Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 653072, 10 pages
http://dx.doi.org/10.1155/2014/653072

Research Article

A Joint Optimization of Momentum Item and Levenberg-Marquardt Algorithm to Level Up the BPNN's Generalization Ability

Lei Xiao,[1] Xiaohui Chen,[1] and Xinghui Zhang[2]

[1] *The State Key Lab of Mechanical Transmission, Chongqing University, Chongqing 400030, China*
[2] *Mechanical Engineering College, Shijiazhuang 050003, China*

Correspondence should be addressed to Lei Xiao; leixiao211@163.com

Received 10 December 2013; Accepted 7 March 2014; Published 16 April 2014

Academic Editor: Baocang Ding

Back propagation neural network (BPNN) as a kind of artificial neural network is widely used in pattern recognition and trend prediction. For standard BPNN, it has many drawbacks such as trapping into local optima, oscillation, and long training time. Because training the standard BPNN is based on gradient descent method, and the learning rate is fixed. Momentum item and Levenberg-Marquardt (LM) algorithm are two ways to adjust the weights among the neurons and improve the BPNN's performance. However, there is still much space to improve the two algorithms. The hybrid optimization of damping factor of LM and the dynamic momentum item is proposed in this paper. The improved BPNN is validated by Fisher Iris data and wine data. Then, it is used to predict the visit_spend. The database is provided by Dunnhumby's Shopper Challenge. Compared with the other two improved BPNNs, the proposed method gets a better performance. Therefore, the proposed method can be used to do the pattern recognition and time series prediction more effectively.

1. Introduction

Artificial neural network is a computing model which is similar to biological neural network. It is widely used in stimulation, trend prediction, pattern recognition, and control system. It could realize self-study without knowing the function and relationship among the training datasets. It is widely used in practice and researched in academia. For example, Jing and Cheng [1] developed a new optimal PID learning for training feedforward neural networks (FNN) for any purpose (system identification, function approximation, pattern recognition, control, etc.). And, in their paper, they compared its effect with some types of neural networks, such as BP, scale conjugate BP, and LM-BP. Although standard BPNN is one of the typical neural networks, it still has many unavoidable shortcomings. Because, the learning rate is fixed and the weights in standard BPNN are adjusted by gradient descent method according to error back propagation, for BPNN, it's easy to trap into local optima, oscillation and long training time. These issues make the standard BPNN unable to meet the demands of some pattern recognition or data regression items which need fast processing (e.g., the real-time condition monitoring of complex mechanical equipment and some control systems). In recent decades, many improvements were developed on the standard BPNN to speed up the training process and get more accurate results. And some are variants of neural networks. Among the published works, the improvements engage in solving the three main drawbacks: (1) slow learning speed, (2) oscillation, and (3) convergence to local optima.

(1) Alternative Learning Rate. In the standard BPNN, the learning rate is fixed in the whole iteration process. The magnitude of the learning rate decides the steps of the weights update, which is strongly related to the learning time. Roy [2] introduced the near optimal learning rate into the adjustment of the learning rate. Song et al. [3] adjusted the learning rate value according to the change of system error between two consecutive steps. A self-adaptive learning rate based on the adjustment of weights and biases changes was reported in [4].

Reference [5] proposed a new dynamic optimal learning rate which is related to the previous approach after the first iteration. Hasan et al. [6] developed Hanning window neural network and used Hanning window function to make the learning rate dynamic. The linear matrix inequality techniques were used to find the appropriate learning rates to guarantee the fast and robust convergence [7]. These methods make a good performance in the learning speed.

(2) Momentum Item Methods. Momentum method is another way to modify the adjustment of weights. It reflects the previous information of the weights adjustment and could accelerate the training speed and at the same time weaken the oscillation. A new way to accelerate the convergence by adjusting the learning rate and momentum factor at the same iteration was reported by Yu and Liu [8]. Wang et al. [9] proposed a restart strategy for the momentum in order to converge the cyclic and almost-cyclic learning with a single hidden layer neural network. Another way to adjust the momentum coefficient according to error function and weights in the network was given by Wu et al. [10].

(3) Second Order Methods. The gradient descent method only uses the first-order derivative of the error function; many works have proved that the second-order derivative methods are more efficient in increasing the convergence speed and getting more accurate results than gradient descent. These second order methods include quasi-Newton [11], conjugate gradient [12], and LM algorithm [13, 14]. Specially, LM algorithm is a good adjuster of the Gauss-Newton technique and the steepest-descent algorithm but avoids many of their limitations [13]. The adjuster is based on the conception of damping factor. The adjustment of the parameter in the LM algorithm is according to the iteration effectiveness [15–17], while most works just regard the LM algorithm as the training method without any improvement [18].

Although there are many aspects proposed for improving BPNN, it is still deficient. Especially for some engineering demands, the real-time condition monitoring and trend prediction stress the BPNN less training time and more accurate results so that some timely operations could be implemented. Therefore, in this paper, hybrid optimization of dynamic momentum item and damping LM algorithm is proposed to accelerate the convergence speed and get more accurate results. The momentum item is added to weaken the oscillation. And LM algorithm is involved to speed up the iteration. Being different from the standard BPNN, the adjustment of the weights in the proposed method utilizes the second order derivative information and the previous iteration. The weights in the next iteration are decided by three sections, the current weight, the weight adjustment, and hybrid optimized previous weight adjustment. The influence of the previous weights is determined by the momentum coefficient and damping factor in LM algorithm. The momentum coefficient and damping factor vary in each iteration. The adjustment of the momentum coefficient is decided by the previous momentum parameter in the last iteration. It increases if the error of the BPNN declines. When it reaches the maximum, it returns to a certain value. Oppositely, damping factor in LM declines if the iteration performs better than before. Overall, the parameters vary according to the former values and the iteration efficiency.

The rest of the paper is structured as follows. Section 2 presents the novel momentum and LM algorithm to train the BPNN with one hidden layer. Section 3 validates the proposed training method by Fisher Iris data and wine data compared in the aspect of pattern recognition with LM proposed by El-Alfy [17] training BPNN and the BPNN trained by improved LM algorithm which is provided by Nørgaard [19]. In Section 4, Dunnhumby's Shopper Challenge dataset is used to prove the improvements of prediction. Section 5 outlines the conclusions and presents the future work.

2. Modified Training Algorithm for BPNN

2.1. Basic BPNN. BPNN is a kind of feedback neural network. Its main principle is the error back propagation. The adjustment of weights is based on the gradient descent which requires that the activation function has the first order derivative. The iteration terminal condition is meeting the predetermined error goal or reaching the maximum iteration steps. Therefore, it is a supervised network. Its learning rule with one hidden layer is shown in Figure 1.

Considering the convenience of description, supposing that there is one input layer, one hidden layer, and one output layer, x_m and d_m are the input vectors and the desired output vectors, respectively, where $m = 1, 2, \ldots, M$. In the standard BPNN, the adjustments of weights are based on the derivative of error function. So the error function is vital. Fontenla-Romero et al. [20] considered the influence of the slope of the nonlinear activation functions and proposed a new way to measure the error. Nguyen et al. [18] proposed a new cost function considering the system error and the cluster weight which represents an approximation to the probability mass. In this paper, the error function is defined as follows:

$$E(w) = \frac{\sum (y_m - d_m)^2}{2M}, \tag{1}$$

where y_m is the real network output and w is the weight vector.

For the standard BPNN, the gradient descent is implemented to train the network. And the weights update based on the last iteration and the changes

$$w(t+1) = w(t) + \Delta w(t). \tag{2}$$

$w(t)$ means weight vector in the tth iteration time, and $\Delta w(t)$ denotes the changes of weights in the tth iteration time. $w(t+1)$ is the weight vector for the next iteration time.

2.2. The Classical Momentum Item and LM Algorithm. Among the published improvements in standard BPNN, momentum item and LM algorithm are two common and effective ways to improve BPNN's performance. They are used to weaken the oscillation and speed up convergence. The main characteristics of the two algorithms are as follows. More details can be seen in the works [21, 22], respectively.

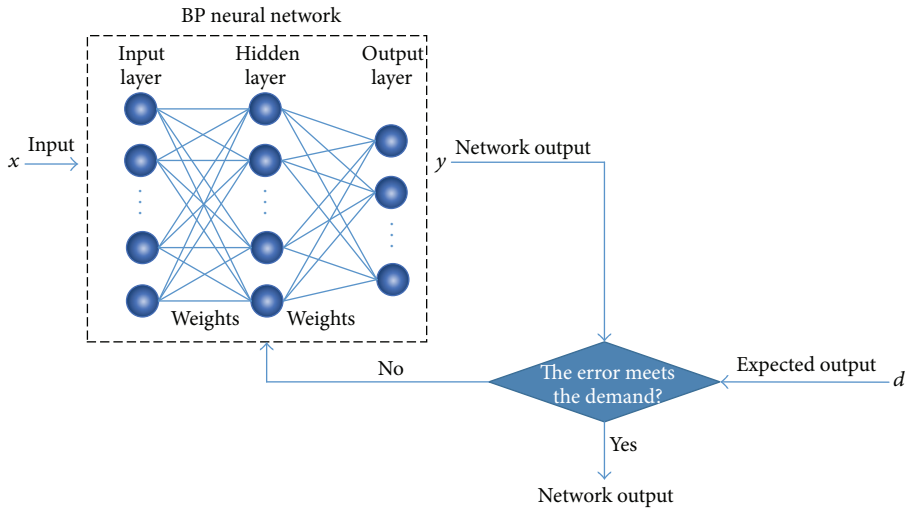


Figure 1: BPNN construction and its supervised learning rule with one hidden layer.

Table 1: The recognition results of different training methods for Fisher Iris data.

Real labels	Preprocessed output			Postprocessed output		
	ELM-BPNN	NLM-BPNN	Proposed LM-BPNN	ELM-BPNN	NLM-BPNN	Proposed LM-BPNN
1	0.99999933	0.99999897	0.99999953	1	1	1
1	0.99999930	0.99999965	0.99999953	1	1	1
1	0.99999937	0.99999897	0.99999953	1	1	1
1	0.99999933	0.99999897	0.99999953	1	1	1
1	0.99999933	0.99999897	0.99999953	1	1	1
2	1.99999855	31.3832497	2.00000393	2	**31**	2
2	2.00000235	2.51428256	2.00000169	2	**3**	2
2	1.99996337	−0.8690897	2.00210956	2	**−1**	2
2	2.00000217	1.99807531	2.00001190	2	2	2
2	2.00000623	1.80912596	2.00193468	2	2	2
3	2.99999644	2.99999999	3.00000059	3	3	3
3	2.49818368	1.99998783	2.50605341	**2**	**2**	3
3	2.99037581	2.99988701	2.99999197	3	3	3
3	3.00000291	2.99999999	3.00000059	3	3	3
3	2.99662415	2.99542888	2.79537363	3	3	3

Table 2: Comparison of the three methods on benchmark datasets for Fisher Iris data.

	Iteration steps	Iteration time	Accuracy rate
ELM-BPNN	483	1.092726 seconds	93.33%
NLM-BPNN	1133	2.337680 seconds	73.33%
Proposed LM-BPNN	244	0.599163 seconds	100%

(1) Gradient descent with momentum item: the weight's change is related to the previous weight update:

$$\Delta w(t) = \eta\delta(t)\, v(t) + \alpha\Delta w(t-1), \quad (3)$$

where η is the learning rate. δ is the gradient which is derived from the standard BPNN derivation process. When the weight update of input-hidden layer is calculated, v is the input of the samples, while, when the weight update of hidden-output layer is calculated, v is the output of the hidden layer. α is the momentum coefficient, $0 < \alpha < 1$.

(2) Levenberg-Marquardt: the weight update rule is as follows:

$$\Delta w = -\left[J^T(w)\cdot J(w) + \lambda I\right]^{-1}\cdot J^T(w)\, e(w), \quad (4)$$

where the term $e(w)$ denotes the error vector of the neural network. λ is a damping factor which impacts the performance of the convergence. It is also the adjuster of steepest-descent method and Gauss-Newton method. If λ is large, expression (4) approximates steepest-descent method; otherwise, when λ

is small, the equation approximates Gauss-Newton method. J is the Jacobian matrix which is defined as follows:

$$J(w) = \begin{bmatrix} \frac{\partial e_1(w)}{\partial w_1} & \frac{\partial e_1(w)}{\partial w_2} & \cdots & \frac{\partial e_1(w)}{\partial w_n} \\ \frac{\partial e_2(w)}{\partial w_1} & \frac{\partial e_2(w)}{\partial w_2} & \cdots & \frac{\partial e_2(w)}{\partial w_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_N(w)}{\partial w_1} & \frac{\partial e_N(w)}{\partial w_2} & \cdots & \frac{\partial e_N(w)}{\partial w_n} \end{bmatrix}. \quad (5)$$

2.3. The Hybrid Optimization in BPNN. In the standard algorithm, the parameter α which is the momentum coefficient is static. It does not change in the whole iteration process, which makes the impacts of the momentum limited. Traditionally, the damping parameter λ becomes larger or smaller according to the performance. For example, it is decreased or increased by a factor 10 based on whether the performance is improved or not, respectively [17]. However, the merits of momentum item and LM algorithm have not been exerted sufficiently. So, in this paper, the two algorithms are optimized simultaneously and the weight equation for the proposed BPNN is as follows:

$$w(t+1) = w(t) + \Delta w(t) + \lambda(t) \times \alpha(t) \times \Delta w(t-1). \quad (6)$$

(1) The Adjustment of α. The proposed method considers the momentum coefficient dynamic. The α updates according to the error alteration and the former iteration. If the error reduces in this iteration, it means the previous weight update is beneficial to convergence; the researching direction is correct. Therefore, α should be bigger to encourage researching on this direction next time. Otherwise, the momentum coefficient should decline. The weight update can be formulated as follows:

$$\begin{aligned} \alpha(t) &= 1.2\alpha(t-1), \qquad E(t) < E(t-1), \\ \alpha(t) &= \alpha(t), \qquad E(t) = E(t-1), \\ \alpha(t) &= \frac{\alpha(t-1)}{1.2}, \qquad E(t) > E(t-1). \end{aligned} \quad (7)$$

While the momentum coefficient should not increase all the time and infinitely, when it is too big, it can influence the network; therefore, it should be reset as a decimal in the interval (0,1). The restriction rule of α is described as follows:

$$\begin{aligned} &\text{if } \alpha > 1 \\ &\alpha = 0.01 \\ &\text{end.} \end{aligned} \quad (8)$$

(2) The Adjustment of λ. λ could be regarded as the learning rate in the standard momentum method, while it is the damping parameter of LM algorithm. The principle of λ update is different from α's. For a given λ, if the error reduces, λ should decline to perform LM as the analogous Gauss-Newton advantage of fast convergence for local search. Otherwise, λ increases to make LM analogous steepest-descent algorithm for searching global optima. Too big or too small update of λ makes the network need long time to train. The disadvantage is similar to the learning rate in the standard BPNN. An appropriate update of λ is more efficient for convergence. The rule of the adjustment of λ is as follows:

$$\begin{aligned} \lambda(t) &= 1.2\lambda(t-1) \qquad E(t) > E(t-1), \\ \lambda(t) &= \lambda(t) \qquad E(t) = E(t-1), \\ \lambda(t) &= \frac{\lambda(t-1)}{1.2} \qquad E(t) < E(t-1). \end{aligned} \quad (9)$$

Through (4)–(9), the weight update is performed by the hybrid optimization of LM algorithm and momentum method.

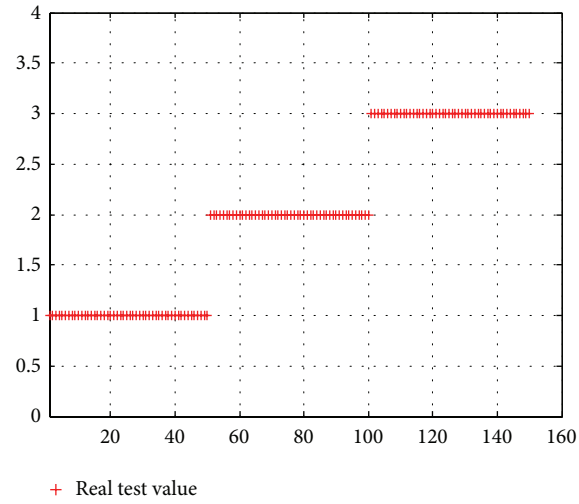


Figure 2: The classification of Fisher Iris data.

3. Validation on Pattern Recognition

In this paper, we suppose there is just one hidden layer. The hyperbolic tangent activation function is in the hidden neurons and linear activation function in the output neurons.

The number of the neurons in the hidden layer is important for convergence speed. Too many or too few neurons make the network need long time for training. Traditionally, it is set by personal experience. In this paper, the number of the neurons in the hidden layer depends on the empirical formula which is shown as follows:

$$\text{hn} = \sqrt{\text{in} + \text{on}} + \sigma. \quad (10)$$

in and on are the number of the neurons in the input layer and output layer, respectively. σ is a constant in interval $(1, 10)$. in and on are determined by the dimensionality of the input vector and the output vector. For example, the input and output dataset have 4 attributes and 1 attribute, respectively;

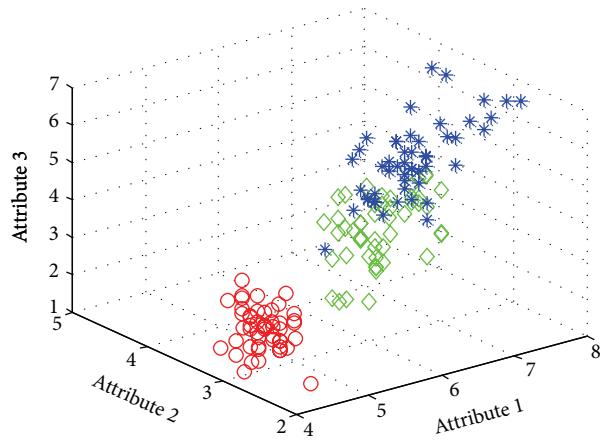
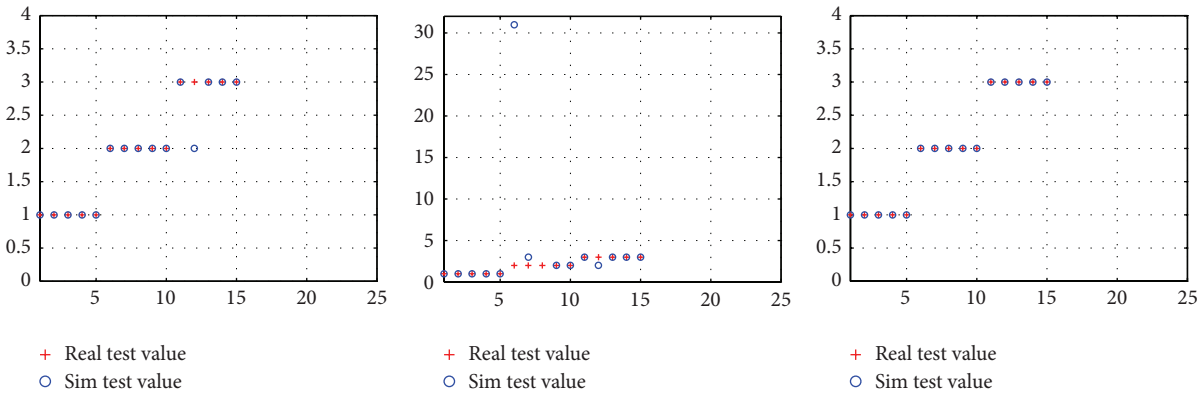
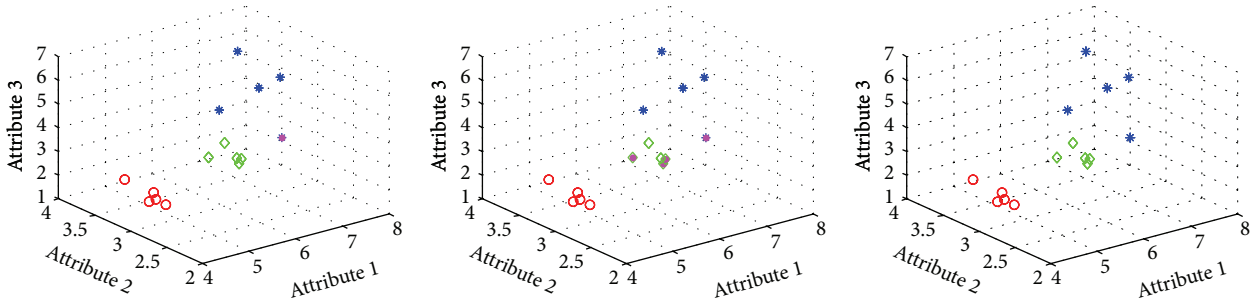


Figure 3: Classification map in the first three attributes of Fisher Iris data.



(a) ELM-BPNN (b) NLM-BPNN (c) Proposed LM-BPNN

Figure 4: Classification results compared with the real labels in 2D of Fisher Iris data.



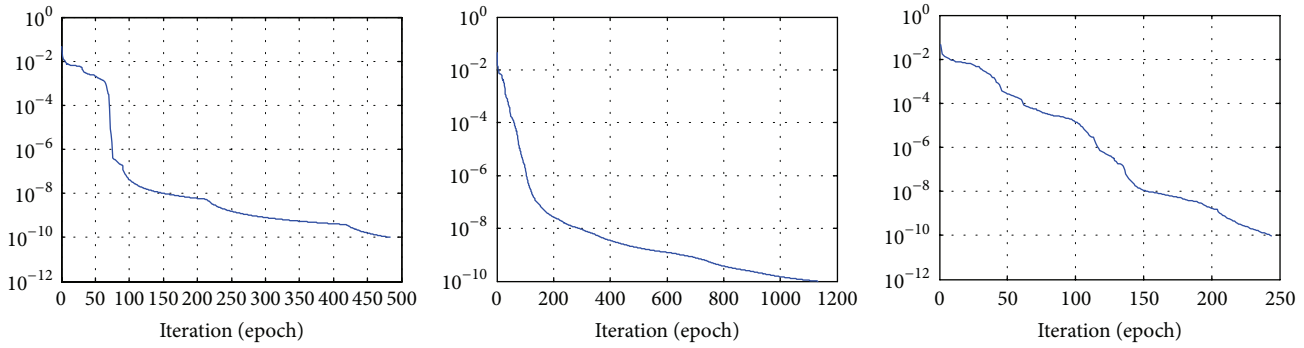
(a) ELM-BPNN (b) NLM-BPNN (c) Proposed LM-BPNN

Figure 5: Classification errors mapping in the first three attributes of Fisher Iris data.

the numbers of neurons in the input layer and output layer are 4 and 1.

To validate our proposed method on pattern recognition, Fisher Iris data and wine data which are provided by UCI are used. Fisher Iris data and wine data should be normalized and the "mapminmax" function provided by Matlab toolbox is used. All the parameters in the compared three methods are the same. The maximum iteration step is 50000000, the training error goal is 1e-10, and the initial λ is 1. In our improved program, the initial momentum coefficient α is 0.01. The adjustment of α and λ is according to (7)–(9).

3.1. The Fisher Iris Data Example. The Fisher Iris data is one of the most famous databases in the pattern recognition works. The dataset includes 3 classes of 50 instances each. These classes refer to "setosa," "versicolor," and "virginica" which are labeled as "1", "2," and "3," respectively. The attributes contain sepal length, sepal width, petal length, and petal width.



(a) ELM-BPNN (b) NLM-BPNN (c) Proposed LM-BPNN

Figure 6: The errors in the iteration process of Fisher Iris data.

Table 3: The results of different training methods for wine data.

Real labels	Preprocessed output			Postprocessed output		
	ELM-BPNN	NLM-BPNN	Proposed LM-BPNN	ELM-BPNN	NLM-BPNN	Proposed LM-BPNN
1	1.00009249	1.00020181	1.00008609	1	1	1
1	0.99999165	0.99970850	1.00011684	1	1	1
1	1.00004976	1.00013716	1.00007315	1	1	1
1	1.00009044	1.06945969	1.05066960	1	1	1
1	0.99992444	1.01434834	1.01257733	1	1	1
1	0.99999324	1.00008633	1.00014440	1	1	1
2	2.00031311	2.00049736	2.00020179	2	2	2
2	1.80813014	2.02180286	2.03638972	2	2	2
2	2.00003265	1.99999689	1.99998956	2	2	2
2	1.99997582	2.00001218	2.00002534	2	2	2
2	0.40518282	1.99552685	1.98939772	**0**	2	2
2	1.99999567	1.99998810	1.99994678	2	2	2
2	1.99999881	2.00000198	2.00001719	2	2	2
2	1.99992264	2.00011454	1.99995018	2	2	2
3	2.98399414	2.68343848	2.52157921	3	3	3
3	2.99991935	3.00078223	2.99958012	3	3	3
3	3.00001821	2.99949485	3.00003921	3	3	3
3	3.00001865	2.99820215	2.99558228	3	3	3
3	2.56607392	2.32976636	2.77938018	3	**2**	3

Table 4: Comparison of the three methods on benchmark datasets for wine data.

	Iteration steps	Iteration time	Accuracy rate
ELM-BPNN	162	0.990234 seconds	94.74%
NLM-BPNN	69	0.382775 seconds	94.74%
Proposed LM-BPNN	101	0.504761 seconds	100%

The unit is centimeter. The whole data is divided into 2 parts; one is for training and the other for testing. The data classification and the corresponding classification map in the first three attributes are shown in Figures 2 and 3.

The output of the test data should be processed in order to be compared with the real value. Because of calculation, the real output of the network may be nonintegral. So "round" function which is provided by Matlab toolbox is used to process the real output. For simplifying the items, ELM-BPNN replaces El-Alfy's LM training BPNN, and NLM-BPNN denotes the LM-BPNN improved by Nørgaard. The compared results are shown in Table 1.

The classification results compared with the real labels in 2D and mapping in the first three attributes are shown in Figures 4 and 5.

In these 3D figures, the points with pink color are the wrong recognition. From the above figures and table, the proposed method has better performance in correct rate.

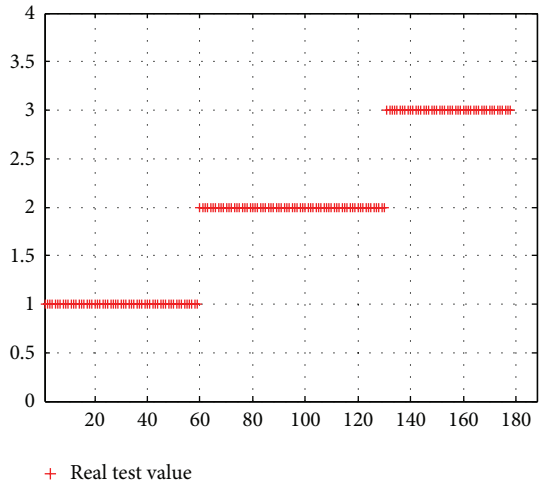


Figure 7: The total figure classification of wine data.

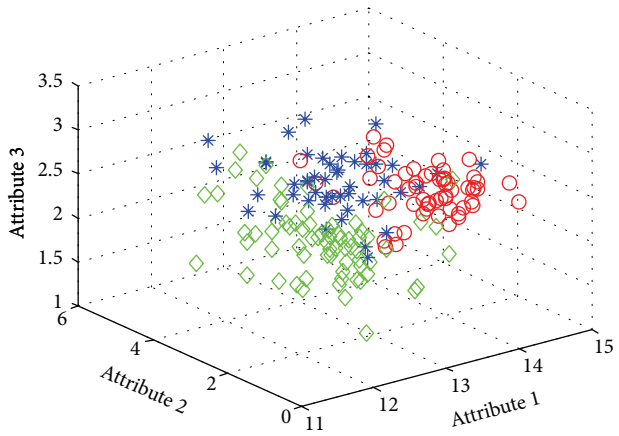


Figure 8: Classification map in the first three attributes of wine data.

The errors of iteration process are shown in Figure 6.

The statistics of iteration steps, iteration time, and accuracy are listed in Table 2.

3.2. Wine Data Example. The wine data is provided by UCI. The data is the results of a chemical analysis of wines grown in the same region in Italy but derived from 3 different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. These attributes are (1) alcohol, (2) malic acid, (3) ash, (4) alkalinity of ash, (5) magnesium, (6) total phenols, (7) flavanoids, (8) nonflavonoid phenols, (9) proanthocyanins, (10) color intensity, (11) hue, (12) OD280/OD315 of diluted wines, and (13) proline. The numbers of instances are Class 1 with 59, Class 2 with 71, and Class 3 with 48. Among these instances, 19 instances are regarded as the test set, and the rest are the training set. The classification and the corresponding map in the first three attributes are shown in Figures 7 and 8.

Similarly, the compared outputs of different networks are operated by "round" function, so the results are listed in Table 3.

The classification errors in 2D and mapping in the first three attributes are shown in Figures 9 and 10.

The errors of iteration process are shown in Figure 11.

The statistics of iteration steps, iteration time, and accuracy are listed in Table 4.

By comparison, the proposed LM-BPNN gets more accurate results than ELM-BPNN and NLM-BPNN. In issue of the training speed, the proposed LM-BPNN is faster than ELM training BPNN.

4. Validation on Prediction

In the aspect of prediction, Dunnhumby's Shopper Challenge database is used to compare the performances of the three methods. The dataset consists of details of every visit made by 100,000 customers over a year from April 2010 to March 31, 2011. Each visit is stamped with the date and the customer's spend in that visit. The challenge is to predict the visit_date and visit_spend of this next visit for each customer_id. But in this section, we just predict the visit_spend during April to December in 2010.

In order to be trained in the network, the data should be limited in the interval $(0, 1)$. With the consideration of the external character of prediction, all the data is divided by an enough big constant, $1e + 6$. Besides, two performance indexes are given to assess the trained networks:

$$P1 = \max\left(\operatorname{abs}\left(\operatorname{error}_m\right)\right), \tag{11}$$

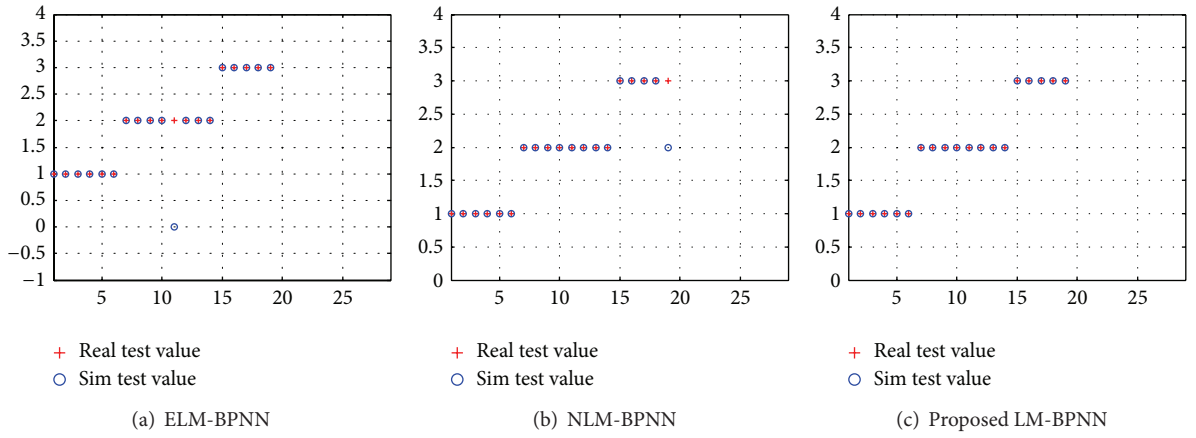
$$P2 = \frac{\sum_{m=1}^{M} \operatorname{error}_m^2}{2M}, \tag{12}$$

$$\operatorname{error}_m = y_m - d_m. \tag{13}$$

Expressions (11) and (12) could evaluate the deviation from the real value. abs means the absolute values. So the basic information of test dataset and performances of the three networks are given in Table 5.

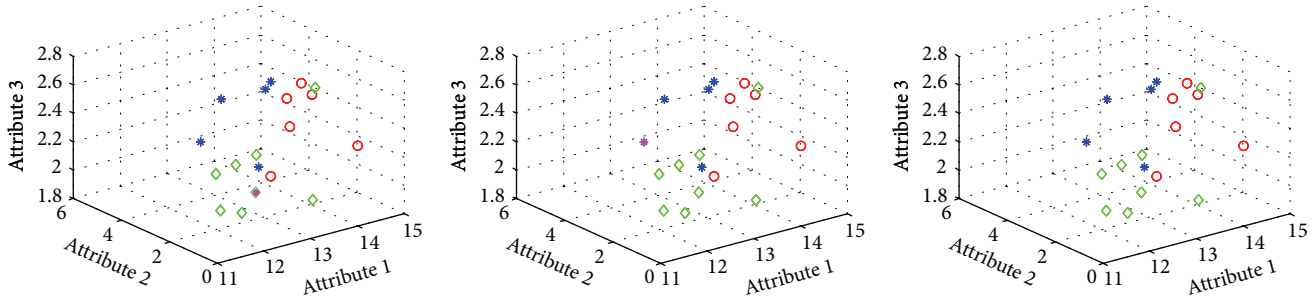
By comparing the results from Tables 5 and 6, the proposed LM-BPNN has a better performance than ELM-BPNN in the aspect of prediction. Two performance indexes could illustrate that the proposed method is more stable.

The results from pattern recognition (in Section 3) and prediction (in Section 4) show the proposed BPNN performs better than the other two methods. It could finish calculation in less time and less iteration step with higher accuracy. Because it uses the error information and efficiency of pervious iteration sufficiently. Momentum item could overcome the oscillation and make the iteration in a less error direction, simultaneously, LM algorithm is a good balance of steepest-descent method and Gauss-Newton method. Therefore, the proposed method not only weakens the oscillation but also converges into the optimum in less iteration steps.



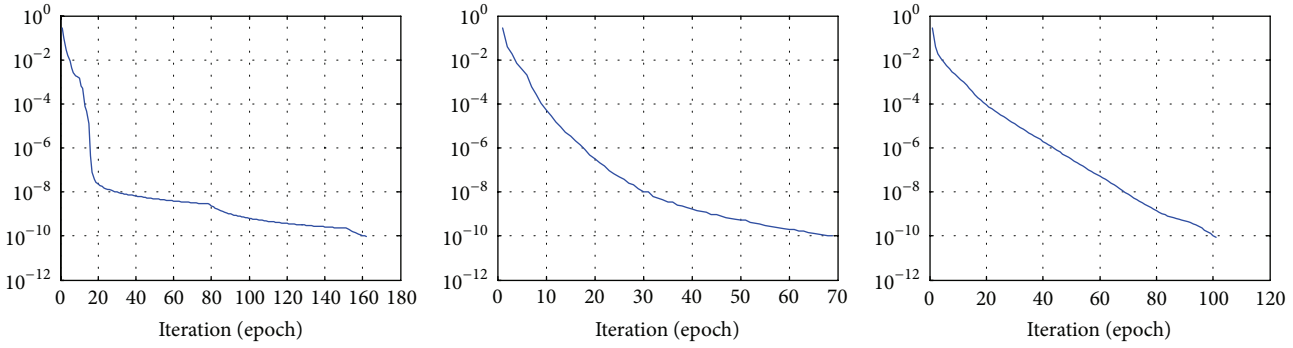
(a) ELM-BPNN (b) NLM-BPNN (c) Proposed LM-BPNN

Figure 9: Classification errors in 2D of wine data.



(a) ELM-BPNN (b) NLM-BPNN (c) Proposed LM-BPNN

Figure 10: Classification errors mapping in the first three attributes of wine data.



(a) ELM-BPNN (b) NLM-BPNN (c) Proposed LM-BPNN

Figure 11: The errors of iteration process of wine data.

5. Conclusions

In this paper, a joint optimization of momentum item and Levenberg-Marquardt is proposed to train BPNN. Its performance is compared with ELM-BPNN and NLM-BPNN in the aspect of pattern recognition and prediction. The validated data is provided by UCI and public challenge. The results proved that the proposed LM-BPNN has a better performance. Although the proposed method shows its better performance, there is still much work to do. For example, the

Table 5: The basic information of test dataset and performances of the three networks.

	Number of tuples	ELM-BPNN		NLM-BPNN		Proposed LM-BPNN	
		$P1$	$P2$	$P1$	$P2$	$P1$	$P2$
Apr.	83853	0.001648	$2.457953e-007$	0.001637	$2.423426e-007$	0.001637	$2.423272e-007$
May	87518	0.001650	$2.489964e-007$	0.001363	$1.681887e-007$	0.001356	$1.664438e-007$
Jun.	84380	0.001594	$2.321456e-007$	0.001573	$2.260457e-007$	0.001573	$2.260597e-007$
Jul.	87222	0.001491	$2.002518e-007$	0.001418	$1.806153e-007$	0.001413	$1.792959e-007$
Aug.	82798	0.001621	$2.415615e-007$	0.001303	$1.544337e-007$	0.001295	$1.526117e-007$
Sep.	82473	0.001649	$2.469237e-007$	0.001347	$1.631628e-007$	0.001336	$1.603953e-007$
Oct.	85797	0.001534	$2.153020e-007$	0.001504	$2.067444e-007$	0.001504	$2.067388e-007$
Nov.	85131	0.001469	$1.968304e-007$	0.001440	$1.890057e-007$	0.001436	$1.878854e-007$
Dec.	81871	0.001682	$2.544444e-007$	0.001682	$2.543521e-007$	0.001682	$2.543497e-007$

Table 6: The mean statistic results of the performance for the three compared networks.

	$P1$	$P2$
ELM-BPNN	0.001592202753327	$2.311075035045326e-007$
NLM-BPNN	0.001473341461652	$1.981014712133217e-007$
Proposed LM-BPNN	0.001469453577525	$1.971251110370899e-007$

adjustment strategy of λ should be more adaptive. We also found the initial λ could influence the network's convergence speed, so how to select appropriate λ is worth researching.

Conflict of Interests

The authors declare that they have no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank anonymous referees for their remarkable comments and great support by Key Project supported by National Science Foundation of China (51035008), the Natural Science Foundation Project of Chongqing (CSTC, 2009BB3365), and the Fundamental Research Funds for the State Key Laboratory Of Mechanical Transmission, Chongqing University (SKLMT-ZZKT-2012 MS 02).

References

[1] X. J. Jing and L. Cheng, "An optimal PID control algorithm for training feedforward neural networks," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 6, pp. 2273–2283, 2013.

[2] S. Roy, "Factors influencing the choice of a learning rate for a backpropagation neural network," in *Proceedings of the IEEE World Congress on Computational Intelligence, IEEE International Conference on Neural Networks*, pp. 503–507, June-July 1994.

[3] G. J. Song, J. L. Zhang, and Z. L. Sun, "The research of dynamic change learning rate strategy in BP neural network and application in network intrusion detection," in *Proceedings of the 3rd International Conference on Innovative Computing Information and Control (ICICIC '08)*, p. 513, June 2008.

[4] Y. Li, Y. Fu, H. Li, and S.-W. Zhang, "The improved training algorithm of back propagation neural network with selfadaptive learning rate," in *Proceedings of the International Conference on Computational Intelligence and Natural Computing (CINC '09)*, pp. 73–76, June 2009.

[5] T. Zhang, C. L. P. Chen, C. H. Wang, and S. C. Tam, "A new dynamic optimal learning rate for a two-layer neural network," in *Proceedings of the International Conference on System Science and Engineering (ICSSE '12)*, pp. 55–59, June-July 2012.

[6] M. M. Hasan, A. Rahaman, M. Talukder, M. Islam, M. M. S. Maswood, and M. M. Rahman, "Neural network performance analysis using hanning window function as dynamic learning rate," in *Proceedings of the International Conference on Informatics, Electronics & Vision (ICIEV '13)*, pp. 1–5, May 2013.

[7] X. Jing, "Robust adaptive learning of feedforward neural networks via LMI optimizations," *Neural Networks*, vol. 31, pp. 33–45, 2012.

[8] C.-C. Yu and B.-D. Liu, "A backpropagation algorithm with adaptive learning rate and momentum coefficient," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '02)*, pp. 1218–1223, May 2002.

[9] J. Wang, J. Yang, and W. Wu, "Convergence of cyclic and almost-cyclic learning with momentum for feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 22, no. 8, pp. 1297–1306, 2011.

[10] W. Wu, N. Zhang, Z. Li, L. Li, and Y. Liu, "Convergence of gradient method with momentum for back-propagation neural networks," *Journal of Computational Mathematics*, vol. 26, no. 4, pp. 613–623, 2008.

[11] A. Likas and A. Stafylopatis, "Training the random neural network using quasi-Newton methods," *European Journal of Operational Research*, vol. 126, no. 2, pp. 331–339, 2000.

[12] W. W. Hager and H. Zhang, "A new conjugate gradient method with guaranteed descent and an efficient line search," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 170–192, 2005.

[13] I. Güler and E. D. Übeyli, "A recurrent neural network classifier for Doppler ultrasound blood flow signals," *Pattern Recognition Letters*, vol. 27, no. 13, pp. 1560–1571, 2006.

[14] P. S. Lande and A. S. Gadewar, "Application of artificial neural networks in prediction of compressive strength of concrete by

using ultrasonic pulse velocities," *IOSR Journal of Mechanical and Civil Engineering*, vol. 3, no. 1, pp. 34–42, 2012.

[15] M.-Q. Ling and W.-W. Liu, "Research on intrusion detection systems based on Levenberg-Marquardt algorithm," in *Proceedings of the 7th International Conference on Machine Learning and Cybernetics (ICMLC '08)*, pp. 3684–3688, July 2008.

[16] X.-P. Wang and Y.-S. Huang, "Predicting risks of capital flow using artificial neural network and levenberg marquardt algorithm," in *Proceedings of the 7th International Conference on Machine Learning and Cybernetics (ICMLC '08)*, pp. 1353–1357, July 2008.

[17] E. S. M. El-Alfy, "Detecting pixel-value differencing steganography using Levenberg-Marquardt neural network," in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM '13)*, pp. 160–165, April 2013.

[18] G. H. Nguyen, A. Bouzerdoum, and S. L. Phung, "A supervised learning approach for imbalanced data sets," in *Proceedings of the 19th International Conference on Pattern Recognition (ICPR '08)*, pp. 1–4, December 2008.

[19] M. Nørgaard, "Neural network based system identification toolbox," Tech. Rep., Department of Automation, Technical University of Denmark, 2000.

[20] O. Fontenla-Romero, B. Guijarro-Berdiñas, B. Pérez-Sánchez, and A. Alonso-Betanzos, "A new convex objective function for the supervised learning of single-layer neural networks," *Pattern Recognition*, vol. 43, no. 5, pp. 1984–1992, 2010.

[21] V. V. Phansalkar and P. S. Sastry, "Analysis of the back-propagation algorithm with momentum," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 505–506, 1994.

[22] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.