International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:6, 2007

Modified Levenberg-Marquardt Method for Neural Networks Training

Amir Abolfazl Suratgar, Mohammad Bagher Tavakoli, and Abbas Hoseinabadi

***Abstract*—**In this paper a modification on Levenberg-Marquardt algorithm for MLP neural network learning is proposed. The proposed algorithm has good convergence. This method reduces the amount of oscillation in learning procedure. An example is given to show usefulness of this method. Finally a simulation verifies the results of proposed method.

***Keywords*—**Levenberg-Marquardt, modification, neural network, variable learning rate.

I. Introduction

THE Error Back Propagation (EBP) algorithm [1]–[4] has been a signification improvement in neural network research, but it has a weak convergence rate.

Many efforts have been made to speed up EBP algorithm [5]–[9]. All of these methods lead to little acceptable results. The Levenberg-Marquardt (LM) algorithm [4], [10]–[13] ensued from development of EBP algorithm dependent methods. It gives a good exchange between the speed of the Newton algorithm and the stability of the steepest descent method [11], that those are two basic theorems of LM algorithm. An attempt has been made to speed up LM algorithm with modified performance index and gradient computation [14], although it is unable to reduce error oscillation. Other effort with variable decay rate has been ensued to reduce error oscillation [15], but offered algorithm had low speed compared standard LM algorithm.

In this paper a modification is made on Learning parameter resulted in to decrease together both learning iteration and oscillation. A modification method by varying the learning parameter has been made to speed up LM algorithm. In addition, the error oscillation has been decreased.

Section II describes the LM algorithm. Section III the proposed form of the modification on learning parameter is introduced. In section IV a simulation is discussed.

II. The Levenberg-Marqurадt Method Review

In the EBP algorithm, the performance index F(w) to be minimized is defined as the sum of squared errors between the target outputs and the network's simulated outputs, namely:

$$F(w) = e^T e \tag{1}$$

Where **w** = [w1, w2, …., w_N] consists of all weights of the network, **e** is the error vector comprising the error for all the training examples.

When training with the LM method, the increment of weights **Δw** can be obtained as follows:

$$\Delta w = \left[J^T J + \mu I\right]^{-1} J^T e \tag{2}$$

Where J is the Jacobian matrix, μ is the learning rate which is to be updated using the β depending on the outcome. In particular, μ is multiplied by decay rate β ($0<\beta<1$) whenever $F(w)$ decreases, whereas μ is divided by β whenever $F(w)$ increases in a new step.

The standard LM training process can be illustrated in the following pseudo-codes,

```
1. Initialize the weights and parameter μ (μ=.01 is appropriate).
2. Compute the sum of the squared errors over all inputs $F(w)$.
3. Solve (2) to obtain the increment of weights Δw
4. Recomputed the sum of squared errors $F(w)$
Using w + Δw as the trial w, and judge
IF trial $F(w) < F(w)$ in step 2 THEN
        $w = w + \Delta w$
        $\mu = \mu \cdot \beta (\beta = .1)$
        Go back to step 2
ELSE
        $\mu = \mu / \beta$
        go back to step 4
END IF
```

III. Modification of the LM Method

Considering performance index is $F(w) = e^T e$ using the Newton method we have as:

Amir Abolfazl suratgar is Assistant professor in the Electrical Engineering Department, University of Arak, Arak, Iran. (phone: +98-861-22 25 946; fax: +98-861-22 25 946; e-mail: a-surtagar@ araku.ac.ir).

Mohammad Bagher Tavakoli is Msc. Student of Electrical Engineering Department of Azad University, Arak, Iran (e-mail: m-tavakoli@ iau-arak.ac.ir).

Abbas Hoseinabadi is Msc. Student of Electrical Engineering Department of Azad University, Arak, Iran (e-mail: a-hoseinabadi@ iau-arak.ac.ir).

$$W_{K+1} = W_K - A_K^{-1} \cdot g_K \tag{3}$$

$$A_k = \nabla^2 F(w)\big|_{w=w_k} \tag{4}$$

$$g_k = \nabla F(w)\big|_{w=w_k} \tag{5}$$

$$\begin{aligned}[\nabla F(w)]_j &= \frac{\partial F(w)}{\partial w_j} \\ &= 2\sum_{i=1}^{N} e_i(w).\frac{\partial e_i(w)}{\partial wj}\end{aligned} \tag{6}$$

The gradient can write as:

$$\nabla F(x) = 2J^T e(w) \tag{7}$$

Where

$$J(w) = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_1} & \frac{\partial e_{11}}{\partial w_2} \cdots\cdots & \frac{\partial e_{11}}{\partial w_N} \\ \frac{\partial e_{21}}{\partial w_1} & \frac{\partial e_{21}}{\partial w_2} \cdots\cdots & \frac{\partial e_{21}}{\partial w_N} \\ \vdots & & \\ \frac{\partial e_{KP}}{\partial w_1} & \frac{\partial e_{KP}}{\partial w_2} \cdots\cdots & \frac{\partial e_{KP}}{\partial w_N} \end{bmatrix} \tag{8}$$

$J(w)$ is called the Jacobian matrix.

Next we want to find the Hessian matrix. The k, j elements of the Hessian matrix yields as:

$$\begin{aligned}[\nabla^2 F(w)]_{k,j} &= \frac{\partial^2 F(w)}{\partial w_k \partial w_j} \\ &= 2\sum_{i=1}^{N}\{\frac{\partial e_i(w)}{\partial w_k}\frac{\partial e_i(w)}{\partial w_j} \\ &\quad + e_i(w).\frac{\partial^2 e_i(w)}{\partial w_k \partial w_j}\}\end{aligned} \tag{9}$$

The Hessian matrix can then be expressed as follows:

$$\nabla^2 F(W) = 2J^T(W)\cdot J(W) + S(W) \tag{10}$$

Where

$$S(w) = \sum_{i=1}^{N} e_i(w)\cdot \nabla^2 e_i(w) \tag{11}$$

If we assume that $S(w)$ is small, we can approximate the Hessian matrix as:

$$\nabla^2 F(w) \cong 2J^T(w)J(w) \tag{12}$$

Using (12) and (4) we obtain the Gauss-Newton method as:

$$\begin{aligned}W_{k+1} &= \\ W_k &- \left[2J^T(w_k)\cdot J(w_k)\right]^{-1} 2J^T(w_k)e(w_k) \\ &\cong W_k - \left[J^T(w_k)\cdot J(w_k)\right]^{-1} J^T(w_k)e(w_k)\end{aligned} \tag{13}$$

The advantage of Gauss-Newton is that it does not require calculation of second derivatives.

There is a problem the Gauss-Newton method is the matrix H=J^TJ may not be invertible. This can be overcome by using the following modification.

Hessian matrix can be written as:

$$G = H + \mu I \tag{14}$$

Suppose that the eigenvalues and eigenvectors of H are {λ_1, λ_2,…….,λ_n} and {z_1,z_2,…….,z_n}.Then:

$$\begin{aligned}Gz_i &= [H + \mu I]z_i \\ &= Hz_i + \mu z_i \\ &= \lambda_i z_i + \mu z_i \\ &= (\lambda_i + \mu)z_i\end{aligned} \tag{15}$$

Therefore the eigenvectors of G are the same as the eigenvectors of H, and the eigenvalues of G are (λ_i+μ). The matrix G is positive definite by increasing μ until (λ_i+μ)>0 for all i therefore the matrix will be invertible.

This leads to Levenberg-Marquardt algorithm:

$$w_{K+1} = w_K - \left[J^T(w_K)J(w_k) + \mu I\right]^{-1} J^T(w_K)e(w_K) \tag{16}$$

$$\Delta w_K = \left[J^T(w_K)J(w_K) + \mu I\right]^{-1} J^T(w_K)e(w_K) \tag{17}$$

As known, learning parameter, μ is illustrator of steps of actual output movement to desired output. In the standard LM method, μ is a constant number. This paper modifies LM method using μ as:

$$\mu = 0.01e^T e \tag{18}$$

Where **e** is a $k \times 1$ matrix therefore $\mathbf{e^Te}$ is a 1×1 therefore [$\mathbf{J^TJ+\mu I}$] is invertible.

Therefore, if actual output is far than desired output or similarly, errors are large so, it converges to desired output with large steps.

Likewise, when measurement of error is small then, actual output approaches to desired output with soft steps. Therefore error oscillation reduces greatly.

IV. Simulation Results

In this section we consider XOR gate as a case study. We simulate this gate with standard LM and modified LM. We use a Neural Network with two layers and the transfer function is as:

$$f(net) = \frac{2}{1+\exp(-net)} - 1 \tag{19}$$

Fig.1 shows the error of Neural Network simulation with standard LM and Fig.2 shows the result of modified LM. As described in Table I, the iteration of learning and the oscillation is reduced in modified LM.

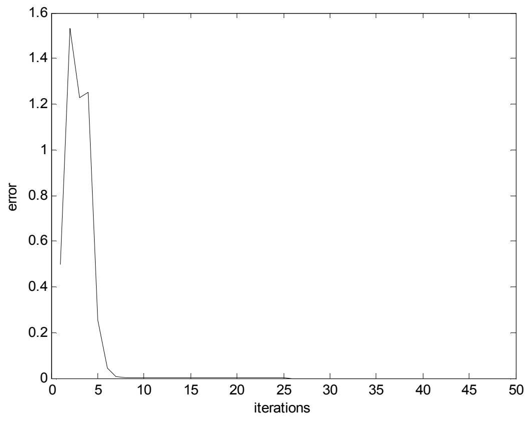

Fig.1 XOR with two hidden neurons for standard LM

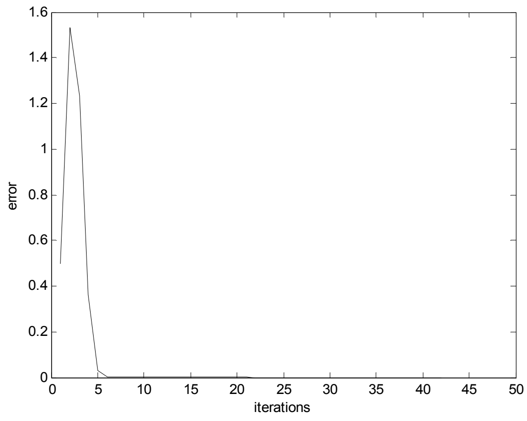

Fig.2 XOR with two hidden neurons for modified LM

Results of the standard LM and modified LM algorithms for XOR example are shown in Table I.

TABLE I
COMPARISON OF METHODS

Method	Number of hidden neurons	Number of iteration
Standard LM	2	52
Modified LM	2	41

V. ACKNOWLEDGMENT

The authors would like to thanks Azad University of Arak for their financial supports of this paper.

REFERENCES

[1] Rumelhart, D. E., Hinton, G. E. and Williams, R. J, “*Learning internal representations by error propagation*,” In Parallel Distributed Processing, Cambridge, MA: MIT Press, vol 1, pp. 318-362.

[2] Rumelhart, D. E., Hinton, G. E. and Wiliams, R. J, “*Learning representations by back-propagating errors*,” Nature, vol. 323, pp. 533-536, 1986.

[3] Werbos, P. J. “*Back-propagation: Past and future*,” Proceeding of International Conference on Neural Networks, San Diego, CA, 1, pp. 343-354, 1988.

[4] .M. T .Hagan and M. B. Menhaj, “*Training feed forward network with the Marquardt algorithm*,” IEEE Trans. on Neural Net., vol. 5, no. 6, pp.989-993, 1994.

[5] Bello, M. G. “*Enhanced training algorithms, and integrated training/architecture selection for multi layer perceptron networks*,” *IEEE Trans. on Neural Net.*, vol. 3, pp. 864-875, 1992.

[6] Samad, T. “*Back-propagation improvements based on heuristic arguments*,” Proceedings of International Joint Conference on Neural Networks, Washington, 1, pp. 565-568, 1990.

[7] Solla, S. A., Levin, E. and Fleisher, M. “*Accelerated learning in layered neural networks*,” Complex Systems, 2, pp. 625-639, 1988.

[8] Miniani, A. A. and Williams, R. D. “*Acceleration of back-propagation through learning rate and momentum adaptation*,” Proceedings of International Joint Conference on Neural Networks, San Diego, CA, 1, 676-679, 1990.

[9] Jacobs, R. A., “*Increased rates of convergence through learning rate adaptation*,” Neural Networks, vol. 1, no. 4, pp. 295-308, 1988.

[10] Andersen, T. J. and Wilamowski, B.M. “*A Modified Regression Algorithm for Fast One Layer Neural Network Training*,” World Congress of Neural Networks, Washington DC, USA, vol. 1, pp. 687-690, July 17-21, 1995.

[11] Battiti, R., “*First- and second-order methods for learning between steepest descent and Newton’s method*,” Neural Computation, vol. 4, no. 2, pp. 141-166, 1992.

[12] Charalambous, C., “*Conjugate gradient algorithm for efficient training of artificial neural networks*,” IEE Proceedings, vol. 139, no. 3, pp. 301-310, 1992.

[13] Shah, S. and Palmieri, F. “*MEKA - A fast, local algorithm for training feed forward neural networks*,” Proceedings of *International Joint Conference on Neural Networks*, San Diego, CA, 3, pp. 41-46, 1990.

[14] B. M. Wilamowski , Y. Chen, and A. Malinowski, “*Efficient algorithm for training neural networks with one hidden layer*,” In Proc. IJCNN, vol.3, pp.1725-728, 1999.

[15] T. Cong Chen, D. Jian Han, F. T. K. Au, L. G. Than, “*Acceleration of Levenberg-Marquardt training of neural networks with variable decay rate*,” IEEE Trans. on Neural Net., vol. 3, no. 6, pp. 1873 - 1878, 2003.