Journal of AI and Data Mining
Vol 3, No 2, 2015, 141-147. 10.5829/idosi.JAIDM.2015.03.02.03

Designing stable neural identifier based on Lyapunov method

F. Alibakhshi[1*], M. Teshnehlab[2], M. Alibakhshi[3] and M. Mansouri[4]

1. Control Department, Islamic Azad University South Tehran Branch, Tehran, Iran.
2. Center of Excellence in Industrial Control, K.N. Toosi University, Tehran, Iran.
3. Young Researchers & Elite Club, Borujerd Branch, Islamic Azad University, Borujerd, Iran.
4. Intelligent System Laboratory (ISLAB), Electrical & Computer engineering department, K.N. Toosi University, Tehran, Iran.

Received 29 August 2014; Accepted 21 September 2015
**Corresponding author: Alibakhshi.fatemeh@gmail.com (F. Alibakhshi)*

Abstract
The stability of learning rate in neural network identifiers and controllers is one of the challenging issues, which attract many researchers' interest in neural networks. This paper suggests adaptive gradient descent algorithm with stable learning laws for modified dynamic neural network (MDNN) and studies the stability of this algorithm. Also, stable learning algorithm for parameters of MDNN is proposed. By the proposed method, some constraints are obtained for learning rate. Lyapunov stability theory is applied to study the stability of the proposed algorithm. The Lyapunov stability theory guaranteed the stability of the learning algorithm. In the proposed method, the learning rate can be calculated online and will provide an adaptive learning rate for the MDNN structure. Simulation results are given to validate the results.

Keywords: *Gradient Descent Algorithm, Identifier, Learning Rate, Lyapunov Stability Theory.*

1. Introduction

In recent decades, soft computing is frequently used in business and industry. Artificial neural networks are essential parts of any computing software [4]. The most widely used neural network architecture is the multilayer feed forward neural network. The most popular approach for training the multilayer feed-forward neural network (FNN) is the backpropagation (BP) algorithm based on gradient descent (GD) method. Determining an appropriate learning rate for this algorithm is important. The training algorithms (learning rules) could be defined as "a procedure for modifying the weights and biases of a network in order to train the network to perform some tasks" [6,11]. The network model having a good function approximation capability through the training samples can well reflect the complex nonlinear relationship between objects [17].

However, one problem inherent within them is their convergence to local minima and the user set acceleration rates and inertia factor parameters that are sensitive to the learning process [1–3]. The FNNs with the BP learning algorithm have been used successfully in pattern recognition, optimization, classification, modeling, identification and controlling [13,31]. However, the problems of the slow convergence rate, local minimum and instability are the most challenging issues in this algorithm.

In recent decades, many efforts have been made to improve the convergence of the BP algorithm. There are some works to improve BP algorithm in order to have online training [9,19-21]. For this algorithm determining, an appropriate learning rate is necessary, so that the learning process become stable. If the learning rate is large, learning may happen rapidly, but it may also become unstable. To ensure stable learning, the learning rate is small enough. Small learning rate may also lead to a long training time. These problems are inherent to the basic learning rule of FNN that are based on GD optimization methods [15,30]. The convergence properties of such algorithms are discussed in [5,7,12,15,16,18], and [22]. Learning algorithms based on GD includes real-time recurrent learning (RTRL), ordered derivative learning and so on [1].

Derivative-based methods have the advantage of fast convergence, but they tend to converge to local minima [2]. In addition, due to their

dependence on the analytical derivatives, they are limited to specific objective functions, inferences, and MFs [2].

Some papers [1-3,25,26] have investigated the stability of fuzzy neural networks. The popular method for stability analysis is Lyapunov stability. Also, in [8,10,27,28] Lyapunov stability theorem is considered.

The learning algorithm in neural and fuzzy neural networks not only has the role of updating parameters but also has influence on stability and convergence. The stability and convergence of learning algorithms are rarely investigated in the papers. In this study, the stability of learning algorithm is addressed in dynamic neural networks.

In this paper, the main concern is using Lyapunov stability approach for determining stable learning rate in system identification via modified dynamic neural network. The GD training the parameters of update rule for MDNN is considered.

The rest of article is organized as follows: in section 2, the structure of the dynamic neural network is discussed. In section 3, MDNN learning algorithm applied to process. Simulations and results for three nonlinear systems are presented in section 4. Section 5 presents conclusions.

2. Dynamic neural network

In the feed forward artificial neural, a neuron receives its inputs from other neurons. The weight sum of these signals is the input to the activation function. The resulting value of the activation function is the output of a neuron. This output is branched out to other processing units. This simple model of the artificial neuron ignores many of the characteristics of its biological counterpart. For example, it does not take into account time delays that affect the dynamics of the system [23]. The dynamic neural network (DNN) for the first time proposed by Gupta [24]. The basic structure of dynamic neuron (DN) is shown in figure 1.

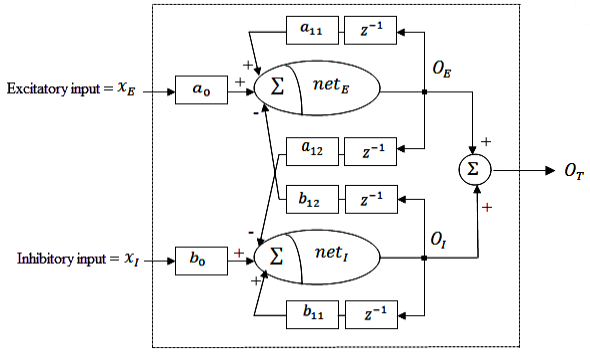

Figure 1. Structure of dynamic neuron [29].

Each neuron composed of two units: the inhibitory (negative) unit and excitatory (positive) unit. The inhibitory units received the summation of positive inputs, a delay of own outputs and abstraction a delay of excitatory outputs by multiple to determined weights. The excitatory units received the summation of negative inputs, a delay of own outputs and abstraction a delay of initiatory outputs by multiple to the determined weights [29].

The final output of the neuron can be written as follow:

$$O_T(t) = O_E(t) + O_I(t) \quad (1)$$

where, O_E, O_I are represent the output of excitatory (net_E) and inhibitory units (net_I), respectively and can be written as:

$$net_E(t) = a_0 X_E(t) + a_{11} net_E(t-1) - b_{12} net_I(t-1) \quad (2)$$

$$net_I(t) = b_0 X_I(t) + b_{11} net_I(t-1) - a_{12} net_E(t-1) \quad (3)$$

And:

$$net_T(t) = net_E(t) + net_I(t) \quad (4)$$

where, X_E, X_I are the positive and negative inputs, respectively and the parameters of a_0, a_{11}, a_{12}, b_0, b_{11}, b_{12} are the weights of DN.

3. Stability analysis of learning algorithm

Suppose an MDNN as an identifier shown in figure 2.

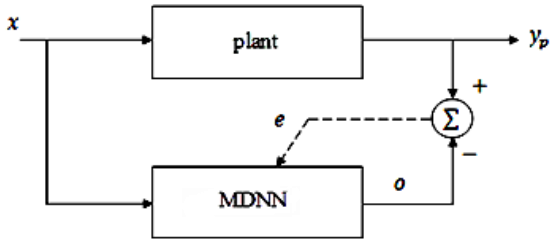

Figure 2. Modified dynamic neural network as the identifier.

Details of the MDNN network are illustrated in the figure 3.

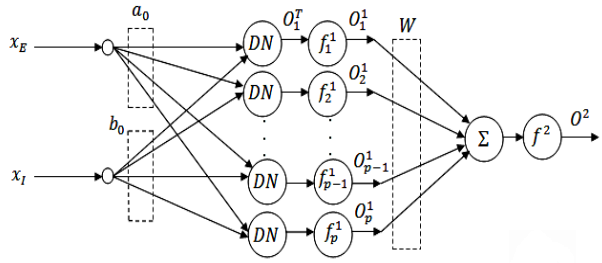

Figure 3. Modified dynamic neural network architecture.

Assume that the parameters of the MDNN are changed with time. In figure 3, $W(k)=\left[w_1(k),w_2(k),\ldots,w_p(k)\right]^T$ the is the weight vector of the MDNN output layer, $O^2(k)$ is the final output of MDNN and $O^1(k)$ is the output of hidden layer of MDNN, $O^T(k)$ is the output of DN neuron. DN structure of neurons is shown in figure 1. This network has *n* inputs in the input layer, *p* DN in the hidden layer and one conventional neuron in the output layer. $f^1(.)$ and $f^2(.)$ are nonlinear activation functions. According to figure 3 it is obvious that:

$$O^1(k)=f^1\left(net_T(k)\right),\ O^2(k)=f^2\left(WO^1(k)\right) \quad (5)$$

$$O(k)=O^2(k)=f^2\left(Wf^1\left(net_T(k)\right)\right) \quad (6)$$

The cost function for the training algorithm is defined as:

$$e(k)=y_p(k)-O(k),\ E(k)=\frac{1}{2}e^2(k) \quad (7)$$

where, $e(k)$ is real output error, $y_p(k)$ is the output of plant, $O(k)$ is the final output of MDNN. Weights of output layer are updated by GD method as follows:

$$W(k+1)=W(k)+\eta^o(k)\left(-\frac{\partial E(k)}{\partial W(k)}\right) \quad (8)$$

where, $\eta^o(k)$ is the learning rate parameters of the output layer, and we have:

$$\frac{\partial E(k)}{\partial W(k)}=\left(\frac{\partial E(k)}{\partial e(k)}\right)\times\left(\frac{\partial e(k)}{\partial O(k)}\right)\times\left(\frac{\partial O(k)}{\partial W(k)}\right) \quad (9)$$
$$=-e(k)\left(\frac{\partial O(k)}{\partial W(k)}\right)$$

From (8) and (9), it will be inferred that:

$$W(k+1)=W(k)+\eta^o(k)e(k)\left(\frac{\partial O(k)}{\partial W(k)}\right) \quad (10)$$

$$\Delta W(k)=\eta^o(k)e(k)\left(\frac{\partial O(k)}{\partial W(k)}\right)$$

$$D_W(k)=\frac{\partial O(k)}{\partial W(k)} \quad (11)$$

And the updating rule for hidden layer parameters by GD method is as follows:

$$\Gamma(k+1)=\Gamma(k)+\eta^h(k)\left(-\frac{\partial E(k)}{\partial \Gamma(k)}\right) \quad (12)$$

where, $\Gamma=\left[a_0\ a_{11}\ a_{12}\ b_0\ b_{11}\ b_{12}\right]$ is the weight vector and, $\eta^h(k)$ is the learning rate of DN. Assuming $\Delta\Gamma(k)=\Gamma(k+1)-\Gamma(k)$ the weights are updated as follows:

$$a_0(k+1)=a_0(k)+\eta^h(k)\left(-\frac{\partial E(k)}{\partial a_0(k)}\right) \quad (13)$$

$$\frac{\partial E(k)}{\partial a_0(k)}=\left(\frac{\partial E(k)}{\partial e(k)}\right)\times\left(\frac{\partial e(k)}{\partial O(k)}\right)\times\left(\frac{\partial O(k)}{\partial a_0(k)}\right) \quad (14)$$
$$=-e(k)\left(\frac{\partial O(k)}{\partial a_0(k)}\right)$$

From (13) and (14), it will be inferred that:

$$\Delta a_0(k)=\eta^h(k)e(k)\left(\frac{\partial O(k)}{\partial a_0(k)}\right) \quad (15)$$

It can also be written as:

$$\Delta a_{11}(k)=\eta^h(k)e(k)\left(\frac{\partial O(k)}{\partial a_{11}(k)}\right) \quad (16)$$

$$\Delta a_{12}(k)=\eta^h(k)e(k)\left(\frac{\partial O(k)}{\partial a_{12}(k)}\right) \quad (17)$$

$$\Delta b_0(k)=\eta^h(k)e(k)\left(\frac{\partial O(k)}{\partial b_0(k)}\right) \quad (18)$$

$$\Delta b_{11}(k)=\eta^h(k)e(k)\left(\frac{\partial O(k)}{\partial b_{11}(k)}\right) \quad (19)$$

$$\Delta b_{12}(k)=\eta^h(k)e(k)\left(\frac{\partial O(k)}{\partial b_{12}(k)}\right) \quad (20)$$

$$D_\Gamma(k)=\frac{\partial O(k)}{\partial \Gamma(k)}\quad \Gamma=\left[a_0\ a_{11}\ a_{12}\ b_0\ b_{11}\ b_{12}\right] \quad (21)$$

In this paper, candidate Lyapunov function is a function of error which is associated with GD based learning algorithms. This means when Lyapunov function converges to zero GD based learning algorithm is converged to zero too.
Now a discrete Lyapunov function is considered as follows:

$$V(k)=\frac{1}{2}e^2(k) \quad (22)$$

Then, the variation of Lyapunov function on each iteration will be:

$$\Delta V(k)=V(k+1)-V(k) \quad (23)$$
$$=\frac{1}{2}\left(e^2(k+1)-e^2(k)\right)$$
$$=\frac{1}{2}\left(e(k+1)-e(k)\right)\left(e(k+1)+e(k)\right)$$
$$=\frac{1}{2}\Delta e(k)\left(2e(k)+\Delta e(k)\right)$$

The variation of error can be approximated by:

$$\Delta e(k)=\left\{\left(\frac{\partial e(k)}{\partial W(k)}\right)^T \Delta W(k)+\overbrace{tr\left[\left(\frac{\partial e(k)}{\partial a_0(k)}\right)^T \Delta a_0(k)\right]}^{\alpha_1}+\overbrace{tr\left[\left(\frac{\partial e(k)}{\partial a_{11}(k)}\right)^T \Delta a_{11}(k)\right]}^{\alpha_2}+\overbrace{tr\left[\left(\frac{\partial e(k)}{\partial a_{12}(k)}\right)^T \Delta a_{12}(k)\right]}^{\alpha_3}\right.$$
$$\left.+\overbrace{tr\left[\left(\frac{\partial e(k)}{\partial b_0(k)}\right)^T \Delta b_0(k)\right]}^{\alpha_4}+\overbrace{tr\left[\left(\frac{\partial e(k)}{\partial b_{11}(k)}\right)^T \Delta b_{11}(k)\right]}^{\alpha_5}+\overbrace{tr\left[\left(\frac{\partial e(k)}{\partial b_{12}(k)}\right)^T \Delta b_{12}(k)\right]}^{\alpha_6}\right\} \quad (24)$$

where, the $tr(.)$ is the trace of matrices. From (10) it will be concluded that:

$$\left(\frac{\partial e(k)}{\partial W(k)}\right)^T \Delta W(k)=\left(\frac{\partial e(k)}{\partial O(k)}\right)^T \times\left(\frac{\partial O(k)}{\partial W(k)}\right)^T \times\eta^o(k)e(k)\left(\frac{\partial O(k)}{\partial W(k)}\right)$$
$$=-\eta^o(k)e(k)\left(\frac{\partial O(k)}{\partial W(k)}\right)^T \times\left(\frac{\partial O(k)}{\partial W(k)}\right)$$
$$=-\eta^o(k)e(k)\left\|D_W(k)\right\|^2 \quad (25)$$

From (15) and (21), α_1 obtained as follows:

$$\alpha_1=tr\left[\left(\frac{\partial e(k)}{\partial a_0(k)}\right)^T \Delta a_0(k)\right]$$
$$=tr\left[\left(\frac{\partial e(k)}{\partial O(k)}\right)^T \times\left(\frac{\partial O(k)}{\partial a_0(k)}\right)^T \times\ \eta^h(k)e(k)\left(\frac{\partial O(k)}{\partial a_0(k)}\right)\right]$$
$$=tr\left[-\eta^h(k)e(k)\left(\frac{\partial O(k)}{\partial a_0(k)}\right)^T \times\left(\frac{\partial O(k)}{\partial a_0(k)}\right)\right]$$
$$=-\eta^h(k)e(k)tr\left[\left(\frac{\partial O(k)}{\partial a_0(k)}\right)^T \times\left(\frac{\partial O(k)}{\partial a_0(k)}\right)\right]$$
$$=-\eta^h(k)e(k)\left\|D_{a_0}(k)\right\|_F^2 \quad (26)$$

Let $\|.\|_F$ be Frobenius norm. Thus, using (16) to (20) and (21), the following equations are obtained:

$$\alpha_2=-\eta^h(k)e(k)\left\|D_{a_{11}}(k)\right\|_F^2 \quad (27)$$

$$\alpha_3=-\eta^h(k)e(k)\left\|D_{a_{12}}(k)\right\|_F^2 \quad (28)$$

$$\alpha_4=-\eta^h(k)e(k)\left\|D_{b_0}(k)\right\|_F^2 \quad (29)$$

$$\alpha_5=-\eta^h(k)e(k)\left\|D_{b_{11}}(k)\right\|_F^2 \quad (30)$$

$$\alpha_6=-\eta^h(k)e(k)\left\|D_{b_{12}}(k)\right\|_F^2 \quad (31)$$

So, it can be written as:

$$\Delta e(k)=-\eta^o(k)e(k)\left\|D_W(k)\right\|^2-\eta^h(k)e(k)\left\{\left\|D_{a_0}(k)\right\|_F^2+\left\|D_{a_{11}}(k)\right\|_F^2+\left\|D_{a_{12}}(k)\right\|_F^2+\left\|D_{b_0}(k)\right\|_F^2+\left\|D_{b_{11}}(k)\right\|_F^2+\left\|D_{b_{12}}(k)\right\|_F^2\right\} \quad (32)$$

From (23) and (32), it will be inferred that:

$$\Delta V(k)=\frac{1}{2}\Delta e(k)\left(2e(k)+\Delta e(k)\right)$$
$$=\frac{1}{2}\left(-\eta^o(k)e(k)\left\|D_W(k)\right\|^2-\eta^h(k)e(k)\left\{\left\|D_{a_0}(k)\right\|_F^2+\left\|D_{a_{11}}(k)\right\|_F^2+\left\|D_{a_{12}}(k)\right\|_F^2+\left\|D_{b_0}(k)\right\|_F^2+\left\|D_{b_{11}}(k)\right\|_F^2+\left\|D_{b_{12}}(k)\right\|_F^2\right\}\right)$$
$$\times\left(2e(k)-\eta^o(k)e(k)\left\|D_W(k)\right\|^2-\eta^h(k)e(k)\left\{\left\|D_{a_0}(k)\right\|_F^2+\left\|D_{a_{11}}(k)\right\|_F^2+\left\|D_{a_{12}}(k)\right\|_F^2+\left\|D_{b_0}(k)\right\|_F^2+\left\|D_{b_{11}}(k)\right\|_F^2+\left\|D_{b_{12}}(k)\right\|_F^2\right\}\right)$$
$$=-\frac{1}{2}e^2(k)\left(\eta^o(k)\left\|D_W(k)\right\|^2+\eta^h(k)\left\{\left\|D_{a_0}(k)\right\|_F^2+\left\|D_{a_{11}}(k)\right\|_F^2+\left\|D_{a_{12}}(k)\right\|_F^2+\left\|D_{b_0}(k)\right\|_F^2+\left\|D_{b_{11}}(k)\right\|_F^2+\left\|D_{b_{12}}(k)\right\|_F^2\right\}\right)$$
$$\times\left(2-\eta^o(k)\left\|D_W(k)\right\|^2-\eta^h(k)\left\{\left\|D_{a_0}(k)\right\|_F^2+\left\|D_{a_{11}}(k)\right\|_F^2+\left\|D_{a_{12}}(k)\right\|_F^2+\left\|D_{b_0}(k)\right\|_F^2+\left\|D_{b_{11}}(k)\right\|_F^2+\left\|D_{b_{12}}(k)\right\|_F^2\right\}\right)<0 \quad (33)$$

Then:

$$\Delta V(k)<0\Rightarrow 0<\eta^o(k)\left\|D_W(k)\right\|^2+\eta^h(k)\left\{\left\|D_{a_0}(k)\right\|_F^2+\left\|D_{a_{11}}(k)\right\|_F^2+\left\|D_{a_{12}}(k)\right\|_F^2+\left\|D_{b_0}(k)\right\|_F^2+\left\|D_{b_{11}}(k)\right\|_F^2+\left\|D_{b_{12}}(k)\right\|_F^2\right\}<2 \quad (34)$$

If we choose $\eta^o(k)=\eta^h(k)$ then:

$$0<\eta^o(k)=\eta^h(k)<\frac{2}{\|D_W(k)\|^2+\|D_{a_0}(k)\|_F^2+\|D_{a_{11}}(k)\|_F^2+\|D_{a_{12}}(k)\|_F^2+\|D_{b_0}(k)\|_F^2+\|D_{b_{11}}(k)\|_F^2+\|D_{b_{12}}(k)\|_F^2} \quad (35)$$

From (35) we choose the learning rates as follows:

$$0<\eta^o(k)<\frac{2}{\|D_W(k)\|^2} \quad (36)$$

$$0<\eta^h(k)<\frac{2}{6\left(\|D_W(k)\|^2+\|D_{a_0}(k)\|_F^2+\|D_{a_{11}}(k)\|_F^2+\|D_{a_{12}}(k)\|_F^2+\|D_{b_0}(k)\|_F^2+\|D_{b_{11}}(k)\|_F^2+\|D_{b_{12}}(k)\|_F^2\right)} \quad (37)$$

4. Simulation and results

In this section, the proposed algorithm in sections 3 is simulated on three nonlinear systems as examples 1, 2 and 3.

In each example, there are 1000 random numbers which divide to training and test data sets. Dynamic neural network is used as identifier as illustrated in figure 2.

Example 1: Identification of a nonlinear dynamical system. In this example, the nonlinear plant with multiple time-delays is described as [3]:

$$y(k+1)=f\left(y(k),y(k-1),y(k-2),u(k),u(k-1)\right) \quad (38)$$

where, $u(k)$ and $y(k)$ are the system input and output, respectively.

Where:

$$f(x_1,x_2,x_3,x_4,x_5)=\frac{x_1x_2x_3x_5(x_3-1)+x_4}{1+x_2^2+x_3^3} \quad (39)$$

For simulation of this nonlinear system, a neural network with the structure depicted in figure 2 is employed, where n is assumed to be 5, and p is taken as 15. In this neural network, it has been assumed that f^2 is a linear function, and f^1 is a symmetric sigmoid function defined as below:

$$f^1(net)=\frac{1-e^{-net}}{1+e^{-net}} \quad (40)$$

where, net is the weighted sum of the inputs. For comparison, the mean square error (MSE) criterion has been used.

In the simulation results, figure 4 indicates convergence with fulfillment of the stability conditions.

Example 2: This system of equation is as follows [14]:

$$y(k+1)=\frac{y(k)}{1+y^2(k)}+u^2(k) \quad (41)$$

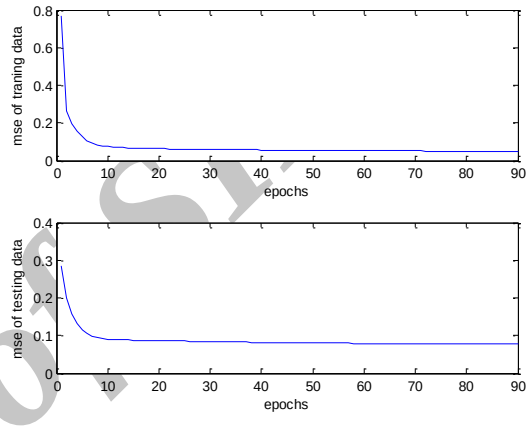

Figure 4. Learning rate is smaller than the upper limit bound.

where, $u(k)$ and $y(k)$ are the system input and output, respectively.

For simulation of this nonlinear system, a neural network with the structure depicted in figure 2 is employed, where n is assumed to be 2, and p is taken as 10.

In the simulation results, figure 5 indicates convergence with fulfillment of the stability conditions.

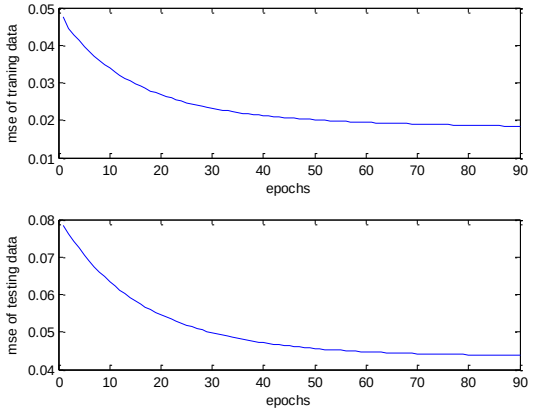

Figure 5. Learning rate is smaller than the upper limit bound.

Example 3: This system of equation is as follows [1-3]:

$$y(k+1)=0.3y(k)+0.6y(k-1)+f(u(k)) \qquad (42)$$

where, $u(k)$ and $y(k)$ are the system input and output, respectively. The unknown function $f(.)$ is described as follows:

$$f(u)=0.6\sin(\pi u)+0.3\sin(3\pi u)+0.1\sin(3\pi u) \qquad (43)$$

For simulation of this nonlinear system, a neural network with the structure depicted in figure 2 is employed, where n is assumed to be 3, and p is taken as 15. In the simulation results, figure 6 indicates convergence with fulfillment of the stability conditions.

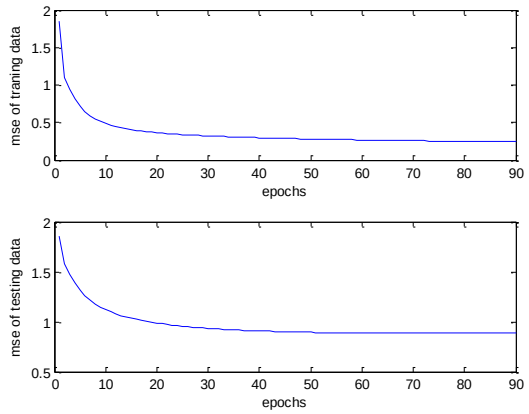

Figure 6. Learning rate is smaller than the upper limit bound.

5. Conclusions

In this paper, for permanent learning of modified dynamic neural network parameters online, Lyapunov stability theory is employed. In this learning algorithm, the associated parameters are trained according to descending gradient. Taking advantage of Lyapunov stability theory, some regions are defined, and through selection of the permanent training rate out of these regions, it can be guaranteed that the learning algorithm is stable throughout the identification process. The results of the obtained theory have been simulated for three examples. The simulation results suggest that in the training process, the system is stable and the convergence rate is desirable. This procedure can be employed in neural controllers.

References

[1] AliyariShoorehdeli, M., Teshnehlab, M., KhakiSedigh, A. & AhmadiehKhanesar M. (2009). Identification using ANFIS with intelligent hybrid stable learning algorithm approaches and stability analysis of training methods. Elsevier, Applied Soft Computing 9, pp. 833–850.

[2] AliyariShoorehdeli, M., Teshnehlab, M. & KhakiSedigh, A. (2009). Training ANFIS as an identifier with intelligent hybrid stable learning algorithm based on particle swarm optimization and extended Kalman filter. Elsevier, Fuzzy SetsandSystems160, pp. 922 –948.

[3] AliyariShoorehdeli, M., Teshnehlab, M. & KhakiSedigh, A. (2009). Identification using ANFIS with intelligent hybrid stable learning algorithm approaches. Springer, Neural Comput & Applic 18, pp. 157–174.

[4] Bonissone, P. P., Goebl, K. Y. T. & khedkar, P. S. (1999). Parameter convergence and learning curves for neural networks. Proc, of IEEE, vol. 87, no. 9, pp. 1641-1667.

[5] Fine, T. L., Goebl, S. & khedkar, P. S. (1997). Hybrid Soft Computing System: Industrial and Commercial Application. Neural Computation, pp. 747-769.

[6] Hagan, M. T., Demuth, H. B. & Beale, M. (1996). Neural network design, McGraw – Hill publishing company. First edition.

[7] Hagan, M. T. & Menhaj, M. (1994). Training feedforward networks with Marquardt algorithm. IEEE Transactions on Neural Networks, vol. 5, no. 6, pp. 989-993.

[8] Zhang, T., Ge, S. S. & Hang, C. C. (May 2000). Adaptive neural network control for strict-feedback nonlinear systems using back stepping design. Elsevier Science Ltd., pp. 1835-1846.

[9] Hedjar, R. (2007). Online Adaptive Control of Non-linear Plants Using Neural Networks with Application to Temperature Control System. J. King Saud Univ., vol. 19, Comp. & Info. Sci, pp. 75-94.

[10] Ge, S. S., Hang, C. C. & Zhang, T. (June 2000). Stable Adaptive Control for Nonlinear Multivariable Systems with a Triangular Control Structure. IEEE Transactions on Automatic Control, vol. 45, no. 6.

[11] Kazem, B. I. & Zangana, N. F. H. (2007). A Neural Network Based Real Time Controller for Turning Process. Jordan Journal of Mechanical and Industrial Engineering, ISSN 1995-6665, pp 43 – 55.

[12] Kuan, C. M. & Hornik, K. (1991). Dynamical systems using neural networks. IEEE Transactions on Neural Networks, vol. 1, no. 1, pp. 4-27.

[13] Meireles, M. R. G., Almeida, P. E. M. & Simões, M .G. (2003). A Comrehensive Review For Industrial Applicability Of Artifical Neural Networks. IEEE Trans. Ind. Electron, vol. 50, no. 3, pp. 585 – 601.

[14] Mandic, D. P., Hanna, A. I. & Razaz, Moe. (2001). A Normalized GraDient Descent Algorithm For Nonlinear Adaptive Filters Using A Gradient Adaptive Step Size. IEEE Signal Processing, Letters . vol. 1, no. 1.

[15] Sha, D. & Bajic, V. B. (2011). An Optimized Recursive Learning Algorithm for Three-Layer Feed forward Neural Networks For MIMO Nonlinear System Identifications. Intelligent Automation and Soft Computing, vol. 17, no. x, pp. 1-15.

[16] Song, Q. & Xiao, J. (1997). On the Convergence Performance of Multilayered NN Tracking Controller. Neural & Parallel Computation, vol. 5, no. 3, pp. 461-470.

[17] Sun, X., Liu, Q. & Zhang, L. (2011). A BP Neural Network Model Based on Genetic Algorithm for Comprehensive Evaluation. IEEE, 978-1-4577-0856.

[18] Torii, M. & Hagan, M. T. (2002). Stability of steepest descent with momentum for quadratic functions. IEEE Trans. On Neural Networks, vol. 13, no. 3, pp. 752-756.

[19] Velagic, J., Osmic N. & Lacevic, B. (2008). Neural Network Controller for Mobile Robot Motion Control, World Academy of Science. Engineering and Technology 47.

[20] Velagic, J. & Hebibovic, M. (2004). Neuro-Fuzzy Architecture for Identification and Tracking Control of a Robot. In Proc, The World Automation Congress - 5th International Symposium on Soft Computing for Industry ISSCI2004, June 28 - July 1, Sevilla Spain, paper no. ISSCI-032, pp. 1-9.

[21] Velagic, J., Lacevic, B. & Hebibovic, M. (2005). On-Line Identification of a Robot Manipulator Using Neural Network with an Adaptive Learning Rate. in Proc. 16th IFAC World Congress, 03-08 June, Prague, Czech Republic, no. 2684, pp. 1-6.

[22] Wu, W., Feng, G. R., Li, Z. X. & Xu, Y. S. (2005). Deterministic Convergence of an Online Gradient Method for BP Neural Networks. IEEE Transaction on Neural Networks, vol. 16, no. 3, pp. 533-540.

[23] Wasserman, P. D. (1989). Neural Computing: Theory and Practice, Van Nostrand, New York.

[24] Gupta, M. M. & Rae, D. H. (1993). Dynamic Neural Units with Applications to the Control of Unknown Nonlinear Systems. 7th Journal of intelligent and Fuzzy Systems, vol. 1, no. 1, pp. 73-92, Jan.

[25] Kim, W. C., Ahn, S. C. & Kwon, W. H. (1995). Stability Analysis and Stabilization of Fuzzy State Space Models. Fuzzy Sets Syst. 71 (April (1)), pp.131–142.

[26] Yu, W. & Li, X. (June 2003), Fuzzy neural modeling using stable learning algorithm. In: Proceedings of the American Control Conference, pp. 4542–4547.

[27] Jafarov, E. M. (August 2013). On Stability Delay Bounds of Simple Input-delayed Linear and Non-Linear systems: Computational Results. International Journal of Automation and Computing (IJAC), vol. 10, no. 4, pp. 327–334.

[28] Sun, H. Y., Li, N., Zhao, De. P. & Zhang, Q. L. (August 2013). Synchronization of Complex Networks with Coupling Delays via Adaptive Pinning Intermittent Control. International Journal of Automation and Computing (IJAC), vol. 10, no. 4, pp. 312–318.

[29] Sabahi, K., Nekoui, M. A., Teshnehlab, M., Aliyari M. & Mansouri, M. (July 2007). Load Frequency Control in Interconnected Power System Using Modified Dynamic Neural Networks. Proceedings of the 15th Mediterranean Conference on Control & Automation, Athens - Greece.

[30] Widrow, B. & Lehr, M. A. (1990). Adaptive Neural Networks: Perceptron, Madaline, and Back propagation. Proceedings of the IEEE, Special Issue on Neural Networks, I: Theory & Modeling; 78(9), pp.1415-1442.

[31] Heydari, A. & Balakrishnan, S. N. (2014). Optimal Switching and Control of Nonlinear Switching Systems Using Approximate Dynamic Programming. IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 6, pp. 1106-1117.

نشریه هوش مصنوعی و داده کاوی

دانشگاه شاهرود

طراحی شناساگر عصبی پایدار بر اساس روش لیاپانوف

فاطمه علی بخشی[1*]، محمد تشنه لب[2]، مهدی علی بخشی[3] و محمد منثوری[4]

[1] دانشکده فنی و مهندسی، دانشگاه آزاد اسلامی واحد تهران جنوب ، تهران، ایران.

[2] قطب علمی کنترل صنعتی، دانشگاه صنعتی خواجه نصیرالدین طوسی، تهران، ایران.

[3] دانشگاه آزاد اسلامی واحد بروجرد، باشگاه پژوهشگران جوان و نخبگان، بروجرد، ایران.

[4] آزمایشگاه سیستم‌های هوشمند، دانشکده برق و کامپیوتر، دانشگاه صنعتی خواجه نصیرالدین طوسی، تهران، ایران.

ارسال ۲۰۱۴/۰۸/۲۹؛ پذیرش ۲۰۱۵/۰۹/۲۱

چکیده:

بحث پایداری در شبکه‌های عصبی شناساگر و کنترل‌کننده یکی از موارد پراهمیت در مهندسی کنترل می‌باشد، که در این راستا تحقیقات متنوعی صورت پذیرفته است که عمدتاً در چهار چوب شبکه‌های عصبی معمولی بوده است. در این مقاله الگوریتم گرادیان نزولی تطبیقی با قوانین یادگیری پایدار برای پارامترهای شبکه عصبی پویا پیشنهاد شده و پایداری الگوریتم آموزشی بررسی شده است. توسط روش پیشنهادی محدوده‌هایی برای نرخ آموزش تعریف می‌شوند. قضیه پایداری لیاپانوف برای بررسی پایداری الگوریتم پیشنهادی بکار برده می‌شود. قضیه پایداری لیاپانوف، پایداری الگوریتم آموزش را تضمین می‌کند. در روش پیشنهادی، نرخ آموزش بصورت بهنگام محاسبه و نرخ آموزش تطبیقی برای شبکه عصبی پویا بکار برده می‌شود. نتایج شبیه‌سازی صحت نتایج را نشان می‌دهد.

کلمات کلیدی: الگوریتم گرادیان نزولی، شناساگر، قضیه پایداری لیاپانوف، نرخ آموزش.