한국퍼지 및 지능 시스템학회 논문지 1997, Vol. 7, No. 5. (1997) 21~27

A Fuzzy Model of Systems using a Neuro-fuzzy Network

Kwang Son Jeong* and Chong Kug Park**

**Dept. of Electronic Eng., Sang Ji Junior College*
***Dept. of Electronic Eng., Kyung Hee University*

ABSTRACT

Neuro-fuzzy network that combined advantages of the neural network in learning and fuzzy system in inferencing can be used to establish a system model in the design of a controller. In this paper, we presented the neuro-fuzzy system that can be able to generate a linguistic fuzzy model which results in a similar input/output response to the original system. The network was used to model a system. We tested the performance of the neuro-fuzzy network through computer simulations.

1. Introduction

Neural networks and fuzzy logic systems are two of the most important in the area of artificial intelligence. These techniques have been effectively applied to everything from voice and image recognition to toasters and automobile transmissions. Neural networks are best known for their learning capabilities. Fuzzy logic is a method of using human skills and thinking processes in a machine. In many applications, the training of a neural network requires millions of iterative calculations. Sometimes network can not adequately learn the desired function. Fuzzy logic, on the other hand, acquire their knowledge from an expert who encodes his knowledge in a series of IF/THEN rules. But, the problem arises when systems have many inputs and outputs. Obtaining a rule base for large systems is difficult, if not impossible. Prompted by the weaknesses inherent in the two technologies and their complementary strengths, researchers have looked at ways of combining neural network and fuzzy logic. Because a consensus on the best way to utilize their individual strengths and compensate for their individual shortcomings has not yet been established, many researchers have studied the neuro-fuzzy systems in many directions. Nauck *et.al* [1] researched the fusing of neural networks and fuzzy systems in an attempt to overcome the disadvantage expressed by other researchers using only one of the technologies. Jang[2] designed a self-learning fuzzy controller based on temporal back propagation. The current state of the system is compared to the desired state and the error is back propagated through the system to adjust individual fuzzy parameters. Blanco and Delgado[3] presented their work in the area of neural-fuzzy techniques. They suggested that a neural networks's strength lies in its ability to approximate a function from sample data. The parallel in fuzzy system applications would be the need to infer an output from a predefined rule base. Pedrycz [4] successfully trained a fuzzy-neuron neural network to control a system with two state variables and two control variables. His approach attaches linguistic terms in the place of numberic weights between the individual processing elements. Lee *et.al*[5] proposed a neuro-fuzzy identifier which had a cascaded structure of fuzzification, neural network and defuzzification. The neural network was used to learn the rules that associate the fuzzified inputs and defuzzified output. Errors due to the fuzzification and defuzzification processes was compensated by the neural network. They used the neuro-fuzzy identifier to model nonlinear dynamic systems. Horikawa *et.al* [6] implemented a fuzzy neural network that was able to automatically identify fuzzy rules and tune the membership functions. The system is composed of a multilayer network. Lin and Lee[7] as well as Yang and Stachowicz[8] implemented a self-learning fuzzy logic system embedded in a five layer network.

In this paper, we present the neuro-fuzzy system that can be able to generate a linguistic fuzzy model which results in a similar input/output response to the

original system. The network should be able to incorporate existing expert knowledge about the system as well to learn any additional features in the system behavior. We tested the performance of the neuro-fuzzy network with the different input and output membership functions through computer simulations.

2. Neural network and fuzzy system

2. 1 Neural network

The neural network that made use of this research is a multi-layers perceptron network. Trainig of this network is done using the backward error propagation algorithm. The general structure of this neural network is consisted of three layers-input, hidden and output layer. Then, the number of hidden nodes are selected appropriately from the results of the learning. Also the weights between layers are determined by the delta rule and Least Mean Square training rule. For LMS training the weights are modified according to the equation (1).

$$w_n = w_o + \beta \frac{x}{\left| x \right|^2} (y_{desired} - y_{actual}) \tag{1}$$

where, w is the weight vector for a processing element, x is the input vector, β is a programmer defined constant between 0 and 1, and y is the desired and actual outputs. The value of β determines how fast the weighting matrices converge to the point of the minimum least square error. The output of j-th neuron of the hidden layer is as follows.

$$I_j = f\left(\sum_i w_{ji} * x_j\right) \tag{2}$$

where, i is the input index, $f(\cdot)$ is the activation function. If the activation function select the sigmoidal function, then the output of j-th neuron from the equation (2) represents as following.

$$I_j = \frac{1}{1 + e^{-A_j}} \tag{3}$$

where, A_j is $f(\sum_i w_{ji} * x_i)$.

The delta rule made use of the adjusting weights of a back propagation neural network for minimizing the error. The error of the j-th neuron in the hidden layer is calculated as follows.

$$e_j = f'(I) * \sum_n (w_{nj} * E_n) \tag{4}$$

where, $f'(I)$ is the derivative of $f(I)$, $f'(I) \approx f(I)(1\text{-}f(I))$ and the derivative term contributes to stability and helps to prevent excessive blame being attached to the middle layer nodes, E_n is the error in the n-th nodes of the output layer. One of the ways to solve the local minimum problem is to include a momentum in the delta rule. Then the weights connected input, hidden, and output layer are modified according to the equation (5).

$$w_n = w_o + \beta \frac{Ex}{\left| x \right|^2} + \alpha (w_n - w_o)_{prov} \tag{5}$$

where, is the momentum constant.

2. 2 Fuzzy system

Fuzzy logic provides a platform for easily encoding human knowledge into the control of a system and fuzzy logic is a method of characterizing knowledge in terms of fuzzy sets and a rule base. A fuzzy system has one or more inputs that are fuzzified, a rule base that is evaluated according to the inputs and one or more outputs that are defuzzified into crisp values.

In this paper, fuzzy logic made use of the fuzzification for the input value and the defuzzification for the output value. A triangle selected to the type of membership function for the input and output variables. The defuzzification to obtain a crisp output variable made use of the Simplified Center Of Gravity method[9] such as the following equation (6).

$$u^* = \frac{\sum_{i=0}^{N} u_i^* w_i}{\sum_{i=0}^{N} w_i} \tag{6}$$

3. Neuro-Fuzzy System

It is very hard to model all the behaviors of a system unless it is a simple linear system. The main objective in most of neuro-fuzzy research is to model an existing system. The neuro-fuzzy network that

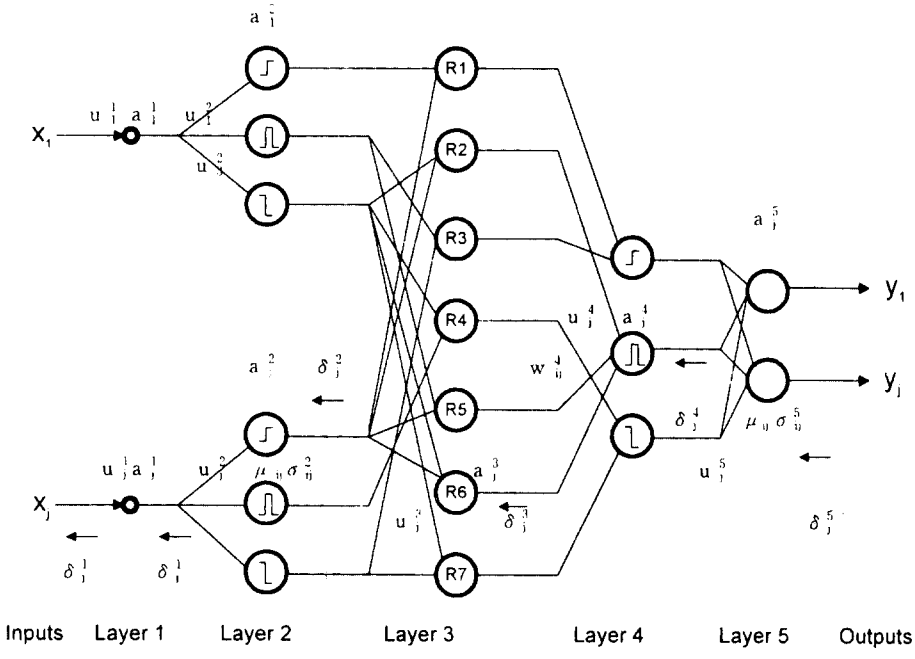


Fig. 1. Neuro-fuzzy network

selected to model the system is similar to the one by Yang and Stachowicz[8]. As shown in Fig. 1, the structure of the network are composed of five layers. Each layer is composed of several nodes. Two functions are used for every nodes. The first function is the integration function f_j^l which combines information from the previous layer l-1. The second layer is the activation function a_j^l which generates the output from node j in layer l. In the description of these functions, the following notations will be used :

u_j^l denotes the input to node j in layer l from the only node connected to it in the previous layer. u_{ij}^l denotes the input to node j in layer l from node i connected to it in the previous layer. $c^{((l-1)l)}(i, j)$ denotes the existence of connection between node i in layer l-1 and node j in layer l. Using the c matrices, the topology of the network is defined.

Layer 1 : The first layer is the input layer which transmit the inputs to their corresponding membership functions in the next layer.

$$u_j^1 = x_i \cdot f_j^1 = u_j^1 \cdot a_j^1 = f_j^1, \quad where\ i = j \tag{7}$$

where, x_i is the i-th input to the network.

Layer 2 : This is the input membership functions layer. Gaussian functions are used to implement the membership functions because they are continuous, differentiable and their means and spreads are parameterized as a part of the functions. Gaussian functions can also represent the common membership function shapes. The second layer is not fully connected with the first layer. The connections exist only between the input nodes in the first layer and their corresponding membership nodes in the second layer.

$$u_j^2 = a_i^1 \cdot f_j^2 = -\frac{(u_j^2 - \mu_{ij}^2)^2}{(\sigma_{ij}^2)^2} \cdot a_j^2 = e^{f_j^2}, \ c^{12}(i, j) = 1 \tag{8}$$

where, μ_{ij}^2 and σ_{ij}^2 are the mean and variance of the j-th Gaussian membership function for the i-th input.

Layer 3 : This layer represents the rules. Each node in layer 3 performs a fuzzy AND operation using the algebraic product. The rule association is done between two membership functions of different inputs. The output of each node in layer 3 represents the strength of firing the rule defined by that node. The synapses connecting layer 2 with layer 3 are either 1 or 0 to indicate the existence or the nonexistence of the corresponding rule.

$$u_{ij}^3 = a_i^2 \cdot f_j^3 = \prod_i u_{ij}^3 \cdot a_j^3 = f_j^3,\ c^{23}(i,j) = 1 \quad (9)$$

Layer 4: This layer performs the fuzzy OR operation between the rules that result in the same consequences. Initially this layer fully connected to layer 3 which means that any rule can result into any output membership function. As the network is trained, the synapses between layer 3 and layer 4 are enforced or decreased so that the rules are connected to the right output membership functions. In this scheme one rule can result into more than one output membership function which is the general form of fuzzy inferencing.

$$u_{ij}^4 = a_i^3 \cdot f_j^4 = \sum_i u_{ij}^4 \phi(w_{ij}^4),$$
$$a_j^4 = min(1,\ f_j^4),\ c^{34}(i,j) = 1 \quad (10)$$

where, w_{ij}^4 is the rule weight factor which indicates the importance of the rule i. $\varphi(w_{ij}^4)$ maps w_{ij}^4 from the range $[-\infty, \infty]$ to the range[0,1].

$$\phi(w_{ij}^4) = \frac{1}{1+e^{(-\alpha^4 w_{ij}^4)}} \quad (11)$$

where, α^4 is the steepness of the threshold function φ.

Layer 5: This layer uses Gaussian output membership functions and performs the defuzzification operation.

$$u_{ij}^5 = a_i^4 \cdot f_j^5 = \sum_i \sigma_{ij}^5 \mu_{ij}^5 u_{ij}^5 \cdot a_j^5 =$$
$$\frac{f_j^5}{\sum_i \sigma_{ij}^5 u_{ij}^5},\ c^{45}(i,j) = 1 \quad (12)$$

where, μ_{ij}^5 and σ_{ij}^5 are the mean and the variance of the i-th Gaussian membership function for the j-th output. The j-th output of the network is as following equation.

$$\hat{y}_j = a_j^5 \quad (13)$$

Learning uses a error back propagation algorithm to adjust the parameters of the membership functions and the rule weights. The network are trained with the input and output data. The learning rule uses the steepest descend algorithm[10] such that it minimizes the error E of the following equation.

$$E = \frac{1}{2}(y_j - \hat{y}_j)^2 \quad (14)$$

where, y_j is the desired output and $\hat{y}_j$ is the network output. For the steepest descent algorithm,

$$\triangle w \propto -\frac{\partial E}{\partial w} \quad (15)$$

$$w(t+1) = w(t) + \eta(-\frac{\partial E}{\partial w}) \quad (16)$$

where, w is the adjustable parameter in a node and η is the learning rate.

Using the delta rule and the equations for each layer the adaptation rule for each parameter can be computed. In the description of the adaptation rule, δ_i^l denotes the delta term for node i in layer l back propagated from the nodes connected to it in the next layer.

Layer 5:

$$\delta_j^5 = y_j - \hat{y}_j \quad (17)$$

$$\triangle \mu_{ij}^5 = \eta^5 \delta_j^5 \frac{\sigma_{ij}^5 u_{ij}^5}{\sum_k \sigma_{kj}^5 u_{kj}^5},$$
$$c^{45}(i,j) = 1,\ c^{45}(k,j) = 1 \quad (18)$$

$$\triangle \sigma_{ij}^5 =$$
$$\eta^5 \delta_j^5 \frac{\mu_{ij}^5 u_{ij}^5 (\sum_k \sigma_{kj}^5 u_{kj}^5) - u_{ij}^5 (\sum_k \mu_{kj}^5 \sigma_{kj}^5 u_{kj}^5)}{(\sum_k \sigma_{kj}^5 u_{kj}^5)^2} \quad (19)$$

Layer 4:

$$\delta_i^4 = \sum_j (\delta_j^5 \frac{\mu_{ij}^5 \sigma_{ij}^5 (\sum_k \sigma_{kj}^5 u_{kj}^5) - \sigma_{ij}^5 (\sum_k \mu_{kj}^5 \sigma_{kj}^5 u_{kj}^5)}{(\sum_k \sigma_{kj}^5 u_{kj}^5)^2}) \quad (20)$$

$$\triangle w_{ij}^4 = \eta^4 \delta_j^4 u_{ij}^4 \alpha^4 \phi(w_{ij}^4)(1 - \phi(w_{ij}^4)),$$
$$c^{34}(i,j) = 1 \quad (21)$$

Layer 3:

$$\delta_i^3 = \sum_j (\delta_j^4 \phi(w_{ij}^4)),\ c^{34}(i,j) = 1 \quad (22)$$

Layer 2:

$$\delta_i^2 = \sum_j \delta_j^3 (\prod_k u_{kj}^3),\ c^{23}(i,j) = 1,$$
$$c^{23}(k,j) = 1 \quad (23)$$

$$\triangle \mu_{ij}^2 = \eta^2 \delta_j^2 a_j^2 \frac{2(u_{ij}^2 - \mu_{ij}^2)}{(\sigma_{ij}^2)^2},$$
$$c^{12}(i,j) = 1 \quad (24)$$

$$\triangle \sigma_{ij}^2 = \eta^2 \delta_j^2 a_j^2 \frac{2(u_{ij}^2 - \mu_{ij}^2)^2}{(\sigma_{ij}^2)^3},$$

$$c^{12}(i, j) = 1 \tag{25}$$

Different learning rates were used to define the relative rates of training among the input membership functions, the rules and the output membership functions. Momentum was also used to stabilize the convergence against oscillation. The parameters updating rule adding to the momentum was as the following equation[10].

$$\triangle w(k) = \eta(-\frac{\partial E}{\partial w}) + \alpha \triangle w(k-1) \tag{26}$$

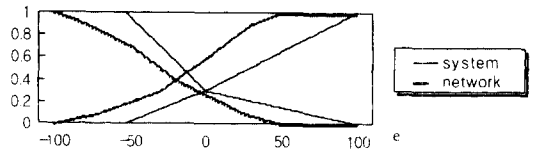
where, α is momentum. After the network was trained, the rules with weights below a certain threshold were eliminated and training continued to tune the membership functions with the new set of rules. This procedure was repeated until the number of the rules did not changed. The network was used to model different systems including a predefined known fuzzy test model and was able to reduce the rules and membership a great deal.

4. Computer simulation and results.

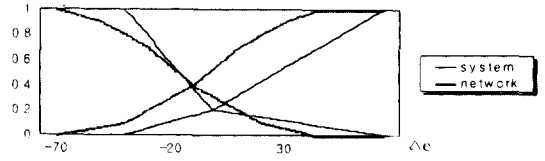
We performed computer simulation to test the performance of neuro-fuzzy network with the different input and output membership funtions. A fuzzy model was constructed to test the performance of the network to converge to the original knowledge base using the input and output data. The model had two inputs and one output. Every input and output had two membership functions. The rules were defined such that the system had the step response of the second order system with damping ratio ζ=0.2. In the first modeling test, the input and output of the network had two membership functions as the known system. Fig. 2 and 3 show the initial input and output membership functions, Fig. 4 and 5 show the final membership functions identified by the supervised learning after training. In Figures, the thin lines are the system membership functions and the thick lines are the network membership functions. The network was able to converge to a fuzzy knowledge base very similar to the oringinal system's knowledge base. Also, the network reduced the number of the rules from eight possible rules to only the four rules used in the system's model. These rules are as following.

IF (in1=MF1) and (in2=MF1) THEN (out=MF2)

IF (in1=MF1) and (in2=MF2) THEN (out=MF2)



(a) Membership functions of input 1



(b) Membership functions of input 2

Fig. 2. Initial input membership functions

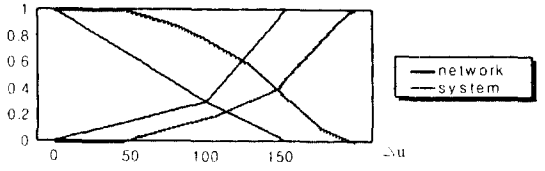
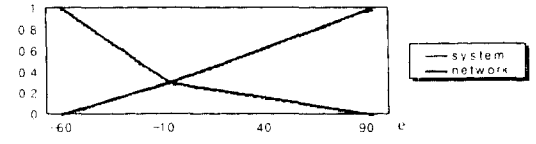
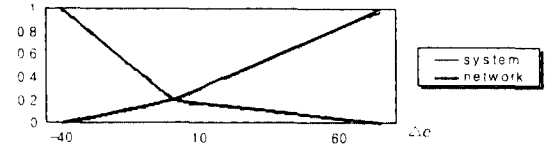


Fig. 3. Initial output membership functions



(a) Membership functions of input 1



(b) Membership functions of input 2

Fig. 4. Final input membership functions

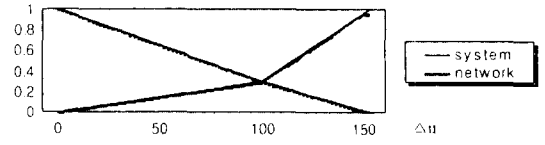


Fig. 5. Final output membership functions

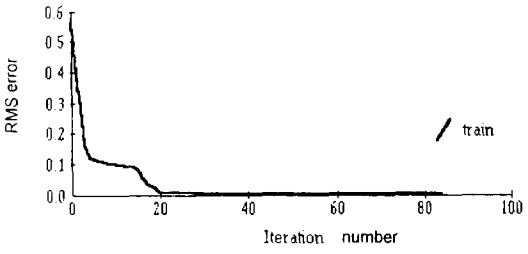
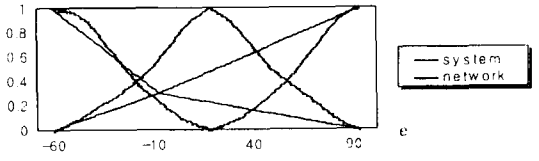


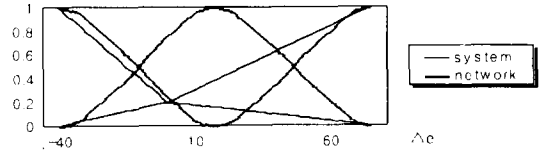
Fig. 6. Learning error of the network

IF (in1=MF2) and (in2=MF1) THEN (out=MF1)
IF (in1=MF2) and (in2=MF2) THEN (out=MF1)

Fig. 6 shows the learning error of the network. As shown Fig. 6, the learning error was to be near zero after the first 25 epochs and the network converged to the original system model approximately. The second modeling test was performed with the same systems model but the network had three membership functions for every input. Fig. 7 and 8 show the initial input and output membership functions, Fig. 9 and 10 show the final membership functions after training. Fig. 11 shows the learning error of the network. As shown Fig. 11, the learning error was to be near zero after the first 50 epochs and the network converged to the original system model approximately. Therefore, There was no noticeable difference between the response of the network and system. Also, the network reduced the number of the rules from eighteen possible rules to only the four rules used in the system's model.



(a) Membership functions of input 1



(b) Membership functions of input 2

Fig. 7. Initial input membership functions

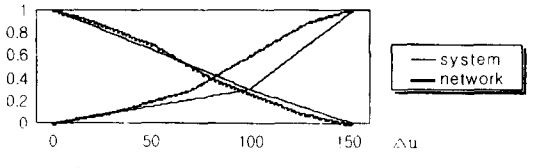
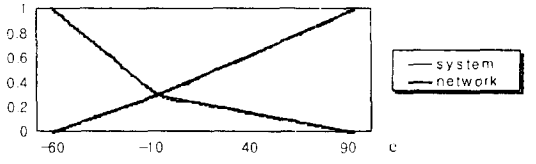
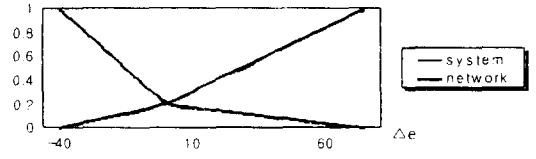


Fig. 8. Initial output membership functions



(a) Membership functions of input 1



(b) Membership functions of input 2

Fig. 9. Final input membership functions

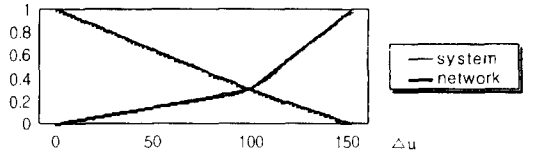


Fig. 10. Final output membership functions

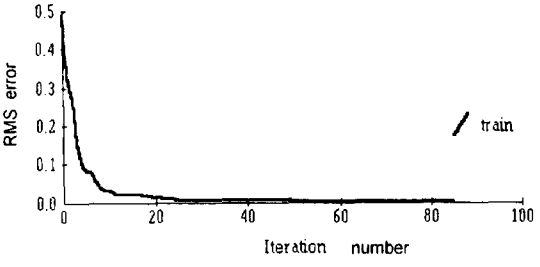


Fig. 11. Learning error of the network

5. Conclusion

We presented the neuro-fuzzy system that can be able to generate a linguistic fuzzy model which results in a similar input/output response to the original system. The architecture allows knowledge incorporation as well as rules and membership functions

tuning. Also, we showed the performance of the neuro-fuzzy network for the different input and output membership functions through computer simulations. From the results, we knew that the response of the network was similar to the response of the system.

References

[1] D. Nauck, F. Klawonn and R. Kruse, "Combining Neural Networks and Fuzzy Controllers," *Fuzzy Logic in Artificial Intelligence*(FLAI93), pp. 35-46, 1993.

[2] J. Jang, "Self-Learing Fuzzy Controllers Based on Temporal Back Propagation," *IEEE Trans. on Systems, Man and Cybernetics*, 1992.

[3] A. Blanco and M. Delgado, "A Direct Fuzzy Inference Procedure By Neural Networks," *Fuzzy Sets and Systems*, pp. 133-141, 1993.

[4] W. Pedrycz, "Fuzzy sets and Neurocomputations : Knowledge representation and pro-cessing in intelligent controllers," fifth International Symposium on Intelligent Control, pp. 626-630, 1990.

[5] M. Lee, S. Lee, and C. H. Park, "A new neuro-fuzzy identification model of non-linear dynamic systems," International Journal of Approximate Reasoning, vol. 10, pp. 29-44, 1994.

[6] S. Horikawa, T. Furuhashi and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm," *IEEE Trans.* on Neural Networks, vol. 3, pp. 801-806, 1992.

[7] C. Lin and C. S. G. Lee, "Reinforcement structure/parameter learning for neural network based fuzzy logic control systems," *IEEE Trans.* on Fuzzy Systems vol. 2, pp. 46-63, 1994.

[8] C. Yang and M. S. Stachowicz, "An algorithm for designing a self-learning fuzzy logic system," in Proc. of the 1994 International Fuzzy Systems and Intelligent Control Conference, pp. 258-267, 1994.

[9] S. Chae, Y. S. Oh, "Fuzzy theorem and control," Cheongmoongak Press, 1995.

[10] D. S. Kim, "Neural network : theoy and application I," Hi-tech information Press, 1994.

정광손(Kwang-Son Jeong) 정회원

1986년: 한양대학교 전자공학과 (공학사)
1988년: 경희대학교 전자공학과 (공학석사)
1994년~현재: 경희대학교 전자공학과 박사과정
1993년~현재: 상지대학교 병설전문대학 전자과 조교수

주관심분야: 뉴로-퍼지제어, 로보틱스, 적응제어

박종국(Chong-Kug Park) 정회원

제 6권 3호 참조
현재: 경희대학교 전자공학과 교수