International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:1, 2007

Improving the Convergence of the Backpropagation Algorithm Using Local Adaptive Techniques

Z. Zainuddin, N. Mahat, and Y. Abu Hassan

***Abstract*—**Since the presentation of the backpropagation algorithm, a vast variety of improvements of the technique for training a feed forward neural networks have been proposed. This article focuses on two classes of acceleration techniques, one is known as Local Adaptive Techniques that are based on weight-specific only, such as the temporal behavior of the partial derivative of the current weight. The other, known as Dynamic Adaptation Methods, which dynamically adapts the momentum factors, α, and learning rate, η, with respect to the iteration number or gradient. Some of most popular learning algorithms are described. These techniques have been implemented and tested on several problems and measured in terms of gradient and error function evaluation, and percentage of success. Numerical evidence shows that these techniques improve the convergence of the Backpropagation algorithm.

***Keywords*—** Backpropagation, Dynamic Adaptation Methods, Local Adaptive Techniques, Neural networks.

I. Introduction

THE most popular Artificial Neural Networks (ANN) architectures is called multilayer perceptrons (MLP) because of its similarity to perceptron networks with more than one layer. The MLP refer to the network consisting of a set of sensory units (source nodes) that constitute the input layer, one or more hidden layers of computation nodes, and an output layer of computation nodes. Nodes or neurons in any layer of the network, is connected to all the neurons in the previous layer. The input signal propagates through the network in a forward direction, from left to right and on a layer-by-layer basis. The Back-Propagation is the best known and widely used learning algorithm in training multilayer perceptrons.

Manuscript received December 6, 2004.

Zarita Zainuddin is with the School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Pulau Pinang, Malaysia. (Phone: +60 4 653940; Fax: +60 4 6570910; e-mail: zarita@cs.usm.my).

Norpah Mahat, was with the School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Pulau Pinang, Malaysia. (e-mail: norpahmahat@yahoo.co.uk).

Yahya Abu Hassan is with the the School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Pulau Pinang, Malaysia. (e-mail: ahyahya@cs.usm.my).

II. Backpropagation Algorithm

The error signal, $e_j(n)$ at the output of neuron j at iteration n is defined by

$$e_j(n) = d_j(n) - y_j(n) \tag{1}$$

where $d_j(n)$ refers to the desired response for neuron j and is used to compute $e_j(n)$ and

$y_j(n)$ refers to the function signal appearing at the output of neuron j at iteration n.

The objective of back-propagation algorithm is to minimize $e_j(n)$ so that the desired response will be close to the actual response.

We define the instantaneous value of the error energy for neuron j as $\frac{1}{2}e^2{}_j(n)$. Correspondingly, the instantaneous value $\xi(n)$ of the total error energy is obtained by summing $\frac{1}{2}e^2{}_j(n)$ over all neurons in the output layer. We may thus write

$$\xi(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \tag{2}$$

where the set C includes all the neurons in the output layer of the network. Let N denote the total number of patterns (examples) contained in the training set. The average squared error energy is obtained by summing $\xi(n)$ over all n and then normalizing with respect to the set size N, as shown by

$$\xi_{av} = \frac{1}{N}\sum_{n=1}^{N}\xi(n) \tag{3}$$

The objective of the learning process is to adjust the free parameters (i.e. synaptic weights and bias levels) of the network to minimize ξ_{av}. To do this minimization, the weights are updated on a pattern-by pattern basis until one epoch, that is, one complete presentation of the entire training set. The arithmetic average of these individual weight changes over the training set is therefore an estimate of the true change as would result from modifying the weights based on minimizing the cost function ξ_{av} over the entire training set. In its most basic form, it is a simple gradient optimization procedure:

$$w_{ji}(n+1) = w_{ji}(n) - \eta\partial\xi / \partial w_{ji} \tag{4}$$

In the batch mode variant the descent is based on the gradient $\Delta\xi$ for the total training set :

$$\Delta w_{ji}(n) = -\eta * \frac{\partial \xi}{\partial w_{ji}} + \alpha * \Delta w_{ji}(n-1) \qquad (5)$$

where ξ is the cost function being minimized, w_{ji} is a generic weight in the network, α is a momentum factor and η is the learning rate or step size parameter.

III. Local Adaptive Techniques

Many techniques have been proposed to date to deal with problems of gradient descent. These techniques can be roughly divided into two categories, Global and Local adaptive techniques. Global techniques are algorithms that use global knowledge of the state of the entire network, such as the direction of the overall weight-update vector. For example, a class of global algorithms, are Steepest Descent and Conjugate Gradient (CG) methods. The CG methods include Fletcher Reeves, Powell Beale and Polak Ribiere method.

Local adaptation strategies are based on weight specific information only, which means that they use an independent learning rate for every adjustable parameter (every connection) [1]. Therefore they are able to find an optimal learning rate for every weight. Some of the local adaptive techniques will discussed below.

A. Sign Changes

Learning rate adaptation by sign changes will adapt the step size using a separate learning rate, η_{ji} for each connection [2]. The adaptation is done by observing the signs of the last two gradients. As long as no change in sign is detected, the corresponding learning rate is increased. If the sign changes, the learning rate is decreased.

If, in two successive iterations, the updates of x (or equivalently, the gradient values) have opposite signs, that means that we have "jumped over a minimum" and that the step size is too large. On the other hand, if two successive signs are equal, it appears that we could have moved somewhat faster, while still not passing beyond the minimum. The basic heuristic for adaptation is then simply to decrease the step size of two successive updates that have opposite signs, and to increase it if they have the same sign. It was proposed to use a linear step size increase, and an exponential step size decrease. The step size update is according to the following:

$$\begin{aligned} &\eta_{ji}(n) = \eta_{ji}(n-1) * u, && \text{if } \frac{\partial \xi}{\partial w_{ji}}(n) * \frac{\partial \xi}{\partial w_{ji}}(n-1) \geq 0 \\ &\eta_{ji}(n) = \eta_{ji}(n-1) * d, && \text{else.} \end{aligned} \qquad (6)$$

Then, update the weight according to equation (5).

The choice of proper parameters *u* and *d* is easy as long as *u=1/d* holds. From the simulation that have been tested, the recommended values are 1.1-1.3 for *u* or 0.7-0.9 for *d.* They also use a backtracking strategy which restarts an update step if the total error increases. For this restart all learning rates are halved.

B. Delta Bar Delta Rule

Delta-Bar-Delta algorithm controls the learning rates by observing the sign changes of an exponential averaged gradient [3]. Increase the learning rates by adding a constant value instead of multiplying it. Hence,

1. Choose some small initial value for every $\eta_{ji}(0)$.
2. Adapt the learning rates:

$$\begin{aligned} &\eta_{ji}(n) = \eta_{ji}(n-1) + u, && \text{if } \frac{\partial \xi}{\partial w_{ji}}(n) * \frac{\partial \xi}{\partial w_{ji}}(n-1) \geq 0 \\ &\eta_{ji}(n) = \eta_{ji}(n-1) * d, && \text{if } \frac{\partial \xi}{\partial w_{ji}}(n) * \frac{\partial \xi}{\partial w_{ji}}(n-1) \leq 0 \\ &\eta_{ji}(n) = \eta_{ji}(n-1), && \text{else.} \end{aligned} \qquad (7)$$

In particular it is difficult to find a proper *u*. Small values may result in slow adaptations while big ones endanger the learning process. Very different values are recommended for *u* (5.0,0.095,0.085,0.035) and *d* (0.9,0.85,0.666).

C. SuperSAB

Super SAB is also based on the idea of sign independent learning rate adaptation [4]. The basic change is to increase the learning rate exponentially instead of linearly as with Delta-Bar-Delta method. This is done to take the wide range of temporarily suitable learning rates into account. By using a proper upper limit η_{max}, the algorithm behaves perfectly all over the training period.

Recommended values for *u* is 1.05 and *d* is 0.5 and recommended values for η_{max} is between 0 and 1. Hence,

$$\begin{aligned} &\eta_{ji}(n) = \eta_{ji}(n-1) * u, && \text{if } \frac{\partial \xi}{\partial w_{ji}}(n) * \frac{\partial \xi}{\partial w_{ji}}(n-1) \geq 0 \wedge \eta_{ji}(n-1) \leq \eta_{\max} \\ &\eta_{ji}(n) = \eta_{ji}(n-1) * d, && \text{if } \frac{\partial \xi}{\partial w_{ji}}(n) * \frac{\partial \xi}{\partial w_{ji}}(n-1) \leq 0 \\ &\eta_{ji}(n) = \eta_{ji}(n-1), && \text{else.} \end{aligned} \qquad (8)$$

D. Quickprop

This is an optimization of back-propagation based on Newton's method [5]. It is applicable when, between two steps, the gradient has decreased in magnitude and has changed sign. Then a parabolic estimate of the MSE is used to determine the weights for the next step. Quickprop computes the derivatives in the direction of each weight [6]. After computing the first gradient with regular back-propagation, a direct step of the error minimum is attempted by

$$\Delta x(t) = \frac{f'(x(t))}{f'(x(t-1)) - f'(x(t))} \Delta x(t-1) \qquad (9)$$

E. Rprop

Rprop uses an adaptive version of the "Manhatan-Learning" rule and is a local adaptive learning scheme [7]. The basic principle of Rprop is to eliminate the harmful influence of the size of the partial derivative on the weight step not influenced by the magnitude of the gradient. Only the sign of the derivative is used to find the proper update direction.

Rprop uses independent update step size Δ_{ji} for every connection. Furthermore, these step sizes are adapted with respect to the sign of the actual and the last derivative. The step sizes are bounded by upper and lower limits in order to avoid oscillation and arithmetic underflow of floating point values. Finally, local backtracking is applied to those connections where sign changes of the derivative are detected. Hence,

1. Choose some small initial value for every update step size $\Delta_{ji}(0)$.

2. Adapt the step sizes:

$$\begin{aligned}
&\Delta_{ji}(n)=\Delta_{ji}(n-1)*u, && \text{if } \frac{\partial\xi}{\partial w_{ji}}(n)*\frac{\partial\xi}{\partial w_{ji}}(n-1)\geq 0\\
&\Delta_{ji}(n)=\Delta_{ji}(n-1)*d, && \text{if } \frac{\partial\xi}{\partial w_{ji}}(n)*\frac{\partial\xi}{\partial w_{ji}}(n-1)<0\\
&\Delta_{ji}(n)=\Delta_{\max}, && \Delta_{ji}(n)\geq\Delta_{\max}\\
&\Delta_{ji}(n)=\Delta_{\min}, && \Delta_{ji}(n)\leq\Delta_{\min}
\end{aligned} \tag{10}$$

3. Update the connection:

$$\begin{aligned}
&\Delta w_{ji}(n)=-\Delta_{ji}(n) && \text{if } \frac{\partial\xi}{\partial w_{ji}}(n)>0\\
&\Delta w_{ji}(n)=+\Delta_{ji}(n) && \text{if } \frac{\partial\xi}{\partial w_{ji}}(n)<0\\
&\Delta w_{ji}(n)=0, && \text{else}
\end{aligned} \tag{11}$$

Recommended values for the parameters are:
Δ_{max} = 50.0, Δ_{min} = 0.000001 and u = 1.2.

IV. Dynamic Adaptation Methods

These techniques can be classified into the local adaptive techniques category since an optimal learning rate or momentum factor is assigned to each individual weight at different iterations. These techniques have been mathematically derived and proven to be effective and superior in terms of convergence when tested and compared with the batch Backpropagation [8],[9],[10].

A. Dynamic Momentum Factor(DMF)

DMF is an adjustment applied to the momentum factor at iteration *n*. Let $\Delta\alpha_{ji}(n,0)$ denote the positive adjustment applied at iteration *n* to the momentum constant at iteration 0, $\alpha_{ji}(0,0)$. We define $\Delta\alpha_{ji}(n,0)$ as

$$\Delta\alpha_{ji}(n,0)=\gamma_a^b+\alpha_{ji}(0,0) \tag{12}$$

for all $n\in[a,b]$ where $0\leq\gamma_a^b\leq 1-\alpha_{ji}(0,0)$ and $\gamma_a^b>\gamma_c^d$ for $a>c$ and $b>d$.

B. Dynamic Learning Rate 1(DLR 1)

DLR method 1 is an adjustment applied at iteration *n* to the learning rate parameter, based on the gradient of each weight. Subsequently, it varies with every iteration.

Let $\Delta\eta_{ji}(n)$ denote the adjustment applied at iteration *n* to the learning rate parameter at iteration 0, $\eta_{ji}(0)$. We define $\Delta\eta_{ji}(n)$ as

$$\Delta\eta_{ji}(n)=\lambda_{\delta b}^{\delta a}+\eta_{ji}(0) \tag{13}$$

for all $\delta=\left|\frac{\partial\xi(n)}{\partial w_{ji}(n)}\right|\in(\delta a,\delta b)$ and $\lambda_{\delta b}^{\delta a}<\lambda_{\delta d}^{\delta c}$ for $\delta a>\delta c$ and $\delta b>\delta d$.

C. Dynamic Learning Rate 2 (DLR 2)

DLR method 2 is a positive adjustment applied at iteration *n* to the learning rate parameter. Let $\Delta\eta_{ji}(n,0)$ denote the positive adjustment applied at iteration *n* to the learning rate parameter at iteration 0, $\eta_{ji}(0,0)$. We define $\Delta\eta_{ji}(n,0)$ as

$$\Delta\eta_{ji}(n,0)=\chi_a^b+\eta_{ji}(0,0) \tag{14}$$

for all $n\in[a,b]$ where $\chi_a^b\geq 0$ and $\chi_a^b>\chi_c^d$ for $a>c$ and $b>d$.

The initial value of η, $\eta_{ji}(0,0)$ can be any small value in the interval $[0,1]$.

V. Simulation Problem

A. Human Face Recognition Problem

Computer simulations for the local adaptive techniques, global adaptive techniques and dynamic adaptation methods are presented. The performance of all the techniques was compared with the Backpropagation (BP). The methods include: (1) Backpropagation (BP) (2) Sign Changes (SC) (3) Delta-Bar-Delta Rule (DB) (4) SuperSAB (SAB) (5) Quickprop (QP) (6) Rprop (RP) (7) Dynamic Momentum Factor (DMF) (8) Dynamic Learning Rate 1 (DLR1) (9) Dynamic Learning Rate 2 (DLR2) (10) Powell-Beale (CGB) (11) Fletcher-Reeves (CGF), and (12) Polak-Ribiere (CGP) . Here, the performance of these techniques were compare on the human face recognition problem.

Table 1 shows the percentage of improvement of the algorithms compared to the BP algorithm in terms of gradient evaluation and error function evaluation. Each gradient evaluation and error function evaluation consists of μ, SD and Min/Max which respectively denote the mean number of gradient or error function evaluation, standard deviation and minimum/maximum number of gradient or error function evaluation.

A detailed description of the data sets and architecture of the network can be referred to Zainuddin and Ahmad Fadzil [11]. The data set for training and testing consists of 45 different face images each. A 460-12-5 network was used where there were 460 input nodes, 12 hidden nodes and output nodes corresponding to the 5 classification classes. The training process was terminated when the MSE reached $1*10^{-3}$ in 30,000 epochs.

TABLE I
SIMULATION RESULTS FOR THE HUMAN FACE RECOGNITION PROBLEM

Algorithm	Gradient Evaluation			Error Function Evaluation			Improvement
	μ	SD	Min/Max	μ	SD	Min/Max	%
BP	18523	680.6	17001/20045	18523	680.6	17001/20045	-
SC	617.5	206.4	156/1079	617.5	206.4	156/1079	96.67
DB	387	163.5	154/620	387	163.5	154/620	97.91
SuperSAB	577	186.5	160/994	577	186.5	160/994	96.88
Quickprop	385.5	101.7	158/613	385.5	101.7	158/613	97.92
Rprop	148	16.1	112/184	148	16.1	112/184	99.20
DMF	250	71.6	90/410	250	71.6	90/410	98.65
DLR 1	227.5	66.0	80/375	227.5	66.0	80/375	98.77
DLR 2	326	35.3	247/405	326	35.3	247/405	98.24
CGB	128.5	1.6	125/132	170.5	1.6	167/174	99.19
CGF	557.5	190.3	132/983	566	179.3	165/967	96.97
CGP	140.5	2.0	136/145	173.5	2.9	167/180	99.15

B Discussion of Result

It can be seen from the results exhibited in Table I, that the adaptive methods do significantly accelerate the learning process when compared to the BP method. However, in terms of Local Adaptive Techniques, the Rprop algorithm exhibits the best performance of the average number of gradient and error function evaluations required and it has the highest percentage of success. This algorithm takes advantage of its inherent mechanism to prevent entrapment in the neighborhood of a local minimum. The Quickprop, Delta Dar Delta and SuperSAB algorithms also gave a better performance after the Rprop algorithm, while the Sign Changes algorithm still needs more number of gradient and error function evaluations.

Among the Global Adaptive Techniques, the CGB algorithm shows the best performance followed by CGP and CGF algorithms. This is due to the formula of a restart direction by the CGB algorithm which helps to improve the learning speed. For the Dynamic Adaptation Methods, the DLR 1 shows a better convergence where the percentage of improvement over the BP algorithm is 98.77%. It shows that the use of the learning rate adaptation for each connection weight according to the gradient interval helps the DLR 1 algorithm to improve the learning speed.

On the other hand, the Local Adaptive Techniques and the Dynamic Adaptation Methods are more robust in the sense that they exhibit good performance and are capable of achieving results of over 96%. There were no cases of the solution getting stuck at local minima values implying that the choice of weights were appropriate.

The results with respect to generalization performance of the tested algorithms indicate that all techniques had proven its capability where the correlation coefficient is close to 1. The variation in the network outputs is explained very well by the corresponding targets.

REFERENCES

[1] Magoulas, G.D., Plagianakos, V.P., & Vrahatis, M.N. Globally Convergent Algorithms with Local Learning Rates. In *IEEE Transactions on Neural Networks,* Vol 13, No 3, 774 – 779, 2002.

[2] Silva F.M. & Almeida L.B., Acceleration techniques for the backpropagation algorithm. *Neural Networks EURASIP Workshop*, Sesim, 1990.

[3] Jacobs R. A., Increased rates of convergence through learning rate adaptation, *Neural Networks*, 1(4), 295-308, 1988.

[4] Tollenaere, T., SuperSab:fast adaptive backpropagation with good scaling properties. *Neural Networks*, 3(5), 1990.

[5] Fahlman, S.E., An empirical study of learning speed in backpropagation networks, Technical Report, CMU-CS-88-162, 1988

[6] Vrahatis, M.N., Magoulas, G.D. & Plagaianakos, V.P. Convergence Analysis of the Quickprop Method. University of Patras and Athens, 1999.

[7] Riedmiller, M. & Braun, H., A direct adaptive method for faster backpropagation learning. The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks* (ICNN) (Ruspini, H. ed), p. 586-591, 1993, San Francisco.

[8] Evans, D.J. and Zainuddin, Z., Acceleration of the backpropagation through dynamic adaptation of the momentum. *Neural, Parallel & Scientific Computations*, 5(3), 1997, 297-308. (see also Internal Report No.1028, PARC, Loughborough University of Tech., U.K., 1996).

[9] Zainuddin Z. & Evans D.J., Acceleration of the backpropagation through dynamic adaptation of the learning rate, *Int. Journal of Computer Mathematics*, 334, 1-17, 1997 (see also Internal report No. 1029, PARC, Loughborough University of Tech., U.K. 1996).

[10] Zainuddin Z. & Sathasivam S. , Modeling nonlinear relationships in ecology and biology using neural networks, *Proceedings of the National Workshop in Ecological and environmental modeling (ECOMOD)*, Sept. 3-4, 2001.

[11] Zainuddin Z. & Ahmad Fadzil, M.H., Training Feedforward Neural Networks: An Algorithm Giving Improved Convergence. *Proceedings of research & Development in Computer Science & its Application,p.98-102,* 1998, *Penang, Malaysia.*