Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 41 (2011) 5587-5591

ISSN: 2229-712X

Handwritten English character recognition using neural network

Vijay Patil and Sanjay Shimpi

Department of Computer Engineering, Vidyalankar Institute of Technology, Wadala, Mumbai-37.

ARTICLE INFO

Article history:

Received: 9 September 2011;

Received in revised form:

15 November 2011;

Accepted: 24 November 2011;

Keywords

Handwritten Character Recognition,
Feature Extraction,
Back propagation network,
Multilayer Perceptron Network.

ABSTRACT

Neural Networks are being used for character recognition from last many years. This paper presents creating the Character Recognition System, in which Creating a Character Matrix and a corresponding Suitable Network Structure is key. The Feed Forward Algorithm gives insight into the enter workings of a neural network; followed by the Back Propagation Algorithm which compromises Training, Calculating Error, and Modifying Weights. We have made an attempt to recognize handwritten English characters by using a multilayer perceptron with one hidden layer. In addition, an analysis has been carried out to determine the number of hidden nodes to achieve high performance of back propagation network in the recognition of handwritten English characters. The results showed that the MLP networks trained by the error back propagation algorithm are superior in recognition accuracy and memory usage. The result indicates that the back propagation network provides good recognition accuracy of more than 70% of Handwritten English characters.

Introduction

One of the most classical applications of the Artificial Neural Network is the Character Recognition System. This system is the base for many different types of applications in various fields, many of which we use in our daily lives. Cost effective and less time consuming, businesses, post offices, banks, security systems, and even the field of robotics employ this system as the base of their Operations. Handwritten character recognition is a difficult problem due to the great variations of writing styles, different size (length and height) and orientation angle of the characters. Handwritten Character recognition is an area of pattern recognition that has become the subject of research during the last some decades. Neural network is playing an important role in handwritten character recognition. Many reports of character recognition in English have been published but still high recognition accuracy and minimum training time of handwritten English characters using neural network is an open problem. Therefore, it is a great important to develop an automatic handwritten character recognition system for English language. In this paper, efforts have been made to develop automatic handwritten character recognition system for English language with high recognition accuracy and minimum training and classification time.

Character Modeling

The English language consists of 26 characters (5 vowels, 21 consonants) and is written from left to right. A set of hand written English characters is shown in Figure 1 (Lower case) and Figure 2 (Upper case). The experiments are performing on both the 26 Uppercase Characters and Lowercase Characters.

Fig. 1: A Set of Handwritten English Characters (Lower Case)

Tele:

E-mail addresses: vijay_900@yahoo.com, sanjay_shimpi@yahoo.com

Fig. 2: A Set of Handwritten English Characters (Upper Case)

Character Recognition System

The Character Recognition System must first be created through a few simple steps in order to prepare it for presentation into MATLAB. The matrixes of each letter of the alphabet must be created along with the network structure. In addition, one must Understand how to pull the Binary Input Code from the matrix, and how to interpret the Binary Output Code, which the computer ultimately produces.

Character Matrixes: A character matrix is an array of black and white pixels; the vector of 1 represented by black, and 0 by white. They are created manually by the user, in whatever size or font imaginable; in addition, multiple fonts of the same alphabet may even be used under separate training sessions [3].

Creating a Character Matrix: First, in order to endow a computer with the ability to recognize characters, we must first create those characters. The first thing to think about when creating a matrix is the size that will be used. Too small and all the letters may not be able to be created, especially if you want to use two different fonts. On the other hand, if the size of the matrix is very big, their may be a few problems. Training may take more time, and results may take hours.

In addition, the computer's memory may not be able to handle enough neurons in the hidden layer needed to efficient and accurately process the information. However, the number of neurons may just simply be reduced, but this in turn may greatly increase the chance for error. For experimental purpose a matrix size of 8 x 5 was created, through the steps as explained above, because it may not be able to process in real time. (See Figure 3)

First Character 'A'	Second Character 'B'	Third Character 'C'
0 0 1 0 0	1 1 1 1 0	0 1 1 1 1
0 1 0 1 0	1 0 0 0 1	1 0 0 0 0
1 0 0 0 1	1 0 0 0 1	1 0 0 0 0
1 0 0 0 1	1 1 1 1 0	1 0 0 0 0
1 1 1 1 1	1 0 0 0 1	1 0 0 0 0
1 0 0 0 1	1 0 0 0 1	1 0 0 0 0
1 0 0 0 1	1 0 0 0 1	1 0 0 0 0
1 0 0 0 1	1 1 1 1 0	0 1 1 1 1

Fig. 3: A matrix size of 8 x 5 was created for the all alphabets

Neural Network: The network receives the 40 Boolean values as a 40-element input vector. It is then required to identify the letter by responding with a 26-element output vector. The 26 elements of the output vector each represent a letter. To operate correctly, the network should respond with a 1 in the position of the letter being presented to the network. All other values in the output vector should be 0. In addition, the network should be able to handle noise.

Architecture: The neural network needs 40 inputs and 26 neurons in its output layer to identify the letters. The network is a two-layer log-sigmoid network. The log-sigmoid transfer function was picked because its output range (0 to 1) is perfect for learning to output Boolean values. The hidden (first) layer has 5 neurons. This number was picked by guesswork and experience. If the network has trouble learning, then neurons can be added to this layer. The network is trained to output a 1 in the correct position of the output vector and to fill the rest of the output vector with 0's. However, noisy input vectors may result in the network not creating perfect 1's and 0's.

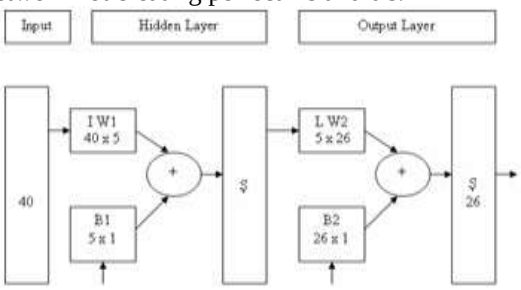


Fig 4: Neural Network Architecture

Setting the Weights: There are two sets of weights; input-hidden layer weights and hidden-output layer weights. These weights represent the memory of the neural network, where final training weights can be used when running the network. Initial weights are generated randomly there, after; weights are updated using the error (difference) between the actual output of the network and the desired (target) output. Weight updating occurs each iteration, and the network learns while iterating repeatedly until a net minimum error value is achieved. First we must define notion for the patterns to be stored Pattern p. a vector of 0/1 usually binary– valued. Inputs arrive from the left and each incoming interconnection has an associated weight, wji. The perception processing unit performs a weighted sum at its input value.

The sum takes the form net $\sum_{i=1}^{n} OiWi$

Weights associated with each inter connection are adjusted during learning.

Training: To create a network that can handle noisy input vectors it is best to train the network on both ideal and noisy vectors. To do this, the network is first trained on ideal vectors until it has a low sum squared error. Then, the network is trained on all sets of ideal and noisy vectors. The network is trained on two copies of the noise-free alphabet at the same time as it is trained on noisy vectors. The two copies of the noise-free alphabet are used to maintain the network's ability to classify ideal input vectors. The network is again trained on just ideal vectors. This ensures that the network responds perfectly when presented with an ideal letter. All training is done using back propagation [1].

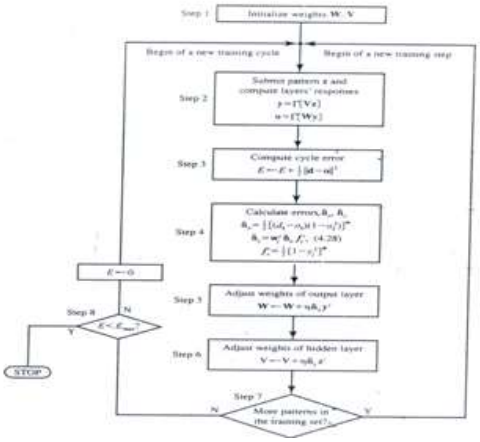


Fig 5: Error back-propagation training (EBPT algorithm) : algorithm flowchart

Training Procedure:

1. Split the data set into a training set and a test set. Normally the training set is larger than the test set. Often the desired outputs have to be normalized to the range [0: 1] since the sigmoid function only returns values in this range. The input patterns do not have to be normalized.
2. Initialize all weights, including all biases, to small random values normally in the range of [-1: +1]. This determines the starting point on the error surface for the gradient descent method, whose position can be essential for the convergence of the network.
3. 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.
4. Calculation of the difference between the actual output of each output neuron and its corresponding desired output. This is the error associated with each output neuron.
5. Back propagating this error 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.
6. Correcting each weight by its individual weight update.
7. Presenting and forward propagating the next input pattern. Repeat steps 3-7 until a certain stopping criterion is reached, for example that the error falls below a predefined value.

The one-time presentation of the entire set of training patterns to the net constitutes a training epoch. After terminating the training phase the trained net is tested with new, unseen patterns from the test data set. The patterns are forward propagated, using the weights now available from training, and the error at the output layer is determined (no weight-update is performed!). 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, ...).

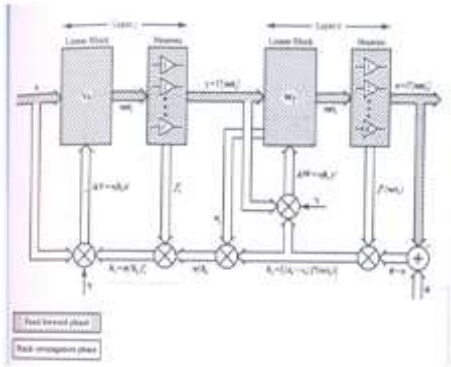
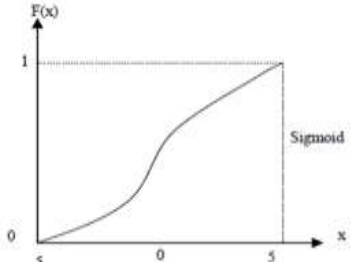


Fig 6: Error back propagation training (EBPT algorithm): block diagram illustrating forward and backward signal flow.

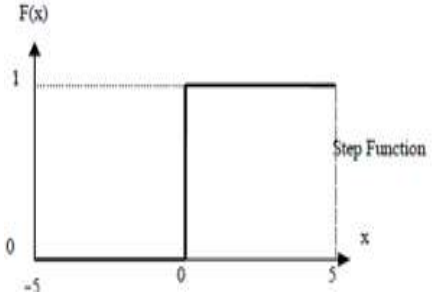
Figure-6 depicts the block diagram of the error back propagation trained network operation and explains both the flow of signal, and the flow of error within the network. The feed forward phase is self-explanatory. The shaded portion of the diagram refers to the feed forward recall. The blank portion of the diagram refers to the training mode of the network. The back propagation of error d-o from each output, for k=1,2-----K, using the negative gradient descent technique is divided into functional steps such as calculation of the error signal vector δ0 and calculation of the 9 weight matrix adjustment ΔW of the output layer. The diagram also illustrates the calculation of internal error signal vector δy and of the resulting weight adjustment ΔV of the input layer.

Forward- pass:

The forward – pass phase is initiated when an input pattern is presented to the network, each input unit corresponds to an entry in the input pattern vector, and each unit takes on the value of this entry. Incoming connection to unit J are at the left and Originate at units in the layer below. The function F(x), a sigmoid curve is illustrated as in fig below:



There is a transition from 0 to 1 that takes place when x is approximately $(-3 < \chi < 3)$ the sigmoid function performs assort at soft threshold that is rounded as shown in figure bellow.



The equation for the sigmoid function is

$$f(x) = \frac{1}{1 + e^{\sum_{i=1}^{n} oi\, wi}}$$

a. Input layer (i)

For input we have 26 inputs will be saved by the DAT file. Input Layer at neuron = output layer of neuron $Ii = Oi$

b. Hidden layer (h)

Hidden Layer input $h = Ik = \Sigma\, WkiOi$ as we have suggest that our weight is this and we are taking the value at our input at Character is A*I*S. Where A is the input-matrix, I is the hidden-layer input matrix and S is the Sigmoid function matrix as shown above figure.

Learning Factors

Necessary number of Hidden neurons: Single hidden layer networks can form arbitrary decision regions in n dimensional input pattern space. There exist certain useful solutions as to number *J* of hidden neurons needed for the network to perform properly. The number of hidden neurons depends on the dimension *n* of the input vector and on the number of separable regions in n dimensional Euclidean input space. Let us assume that the n dimensional no augmented input space is linearly separable into the *M* regions in the input space can be labeled as belonging to one of the R classes, where R <=M. Let us consider the case of input patterns of large dimension assuming that the expected, or estimated, size of the hidden nodes is small. For large-size input vectors compared to the numbers of hidden nodes or when n>=j, we have a form.

$$M=\binom{J}{0}+\binom{J}{1}+\binom{J}{2}+\text{---------}+\binom{J}{0}=2^{J}$$

It follows that the hidden neuron layer with three nodes would be capable of providing classification into up to eight classes; but since n>=j, the size of the input vector has to be larger than three.

The formulas can be inverted to find out how many hidden layer neurons J need to be used to achieve classification into M classes in n dimensional pattern space. This number constitutes the solution of the equation

$$M=1+J+\frac{J(J-1)}{2!}+\frac{J(J-1)(J-2)}{3!}+\frac{J(J-1)--(J-n+1)}{n!},\quad \text{for } J<n$$

For the case J<=n we have for $J = \log_2 M$

For a small number of inputs (fewer than 5), approximately twice as many hidden neurons as there are network inputs are used. As the number of inputs increases, the ration of hidden layer neurons to inputs decreases. The number of hidden neurons, with their associated weights, should be minimized in order to keep the number of free variable small, decreasing the need for large training sets. Validation set error is often used to determine the optimal number of hidden neurons for a given classification problem [2]

Number of Hidden Layers: Cybenko demonstrated that a single hidden layer, given enough neurons, can form any mapping needed. In practice, two hidden layers are often used to speed up convergence. While some feed forward networks have been reported in the literature to contain as many as five or six hidden layers, the additional layers are not necessary. The important thing to remember is that the network learns best when the mapping is simplest, so use good features. A network designer should be able to solve almost any problem with one or two hidden layers [2]. Single hidden layer is sufficient for any desired accuracy. Network never need more than two hidden layers. Fewer hidden nodes are preferable for better generalization ability [16]. There is a theorem which guaranties

that one hidden layer network can solve any problem if it has an appropriate number of neurons in its hidden layer, so at first, there is no reason why we must think of using two hidden layers in a network.

Initial Weights: The weights of the network to be trained are typically initialized at small random values. The EBP learning based on the single pattern error reduction requires a small adjustment of weights which follows each presentation of the training pattern. This scheme is called incremental updating.

Case Study and Result

Case Study-1

Typical application of the Error Back-propagation algorithm for handwritten character recognition. (Burr 1988) An input character is first normalized so that it extends to the full height and width of the bar mask. The handwritten alphabet character is then encoded into 13 line segments arranges in a template as in fig. The encoding takes the form of shadow projection. A shadow projection operation is defined as simultaneously projecting a point of the character into its three closest vertical, horizontal and diagonal bars. After all points are projected, shaded encoded bars are obtained. The shadow codes can be understood as extracted pattern features. The shadow does need to be normalized to within the 0,1, which is the range for unipolar neurons used in the network. The network thus has 13 valued inputs and 26 outputs, one for each alphabetic character. The convention used has been that the output corresponding to a letter should be 1 if the character is detected and 0 otherwise. According to the Burr the network has been tested with 12, 16, 20 and 24 hidden layer neurons. Networks with 20 or 24 hidden layer neurons usually trained faster with n=0.3 and 2.0.

Case study 2

In this case study case, the network training using the EBPT for the bit map classification of a single hidden layer network. The task is to design a network that can classify the simplified bit maps of characters. The matrixes if each letter of the alphabet must be created along with network. A character matrix of 8x5 is created. The vector of 1 represented by black and 0 by white. The network receives the 40 input Boolean values as a 40 element input vector. It is then required to identity the letter by responding with a 26 element output vector. The convention used has been that the output corresponding to latter should be max if the character is detected and 0 otherwise.

Conclusion

The problem demonstrates how simple pattern recognition can be designed. An experimental result shows that back propagation network yields good recognition accuracy of more than 70%. Note that the training process did not consist of a single call to a training function. Instead, the network was trained several times on various input vector and changing of value of learning constants and hidden nodes till we are not get the exact results. It is continuously learning process. The training time may be reduced by using good feature extraction techniques.

References

[1] Jacek M. Zurada, *"Introduction to Artificial Neural Systems"*, Jaico Publishing House, Delhi.

[2] Kevin L. Priddy, Paul E. keller, *"Artificial Neural Networks: An introduction"*

[3] Fakhraddin Mamedov, Jamal Fathi Abu Hasna, *"Character Recognition using Neural Network"*, Near East University, North Cyprus, Turkey via Mersin-10, KKTC

[4] Srinivasa Kumar Devireddy, Settipalli Appa Rao, *"Handwritten Character Recognition using Back Propagation Network",* 2005-2009 JATIT

[5] Anita Pal & Dayashankar Singh, *"Handwritten English Character Recognition using Neural Network",* International Journal of Computer Science & CommunicationVol. 1, No. 2, July-December 2010, pp. 141-144

[6] Saleh Ali K. Al-Omari, Putra Sumari, Sadik A. Al-Taweel and Anas J.A. Husain, *"Digital Recognition using Neural Network",* Journal of Computer Science 5 (6): 427-434, 2009 ISSN 1549-3636 © 2009 Science Publications

[7] Eric W. Brown, *"Applying Neural Networks to Character Recognition"*

[8] Vamsi K. Madasu1, Brian C. Lovell2, M. Hanmandlu3, "Hand printed Character Recognition using Neural Networks", 1School of ITEE, University of Queensland, Australia 2NICTA and School of ITEE, University of Queensland, Australia 3Department of Electrical Engineering, I.I.T. Delhi, India

[9]Srinivasa Kumar Devireddy, Settipalli Appa Rao, *"Handwritten Character Recognition using Back Propogation Network",* Nalanda Institute of Engineering & Technology, Kantepudi , Sattenapalli (M), Guntur (Dt.), A.P., India.

[10] O Matan, R.K.Kaing, "Handwritten Character Recognition Using Neural Network Architecture", in Proceedings of the 4th USPS advanced Technology conference, Washington D.C. PP1003-1011, November 1990.

[11]Carlos Agell, *"Neural Networks for Pen Characters Recognition"*, Department of Electrical Engineering and Computer Science University of California, Irvine Irvine, CA 96217

[12] Deepayan Sarkar," Optical Character Recognition using Neural Network", Department of Statistics University of Wisconsin, Madison UW ID: 9017174450, December 18, 2003

[13] Alexander J. Faaborg, *"Using Neural Networks to create an Adaptive Character Recognition System",* Cornell University, Ithaca NY *(May 14, 2002)*

[14] Velappa Ganapathy, and Kok Leong Liew, *"Handwritten Character Recognition Using Multiscale Neural Network Training Technique"*, World Academy of Science, Engineering and Technology 39 2008

[15] Shashank Araokar,"Visual Character Recognition using Artificial Neural Networks"

[16] P. Mangesh Kumar (M.E. environment Engg.), Dr, S. Amal Raj (Asst. Prof.) Centre of Environmental studies, anna University, Chennai

Comparison:

	Using 13 line segment	Using 8 x 5 matrix
Number of Layers	3	3
Input vectors	13	40
Feature method	13 segment bar	Mapping of character image into 8 x 5 matrix
Numberof neurons in hidden layers	12	5
Learning Rate	0.095	0.1
Error Rate	0.01	0.01
No of Epcohes	16000	20000
Result	The network has been tested with n=1.0 and 24 hidden layer neurons, we get 40 to 60% result	The network has been tested with n=0.1 and 22 hidden layer neurons, we get 60 to 70% result