvantages of each method

Neural Network:

Advantages:

1. We need not describe the problem or implement a solution method for learning. No algorithms need to be implemented. Only a large dataset of training examples is necessary.
2. Can be implemented in the hardware.
3. Can be implemented on a distributed computing architecture.

### Disadvantages:

1. A huge training set is
2. The VC dimension of neural networks is unclear.
3. Due to the complex structure of weight matrices the internal working is never clear.
4. An error is inevitable in a neural network.

## K-Nearest Neighbor:

### Advantages:

1. Very simple to understand and easy to implement.
2. The process is transparent and not unclear like neural network.
3. It is easy to debug since the result and method is available in real world

### Disadvantages:

1. Since all the learning is done at run-time, k-NN can have poor performance for large training dataset.
2. It is sensitive to redundant or irrelevant features since they generate points which can contribute to similarity. These can be avoided by careful feature selection.

# References:

1. Wikipedia.org
2. “k-Nearest Neighbour Classiﬁers” - P´adraig Cunningham and Sarah Jane Delany