Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Review article

Neural network control of nonlinear dynamic systems using hybrid algorithm

Mounir Ben Nasr*, Mohamed Chtourou

Control and Energy Management Laboratory (CEMLab), Department of Electrical Engineering, ENIS, BP 1173 Sfax, Tunisia

ARTICLE INFO

Article history:
Received 12 February 2014
Received in revised form 27 May 2014
Accepted 28 July 2014
Available online 10 August 2014

Keywords:
Neural network control
Nonlinear systems
Gradient descent method
Supervised and unsupervised learning
Self-organizing map
Hybrid learning

ABSTRACT

In this paper, a hybrid method is proposed to control a nonlinear dynamic system using feedforward neural network. This learning procedure uses different learning algorithm separately. The weights connecting the input and hidden layers are firstly adjusted by a self organized learning procedure, whereas the weights between hidden and output layers are trained by supervised learning algorithm, such as a gradient descent method. A comparison with backpropagation (BP) shows that the new algorithm can considerably reduce network training time.

Contents

Introduction

The design of the motion control of nonlinear systems has attracted an ever increasing interest. Various advanced methods have been proposed to obtain satisfactory tracking performances. These include robust adaptive control [1,2], sliding mode control [3,4], backstepping control [5,6], and decentralized control [7,8]. However, it is now well know that nonlinear dynamical systems can exhibit extremely complex dynamic behaviour and for this reason it becomes necessary to use intelligent control techniques. More recently, techniques like neural networks, fuzzy logic and genetic algorithm have been applied with some success to problems of control and identification of nonlinear systems for several domains of application [9–14]. Moreover, in some works, both Neural Networks and advanced methods are used to get the better

* Corresponding author. Tel.: +216 4 274 088; fax: +216 4 275 595.
E-mail addresses: Nasr_mounir@yahoo.fr (M. Ben Nasr), Mohamed.Chtourou@enis.rnu.tn (M. Chtourou).

http://dx.doi.org/10.1016/j.asoc.2014.07.023
1568-4946/© 2014 Elsevier B.V. All rights reserved.

performances [15–22]. For example [18], propose adaptive Neural Network decentralized backstepping output-feedback control for nonlinear large scale systems with time delays. In [21], robust adaptive Neural Networks control was developed for a class of uncertain MIMO nonlinear systems with input nonlinearities. A neural network, one of the most popular intelligent computation approaches, has the capability to approximate any desired nonlinear function to an arbitrary degree of accuracy [23]. The principal type of neural networks based control scheme is the feedforward neural networks (FNN) [24–28]. The most commonly used method to train FNN is based on backpropagation (BP) algorithm [29]. Although the gradient descent based learning method has achieved many practical successes in the neural control of nonlinear systems, this method usually encounters problem of slow convergence. A number of learning algorithms have been proposed in the literature in an attempt to speed up this method [30–38]. Training speed is significantly increased by using various second orders algorithm [39–41]. Many other algorithms with the emphasis on hybrid techniques have been developed to accelerate the training method of feedforward neural network [42–49].

Our proposed work is motivated by the very recently introduced hybrid algorithm called a FN-based (Fuzzy Neighbourhood) hybrid which combines unsupervised and supervised learning [45]. In this approach, the weights between input and hidden layers were determined according to an unsupervised procedure relying on the Kohonen algorithm with fuzzy neighbourhood function and the weights between hidden and output layers were updated according to a supervised procedure based on gradient descent method. By combining the above two modes, the learning algorithm converge much faster than other well-known algorithms.

The main issues in the field of neural networks based control include the architecture of the identifier and controller, and the algorithm to be used in training their parameters [50,51]. The proposed control architecture adopted in this paper is the basis for many current neurocontroller. The neurocontroller contains two neural networks. One is the neural model (NM) and the other is the neural controller (NC). In this approach, a neural model (NM) is first trained to model the forward dynamic of the plant with a recently proposed approach called a FN (Fuzzy neighbourhood)-based hybrid [45]. Then, an untrained controller network is placed in series with the network model of the plant. Finally, the weights for the neural controller (NC) are trained using a proposed hybrid algorithm, while holding the neural model weights constant. In this paper, we propose a learning method for the neural controller (NC). This proposed hybrid approach uses a Kohonen algorithm with fuzzy neighbourhood for clustering the weights of the hidden layer and gradient descent method for training the weights of the output layer.

The paper is organized as follows: in "Problem statement", we formulate the control problem and explain the control architecture method using two neural networks. In "Learning algorithm for the neural controller", the control of a nonlinear plant using hybrid algorithm is illustrated. In "Simulation results", simulation results and comparisons between two learning algorithms are given. Finally, in "Conclusion", we present the main conclusions.

Problem statement

Assume that the unknown nonlinear system to be considered is expressed by:

$$y(k)=f[y(k-1), y(k-2), \ldots, y(k-n), u(k), u(k-1), \ldots, u(k-m)] \tag{1}$$

where $y(k)$ is the scalar output of the system, $u(k)$ is the scalar input to the system, n and m are the output and the input orders respectively, $f[\ldots]$ is the unknown nonlinear function to be estimated by a Neural Network. The control problem is to select a control input $u(k)$, such that the output of the system $y(k)$ tracks the given reference y^d as close as possible.

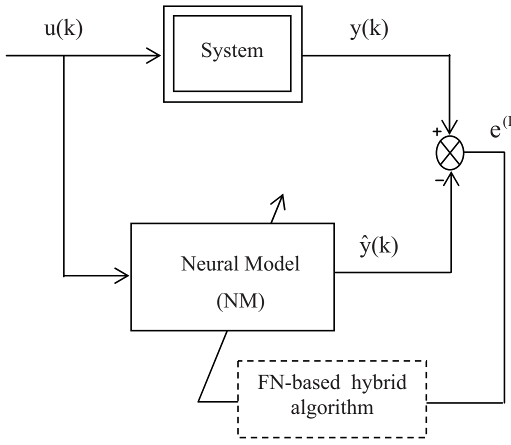

Fig. 1. Block diagram of system identification based on the hybrid algorithm.

A feedforward neural network is used to learn the system (1) and FN (Fuzzy Neighbourhood)-based hybrid algorithm [45] is employed to train the weights. The block diagram of identification system is shown in Fig. 1. Since the input to neural network is

$$P = [y(k-1), y(k-2), \ldots, y(k-n), u(k), u(k-n), \ldots, u(k-m)] \tag{2}$$

The neural model for the unknown system (1) can be expressed as:

$$\hat{y}(k)=\hat{f}[y(k-1), y(k-2), \ldots, y(k-n), u(k), u(k-1), \ldots, u(k-m)] \tag{3}$$

where $\hat{y}(k)$ is the output of the neural model and $\hat{f}$ is the estimate of f.

The weights of the neural model are adjusted to minimize the cost function given by:

$$E^{(I)} = \frac{1}{2}(e^{(I)})^2 \tag{4}$$

where $e^{(I)} = y(k) - \hat{y}(k)$ is the identification error, the difference of the outputs between the plant and the neural model, $y(k)$ is the plant output and $\hat{y}(k)$ is the neural model output.

To solve control problem the specialized control structure [27] is considered. This strategy consists of the system and two feedforward neural networks (FNN), the NM (neural model) for identification and the NC (neural controller) for control as shown in Fig. 2.

The cost function to be minimized for the neural controller is given by

$$E^{(C)} = \frac{1}{2}(e^{(C)})^2 \tag{5}$$

where $e^{(C)} = \hat{y}(k) - y^d$ is the control error, between the output of NM $\hat{y}(k)$ and the desired value y^d.

Learning algorithm for the neural controller

In this section, we introduce the hybrid learning algorithm which combines gradient descent method [29] and Kohonen algorithm [52] to obtain faster convergence of network parameters.

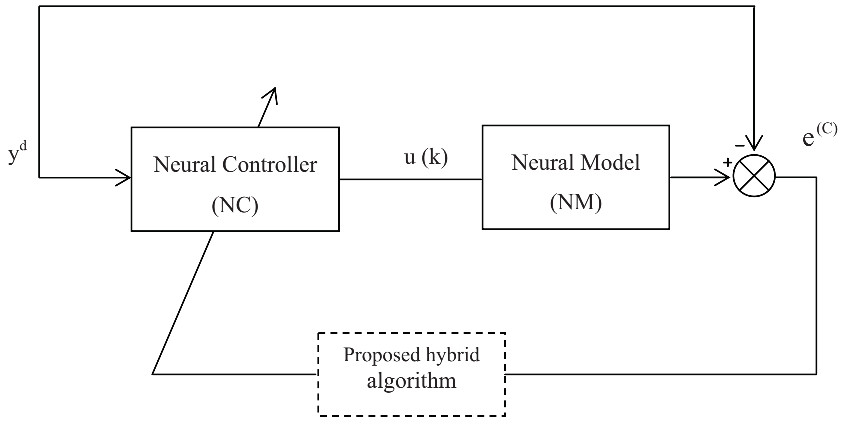

Fig. 2. Specialized learning architecture based on the hybrid algorithm.

Preliminary

The neural model used in this work is a feedforward neural network shown in Fig. 3. The network structure consists on an input layer, a hidden layer and an output layer.

The output of each neuron in the hidden layer is as follows:

$$o_j = \frac{1}{1+\exp(-\text{net}_j)} \tag{6}$$

$$\text{net}_j = \sum_i w_{ji} \cdot x_i - \theta_j, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, L \tag{7}$$

where o_j is the output of jth neuron in the hidden layer, x_i is the ith input variable, w_{ji} is the connection weight between the ith neuron in the input layer and the jth neuron in the hidden layer, θ_j is the threshold of the jth neuron in the hidden layer, n and L are the numbers of neurons in the input and hidden layers, respectively.

The output of each neuron in the output layer is as follows:

$$\hat{y}_k = \frac{1}{1+\exp(-\text{net}_k)} \tag{8}$$

(9)$\text{net}_k = \sum_j w_{kj} \cdot o_j - \theta_k, \quad k = 1, 2, \ldots, m$where w_{kj} is the connection weight between the jth neuron in the hidden layer and the kth neuron in the output layer, θ_k is the threshold of the kth neuron in the output layer, m is the numbers of neurons in the output layer.

The BP algorithm has become the standard algorithm used for training FNN. It is a generalized least mean squared algorithm that minimizes a criterion which equals to the sum of the squares of the errors between the actual and the desired outputs.

Training algorithm

We consider a FNN with three layers (Fig. 4): an input layer, a hidden layer and an output layer. The intermediate layer is considered like a self organizing map of Kohonen [52].

Our approach consists of two stages. In the first stage, shown in detail in Fig. 1, a neural network is trained to approximate the input–output behaviour of the plant. When the error between plant and the neural model becomes sufficiently small, we proceed to a second stage, shown in Fig. 2 that indicates the structure normally utilized for training the neural controller based on hybrid algorithm. The output of the block neural network controller is the input to neural model, obtained in the previous stage. This stage concerns the training procedure of the feedforward neural controller. It is composed of two phases.

During the first phase of training, the weight of the winning b is tuned by the difference between the input vector and the weight vector. Not only the winning node is learning but also its neighbourhood nodes are learned as well. Therefore, their weight vectors become more similar to the input pattern. As a result, the respective node is more likely to win at future presentations of this input

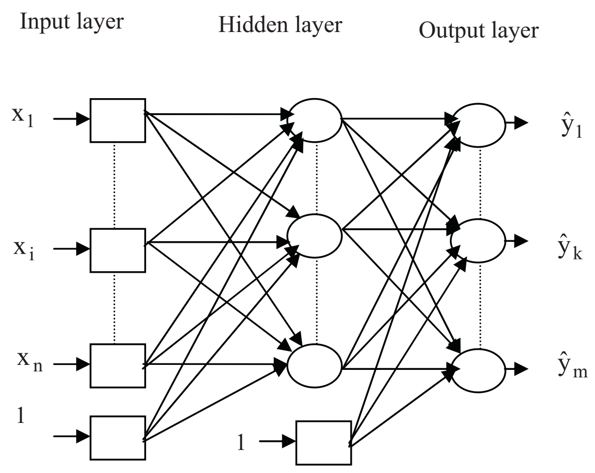

Fig. 3. The neural model.

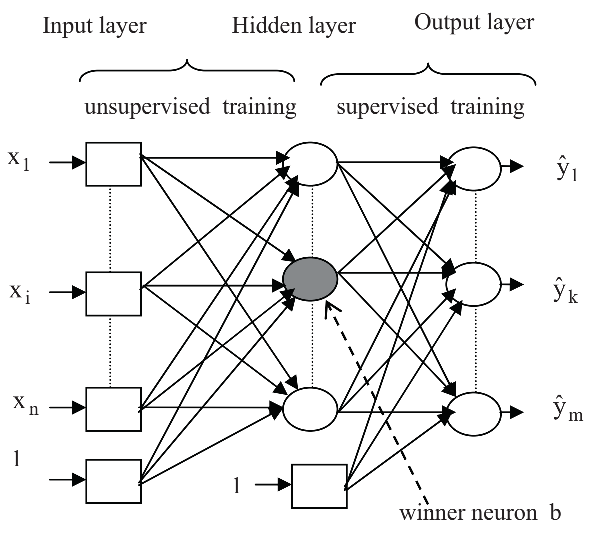

Fig. 4. Feedforward neural network with a linear self organizing map.

pattern. The effect of this procedure is that only a small number of hidden units are excited by a given input vector. If the weight vector of the winner is attracted to a region in input space, the neighbours are also attracted, although to a lesser degree.

During the second phase of training, weights leaving from the winner neuron and its neighbours are adjusted by the gradient descent method. The learning algorithm can be described as two stages process as below:

1.1.1. *First stage: neural model training*

The neural model (NM) is trained using the FN-based hybrid learning algorithm [45]. The weights from the input neurons to intermediate layer, and from the intermediate layer to the output neurons, are adjusted so as to minimize the cost function (4). For more details about FN-based hybrid, refer to [45].

The weight tuning update for the input layer using Kohonen procedure [52] is given by

$$\Delta w_{ji}^{(I)}(t)=\varepsilon^{(I)}(t)h_{bi}^{(I)}(t)(x-w_{ji}^{(I)}(t)) \tag{10}$$

where $\varepsilon^{(I)}(t)$ is the learning rate parameter and $h_{bi}^{(I)}(t)$ is the neighbourhood function centred around the winning b, both $\varepsilon^{(I)}(t)$ and $h_{bi}^{(I)}(t)$ are varied dynamically during learning for best result.

The weight tuning update for the output layer using gradient descent procedure [29] is given by

$$\begin{aligned}\Delta w_{kj}^{(I)}(t) &= -\alpha^{(I)}\frac{\partial E^{(I)}}{\partial w_{kj}^{(I)}}+\beta^{(I)}(t)\Delta w_{kj}^{(I)}(t-1)\\ &= -\alpha^{(I)}(t)(y_k-\hat{y}_k)\cdot\hat{y}_k(1-\hat{y}_k)o_j^{(I)}+\beta^{(I)}(t)\Delta w_{kj}^{(I)}(t-1)\end{aligned} \tag{11}$$

$\alpha^{(I)}(t)$ is the learning rate, $\beta^{(I)}(t)$ is the momentum rate. y_k and $\hat{y}_k$ are the plant and the neural network identifier outputs, respectively. $o_j^{(I)}$ is the output of jth neuron in the hidden layer.

1.1.2. *Second stage: neural controller training*

The step by step procedure of the learning control method is listed as follows:

Step 1: Initialize the weights at small random values.
Step 2: Present an input vector x to the NC.
Step 3: The winner neuron b is selected by evaluation of the distance measure between the input and the neuron weights [52]:

$$||x-w_b^{(C)}||=\min_i||x-w_b^{(C)}|| \tag{12}$$

The distance measure can be any distance norm, in the practice mostly the Euclidean one is used.
Step 4: Calculate

$$\varepsilon^{(C)}(t)=\varepsilon_0\ \exp\left(-\frac{t}{T}\right) \tag{13}$$

where ε_0 is its initial value, T is the total number of iterations and t is the current iteration.
Step 5: Calculate

$$\sigma^{(C)}(t)=\sigma_0\left(\frac{\sigma_f}{\sigma_0}\right)^{\frac{t}{T}} \tag{14}$$

where σ_0 and σ_f control the initial and final values of neighbourhood width.
Step 6: A Gaussian neighbourhood function is calculated as follows:

$$h_{bi}^{(C)}(t)=\varepsilon^{(C)}(t)\ \exp\left(-\frac{(b-i)^2}{2\sigma^{(c)^2}}\right),\quad h_{bi}^{(C)}(t)\in[0,1] \tag{15}$$

where b denotes the index of the winner neuron and i denotes the index of any neuron. The factors $\varepsilon^{(C)}(t)$ and $\sigma^{(C)}(t)$ are the learning rate factor and the neighbourhood width factor, respectively. They are decreasing functions of time. The neighborhood function $h_{bi}^{(C)}(t)$ usually is equal to 1 for the winner neuron and decrease with the distance of the neurons from the winner.
Step 7: The weights of the winner and its neighbouring neurons are modified as follows [52]:

$$\Delta w_{ji}^{(C)}(t)=h_{bi}^{(C)}(t)(x-w_{ji}^{(C)}(t)) \tag{16}$$

The weights leaving from the winner neuron and its neighbours are modified based on the gradient descent method [29]:

$$\begin{aligned}\Delta w_{kj}^{(C)}(t) &= -\alpha^{(C)}\frac{\partial E^{(C)}}{\partial w_{kj}^{(C)}}+\beta^{(C)}\cdot\Delta w_{kj}^{(C)}(t-1)\\ &= -\alpha^{(C)}(t)\cdot(\hat{y}_k-y^d)\cdot\sum_j w_{kj}^{(I)}\cdot\hat{y}_k\cdot(1-\hat{y}_k)\cdot w_{ji}^{(I)}\cdot o_j^{(I)}\\ &\quad\cdot(1-o_j^{(I)})\cdot u\cdot(1-u)\cdot o_j^{(C)}+\beta^{(C)}\cdot\Delta w_{kj}^{(C)}(t-1)\end{aligned} \tag{17}$$

$\alpha^{(C)}(t)$ is the learning rate, $\beta^{(C)}(t)$ is the momentum rate. y^d and $\hat{y}_k$ are the desired and the neural network identifier outputs, respectively. $o_j^{(C)}$ is the output of jth neuron in the hidden layer.
Step 8: The algorithm is stopped when the value of the error function (5) has become sufficiently small, otherwise go back to step 2.

Simulation results

In this section, simulation results are presented and discussed in order to evaluate the performance of the proposed algorithm and to compare it with the conventional BP algorithm (with a momentum term). Two examples are provided. First example is dynamical equation and second example is a general benchmark problem.

Simulation results were obtained using a personal computer Pentium 4 2.8 GHz.

Example 1

The nonlinear dynamical system [24] to be controlled is given by:

$$y(k+1)=\frac{y(k)}{1+y^2(k)}+u^3(k) \tag{18}$$

The objective in the present example is to control the plant so as to track a desired output given by the following 100 samples of data.

$$y^d=1.5\left(\frac{\sin 2\pi k}{10}+\frac{\sin 2\pi k}{25}\right) \tag{19}$$

The model has two inputs $u(k)$ and $y(k)$ and a single output $y(k+1)$, and system identification was initially performed with the plant input with amplitude uniformly distributed over the interval [−2,+2]. Training samples and testing samples contained 100 and 100 data points, respectively. The desired output was normalized in the interval [0.1, 0.9]. The controller is a one hidden layer FNN, with 20 neurons in the hidden layer and one neuron in the output layer. The neurocontroller uses the two inputs, the current state $y(k)$ and the desired state $y^d(k)$, to produce an output which is $u(k)$. The initial weight values of the conventional BP are set to small random values.

A learning rate of 0.015 and momentum of 0.15 is used in BP algorithm and parameters for the proposed algorithm are: $\varepsilon_0=0.5$,

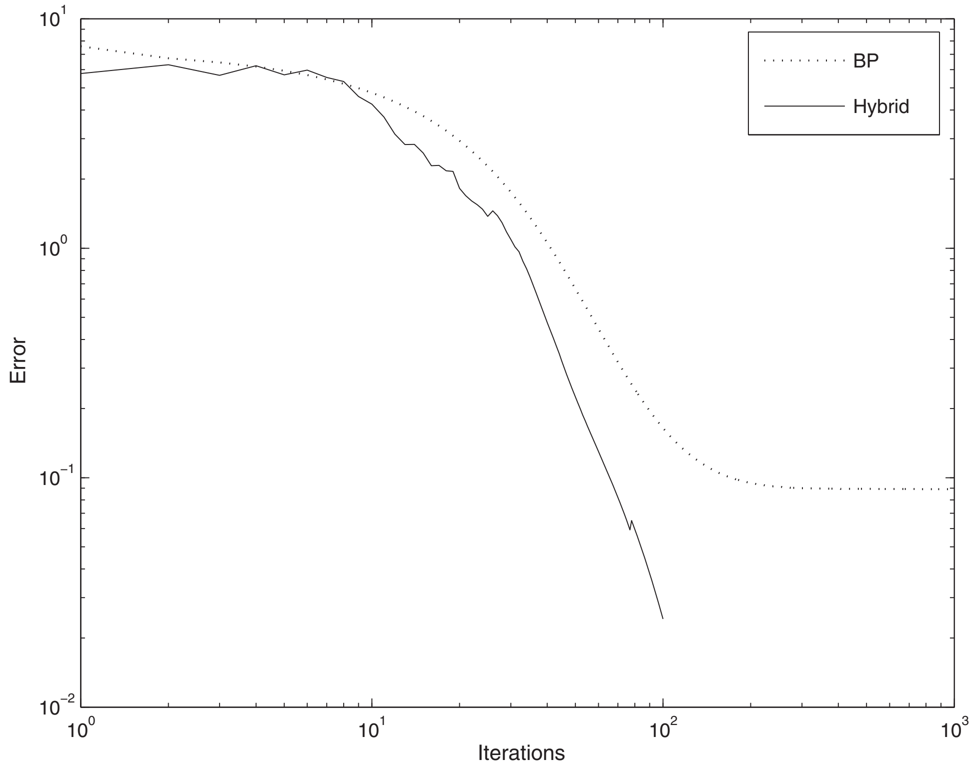

Fig. 5. The evolution of the neural controller training criterion for the hybrid versus BP for example 1.

$\alpha^{(C)} = 0.15$, $\beta^{(C)} = 0.15$, $\sigma_0 = 2$, $\sigma_f = 0.5$. These parameters are determined by trial and error tests.

Fig. 5 compares the mean squared control errors of the two methods. It can be seen from Fig. 5 that the hybrid algorithm converges rapidly than the BP algorithm.

Figs. 6 and 7 show the performance of the controllers with hybrid algorithm and BP algorithm, respectively.

Example 2

In [53], Box and Jenkins gave 296 pairs of data measured from a gas furnace system with a single input $u(t)$ being gas flow rate and a single output $y(t)$ being CO_2 concentration in outlet gas.

Following previous researchers [54] in order to make a meaningful comparison, the inputs of the prediction model are selected

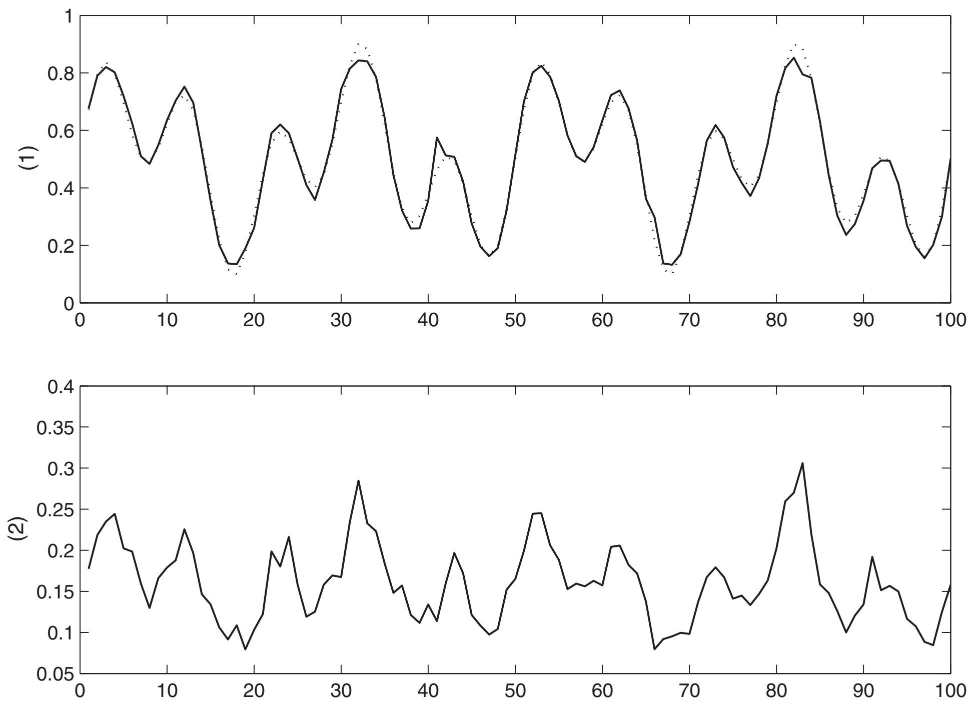

Fig. 6. Performance of the controller with hybrid algorithm for example 1: (1) output (solid lines) and reference (dashed lines); (2) control input.

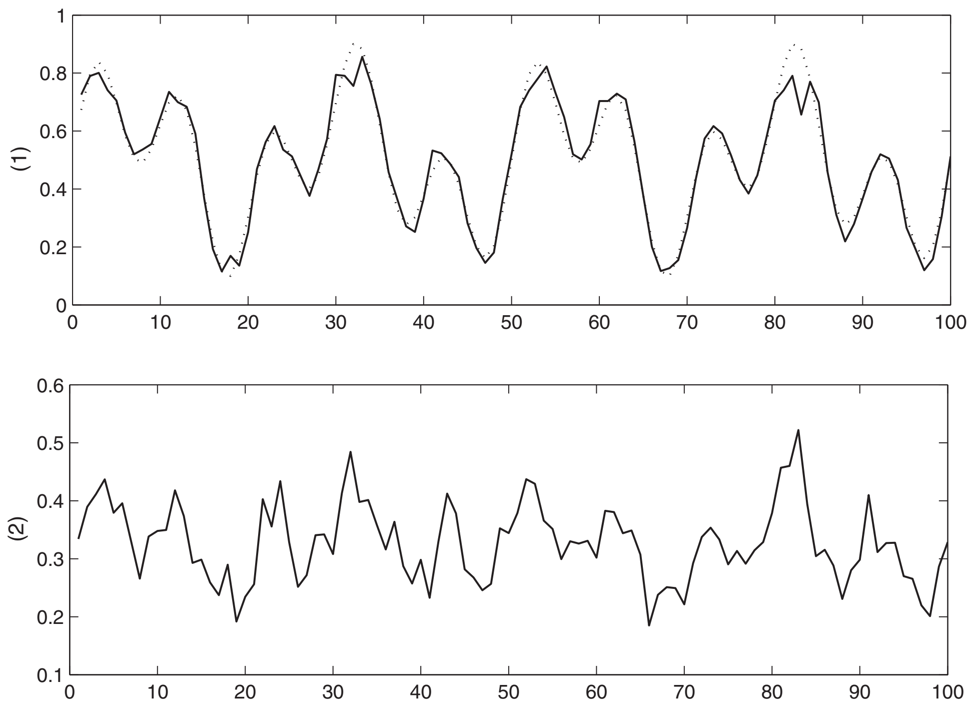

Fig. 7. Performance of the controller with BP algorithm for example 1: (1) output (solid lines) and reference (dashed lines); (2) control input.

as $u(t-4)$ and $y(t-1)$, and the output is $y(t)$ The aim of control task is to keep the concentration as close as possible to the reference trajectory by manipulating the flow rate. Training samples and testing samples contained 196 and 100 data points, respectively.

The desired output was normalized in the interval [0.1, 0.9]. The controller is a one hidden layer FNN, with 20 neurons in the hidden layer and one neuron in the output layer. The neurocontroller uses the two inputs, the current state $y(k)$ and the desired state $y^d(k)$, to produce an output which is $u(k)$. The initial weight values of the conventional BP are set to small random values. A learning rate of 0.2 and momentum of 0.5 is used in BP algorithm and parameters for the proposed algorithm are: $\varepsilon_0 = 0.5$, $\alpha^{(C)} = 0.15$, $\beta^{(C)} = 0.15$, $\sigma_0 = 2$, $\sigma_f = 0.5$. These parameters are determined by trial and error tests.

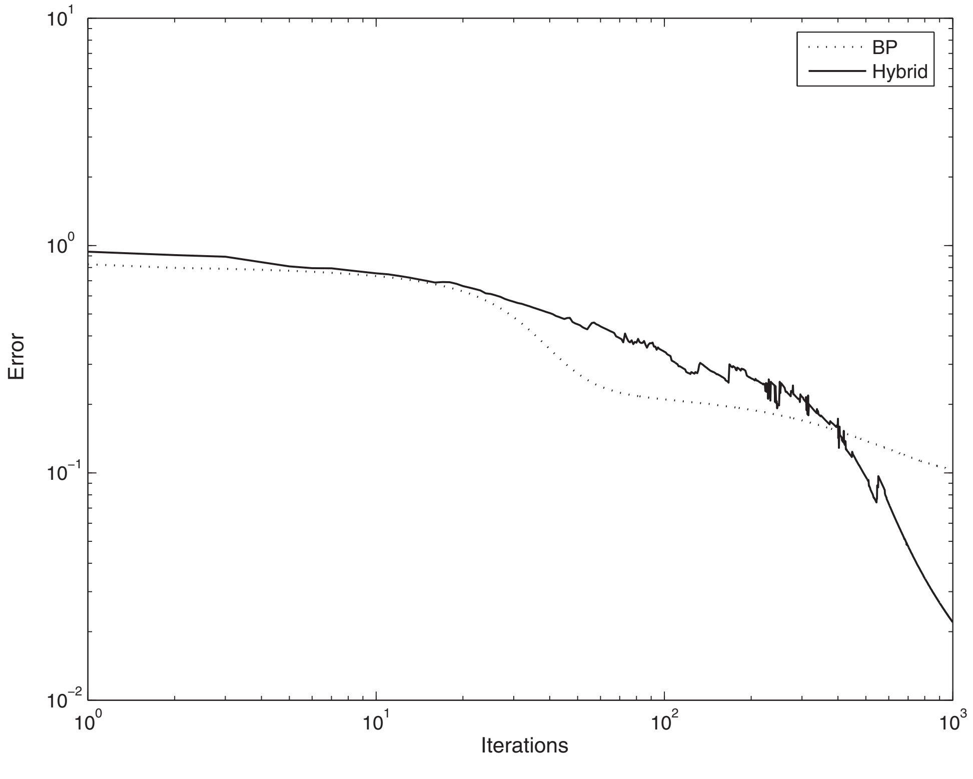

Fig. 8. The evolution of the neural controller training criterion for the hybrid versus BP for example 2.

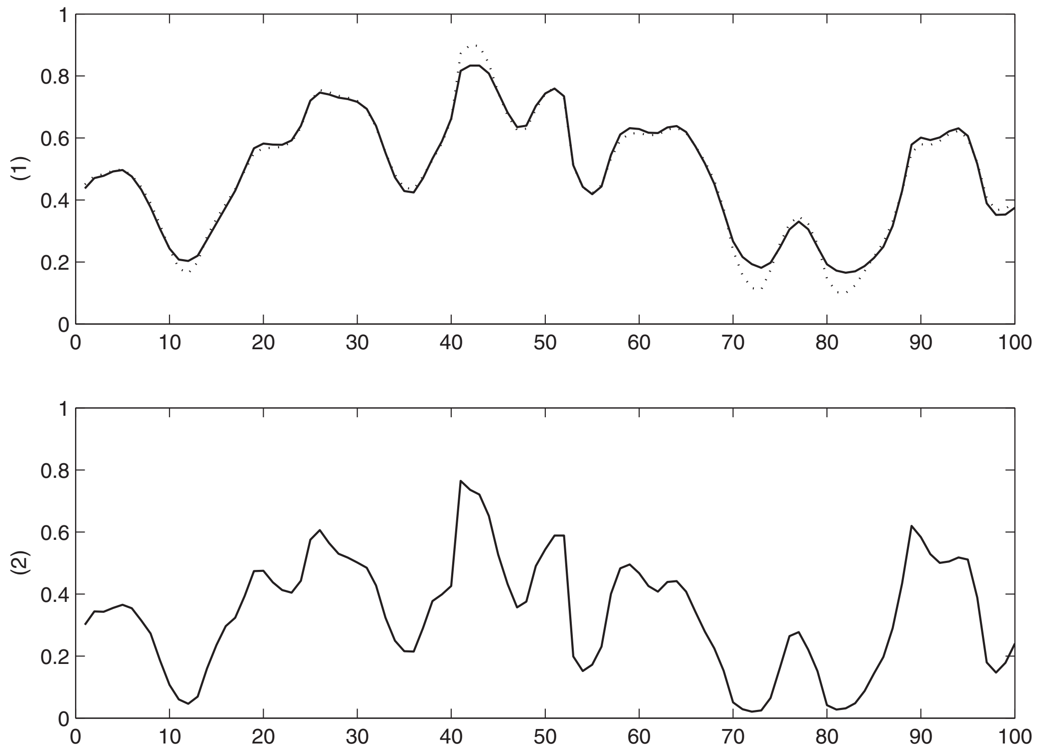

Fig. 9. Performance of the controller with hybrid algorithm for example 2: (1) output (solid lines) and reference (dashed lines); (2) control input.

Fig. 8 compares the mean squared control errors of the two methods. It can be seen from Fig. 8 that the hybrid algorithm converges rapidly than the BP algorithm.

Figs. 9 and 10 show the performance of the controllers with hybrid algorithm and BP algorithm, respectively.

Discussion and results

The algorithm presented here enables us to define an appropriate number of neuron in the hidden layer of Feedforward neural network, based on clustering method. The number of active hidden

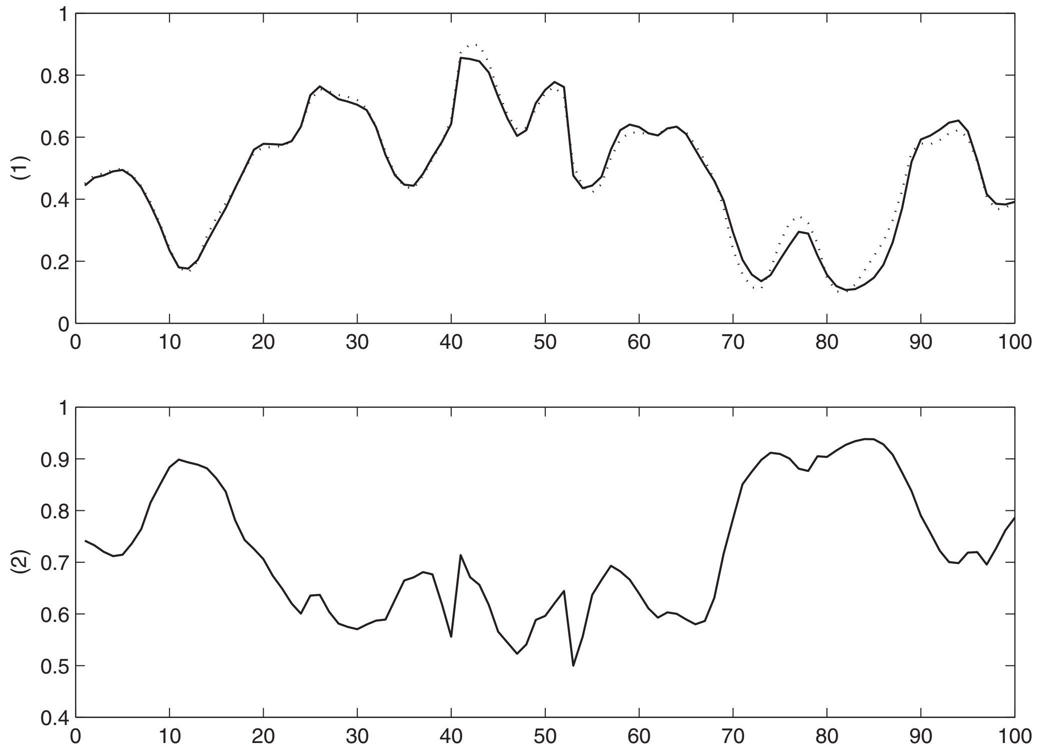

Fig. 10. Performance of the controller with hybrid algorithm for example 2: (1) output (solid lines) and reference (dashed lines); (2) control input.

Table 1
Comparative results of learning methods.

Example	Learning method	Testing samples size	Number of iterations	Total time of convergence(s)	Training error of the NC	Testing error
Example 1	BP	100	340	2.375	0.089	0.1
	Hybrid	100	68	0.544	0.089	0.071
Example 2	BP	100	1000	8.560	0.1	0.1
	Hybrid	100	488	4.104	0.1	0.05

neuron directly affects the generalization and training time, which are two important factors in neural network modelling. The backpropagation algorithm updates all network weights after every input–output case. The total number of weights is where n, N, and m denote the input vector dimension, the hidden node number and the output node number, respectively.

Letting P denotes the number of the selected hidden nodes $(1 \leq P < N)$, the hybrid algorithm updates only parameters of the selected hidden nodes after every input–output case. The total number of weights becomes $\omega_h = (n+1) \cdot P + (P+1) \cdot m$.

The computed weighted sum number in the case of the proposed hybrid algorithm is inferior to that of the backpropagation algorithm since P is always inferior to N. The nonlinearity computation number depends on the selected hidden node number (P) for each presented pattern. This fact makes the learning speed of the proposed algorithm is faster than that of the backpropagation.

The comparative performances of the proposed training algorithm and Backpropagation BP algorithm, such as the number of iterations, total time of convergence and training error are summarized in Table 1. It can be seen that the proposed training algorithm takes minimum number of iterations for convergence as compared to the standard Backpropagation algorithm in all two examples. From the results we can also see, when achieving the same error, the proposed training algorithm is faster than the popular BP in terms of computation time.

Conclusion

A new neural control method was developed and presented. The method is based on the concept of combining Kohonen algorithm and gradient descent method. The key ideas explored are the use of the Kohonen algorithm for training the weights of the hidden layers and gradient descent method for training the weights of the output layer. The performances of the neurocontroller based hybrid training are tested using two different examples and compared with the standard backpropagation algorithm. The results show that the new algorithm can greatly improve the learning rate of the neural network control structure.

Acknowledgement

The author would like to thank the reviewers for their valuable comments that improved this paper.

References

[1] N. Yao, M. Tomizuka, Adaptive robust control of MIMO non linear system in semi-strict feedback forms, Automatica 37 (9) (2001) 1305–1321.
[2] W.M. Haddad, T. Hayakawa, V. Chellaboina, Adaptive control for uncertain systems, Automatica 39 (3) (2003) 551–556.
[3] H. Elmali, N. Olgac, Robust output tracking control of nonlinear MIMO system via sliding mode technique, Automatica 28 (1) (1992) 145–151.
[4] N. Sadati, R. Ghadami, Adaptive multi-model sliding mode control of robotic manipulators using soft computing, Neurocomputing 71 (2008) 2702–2710.
[5] I. KanellaKopoulas, P.V. Kokotovic, A.S. Morse, Systematic design of adaptive controllers for feedback linearizable systems, IEEE Trans. Autom. Control 36 (11) (1991) 1241–1253.
[6] P.V. Kokotovic, The joy feedback: nonlinear and adaptive, IEEE Control Syst. Mag. 12 (3) (1992) 7–17.
[7] M. Liu, Decentralized control of robot manipulators: nonlinear and adaptive approaches, IEEE Trans. Autom. Control 44 (2) (1999) 357–366.
[8] Y. Guo, D.J. Hill, Y. Wang, Nonlinear decentralized control of large scale power systems, Automatica 36 (9) (2000) 1275–1289.
[9] K. Theofilatos, G. Beligiannis, S. Likothanassis, Combining evolutionary and stochastic gradient techniques for system identification, J. Comput. Appl. Math. 227 (1) (2009) 147–160.
[10] G.N. Beligiannis, L.V. Skarlas, S.D. Likothanassis, K.G. Perdikouri, Nonlinear model structure identification of complex biomedical data using a genetic programming-based technique, IEEE Trans. Instrum. Meas. 54 (6) (2005) 2184–2190.
[11] L. Chen, K. Narendra, Identification and control of non linear discrete time system based on its linearization: a unified framework, IEEE Trans. Neural Netw. 15 (3) (2004) 663–673.
[12] Y.K. Kwon, B.R. Moon, A hybrid neurogenetic approach for stock forecasting, IEEE Trans. Neural Netw. 18 (3) (2007) 851–864.
[13] Y. Zhang, T. Chai, H. Wang, A nonlinear control method based on ANFIS and multiple models for a class of SISO nonlinear systems and its application, IEEE Trans. Neural Netw. 22 (11) (2011) 1783–1795.
[14] Y.M. Li, S.C. Tong, Y. Liu, T. Li, Adaptive fuzzy robust output feedback control of nonlinear systems with unknown dead zones based on a small – gain approach, IEEE Trans. Fuzzy Syst. 22 (1) (2014) 164–176.
[15] A.Y. Alanis, E.N. Sanchez, A.G. LouKianov, Discrete-time adaptive backstepping nonlinear control via high order neural networks, IEEE Trans. Neural Netw. 18 (4) (2007) 1185–1195.
[16] C.M. Lin, A.B. Ting, M.C. Li, Neural network-based robust adaptive control for a class of nonlinear systems, Neural Comput. Appl. 20 (2011) 557–563.
[17] Y. Wen, X. Ren, Neural networks-based adaptive control for nonlinear time varying delays systems with unknown control direction, IEEE Trans. Neural Netw. 22 (9) (2011) 1599–1612.
[18] S.C. Tong, Y.M. Li, H.G. Zhang, Adaptive neural network decentralized backstepping output-feedback control for nonlinear large scale systems with time delays, IEEE Trans. Neural Netw. 22 (7) (2011) 1073–1086.
[19] I. Rivals, L. Personnaz, Nonlinear internal model control using neural networks: application to processes with delay and design issues, IEEE Trans. Neural Netw. 11 (1) (2000) 80–90.
[20] M. Han, J. Fan, J. Wang, A dynamic feedforward neural based on Gaussian particle swarm optimization and its application for predictive control, IEEE Trans. Neural Netw. 22 (9) (2011) 1457–1468.
[21] M. Chen, S.S. Ge, B.V.E. How, Robust adaptive neural network control for a class of uncertain MIMO nonlinear systems with input nonlinearities, IEEE Trans. Neural Netw. 21 (5) (2010) 796–812.
[22] T. San, H. Pei, Y. Pan, H. Zhou, C. Zhang, Neural networks based sliding mode adaptive control for robot manipulators, Neurocomputing 74 (2011) 2377–2384.
[23] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.
[24] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Netw. 1 (1) (1990) 4–27.
[25] K.J. Hunt, D. Sbarbaro, R. Zbikowski, P.J. Gawthrop, Neural networks for control systems – a survey, Automatica 28 (1992) 1083–1112.
[26] P.J. Werbos, An overview of neural networks for control, IEEE Control Syst. Mag. 11 (1991) 40–41.
[27] D. Psaltis, A. Sideris, A.A. Yamamura, A multilayer neural network controller, IEEE Control Syst. Mag. 8 (1988) 17–21.
[28] D.N. Nguyen, B. Widrow, Neural networks for self – learning control systems, IEEE Control Syst. Mag. 10 (3) (1990) 18–23.
[29] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation Parallel Distributed Processing: Explorations in the Microstructures of Cognition, vol. 1, MIT Press, 1986, pp. 318–362.
[30] N. Zhang, W. Wu, G. Zheng, Convergence of gradient method with momentum for two-layer feedforward neural networks, IEEE Trans. Neural Netw. 17 (2) (2006) 522–525.
[31] J. Wang, J. Yang, W. Wu, Convergence of cyclic and almost – cyclic learning with momentum for feedforward neural networks, IEEE Trans. Neural Netw. 22 (8) (2011) 1297–1306.
[32] L. Behera, S. Kumar, A. Patnaik, On adaptive learning rate that guarantees convergence in feedforward in feedforward networks, IEEE Trans. Neural Netw. 17 (5) (2006) 1116–1125.
[33] M. Kordos, W. Duch, Variable step search algorithm for feedforward neural networks, Neurocomputing 71 (2008) 2470–2480.
[34] J.C. Zweiri, J.F. Whidborne, L.D. Seneviratne, Three-term backpropagation algorithm, Neurocomputing 50 (2003) 305–318.

[35] W. Wan, S. Mabu, K. Shimada, K. Hirasawa, J. Hu, Enhancing the generalization ability of neural networks through controlling the hidden layers, Appl. Soft Comput. 9 (2009) 404–414.
[36] S. McLoone, G. Irwin, Improving neural network training solution using regularisation, Neurocomputing 37 (2001) 71–90.
[37] R. Zhang, Z. Xu, G.B. Huang, D. Wang, Global convergence of online BP training with dynamic learning rate, IEEE Trans. Neural Netw. Learn. Syst. 23 (2) (2012) 330–341.
[38] M. Jahangir, M. Golshan, S. Khosravi, H. Afkhami, Design of a fast convergent backpropagation algorithm based on optimal control theory, Nonlinear Dyn. 70 (2012) 1051–1059.
[39] A. Bortoletti, C. Di FIore, S. Fanelli, P. Zellini, A new class of Quasi-Newtonian methods for optimal learning in MLP-networks, IEEE Trans. Neural Netw. 14 (2) (2003) 263–273.
[40] G. Lera, M. Pinzolas, Neighborhood based Levenberg-Marquardt algorithm for neural network training, IEEE Trans. Neural Netw. 13 (5) (2002) 1200–1203.
[41] B.M. Wilamowski, H. Yu, Improved computation for Levenberg-Marquart, IEEE Trans. Neural Netw. 21 (6) (2010) 930–937.
[42] S.R. Nickolai, The layer-wise method and the backpropagation hybrid approach to learning a feedforward neural network, IEEE Trans. Neural Netw. 11 (2) (2000) 295–305.
[43] P. Liu, H. Li, Efficient learning algorithms for three-layer regular feedforward fuzzy neural networks, IEEE Trans. Neural Netw. 15 (3) (2004) 545–558.
[44] M. Ben Nasr, M. Chtourou, A hybrid training algorithm for feedforward neural networks, Neural Process. Lett. 24 (2) (2006) 107–117.
[45] M. Ben Nasr, M. Chtourou, A fuzzy neighborhood-based training algorithm for feedforward neural networks, Neural Comput. Appl. 18 (2) (2009) 127–133.
[46] M. Ben Nasr, M. Chtourou, A self organizing map-based initialization for hybrid training of feedforward neural networks, Appl. Soft Comput. 11 (2011) 4458–4464.
[47] T.C. Liu, R.L. Li, A new ART-counterpropagation neural network for solving a forecasting problem, Expert Syst. Appl. 28 (2005) 21–27.
[48] S. McLoone, M.D. Brown, G. Irwin, G. Lighbody, A hybrid linear/nonlinear training algorithm for feedforward neural networks, IEEE Trans. Neural Netw. 9 (4) (1998) 669–684.
[49] H. Malek, M.M. Ebadzadeh, M. Rahmati, Three new fuzzy neural networks learning algorithm based on clustering, training error and genetic algorithm, Appl. Intell. 37 (2012) 280–289.
[50] J.B.D. Cabrera, K.S. Narendra, Issues in the application of neural networks for tracking based on inverse control, IEEE Trans. Autom. Control 44 (11) (1999) 2007–2027.
[51] G.J. Jeon, I. Lee, Neural networks indirect adaptive control with fast learning algorithm, Neurocomputing 13 (1996) 185–199.
[52] T. Kohonen, The self-organizing map, Proc. IEEE 78 (9) (1990) 1464–1480.
[53] G.E. Box, G.M. Jenkins, Time Series Analysis, Forecasting and Control, Holden Day, San Francisco, 1970.
[54] J. Kim, Adaptive neuro-fuzzy inference system and their application to non linear dynamical system, Neural Netw. 12 (1999) 1301–1319.