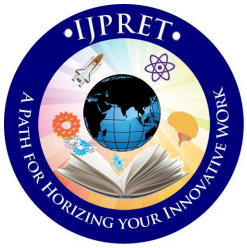
Research Article *ISSN: 2319-507X*
Shama Naik, IJPRET, 2013; Volume 1(8): 979-987 *IJPRET*



INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

HANDWRITING RECOGNITION USING NEURAL NETWORKS

SHAMA D NAIK, PROF. H.G. VIRANI

Dept. of Electronics & Telecommunication Engineering, Goa College of Engineering.

Accepted Date:

27/02/2013

Publish Date:

01/04/2013

Keywords

Neural Networks,
Feature Extraction,
Classification

Corresponding Author

Ms. Shama D Naik

Abstract

This paper presents a simple learning rule for recognition of handwritten character on our computer screen using artificial neural network. Kohonen self organization map for pattern classification which employs unsupervised learning algorithm is used. One advantage of proposed scheme is that the system is quite tolerant to changing conditions and inputs. Here first the preprocessing is done of the images. These images are then trained using NN .The trained images are then used for classification and recognition.

I. INTRODUCTION

Character recognition is the process to classify the input character according to the predefine character class. With increasing the interest of computer applications, modern society needs the input text into computer readable form. This research is a simple approach to implement that dream as the initial step to convert the input text into computer readable form. Some research for hand written characters are already done by researchers with artificial neural networks. In this paper Kohonen neural network is being used. The self-organizing map differs considerably from the feed forward back propagation neural network in both how it is trained and how it recalls a pattern. The self-organizing map does not use an activation function or a threshold value. In addition, output from the self-organizing map is not composed of output from several neurons; rather, when a pattern is presented to a self-organizing map, one of the output neurons is selected as the "winner." This "winning" neuron provides the output from the self-organizing map. Often, a "winning" neuron represents a group in the data that is presented to the self-organizing map.

II. THE PROPOSED SYSTEM

The overall method of the implemented system is illustrated.

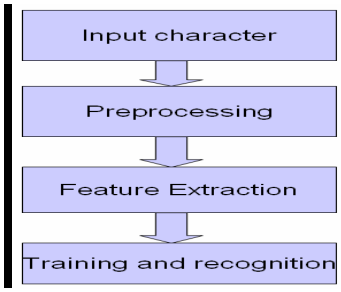


Fig (1) Overall model of the system

Feature Extraction and Preprocessing

Feature extraction is the process of extracting essential information contain from the image segment containing a character. It plays a vital role in the whole recognition process. This effectively reduces the number of computation and hence reduce the learning time in the training session of the neural network and faster the recognition process.

Preprocessing involves finding the appropriate character boundaries of the character. This involves finding the top,

bottom, right and left margins for the character.

Algorithm:

1. start at the first x and first y pixel of the image pixel(0,0), Set number of lines to 0
2. scan up to the width of the image on the same y-component of the image

(a) if a black pixel is detected register y as top of the first line

(b) if not continue to the next pixel

(c) if no black pixel found up to the width increment y and reset x to scan the next horizontal line

3. start at the top of the line found and first x-component pixel(0,line_top)
4. scan up to the width of the image on the same y-component of the image

(a) if no black pixel is detected register y-1 as bottom of the first line. Increment number of lines

(b) if a black pixel is detected increment y and reset x to scan the next horizontal line

5. Start below the bottom of the last line found and repeat steps 1-4 to detect subsequent lines.

Similarly the above algorithm can be used to find all the boundaries of the character.

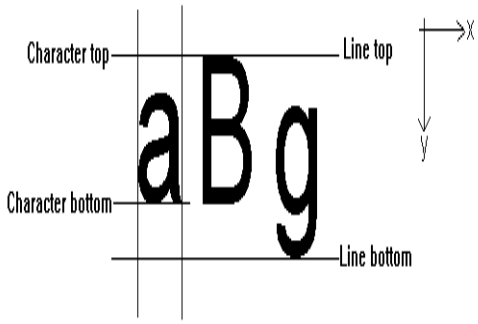


Fig (2) Boundaries of the characters

Once the cropping has taken place, the image must be down sampled. This involves reducing the image to a 5 X 7 resolution. To understand how to reduce an image to 5X 7, begin by thinking of an imaginary grid being drawn on top of the high-resolution image. The grid divides the image into regions, five across and seven down. If any pixel in a region is filled, then the corresponding pixel in the 5 X 7 down sampled image is also filled in. Once the boundaries are obtained certain features are extracted from the image that will help in the recognition process. Down sampling helps in getting the images of constant size

no matter how the character is drawn. This down sampled image is then provided to the neural network which through series of steps recognizes the pattern.

III LEARNING AND RECOGNITION USING SELF ORGANIZING MAP

The self-organizing map works differently than the feed forward neural network. The self-organizing map only contains an input neuron layer and an output neuron layer. There is no hidden layer in a self-organizing map. The input to a self-organizing map is submitted to the neural network via the input neurons. The input neurons receive floating point numbers that make up the input pattern to the network. A self-organizing map requires that the inputs be normalized to fall between -1 and 1. Presenting an input pattern to the network will cause a reaction from the output neurons. The output of a self-organizing map is very different from the output of a feed forward neural network. In a self-organizing map, only one of the output neurons actually produces a value. Additionally, this single value is either **true** or **false**. Therefore, the output from the self-organizing map is usually the index of the neuron that is fired (e.g. Neuron #5). The structure of a typical self-organizing map is shown in Figure 3.

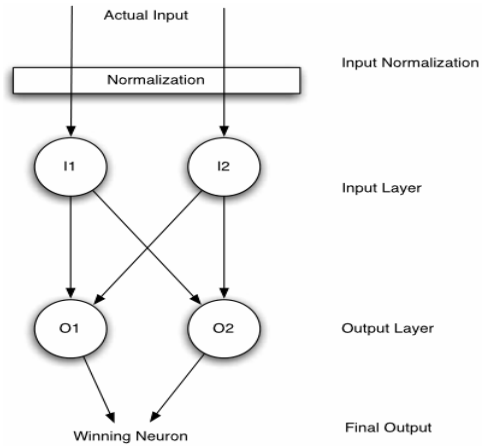


Fig (3) Structure of SOM

This network will have only two input neurons and two output neurons. The input to be given to the two input neurons is shown in Table 1

Input Neuron 1 (I1)	0.5
Input Neuron 2 (I2)	0.75

Connection Weights in the Sample Self-Organizing Map are given below:

I1 -> O1	0.1
I2 -> O1	0.2
I1 -> O2	0.3
I2 -> O2	0.4

Table 1: Connection weights

Using these values, any one neuron will win and produce output.

A. Normalizing the Input

The self-organizing map requires that its input be normalized.

The self-organizing map places strict limitations on the input it receives. Input to the self-organizing map must be between the values of –1 and 1. In addition, each of the input neurons must use the full range. If one or more of the input neurons were to only accept the numbers between 0 and 1, the performance of the neural network would suffer.

Input for a self-organizing map is generally normalized using multiplicative normalization. To perform multiplicative normalization, we must first calculate the vector length of the input data, or vector. This is done by summing the squares of the input vector and then taking the square root of this number, as shown in Equation 1.

$$f = \frac{1}{\sqrt{\sum_{i=0}^{n-1} x_i^2}}$$

The above equation produces the normalization factor that each input is multiplied by to properly scale them. Using the sample data provided in Tables 1 the normalization factor is calculated as follows:

```
1.0 / Math.sqrt( (0.5 * 0.5) + (0.75 * 0.75) )
```

This produces a normalization factor of 1.1094.

B. Calculating Each Neuron's Output

To calculate the output, the input vector and neuron connection weights must both be considered. First, the dot product of the input neurons and their connection weights must be calculated. To calculate the dot product between two vectors, you must multiply each of the elements in the two vectors as shown in Equation 2

$$[0.5 \quad 0.75]*[0.1 \quad 0.2]=(0.5*0.75)+(0.1*0.2)=0.395$$

As you can see from the above calculation, the dot product is 0.395. This calculation will have to be performed for each of the output neurons. The calculations necessary for the second output neuron is carried out in the same way. The output must now be normalized by multiplying it by the normalization factor that was determined in the previous step. You must multiply the dot product of 0.395 by the normalization factor of 1.1094. The result is an output of 0.438213. Now that the output has been calculated and normalized, it must be mapped to a bipolar number.

C. Mapping to a Bipolar Ranged Number

A bipolar number is an alternate way of representing binary numbers. In the bipolar system, the binary zero maps to –1 and the binary one remains a 1. Because the input to the neural network has been normalized to this range, similar normalization on the output of the neurons is done. To make this mapping, we multiply by two and subtract one. For the output of 0.438213, the result is a final output of –0.123574.The value –0.123574 is the output of the first neuron. This value will be compared with the outputs of the other neuron. By comparing these values we can determine a "winning" neuron.

D. Choosing the Winner

To determine a winning neuron, value for the second output neuron must be calculated. The same normalization factor is used to calculate the second output neuron as was used to calculate the first output neuron. . If we apply the dot product for the weights of the second output neuron and the input vector, we get a value of 0.45. This value is multiplied by the normalization factor of 1.1094, resulting in a value of 0.0465948. We can now calculate the final output for neuron 2 by converting the output of 0.0465948 to bipolar, which yields –0.9068104. The first neuron has an output value of –0.123574 and the second neuron has an output value of –0.9068104. To choose the winning neuron, we select the neuron that produces the largest output value. In this case, the winning neuron is the second output neuron with an output of –0.9068104, which beats the first neuron's

output of –0.123574. In the next section these weights can be adjusted to produce output that is more suitable for the desired task. The training process modifies these weights and will be described in the next section.

IV. How a Self-Organizing Map Learns

There are several steps involved in the training process. Overall, the process for training a self-organizing map involves stepping through several epochs until the error of the self-organizing map is below an acceptable level. The training process for the self-organizing map is competitive. For each training set, one neuron will "win." This winning neuron will have its weight adjusted so that it will react even more strongly to the input the next time it sees it. As different neurons win for different patterns, their ability to recognize that particular pattern will increase.

Learning Rate

The learning rate is a variable that is used by the learning algorithm to adjust the weights of the neurons. The learning rate must be a positive number less than 1, and is typically 0.4 or 0.5. In the following sections, the learning rate will be specified by the symbol alpha.

Adjusting Weights

The memory of the self-organizing map is stored inside the weighted connections between the input layer and the output layer. The weights are adjusted in each epoch. An epoch occurs when training data is presented to the self-organizing map and the weights are adjusted based on the results of this data. The adjustments to the weights should produce a network that will yield more favorable results the next time the same training data is presented. Epochs continue as more and more data is presented to the network and the weights are adjusted.

Eventually, the return on these weight adjustments will diminish to the point that it is no longer valuable to continue with this particular set of weights. When this happens, the entire weight matrix is reset to new random values; thus beginning a new cycle. The final weight matrix that will be used will be the best weight matrix from each of the cycles. We will now examine how weights are transformed.

The original method for calculating changes to weights, which was proposed by Kohonen, is often called the additive method. This method uses the following equation:

$$w^{t+1} = \frac{w^t + \alpha x}{length(w^t + \alpha x)}$$

The variable **x** is the training vector that was presented to the network. Variable **wt** is the weight of the winning neuron, and the result of the equation is the new weight. This additive method generally works well for most self-organizing maps; however, in cases for which the additive method shows excessive instability and fails to converge, an alternate method can be used. This method is called the subtractive method.

The subtractive method uses the following equations:

$$e = x - w^t$$

$$w^{t+1} = w^t + \alpha e$$

Calculating the Error

The purpose of the self-organizing map is to classify input into several sets. The error for the self-organizing map, therefore, provides a measure of how well the network is classifying input into output groups. The error itself is not used to modify the weights, as is the case in the back propagation algorithm.

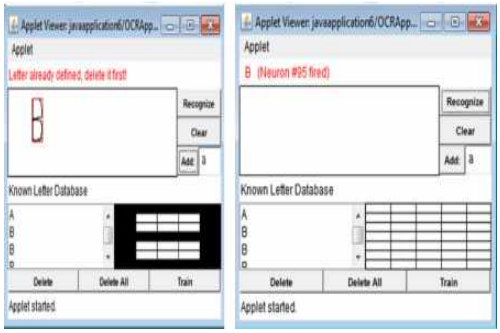


Fig (4) Results

The complexity of the problem is greatly increased by noise in data and by an almost infinite variability of hand writing as a result of the writer and the nature of the writing.

Following table depicts the percentage/rate of success in recognition of some characters drawn in different styles.

Character Drawn	No of Patterns Given	#Recognized	#Not Recognized	Rate (%) of Recognition
A	5	4	1	80
D	5	4	1	80
E	5	4	1	80

Table 2: Recognition rates

V. CONCLUSIONS

Recognition of handwritten characters has been a considerable interest to researchers working on OCR. The complexity of the problem is greatly increased by noise in data and by an almost infinite variability of hand writing as a result of the writer and the nature of the writing. Here generalized recognition system for hand written character is being developed. Some time humans can't even recognize their own hand writings and the handwritten character varies from man to man and depends on many factors i.e. emotion, pen pressure, and environment. This is why; it is too difficult to get accurate efficiency. Though it is problematic if a man follows standard writing rules, the filtering and feature extraction is done more accurately, and then it is possible to recognize the handwritten text into computer readable form in more precise manner.

REFERENCES

1. O, Trier, A.K. Jain, T. Taxt. "Feature extraction methods for character recognition", pattern recognition, 29(4): 641-662, 1996

2. Vuokko Vuori, Erkki Oja, "Analysis of different writings styles with the self organizing map". In Proceedings of the 7th International Conference on neural information processing. Vol. 2, 2000, pp 1243-1247. November 2000

3. Davis, R.H., Lyall, "Recognition of hand written character – A review", Image and vision computing 4,4, (1986) 208-218

4. Introduction to Neural Networks with Java, 1st Edition, 097732060X, Jeff Heaton.

5. Pankaj Agrawal, " Hand-Written Character Recognition Using Kohonen Network", Issue 3, September 2011