Online Handwritten Character Recognition Using an Optical Backpropagation Neural Network

Walid A. Salameh
Princess Summaya University for Science and Technology, Amman, Jordan

walid@psut.edu.jo

Mohammed A. Otair
Arab Academy for Financial Sciences and Banking, Amman, Jordan

otair@just.edu.jo

Abstract

There are many successful applications of Backpropagation (BP) for training multilayer neural networks. However, they have many shortcomings. Learning often takes insupportable time to converge, and it may fall into local minima at all. One of the possible remedies to escape from local minima is using a very small learning rate, but this will slow the learning process. The proposed algorithm is presented for the training of multilayer neural networks with very small learning rate, especially when using large training set size. It can apply in a generic manner for any network size that uses a backpropgation algorithm through optical time. This paper studies the performance of the Optical Backpropagation algorithm OBP (Otair & Salameh, 2004a, 2004b. 2005) on training a neural network for online handwritten character recognition in comparison with backpropagation BP.

Keywords: Neural Networks (NN), Backpropagation (BP), Optical backprpoagation (OBP), Online Handwritten Character Recognition.

Introduction

Pattern recognition applications, especially handwritten character recognition, is one of the most widely used applications of Backpropagation Neural Networks (*BPNN*). However, the problems of slow training process, choosing suitable values for the parameters and escaping from local minima remain a major problem that face the developers when using *BPNN*.

In recent years, a pen-based computer has become popular as an input device. Since online handwritten character recognition systems are the preferred input methods, there are many online character recognition methods that have been introduced (Bishop, 1995; Okamoto, & Yamamoto, 1999; Plamondon, Lopresti, Schomaker, & Shrihari, R., 1999; Tappert, Suen, & Wakahara, 1990; Wakahara, Murase, & Odaka, 1992).

In this paper, two neural networks are developed and trained to recognize handwritten characters. Two algorithms are used, classical Backpropagation (*BP*) and Optical Backpropagation (*OBP*), which applies a non-linear function on the error from each output unit before applying the backpropagation phase. The *OBP* aims to speed up the training process and escape from local minima. A comparison is made between results obtained from each algorithm. Several architectures are used with different learning rate values (Callan, 1999).

The paper is divided into four sections. In section 2, the proposed algorithm is presented. In section 3, an overall description of the developed system is given. In section 4, a comparative study between the *OBP* and *BP* is given. Results and conclusions are given after section 4.

Optical Backpropagation (OBP)

The backpropagation algorithm (as described in Appendix-A and its notations described in Appendix-B) encounters the following difficulty. When the actual value $f^o{}_k(net^o{}_{pk})$ (1) approaches either extreme value, the factor $f^o{}_k(net^o{}_{pk}) \bullet (1 - f^o{}_k(net^o{}_{pk}))$ in equation ***A.6 in*** Appendix-A makes the error signal very small. This implies that an output unit can be maximally wrong without producing a strong error signal with which the coupling strengths could be significantly adjusted. This retards the search for a minimum in the error. For instance, this occurs when some of the output units are pushed towards the wrong extreme value by competition in the network, thereby not increasing their error signal but instead decreasing it.

The proposed algorithms focus on this delay of the convergence is caused by the derivative of the activation function. Unfortunately, any saturating response function is bound to have this property: near the saturation points the derivative vanishes. This proposed algorithm shows, however, that a slightly modified error signal function of the backpropagation algorithm resolves this shortcoming and indeed greatly accelerates the convergence to a solution. This applies not only to the initial approach of the desired values, but also to the final convergence process when the response is already nears the target vector.

In this section, the adjustment of the new algorithm *OBP* will be described at which it would improve the performance of the *BP* algorithm. The convergence speed of the training process can be improved significantly by *OBP* through maximizing the error signal, which will be transmitted backward from the output layer to each unit in the intermediate layer.

In *BP*, the error at a single output unit is defined as:

$$\delta^o{}_{pk} = (Y_{pk} - O_{pk}) \bullet f^{o'}{}_k(net^o{}_{pk}) \qquad \textbf{(2.1)}$$

Where the subscript "*P*" refers to the p_{th} training vector, and "*K*" refers to the k_{th} output unit. In this case, Y_{pk} is the desired output value, and O_{pk} is the actual output from k_{th} unit, then $\delta^o{}_{pk}$ will propagate backward to update the output-layer weights and the hidden-layer weights.

While the error at a single output unit in adjusted *OBP* (Otair, & Salameh, 2004a, 2004b, 2005) will be as:

$$New\delta^o{}_{pk} = (1 + e^{(Y_{pk} - O_{pk})^2}) \bullet f^{o'}{}_k(net^o{}_{pk}) \qquad , \textit{if } (Y - O) >= \textit{zero}. \qquad \textbf{(2.2a)}$$

$$New\delta^o{}_{pk} = -(1 + e^{(Y_{pk} - O_{pk})^2}) \bullet f^{o'}{}_k(net^o{}_{pk}) \qquad , \textit{if } (Y - O) < \textit{zero}. \qquad \textbf{(2.2b)}$$

An *OBP* uses two forms of $New\delta^\circ{}_{pk}$, because the *exp* function always returns *zero* or *positive* values, while adapts operation for many output units need to decrease the actual outputs rather than increasing it. The $New\delta^\circ{}_{pk}$ will propagate backward to update the output-layer weights and the hidden-layer weights. This $New\delta^\circ{}_{pk}$ will minimize the errors of each output unit more quickly than the old $\delta^o{}_{pk}$, and the weights on certain units change very *large* from their starting values.

Notice that the equations in step 6, 7, 8, 9 change, but all other equations and steps will be as outlined in Appendix-A remain unchanged

The steps of an OBP:

1. Apply the input example to the input units.
2. Calculate the net-input values to the hidden layer units.
3. Calculate the outputs from the hidden layer.
4. Calculate the net-input values to the output layer units.
5. Calculate the outputs from the output units.
6. Calculate the error term for the output units, using the $New\delta^{\circ}_{pk}$ *(using equations (2.2a) (2.2b).)* instead of δ°_{pk}.
7. Calculate the error term for the hidden units, through applying $New\delta^{\circ}_{pk}$, also.

$$\text{New}\delta^{h}{}_{pj} = f^{h'}{}_{j}(\text{net}^{h}{}_{pj}) \bullet (\sum_{K=1}^{M} \text{New}\delta^{o}{}_{pk} \bullet W^{o}{}_{kj}) \qquad \textbf{(2.3)}$$

8. Update weights on the output layer.

$$W^{o}kj(t{+}1) = W^{o}kj(t) + (\eta \bullet New\delta^{o}pk \bullet i_{pj}) \qquad \textbf{(2.4)}$$

9. Update weights on the hidden layer.

$$W^{h}ji(t{+}1) = W^{h}ji(t) + (\eta \bullet New\delta^{h}pj \bullet X_{i}) \qquad \textbf{(2.5)}$$

10. Repeat steps from step **1** to step **9** until the error ($Y_{pk} - O_{pk}$) is acceptably small for each training vector pairs.

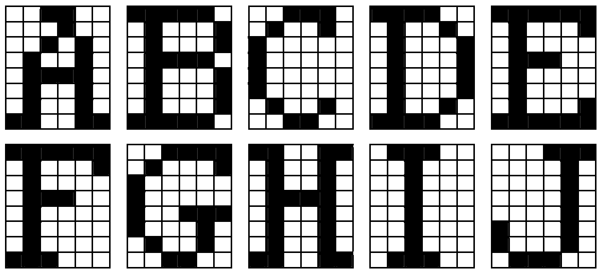
The proposed algorithm as classical BP is stopped when the squares of the differences between the actual and target values summed over the output units and all patterns are acceptably small.

Methodology

To study the performance of the *OBP* and *BP*, the neural networks are applied on handwritten English character recognition. Binary images of the characters are represented with bipolar values (-1, 1), and given to the input layer. As shown in Figure 1, two types of images are used in the training set an *8X6* binary image for the first *NN* and *12X8* for the second *NN*.

Experimental Parameters

Two three-layered neural networks are implemented, each with different architecture and learning parameters. Figure 1 shows the training set for each neural network. Although there exist numerous variants of the classical Backpropagation training algorithm (Rumelhart, Hinton, & Williams, 1986; Rumelhart, Durbin, Golden, & Chauvin, 1992), the comparison is made only between classical Backpropagation *BP* and Optical Backpropagation *OBP*.


(a) The training set for the 48x8x4 neural Network

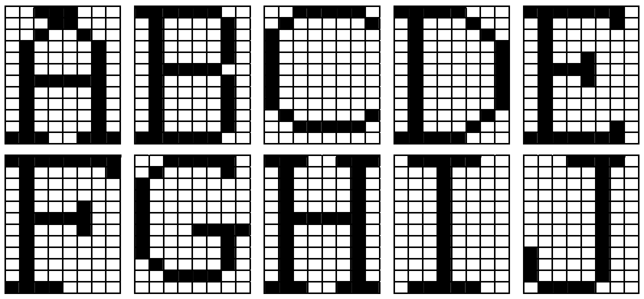

(b) The training set for the 96x16x4 Network

Figure 1: Training Set for Each Neural Network

The experiments were made on recognizing characters from (**A- J**). Characters are represented as N-vector which given to the input layer of each neural network. As shown in Figure 1 the number of pixels (number of units in input layer) of the first neural network is **48** (*8x6* matrix), while **96** (*12x8* matrix) for the second neural network. Black pixels are represented by 1's and white pixel by –1's (i.e. Bipolar input values were used). The activation function, which is used by each unit in the network, is the sigmoid function.

Binary vectors of size 4 represent the output values. As shown in Table 1, character *A* is represented with zero, while *J* is represented with 9.

Table 1: Desired output for each character

Binary Code	
0 0 0 0	A
0 0 0 1	B
. . .	. . .
1 0 0 1	J

As shown in Table 1, each character image is mapped to its corresponding binary code. To train the network for larger set of characters a large output vector can be used.

The size of the hidden layer for the first network is 8 producing *48x8x4* network while the second network has 16 in its hidden layer (therefore, *96x16x4* architecture).

Small values for the learning rates were used to avoid local minima, the value ranges from 0.1 to 0.4. In addition, several sets with different random initial weights between –0.5 to +0.5 were used in each training process with various values for the learning rate. The training processes were terminated when the *MSE* (Mean Square Error) is less than or equal to 0.0001.

Comparative Study

Training Processes for the 48x8x4 Neural Network

Table 2 shows training summary for the *48x8x4* neural network using the *OBP* and *BP* with different values of the learning rate

Table 2: Number of epochs for the 48x8x4 NN

Learning Rate	OBP	BP
0.1	359	53188
0.2	226	32222
0.3	213	18063
0.4	155	13354

As shown in Table 2, the decrease in the number of epochs required (in *OBP*) to achieve a MSE less than or equal to 0.0001 is noticeable.

Training Processes for the 96x16x4 Neural Network

Table 3 shows the training summary for the *96x16x4* neural network.

Table 3: Number of epochs for 96x16x4 epochs

Learning Rate	OBP	BP
0.1	83	4702
0.2	24	2479
0.3	28	1744
0.4	57	1191

Comparing the Training Process for the 48 x 8 x 4 Neural Networks with a Learning Rate =0.1

Table 4 show the convergence of error in training the *48x8x4* neural network with a learning rate=0.1. Using the BP the MSE was 0.00706 after 998 epochs while OBP only needed 75 epochs.

Table 4: Decrease in MSE for the 48x8x4 network using learning rate =0.1.

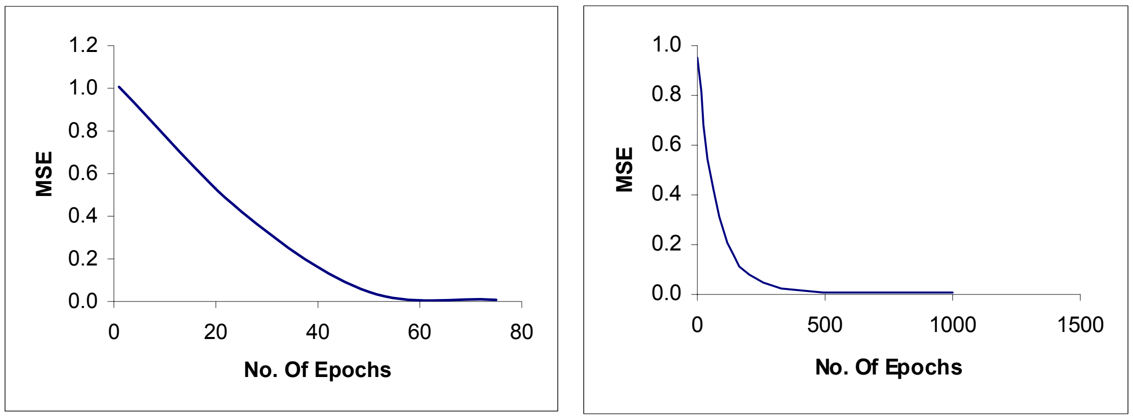
OBP		BP	
Epoch	**MSE**	**Epoch**	**MSE**
1	1.00705	1	0.95052
25	0.42370	64	0.42399
50	0.04571	259	0.04579
75	0.00706	998	0.00706

Part One

100	0.00267	2302	0.00268
125	0.00139	4203	0.00139
150	0.00084	6714	0.00084
175	0.00056	9852	0.00056
200	0.00040	13636	0.00040
225	0.00030	18056	0.00030
250	0.00023	23177	0.00023
275	0.00019	28893	0.00019
300	0.00015	35222	0.00015
325	0.00013	42357	0.00013
350	0.00011	50218	0.00011
359	0.00010	53188	0.00010

Part Two

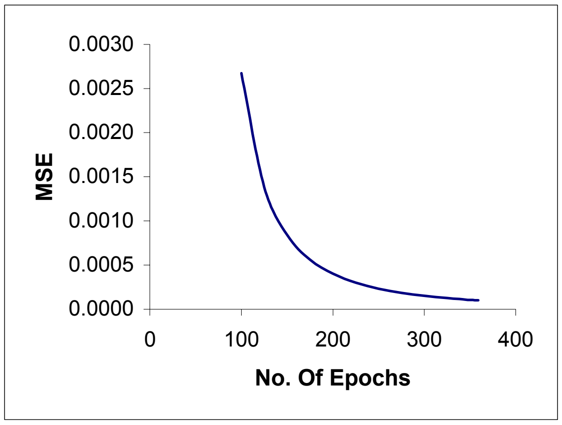
Figure 2 shows the MSE convergence using the OBP and BP, which represent the part one of table 4.



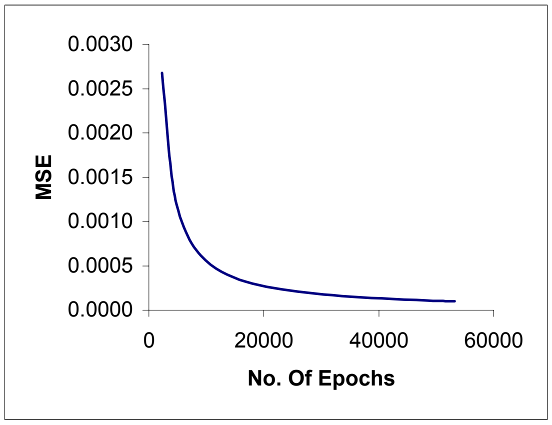
(a) (b)

Figure 2: (a) Decrease in MSE using the OBP to achieve MSE=0.00706. (b) Decrease in MSE using the BP to achieve MSE=0.00706.

In addition, Figure 3 shows the results obtained if training is completed to reach an error rate less than or equal to 0.0001 for the neural network as shown in part two of table 3.



(a)



(b)

Figure 3: (a) Decrease in MSE using the OBP from 0.00706 to 0.0001. (b) Decrease in MSE using OBP from 0.00706 to 0.0001.

Conclusions

This paper aims to apply the proposed OBP on one of the Backpropagation neural network applications, which is handwritten character recognition. To obtain fair and independent results, two different architectures were used each with various values for the learning rate.

The experimental results show that using the OBP can speed up convergence of the training process. Although small values for the learning rate were used, the OBP was able to train the neural network with much fewer numbers of epochs in compare with the classical BP.

Future Work

Future work will study the effect of using the momentum and adaptive learning rate on the performance of the proposed OBP algorithm.

Appendix-A

Freeman and Skapura's book (1992) describes the procedure of training feedforward neural networks using the backpropagation algorithm. The detailed formulas are described as follows:

Assume there are *m* input units, *n* hidden units, and *p* output units.

1. Apply the input vector, $X_p=(X_{p1}, X_{p2}, X_{p3}, \ldots, X_{pN})^t$ to the input units .
2. Calculate the net- input values to the hidden layer units:
$$net^h pj=(\sum_{i=1}^{N} W^h ji \bullet X_{pi}) \quad \textbf{(A.1)}$$
3. Calculate the outputs from the hidden layer:
$$i_{pj} = f^h j(net^h pj) \quad \textbf{(A.2)}$$
4. Move to the output layer. Calculate the net-input values to each unit:
$$\mathrm{net}^{o}\mathrm{pk}=(\sum_{j=1}^{L} \mathrm{W}^{o}\mathrm{kj}\bullet \mathrm{i}_{\mathrm{pj}}) \quad \textbf{(A.3)}$$
5. Calculate the outputs:

$$O_{pk} = f^{o}{}_{j}(net^{o}{}_{pk}) \qquad \textbf{(A.4)}$$

6. Calculate the error terms for the output units:

$$\delta^{o}{}_{pk} = (Y_{pk} - O_{pk}) \bullet f^{o'}{}_{k}(net^{o}{}_{pk}) \qquad \textbf{(A.5)}$$

$$\textit{Where, } f^{o'}{}_{k}(net^{o}{}_{pk}) = f^{o}{}_{k}(net^{o}{}_{pk}) \bullet (1 - f^{o}{}_{k}(net^{o}{}_{pk})) \qquad \textbf{(A.6)}$$

7. Calculate the error terms for the hidden units

$$\delta^{h}{}_{pj} = f^{h'}{}_{j}(net^{h}{}_{pj}) \bullet (\sum_{K=1}^{M} \delta^{o}{}_{pk} \bullet W^{o}{}_{kj}) \qquad \textbf{(A.7)}$$

Notice that the error terms on the hidden units are calculated *before* the connection weights to the output-layer units have been updated.

8. Update weights on the output layer

$$W^{o}{}_{kj}(t+1) = W^{o}{}_{kj}(t) + (\eta \bullet \delta^{o}{}_{pk} \bullet i_{pj}) \qquad \textbf{(A.8)}$$

9. Update weights on the Hidden layer

$$W^{h}{}_{ji}(t+1) = W^{h}{}_{ji}(t) + (\eta \bullet \delta^{h}{}_{pj} \bullet X_{i}) \qquad \textbf{(A.9)}$$

Appendix B

Notations

The notation described below were used throughout section *Appendix-A* and *section 2*

X_{pi}	net input to the *i*th input unit
$net^{h}{}_{pj}$	net input to the *j*th hidden unit
$W^{h}{}_{ji}$	weight on the connection from the *i*th input unit to *j*th hidden unit .
i_{pj}	net input to the *j*th hidden unit
$net^{o}{}_{pk}$	net input to the *k*th output unit
$W^{o}{}_{kj}$	weight on the connection from the *j*th hidden unit to *k*th output unit .
O_{pk}	actual output for the *k*th output unit .
Y_{pk}	desired output for the *k*th output unit .
f	(Sigmoid) activation function
f'	derivative of activation function
$\delta^{o}{}_{pk}$	signal error term for the *k*th output unit.
$\delta^{h}{}_{pj}$	signal error term for the *j*th hidden unit.
η	learning rate

References

Bishop, C.M. (1995). *Neural Networks for Pattern Recognition.* Oxford, UK: Oxford University.

Callan, R. (1999). *The Essence of Neural Networks.* Southampton Institute, pp. 33-52.

Freeman, J. A. &, Skapura, D. M. (1992). Backpropagation. *Neural Networks Algorithm Applications and Programming Techniques*, 89-125.

Okamoto, M. & Yamamoto, K. (1999). On-line handwriting recognition using direction-change features that consider imaginary strokes. *Pattern Recognition, 32*, 1115-1128.

Otair, M. A. & Salameh, W. A. (2004a). An improved back-propagation neural networks using a modified non-linear function. *Proceedings of the IASTED International Conference, 2004*, 442-447.

Otair, M. A. & Salameh, W. A. (2004b). Optical back-propagation neural networks with a momentum factor –A case study. *WSEAS Transactions on Computers, 9*(4), Nov. 2004.

Otair, M. A. & Salameh, W. A. (2005). Speeding up back-propagation neural networks. Manuscript submitted for publication.

Plamondon, R., Lopresti, D. P., Schomaker, L. R. B., & Shrihari, R. (1999). On-line handwriting recognition. *Wiley Encyclopedia of Electrical and Electronics Engineering, 15*, 123-146.

Rumelhart, D. E., Durbin, R. Golden, R., & Chauvin, Y. (1992). Backpropagation: Theoretical foundations. In Y.Chauvin & D. E Rumelhart (Eds.), *Backpropagation and connectionist theory*. Lawrence Erlbaum.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel Distributed Processing*, pp. 318-362.

Tappert, C. C., Suen, C. Y. & Wakahara, T. (1990). The state of art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12* (8), 787-808.

Wakahara, T., Murase, H., & Odaka, K.(1992). On-line handwriting recognition. *Proceedings of the IEEE, 80* (7), 1181-1194.

Biographies

Walid A. Salameh is an Associate Professor of computer Science, at the PSUT-RSS, Amman-Jordan. He received his B.Sc. in Computer Science from YU-Jordan, and his M.Sc. and Ph.D in 1987, 1991, respectively, from the Department of Computer Engineering-METU. His major interests are Machine Learning, Neural Network Learning Paradigms, and Intelligent Learning Paradigm through the Web (Intelligent Web-learning based Paradigms).

Mohammed A. Otair is an Instructor of computer information systems, at the Jordan University of Science and Technology, Irbed-Jordan. He received his B.Sc. in Computer Science from IU-Jordan and his M.Sc. and Ph.D in 2000, 2004, respectively, from the Department of Computer Information Sysems-Arab Academy. His major interests are Machine Learning, Neural Network Learning Paradigms, Web-computing, E-Learning.