Malaysian Journal of Computer Science, Vol. 11 No. 2, December 1998, pp. 68-73

A MODIFIED NEURAL NETWORK LEARNING APPROACH AND ITS APPLICATION TO BENGALI CHARACTER RECOGNITION

Md. Enamul Karim, Mahmood Hossain, Md. Abdul Mottalib and Tarequz Zaman
Department of Computer Science
University of Dhaka
Dhaka 1000
Bangladesh
Tel: 9661900-59 Ext-4900
*email: enam@du.bangla.net

ABSTRACT

Presents a new learning model for neural networks. It combines two previous modifications to the back propagation algorithm with a new scheme for pruning less contributory training cycles to achieve faster training. The total training time is reduced by predicting future weights at regular intervals from the nature of the previous weight changes. The oscillations among different patterns are reduced by updating the weights as a function of sum of errors of all input patterns. The predefined error level is adjusted by aborting the training session at an early stage when the weight changes become insignificant with respect to changes in iterations. This model was used to recognize Bengali alphabets and a significant reduction in training time was observed.

Keywords: Neural Network, Learning, Back propagation

1.0 INTRODUCTION

Among all neural networks learning techniques, the *back propagation* (BP) algorithm [1, 2, 3, 4] is the most generalized and popular one because of its application to a wide class of problems. Although it has the ability to learn any arbitrarily complex non-linear mapping because of hidden layer, it may involve extremely long training time. Solutions have been proposed to shorten its training time [5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. In spite of these efforts, researchers are still looking for more lucrative solutions since the learning time is not yet convenient for complex problems.

Hasanat and Sontag [13] proposed that the training time of the BP algorithm can be reduced by predicting the change of weights for individual patterns. Vogl et al. [14] proposed a method for reducing the oscillations among different patterns by changing all weights as a function of sum of all errors. In the BP algorithm, very small modifications of weights occur in the last training cycles where the weights go through an asymptotic change. This paper presents a new learning model which combines the ideas of the above mentioned papers [13, 14] and cuts off the less contributory asymptotic part of the training process. Bengali character recognition has been done by the BP or the *nearest neighbour* algorithm [15, 16]. A two-phase scheme has also been used [17] where nearest neighbour method is employed to classify symmetrical characters in common groups and characters in each group are classified using BP algorithm. But due to the complex and confusing nature of Bengali characters, training becomes extremely time consuming. The proposed learning model was applied for Bengali character recognition and a significant reduction in training time was observed.

2.0 THE BACK PROPAGATION TECHNIQUE

The back propagation method [1, 2, 3, 4] is a gradient descent method that establishes the weights in a multi-layer, feed-forward adaptive neural network (Fig. 1). Small arbitrary weights are chosen to initialize the system. Learning is accomplished by successively adjusting the weights based on a set of input patterns and the corresponding set of output patterns. During this iterative process, an input pattern is presented to the network and propagated forward to determine the resulting signal at the output nodes. The difference between the actual signal and the desired signal in each output node represents an error that is back propagated through the network in order to adjust the weights. The learning process continues until it converges to a minimum so that the sum of squares of the errors at the output nodes will be less than a predefined value for all patterns.

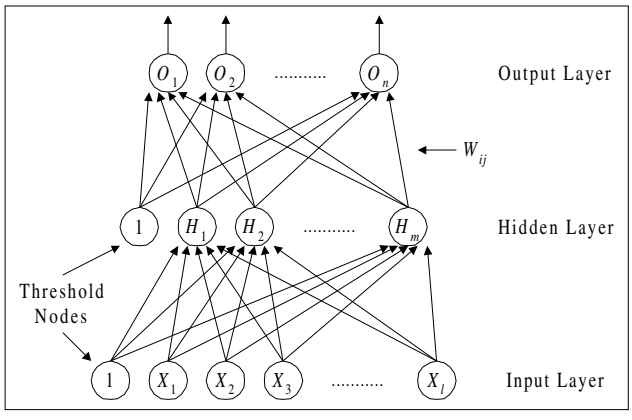


Fig. 1: A multi-layer network for back propagation

When an input pattern is applied to the network, the activation level O_j of each node j is dynamically determined by $O_j = 1/(1+\exp(-\sum_i O_i W_{ij}))$, where O_i is the activation of an input to node j, W_{ij} is the weight from node i to node j. Back propagation is then invoked to update all weights in the network according to the following rule:

$$W_{ij(t+1)} = W_{ij(t)} + \eta \delta_j O_j + \alpha (W_{ij(t)} - W_{ij(t-1)})$$

where t is the presentation number, η is the learning rate, α is a momentum factor, and δ_j is the error gradient of node j given by,

$\delta_j = O_j(1-O_j)(T_j - O_j)$ [for output nodes]
$\delta_j = O_j(1-O_j)\sum_i \delta_i W_{ji}$ [for hidden nodes]

where T_j is the desired activation level of node j.

3.0 THE NEW METHOD

3.1 Combining Two Earlier Modifications

Extrapolation of weights: Hasanat and Sontag [13] observed that the path of change of weights is somewhat predictable. So the weights can be pre-calculated and the network can try with that pre-calculated weight for an early convergence. If small segments on the curve of Fig. 2 are considered, it can be seen that they are almost linear. So for mathematical simplicity small linear jumps can be used by taking the weight changes in the last Z iterations as a measure of the weight changes in future Z iterations.

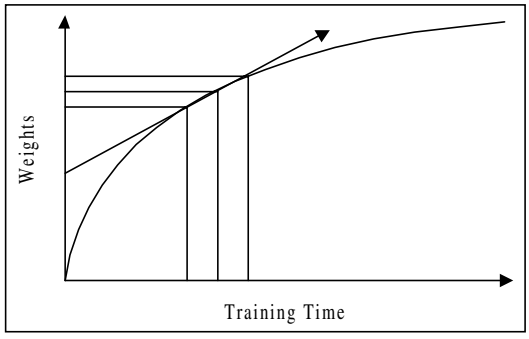


Fig. 2: Predicting future weights

Delayed modifications of weights: Vogl et al. [14] noticed that a step (a change of weights and activation based on the corrections produced by BP) that reduces the error with respect to one pattern will not, in general, produce a network with reduced error with respect to all other patterns which the system is to learn. Such a step may misdirect the optimization path and thus may considerably increase the number of iterations required for convergence. It was proposed to change weights as a function of sum of errors of all patterns. So rather than updating the network weights after each pattern, they are modified only after all input patterns are presented.

Each of the above two modifications individually reduces the training time to some extent but have some drawbacks. The first one reduces the training time for individual patterns but provides no solution to the oscillations between different training patterns, whereas the second one removes such oscillations greatly but offers no idea regarding the reduction of individual training phases. These two approaches can be combined to reduce the total training time as well as the oscillations among different patterns.

3.2 Early Aborting

In the BP algorithm, all weights, target outputs and acceptable range of errors are initialized arbitrarily. After an input pattern is presented, the consequent error vector across output nodes is determined and back propagated through the network to update the weights. The process is repeated for each input pattern until the sum of squares of the errors at the output nodes will be less than a predefined value for all patterns. It usually requires a large number of iterations. It can be observed that the weights get saturated, i.e., they become approximately equal to their final values long before convergence occurs. It can be observed from Fig. 3 that the weight changes follow an approximately logarithmic path which ends in a long asymptotic path. Experimental evidences show that this asymptotic portion has negligible contribution to divide feature space between individual patterns. This provokes the idea of eliminating this asymptotic region to achieve an early convergence. This can be performed by aborting from the weight adjustment process as soon as the process enters into the asymptotic region and by adjusting the targets according to the final weights.

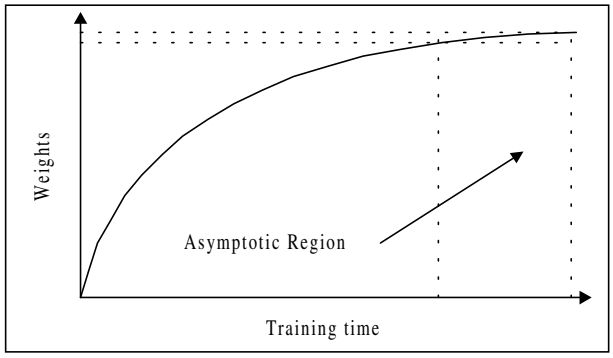


Fig. 3: Asymptotic Region

3.3 The Modified Algorithm

Keeping the above discussion in mind, a modified learning algorithm is proposed in this section which combines the two modifications [13, 14] mentioned in 3.1 and the scheme for early aborting mentioned in 3.2. The new algorithm has the following features:

- The weights are changed as a function of the sum of errors
- The nature of weight changes is observed at regular intervals to predict future weights
- When weights are changed asymptotically the error level is adjusted and the process is aborted

After each input pattern is presented, the activation of each node at hidden layer and output layer is calculated. But rather than updating the weights after each pattern, they are updated after all patterns are presented. The sum of errors of all patterns is used to update all weights. At regular intervals, the weight changes are observed. If they reach the asymptotic region, the predefined error level is adjusted and the training process is aborted. Otherwise, future weights are predicted from the nature of the previous weight changes.

Before presenting the modified technique, let us give the following definitions:

l, m, n	Number of nodes in input, hidden and output layers respectively
k	Number of input patterns
X_{pj}	Activation level of node j in the input layer for pattern p
H_{pj}	Activation level of node j in the hidden layer for pattern p
T_{pj}	Desired target activation level of node j in the output layer for pattern p
O_{pj}	Actual activation level of node j in the output layer for pattern p
E	The acceptable error level
t	Number of iterations
Z	Number of iterations after which predictions are made
$W_{1ij(t)}$	The weight from an input node i to a hidden node j at iteration t
$W_{2ij(t)}$	The weight from a hidden node i to an output node j at iteration t
$W_{a(t)}$	Average weight at iteration t
η	A trial-independent learning rate and $0<\eta<1$
δ_{1pj}	Error gradient of node j in the hidden layer for pattern p
δ_{2pj}	Error gradient of node j in the output layer for pattern p
α	A momentum term
ΔW	The rate of change of weights below which training process is aborted
$S(a)$	A sigmoid function given by $S(a) = 1/(1+e^{-a})$
$M(X)$	A maximum function over the elements of X
$R(x,y)$	A function generating a random number between x and y

The algorithm of the modified back propagation technique is presented below in Pseudo-PASCAL:

```
/*  Initialize weights to small random values, the weight
    from a threshold node is set to a negative value */
for j := 1 to m do
    W10j := R(-0.3, 0);
for j := 1 to n do
    W20j := R(-0.3, 0);
for i := 1 to l do
    for j := 1 to m do
        W1ij := R(-0.3, 0.3);
for i := 1 to m do
    for j := 1 to n do
        W2ij := R(-0.3, 0.3);
/*  For each input pattern p, present input vector
    Xp1Xp2…Xpl and target output vector Tp1Tp2…Tpn. For
    pattern association, Tpj is set to zero except for one that
    corresponds to the class of the input pattern */
for p := 1 to k do
    begin
        for j := 1 to l do
            input Xpj;
        for j := 1 to n do
            input Tpj;
        Xp0 := Hp0 := 1;
    end
/*  Weight adjustment */
while ( sum_{p=1}^{k} sum_{j=1}^{n} (Tpj - Opj)^2 > E or E is not changed) do
    begin
        /* Observe the nature of weight changes after
           each Z iterations */
        if t (mod Z) = 0 then
            begin
                /*  If weight changes are asymptotic,
                    adjust error level and abort,
                    otherwise predict future weights
                    according to previous changes */
                if (Wa(t) - Wa(t-1) / Wa(t)) < ΔW then
                    E := M({(Tpj-Opj)^2: 1≤p≤k, 1≤j≤n});
                else predict( );
            end
        else adjust_weights( );
    end

procedure predict( );
begin
    for i := 1 to m do
        for j := 1 to n do
            W2ij(t+1) := W2ij(t) + Wa(t) - Wa(t-Z);
    for i := 1 to l do
        for j := 1 to m do
            W1ij(t+1) := W1ij(t) + Wa(t) - Wa(t-Z);
end
```

procedure adjust_weights( );
begin

/* For each pattern calculate the activation of each node at hidden and output layers */

for p := 1 to k **do**
begin
for j := 1 to m **do**

$$H_{pj} := S(\sum_{i=0}^{l} X_{pi} W_{1ij}) ;$$

for j := 1 to n **do**

$$O_{pj} := S(\sum_{i=0}^{m} H_{pi} W_{2ij}) ;$$

end

/* For each input pattern calculate error gradients in output layer and hidden layer */

for p := 1 to k **do**
for j := 1 to n **do**

$$\delta_{2pj} := O_{pj}(1-O_{pj})(T_{pj}-O_{pj}) ;$$

for p := 1 to k **do**
for j := 1 to m **do**

$$\delta_{1pj} := H_{pj}(1-H_{pj})\sum_{i=1}^{n} \delta_{2pi} W_{2ji} ;$$

/* Start from output layer and work backward to hidden layer recursively to adjust weights as function of sum of errors for all patterns */

for i := 1 to m **do**
for j = 1 to n **do**

$$W_{2ij(t+1)} := W_{2ij(t)} + \eta \sum_{p=1}^{k} \delta_{2pj} H_{pj} + \alpha(W_{2ij(t)} - W_{2ij(t-1)}) ;$$

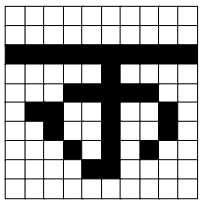
for i = 1 to l **do**
for j = 1 to m **do**

$$W_{1ij(t+1)} := W_{1ij(t)} + \eta \sum_{p=1}^{k} \delta_{1pj} H_{pj} + \alpha(W_{1ij(t)} - W_{1ij(t-1)}) ;$$

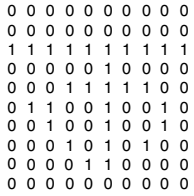
end

4.0 IMPLEMENTATION AND RESULTS

To test the modified algorithm, patterns of first ten Bengali consonants were taken as 10x10 bit matrices, as shown in Fig. 4 for the first Bengali consonant (pronounced "KAW"). Training was performed for an error value of 0.01 using (1) Standard BP, (2) BP with extrapolation of weights, (3) BP with delayed modifications of weights, (4) BP with the combined approach of extrapolation of weights and delayed modifications of weights, and (5) BP with the combined approach of extrapolation of weights, delayed modifications of weights and early aborting. The learning rate η was taken as 0.3. The momentum term α was taken as zero for the first few training cycles and then increased to 0.9 for the rest of the training [4]. To predict future weights and to determine whether the changes in weights are asymptotic, the process was checked after every 50 iterations i.e., Z was taken to be 50. The asymptotic region was considered when weights were changed at a rate less than 5% i.e., ΔW was taken as 0.05.



Graphical Representation



Binary Representation

Fig. 4: First Bengali consonant

The programs were developed in Borland C++ and run on a Pentium MMX 233 MHz machine. The number of iterations was measured for each approach. The results are presented in Table 1. The results imply that two previously existing methods individually reduce training time by about 31% and 47% respectively. If they are combined, they reduce the training time by about 58%. If this combined method is incorporated with the early abort method, the training time is reduced by about 72%. After completion of training, some sample patterns were applied. It was observed that within an acceptable range of error level, 89% unknown patterns were recognized.

Table 1: Experimental results for (A1) The standard BP, (A2) BP with extrapolation of weights, (A3) BP with delayed modifications of weights, (A4) BP with the combined approach of extrapolation of weights and delayed modifications of weights, and (A5) BP with the combined approach of extrapolation of weights, delayed modifications of weights and early aborting

Algorithm	A1	A2	A3	A4	A5
Number of Iterations	72025	49794	37885	30107	20383
% of Iterations wrt standard BP	100%	69%	53%	42%	28%

5.0 CONCLUSION

A modified back propagation algorithm was presented in this paper. This combines the ideas of two earlier modifications to the BP algorithm and a new technique to prune significant training cycles to achieve faster convergence. The new algorithm along with the original BP algorithm and its modifications were applied for Bengali character recognition and were found to outperform the previous ones. The new algorithm exhibited 72% reduction in training time as compared to the BP algorithm. Training time may be reduced further by employing conceptual clustering [18], [19] to divide the pattern space according to some features before applying the learning algorithm.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton and R. J. Williams. "Learning Internal Representations by Error Propagation". In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol 1, ed. D. E. Rumelhart, J. L. McClelland and the PDP Research Group, MIT Press, 1986, pp. 318-362.

[2] L. M. Fu. *Neural Networks in Computer Intelligence*. McGraw Hill, 1994.

[3] S. I. Gallant. *Neural Network Learning and Expert Systems*. MIT Press, 1993.

[4] E. Rich and K. Knight. *Artificial Intelligence*. 2nd ed. McGraw Hill, 1991.

[5] J. L. McClelland and D. E. Rumelhart. *Explorations in Parallel Distributed Processing*. MIT Press, 1988.

[6] R. A. Jacobs. "Increased Rates of Convergence Through Learning Rate Adaptation". *Neural Networks*, Vol. 1, No. 4, 1988, pp. 295-307.

[7] S. E. Fahlman. "Faster-Learning Variations on Back Propagation: An Empirical study". In *Proceedings of the 1988 Connectionist Models Summer School*, San Mateo, CA, 1988, pp. 38-51.

[8] W. S. Stornetta and B. A. Huberman. "An Improved Three-layer Back Propagation Algorithm". In *Proceedings of the IEEE First International Conference on Neural Networks*, Vol. 2, San Diego, CA, 1987, pp. 637-643.

[9] M. A. Mottalib and M. E. Karim. "Accelerating the Convergence of the Back Propagation Method". In *Proceedings of Fourth International Conference on Control, Automation, Robotics and Vision*, Vol. 2, Singapore, 1996, pp. 1540-1544.

[10] V. Ooyen and B. Nienhuis. "Improving the Convergence of the Back Propagation Algorithm". *Neural Networks*, Vol. 5, 1992, pp. 465-471.

[11] G. Parlos, B. Fernandez, A. F. Atiya, J. Muthusami and W. K. Tsai. "An Accelerated Learning Algorithm for Multi-Layer Perceptron Networks". *IEEE Transactions on Neural Networks*, Vol. 5, No. 3, 1994, pp. 493-497.

[12] S. Ergezinger and E. Thomsen. "An Accelerated Learning Algorithm for Multi Layer Perceptrons: Optimizing Layer by Layer". *IEEE Transactions on Neural Networks*, Vol. 6, No. 1, 1994, pp. 31-42.

[13] D. M. Hasanat and E. D. Sontag. "Exploratory Methods for Speeding up the BP Algorithm". In *Proceedings of International Joint Conference on Neural Networks*, Washington, DC, 1990, pp. 613-616.

[14] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink and D. L. Alkon. "Accelerating the Convergence of the Back Propagation Method". *Biological Cybernetics*, Vol. 59, 1988, pp. 257-263.

[15] A. K. Roy and B. Chatterjee. "Design of Nearest Neighbour Classifier for Bengali Character Recognition". *J. IETE*, Vol. 30, 1984.

[16] A. Dutta and S. Chaudhury. "Bengali Alpha-Numeric Character Recognition Using Curvature Features". *Pattern Recognition*, Vol. 26, No. 12, 1993, pp. 1757-1770.

[17] M. A. Mottalib and M. E. Karim. "Combined Nearest Neighbour-Back Propagation Approach to Recognize Bengali Vowels". *Dhaka University Journal of Science*, Vol. 46, No. 2, 1998.

[18] R. S. Michalski and R. E. Stepp. "Learning From Observation: Conceptual Clustering". In *Machine Learning*: *An Artificial Intelligence Approach*, Vol 1, ed. R. S. Michalski, J. G. Carbonell and T. M. Mitchell, Morgan Kaufmann, 1983, pp. 331-363.

[19] D. H. Fisher. "Knowledge Acquisition via Incremental Conceptual Clustering". *Machine Learning*, Vol. 2, 1987, pp. 139-172.

BIOGRAPHY

Md. Enamul Karim is a lecturer in the Department of Computer Science, University of Dhaka, Bangladesh. He obtained his M.Sc. in Computer Science from University of Dhaka in 1996. He has a number of research papers to his credit. His research interest includes Neural Networks, Artificial Intelligence, and Algorithm Analysis.

Mahmood Hossain is a lecturer in the Department of Computer Science, University of Dhaka. He obtained his M.Sc. in Computer Science from University of Dhaka in 1995. His research interest includes Artificial Intelligence, Database Systems.

Md. Abdul Mottalib is a professor in the Department of Computer Science, University of Dhaka. He obtained his M.Sc. in Applied Physics and Electronics from University of Dhaka in 1980, MS in Computer Science from Asian Institute of Technology, Thailand, and Ph.D. in Computer Science and Engineering from Indian Institute of Technology, Kharagpur, India. He has numerous research papers in Bangladeshi and internationally reputed journals to his credit. His research interest includes VLSI Design and Testing, Logic Minimization, Fault Tolerant Systems. Recently, he has been working on Artificial Intelligence.

Tarequz Zaman obtained his M.Sc. in Computer Science from University of Dhaka in 1998.