# 신경회로망을 이용한 퍼지제어기 설계 알고리즘에 관한 연구

채 명 기[1], 李 尙 培[2]

# The study on the Algorithm for Design of Fuzzy Logic Controller using Neural Network

Myong Gi Chae , Sang Bae Lee

## Abstract

In this paper, a general neural-network-based connectionist model, called Fuzzy Neural Network(FNN), is proposed for the realization of a fuzzy logic control system. The proposed FNN is a feedforward multi-layered network which integrates the basic elements and functions of a traditional fuzzy logic controller into a connectionist structure which has distributed learning abilities. Such FNN can be constructed from training examples by learning rule, and the connectionist structure can be trained to develop fuzzy logic rules and find optimal input/output membership functions. Computer simulation examples will be presented to illustrate the performance and applicability of the proposed FNN, and their associated learning algorithms.

1) 한국해양대학교 대학원 전자통신공학과 석사과정 전자진산공학 전공
2) 한국해양대학교 전자통신공학과 교수

## 1. 서 론

많은 제어분야에서 비선형성이 강하고, 명확하지 않은 플랜트를 효과적으로 제어하기 위해서 퍼지 이론이 이러한 분야에서 효과적으로 응용되고 있다. 그러나 퍼지제어에 필요한 적절한 소속함수와 규칙을 구성하는 것은 전문가의 전적인 도움이 필요하거나 실제로 구성하기 어려운 경우가 많다. 그러므로 신경회로망의 학습기능을 통해 적합한 퍼지 소속함수와 규칙을 학습시킬 수 있다. 또한 본 논문에서 제시한 퍼지뉴럴 네트워크는 학습기능을 통해서 최적의 규칙과 입, 출력 소속함수를 결정해 주므로 전통적인 퍼지 제어 시스템에서의 rule-matching에 소비되는 시간을 줄일 수 있다. 또한 퍼지 제어기의 함수 수행을 위한 VLSI 구현이 쉬운 연결 구조를 가지고 있다.

본 논문에서는 신경회로망을 기반으로 한 퍼지 제어 시스템을 개발하는데 있어서의 on-line supervised learning algorithm을 제안한다. 여기서 제안된 학습 알고리즘은 실시간에서 structure와 parameter를 동시에 적합하게 학습할 수 있다.

## 2. 퍼지뉴럴 네트워크(Fuzzy Neural Network ; FNN)의 구조

여기서 제시된 퍼지뉴럴 네트워크(Fuzzy Neural Network ; FNN)는 분산학습 능력을 가진 신경회로망의 구조에 퍼지제어기(Fuzzy Logic Controller ; FLC)의 기본요소와 함수를 통합시킨 다층의 신경회로망의 구조로 구성하였다.

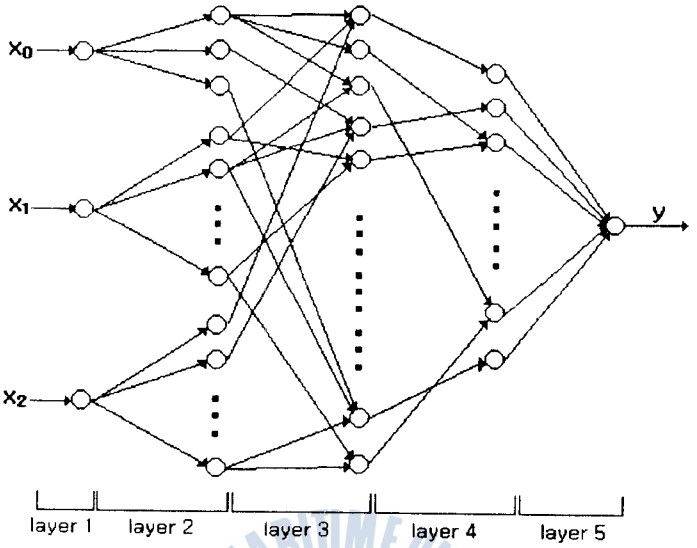
<그림 1>은 여기서 제안되어진 퍼지뉴럴 네트워크의 구조를 나타낸다.

이 시스템은 총 5개의 층으로 구성되어 있다.

<그림2>는 FNN의 블록 다이어그램을 나타낸다.

각 층들의 기능을 간략하게 살펴보면,

layer 1에서의 노드는 입력노드로서 입력 언어 변수를 표현한다. layer 2와 4의 노드는 term 노드로서 각각의 언어변수의 term을 표현하기 위한 소속함수로 작용한다. layer 3에서의 노드는 rule 노드로서 하나의 fuzzy rule을 표현한다. 여기서 모든 노드는 fuzzy rule base를 형성한다. layer 5는 출력 layer로서 각 출력 언어변수를 표현한다. 이 층에서는 원하는 출력으로서 training data가




< 그림 1 > 제안된 FNN의 구조

네트워크로 입력되고, 실제 출력값이 네트워크에서 출력된다. layer 3과 4의 링크는 추론기관의 역할을 한다. 즉, 여기서 신경회로망의 구조를 가지므로 rule matching process를 피하게 된다. layer 3의 링크는 rule node의 조건부를 정의하고, layer 4의 링크는 rule node의 결과를 정의한다. layer 2와 5의 링크는 언어노드와 그에 상응하는 term 노드 사이에는 모두 연결되어 있다.

layer 1에서의 노드는 단지 입력값을 다음층으로 바로 전달한다. 즉,

$$f = u_i^1 \quad \text{이고} \quad a = f \tag{1}$$

layer 2에서는 종형함수에 대해서

$$f = M_{x_i}{}^{j}(m_{ij}, \sigma_{ij}) = -\frac{(u_i^2 - m_{ij})^2}{\sigma_{ij}{}^2} \quad \text{그리고} \quad a = e^f \tag{2}$$

$m_{ij}$는 i번째 입력언어변수 $x_i$의 j번째 term의 종형함수의 중심값(center)이고, $\sigma_{ij}$는 위 함수의 폭(width)을 나타낸다.

layer 3에서의 link는 fuzzy logic rule의 precondition matching을 수행하기 위해 사용된다. 그러므로 fuzzy AND operation을 수행한다.

$$f = \min(u_1^3, u_2^3, \cdots\cdots, u_p^3) \quad \text{그리고} \quad a = f \tag{3}$$

layer 4에서의 link는 같은 결과를 갖는 fired rule을 통합하기 위해 fuzzy OR operation을 수행한다.

$$f = \sum_{i=1}^{p} u_i^4 \quad \text{그리고} \quad a = \min(1, f) \tag{4}$$

layer 5에서, 다음 함수는 비퍼지화방법인 무게중심법을 실행하기 위해 사용될 수 있다.

$$f = \Sigma w_{ij}^5 u_{ij}^5 = \Sigma(m_{ij}\sigma_{ij})u_i^5 \quad \text{그리고} \quad a = \frac{f}{\Sigma\sigma_{ij}u_i^5} \tag{5}$$

## 3. 온라인 학습 알고리즘

### 3-1. 퍼지집합의 유사도(Similarity Measure)

이것은 두 퍼지집합이 같은가 또는 얼마나 유사한가의 정도를 나타내는 척도로서 퍼지뉴럴 네트워크 구조의 학습에 사용된다.

$$M(A) = \sum_{u \in U} \mu_A(u) \tag{6}$$

종형함수를 사용하여 유사도 계산을 일반화시키면,

$$M(A \cap B) = \frac{1}{2}\frac{h^2(m_2 - m_1 + \sqrt{\pi}(\sigma_1 + \sigma_2))}{\sqrt{\pi}(\sigma_1 + \sigma_2)} + \frac{1}{2}\frac{h^2(m_2 - m_1 + \sqrt{\pi}(\sigma_1 - \sigma_2))}{\sqrt{\pi}(\sigma_2 - \sigma_1)}$$

$$+ \frac{1}{2}\frac{h^2(m_2 - m_1 - \sqrt{\pi}(\sigma_1 - \sigma_2))}{\sqrt{\pi}(\sigma_1 - \sigma_2)} \tag{7}$$

여기서, h(x) = max{0, x}를 나타낸다.

근사화시킨 퍼지집합(fuzzy set)의 유사도는,

$$E(A, B) = \frac{M(A \cap B)}{M(A \cup B)} = \frac{M(A \cap B)}{M(A) + M(B) - M(A \cap B)}$$

$$= \frac{M(A \cap B)}{\sigma_1\sqrt{\pi} + \sigma_2\sqrt{\pi} - M(A \cap B)} \tag{8}$$

## 3-2. 학습 알고리즘(Learning Algorithm)

본 논문에서 제안하는 학습 알고리즘의 특징은 네트워크의 파라미터와 구조를 동시에 학습시킬 수 있다는 것이다. 네트워크 구조의 학습에서는 퍼지 규칙의 후건부에 해당하는 rule과 출력소속함수의 노드들의 연결상태를 적합하게 학습시켜 주고, 앞 절에서 유도하였던 유사도(similarity measure)의 개념이 사용된다. 그리고 네트워크의 파라미터 학습에서는 입,출력소속함수의 파라미터를 적합하게 학습시켜 주고, 일반적인 신경회로망의 오류 역전파 학습방법이 사용된다.

learning scheme을 초기화하기 위해, 원하는 input fuzzy partition(즉, 입력언어변수의 term set의 크기)와 초기의 output fuzzy partition을 결정하고, 네트워크를 학습하기 전에 초기형태를 구성한다. 학습을 행하는 동안 일부 연결상태가 변화된다. 결국 학습후에 일부 link가 삭제 또는 통합되어 최종적인 네트워크의 구조가 결정된다.

초기 형태에는 $\Pi|T(x_i)|$개의 rule node를 갖는다. $\Pi|T(x_i)|$는 $x_i$의 term의 수, 즉 입력상태 언어변수 $x_i$의 fuzzy partition의 수를 의미한다. 각 입력변수의 term set중 한 개의 term만이 rule node의 입력이 될 수 있고, rule node와 출력언어변수 사이에는 한 개의 link만이 존재해야 한다.

Training loop 로 들어가기 전에 structure와 parameter의 초기화가 필요하다. 초기화 절차가 끝나면 training loop 로 들어간다. 기본적으로 각 layer의 node output의 error를 계산하기 위해 오류 역전파 방법을 사용한다. 본 알고리즘의 목적은 error function을 최소화하는 것이다.

$$E = \frac{1}{2}[y(t) - \hat{y}(t)]^2 \tag{9}$$

여기서, y(t)는 원하는 출력, $\hat{y}(t)$는 퍼지뉴럴 네트워크의 실제 출력을 나타낸다. 종형함수를 사용할 경우 조절가능한 parameter는 종형함수의 중심값(center) $m_i$와 폭(width) $\sigma_i$ 이다. 그러면 네트워크의 학습과정을 살펴보겠다.
Layer 5 에서는,

$$\frac{\partial E}{\partial m_i} = \frac{\partial E}{\partial a}\frac{\partial a}{\partial f}\frac{\partial f}{\partial m_i} = -[y(t) - \hat{y}(t)]\frac{\sigma_i u_i}{\sum \sigma_i u_i} \tag{10}$$

그러므로, center parameter는 다음과 같이 갱신된다.

$$m_i(t+1) = m_i(t) + \eta[y(t) - \hat{y}(t)]\frac{\sigma_i u_i}{\sum \sigma_i u_i} \tag{11}$$

역시 같은 방법으로,

$$\frac{\partial E}{\partial \sigma_i} = \frac{\partial E}{\partial a}\frac{\partial a}{\partial f}\frac{\partial f}{\partial \sigma_i} \tag{12}$$

그러므로,

$$\sigma_i(t+1) = \sigma_i(t) + \eta[y(t) - \hat{y}(t)]\frac{m_i u_i(\sum \sigma_i u_i) - (\sum m_i \sigma_i u_i)u_i}{(\sum \sigma_i u_i)^2} \tag{13}$$

앞으로 전파되어지는 오차는,

$$\delta^5 = -\frac{\partial E}{\partial a^5} = y(t) - \hat{y}(t) \tag{14}$$

< Similarity Measure of Fuzzy Set >

먼저 갱신된 출력 소속함수의 중심값(center)과 폭(width)의 parameter에 의해서 현재구조를 변화시킬 것인가를 결정해야 한다. 파라미터의 갱신은 다음과 같이 이루어진다.

$$m_{i-new} = m_i(t) + \Delta m_i(t) \tag{15}$$
$$\sigma_{i-new} = \sigma_i(t) + \Delta \sigma_i(t)$$

출력 언어변수의 현재 소속함수로부터 유사도에 의해 예상되는 소속함수와 가장 유사한 소속 함수를 찾을 수 있다.

$$\deg ree(i,\ t) = E[M(m_{i-new},\ \sigma_{i-new}),\ M(m_{i-closest},\ \sigma_{i-closest})]$$
$$= \max_{i \le j \le k} E[M(m_{i-new}, \sigma_{i-new}),\ M(m_j, \sigma_j)] \tag{16}$$

k는 출력언어변수 y(t)의 fuzzy partition의 크기이다. 갱신된 소속함수 $M(m_{i\text{-new}}, \sigma_{i\text{-new}})$에 가장 유사한 소속함수 $M(m_{i\text{-closest}}, \sigma_{i\text{-closest}})$을 찾은 후에는 다음과 같은 조절을 한다. 만일 $M(m_{i\text{-closest}}, \sigma_{i\text{-closest}})$가 $M(m_i, \sigma_i)$가 아니라면, structure 학습 과정을 수행한다. 그러나, $M(m_{i\text{-closest}}, \sigma_{i\text{-closest}})$가 $M(m_i, \sigma_i)$라면, structure 학습을 하지 않고, layer 5의 파라미터를 다음과 같이 조절한다.

$$m_i(t+1) = m_{i-new}\ ,\quad \sigma_i(t+1) = \sigma_{i-new}$$

< Structure Learning >

이 process로 들어온 것은 layer 4의 i번째 term node가 fuzzy logic rule의 결과로서 부적합하게 할당됨을 의미한다. fuzzy logic rule의 더 정확한 결과는 i-closest node가 되어야 한다. 바뀌어야 하는 rule의 결과를 찾기 위해서 firing strength threshold β를 설정한다. 단지 rule의 firing strength가 threshold보다 클 때만 진정한 firing rule로 간주한다. 여기서 layer 4의 term node $M(m_i, \sigma_i)$는 layer 3의 rule node(1~l)로부터 입력을 갖고, 그 firing strength는 $a_i^3(i=1,2,....,l)$가 된다. 즉, $a_i^3 \geq \beta$ 이면, rule node의 결과를 $M(m_i, \sigma_i)$에서 $M(m_{i\text{-closest}}, \sigma_{i\ closest})$로 바꾼다.

Layer 4에서는 error signal만 계산되고 전파되어진다.

$$\delta_i^4 = [y(t) - \hat{y}(t)] \frac{m_i u_i (\sum \sigma_i u_i) - (\sum m_i \sigma_i u_i) u_i}{(\sum \sigma_i u_i)^2} \tag{17}$$

Layer 3 에서도 error signal만 계산된다.

$$\delta_i^3 = -\frac{\partial E}{\partial a_i} = \delta_i^4 \tag{18}$$

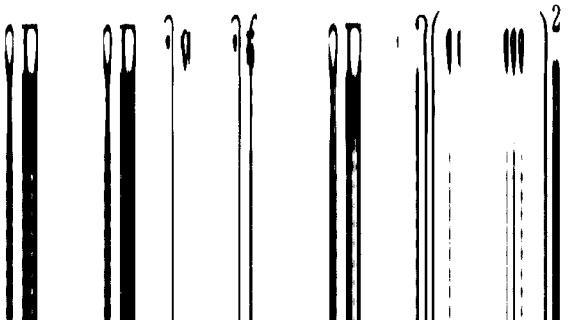
Layer 2 에서는,

$$\frac{\partial E}{\partial m_{ij}} = \frac{\partial E}{\partial a_i} \frac{\partial a_i}{\partial m_{ij}} = \frac{\partial E}{\partial a_i} e^{f_i} \frac{2(u_i - m_{ij})}{\sigma_{ij}^2} \tag{19}$$

그러므로 $m_{ij}$의 adaptive rule은,

$$m_{ij}(t+1) = m_{ij}(t) - \eta \frac{\partial E}{\partial a_i} e^{f_i} \frac{2(u_i - m_{ij})}{\sigma_{ij}^2} \tag{20}$$

같은 방법으로,



< Structure Learning >

이 process로 들어온 것은 layer 4의 i번째 term node가 fuzzy logic rule의 결과로서 부적합하게 할당됨을 의미한다. fuzzy logic rule의 더 정확한 결과는 i-closest node가 되어야 한다. 바뀌어야 하는 rule의 결과를 찾기 위해서 firing strength threshold β를 설정한다. 단지 rule의 firing strength가 threshold보다 클 때만 진정한 firing rule로 간주한다. 여기서 layer 4의 term node $M(m_i, \sigma_i)$는 layer 3의 rule node(1~l)로부터 입력을 갖고, 그 firing strength는 $a_i^3(i=1,2,....,l)$가 된다. 즉, $a_i^3 \geq \beta$ 이면, rule node의 결과를 $M(m_i, \sigma_i)$에서 $M(m_{i\text{-closest}}, \sigma_{i\ closest})$로 바꾼다.

Layer 4에서는 error signal만 계산되고 전파되어진다.

$$\delta_i^4 = [y(t) - \hat{y}(t)] \frac{m_i u_i (\sum \sigma_i u_i) - (\sum m_i \sigma_i u_i) u_i}{(\sum \sigma_i u_i)^2} \tag{17}$$

Layer 3 에서도 error signal만 계산된다.

$$\delta_i^3 = -\frac{\partial E}{\partial a_i} = \delta_i^4 \tag{18}$$

Layer 2 에서는,

$$\frac{\partial E}{\partial m_{ij}} = \frac{\partial E}{\partial a_i}\frac{\partial a_i}{\partial m_{ij}} = \frac{\partial E}{\partial a_i} e^{f_i} \frac{2(u_i - m_{ij})}{\sigma_{ij}^2} \tag{19}$$

그러므로 $m_{ij}$의 adaptive rule은,

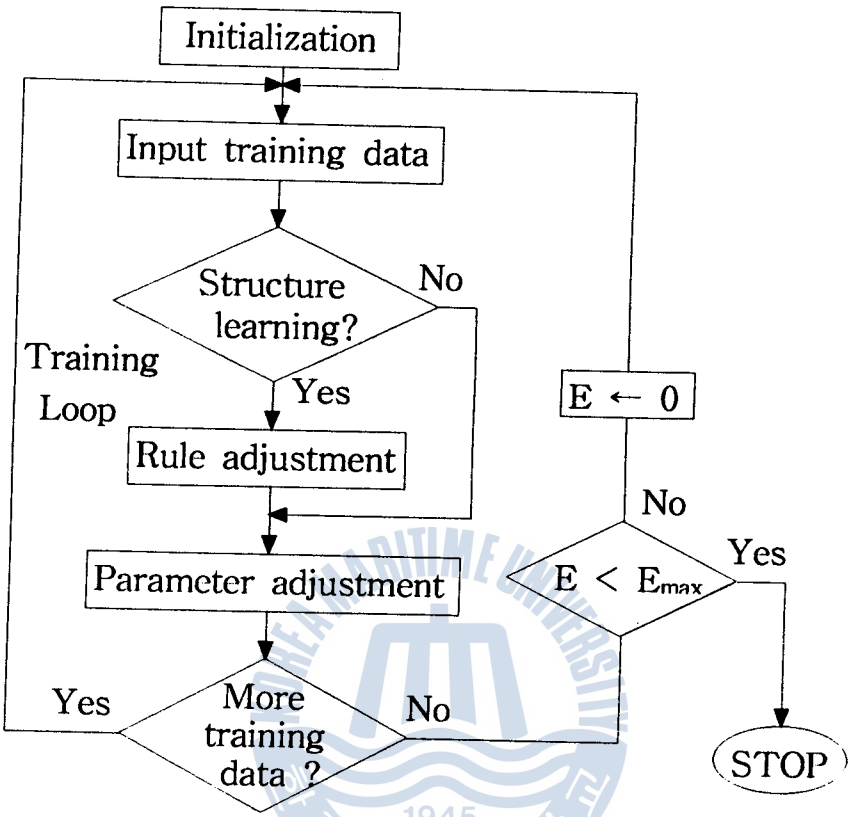
$$m_{ij}(t+1) = m_{ij}(t) - \eta \frac{\partial E}{\partial a_i} e^{f_i} \frac{2(u_i - m_{ij})}{\sigma_{ij}^2} \tag{20}$$

같은 방법으로,

$$\frac{\partial E}{\partial \sigma_{ij}} = \frac{\partial E}{\partial a_i}\frac{\partial a_i}{\partial f_i}\frac{\partial f_i}{\partial \sigma_{ij}} = \frac{\partial E}{\partial a_i} e^{f_i} \frac{2(u_i - m_{ij})^2}{\sigma_{ij}^3} \tag{21}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) - \eta \frac{\partial E}{\partial a_i} e^{f_i} \frac{2(u_i - m_{ij})^2}{\sigma_{ij}^3} \tag{22}$$

학습과정에 대한 전체적인 흐름도를 <그림 2>에 나타내었다.

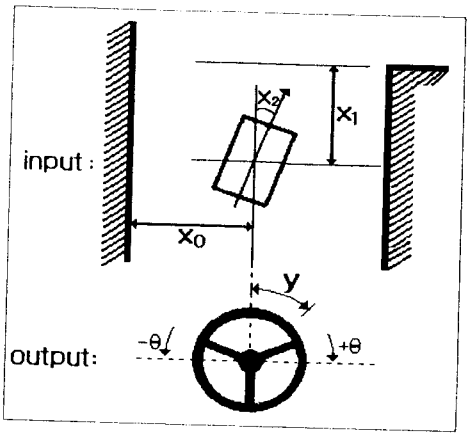



< 그림 2 > 학습의 전체 흐름도

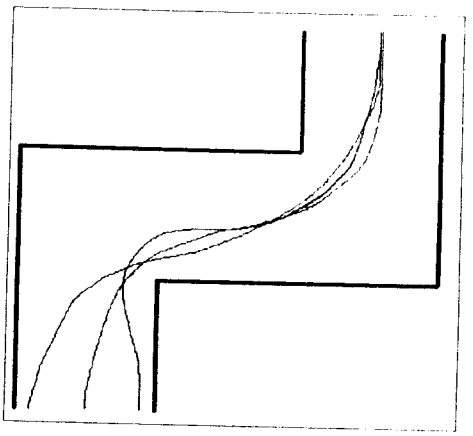
## 4. Fuzzy Car Control에 대한 시뮬레이션

본 장에서는 제시한 네트워크의 구조와 알고리즘을 이용하여 Fuzzy Car의 제어에 적용해 보고자 한다. 시뮬레이션에서는, 제안된 퍼지뉴럴 네트워크가 과거 운전 경험의 data set를 사용하여 잘못된 퍼지 소속함수와 rule을 최적으로 학습하여 퍼지제어의 수행을 보일 것이다. 여기서 도로의 간격과 차의 속도는 일정하다고 가정하였다.

<그림 3>은 시뮬레이션에 사용된 fuzzy car의 입,출력 상태변수를 나타낸다. 입력 언어 변수 $x_0$ 는 track의 side와 car와의 거리, $x_1$ 은 corner와 차와의 거리, $x_2$ 는 현재의 steering angle을 나타낸다. 그리고 출력 언어 변수 y는 다음 steering angle의 변화량을 나타낸다. 입,출력 fuzzy partition의 수는 $x_0$, $x_1$, $x_2$, y 각각 3, 5, 5, 9개로 설정하였다. 그리고 초기의 fuzzy logic rule (layer 3과 4의

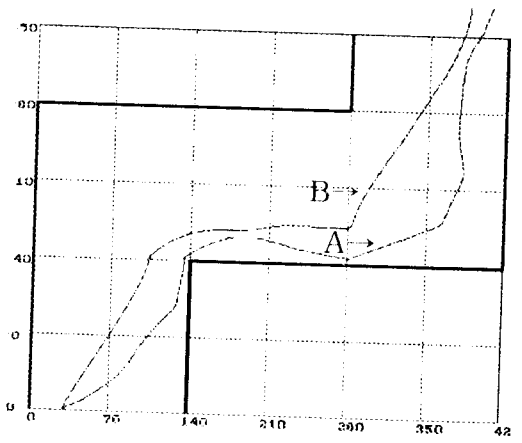



< 그림 3 > 시뮬레이션에 사용된 Fuzzy Car의 입,출력 상태변수

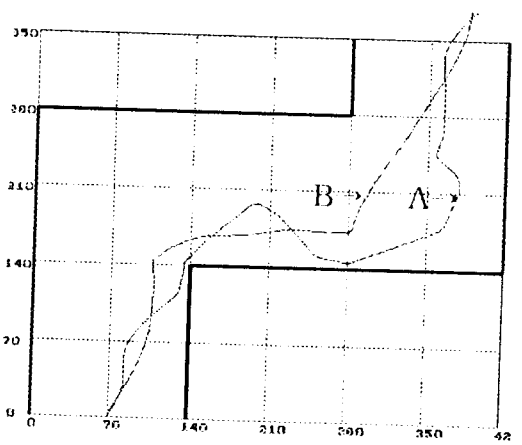


< 그림 4 > 시뮬레이션에 사용된 학습 데이타

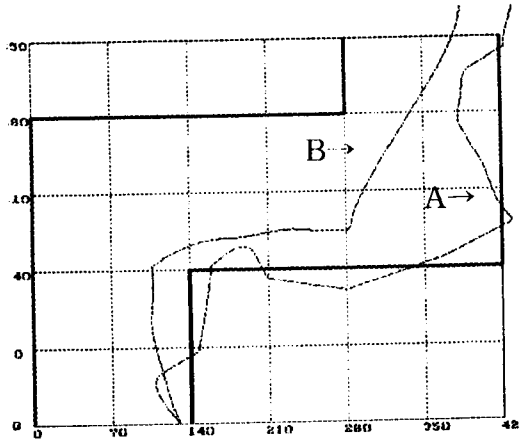
연결)은 random하게 설정하였다. FNN 학습은 <그림 4>와 같은 세 가지 정로를 따르는 150개의 학습데이타를 사용하였다. <그림 4>와 같은 우수한 학습데이타를 사용하여 학습한 후 fuzzy car의 시운전을 위하여 몇가지의 예로서 컴퓨터 시뮬레이션을 행하였다. 그 실험결과는 <그림 5, 6, 7>과 같으며 초기에 임의의 rule과 소속함수를 설정해 주었을 때, A와 B는 각각 학습전과 학습후의 운전궤적을 나타낸다.




< 그림 5 > 시뮬레이션 예 (1)




< 그림 6 > 시뮬레이션 예 (2)




< 그림 7 > 시뮬레이션 예 (3)

## 5. 결 론

제안한 FNN에서 우수한 학습 데이터를 이용하여 학습한 결과 만족할 만한 제어규칙과 소속함수를 얻을 수 있었다. 초기에 임의로 규칙을 정해 주었기 때문에 제대로 제어할 수 없었던 것을 학습한 후에는 제어할 수 있었지만 완벽한 숙련 운전자의 효과를 내지 못해서 좀 더 보완이 필요하다 하겠다. 현재의 FNN에서 입,출력 소속함수의 partition의 수는 정해져 있으므로 신경회로망의 구조를 동적으로 변화시킨다면, 즉 rule과 각 변수의 partition의 수를 동적으로 변화시킨다면 매우 좋은 결과가 기대되며 앞으로 계속 연구하여 결과를 제시할 계획이다. 여기서 제시한 FNN의 구조는 여러 가지 상황의 에러에도 대처할 수 있는 기능을 가진 것으로 다른 여러 가지의 플랜트의 제어에도 효과적으로 응용할 수 있을 것으로 사료된다.

# 참 고 문 헌

[1] 채명기, 이상배 "신경회로망을 이용한 퍼지제어기 설계 알고리즘에 관한 연구" 1996년도 한국퍼지 및 지능시스템학회 추계종합학술대회 논문집 vol. 6 No. 2, pp. 243-248

[2] Jacek M. Zurada, Introduction to Artificial Neural Systems., West Publishing Company, pp. 163-235, 1992.

[3] B. Kosko, Neural Networks and Fuzzy Systems, Prentice Hall, Inc., New Jersey, 1992

[4] M. Sugeno and M. Nishida, "Fuzzy control of model car," Fuzzy Sets Syst., Vol.16, 1985, pp.103-113

[5] Ronald R. Yager, "Implementing fuzzy logic controllers using a neural network framework" Fuzzy Sets Syst., Vol.48, 1992, pp.53-64

[6] Li-Xin Wang and Jerry M. Mendel, "Generating Fuzzy Rules by Learning from Examples" IEEE Trans. Syst. Man Cybern., Vol.22, No.6, 1992, pp.1414-1427

[7] Clarence W. de Silva, Intelligent Control, Fuzzy Logic Application, CRC press, 1995