International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 1, January 2013 ISSN 2319 - 4847

Neural Network Classifier for Isolated Character Recognition

[1]Ruby Mehta, [2]Ravneet Kaur

[1]M.Tech (CSE), Guru Nanak Dev University, Amritsar (Punjab), India

[2]M.Tech Scholar, Computer Science & Engineering (CSE), Lovely Professional University, Punjab, India

ABSTRACT

The explosion of computer power over the last few decades has made way for the development of intelligent systems that once existed only in the domain of science fiction novels. Amongst the vast array of topics in intelligent systems is that of computer vision and comprehension and a particularly important area of computer vision is that of character recognition. Character recognition allows computers go beyond matrices of ones and zeros by gleaning useful information from scanned images of text. Character recognition systems have enormous potential to solve real problems in a wide range of practical applications such as mail sorting, data compression, and many more.

A variety of tools are available for character recognition. OCR tools are basically of two types: Online tools and Offline tools. Character recognition is generally attempted by many methods. One of them is by neural networks. The general idea behind neural networks is that they use a large set of training data to slowly 'learn' what makes a one class different from another. Neural networks require large amounts of data to accurately learn how to classify one class from another. In this study, an algorithm has been proposed for recognizing isolated characters using backpropagation model of Neural Network.

Keywords: neural network, OCR, OCR tools, Segmentation, back propagation.

1. INTRODUCTION

Character recognition is a process, which associates a symbolic meaning with objects (letters, symbols and numbers) drawn on an image, *i.e.*, character recognition techniques associate a symbolic identity with the image of a character. Mainly, character recognition machine takes the raw data that further implements the process of preprocessing of any recognition system. On the basis of that data acquisition process, character recognition system can be classified into following categories: -

- Online Character Recognition
- Offline Character Recognition

In character recognition, the process starts with reading of a scanned image of a series of characters, determines their meaning, and finally translates the image to a computer written text document.

Many researches have been done on character recognition in last 56 years. Some books and many surveys have been published on the character recognition. Most of the work on character recognition has been done on Japanese, Latin, Chinese characters in the middle of 1960s. In this paper, RBF neural network is used for isolated character recognition. In first section we discuss neural network, in second section neural network classifier in isolated OCR and in next section we proposed an algorithm and at last experimental work is given.

2. AN ARTIFICIAL NEURAL NETWORK (ANN)

An Artificial Neural is information processing paradigm that is designed to model the way in which the brain performs a particular task or function of interest. The network is usually implemented by using electronic components or is simulated in software on a digital computer. A neural network is a massively parallel distributed processor made up of simple processing units which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1) Knowledge is required by the network from its environment through a learning process

2) Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge" [10]

A neural network has its neurons divided into subgroups of fields and elements in each subgroup are placed in a row or a column.

A neural network model is as:

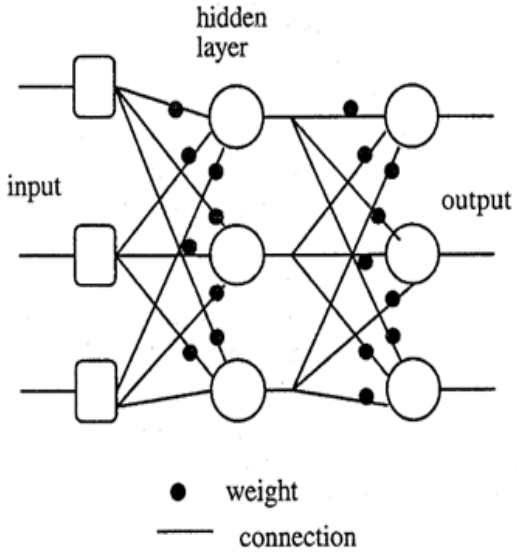


Fig. 2.1: Neural Network model

2.1 Neural network Classifier in isolated OCR

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, are an excellent solution to the classification stage of OCR. They can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. The OCR process is complicated by noisy inputs, image distortion, and differences between typefaces, sizes, and fonts. Artificial neural networks are commonly used to perform character recognition due to their high noise tolerance.

A popular and simple NN approach to the OCR problem is based on feed forward neural networks with backpropagation learning. Basic procedure involves following steps: Training and Testing.

i)Training: Prepare a training set and train network to recognize patterns from the training set. Here, we teach the network to respond with the desired output for a specified input. For this purpose, each training sample is represented by two components: possible input and the desired network are output given that input.

ii) Testing: After the training step is done, we can give an arbitrary input to the network and the network will form an output, from which we can resolve a pattern type presented to the network.

Here we are dealing with the recognition of the segmented or isolated characters. The procedure involved in the classification stage of Segmented Character Recognition using Neural Network is given.

3. PROPOSED ALGORITHM

Let I be the number of units in the input layer, as determined by the length of the training input vectors. Let O be the number of units in the output layer. Now choose H, the number of units in the hidden layer. The input layer has an extra unit used for thresholding.

The neurons or the activation levels of the units in all layers of the network are denoted by u_i, where (i = 0 to I+H+O).. Weights are denoted by $w_{i,j}$, where the subscript i indexes the input units and j indexes the hiddenunits.(if input-hidden layers are being considered) and i indexes hidden units and j indexes output units(if hidden-output layers are being considered).

The neural network classifier does the classification task in two steps: Training and Testing.

3.1 Training

Training process consists of following steps:

1. Selection of Training Data: The training data set contains the numerals 0 to 9. The set of training data containing the input vectors of size 15 and output vector of size 10 for numerals 0 to 9 are given in following table:

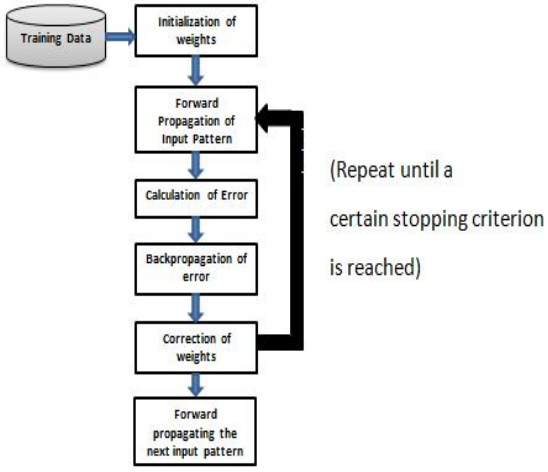


Fig. 3.1: Training procedure

2. Initialization of Weights of network: Initialize the weights in the network. Each should be set randomly to a number between -0.1 and 0.1.
Wij = random (-0.1, 0.1) for all i =0 to I, j =I + 1 to H
Wij = random (-0.1, 0.1) for all i =I+1 to H, j= H+1 to O

Table 3.1: Input and Output Vectors of system

Numeral	Input vector of size 15	Output vector of size 10
0	010101101101010	1000000000
1	010010010010010	0100000000
2	110001010100111	0010000000
3	110001010001110	0001000000
4	100101111001001	0000100000
5	111100110001110	0000010000
6	011100110101010	0000001000
7	111001001001001	0000000100
8	010101010101010	0000000010
9	010101010001110	0000000001

3. Forward propagation of input pattern: Forward propagation of the first input pattern of the training set from the input layer over the hidden layer(s) to the output layer, where each neuron sums the weighted inputs, passes them through the nonlinearity and passes this weighted sum to the neurons in the next layer. It is done by choosing the first input-output pair and propagating the activations from the units in the input layer to the units in the hidden layer and then from units in the hidden layer to the units in the output layer. A bottom-up pass through the network is made to compute

weighted sum, $S_i = \sum w_{i,j}u_j$ ***, and***

activations $u_i = f(S_i)$

$f(S_i) = 1/(1+e^{-Si})$ ***is the sigmoid function.***

Here, i= 0 to I+H+O, (when i= I+1 to I+ H, then j=0 to I,
And when, i= H+1 to I+H+O, then j= I+1 to I+H)

4. Error: If there is difference between the actual output of each output neuron and its corresponding desired output, then it means there is an error and backpropagation step is required.

5. Backward-propagation Step: Back propagating through each connection by using the Back propagation learning rule and thus determining the amount each weight has to be changed in order to decrease the error at the output layer. Starting with the outputs, make a top-down pass through the output and intermediate cells computing:

$$f`(S_i) = u_i(1 - u_i)$$

$$\delta_t = (C_i - u_i)\, f`(S_i) \text{ if } u_i \text{ is an output unit}$$

$$(\sum w_{mi}\, \delta_m)\, f`(S_i)\ ,\ m{:}m>i \quad \text{for other units}$$

6. Correction of weights: Update each network weights $w_{i,j}$ as $W_{i,j}$

$$W_{i,j} = w_{i,j} + \mu\, \delta_t\, u_j$$

μ is the learning rate and it can have arbitrary small positive value. Here μ= 0.1 is taken. Weights between hidden-output layer or input-hidden layers are adjusted

i= 0 to I+H+O, (when i= I+1 to I+ H, then j=0 to I,

And when, i= H+1 to I+H+O, then j= I+1 to I+H)

Presenting and forward propagating the same input pattern again. Repeat steps 3-6 until a certain stopping criterion is reached, for example that the error falls below a predefined value.

7. Forward propagating next input pattern: Next input patterns are presented to the network. Again the same steps from 3-6 are repeated for all the input patterns. The one-time presentation of the entire set of training patterns to the net constitutes a training epoch.

3.2 TESTING

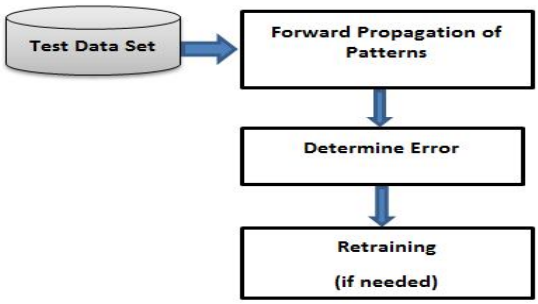


Fig. 4.17: Testing procedure

After terminating the training phase the trained net is tested with patterns from the test data set.

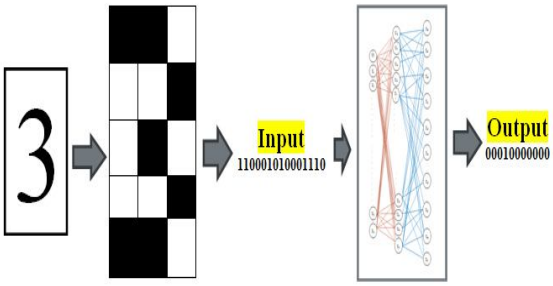
1. Forward Propagation of Patterns: The patterns are forward propagated, using the weights now available from training.

2. Determine Error: The error at the output layer is determined (no weight-update is performed!).

3. Retraining: If performance is sufficiently good, the net is ready for- use. If not, it has to be retrained with the same patterns and parameters or something has to be changed (e.g. number of hidden neurons, additional input patterns, different kinds of information contained in the input patterns).

4. EXPERIMENT WORK

Procedure in numeral '3' recognition is as: Activations of neurons u_1 to u_{15} : The input vector corresponding to '3' is '110001010001110' which is given to the network and acts as activation for the input layer neurons u_1 to u_{15}.



Thus we have:

Neuron i	Activation u_i
1	1
2	1
3	0
4	0
5	0
6	1
7	0
8	1
9	0
10	0
11	0
12	1
13	1
14	1
15	0

Table a. (conti.): Activations of neurons u_1 to u_{15}ndomly Initialized weights between input layer and hidden layer:

$w_{16,1}$	0.2	$w_{17,1}$	0.3	$w_{18,1}$	0.4	$w_{19,1}$	0.3	$w_{20,1}$	0.1	$w_{21,1}$	0.2	$w_{22,1}$	0.5	$w_{23,1}$	0.3
$w_{16,2}$	0.4	$w_{17,2}$	0.5	$w_{18,2}$	0.5	$w_{19,2}$	0.5	$w_{20,2}$	0.5	$w_{21,2}$	0.4	$w_{22,2}$	0.5	$w_{23,2}$	0.5
$w_{16,3}$	0.1	$w_{17,3}$	0.2	$w_{18,3}$	0.1	$w_{19,3}$	0.2	$w_{20,3}$	0.1	$w_{21,3}$	0.1	$w_{22,3}$	0.1	$w_{23,3}$	0.2
$w_{16,4}$	0.3	$w_{17,4}$	0.1	$w_{18,4}$	0.2	$w_{19,4}$	0.1	$w_{20,4}$	0.2	$w_{21,4}$	0.3	$w_{22,4}$	0.2	$w_{23,4}$	0.1
$w_{16,5}$	0.2	$w_{17,5}$	0.1	$w_{18,5}$	0.5	$w_{19,5}$	0.1	$w_{20,5}$	0.5	$w_{21,5}$	0.2	$w_{22,5}$	0.5	$w_{23,5}$	0.1
$w_{16,6}$	0.1	$w_{17,6}$	0.4	$w_{18,6}$	0.3	$w_{19,6}$	0.4	$w_{20,6}$	0.3	$w_{21,6}$	0.1	$w_{22,6}$	0.3	$w_{23,6}$	0.4
$w_{16,7}$	0.4	$w_{17,7}$	0.3	$w_{18,7}$	0.1	$w_{19,7}$	0.3	$w_{20,7}$	0.1	$w_{21,7}$	0.4	$w_{22,7}$	0.1	$w_{23,7}$	0.3
$w_{16,8}$	0.3	$w_{17,8}$	0.2	$w_{18,8}$	0.4	$w_{19,8}$	0.2	$w_{20,8}$	0.4	$w_{21,8}$	0.3	$w_{22,8}$	0.4	$w_{23,8}$	0.2

Table b: Initials weights between the input layer neurons and hidden layer neurons

$w_{16,9}$	0.1	$w_{17,9}$	0.1	$w_{18,9}$	0.5	$w_{19,9}$	0.1	$w_{20,9}$	0.5	$w_{21,9}$	0.1	$w_{22,9}$	0.5	$w_{23,9}$	0.1
$w_{16,10}$	0.4	$w_{17,10}$	0.5	$w_{18,10}$	0.3	$w_{19,10}$	0.5	$w_{20,10}$	0.3	$w_{21,10}$	0.4	$w_{22,10}$	0.3	$w_{23,10}$	0.5
$w_{16,11}$	0.3	$w_{17,11}$	0.4	$w_{18,11}$	0.1	$w_{19,11}$	0.4	$w_{20,11}$	0.1	$w_{21,11}$	0.3	$w_{22,11}$	0.1	$w_{23,11}$	0.4
$w_{16,12}$	0.5	$w_{17,12}$	0.3	$w_{18,12}$	0.2	$w_{19,12}$	0.3	$w_{20,12}$	0.2	$w_{21,12}$	0.5	$w_{22,12}$	0.2	$w_{23,12}$	0.3
$w_{16,13}$	0.2	$w_{17,13}$	0.2	$w_{18,13}$	0.3	$w_{19,13}$	0.2	$w_{20,13}$	0.3	$w_{21,13}$	0.2	$w_{22,13}$	0.3	$w_{23,13}$	0.2
$w_{16,14}$	0.2	$w_{17,14}$	0.1	$w_{18,14}$	0.4	$w_{19,14}$	0.1	$w_{20,14}$	0.4	$w_{21,14}$	0.2	$w_{22,14}$	0.4	$w_{23,14}$	0.1
$w_{16,15}$	0.1	$w_{17,15}$	0.3	$w_{18,15}$	0.5	$w_{19,15}$	0.3	$w_{20,15}$	0.5	$w_{21,15}$	0.1	$w_{22,15}$	0.5	$w_{23,15}$	0.3

Table c(conti.): Initials weights between the input layer neurons and hidden layer neurons

$w_{24,1}$	0.1	$w_{25,1}$	0.3	$w_{26,1}$	0.4	$w_{27,1}$	0.1	$w_{28,1}$	0.3	$w_{29,1}$	0.2	$w_{30,1}$	0.2	$w_{31,1}$	0.3
$w_{24,2}$	0.5	$w_{25,2}$	0.5	$w_{26,2}$	0.5	$w_{27,2}$	0.5	$w_{28,2}$	0.5	$w_{29,2}$	0.4	$w_{30,2}$	0.4	$w_{31,2}$	0.5
$w_{24,3}$	0.1	$w_{25,3}$	0.2	$w_{26,3}$	0.1	$w_{27,3}$	0.1	$w_{28,3}$	0.2	$w_{29,3}$	0.1	$w_{30,3}$	0.1	$w_{31,3}$	0.2
$w_{24,4}$	0.2	$w_{25,4}$	0.1	$w_{26,4}$	0.2	$w_{27,4}$	0.2	$w_{28,4}$	0.1	$w_{29,4}$	0.3	$w_{30,4}$	0.3	$w_{31,4}$	0.1
$w_{24,5}$	0.5	$w_{25,5}$	0.1	$w_{26,5}$	0.5	$w_{27,5}$	0.5	$w_{28,5}$	0.1	$w_{29,5}$	0.2	$w_{30,5}$	0.2	$w_{31,5}$	0.1
$w_{24,6}$	0.3	$w_{25,6}$	0.4	$w_{26,6}$	0.3	$w_{27,6}$	0.3	$w_{28,6}$	0.4	$w_{29,6}$	0.1	$w_{30,6}$	0.1	$w_{31,6}$	0.4
$w_{24,7}$	0.1	$w_{25,7}$	0.3	$w_{26,7}$	0.1	$w_{27,7}$	0.1	$w_{28,7}$	0.3	$w_{29,7}$	0.4	$w_{30,7}$	0.4	$w_{31,7}$	0.3

Table d(conti.): Initials weights between the input layer neurons and hidden layer neurons

$W_{24,8}$	0.4	$W_{25,8}$	0.2	$W_{26,8}$	0.4	$W_{27,8}$	0.4	$W_{28,8}$	0.2	$W_{29,8}$	0.3	$W_{30,8}$	0.3	$W_{31,8}$	0.2
$W_{24,9}$	0.5	$W_{25,9}$	0.1	$W_{26,9}$	0.5	$W_{27,9}$	0.5	$W_{28,9}$	0.1	$W_{29,9}$	0.1	$W_{30,9}$	0.1	$W_{31,9}$	0.1
$W_{24,10}$	0.3	$W_{25,10}$	0.5	$W_{26,10}$	0.3	$W_{27,10}$	0.3	$W_{28,10}$	0.5	$W_{29,10}$	0.4	$W_{30,10}$	0.4	$W_{31,10}$	0.5
$W_{24,11}$	0.1	$W_{25,11}$	0.4	$W_{26,11}$	0.1	$W_{27,11}$	0.1	$W_{28,11}$	0.4	$W_{29,11}$	0.3	$W_{30,11}$	0.3	$W_{31,11}$	0.4
$W_{24,12}$	0.2	$W_{25,12}$	0.3	$W_{26,12}$	0.2	$W_{27,12}$	0.2	$W_{28,12}$	0.3	$W_{29,12}$	0.5	$W_{30,12}$	0.5	$W_{31,12}$	0.3
$W_{24,13}$	0.3	$W_{25,13}$	0.2	$W_{26,13}$	0.3	$W_{27,13}$	0.3	$W_{28,13}$	0.2	$W_{29,13}$	0.2	$W_{30,13}$	0.2	$W_{31,13}$	0.2
$W_{24,14}$	0.4	$W_{25,14}$	0.1	$W_{26,14}$	0.4	$W_{27,14}$	0.4	$W_{28,14}$	0.1	$W_{29,14}$	0.2	$W_{30,14}$	0.2	$W_{31,14}$	0.1
$W_{24,15}$	0.5	$W_{25,15}$	0.3	$W_{26,15}$	0.5	$W_{27,15}$	0.5	$W_{28,15}$	0.3	$W_{29,15}$	0.1	$W_{30,15}$	0.1	$W_{31,15}$	0.3

Table:e (conti.): Initials weights between the input layer neurons and hidden layer neurons

$W_{32,1}$	0.3	$W_{33,1}$	0.2	$W_{34,1}$	0.5	$W_{35,1}$	0.4	$W_{36,1}$	0.3	$W_{37,1}$	0.4
$W_{32,2}$	0.5	$W_{33,2}$	0.4	$W_{34,2}$	0.5	$W_{35,2}$	0.5	$W_{36,2}$	0.5	$W_{37,2}$	0.5
$W_{32,3}$	0.2	$W_{33,3}$	0.1	$W_{34,3}$	0.1	$W_{35,3}$	0.1	$W_{36,3}$	0.2	$W_{37,3}$	0.1
$W_{32,4}$	0.1	$W_{33,4}$	0.3	$W_{34,4}$	0.2	$W_{35,4}$	0.2	$W_{36,4}$	0.1	$W_{37,4}$	-0.2
$W_{32,5}$	0.1	$W_{33,5}$	0.2	$W_{34,5}$	0.5	$W_{35,5}$	0.5	$W_{36,5}$	0.1	$W_{37,5}$	0.5
$W_{32,6}$	0.4	$W_{33,6}$	0.1	$W_{34,6}$	0.3	$W_{35,6}$	0.3	$W_{36,6}$	0.4	$W_{37,6}$	0.3
$W_{32,7}$	0.3	$W_{33,7}$	0.4	$W_{34,7}$	0.1	$W_{35,7}$	0.1	$W_{36,7}$	0.3	$W_{37,7}$	0.1
$W_{32,8}$	0.2	$W_{33,8}$	0.3	$W_{34,8}$	0.4	$W_{35,8}$	0.4	$W_{36,8}$	0.2	$W_{37,8}$	-0.4

Table:f (conti.): Initials weights between the input layer neurons and hidden layer neurons

$W_{32,9}$	0.1	$W_{33,9}$	0.1	$W_{34,9}$	0.5	$W_{35,9}$	0.5	$W_{36,9}$	0.1	$W_{37,9}$	-0.5
$W_{32,10}$	0.5	$W_{33,10}$	0.4	$W_{34,10}$	0.3	$W_{35,10}$	0.3	$W_{36,10}$	0.5	$W_{37,10}$	-0.3
$W_{32,11}$	0.4	$W_{33,11}$	0.3	$W_{34,11}$	0.1	$W_{35,11}$	0.1	$W_{36,11}$	0.4	$W_{37,11}$	0.1
$W_{32,12}$	0.3	$W_{33,12}$	0.5	$W_{34,12}$	0.2	$W_{35,12}$	0.2	$W_{36,12}$	0.3	$W_{37,12}$	0.2
$W_{32,13}$	0.2	$W_{33,13}$	0.2	$W_{34,13}$	0.3	$W_{35,13}$	0.3	$W_{36,13}$	0.2	$W_{37,13}$	0.3
$W_{32,14}$	0.1	$W_{33,14}$	0.2	$W_{34,14}$	0.4	$W_{35,14}$	0.4	$W_{36,14}$	0.1	$W_{37,14}$	0.4
$W_{32,15}$	0.3	$W_{33,15}$	0.1	$W_{34,15}$	0.2	$W_{35,15}$	0.3	$W_{36,15}$	0.5	$W_{37,15}$	0.4

Table: g (conti.): Initials weights between the input layer neurons and hidden layer neurons Randomly Initialized weights between hidden layer and output layer:

$W_{38,16}$	0.4	$W_{39,16}$	0.5	$W_{40,16}$	0.3	$W_{41,16}$	0.5	$W_{42,16}$	0.2	$W_{43,16}$	0.5	$W_{44,16}$	0.2	$W_{45,16}$	0.2
$W_{38,17}$	0.5	$W_{39,17}$	0.5	$W_{40,17}$	0.5	$W_{41,17}$	0.5	$W_{42,17}$	0.4	$W_{43,17}$	0.5	$W_{44,17}$	0.4	$W_{45,17}$	0.4
$W_{38,18}$	0.1	$W_{39,18}$	0.1	$W_{40,18}$	0.2	$W_{41,18}$	0.1	$W_{42,18}$	0.1	$W_{43,18}$	0.1	$W_{44,18}$	0.1	$W_{45,18}$	0.1
$W_{38,19}$	0.2	$W_{39,19}$	0.2	$W_{40,19}$	0.1	$W_{41,19}$	0.2	$W_{42,19}$	0.3	$W_{43,19}$	0.2	$W_{44,19}$	0.3	$W_{45,19}$	0.3
$W_{38,20}$	0.5	$W_{39,20}$	0.5	$W_{40,20}$	0.1	$W_{41,20}$	0.5	$W_{42,20}$	0.2	$W_{43,20}$	0.5	$W_{44,20}$	0.2	$W_{45,20}$	0.2
$W_{38,21}$	0.3	$W_{39,21}$	0.3	$W_{40,21}$	0.4	$W_{41,21}$	0.3	$W_{42,21}$	0.1	$W_{43,21}$	0.3	$W_{44,21}$	0.1	$W_{45,21}$	0.1
$W_{38,22}$	0.1	$W_{39,22}$	0.1	$W_{40,22}$	0.3	$W_{41,22}$	0.1	$W_{42,22}$	0.4	$W_{43,22}$	0.1	$W_{44,22}$	0.4	$W_{45,22}$	0.4
$W_{38,23}$	0.4	$W_{39,23}$	0.4	$W_{40,23}$	0.2	$W_{41,23}$	0.4	$W_{42,23}$	0.3	$W_{43,23}$	0.4	$W_{44,23}$	0.3	$W_{45,23}$	0.3
$W_{38,24}$	0.5	$W_{39,24}$	0.5	$W_{40,24}$	0.1	$W_{41,24}$	0.5	$W_{42,24}$	0.1	$W_{43,24}$	0.5	$W_{44,24}$	0.1	$W_{45,24}$	0.1
$W_{38,25}$	0.3	$W_{39,25}$	0.3	$W_{40,25}$	0.5	$W_{41,25}$	0.3	$W_{42,25}$	0.4	$W_{43,25}$	0.3	$W_{44,25}$	0.4	$W_{45,25}$	0.4
$W_{38,26}$	0.1	$W_{39,26}$	0.1	$W_{40,26}$	0.4	$W_{41,26}$	0.1	$W_{42,26}$	0.3	$W_{43,26}$	0.1	$W_{44,26}$	0.3	$W_{45,26}$	0.3
$W_{38,27}$	0.2	$W_{39,27}$	0.2	$W_{40,27}$	0.3	$W_{41,27}$	0.2	$W_{42,27}$	0.5	$W_{43,27}$	0.2	$W_{44,27}$	0.5	$W_{45,27}$	0.5
$W_{38,28}$	0.3	$W_{39,28}$	0.3	$W_{40,28}$	0.2	$W_{41,28}$	0.3	$W_{42,28}$	0.2	$W_{43,28}$	0.3	$W_{44,28}$	0.2	$W_{45,28}$	0.2
$W_{38,29}$	0.4	$W_{39,29}$	0.4	$W_{40,29}$	0.1	$W_{41,29}$	0.4	$W_{42,29}$	0.2	$W_{43,29}$	0.4	$W_{44,29}$	0.2	$W_{45,29}$	0.2

Table h: Initials weights between the hidden layer neurons and output layer neurons

$W_{38,30}$	0.5	$W_{39,30}$	0.2	$W_{40,30}$	0.5	$W_{41,30}$	0.2	$W_{42,30}$	0.1	$W_{43,30}$	0.2	$W_{44,30}$	0.1	$W_{45,30}$	0.1
$W_{38,31}$	0.4	$W_{39,31}$	0.5	$W_{40,31}$	0.3	$W_{41,31}$	0.5	$W_{42,31}$	0.2	$W_{43,31}$	0.5	$W_{44,31}$	0.2	$W_{45,31}$	0.2
$W_{38,32}$	0.5	$W_{39,32}$	0.5	$W_{40,32}$	0.5	$W_{41,32}$	0.5	$W_{42,32}$	0.4	$W_{43,32}$	0.5	$W_{44,32}$	0.4	$W_{45,32}$	0.4
$W_{38,33}$	0.1	$W_{39,33}$	0.1	$W_{40,33}$	0.2	$W_{41,33}$	0.1	$W_{42,33}$	0.1	$W_{43,33}$	0.1	$W_{44,33}$	0.1	$W_{45,33}$	0.1
$W_{38,34}$	0.2	$W_{39,34}$	0.2	$W_{40,34}$	0.1	$W_{41,34}$	0.2	$W_{42,34}$	0.3	$W_{43,34}$	0.2	$W_{44,34}$	0.3	$W_{45,34}$	0.3
$W_{38,35}$	0.5	$W_{39,35}$	0.5	$W_{40,35}$	0.1	$W_{41,35}$	0.5	$W_{42,35}$	0.2	$W_{43,35}$	0.5	$W_{44,35}$	0.2	$W_{45,35}$	0.2
$W_{38,36}$	0.3	$W_{39,36}$	0.3	$W_{40,36}$	0.4	$W_{41,36}$	0.3	$W_{42,36}$	0.1	$W_{43,36}$	0.3	$W_{3944,36}$	0.1	$W_{45,36}$	0.1
$W_{38,37}$	0.1	$W_{39,37}$	0.1	$W_{40,37}$	0.3	$W_{41,37}$	0.1	$W_{42,37}$	0.4	$W_{43,37}$	0.1	$W_{44,37}$	0.4	$W_{45,37}$	0.4

Table i: Initials weights between the hidden layer neurons and output layer neurons

$W_{46,16}$	0.4	$W_{47,16}$	0.2
$W_{46,17}$	0.5	$W_{47,17}$	0.4
$W_{46,18}$	0.1	$W_{47,18}$	0.1
$W_{46,19}$	0.2	$W_{47,19}$	0.3
$W_{46,20}$	0.5	$W_{47,20}$	0.2
$W_{46,21}$	0.3	$W_{47,21}$	0.1
$W_{46,22}$	0.1	$W_{47,22}$	0.4
$W_{46,23}$	0.4	$W_{47,23}$	0.3
$W_{46,24}$	0.5	$W_{47,24}$	0.1
$W_{46,25}$	0.3	$W_{47,25}$	0.4
$W_{46,26}$	0.1	$W_{47,26}$	0.3
$W_{46,27}$	0.2	$W_{47,27}$	0.5
$W_{46,28}$	0.3	$W_{47,28}$	0.2
$W_{46,29}$	0.4	$W_{47,29}$	0.2
$W_{46,30}$	0.5	$W_{47,30}$	0.1
$W_{46,31}$	0.4	$W_{47,31}$	0.2
$W_{46,32}$	0.5	$W_{47,32}$	0.4
$W_{46,33}$	0.1	$W_{47,33}$	0.1
$W_{46,34}$	0.2	$W_{47,34}$	0.3
$W_{46,35}$	0.5	$W_{47,35}$	0.2
$W_{46,36}$	0.3	$W_{47,36}$	0.1
$W_{46,37}$	0.1	$W_{47,37}$	0.4

Table j: Initials weights between the hidden layer neurons and output layer neurons

4.1 Forward pass

The activations of hidden layer neurons (u_{16} to u_{37}) and output layer neurons (u_{38} to u_{47}) will be calculated.
Activations of Hidden Layer neurons:

u_{16}	0.86	u_{24}	0.89	u_{32}	0.43
u_{17}	0.37	u_{25}	0.39	u_{33}	0.98
u_{18}	0.92	u_{26}	0.72	u_{34}	0.21
u_{19}	0.37	u_{27}	0.52	u_{35}	0.52
u_{20}	0.38	u_{28}	0.46	u_{36}	0.78
u_{21}	0.88	u_{29}	0.87	u_{37}	0.87
u_{22}	0.92	u_{30}	0.68		
u_{23}	0.87	u_{31}	0.79		

Table 4.1: Activations of hidden layer neurons Activations of Output Layer neurons:

u_{38}	0.99
u_{39}	0
u_{40}	0
u_{41}	078
u_{20}	0.10
u_{42}	0.66
u_{43}	0.12
u_{45}	0.10
u_{46}	0.33
u_{47}	0.22

4.2. Backward pass

Backward pass is done by starting with the outputs, make a top-down pass through the output and intermediate cells computing:

All δ_t s can be computed for each neuron of output layer. These have been computed and given as :

δ s for neurons of output layer:

δ_{38}	-0.01
δ_{39}	0
δ_{40}	0
δ_{41}	0.03
δ_{42}	0.01
δ_{43}	-0.06
δ_{44}	0.01
δ_{45}	0.01
δ_{46}	-0.03
δ_{47}	0.02

Table 13: δ s for neurons of output layer.

δs for neurons of hidden layer:

δ_{16}	0.01	δ_{24}	0.04	δ_{32}	0.03
δ_{17}	0.03	δ_{25}	0.03	δ_{33}	0.03
δ_{18}	0.02	δ_{26}	0.02	δ_{34}	0.01
δ_{19}	0.07	δ_{27}	0.05	δ_{35}	0.02
δ_{20}	0.08	δ_{28}	0.06	δ_{36}	0.09
δ_{21}	0.08	δ_{29}	0.07	δ_{37}	0.07
δ_{22}	0.02	δ_{30}	0.04		
δ_{23}	0.01	δ_{31}	0.02		

Table 4.2: δs for neurons of output layer.

Since all the δs are there. Now weights of the network can be computed by following formula:

$$W_{i,j} = w_{i,j} + \mu\ \delta_t u_j$$

μ=0.1 is the learning rate taken randomly.

Now, $W_{16,1} = w_{16,1} + \mu\delta_{16}u_1$

$= 0.2 + 0.1 * 0.01* 1$

$=0.201$

~ 0.2

Thus the updated weight

$\mathbf{W_{16,1}= 0.2}$

Similarly all the weights of the network can be updated.

Now after weights updating again the input vector (110001010001110) corresponding to '3' will be propagated forward again and the output will be calculated. If this time also the output does not come out as (00010000000) then again the input vector will be propagated forward. This process will continue until the desired output comes out. Once desired output comes out, these corrected weights will be saved for this particular input pattern and then the next input pattern will be given to network.

5. CONCLUSION AND FURTHER SCOPE

An Algorithm has been proposed for a neural network classifier for isolated character recognition. The brief introduction about optical character recognition process has been provided. An algorithm for neural network classifier for isolated character recognition has been given.

- The neural network classifier given can be implemented in programming languages like C++, C#, Matlab etc.
- More algorithms can be devised which can be implemented to form a character recognition system having human eye like efficiency.
- Various other parameters can also be used for comparative study of OCR tools.

Various other services and tools can be explored and compared.

REFERENCES

[1]Mani and Srinivasan, "Application of Artificial Neural Network Model for Optical", Character Recognition", IEEE: 1997.

[2] Arica and Yarman-Vural, "Optical Character Recognition for Cursive Handwriting", IEEE.

[3] Srinivasa Kumar Devireddy, SettipalliAppaRao, "Hand Written Character Recognition using Back Propagation Network", Journal of Theoretical and Applied Information Technology, (2009) JATIT

[4] Sang Sung Park, Won Gyo Jung, Young Geun Shin, Dong-Sik Jang, "Optical Character Recognition System Using BP Algorithm", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.12, December 2008.

[5]Rinki Singh & Mandeep Kaur, OCR for Telugu Script Using Back-Propagation Based Classifier, International Journal of Information Technology and Knowledge Management July-December 2010, Volume 2, No. 2, pp. 639-643

[6]Nafiz Arica, Fatos T. Yarman-Vural, (2000), "An Overview of Character Recognition Focused On Off-line Handwriting", IEEE, C99-06-C-203, 2000.

[7]Anoop M. Namboodiri, Anil K. Jain," Online Handwritten Script Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 26, No. 1, January 2004.

[8] S.P.Kosbatwar, S.K.Pathan, "Pattern Association for character recognition by Back-Propagation algorithm using Neural Network approach", International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.1, February 2012.