第 14 卷第 5 期
1997 年 10 月

控制理论与应用
CONTROL THEORY AND APPLICATIONS

Vol. 14, No. 5
Oct., 1997

A Neural Fuzzy Logic Self-Organizing Controller for Nonlinear System Control

WANG Yaonan
(Department of Electronic Engineering, Hunan University · Changsha, 410082, PRC)

Abstract: A neural fuzzy logic self-organizing controller is proposed and applied to control of a non-linear learning tracking problem. The proposed controller can self-adjust its control rules by modifying the weights of the network on-line such that it can improve response performance of the nonlinear plant. In this paper, a weight-learning algorithm is presented using the gradient descent method with a veriable-slopes, which is useful to accelerate learning and improve convergence, The effectiveness of the proposed control scheme is illustrated through simulations.

Key words: fuzzy logic; neural networks; fuzzy self-organizing control; intelligent control

1 Introduction

Conventional fuzzy logic controllers generally require a certain reasonable set of fuzzy rules that integrate heuristics and intuition of human operators. The most important and difficult problem in the fuzzy logic controller design is how to obtain the proper control rule for a given plant, The control rules may be obtained from the operator's control action, the expert's experience, control engineering knowledge, or the fuzzy model of the process. However in the case of a system that has very complicated dynamic characteristics such as robotic manipulators, we would encounter significant difficulties to find the best fitted or at least reasonable fuzzy rules to such a system.

The fuzzy logic self-organizing controller (FLSOC) proposed by Procyk and Mamdani[1] provides an adaptive rule-accumulating capability and overcome some difficulties of the control rule generating from the design requirement. The FLSOC makes use of reinforcement learning to generate new or to modify the existing rules until a successful control strategy is developed. The main problem which restricts the application of the FLSOC is the unclarity of the rules learning and accumulating procedure. Also, the modification of the control rules in real-time is usually time consuming. To solve this problem, we present methods that are neural network to establish fuzzy associations (fuzzy antecedent processing, fuzzy inferencing, and defuzzification). The fuzzy logic self-organizing controller is implemented by adjusting the weights of the network to improve dynamic response performance (e. g. , risetime, overshoot, steady-state error) for control of complicated plant.

2 Neural Network Fuzzy Logic Self-Organizing Control (NFSOC)

The FLSOC is capable of generating and modifying control rules based on evaluation of the system performance. It consists of three basic function bolcks: 1) a control performance measure table; 2) an incremental model; and 3) controller modification algorithms[1].

Manuscript received Dec. 14, 1994, revised Dec. 16, 1996.

For the FLSOC implemented[1], the rule modification is achieved either by modifying the relation matrix(M)or by replacing individual rules by new rules. The modification of the relation matrix is usually very time consuming because of a large size of the relation matrix. If the individual rules are modified, each rule change leads to three new rules. The number of rules therefore become very large as the control evolves. To solve this problem, we present the neural network based fuzzy logic self-organizing controller(NFSOC)as shown in Fig. 1. The controller are implemented through a neural network and rule modification procedure is replaced by the weight-learning algorithm.

The performance measure and process model used in the NFSOC are the same as in the FLSOC.

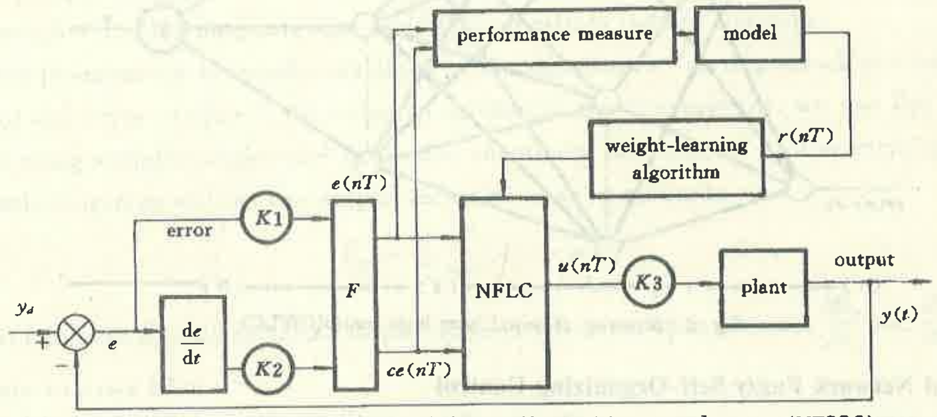

Fig. 1 Structure of the neural network fuzzy self-organizing control system (NFSOC)

2.1 Neural Fuzzy Logic Controller(NFLC)

The structure of a neural fuzzy logic controller is shown in Fig. 2. Assume that each control rule has two input variable (e, ce) and output variable(u).

In Fig. 2, the layers (Ⅰ)~(Ⅱ)correspond to the antecedent part of the fuzzy control rules. and the layer(Ⅲ)~(Ⅳ)correspond to conclusion part. The input-output relationships of the units in the NFLC are defined as

1) Output units: $$O_{1k}=x_k \quad k=1,2. \tag{1}$$

2) Output units: $$O_{2j}=f(s_j, I_{2j}), \tag{2}$$

Input untis: $$I_{2j}=\sum_{k=1}^{n} W_{jk}\cdot x_k, \quad j=1,2\cdots,m. \tag{3}$$

3) Output units: $$O_{3j}=f(s_j, I_{3j}), \tag{4}$$

Input untis: $$I_{3i}=\sum_{j=1}^{m} W_{ij}O_{2j} \quad i=1,2\cdots,m_1. \tag{5}$$

4) Output units: $$u_p=\left[\frac{\sum_{i=0}^{m_1} W_{pi}\cdot O_{3i}}{\sum_{j=1}^{m_1} O_{3j}}\right], \quad p=1. \tag{6}$$

Where x_k is the kth input pattern to the network; W_{jk}, W_{ij} and W_{pi} are the connection weight values of the NFLC; $f(\cdot)$ is defined as $f(S,I)=\dfrac{1-e^{-SI}}{1+e^{-SI}}$, the parameter S specifies the steepness of the curve.

In order to train the NFLC to establish fuzzy logic control associations, first, the training of the NFLC is performed off-line by using numerical patterns $(e, ec \rightarrow u)$.

Once the weights of the NFLC are obtained, the NFLC can be applied to the NFSOC.

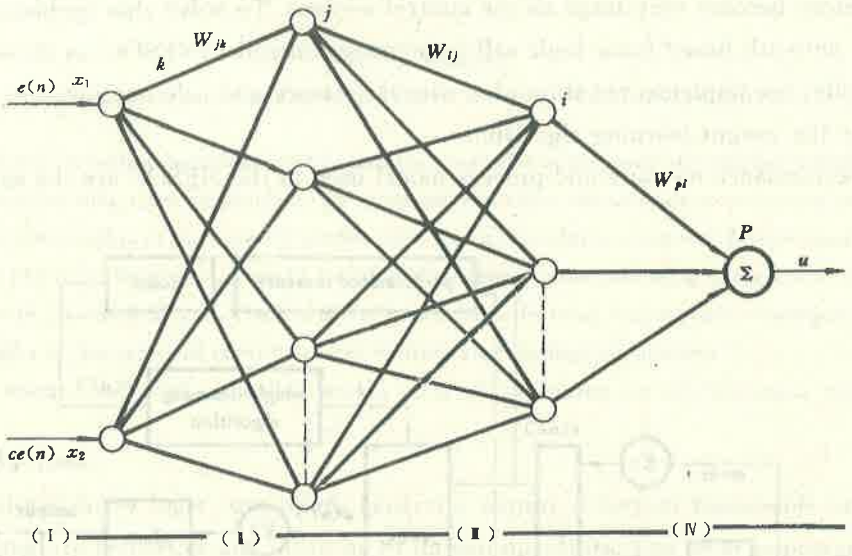

Fig. 2 Structure of neural fuzzy logic control (NFLC)

2.2 Neural Network Fuzzy Self-Organizing Control

For the NFSOC shown in Figure 1, the controller modification can be performed by adjust the weight (W_{jk}, W_{ij}, W_{pi}) of the network such that the input pattern at time $(n-m)T$ now maps into an output $u(nT-mT)+r(nT)$ rather than $u(nT-mT)$. The weight of the network can modified this maping by using the back-propagation algorithms[3] as follows:

The error function E_p is defined as

$$E_p = \frac{1}{2}\sum_{i=1}^{n}(u_{di} - u_i)^2, \tag{7}$$

where u_{di} is the desirable output, u_i is the actual output from the network. To minimize E the weight of the NFLC are adjusted by the equation

$$W_{pi}(t+1) = W_{pi}(t) - \eta\frac{\partial E_p}{\partial W_{pi}} + \alpha[W_{pi}(t) - W_{pi}(t-1)], \tag{8}$$

$$W_{ij}(t+1) = W_{ij}(t) - \eta\frac{\partial E_p}{\partial W_{ij}} + \alpha[W_{ij}(t) - W_{ij}(t-1)], \tag{9}$$

$$W_{jk}(t+1) = W_{jk}(t) - \eta\frac{\partial E_p}{\partial W_{jk}} + \alpha[W_{jk}(t) - W_{jk}(t-1)], \tag{10}$$

where η is a suitably small number known as step size, α is the momentum constant. The gradient of with respect to W can be derived as follows:

$$\frac{\partial E_p}{\partial W_{pi}} = \delta_p \cdot O_{3i}, \quad \frac{\partial E_p}{\partial W_{ij}} = \delta_j \cdot O_{2j}, \quad \frac{\partial E_p}{\partial W_{jk}} = \delta_j \cdot X_k, \tag{11}$$

where
$$\delta_p = -\left[\frac{u_{di} - u_i}{\sum_{i=1}^{m_1} O_{3i}}\right], \quad \delta_i = f_{I_{3i}}(S_i, I_{3i}) \sum_p \delta_p \cdot W_{pi}, \tag{12}$$

$$\delta_j = f'_{I_{2i}}(S_j, I_{2j}) \sum_i \delta_i \cdot W_{ij}. \tag{13}$$

Compute the deriveation of $f(S,I)$,using

$$\begin{aligned} f'_{I_{3i}}(S_i, I_{3i}) &= (S_i/2)(1 - f^2(S_i, I_{3i})), \\ f'_{I_{2j}}(S_j, I_{2j}) &= (S_j/2)(1 - f^2(S_j, I_{2j})). \end{aligned} \tag{14}$$

2.3 Learning Method Using Variable Slopes

Since the convergence of the back-propagation algorithm is very sensitive to the initial set of weights and learning rate, one finds that algorithm fails to converge.

This phenomenon is usually explained as the algorithm's "getting struck at a local minimum" of the error surface[6]. In order to accelerate the convergence, we use the learning method using variable slopes, and determine adaptively the slope of nonlinearitys associated with each neuron as well as the weight vector W so as to minimize

$$E_p = \frac{1}{2}\sum_{i=1}^{n}(u_{di} - u_i)^2. \tag{15}$$

To minimize E_p with response to the slopes S ,we must then evaluate $\frac{\partial E_p}{\partial S_i}$ and $\frac{\partial E_p}{\partial S_j}$. using the chain rule, we have

$$\frac{\partial E_p}{\partial S_i} = \frac{\partial E_p}{\partial u_i}\frac{\partial u_i}{\partial O_{3i}}\frac{\partial O_{3i}}{\partial S_i}, \tag{16}$$

since

$$\frac{\partial O_{3i}}{\partial S_i} = f'_{s_i}(S_i, I_{3i}) = (I_{3i}/2)(1 - f^2(S_i, I_{3i})), \tag{17}$$

$$\frac{\partial u_i}{\partial O_{3i}} = \left[\frac{W_{pi}(\sum_{i=1}^{m_1} O_{3i}) - \sum_{i=1}^{m_1} W_{pi} \cdot O_{3i}}{(\sum_{i=1}^{m_1} O_{3i})^2}\right] \tag{18}$$

$$\frac{\partial E_p}{\partial u_i} = -(u_{di} - u_i).$$

We have

$$\frac{\partial E}{\partial S_i} = \left\{-(u_{di} - u_i)\left[\frac{W_{pi}(\sum_{i=1}^{m_1} O_{3i}) - \sum_{i=1}^{m_1} W_{pi} \cdot O_{3i}}{2(\sum_{i=1}^{m_1} O_{3i})^2}\right](1 - f^2(S_i, I_{3i}))\right\} I_{3i} = \delta_{s_i} \cdot I_{3i}, \tag{19}$$

$$\frac{\partial E}{\partial S_j} = \left(\frac{\partial E_p}{\partial u_j} \cdot \frac{\partial u_j}{\partial O_{3i}} \cdot \frac{\partial O_{3i}}{\partial I_{3i}} \cdot \frac{\partial I_{3i}}{\partial O_{2j}}\right) \cdot \frac{\partial O_{2j}}{\partial S_j} = \left(\sum_i \delta_{s_I} \cdot W_{ij} \cdot S_i\right) f'_{s_i}(S_j, I_{2j}), \tag{20}$$

where

$$f'_{s_j} = (I_{2_j}/2)(1 - f^2(S_j, I_{2j})). \tag{21}$$

The slopes of the activation function are updated according to the Delta rule by

$$S_i(t+1) = S_i(t) - \beta \frac{\partial E_p}{\partial S_i} + \rho \Delta S_i(t), \tag{22}$$

$$S_j(t+1) = S_j(t) - \beta \frac{\partial E}{\partial S_j} + \rho \Delta S_j(t), \tag{23}$$

where $S(t)$ is the slopes at time t, β is the step size, ρ is the momentum constant.

The new algorithm has only two more terms than the classical algorithm (BP) : δ_{s_j} and $f'_s(S,I)$,The term $f_s(S,I)$ has the same computational complexity as the term$f'_s(S,I)$. The term δ_{s_j} is used to update the weights, are also used to update the slopes. Therefore, the new algorithm does not have any more computation burden than the classical one. Application results show that the variable-slopes algorithm converges faster than the BP algorithm.

3 System Simulation Results

In this section, the proposed controller is illustrated using two nonlinear plant. To implement the NFSOC, we selected the network of the 2-14-7-1 neuron. Initially, the weights of the network were chosen randomly, the parameters of the updating weights were set at $\eta = 0.1, \alpha = 0.2, \beta = 0.005, \rho = 0.001$, and $S_{\min} = 0.2$. The values of constants in Fig. 1 were chosen to be $K_1 = 1.0, K_2 = 1.82, K_3 = 0.3$.

Consider a nonlinear plant model described by.

Example 1

$$y(k) = 0.2y^2(k) + 0.2y(k-1) + 0.4\sin[0.5(y(k)+y(k-1))] \cdot \cos[0.5(y(k)+y(k-1))] + 1.2u(k).$$

The output trackig trajectories obtained by applying desired sine wave output, is shown in Fig. 3.

Example 2

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

The output tracking trajectories obtained by apply desired square wave output as shown in Fig. 4.

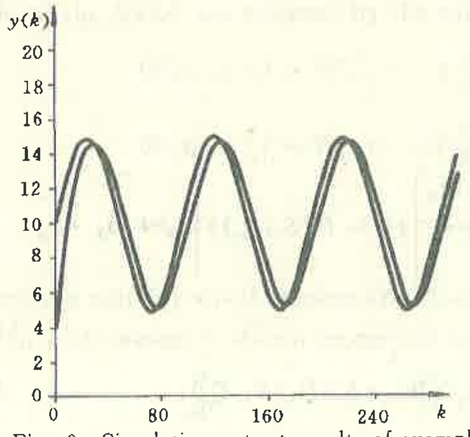

Fig. 3 Simulation output results of example 1

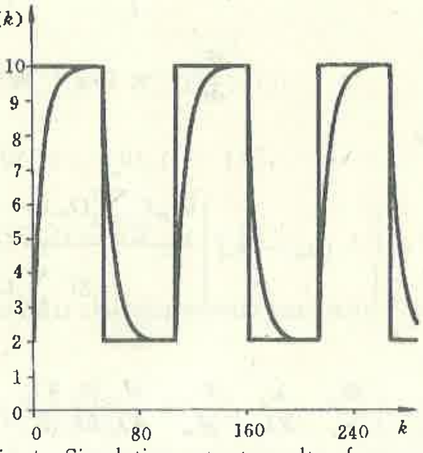

Fig. 4 Simulation output results of example 2

4 Conclusion

We have used multilayered neural networks to construct a fuzzy logic self-organizig controller for a class of nonlinear control system. The simulation show that the controller can improve response performance of the plant. The procedure used for modification of parameters in NFSOC is much simpler than the rule modification procedure in the FLSOC. The problem of rule explosion associated with the FLSOC does not arise in the NFSOC. Also, the computaion time for evaluation of a control action remains constant because of the forward propagation time of the network does not change. Thus, the NFSOC is suitable for nonlinear control system.

References

1　Procyk, I. J. and Mamdani, E. H.. A linguistic self-organizing process controller. Automatica, 1979, 15(1): 15--30

2　Lee, C. C.. Fuzzy logic in control system: Fuzzy logic controller-part 1. IEEE Tran, Syst. Man Cybern, 1990, 20(2): 408—435

3　Rumelhart. D. and Mcclelland, J.. Parallel distributed processing. Cambridge. MA: MIT. Press, 1986

4　Kosko, B.. Neural networks and fuzzy systems. Prentice Hall, 1992

5　Wang yao nan. Intelligent control integrating expert system and neural networks. Int. J. Advances in Modelling & Analysis, c, 1994, (43): 23—28

6　王耀南．智能控制系统——模糊逻辑，专家系统，神经网络控制．长沙：湖南大学出版社，1996

一种用于非线性控制的神经网络模糊自组织控制器

王耀南

（湖南大学电气工程系·长沙，410082）

摘要：本文提出一种神经网络自组织控制器，并应用于非线性跟踪控制中．为了加快神经模糊控制器的在线学习，文中给出了一种变斜率的最速梯度下降学习算法．仿真结果表明，该控制是有效的．

关键词：模糊逻辑；神经网络；模糊自组织控制；智能控制

本文作者简介

王耀南　1957年生．现为湖南大学电气工程系教授，博士生导师．主要从事计算机应用，人工智能，工业自动化，智能控制理论与应用，计算机视觉与图像处理方面研究．