RECOGNITION OF HANDWRITTEN DIGITS USING MULTILAYER PERCEPTRONS

BY

VIOLETA SANDU and FLORIN LEON

Abstract. Neural networks are often used for pattern recognition. They prove to be a popular choice for OCR (Optical Character Recognition) systems, especially when dealing with the recognition of printed text. In this paper, multilayer perceptrons are used for the recognition of handwritten digits. The accuracy achieved proves that this application is a working prototype that can be further extended into a full handwritten text recognition system, addressing both digits and letters.

Key words: neural networks, multilayer perceptrons, handwritten text recognition, backpropagation.

2000 Mathematics Subject Classification: 92B20, 68T24.

1. Introduction

The subject chosen for this project is circumscribing both field of optical character recognition and intelligent character recognition field.

Optical character recognition [15], usually abbreviated as OCR, represents the mechanical or electronic translation of images containing handwritten, typewritten or printed text (usually captured by a scanner) into computer editable text.

Handwriting recognition [16] is the ability of a computer to receive and interpret intelligible handwritten text from sources such as paper, photographs, touch-screen and other devices. The image of the text written can be processed “off-line”, from a paper, by optical scanning (optical character recognition) or by intelligent word recognition. Alternatively, the movement of the writing instrument can be processed “on-line”, for instance, by a touch screen with stylus.
Handwriting recognition mainly entails optical character recognition. However, a complete handwriting recognition system also handles formatting, performs correct segmentation into characters and finds the most plausible words.

**On-line** handwriting recognition involves the automatic conversion of text, as it is written, on a special digital device or PDA, where a sensor captures the movements of the writing instrument and also the pen-up/pen-down switching. This kind of data is known as digital ink and can be regarded as a dynamic representation of handwriting. The obtained signal is converted into letter codes which can be used by the computer and in text processing applications. On-line character recognition is sometimes confused with optical character recognition [13].

**Off-line** handwriting recognition involves the automatic conversion of text from an image into letter codes that can be used within a computer or other text processing applications. The data obtained in this form is regarded as a static representation of handwriting. This technology is successfully used in businesses that process a large amount of handwritten documents, such as insurance companies. The off-line handwriting recognition is difficult, because different people have different handwriting styles. Nevertheless, limiting the types of input data allows improving the recognition process.

Real time on-line handwriting recognition systems became well known as commercial products in the last few years. Among these systems can be enumerated the input devices for personal digital assistants, such as those running Palm OS. The Apple Newton [17] pioneered this product. The algorithms used in these devices have the advantage that the order, speed and dimension of individual lines segments at the input are known. Also, the user can be constrained to use just certain shapes for the letters. These methods cannot be used in software scanning paper documents, so that the accurate recognition of handwritten documents is still an open problem. Accuracy rates of 80% to 90% for neat, clean characters, can be achieved, but this rate still translates to dozens of errors per page, making the technology useful only in very limited applications [12].

Recognition of cursive text is an active area of research, with recognition rates even lower than those of hand-printed text. Higher rates of recognition for the cursive text will hardly be achieved without using the contextual or grammatical information. For instance, the recognition of entire words from a dictionary is easier than trying to parse individual characters from a text. Reading the “Amount” line of a check (which is always a written number) is an example where using a small dictionary can increase recognition rates greatly. Knowledge of the grammar for the language that is scanned can also help to determine if a word is either a noun or a verb, for example, allowing greater accuracy. The shapes itself of individual cursive characters don’t contain sufficient information to accurately (greater than 98%) recognize all the handwritten cursive texts.
Here are some concrete software solutions for optical character recognition and handwriting recognition: *Abbyy Fine Reader* (it includes optical character recognition, intelligent character recognition, optical mark recognition, barcode recognition) [18], *OmniPage* (an OCR application) [19], *ReadIris* (a Belgian OCR software for Microsoft Windows and Mac OS) [20], *CuneiForm* (a Russian OCR system) [21], *Microsoft Office OneNote* (a digital notebook that allows the user to organize text, pictures, digital handwriting) [22], *MyScript* (an online handwriting recognition software) [23], *CellWriter* (an Open Source handwriting recognition program written for Linux) [24], *CalliGrapher* (a handwriting recognition software that supports all handwriting styles – cursive, print or mixed) [25].

2. Pattern Recognition

Shape recognition [7], [14] studies the classification of a set of objects, processes or events. Classification is a fundamental process which characterizes not only science but also social life. The purpose of shape recognition is determining the class from which the elements of a set are part of. Establishing the number of classes is a problem directly tied to the characteristics of the application. There can be applications with the number of classes known apriori, but also applications in which the number of classes must be determined with the help of an algorithm.

The classifier is the system that implements the operation of shape recognition, so it realizes a procedure of classification which was pre-determined. Some of the most important properties that permit the evaluation of a classifier are the following:

*Recognition* is expressed through the *rate of recognition*, which is the percent of shapes from the set which were correctly recognized by the classifier.

*Convergence* expresses the answering speed of a classifier. In most cases, there is a compromise between rate of recognition and the answering speed which characterizes a certain classifier.

*Safety* is described by a *classifier’s degree of trust*, which characterizes the classifier’s capacity to correctly classify distorted shapes given as input.

*Prediction* expresses the classifier’s capacity to correctly recognize shapes that are not in the training set. A measurement of this property is the *predictive ability*, which expresses the percent of shapes from the prediction set (so belonging to previously unknown classes) correctly recognized.

3. Neural Networks

A neural network can be defined as a reasoning model based on the structure of human brain. The human brain consists of a dense group of
interconnected neural cells – base units of information processing, called neurons. The human brain incorporates approximately 10 billion neurons and 60 trillion connections, called synapses, between them [10].

An artificial neural network consists of a number of processors very simple and strongly interconnected, called neurons, which are analogue to the biologic neurons of human brain. The main property of artificial neural networks is the capacity of learning. Neurons are interconnected through links and every link has a numerical weight associated. Weights are the basic means of realizing the long term memory of artificial neural networks. They express the power, or, in other words, the importance of every neuron. A neural network can “learn” through repeated adjustments of these weights.

The multilayer perceptron (MLP) is the most widely known and used type of neural network. Most of the times, the signals are transmitted through the network only in one way: from input to output. There are no loops so the output of every neuron does not affect that neuron. This architecture is called “feed forward”. The layers that are not directly connected to the environment are called hidden.

Up to present day, more than a hundred different learning algorithms for multilayer neural network are available (such as Quickprop [6], Rprop [8], BFGS [2], [3], [4], [5]), but most widely used remains the retro-propagation algorithm, or back-propagation algorithm.

Back-propagation is the most widely known and used supervised learning algorithm (supervised learning is a general method of representing a function that approximates a parameterized function; supervised learning requires input-output pairs of the learning function). Also called generalized delta algorithm because it extends the means to train a one layer perceptron (delta rule), it is based on minimizing the difference between the desired output and the real output, through descent error gradient method (the gradient informs us of how different functions vary).

In a back-propagation neural network, the learning algorithm has two phases. First, a training input pattern is presented to the network input layer. The network then propagates the input pattern from layer to layer until the output pattern is generated by the output layer. If this pattern is different from the desired output, an error is calculated and then propagated backwards through the network from the output layer to the input layer. The weights are modified as the error is propagated. As with any other neural network, a back-propagation one is determined by the connections between neurons (the network’s architecture), the activation function used by the neurons and the learning algorithm (or the learning law) that specifies the procedure for adjusting weights.

3.1. Optimizations of the Back-Propagation Algorithm

Training a multilayer perceptron is generally fairly slow, taking thousands or tens of thousands of epochs for complex problems. The most
widely known methods of accelerating the learning are: momentum method and applying a variable learning rate.

*Momentum* [9] proposes adding a term to weight adjustment. This term is directly proportionate to the last modification of the weight, meaning that the values with which the weights are adjusted are memorized and directly influence all future adjustments

\[
\Delta w_{jk}(p) = \beta \cdot \Delta w_{jk}(p-1) + \alpha \cdot y_j(p) \cdot \delta_k(p).
\]

The \( \beta \) term is a positive number \( 0 \leq \beta < 1 \) called *momentum constant*. Adding the new term will be done after updating the weight corrections.

*Variable learning rate* [11] consists in using an individual learning rate for each weight and adapting these parameters in each iteration, depending on the successive algebraic signs of the gradients.

If the change of the sum of squared errors \( \Delta \hat{E} \) has the same algebraic sign for several consequent epochs, then the learning rate parameter \( \alpha \), should be increased. If the algebraic sign of the change of the sum of squared errors \( \Delta \hat{E} \) alternates for several consequent epochs, then the learning rate parameter, \( \alpha \), should be decreased.

Given \( E_p \) sum of squared errors in the current epoch.

If \( \frac{E_p}{E_{p-1}} > r \), then \( \alpha \leftarrow \alpha \cdot d \).

If \( \frac{E_p}{E_{p-1}} < r \), then \( \alpha \leftarrow \alpha \cdot u \).

After that the new weights and thresholds are computed.

### 4. The Architecture of the Software Application

Logically, the application is divided into five modules: the image processing module, the neural network building module, the neural network training module, the dataset building module and a central module that unites all the others and also incorporates the GUI of the application.

#### 4.1. Image Processing Module

Processing an image consists of the following phases:

I) Detecting the lines of writing from an image (in the case in which the numbers are written on more than one line);

II) Detecting each digit from the image;
III) Mapping each digit to a matrix of values corresponding to black and white pixels;
IV) Extracting the specific features.

**Extracting the specific features.** In order for the data contained in the value matrix corresponding to each image of a digit to be an input set for the neural network, the matrix has to be transformed to a vector of characteristic values.

Extracting characteristic features from an image has the following steps:

a) *Scaling the symbols*
Each image of a symbol has been scaled to a matrix of 20x20 elements.

b) *Discrete Cosine Transform*
Using an orthogonal transformation (discrete Karhunen-Loeve, discrete Fourier, discrete Cosine, Hadamard etc.) and based on the decorrelation properties of these transforms, the coefficients of these transforms can be used as features.

Because the implementation of the Karhunen-Loeve discrete transform is difficult, the best approximation of this transform is DCT. This model is not very restrictive and can be successfully used in many applications.

The formula for the bidimensional case is:

\[
S(u, v) = \frac{2}{\sqrt{n \cdot m}} \cdot C(u) \cdot C(v) \cdot \sum_{y=0}^{n-1} \sum_{x=0}^{n-1} s(x, y) \cdot \cos \left( \frac{(2x+1)u}{2n} \pi \right) \cdot \cos \left( \frac{(2y+1)v}{2m} \pi \right)
\]

for \( u=0,\ldots,n, v=0,\ldots,m \), \( s \) is a matrix of \( n \times m \) elements, \( S \) is the Discrete Cosine Transform. \( C(u) \) and \( C(v) \) are calculated using the following formula:

\[
C(u) = \begin{cases} 
2^{-\frac{1}{2}}, & \text{for } u = 0 \\
1, & \text{otherwise}
\end{cases}
\]

Applying this transform, it results a matrix having the same dimensions, with floating point values, weakly correlated. The significant values of this matrix are in the upper left corner.

c) *Zig-Zag Sorting*
The Zig-Zag sorting [1] is a method of vectoring a DCT matrix, following the rule described in Fig. 1.
Because the DCT bidimensional transform concentrates the information contained in the initial image in the upper left corner of the resulting matrix, after the Zig-Zag sorting, the same information will be found in the first elements of the resulting vector. Only the first F values of the Z[D*D] vector are used, only a fraction of the values, the ones marked in black in the above figure. As previously stated, these elements concentrate most of the information from the initial image and only they will be used in the recognition phase.

For the current case, about 10% (41 elements) of the vector values obtained after the Zig-Zag sort were chosen to be used as an input for the neural network. The percent of 10% was chosen empirically, based on a series of trials and observations.

4.2. Neural Network Building Module

The characteristic features vector obtained in the previous phase will be used as an input for a perceptron neural network with four layers (two hidden layers) and with unipolar sigmoid activation function.

The number of neurons used is 41 for the input layer, 30 for the first hidden layer, 20 for the second hidden layer and 10 for the output layer.

For training the back-propagation algorithm (Sec. 3) with adaptive learning rate was used. Each neuron in the output layer represents a digit, the desired output for each digit consists in activating the neuron on the position corresponding to that specific digit and suppressing the other neurons.

The convergence of the network was a problem that appeared throughout the implementation of the project. Initially, a constant learning rate was used, but, with such an approach, the neural network converged very slowly. This is why an adaptive learning rate was chosen, which considerably lowers convergence time and avoids the cases in which the neural network is
blocked in a local minimum. It was observed that using adaptive learning rate, a convergence time about 60 times smaller was obtained.

5. Case Studies

5.1. Recognition Accuracy

In order to test the application, ten examples of every digit were given for recognition. Table 1 presents the results obtained for each digit, more exactly, the numbers of examples recognized of the ten.

Table 1

<table>
<thead>
<tr>
<th>Digit</th>
<th>No. of recognized examples</th>
<th>No. of given examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

Fig. 2 presents the data from the table in the form of a graphic, in order to emphasize the percent of correctly recognized examples.

During the testing of the application, it was found that two types of erroneous recognitions can appear: either the digit is recognized as being another digit, or the digit was not recognized at all. It’s important to notice the fact that the second type of errors is preferably to the first, because this can be solved by adding the unrecognized examples to the training set and retraining the network.

Fig. 2 – Recognition degree of tested examples.
5.2. Convergence of the Neural Network

For comparison purposes, the graph of the evolution of error depending on the epoch in the case of using adaptive learning rate (Fig. 3) and in the case of using constant learning rate (Fig. 4) shall be presented.

It can be seen that with the use of constant learning rate the network converges after about 6000 epochs, while with the use of an adaptive learning rate the network converges after about 160 epochs.

Fig. 3 – Error graph depending on the epoch, in the case of using a constant learning rate.

Fig. 4 – Error graph depending on the epoch, in the case of using an adaptive learning rate.
6. Conclusions

This project doesn't claim to be an innovation in the domain of handwriting recognition. It was primarily built as an experiment and it should be noticed that very good results were obtained (such as small response time and high recognition rate). The originality of this project consists, on one hand, in combining different optimization methods for building and training the neural network (Discrete Cosine Transform, Zig-zag sorting, adaptive learning rate) and, on the other hand, in the easy to use graphical interface, that includes even a histogram that indicates to what extent the digit drawn by the user is recognized by the neural network as being closer of one or another of the ten digits.

Of course, there are a series of improvements that could be made to this project.

A first direction of improvement would be modifying the user interface module. New menu options could be added, depending on the user’s specifications. For example, when a digit cannot be recognized, a panel containing all the digits can be displayed in order for the user to be able to choose the desired digit.

Another direction of improvement would be adding new modules, for example a module for recognizing letters – in the case in which the application would want to be used for recognizing cursive handwritten text.

This application could also be developed for mobile telephones with touch screen and PDAs. For this purpose, the application would require a new module used for communicating to the mobile device (for example, using Bluetooth). Once this module is realized, the digits could be drawn on the mobile device, saved as an image and then sent to the computer for processing and recognition. Even more, the current project could be ported on a mobile device and it could be an application through which the user could save the telephone numbers written on the touch screen with the aid of the stylus.

Acknowledgements. This work was supported in part by CNCSIS grant 316/2008, contract no. 671/2009, Behavioural Patterns Library for Intelligent Agents Used in Engineering and Management.

Received: September 10, 2009

"Gheorghe Asachi" Technical University of Iași,
Department of Computer Science and Information Technology,
e-mail: fleon@cs.tuiasi.ro

REFERENCES


RECUNOAȘTEREA CIFRELOR SCRISE DE MÂNĂ 
UTILIZÂND PERCEPTRONI MULTISTRAT

(Rezumat)

Rețelele neuronale sunt deseori utilizate pentru recunoașterea formelor. Ele se dovedesc a fi o alegere adecvată pentru sistemele de OCR (Recunoașterea Optică a Caracterelor), mai ales pentru recunoașterea textelor tipărite. În acest articol, perceptronii multistrat sunt folosiți pentru recunoașterea cifrelor scrise de mâna. Acuratețea obținută dovedește că aplicația este un prototip funcțional care poate fi extins în viitor la un sistem complet de recunoaștere a textelor scrise de mână, care să trateze atât cifre cât și litere.