Hand written character recognition using Neural Network

Shruti Jain, Vaishali Choudhary
RKGITW, Ghaziabad

Abstract
This paper presents the idea of recognition of any symbol drawn by dragging mouse in the area provided. For the purpose, we will first train the neural network. For the purpose of training we use Backpropogation algorithm and to match the input and output we use simple K nearest neighbour algorithm. Simple Kohonen Network will be used with bipolar sigmoidal activation function to realize the functionality. The advantage of proposed system is that it is flexible and more tolerant to changed input conditions. Also, the recognition ratio is excellent.

Keywords
Artificial Neural Networks, Handwritten Character, Kohonen Network, Mouse Dragging, K Nearest Neighbour, Matrix, Neuron etc.

1. Introduction
Character Recognition basically is the process of classifying the input of user into one of predefined character classes. It is a modern fascination for a computer to be able to understand the character drawn with help of simply dragging the mouse on the screen. Some research for hand written characters are already done by researchers with artificial neural networks. In this paper we use Kohonen neural network. An artificial neural Network basically is used as a data modelling tool to capture and represent complex input output relationships. The Kohonen Self-Organizing Map (SOM) designed by Tuevo Kohonen is a variation of the traditional Artificial Neural Network. It is a third generation neural network, meaning that many of its functional characteristics are thought to mirror those found in biological fact. An SOM consists of a collection of nodes of neurons that are each connected to every other node and each node has associated with it a set of input weights w. The SOM also has associated with it a metric for determining which nodes are in the neighborhood N of a given node [1].

Though not directly related to neural networks, the process by which the user is allowed to draw the characters on the computer screen using mouse dragging is an important aspect this paper. Most of the actual drawing is handled by the process MouseMotionEvent. If the mouse is being drug, then a line will be drawn from the last reported mouse drag position to the current mouse position. It is not enough to simply draw a dot. The mouse moves faster than the program has time to As the program proceeds, the method is called repeatedly.
2. Proposed System

The overall method of the proposed system is presented in Fig 1.

A. Input Image

The input for the network is obtained simply by dragging the mouse on the area provided to draw the image. Ignoring the concept of coloured paper or character, the black part of the image is considered as the character and the white part is considered as the paper.

B. Feature Extraction

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 [2,3]

C. Drawing Images

accept & process all values for. This is done with the following lines of code [1].

```java
entryGraphics.setColor(Color.black);
entryGraphics.drawLine(lastX,lastY,e.getX(),e.getY());
getGraphics().drawImage(entryImage,0,0,this);
lastX = e.getX(); lastY = e.getY();
```

Fig 1: Overall Model of Implemented System

Fig 2: Illustration of Input
D. Downsampling of Image

Fig 3: Sample Downsampling for Letter ‘K’

For every input to be processed, the image must be downsampled. We take a matrix, say of size 5*7. At every pixel, where the mouse is dragged, we take .5, and for white space, we take -.5. In neural network training tasks, it's preferred to represent training patterns in "bipolar" way, placing into input vector "0.5" instead of "1" and "-.5" instead of "0". Such sort of pattern coding will lead to a greater learning performance improvement. This is done using a PixelGrabber class. First thing the program does is to draw a box around the letter, which eliminates all the white space. Later the image is cropped as:

```java
int w = entryImage.getWidth(this); int h = entryImage.getHeight(this);
PixelGrabber grabber = new PixelGrabber(entryImage, 0, 0, w, h, true);
grabber.grabPixels();
pixelMap = (int[]) grabber.getPixels();
```

After this code completes, the pixelMap variable which is an array of int datatypes, now contains the bit pattern of the image. Now, actual downsampling takes place and the image is converted to 5*7 matrix. This is one inside the downsampled quadrant.

```java
Protected boolean downSampleQuadrant(int x, int y)
{
int w = entryImage.getWidth(this);
int startX = (int)(downSampleLeft+(x*ratioX)); int startY =
(int)(downSampleTop+(y*ratioY)); int endX = (int)(startX + ratioX);
int endY = (int)(startY + ratioY);

for ( int yy=startY;yy<=endY;yy++ ) {for ( int xx=startX;xx<=endX;xx++ ) {int
loc = xx+(yy*w);
if ( pixelMap[loc] != -1 ) {return true;}
return false;}
}
```

3. Learning and Recognition[1,4]

The Kohonen network has two layers, an input layer and a Kohonen output layer. The input layer is a size determined by the user and must match the size of each row (pattern) in the input data file. A
A. How a Kohonen Network works

Learning process of a Kohonen network involves several steps. We use Backpropogation to train this network. Basic idea is to adjust the input to match the output several time by adjusting the weights so as to bring the error of the Kohonen neural network is below acceptable level. 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. As different neurons win for different patterns, their ability to recognize that particular pattern will be increased.
B. Adjusting the weights

Entire memory of the Kohonen network is within its the weighted connections. Weights are adjusted in each round in Backpropogation. The adjustments to the The variable $x$ is the training vector that was presented to the network. The variable $w^t$ is the weight of the winning neuron, and the variable $w^{t+1}$ is the new weight. The double vertical bars represent the vector length.

D. Training the Program to Recognize the Letters

The program may not be able to recognize any one’s handwriting because it is initially trained for particular handwriting only. Now we are supposed to make the Network learn about the characters that we want the network to recognize. We may also retain the sample weights should produce a network that will yield more favourable results the next time the same training data is presented.

4. Proposed Algorithm

A. Initialise network

For each node I set the initial weight vector $W_i (0)$ to be random. Set the initial neighbourhood $N_i (0)$ to a large value.

B. Present Input

The program feeds network the value of 0.5 for a black pixel and -0.5 for a white pixel. This array of 35 values is fed to the input neurons. This will return one of the 35 neurons which wins. This neuron is stored in the “best” integer. Winner node $c$ is calculated by calculating the maximum activation among all $p$ neurons participating. The neuron with the largest activation number is the winner. The neuron has the final output of 1 or this is the winner neuron. All other neurons in the layer have an output of zero.

C. Updating weights

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

$$w^{t+1}_{i} = \frac{w^t_{i} + \alpha x_i}{||w^t_{i} + \alpha x_i||}$$

we have already trained the network for.
We first train the network and then save it into a file ‘sample.dat’. If you already have a sample.dat file, we may also be able to load the pre-existing database into the program. Also, on saving the current sample.dat, we are also able to upload the same, next time we run the program, by clicking ‘Load’. Further, when we train the network, we are able to recognize the letters for which the network was trained for as shown in Figure 7.

References

1. Hand-Written Character Recognition Using Kohonen Network by Dr. Pankaj Agarwal (IJCT Vol 2)

