Paper

Improving Convergence of Backpropagation Learning using Exponential Cost Function

Joarder Kamruzzaman* Non-member

Backpropagation, one of the most popular learning algorithms in multi-layered feedforward neural networks, suffers from the drawback of slow convergence. Several modifications have been proposed to accelerate the learning process using different techniques. In this paper, a new cost function expressed as exponential of sum-squared or Log-likelihood is proposed. Weight update using this modification varies the learning rate parameter dynamically during training as opposed to constant learning rate parameter used in standard Backpropagation. Simulation results with different problems demonstrate significant improvement in the learning speed of Backpropagation algorithm.

Keywords: neural networks, multilayer feedforward network, gradient descent, convergence, training cycle.

1. Introduction

Since the resurgence of neural network, Backpropagation [1] has become one of the most widely used learning algorithms to train multilayer feedforward network and has demonstrated potential applications in character recognition, image processing, intelligent control, prediction etc. The algorithm uses gradient descent technique to adjust the connection weights between neurons. One of the major drawbacks of Backpropagation is its slow convergence. New applications of neural networks in different areas like data mining, knowledge discovery, and intelligent agents etc. demand faster convergence. Different modifications of standard Backpropagation algorithm have been proposed to improve convergence [2]–[11]. Tawel [2] proposed an adaptive neural net by introducing the temperature of the sigmoid activation function. To train the network both the weights and temperatures are updated. Tariq [3] applied expected values of the net input to hidden and output layer neurons to determine the weight change. Holt [4] proposed a Log-likelihood cost function as an alternate to sum-squared cost function and reported faster convergence. "Resilient Backpropagation" (Rprop) [5] uses the sign of the derivative of error surface with respect to weight to indicate the direction of weight update. Leung *et al.* [6] proposed an adaptive algorithm by combining the adaptive neuron model [2] and Kalman filter. Verma [7] adopted a method that uses an inverse transformation for linearization of nonlinear output activation function, and then employs direct solution method to determine output layer weights and gradient descent to train hidden layer weights. Thus the weights in each layer are determined separately. The learning algorithm requires assigning codes to the hidden units and also identifying the hidden unit with Minimum Bit Distance (MBD) by computing the similarity between the input vector and the weight vector at the hidden layer after presentation of each input during training. S.C. Ng *et al.* [8] proposed a generalized Backpropagation algorithm that increases the back propagated error signals to improve learning speed and avoid local minima. Sureerattanan *et al.* [9] proposed a method combining conjugate gradient and Kalman filter. A new logarithmic activate function is proposed by Bilski [10] in lieu of sigmoid activation function commonly used in Backpropagation. Most of the proposed modifications make the learning algorithm too complex to be used by researchers who use neural networks as an application tool. In this paper, a different cost function is proposed to achieve accelerated convergence: exponential of sum-squared or Log-likelihood cost function. Simulations with different problems have been carried out to investigate the learning characteristic with this modification.

Section 2 briefly describes the weight update equations with sum-squared and Log-likelihood cost function. Section 3 shows the proposed modification. Section 4 and 5 present the simulation results and concluding remarks, respectively.

2. Cost Function in Standard BP Algorithm

Backpropagation updates the weights iteratively to map a set of input vectors ($\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_p}$) to a set of corresponding output vectors ($\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_p}$). A Backpropagation network is shown in Fig. 1. All the interconnecting weights between the layers are initialized to small random values at the beginning. The input is presented to the network and multiplied by the weights. All the weighted inputs to each unit of upper layer are summed up, and produce output governed by the following equations.

* Faculty of Information Technology, Monash University, Australia.

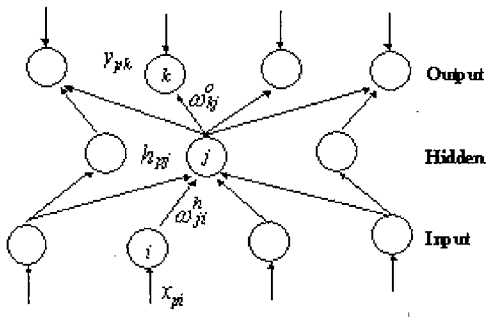


Fig. 1. A Backpropagation network.

$$y_{pk} = f(\sum_j \omega^o_{kj} h_{pj} + \theta^o_k) \cdots\cdots (1)$$

$$h_{pj} = f(\sum_i \omega^h_{ji} x_{pi} + \theta^h_j) \cdots\cdots (2)$$

where h_{pj} and y_{pk} are the outputs of hidden unit 'j' and output unit 'k', respectively. ω's are the connecting weights between units and θ's are the threshold of the units, and $f(.)$ is the sigmoid activation function.

The cost function to be minimized in standard Backpropagation is the sum of squared error defined as

$$E = \sum_p E_p = \frac{1}{2}\sum_p \sum_k (t_{pk} - y_{pk})^2 \cdots\cdots (3)$$

where t_{pk} is the target output at unit 'k' for pattern 'p'.

The error surface with respect to weights can have various complex forms. Formulation of Backpropagation assumes that the reduction of each E_p will reduce the total error E. Therefore, the change in ω for pattern 'p' is given by

$$\Delta\omega \propto -\frac{\partial E_p}{\partial \omega} \cdots\cdots (4)$$

which gives

$$\Delta_p \omega^o_{kj} = \eta\,(t_{pk} - y_{pk})\, y_{pk}\,(1 - y_{pk})\, h_{pj} \cdots\cdots (5)$$

$$\Delta_p \omega^h_{ji} = \eta h_{pj}\,(1 - h_{pj})\, x_{pi} \times \sum_k (t_{pk} - y_{pk})\, y_{pk}\,(1 - y_{pk})\, \omega^o_{kj} \cdots (6)$$

The term $y_{pk}(1 - y_{pk})$ in (5) and (6) is a bell shaped function having a maximum value of 0.25. When the actual output y_{pk} approaches to either of the extreme values, namely 0 or 1, the value of this term diminishes. This will produce very small back propagated error signal resulting in very small weight change. Thus, the output can be maximally wrong without producing any significant weight change. The algorithm may then be trapped into local minima. Consequently, the learning process and weight adjustment will be very slow or even suppressed[8].

Holt[4] proposed a Log-likelihood (LL) cost function as an alternate to sum of squared error which effectively eliminates the term $y_{pk}(1 - y_{pk})$ in weight update equation. This cost function is defined as

$$E = -\sum_p \sum_k \{t_{pk} \ln y_{pk} + (1 - t_{pk}) \ln(1 - y_{pk})\} \cdots\cdots (7)$$

The use of Log-likelihood cost function makes the early stage of learning considerably faster[4]. However, convergence speed does not show significant improvement at the final stage of learning.

3. Proposed Cost Function

If the learning rate parameter is made to vary as a function of error measure during the learning process instead of keeping fixed as in standard Backpropagation, the overall learning speed can be expected to accelerate. The following cost function is proposed to achieve faster learning[11].

$$E^t = \sum_p E^t_p = \sum_p e^{\beta E_p} \cdots\cdots (8)$$

where E^t is the total error and E^t_p is the measure of error at the output layer for pattern 'p'.

Accordingly the weight update in (5) changes to

$$\Delta_p \omega^o_{kj} \propto -\beta e^{\beta E_p}(t_{pk} - y_{pk}) y_{pk}(1 - y_{pk}) h_{pj} \cdots (9)$$

An observation of (9) shows that the term $e^{\beta E_p}$ needs to be calculated each time an input is presented. Calculation of this term involves calculation of E_p for every weight update which requires several multiplications and exponential computations. The computational overload using the proposed cost function can be reduced if the term is replaced by $e^{\beta E}$, where E is defined in (3). Then this term needs to be calculated only once for each training cycle and can be used for presentation of each input. The weight change becomes

$$\begin{aligned}\Delta_p \omega^o_{kj} &= \eta\beta e^{\beta E}(t_{pk} - y_{pk}) y_{pk}(1 - y_{pk}) h_{pj} \\ &= \eta_{eff}(t_{pk} - y_{pk}) y_{pk}(1 - y_{pk}) h_{pj} \cdots\cdots (10)\end{aligned}$$

where

$$\eta_{eff} = \eta\beta e^{\beta E}, \quad \text{and}$$

$$\Delta_p \omega^h_{ji} = \eta_{eff} h_{pj}(1 - h_{pj}) x_{pi} \times \sum_k (t_{pk} - y_{pk}) y_{pk}(1 - y_{pk}) \omega^o_{kj} \cdots (11)$$

Comparison of (10) and (5) shows that the effective learning rate in the proposed modification is $\eta\beta e^{\beta E}$. Thus this modification adapts the learning rate during training as opposed to standard Backpropagation where the learning rate is kept constant. In cases where the actual output is very close to the extreme value (0 or 1) and is maximally wrong, standard Backpropagation algorithm may be trapped into local minima because of very small change in weight. This modification effectively increases the weight change in such cases and may help to avoid trapping into local minima.

Modification proposed by Holt[4] using Log-likelihood cost function eliminates the term $y_{pk}(1 - y_{pk})$ in the output layer weight update equation. Similarly weights in the proposed modification with Log-likelihood measure of E are updated according to the following equations.

$$\Delta_p \omega^o_{kj} = \eta_{eff}(t_{pk} - y_{pk})h_{pj} \cdots\cdots\cdots\cdots\cdots \quad (12)$$

$$\Delta_p \omega^h_{ji} = \eta_{eff} h_{pj}(1 - h_{pj}) x_{pi} \sum_k (t_{pk} - y_{pk}) \omega^o_{kj} \quad (13)$$

where

$$\eta_{eff} = \eta \beta e^{\beta E},$$

E being the Log-likelihood cost function evaluated as in (7).

4. Simulation Results

In order to investigate the convergence behavior and speed of the proposed modification, simulations are carried out with different types of problems. Since no analytical techniques are available to study the learning speed of Backpropagation algorithm, simulation with different problems and comparison with standard Backpropagation is the usually adopted means to evaluate the effectiveness of a modification. In the present work, investigation has been done with three different problems. The first one is the XOR problem which is treated as a benchmark problem in the neural network literature. The second is a character recognition problem chosen for neural network's suitability in character recognition applications. The third is a function approximation problem which uses analog values for both input and output. Although the modifications use different cost functions as a measure of error at the output layer, the same stopping criterion was used for all algorithms in order to have a comparison at equal and fair basis. In all cases learning was stopped when sum of squared error as computed by (3) reduced to a predefined value.

4.1 XOR Problem A 2-2-1 network (two inputs, two hidden units and one output unit) network and 2-3-1 network were trained using both standard Backpropagation and the proposed modification. Investigation was done with different values of η and β (in case of modification only), and training was continued till sum of squared error measured at the output layer reduced to 0.0005. Total training cycles required for learning is also dependent on the initial weights chosen. For this reason, 25 trials with different initial weights were carried out and the average training cycles required for different values of η and β are presented in Table 1.

Simulation results with XOR show that the use of the proposed cost function greatly accelerates the convergence speed with suitable value of β. In case of XOR, this suitable value of β lies within the range of 1.0~4.0. Larger values of β tend to cause oscillation in the learning process. Learning behavior is illustrated in Fig. 2. It shows that error starts to reduce drastically after it has reached nearly 0.5 in both standard Backpropagation and modification. But in case of modification, it starts after 200 cycles whereas in standard Backpropagation after 500 cycles. Similar trend is observed for other values of η and β also. Fig. 2 also shows how the effective

Table 1. Comparison of learning speed in terms of training epochs between standard BP and the proposed modification to reduce sum of squared error as computed by (3) to 0.0005 in case of XOR problem with a 2-2-1 network.

Learning	Standard	Proposed modification in standard BP			
rate, η	BP	β=1.00	β=2.00	β=3.00	β=4.00
0.20	28668	27922	13702	9080	6758
0.40	14312	13948	6858	4555	3503
0.60	9535	9297	4581	3091	2905

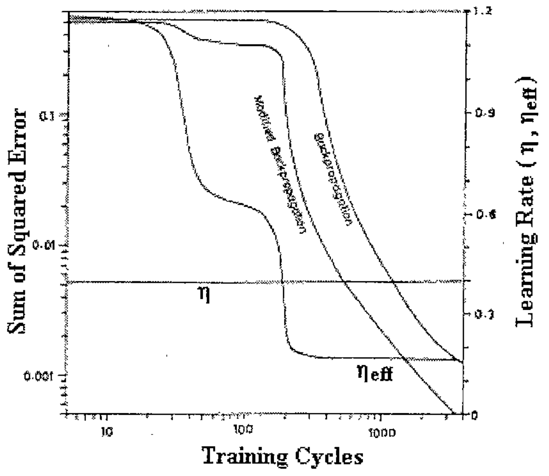


Fig. 2. Learning characteristic and the variation of η_{eff} with error in XOR problem in a 2-2-1 network with η=0.4, β=0.4.

learning rate η_{eff} varies during training. The adaptability of learning rate with changing error results in faster learning. Starting from higher value η_{eff} gradually reduces to lower value. This is quite justified because at the latter stage of learning when error is of small value, an inappropriately large step size can retard the learning process and may even lead to oscillation. By varying the learning rate as required, the training is significantly accelerated.

The problem was also studied with Log-likelihood cost function proposed by Halt[(4)] and the modification proposed in (12) and (13). Although the algorithm[(4)] minimizes Log-likelihood error defined by (7) and updates the weights accordingly, a comparison of learning speed was made on the basis of sum of squared error. Simulation results are shown in Table 2 for different values of η and β. Range of β that yields accelerated convergence is found roughly within 1.0~2.0. Larger values of β caused oscillation in learning. This range of β=1.0~2.0 is smaller than the comparison presented earlier in Table 1. The reason is that, in this case, the absence of the term $y_{pk}(1-y_{pk})$ in (12) and (13) further increases the effective step size. In this case also, the proposed modification shows significant acceleration in convergence speed.

4.2 Character Recognition Problem A Backpropagation network was trained to learn ten numeric digits (0~9). Each digit was an 8×8 pixel. Each black pixel was represented by '1' and white pixel by '0'. Target signal at the output layer was a locally represented one, i.e., only one output unit at the output layer was

Table 2. Comparison of training cycles required to learn XOR problem with a 2-2-1 network using standard BP with Log-likelihood cost function and the proposed modification.

Learning rate, η	Backpropagation(LL)	Proposed modification using (12) and (13)			
		β=1.00	β=1.50	β=1.75	β=2.00
0.20	1039	882	551	462	396
0.40	515	442	282	239	205
0.60	345	298	194	179	164

Table 3. Comparison of learning speed between standard BP and the proposed modification to reduce sum of squared error to 0.001 to train ten numeric digits (0~9) with a 64-5-10 network.

Learning rate, η	Standard BP	Proposed modification in standard BP			
		β=1.00	β=1.50	β=2.00	β=2.50
0.20	8546	8011	5006	4320	3670
0.40	4285	3916	2520	1965	2915
0.60	2788	2676	1815	1683	2133

Table 4. Comparison of training cycles required to realize character recognition problem with a 64-5-10 network using standard BP with Log-likelihood cost function and the proposed modification.

Learning rate, η	Backpropagation(LL)	Proposed modification using (12) and (13)			
		β=1.00	β=1.25	β=1.50	β=2.00
0.05	2602	2466	1856	1421	1252
0.10	1241	1230	1025	764	707
0.20	584	559	447	373	349

active and the rest were zero. The network had 64 inputs and ten output units to represent ten categories. A 64-5-10 and a 64-6-10 network were trained using standard Backpropagation and the proposed modification. The networks were trained with different values of η and β. For each set of η and β, 25 trials were made. Simulation results showing the average training cycles required to reduce sum of squared error to 0.001 are shown in Table 3 and Fig. 3. The results show that learning is considerably faster when modified Backpropagation is used with the value of β within the range of 1.0~2.5. With larger values of β, oscillation occurs in the learning process.

As in the XOR problem, the character recognition problem was also investigated using Log-likelihood cost function and the modification suggested in (12) and (13). Simulation results for different values of η and β are summarized in Table 4. In this case also, for β roughly within the range of 1.0~2.0, the modification accelerates the convergence speed considerably.

4.3 Function Approximation The problem chosen here is to approximate the sine function sin (x), for -5.0 $< x <$ 5.0 radian. A 1-6-1 and a 1-7-1 network were trained with values of sin (x) taken at the interval of 0.1 radian. In this case, the activation of the output unit is linear. Input and target values are the same. A total of 101 input-output values were trained by the network. At sum-squared error of 0.00005, the network approximated the sine function. Simulation result shows that accelerated convergence is achieved by modified Backpropagation for values of β roughly within

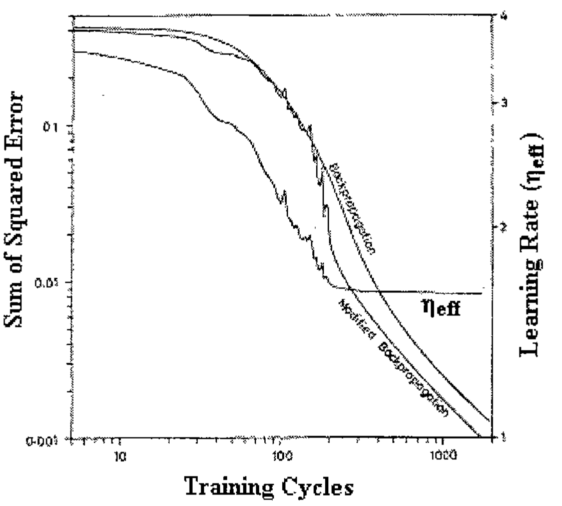


Fig. 3. Learning characteristic and variation of η_{eff} during training in case of character recognition problem with 5 hidden units and η=0.4, β=0.2.

the range of 0.25~1.15 and η within 0.1~2.0. With η=0.20, standard Backpropagation shows severe oscillation in learning process whereas proposed modification yields gradual decrement of error. Modification takes only half of the number of training cycles required by standard Backpropagation.

5. Conclusion

In this paper, a modification of cost function used in standard and Log-likelihood Backpropagation algorithms is proposed. The modification adapts the learning rate according to the measured error during training. Simulation with different problems shows that the modification significantly accelerates the convergence speed. The modification requires an additional parameter β. In our experiments significant improvement of learning speed was achieved when the value of β was set higher than 1.0. However, like η in standard Backpropagation, suitable value of β is also problem specific. In applications users usually have to train a number of networks to select the best one that yields the highest generalization ability. A modification for accelerated learning like the one presented here will be useful in this case.

(Manuscript received December 7, 2001, revised September 17, 2002)

References

(1) D.E Rumelhart, J.L. McClelland, and the PDP research group: *"Parallel Distributed Processing"*, Vol.1, MIT Press, (1986)
(2) R. Tawl: "Does the neuron learn like synapse?", *Advances in Neural Information Processing System1*, ed. D. Touretzy, Morgan-Kaufmann, pp.169-176 (1989)
(3) S. Tariq: "Backpropagation with expected source values", *Neural Networks*, Vol.4, pp.615-618 (1991)
(4) M.J. Holt: "Comparison of generalization in multiplayer perceptron with log-likelihood and least-squares cost function", *Proc. 11th IAPR, Int. Conf. on pattern Recognition*, The Netherlands, **II**, pp.17-20 (1992)
(5) M. Reidmiller and H. Braun: "A direct adaptive method for

faster Backpropagation learning: The EPROP algorithm", *Proc. IEEE Int. Conf. on Neural Networks*, San Francisco, CA, pp. 586-591 (1993)

(6) C.S. Leung and S. Park: "A fast algorithm for training neural networks", *Proc. IASTED Int. Conf. on Modelling and Simulation*, PA, USA, pp.616-619 (1993)

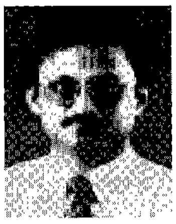
(7) B. Verma: "Fast training of multilayered perceptrons", *IEEE Trans. on Neural Networks*, Vol.**8**, No.**6**, pp.1314-1320 (1997)

(8) S.C. Ng, S.H. Leung, and A. Luk: "Fast convergent generalized back-propagation algorithm with constant learning rate", *Neural Processing Letters*, Vol.**9**, pp.13-23 (1999)

(9) S. Sureerattanan and H.N. Phien: "New developments on backpropagation network training", *IEICE Trans. of Fundamentals of Electronics, Commun. and Computer Science*, Vol.**E83-A**, No.**6**, (2000)

(10) J. Bilski: "The Backpropagation learning with logarithmic transfer function", *Proc. 5th Conf. on Neural Networks and Soft Computing*, Poland, pp.71-76 (2000)

(11) J. Kamruzzaman: "Fast training of multilayered feedforward neural networks", *Proc. 3rd Japan- Australia Joint Workshop on Intelligent and Evolutionary Systems*, Australia, pp.78-85 (1999)



Joarder kamruzzaman (Non-member) received a B.Sc and M.Sc. in electrical engineering from Bangladesh University of Engineering & Technology, Dhaka, Bangladesh in 1986 & 1989 respectively, and a PhD in Information System Engineering from Muroran Institute of technology,Japan in 1993. Currently he is a faculty member in the Faculty of Information Technology, Monash University, Australia. His research interest includes neural networks, fuzzy system, genetic algorithm etc.