International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 20, No. 2 (2006) 129–142

A MODIFIED LEARNING ALGORITHM INCORPORATING ADDITIONAL FUNCTIONAL CONSTRAINTS INTO NEURAL NETWORKS*

FEI HAN† and XU-QIN LI‡

Hefei Institute of Intelligent Machines
Chinese Academy of Sciences
Hefei, Anhui 230031, P. R. China

†*Department of Automation*
University of Science and Technology of China
Hefei, Anhui 230027, P. R. China
†*hanfei1976@iim.ac.cn*
‡*xqli@iim.ac.cn*

MICHAEL R. LYU

Computer Science and Engineering Department
The Chinese University of Hong Kong
Shatin, Hong Kong

TAT-MING LOK

Information Engineering Department
The Chinese University of Hong Kong
Shatin, Hong Kong

In this paper, a modified learning algorithm to obtain better generalization performance is proposed. The cost terms of this new algorithm are selected based on the second-order derivatives of the neural activation at the hidden layers and the first-order derivatives of the neural activation at the output layer. It can be guaranteed that in the course of training, the additional cost terms for this algorithm can penalize both the input-to-output mapping sensitivity and the high frequency components to obtain better generalization performance. Finally, theoretical justifications and simulation results are given to verify the efficiency and effectiveness of the proposed learning algorithm.

Keywords: Feedforward neural networks; generalization performace; constrained learning algorithm; time series prediction.

*This work was supported by the National Science Foundation of China (Nos. 60472111, 30570368 and 60405002).

1. Introduction

Most traditional learning algorithms with feedforward neural networks (FNN) are to use the sum-of-square error criterion to derive the updated formulae. However, these learning algorithms do not considered the network structure and the involved problem properties, thus their capabilities are limited.[13] In order to obtain better generalization capability,[1,3,4] many constrained learning algorithms that incorporate additional functional constraints into neural networks have been proposed in literature.[5–10,14–16]

In literature,[12] two learning algorithms were proposed that are referred to as Hybrid-I method and Hybrid-II method, respectively. The Hybrid-I algorithm incorporates the first-order derivatives of the neural activation at hidden layers into the sum-of-square error cost function to reduce the input-to-output mapping sensitivity. On the other hand, the Hybrid-II algorithm incorporates the second-order derivatives of the neural activation at hidden layers and output layer into the sum-of-square error cost function to penalize the high frequency components in training data. Nevertheless, all the above learning algorithms can almost improve the generalization performance to some degree, but do not show the best generalization performance.

In this paper, a new modified learning algorithm based on Hybrid-I and Hybrid-II algorithms is proposed. The additional cost terms of the new algorithm are selected based on the first-order derivatives of the neural activation at the output layer and the second-order derivatives of the neural activation at the hidden layers. This new algorithm inherits the features from the original Hybrid-I and Hybrid-II algorithms. Moreover, through experiments, it can be found that the generalization performance for this modified algorithm is better than the one for the original Hybrid ones.

2. The New Modified Learning Algorithm

Considering an FNN with one input layer, hidden layers, and one output layer, the units in each layer apart from the input layer receive the inputs from all units in the previous layer. For simplicity, the same activation function for all neurons at all layers, i.e. tangent sigmoid transfer function is adopted:

$$f(x) = (1 - \exp(-2x))/(1 + \exp(-2x)). \tag{1}$$

It can be deduced that this activation function has the following property:

$$f''(x) = -2f(x)f'(x). \tag{2}$$

Before presenting input-to-output sensitivity, the following mathematical notation is first created. Assume that x_k and y_i denote the kth element of the input vector and the ith element of the output vector, respectively; $w_{j_l j_{l-1}}$ denotes the synaptic weight from the j_lth hidden neuron at the lth layer to the j_{l-1}th hidden neuron at $(l-1)$th layer; $w_{ij_{L-1}}$ denotes the synaptic weight from the ith neuron at output layer to the j_{L-1}th hidden neuron at $(L-1)$th layer; $w_{j_1 k}$ denotes the synaptic weight from the j_1th hidden neuron at the first layer to the kth element of the input vector; $f_l'(\cdot)$ is the derivative of the sigmoid function $f_l(\cdot)$ at lth layer; $h_{j_l} = f_l(\hat{h}_{j_l})$ is the activation function of the j_lth element at the lth layer with $\hat{h}_{j_l} = \sum_{j_{l-1}} w_{j_l j_{l-1}} h_{j_{l-1}}$. The t_i and y_i denote the target and actual output values of the ith neuron at output layer, respectively; N_l denotes the number of neurons at the lth layer.

To obtain better generalization performance than Hybrid-I and Hybrid-II algorithms, a new cost function for an L-layered FNN containing the additional output layer penalty term and the weights decay term at the hidden layers is defined as follow:

$$E = \frac{1}{N} \sum_{S=1}^{N} E^S \tag{3}$$

where

$$E^S = \frac{1}{2N_L} \sum_{j_L=1}^{N_L} (t_{j_L}^S - y_{j_L}^S)^2 + \sum_{l=1}^{L-1} \gamma_l E_h^{lS} + \frac{\gamma_L}{N_L} \sum_{j_L=1}^{N_L} f'(\hat{h}_{j_L}^{LS}) \tag{4}$$

and

$$E_h^{lS} = \frac{1}{2N_l} \sum_{j_l=1}^{N_l} f'(\hat{h}_{j_l}^{lS}) \left[\sum_{j_{l-1}=1}^{N_{l-1}} (w_{j_l j_{l-1}}^S)^2 \right]. \tag{5}$$

The cost function E^S denotes the corresponding cost function for the Sth stored pattern. The second term on the right-hand side of Eq. (4) is a kind of weights decay term; the third term on the right-hand side of Eq. (4) denotes the additional output layer penalty term at the output layer; the gains γ_l and γ_L represent the relative significance among the cost terms; N denotes the number of stored patterns.

The network is trained by a steepest-descent error minimization algorithm, the synaptic weight update for Sth stored pattern becomes:

$$\Delta w_{j_l j_{l-1}}^S = -\eta_l \frac{\partial E^S}{\partial w_{j_l j_{l-1}}^S} = \eta_l \delta_{j_l}^S h_{j_{l-1}}^S - \eta_l \frac{\gamma_l}{N_l} w_{j_l j_{l-1}}^S f'(\hat{h}_{j_l}^S) \quad l = 1, 2, \ldots, L-1 \tag{6}$$

$$\Delta w_{j_L j_{L-1}}^S = -\eta_L \frac{\partial E^S}{\partial w_{j_L j_{L-1}}^S} = \eta_L \delta_{j_L}^S h_{j_{L-1}}^S \tag{7}$$

where $\delta_{j_l}^S$ denotes the negative derivative of the cost E^S to $\hat{h}_{j_l}^S$ at lth layer.

The negative derivative of the cost E^S to $\hat{h}_{j_l}^S$ at the hidden layer, i.e. $\delta_{j_l}^S$, can be computed by back-propagation style as follows:

$$\begin{aligned} \delta_{j_l}^S = -\frac{\partial E^S}{\partial \hat{h}_{j_l}^S} &= \sum_{j_{l+1}=1}^{N_{l+1}} \delta_{j_{l+1}}^S w_{j_{l+1} j_l}^S f'(\hat{h}_{j_l}^S) \\ &\quad - \frac{\gamma_l}{2N_l} f''(\hat{h}_{j_l}^S) \sum_{j_{l-1}=1}^{N_{l-1}} (w_{j_l j_{l-1}}^S)^2 \qquad l = 1, \ldots, L-1. \end{aligned} \tag{8}$$

The negative derivative of the cost E^S to the $\hat{h}_{j_L}^S$ at the hidden layer, i.e. $\delta_{j_L}^S$, can be computed by back-propagation style as follows:

$$\delta_{j_L}^S = -\frac{\partial E^S}{\partial \hat{h}_{j_L}^S} = \frac{1}{N_L} f'(\hat{h}_{j_L}^S)(t_{j_L}^S - y_{j_L}^S) - \frac{\gamma_L}{N_L} f''(\hat{h}_{j_L}^S). \tag{9}$$

3. Theoretical Analysis for this Modified Learning Algorithm

According to literature,[11] for an L-layered feedforward neural network, the sensitivity for y_i to x_k can be defined as:

$$\frac{\partial y_i}{\partial x_k} = \sum_{j_1,\ldots,j_{L-1}} w_{ij_{L-1}} w_{j_{L-1}j_{L-2}} \cdots w_{j_1 k} f'_L(\hat{y}_i) f'_{L-1}(\hat{h}_{j_{L-1}}) \cdots f'_1(\hat{h}_{j_1}). \tag{10}$$

From this equation, it can be deduced that while $\hat{h}_{j_l}$ becomes bigger, the derivative $f'(\hat{h}_{j_l})$ may become smaller sharply. As a result, the low input-to-output sensitivity will be achieved. For simplicity, consider a single-layered neural network with tangent sigmoid neuron. If the input vector x is modified by Δx, the change Δy_i, at the ith output neuron may be approximated as:

$$\Delta y_i = y_i(x + \Delta x) - y_i(x) \approx \sum_k \Delta x_k \frac{\partial y_i}{\partial x_k} = \sum_k w_{ik} f'(\hat{y}_i) \Delta x_k. \tag{11}$$

Consequently, $\Delta y_i / y_i$ can be computed as follow:

$$\frac{\Delta y_i}{y_i} = \frac{\sum_k w_{ik} f'(\hat{y}_i) \Delta x_k}{f(\hat{y}_i)} = \frac{f'(\hat{y}_i)}{f(\hat{y}_i)} \sum_k w_{ik} x_k \frac{\Delta x_k}{x_k} = \frac{f'(\hat{y}_i)\hat{y}_i}{f(\hat{y}_i)} \frac{\Delta x_k}{x_k} \tag{12}$$

$g(\hat{y}_i)$ is defined as:

$$g(\hat{y}_i) = \frac{f'(\hat{y}_i)\hat{y}_i}{f(\hat{y}_i)} = \frac{4\hat{y}_i}{\exp(2\hat{y}_i) - \exp(-2\hat{y}_i)}. \tag{13}$$

The $g(\hat{y}_i)$ has generally a maximum at $\hat{y}_i = 0$ and two minima at $\hat{y}_i = \pm\infty$. When the value of $\hat{y}_i$ becomes larger, the value of $g(\hat{y}_i)$ becomes exponentially decreasing. Consequently, a larger value of $\hat{y}_i$ brings on better generalization capability. Apparently, it can be seen that the $g(\hat{y}_i)$ and $f'(\hat{y}_i)$ have similar functional forms according to literature,[12] thus, the second additional cost term in Eq. (4) can result in better generalization performance. As far as an L-layered feedforward neural network is concerned, the same result can be obtained. According to the above results, the network obtains lower input-to-output sensitivity in the first hidden layer, and it means that the changes of the input vector will lead to smaller changes of the values of output vector in the first hidden layer. In a similar way, the smaller changes in the first hidden layer will result in much smaller changes of the values of the output vector in the second hidden layer, because the output vector in the first hidden layer is the input vector for the second hidden layer. The remaining layers may be deduced by analogy. Hence, it can be easily seen that the values of the output vector in the output layer have much smaller changes although the input vector changes a lot.

From a Bayesian perspective, all the additional cost functions designed for the above constrained learning algorithms can be interpreted as a negative logarithm of the prior probability distribution of weights.[2,16] In order to obtain good generalization capability, this new hybrid algorithm tries to reduce the network complexity by introducing weight decay term. For the weight decay form, $E_c(w) = \frac{1}{2}\sum_i w_i^2$, it can be derived by taking negative logarithm on the Gaussian distribution of weights. This weight decay method penalizes large weights and rewards small weights, but it decays weights at the same rates regardless of its sizes.[12] For simplicity, the weight decay terms in the new hybrid learning algorithm, i.e. the first additional cost term in Eq. (4), can be simplified as: $E_c(w) = \frac{f'}{2}\sum_i w_i^2$. It favors large weights only when the corresponding hidden activation is saturated. The derivative of hidden activation, f', can be regarded as a scaling parameter to control whether weights are scaled up or down during learning process.

According to the above results, it can be concluded that in the course of training, the proposed learning algorithm can obtain better generalization performance by penalizing both the input-to-output mapping sensitivity and high frequency components in training data.

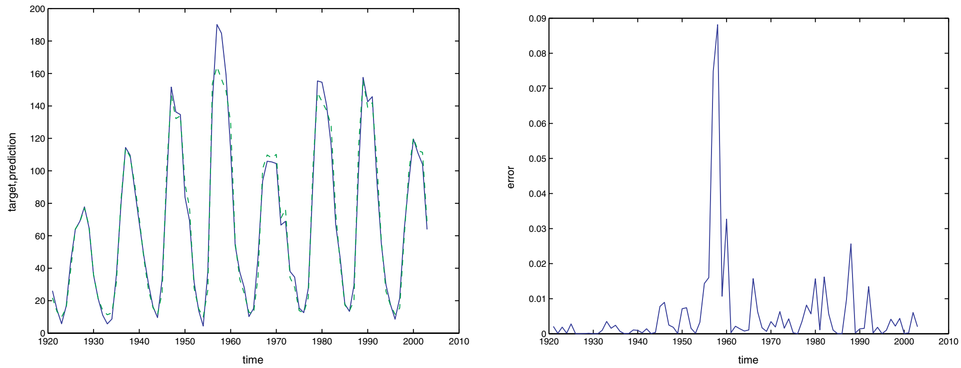
4. Experimental Results and Discussion

To demonstrate the improved generalization capability of the proposed modified learning algorithm, in the following, experiments with two real-world benchmarks of sunspot time series and chaotic laser pulsation data will be done. The latter is obtained from Santa Fe competition data set A.

4.1. *Single-step and iterative-step prediction for sunspot time series*

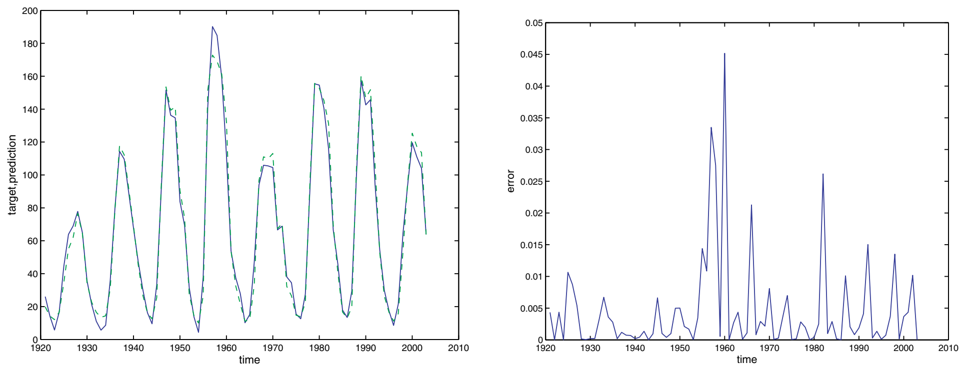
To compare the generalization ability of the proposed learning algorithm with one of the two original Hybrid ones, a (12-8-1)-sized network to solve the sunspot time series single-step and iterative-step prediction is used. Assume that sunspot data from the year 1700 to 1920 are used as training set. The data after the year 1920 are used as testing set. In addition, as for single-step prediction, this testing data is divided into four intervals, that is, from the year 1921 to 1955, 1956 to 1979, 1980 to 2003, and finally 1921 to 2003. As a result, the single-step prediction results are shown in Figs. 1–4 for BP algorithm, Hybrid-I algorithm, Hybrid-II algorithm and the proposed new learning algorithm, respectively. In the meantime, the iterative prediction results are shown in Figs. 5–8 for the above four learning algorithms.

In order to statistically compare the prediction accuracies for sunspot data with the above four algorithms, an experiment is done fifty times for each algorithm and



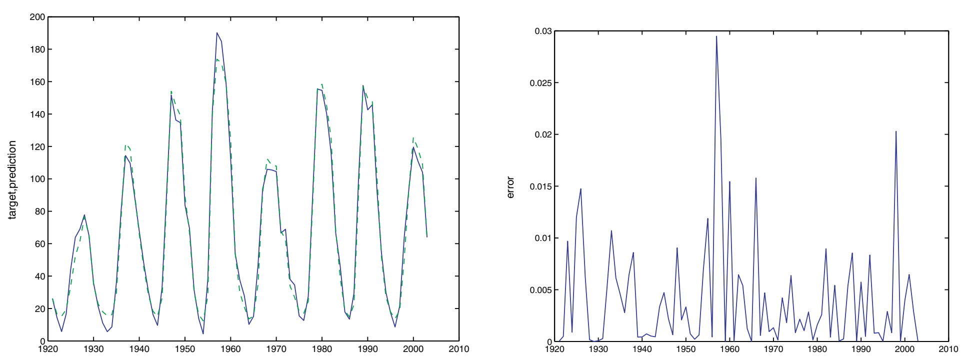
(a) The predicted values (b) The predicted errors

Fig. 1. Results with single-step prediction for sunspot time series by using BP algorithm.



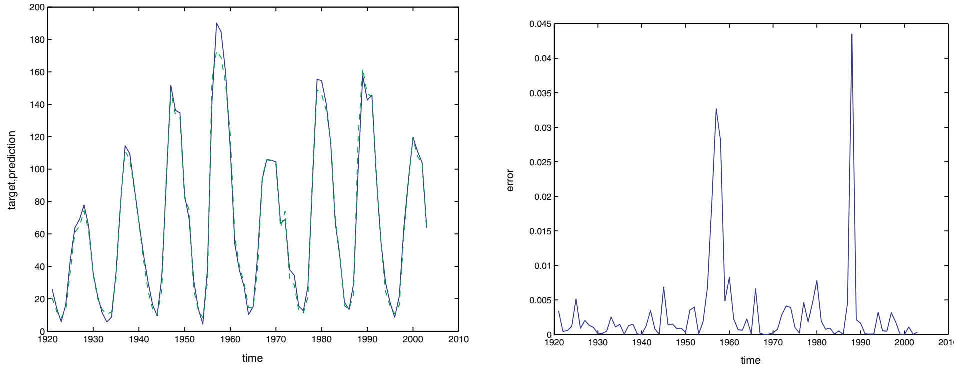
(a) The predicted values (b) The predicted errors

Fig. 2. Results with single-step prediction for sunspot time series by using Hybrid-I algorithm.



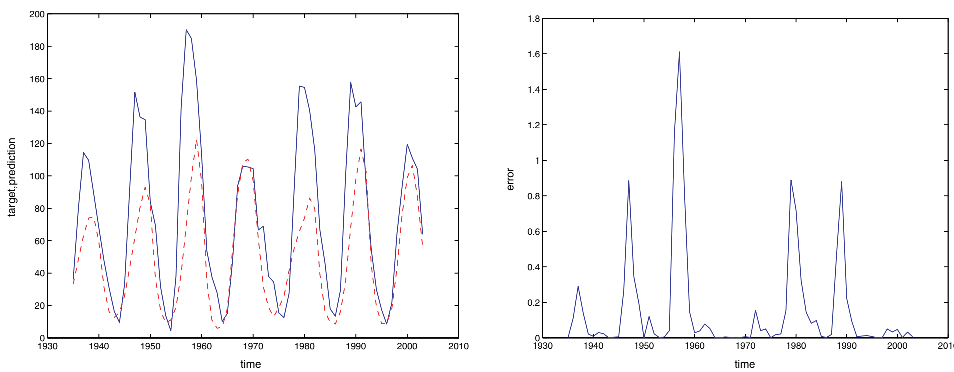
(a) The predicted values (b) The predicted errors

Fig. 3. Results with single-step prediction for sunspot time series by using Hybrid-II algorithm.



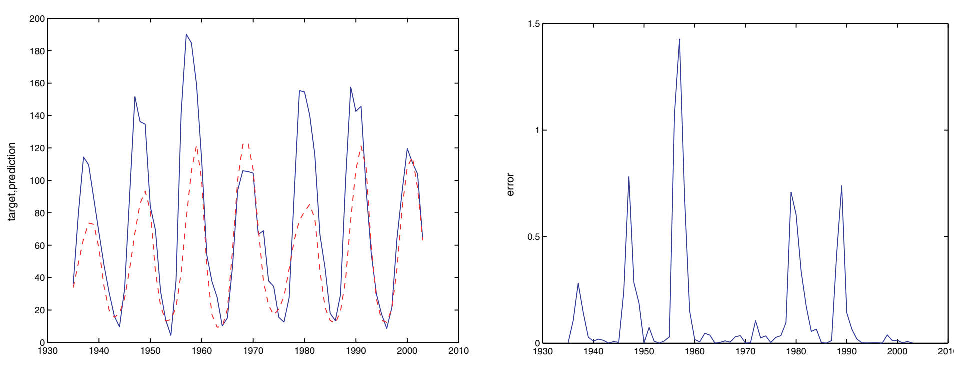
(a) The predicted values (b) The predicted errors

Fig. 4. Results with single-step prediction for sunspot time series by using the new learning algorithm.



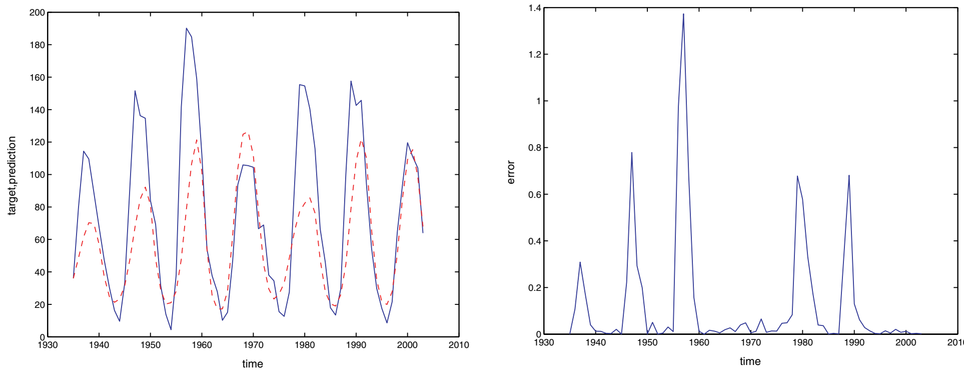
(a) The predicted values (b) The predicted errors

Fig. 5. Results with iterative-step prediction for sunspot time series by using BP algorithm.



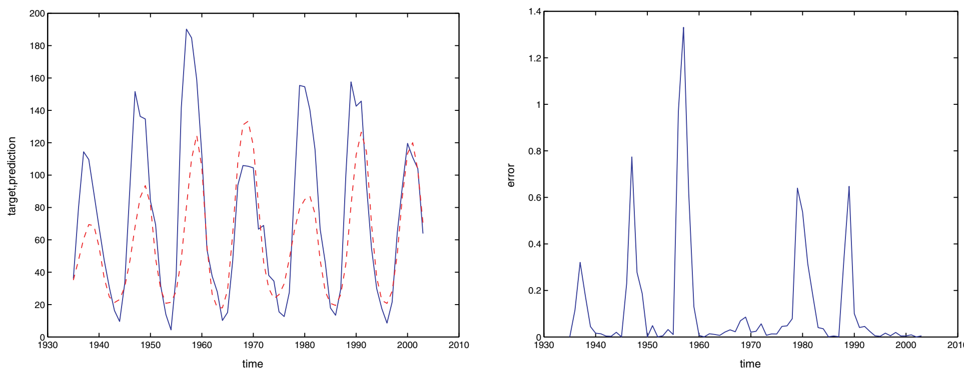
(a) The predicted values (b) The predicted errors

Fig. 6. Results with iterative-step prediction for sunspot time series by using Hybrid-I algorithm.



(a) The predicted values (b) The predicted errors

Fig. 7. Results with iterative-step prediction for sunspot time series by using Hybrid-II algorithm.



(a) The predicted values (b) The predicted errors

Fig. 8. Results with iterative-step prediction for sunspot time series by using the new learning algorithm.

Table 1. The average values of mean squared errors of single-step prediction for the sunspot time series data for fifty times by four algorithms.

LA	Training 1700–1920	Testing 1921–1955	Testing 1956–1979	Testing 1980–2003	Testing 1921–2003
BP	0.00090	0.00053	0.0095	0.0090	0.0055
Hybrid-I	0.00210	0.00037	0.0056	0.0040	0.0045
Hybrid-II	0.00210	0.00046	0.0050	0.0044	0.0042
New LA	0.00074	0.00030	0.0145	0.0129	0.0031

then its average accuracy value calculated. The corresponding results are summarized in Tables 1 and 2 for single-step prediction and iterative-step prediction. From these results, it can be seen apparently that the proposed learning algorithm has better generalization capability than the BP algorithm as well as the two original

Table 2. The average values of mean squared errors of iterative-step prediction for the sunspot time series data for fifty times by four algorithms.

LA	Training (1700–1920)	Testing (1921–2003)
BP	0.0391	0.1881
Hybrid-I	0.0390	0.1380
Hybrid-II	0.0401	0.1330
New LA	0.0385	0.1285

hybrid algorithms, because the mean squared errors of the new algorithm for the testing data set is smaller than the ones for the other three learning ones.

Below, the effects of the four parameters, η_1, η_2, γ_1 and γ_2, with the new modified hybrid learning algorithm for single-step prediction performance on sunspot time series is discussed. Case I: $\eta_1 = 0.3$, $\eta_2 = 0.15$ and $\gamma_1 = 0.001$ are kept unchanged, γ_2 is selected as 0.001, 0.003, 0.005 and 0.007, respectively. From the simulation results, it can be seen that the bigger the γ_2, the worse is the generalization performance. Case II: $\eta_1 = 0.3$, $\eta_2 = 0.15$ and $\gamma_2 = 0.001$ are kept unchanged, γ_1 is selected as 0.001, 0.003, 0.005 and 0.007, respectively. From the simulation results, it can be seen that for bigger γ_1, the generalization performance is the worse. Case III: $\eta_1 = 0.3$, $\gamma_1 = 0.001$ and $\gamma_2 = 0.001$ are kept unchanged, η_2 is selected as 0.15, 0.17, 0.19 and 0.21, respectively. From the simulation results, it can be seen that the bigger the η_2, the worse is the generalization performance. Case IV: $\eta_2 = 0.15$, $\gamma_1 = 0.001$ and $\gamma_2 = 0.001$ are kept unchanged, η_1 is selected as 0.30, 0.32, 0.34 and 0.36, respectively. From the simulation results, it can be seen that for bigger η_1 is, the generalization performance is the worse. All the above results are shown in Table 3.

4.2. *Single-step prediction and iterative-step prediction for chaotic laser pulsation data*

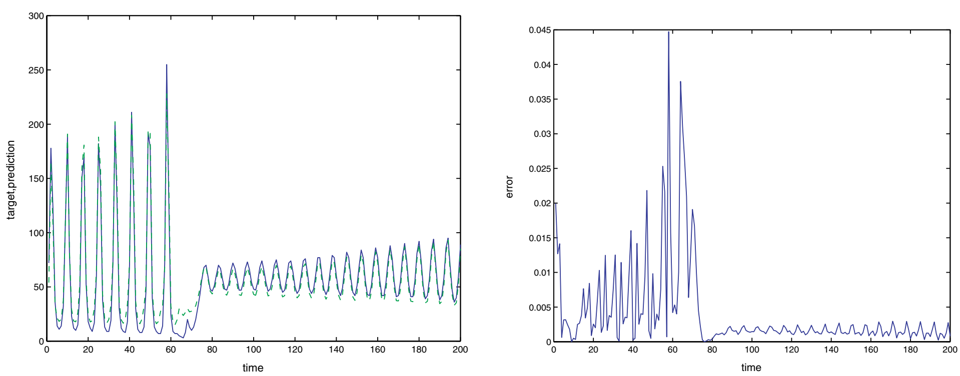
In this subsection, the proposed learning algorithm is also applied to the single-step prediction and iterative-step prediction of chaotic laser pulsation data from the

Table 3. The effects of the parameters with the new modified hybrid learning algorithm for single-step prediction performance on sunspot time series data.

Indices	Mean Squared Errors (1921–2003)			
$\eta_1 = 0.3, \eta_2 = 0.15$	$\gamma_2 = 0.001$	$\gamma_2 = 0.003$	$\gamma_2 = 0.005$	$\gamma_2 = 0.007$
$\gamma_1 = 0.001$	0.0031	0.0041	0.0046	0.0050
$\eta_1 = 0.3, \eta_2 = 0.15$	$\gamma_1 = 0.001$	$\gamma_1 = 0.003$	$\gamma_1 = 0.005$	$\gamma_1 = 0.007$
$\gamma_2 = 0.001$	0.0031	0.0039	0.0042	0.0049
$\eta_1 = 0.3, \gamma_1 = 0.001$	$\eta_2 = 0.15$	$\eta_2 = 0.17$	$\eta_2 = 0.19$	$\eta_2 = 0.21$
$\gamma_2 = 0.001$	0.0031	0.0038	0.0043	0.0045
$\eta_2 = 0.15, \gamma_1 = 0.001$	$\eta_1 = 0.3$	$\eta_1 = 0.32$	$\eta_1 = 0.34$	$\eta_1 = 0.36$
$\gamma_2 = 0.001$	0.0031	0.0036	0.0042	0.0046

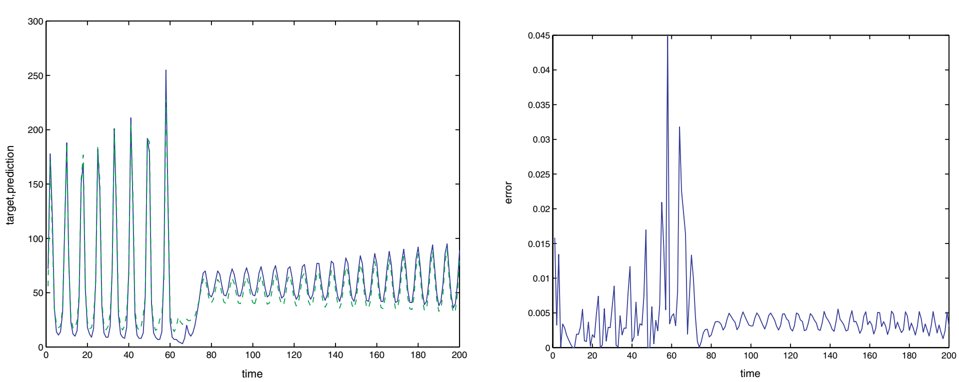
Santa Fe competition data set A. Likely, a (12-8-1)-sized network is also adopted to address this problem. The first 1000 step data are used for the training, and the following 200 steps are used for the testing. As a result, the single-step prediction results are shown in Figs. 9–11 for Hybrid-I algorithm, Hybrid-II algorithm and the proposed new learning algorithm, respectively. The corresponding iterative-step prediction results are shown in Figs. 12–14 for the above three learning algorithms.

Similarly, in order to statistically compare the prediction accuracies for chaotic laser pulsation data with BP algorithm, Hybrid-I algorithm, Hybrid-II algorithm and the proposed new learning algorithm, an experiment is also done fifty times for each algorithm and then its average accuracy value calculated. The corresponding prediction accuracies are summarized in Tables 4 and 5. From these results, it can



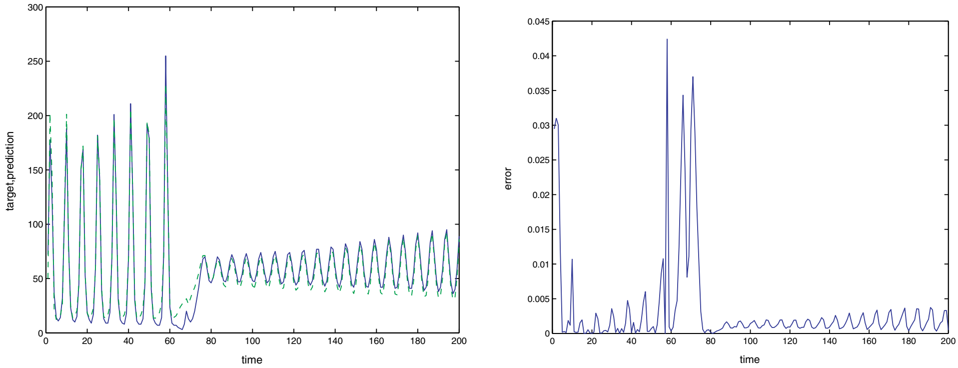
(a) The predicted values (b) The predicted errors

Fig. 9. Results with single-step prediction for chaotic laser pulsation data by using Hybrid-I algorithm.



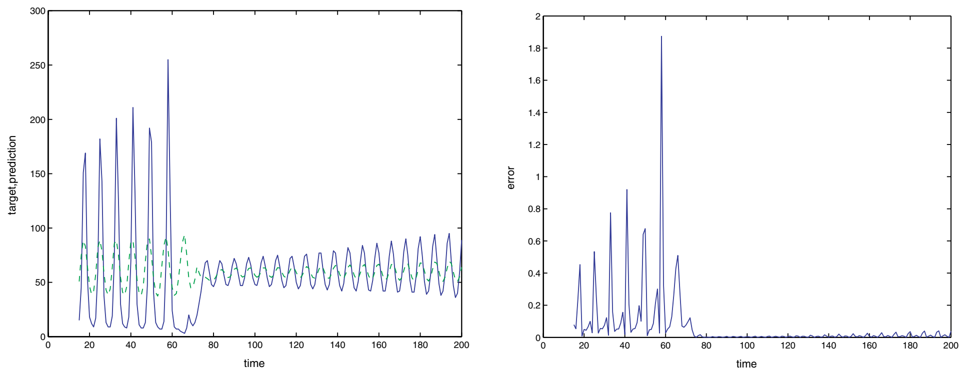
(a) The predicted values (b) The predicted errors

Fig. 10. Results with single-step prediction for chaotic laser pulsation data by using Hybrid-II algorithm.



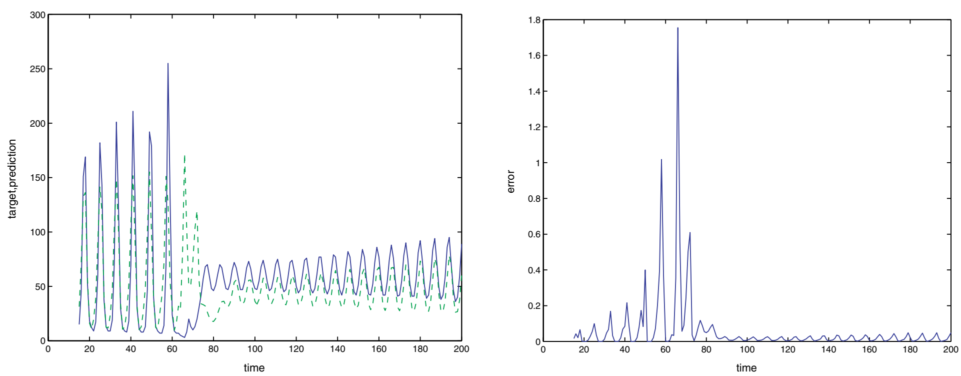
(a) The predicted values (b) The predicted errors

Fig. 11. Results with single-step prediction for chaotic laser pulsation data by using the new learning algorithm.



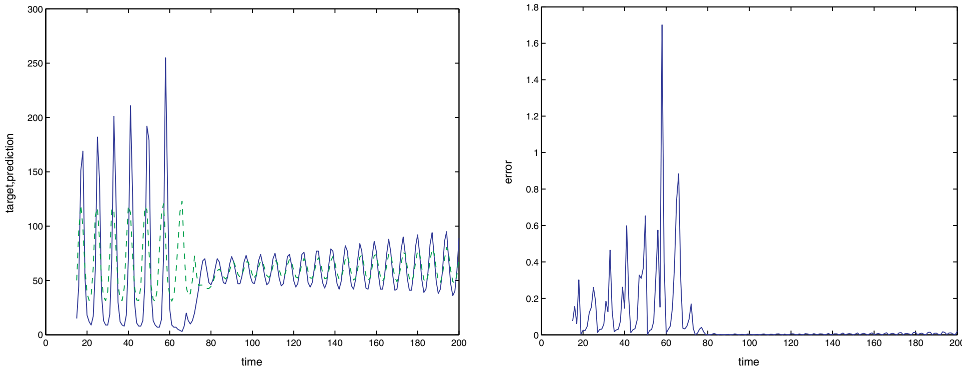
(a) The predicted values (b) The predicted errors

Fig. 12. Results with iterative-step prediction for chaotic laser pulsation data by using Hybrid-I algorithm.



(a) The predicted values (b) The predicted errors

Fig. 13. Results with iterative-step prediction for chaotic laser pulsation data by using Hybrid-II algorithm.



(a) The predicted values (b) The predicted errors

Fig. 14. Results with iterative-step prediction for chaotic laser pulsation data by using the new learning algorithm.

Table 4. The average values of mean squared errors of single-step prediction for the chaotic laser pulsation data for fifty times by four algorithms.

LA	Training	Testing
BP	0.00029725	0.0059
Hybrid-I	0.0016	0.0040
Hybrid-II	0.0012	0.0043
New LA	0.00054626	0.0033

Table 5. The average values of mean squared errors of iterative-step prediction for the chaotic laser pulsation data for fifty times by four algorithms.

LA	Training	Testing
BP	0.0082	0.0821
Hybrid-I	0.0418	0.0680
Hybrid-II	0.0391	0.0615
New LA	0.0310	0.0583

be seen apparently that the proposed learning algorithm has better generalization capability than the BP algorithm as well as the two original hybrid algorithms, since the mean squared errors of the new learning algorithm for the testing data set are smaller than the ones for the other three learning ones.

Obviously, from the above experiments, the conclusion can be drawn that the new learning algorithm has better generalization performance than the original Hybrid-I and Hybrid-II learning algorithms as well as BP learning algorithm. This result mainly rests with the fact that the new learning one incorporates the additional functional constraints such as the input-to-output mapping sensitivity and the weights decay term in training data into the sum-of-square error cost function.

5. Conclusions

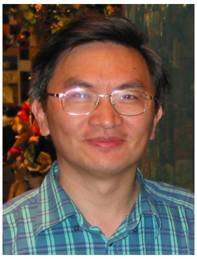
In this paper, a new modified learning algorithm with respect to the Hybrid-I and Hybrid-II learning algorithms introduced in literature[12] is proposed. The additional cost terms for this new algorithm are combined with the ones for Hybrid-I and Hybrid-II learning algorithms, and both the input-to-output mapping sensitivity and high frequency components in training data are penalized in the course of training, thus the better generalization capability with respect to the original hybrid algorithms can be easily obtained. The experimental results about benchmark data of sunspot time series prediction and chaotic laser pulsation data prediction also showed that the generalization performance of the proposed constrained learning algorithm apparently outperforms that of the Hybrid-I and Hybrid-II learning ones as well as BP algorithm. In addition, the effects of the parameters with the proposed learning algorithm on the network performance were discussed. Future research works will include how to apply this new constrained learning algorithm to resolve more numerical computation problems.

References

1. M. Cottrell, B. Girard *et al.*, Neural modeling for time series: a statistical stepwise method for weight elimination, *IEEE Trans. Neural Networks* **6** (1992) 1355–1364.
2. C. Goutte and L. K. Hansen, Regularization with a pruning prior, *Neural Networks* **10** (1997) 1053–1059.
3. D. S. Huang, *Systematic Theory of Neural Networks for Pattern Recognition* (Publishing House of Electronic Industry of China, Beijing, 1996), pp. 111–118.
4. D. S. Huang, Generalization capabilities in feedforward neural networks for pattern recognition, *J. Beijing Inst. Technol.* **5** (1996) 184–192.
5. D. S. Huang, On the comparisons between RLSA and CLA for solving arbitrary linear simultaneous equations, *Lecture Notes in Computer Science* **2690** (Springer-Verlag, 2003), pp. 169–176.
6. D. S. Huang, A constructive approach for finding arbitrary roots of polynomials by neural networks, *IEEE Trans. Neural Networks* **15** (2004) 477–491.
7. D. S. Huang and H. S. Ip Horace, On the choices of the parameters in general constrained learning algorithms, *Lecture Notes in Computer Science* **2690** (Springer-Verlag, 2003), pp. 967–974.
8. D. S. Huang, H. S. Ip Horace and Z. Chi, A neural root finder of polynomials based on root moments, *Neural Comput.* **16** (2004) 1721–1762.
9. D. S. Huang, H. S. Ip Horace, Z. Chi and H.S. Wong, Dilation method for finding close roots of polynomials based on constrained learning neural networks, *Phys. Lett. A* **309** (2003) 443–451.
10. D. S. Huang and Z. Chi, Finding roots of arbitrary high order polynomials based on neural network recursive partitioning method, *Sci. China Ser. F Inform. Sci.* **47** (2004) 232–245.
11. D.-G. Jeong and S.-Y. Lee, Merging back-propagation and Hebbian learning rules for robust classifications, *Neural Networks* **9** (1996) 1213–1222.
12. S. Y. Jeong and S. Y. Lee, Adaptive learning algorithms to incorporate additional functional constraints into neural networks, *Neurocomputing* **35** (2000) 73–90.

13. D. A. Karras, An efficient constrained training algorithm for feedforward networks, *IEEE Trans. Neural Networks* **6** (1995) 1420–1434.
14. S. J. Nowlan and G. E. Hinton, Simplifying neural networks by soft weight sharing, *Neural Comput.* **4** (1992) 473–493.
15. A. S. Weigend, D. E. Rumelhart and B. A. Hyberman, Generalization by weight-elimination applied to currency exchange rate prediction, *Proc. Int. Conf. Neural Network*, Seattle (1991), pp. 837–841.
16. P. M. Williams, Bayesian regularization and pruning using a Laplace prior, *Neural Comput.* **7** (1995) 117–143.

Fei Han received the Associate degree in mathematics education from Anqing Teachers College, China in 1995 and the M.S. degree in computer software from Hefei University of Technology, China in 2003. Currently, he is studying for the Ph.D. degree in pattern recognition and intelligent system at the University of Science and Technology of China, Hefei, China.



Michael R. Lyu received the B.S. degree in electrical engineering from National Taiwan University, the M.S. degree in computer engineering from University of California, Santa Barbara, and the Ph.D. degree in computer science from University of California, Los Angeles in 1981, 1985 and 1988, respectively. He is a IEEE fellow. Currently, he is a Professor at the Computer Science and Engineering Department of the Chinese University of Hong Kong, China.

Xu-Qin Li received his B.S. degree in educational technology from HuaZhong Normal University in 2003. Currently, he is studying for the M.S. degree in pattern recognition and intelligent system in Institute of Intelligent Machines, Chinese Academy of Sciences (IIM/CAS), Hefei, China.

Tat-Ming Lok received the B.S. degree in electronic engineering from the Chinese University of Hong Kong, and the M.S. degree and the Ph.D. degree in electrical engineering from Purdue University. He was a post-doctoral research associate at Purdue University. Currently, he is an Associate Professor at Information Engineering Department of the Chinese University of Hong Kong, China.