Acceleration of Levenberg-Marquardt Training of Neural Networks with Variable Decay Rate

Tai-cong Chen Da-jian Han
Department of Civil Engineering, South China University of Technology, Guangzhou, People's Republic of China, 510640
(cvchentc@scut.edu.cn ardjhan@scut.edu.cn)
Francis T.K. Au L.G. Tham
Department of Civil Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong
(francis.au@hku.hk hrectlg@hkucc.hku.hk)

Abstract — **In the application of the standard Levenberg-Marquardt training process of neural networks, error oscillations are frequently observed and they usually aggravate on approaching the required accuracy. In this paper, a modified Levenberg-Marquardt method based on variable decay rate in each iteration is proposed in order to reduce such error oscillations. Through a certain variation of the decay rate, the time required for training of neural networks is cut down to less than half of that required in the standard Levenberg-Marquardt method. Several numerical examples are given to show the effectiveness of the proposed method.**

I. Introduction

As is known, the Levenburg-Marquardt (LM) method shows the most efficient convergence during the Back Propagation (BP) training process because it acts as a compromise between the first-order optimization method (steepest-descent method) with stable but slow convergence and the second-order optimization method (Gauss-Newton method) with opposite characteristics [1].

However, when using the LM method for neural network training, several disadvantages are still observed in the numerical computations. Firstly, large memory is required for matrix operations in each iteration and the computational complexity increases with the number of weights in a quadratic manner. Researches to improve the standard LM method mostly concentrated on this issue. Memory demand and computational complexity can be greatly reduced by using techniques based on Jacobian deficiency [2], block-diagonal approximation [3] or matrix contraction [4]. Secondly, serious error oscillations during the standard LM training process frequently occur and this phenomenon even aggravates when approaching the required accuracy. As will be discussed in the following sections, such error oscillations resulting in lengthy calculations are mainly ascribed to the use of a fixed decay rate. A modified method by varying the decay rate according to a certain generally applicable rule during the training process is presented in this paper. The proposed method shows great effectiveness in convergence and reduces the amount of training work by more than half of that in the standard LM method.

II. The Levenberg-Marquardt method

In the BP algorithm, the performance index $F(\mathbf{w})$ to be minimized is defined as the sum of squared errors between the target outputs and the network's simulated outputs, namely

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

where $\mathbf{w}=[w_1, w_2,\ldots, w_N]^{\mathrm{T}}$ consists of all weights of the network, $\mathbf{e}$ is the error vector comprising the errors for all the training examples.

When training with the LM method, the increment of weights $\Delta\mathbf{w}$ can be obtained as follows

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

where $\mathbf{J}$ is the Jacobian. matrix, λ is the training parameter which is to be updated using the decay rate β depending on the outcome. In particular, λ is multiplied by the decay rate β ($0<\beta<1$) whenever $F(\mathbf{w})$ decreases, while λ is divided by β whenever $F(\mathbf{w})$ increases in a new step.

The standard LM training process can be illustrated in

0-7803-7898-9/03/$17.00 ©2003 IEEE

the following pseudo-codes,

```
1. Initialize the weights and parameter $\lambda$ ($\lambda = 0.01$ is appropriate).
2. Compute the sum of squared errors over all inputs, $F(\mathbf{w})$.
3. Compute the Jacobian matrix $\mathbf{J}$.
4. Solve Equation (2) to obtain the increment of weights $\triangle\mathbf{w}$.
5. Recompute the sum of squared errors $F(\mathbf{w})$ using $\mathbf{w}+\triangle\mathbf{w}$ as the trial $\mathbf{w}$, and judge
   IF   trial $F(\mathbf{w}) < F(\mathbf{w})$ in Step 2   THEN
            $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$
            $\lambda = \lambda \cdot \beta \quad (\beta = 0.1)$
            go back to Step 2.
   ELSE
            $\lambda = \lambda / \beta$
            go back to Step 4.
   END IF
```

III. Disadvantages of fixing the decay rate as $\beta = 0.1$

During the training process, the loop of Steps 2-5-2 causing a subsequently reduced value of $F(\mathbf{w})$ is called an iteration. According to Marquardt [5], a suitable strategy for decreasing $F(\mathbf{w})$ is to use a small value of λ such that the modified Gauss-Newton method would converge nicely. To achieve this, several trials of varying λ with β are carried out before deciding upon a suitable value of λ for each iteration.

In the Neural Network Toolbox of MATLAB [6], which is commonly used in neural network simulations, the default value of β is taken to be 0.1. This value is fixed during the whole training process, which is also the value recommended by Marquardt [5]. However, some disadvantages are observed in network training adopting such a strategy, as will be seen in the following numerical example.

A. Observations

A two-layer feedforward network $N_{[1\times5\times1]}$ is built to train a set of 21 input/output pairs denoted by the crosses in Fig. 1. The algorithm of the standard LM training (with $\beta = 0.1$ during the batch-mode training process) is coded in Fortran Language. Fig. 2 shows the typical training process using the standard LM method with weights initialized randomly between [-1 1], the performance of which is characterized by the variations of $F(\mathbf{w})$ and λ. As opposed to the normal practice in MATLAB of plotting the curve of $F(\mathbf{w})$ against Training Iterations, Figure 2 plots F(w) against Training Trials, as an increased $F(\mathbf{w})$ caused by improper trial of 0.1λ results in additional computation work of solving Equation (2) and calculating $F(\mathbf{w})$.

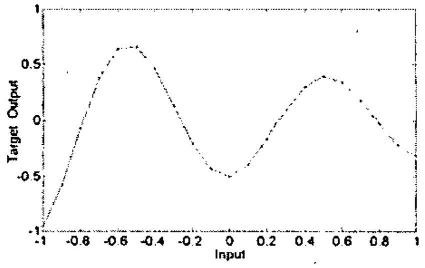


Fig. 1. Prototype of desired curve

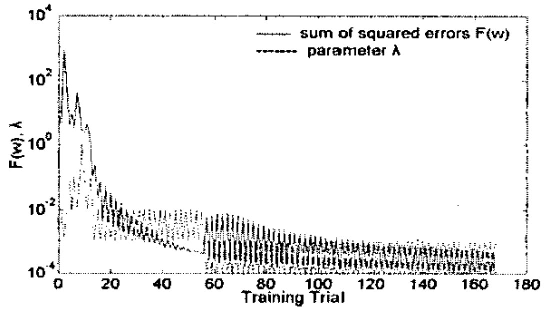


Fig. 2. Typical training process by the standard LM method

From Fig. 2, several characteristics of the training curves are observed:

1. In the curve of $F(\mathbf{w})$ against Training Trials, many error oscillations occur during the training process and this phenomenon aggravates on approaching the required accuracy. It takes only fewer than 10 oscillations (nearly 20 trials) for $F(\mathbf{w})$ to descend from the initial value of 4.4787E0 to 1E-1, while nearly 20 oscillations (nearly 40 trials) subsequently to reach 1E-3 and 50 oscillations (nearly 100 trials) eventually to reach 1E-4. Such a kind of error oscillations implies that the speed of convergence greatly slows down on approaching the required accuracy.

2. In the curve of λ against Training Trials, when

$F(\mathbf{w})$ reaches 1.0016E-1, λ takes the value of 1E-2. Then λ falls into a longtime oscillation of constant amplitude until $F(\mathbf{w})$ reaches 4.2567E-4 when λ takes the value of 1E-3:. Afterwards, an even longer period of oscillation happens until $F(\mathbf{w})$ reaches the required accuracy. This kind of oscillations in λ implies that many trials in decreasing λ by multiplying $\beta = 0.1$ would not be valuable to the reduction of $F(\mathbf{w})$ but cause unexpected ascend of $F(\mathbf{w})$ and therefore waste time.

Although the exact numbers of staggered oscillations differ among different training processes due to the use of different initial weights, they all have serious problems of error oscillations before the required accuracy is reached. The main reason for this kind of error oscillations is the fixed value of the decay rate β of 0.1 during the training process.

B. Discussion

Generally, when a new reduced $F(\mathbf{w})$ is reached, an "ideal" λ is obtained and the weights are updated by adding the new increments. In the subsequent trial, the choice of a new λ is not only a choice of step direction but also a choice of step size [5]. An attempt of decreasing λ by multiplying β ($0 < \beta < 1$) can therefore be seen as a trial roughly along the Gauss-Newton direction with a step size larger than the ones determined by the former λ. In addition, having $\beta = 0.1$ means a much large scale in both aspects.

At the beginning of the training process, when $F(\mathbf{w})$ is much greater than the required accuracy, a trial with step direction close to the Gauss-Newton direction with a large step size may cause a great reduction of $F(\mathbf{w})$ which is expected in fast convergence. Repeated trials with decreasing λ by multiplying by $\beta = 0.1$ often succeed in a great reduction of $F(\mathbf{w})$, as can be seen from Fig.2. Hence, such an approach is valuable and effective at this stage.

However at the subsequent stages when $F(\mathbf{w})$ approaches the neighborhood of the minimum, the use of a small step size is as important for stable convergence as the large reduction of $F(\mathbf{w})$ at the beginning. As stable convergence is essential at this stage, adopting a step close to the Gauss-Newton direction with the corresponding step size by setting $\beta = 0.1$ is not as important. Hence at such stages, some value of β in the range of $0.1 < \beta < 1$ will be helpful for the trial of decreasing λ to obtain a relative large descend of $F(\mathbf{w})$. Actually using the value of $\beta = 0.1$ to experiment with decreasing values of λ seldom succeeds in reducing $F(\mathbf{w})$, as can be seen from Fig. 2. If an unexpected ascend of $F(\mathbf{w})$ is obtained, a redeeming trial should be made to resume λ to the former ideal one by dividing with $\beta = 0.1$. These two trials not only cause unnecessary calculations of two values of $F(\mathbf{w})$, but also obtain a small descend of $F(\mathbf{w})$ determined by the former "ideal" λ. Consequently, the speed of convergence slows down.

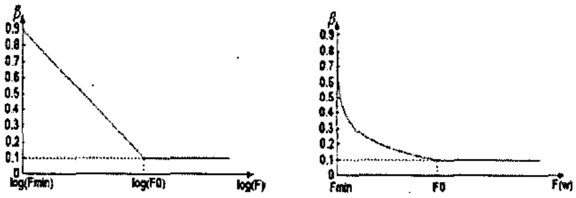
Similarly, the decay rate β has been tested with fixed values of 0.2, 0.3, … 0.9 during the training process, and numerical computations show that fixing the decay rate β at any acceptable value can hardly achieve both fast descending at the beginning stages and stable convergence close to the required accuracy in one training process. In other words, β should assume different values at different stages so as to meet the relevant convergence demand for accelerating the network training.

IV. Rule of decay rate variation

In view of the above discussions, a Log-Linear function is proposed as the rule of varying β after an iteration, namely,

$$\beta = \frac{\log(F) - \log(F_{\min})}{\log(F_0) - \log(F_{\min})} \times 0.8 + 0.1 \qquad (3)$$

where F is the reduced sum of squared errors, F_0 is the first calculated $F(\mathrm{w})$ based on initialized weights and $F_{\min}$ is the required training accuracy.



(a) Logarithmic scale (b) Normal scale

Fig. 3. Rule of decay rate variation

When the weights are initialized with the same values used in Fig. 2, and at the same time the decay rate β is adjusted using the rule shown in Eq. (3) and Fig. 3, the speed of training is greatly accelerated. As shown in Fig. 4, the parameter λ finally reaches a minimum value of 1.677E-04 and the total number of training trials is just 44, which is much lower than the 169 trials in Fig. 2 with the fixed decay rate of $\beta = 0.1$.

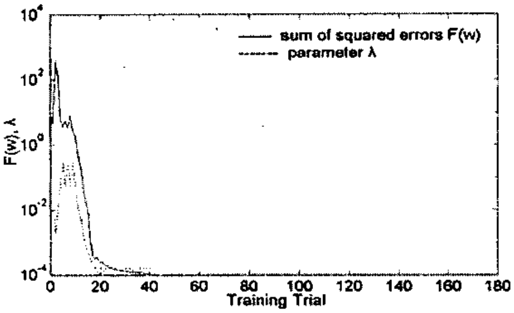


Fig. 4. Improved training process by the modified LM method

V. Case studies

Example 1: Nonlinear Curve Mapping

The objective in this test case is to learn a mapping with a training set of 21 input/output pairs denoted by the crosses in Fig. 1. A two-layer feedforward network $N_{[1\times5\times1]}$ was trained by using the standard LM method and the modified LM method. Two schemes of initializing weight are also experimented to study the performance of the new method.

TABLE I

TRAINING RESULTS FOR NONLINEAR CURVE MAPPING

	Runs	Ave. Ite.	Ave. Trials	Effic- iency	Ave. Time(s)
Standard LM (I)	100	124.14	246.19	50.42%	0.15
Modified LM (I)	100	43.90	68.05	64.51%	0.05
Standard LM (II)	100	44.20	86.20	51.28%	0.06
Modified LM (II)	100	29.76	44.06	67.54%	0.03

(I) Weight initialization by random selection between [-1, 1];

(II) Weight initialization by Nguyen-Widrow method [7].

Efficiency = (Average Iterations) / (Average Trials).

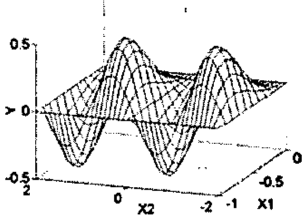
The typical training curves of the standard LM method and the modified LM method using the same initialized weights are shown in Fig. 2 and Fig. 4 respectively. Table I summarizes the training results of the two methods using two different initialization cases.

Example 2: Nonlinear Static Mapping

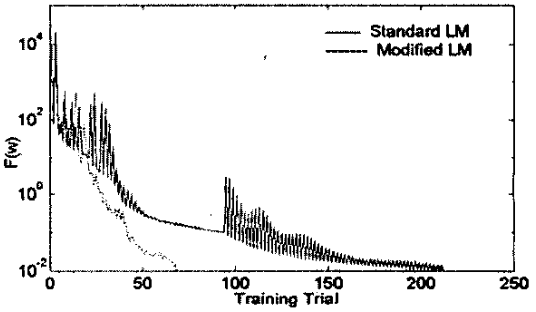
In this example we train neural networks to approximate the surface shown in Fig. 5(a). The function describing this surface is

$$y = 0.5\sin(\pi x_1^2)\sin(2\pi x_2) \qquad (4)$$

A two-layer feedforward network $N_{[2\times20\times1]}$ was trained with 451 intersection points shown in Fig. 5(a). The training results are shown in Fig. 5(b) and Table II.



(a) Prototype of desired surface



(b) Typical training curve

Fig. 5. Simulation of a 2-dimensional desired function

TABLE II

TRAINING RESULTS FOR NONLINEAR SURFACE MAPPING

	Runs	Ave. Ite.	Ave. Trials	Effic- iency	Ave. Time(s)
Standard LM (I)	100	116.73	233.02	50.09%	19.35
Modified LM (I)	100	42.96	73.04	58.82%	6.85
Standard LM (II)	100	82.03	162.36	50.52%	13.45
Modified LM (II)	100	35.17	57.91	60.73%	5.46

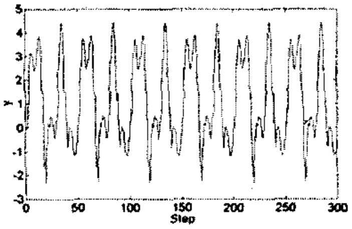
Example 3: Five-Step Advanced Prediction

In this example we train neural networks to predict results five steps ahead in a nonlinear dynamic process as shown in Fig. 6(a). The inputs and outputs of the system satisfy the following equation,

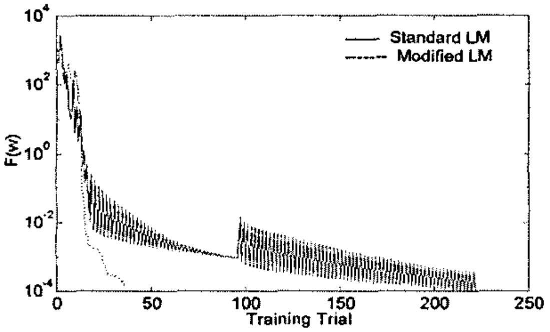
$$y_{(i+1)} = \frac{y_{(i)}y_{(i-1)}(y_{(i)}+2.5)}{1+y_{(i)}^2+y_{(i-1)}^2}+u_{(i)} \qquad (5)$$

where,

$$u_{(i)} = \sin(2\pi i/25)+\sin(2\pi i/10) \qquad (6)$$



(a) Response of the nonlinear dynamic process



(b) Typical training curve

Fig. 6. Simulation of a five-step advanced prediction

TABLE III

TRAINING RESULTS FOR 5-STEP ADVANCED PREDICTION

	Runs	Ave.	Ave.	Effic-	Ave.
		Ite.	Trials	iency	Time(s)
Standard LM (I)	100	125.29	248.01	50.52%	11.41
Modified LM (I)	100	38.04	56.13	67.77%	3.32
Standard LM (II)	100	101.78	201.16	50.60%	9.98
Modified LM (II)	100	30.76	47.67	64.53%	2.98

A two-layer feedforward network $N_{[9\times 10\times 1]}$ was employed to predict a five-step advanced system output $y_{(i+5)}$. The 9 network inputs consist of 6 system inputs and 3 delayed outputs of the system, i.e. $(u_{(i+4)},\ldots, u_{(i-1)}, y_{(i)}, y_{(i-1)}, y_{(i-2)})$. In all the simulations, 200 training samples were used for training the network. The training results are shown in Fig. 6(b) and Table III.

VI. Conclusions

Although the standard LM method is considered to be one of the most effective methods in training feedforward neural networks, error oscillations frequently occur during the process and the problem usually aggravates on approaching the required accuracy due to the fixed decay rate $\beta = 0.1$. As a result, the process is time-consuming because of a lot of unnecessary trials.

In the modified LM method, error oscillations are greatly reduced and therefore network training is greatly accelerated. There are three major advantages in the proposed method:

(1) The decay rate β varying from 0.1 to 0.9 with reduced $F(\mathbf{w})$ guarantees that a subsequent trial of decreasing λ would usually cause a relative large descend of $F(\mathbf{w})$, and this is valuable in the neighborhood of the minimum. As a result, training is greatly accelerated and it often involves less than half the amount of computation required in the standard LM method.

(2) Modification to the decay rate as practiced in the modified LM method is more effective in convergence than improvement in weight initialization.

(3) Training acceleration achieved by the modified LM method is not so sensitive to weight initialization condition as the standard LM method. In other words, special consideration of effective weight initialization is not essential when using the modified LM method.

In the practical application of neural networks, fast convergence in training is always desired, especially when network learning is running under a real-time mode. The method proposed in this paper can meet such requirements pretty well with little additional computation work compared with the standard LM method.

References

[1] M. T. Hagan and M. B. Menhaj, "Training feedforward

networks with the Marquardt algorithm", *IEEE Trans. on Neural Net.*, vol. 5, no. 6, pp. 989-993, 1994.

[2] G. Zhou and J. Si, "Advanced neural-network training algorithm with reduced complexity based on Jacobian deficiency", *IEEE Trans. on Neural Net.*, vol. 9, no. 3, pp. 448-453, 1998.

[3] L. W. Chan and C. C. Szeto, "Training recurrent network with block-diagonal approximated Levenberg-Marquardt algorithm", in *Proc. IJCNN*, vol. 3, pp. 1521 -1526, 1999.

[4] B. M. Wilamowski, Y. Chen, and A. Malinowski, "Efficient algorithm for training neural networks with one hidden layer", in *Proc. IJCNN*, vol. 3, pp. 1725-1728, 1999.

[5] D. Marquardt, "An algorithm for least squares estimation of nonlinear parameters", *J. Soc. Ind. Appl. Math.*, pp. 431-441, 1963.

[6] H. Demuth and M. Beale, *Neural Network TOOLBOX User's Guide. For use with MATLAB.* The MathWorks Inc., 1998.

[7] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of adaptive weights", in *Proc. IJCNN*, vol. 3, pp. 21-26, 1990.