論文 한국정밀공학회지 제14권 제4호 (1997년 4월)
Journal of the Korean Society of Precision Engineering Vol. 14. No. 4. April. 1997.

이 논문은 1995학년도 영남대학교 학술연구조성비에 의한 것임.

뉴럴 네트웍 모델링에서 에러를 최소화하기 위한 퍼지분할법

정 병 묵*

Fuzzy Division Method to Minimize the Modeling Error in Neural Network

Byeong-Mook Chung*

ABSTRACT

Multi-layer neural networks with error back-propagation algorithm have a great potential for identifying nonlinear systems with unknown characteristics. However, because they have a demerit that the speed of convergence is too slow, various methods for improving the training characteristics of backpropagation networks have been proposed. In this paper, a fuzzy division method is proposed to improve the convergence speed, which can find out an effective fuzzy division by the tuning of membership function and independently train each neural network after dividing the network model into several parts. In the simulations, the proposed method showed that the optimal fuzzy partitions could be found from the arbitrary initial ones and that the convergence speed was faster than the traditional method without the fuzzy division.

Key Words : Neural Network Modeling (뉴럴네트웍모델링), Fuzzy Division (퍼지분할), Fuzzy Learning (퍼지학습), Tuning of membership function (멤버쉽함수의 조정)

1. 서 론

일반적으로 기존의 제어 알고리즘은 프로세서의 수학적 모델링을 바탕으로 발달하였다. 그러므로 복잡한 동특성을 갖는 시스템의 전달함수, 상태방정식 형태의 정확한 수학적 모델링이 어렵거나 시스템의 파라미터의 변동, 또는 부하 외란이 존재하는 변하는 환경에서는 기존의 제어이론으로는 강건한 제어기의 구성이 용이하지 못하다. 이러한 어려움을 해결하기 위하여 최근에는 인간의 사고능력과 적응능력을 갖는 지능형 제어기가 제안되고 이를 응용한 제어기의 연구가 활발히 진행되고 있다. 이 가운데서도 뉴럴네트웍은 입력과 출력의 정보로부터 입출력 관계를 학습함으로써 자동적으로 지식습득이 가능하므로 정확한 수학적 모델링이 필요치 않으며 정보의 분산처리에 의한 오차보간 능력과 외란에 대한 강건성 및 적응능력 등의 장점을 갖는다. 그리고 비선형 관계를 나타내는데 있어서 뉴럴네트웍의 뛰어난 능력은 모델을 기준으로 한 제어 전략(Model based control strategy)을 가능

* 영남대학교 기계공학부

하도록 하여 시스템의 역모델을 제어기로 사용하게 하는 Internal Model Control(IMC)방법을 가능하게 만들었다[1-3]. 여기서 나타난 아이디어는 시스템의 입출력 관계와 그에 대응하는 역관계를 학습하도록 하는 네트웍의 훈련 가능성에 근거를 두고 있다. 비선형 시스템의 제어에 대한 IMC의 적용은 Economou[4] 등에 의해 입증된 바 있고 현재에는 뉴럴네트웍이 시스템의 모델과 역모델을 동시에 구축하므로써 IMC제어 구조내에서 직접적으로 사용되고 있다. 따라서 이러한 경우 비선형 시스템의 제어문제는 시스템의 모델링 문제로 해결될 수 있으므로 복잡한 비선형 특성을 갖는 시스템을 뉴럴네트웍으로 근사화 하고자 하는 시도가 최근에 활발해 지고 있다. 최근의 연구 결과들 중에 Hornik[5]은 이층 구조 신경회로망이 임의의 연속함수를 얼마만큼의 오차범위내에 근사화할 수 있음을 해석적으로 보였다. 그러나 임의의 모델식으로 주어진 시스템을 에러가 전혀없도록 표현하고자 하는 것은 거의 불가능하므로 얼마나 근사화할 수 있는가 다시 말해 실제시스템과 모델식과의 에러를 얼마나 줄일 수 있는가하는 것이 관건이다. 이러한 에러를 목적함수로 가질 때 큰 자유도와 많은 제약조건을 갖는 이 목적함수는 여러개의 국부 최소점(Local minimum)을 갖게되므로 전체 최소점(Global minimum)을 탐색해야 하는 최적화 문제로 나타난다. 그러나 일반적으로 전체 최소점을 찾기 위해서는 엄청난 시간과 노력을 필요로 하므로 전체 최소점을 찾기보다는 원하는 에러 범위내에 들어오는 국부 최소점을 찾게 되면 학습을 종료한다. 따라서 이러한 학습에서는 원하는 에러 범위안에 있는 국부 최소점을 빨리 찾도록 하는 것이 중요하므로 뉴럴 네트웍에서는 국부 최소점의 발생확률을 높이는 것이 필요하다. 그런데 이를 위해서는 뉴런의 개수를 늘이거나 은닉층의 개수를 늘여야 하는데 이렇게 되면 학습 소요시간이 기하 급수적으로 늘어나게 되므로 뉴럴 네트웍이 갖는 가장 큰 문제는 수렴속도 또는 학습시간을 개선해야 한다는 점이다. 이와 관련한 문제로서 Romelhart 등[6]은 가중치의 조정에 이전의 조정량을 고려하므로써 학습의 안정도를 개선한 운동량(momentum)이라 불리는 방법을 제안하였고 Stornetta 등[7]은 뉴런의 출력을 0에서 1의 범위로 압축하는 대신에 -0.5에서 +0.5의 범위로 압축하는 것이 가중치의 훈련에 훨씬 효과적이라는 발표를 했고 Pineda[8]는 출력을 입력에 피드백함으로써 순환 네트웍(recurrent network)이라는 방법을 제안함으로써 학습시간을 개선했다.

임의의 시스템에 대한 모델을 구하는 또 다른 방법으로 퍼지규칙에 의한 방법이 있는데 뉴럴네트웍보다는 학습시간이 짧게 걸리는 반면에 시스템이 복잡한 경우에는 엄청난 양의 규칙을 필요로 하는 단점이 있다. 따라서 본 연구에서는 이러한 두가지 방법의 단점을 상호 보완하기 위해 주어진 시스템을 여러개의 뉴럴네트웍 모델로 분할하여 특정영역에 대해서만 근사화를 제한적으로 하게 함으로써 학습의 효율을 높이는 방법을 제시하고자 한다. 이 방법은 구조적으로 볼 때 1985년에 Sugeno[9]가 제시한 퍼지추론법과 유사하나 Sugeno는 후건부에서 나타나는 출력을 입력 변수에 대한 선형 조합으로 제안한 반면 본 논문에서는 다층 뉴럴네트웍으로 구성한다는 점이 다르다. 이러한 경우 규칙에 따른 모델의 개수는 늘어나지만 순수하게 뉴럴네트웍만으로 구성할 때 보다는 한층 간단한 네트웍을 사용할 수 있으므로 효과적인 학습이 가능하다.

또 Sugeno의 방법과 비교해도 뉴럴네트웍을 학습하기 위한 시간은 상대적으로 길어지지만 규칙의 개수는 크게 줄일 수 있다. 여기서 중요한 또 다른 문제는 비록 퍼지분할에 의한 뉴럴네트웍의 모델이 단독의 뉴럴네트웍보다 효과적이라고 해도 규칙의 분할방법이 까다롭다면 사용하기 어렵다는 점이다. 예를 들어 입력 변수가 하나뿐 일 때는 입력의 퍼지 분할이 별 문제가 되지 않지만 입력이 두 개 이상일 때는 2차원, 3차원 공간상에서 영역을 적절히 분할해야 하므로 쉬운 일이 아니다. 따라서 가능한 한 쉽고 간단하게 분할할 수 있는 방법을 제안해야 하므로 본 논문에서는 공간상의 임의의 점을 퍼지 멤버쉽 함수의 중심 좌표로 제공하면 각각의 좌표에서 동일한 거리에 놓인 점들로 영역을 나누고 이러한 영역에서 대응하는 뉴럴네트웍 모델을 구성하게 한다. 그리고 나서 학습을 통해 모델링의 에러가 줄어드는 방향으로 뉴럴네트웍과 멤버쉽 함수를 동시에 조정하게 함으로써 초기에 임의의 영역으로 나뉘어 졌던 입력변수의 퍼지분할이 최적의 영역으로 나누어지도록 한다.

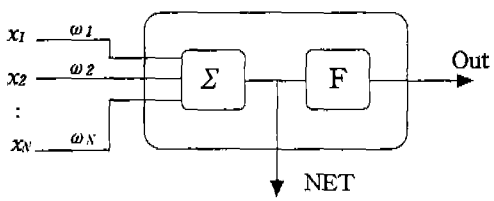
2. 뉴럴네트웍 모델링

신경회로는 상호 연결된 뉴런에 의해 임의의 입력 N차원 공간을 출력 M차원 공간으로 매핑하는 것으로 생각할 수 있으므로 네트웍의 학습은 입력집합의 적용이 원하는 출력집합을 만들어 내도록 뉴런의 연결강도를 조정하는 것이다. 편의상 이들 입출력 집합을 벡터로 나타내면 학

습은 각각의 입력 벡터가 원하는 출력벡터와 쌍을 이룬다고 가정하고 이들을 구현하는 것이다.

2.1 다층 뉴럴네트웍(Multi-layered neural network)

Fig. 1은 다층 뉴럴네트웍에서 기본적인 구조 블록으로 사용되고 있는 뉴런을 보여주고 있다. 그림에서 보는 바와 같이 먼저 외부로부터 주어지거나 바로 앞의 층으로부터 가해진 입력은 임의의 가중치만큼 곱해져서 그 결과가 합해진다. 이렇게 구해진 곱의 합을 NET라고 불였는데 이러한 계산이 네트웍에서 각각의 뉴런에 대해 계산된다. 이러한 NET이 계산된 다음 이것을 변형시키기 위한 활성함수(activation function) F 가 사용되는데 이렇게 함으로써 Out이라는 신호가 만들어진다. 일반적으로 활성함수로는 다음과 같은 함수를 사용하게 되는데 특징으로는 입력 값이 양의 큰 값이나 음의 큰 값인 경우에도 출력 값은 0에서 1 또는 -1에서 +1정도의 범위 내에서 형성된다.



$$NET = x_1 w_1 + x_2 w_2 + \ldots + x_N w_N = \sum_{i=1}^{N} x_i \cdot w_i$$

$$Out = F(NET)$$

Fig. 1 Artificial neuron with activation function

$$Out = 1/(1+e^{-NET}), \quad 0 < Out < 1 \ \ for \ -\infty < NET < \infty \qquad (1)$$

$$\frac{\partial Out}{\partial NET} = Out(1-Out) \qquad (2)$$

또는

$$Out = 2/(1+e^{-NET}) - 1, \ -1 < Out < 1 \quad for \ -\infty < NET < \infty \qquad (3)$$

$$\frac{\partial Out}{\partial NET} = -\frac{1}{2}(1-Out^2) \qquad (4)$$

이러한 함수를 시그모이드(Sigmoid) 함수라고 하는데 역전파 학습에서 사용하게 될 미분 값이 간단히 표현되는 것이 중요하다. 때로는 이 식을 logistic 또는 간단히 squashing function이라고도 하는데 *NET*의 범위를 *0*과 *1*사이의 범위로 압축하기 때문이다. 일단, 하나의 층에 대한 출력 집합이 구해지면 다음 층에 대한 입력으로 사용된다. 이러한 과정이 층별로 행해져 네트웍 출력의 최종 집합이 구해질 때까지 반복된다.

2.2 역전파 학습(Backpropagation learning)

역전파 알고리즘의 발견은 뉴럴네트웍에 대한 관심을 부활시키는데 큰 역할을 했고 다층 뉴럴네트웍을 학습시키는 체계적인 강력한 방법이다. 따라서 이 학습법은 단층 뉴럴네트웍이 갖는 한계를 극복하면서 뉴럴네트웍이 응용될 수 있는 문제의 영역을 극적으로 확장시켰고 또한 성공적인 것으로 입증되었다. Fig. 2에서는 입력층과 출력층 그리고 한 개의 은닉층으로 구성되어 2개의 가중치 행렬로 연결되는 2층의 뉴럴네트웍 모델을 보여주고 있다. 이러한 네트웍에서 훈련을 위한 목표는 입력집합의 응용이 출력의 원하는 집합을 만들어 내도록 연결 가중치를 조정하는 것이다. 편의상 이들 입출력 집합을 벡터로 나타낸 다음, 각 입력 벡터가 원하는 출력을 나타내는 목표치와 쌍을 이룬다고 가정하고 이들을 훈련쌍이라고 하면 보통의 네트웍은 이러한 훈련쌍에 대해 훈련된다. 훈

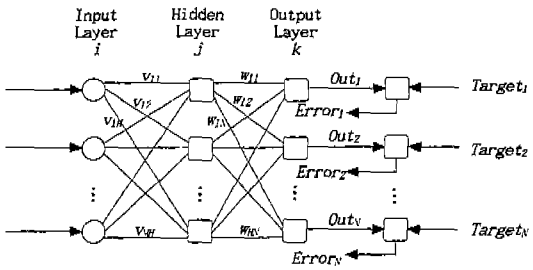


Fig. 2 Two-layer backpropagation neural network

련을 시작하기 전에 모든 가중치는 작은 랜덤 수로 초기화되어 져야 한다. 만일 큰 값의 가중치가 주어져 처음부터 포화되어 버리거나 모두 같은 가중치를 갖게 되면 네트웍은 학습이 될 수 없다.

- 출력층의 가중치 조정

출력층에서 각 뉴런에 대한 목표값이 있으므로 해당 가

중치의 조정은 수정된 델타 법칙을 사용하므로써 쉽게 구할 수 있다. 그러나 내부 층은 이러한 비교를 위한 목표값이 없기 때문에 은닉층이라고 말하며 따라서 학습이 더욱 복잡하다. 먼저 은닉층 j의 뉴런 p에서 출력층 k의 뉴런 q로 가는 하나의 가중치에 대한 훈련을 생각해 보자. 목적함수 J를 에러만의 함수로 나타내면

$$J = \frac{1}{2} Error_{q,k}^{2} \tag{5}$$

이고 여기서 $Error_{q,k} = Target_{q,k} - Out_{q,k}$, $Out_{q,k}$
$= F(NET) = 1/(1+e^{-NET})$, $NET = \sum_{p} Out_{p,j}\ w_{pq,k}$

이다. 따라서 수정되어야 할 가중치의 변화량은 다음과 같다.

$$\begin{aligned}\Delta w_{pq,k} &= -\eta \frac{\partial J}{\partial w_{pq,k}} = -\eta \frac{\partial J}{\partial Out_{q,k}} \frac{\partial Out_{q,k}}{\partial NET} \frac{\partial NET}{\partial w_{pq,k}} \\ &= \eta (Target_{q,k} - Out_{q,k})\ Out_{q,k}(1 - Out_{q,k}) \frac{\partial NET}{\partial w_{pq,k}} \\ &= \eta\ \delta_{q,k}\ Out_{p,j} \end{aligned} \tag{6}$$

여기서 $\delta_{q,k} = Out_{q,k}(1 - Out_{q,k})\ (Target_{q,k} - Out_{q,k})$
이므로 새로운 가중치는 다음과 같이 구한다.

$$w_{pq,k(new)} = w_{pq,k(old)} + \Delta w_{pq,k} \tag{7}$$

- 은닉층에서의 가중치 조정

은닉층의 경우는 목표값을 갖고 있지 않으므로 역전파 학습에서는 출력에러를 층별로 네트웍을 통해 에러를 역전파함으로써 은닉층을 훈련한다. 따라서 윗 식은 출력층 뿐 아니라 은닉층에 대해서도 사용될 수 있다. 그러나 출력층에서 각 뉴런에 대한 δ 가 계산된 것처럼 은닉층에서의 δ 도 목표값의 도움없이 생성되어야 하므로 출력층에서의 δ 가 출력층에 연결되는 가중치를 조정하기 위해 사용되는 것처럼 앞의 층에 역전파된다. 출력층 바로 앞의 은닉층에 있는 뉴런의 경우, 전파 과정에서 자신의 출력값을 연결 가중치를 통해 출력층에 전달한다. 따라서 역전파 과정동안 이들 가중치는 δ 값을 출력층에서 은닉층으로 통과시킴으로써 역으로 동작한다. 은닉층에서 필요로 하는 δ 의 값은 출력층에 연결된 뉴런의 δ 값에 각각의 연결 가중치를 곱한 다음 거기에 압축함수의 미분을 곱함으로써 구해진다.

$$\delta_{p,j} = Out_{p,j}(1 - Out_{p,j})\ (\sum_{q} \delta_{q,k}\ w_{pq,k}) \tag{8}$$

이렇게 은닉층의 각 뉴런에 대한 δ 가 계산되면 그 층과 연관된 모든 가중치가 조정될 수 있고 입력층까지의 모든 가중치도 마찬가지 방법으로 조정이 가능하다. 그 외에도 뉴런에 조정이 가능한 바이어스를 제공하면 뉴런의 역치(threshold)를 조정하는 효과가 있어서 학습이 더 빠르게 수렴할 수 있다.

3. 뉴로 퍼지 모델링

3.1 퍼지 멤버쉽 함수를 갖는 뉴럴네트웍 모델

퍼지 멤버쉽 함수를 갖는 뉴럴네트웍 모델링 시스템의 블록선도는 Fig. 3과 같으며 이 모델의 전건부는 기존의 퍼지규칙과 동일하지만 후건부는 다층의 퍼셉트론을 갖는 뉴럴네트웍으로 구성되어 있다. 따라서 각각의 규칙들은 뉴럴네트웍의 역전파학습 알고리즘에 의해 학습된 다음, 일반적인 퍼지추론법에 의해 추론된다.

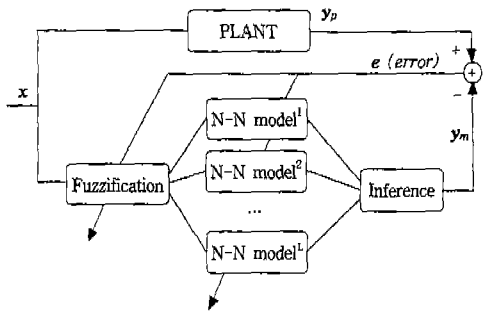


Fig. 3 Neural network model by fuzzy division

만일 입력 변수가 N개이고 출력 변수가 M개인 플랜트의 모델에서 퍼지집합 A의 멤버쉽 함수를 $A(x)$ 라고 나타낼 때, 전제 "*if* x_1 *is* $A_1(x_1)$ *and* x_2 *is* $A_2(x_2)$ *and... and* x_N *is* $A_N(x_N)$" 의 진리값은 $A_1(x_1)\hat{}\ A_2(x_2)\hat{}\ \ldots\hat{}\ A_N(x_N)$ 로 표현할 수 있고 이러한 진리값을 사용한 퍼지함의(fuzzy implication) R은 다음과 같이 나타낸다.

$$R: A_1(x_1)\hat{}\ A_2(x_2)\hat{}\ \ldots\hat{}\ A_N(x_N) \rightarrow (y_1, y_2, \ldots, y_M) = f(x_1, x_2, \ldots, x_N) \tag{9}$$

여기에 사용되는 후건부 $y = f(x)$는 일반적으로 사용하는 뉴럴네트웍 함수로서 전건부의 조건이 만족되면 후건부의 뉴럴네트웍 모델에서 구해진 출력을 해당하는 확률만큼 적용한다. 따라서 이러한 퍼지규칙의 추론은 비록 후건부가 퍼지집합으로 표현되지 않았다 하더라도 뉴럴네트웍 결과를 퍼지 싱글톤 값으로 볼 수 있으므로 기존의 퍼지추론법을 적용할 수 있다.

여기서 k번째 제어규칙 R_k는 다음과 같이 표현할 수 있다.

$$R_k : A_{1,k}(x_1) \hat{} A_{2,k}(x_2) \hat{} \ldots \hat{} A_{N,K}(x_N) \rightarrow y = f_k(x) \quad (10)$$

이 규칙의 의미를 살펴보면 플랜트의 모델이 $y = f_k(x)$라는 값을 가질 가능성은 각각의 퍼지집합 $A_1(x_1)$, $A_2(x_2),\ldots,A_N(x_N)$에서 멤버쉽함수가 갖는 확률의 최소값과 같다는 것을 나타낸다. 전체 규칙의 개수가 L개라면 각각의 규칙은 퍼지 함수에 의해 계산되어야 하므로 다음과 같이 추론될 수 있다.

$$\begin{aligned} \mu R_k &\approx \mu(A_{1,k} \text{ and } A_{2,k} \text{ and } \ldots \text{ and } A_{N,k} \rightarrow y = f_k(x)) \\ &= [\mu A_{1,k}(x_1) \hat{} \mu A_{2,k}(x_2) \hat{} \ldots \hat{} \mu A_{N,K}(x_N) \rightarrow y = f_k(x) \end{aligned} \quad (11)$$

규칙의 전제부(premise)인 $\mu A_{1,k}(x_1) \hat{} \mu A_{2,k}(x_2) \hat{} \ldots \hat{} \mu A_{N,K}(x_N)$는 k번째 모델규칙이 현재의 플랜트 환경과 비교해서 얼마나 적합한지를 나타내므로 k번째 모델식의 영향은 전건부의 적합도로 나타낼 수 있고 모델의 실제출력을 결정할 때에 후건부의 뉴럴네트웍 식을 얼마나 반영해야 할 것인지를 알게 된다. 따라서 k번째 규칙의 적합도(fitness)를 ϕ_k 라고 하면 이 값은 다음과 같이 계산할 수 있다[11].

$$\phi_k = \frac{\mu A_{1,k}(x_1) \hat{} \mu A_{2,k}(x_2) \hat{} \ldots \hat{} \mu A_{N,k}(x_N)}{\sum_{i=1}^{L} \{\mu A_{1,i}(x_1) \hat{} \mu A_{2,i}(x_2) \hat{} \ldots \hat{} \mu A_{N,i}(x_N)\}} \quad (12)$$

여기서 $[\mu A_{1,k}(x_1) \hat{} \mu A_{2,k}(x_2) \hat{} \ldots \hat{} \mu A_{N,k}(x_N)] \approx \min [\mu A_{1,k}(x_1), \mu A_{2,k}(x_2), \ldots, \mu A_{N,k}(x_N)]$ 이다. 이렇게 구해진 적합도에 k번째 규칙의 후건부 값을 곱하면 k번째 규칙에 의한 모델 출력량이 계산되고 마찬가지로 각각의 규칙에 대한 모델 출력의 합을 계산하면 최종적인 모델 출력을 다음과 같이 구할 수 있다.

$$y_m = \sum_{k=1}^{L} \phi_k \times f_k(x) \quad (13)$$

3.2 퍼지 멤버쉽 함수의 학습

뉴럴네트웍 모델의 퍼지분할에서 주어진 영역을 분할하고자 할 때에 만일 그 시스템에 대해 잘 모르는 경우에는 일반적으로 그 영역을 균등하게 나누게 된다. 따라서 시스템의 입력이 하나라면 일차원 시스템이므로 나누는 것이 쉽지만 여러 개의 입력을 갖는 경우라면 그리고 퍼지분할의 개수가 10개 또는 20개인 경우에는 분할자체를 어떻게 할까 하는 것이 더 큰 문제로 등장하게 된다. 따라서 이러한 경우에는 입력 영역에 멤버쉽 함수의 중심점의 위치만을 임의로 준 다음 뉴럴네트웍의 학습과 동시에 이의 위치를 조정하게 함으로써 최적의 영역으로 퍼지분할을 할 수 있게 한다. 이를 위해서는 실제 출력에 대한 모델링 에러를 비용함수로 나타낸 다음, 에러 최소화법(error minimization method)을 사용하여 이를 줄여나간다. 이 가운데서 기울기법(gradient approach)은 뉴럴네트웍에서 가중치를 학습하던 방법과 같으며 에러를 포함한 비용함수 J에 대하여 J가 최소화되도록 규칙의 후건부 및 멤버쉽 함수를 수정해 나간다[10]. 따라서 플랜트의 출력이 모델의 출력과 다른 경우에 그 차이를 모델링 에러로 표현하고 이를 비용함수로 나타내면 비용함수는 다음과 같이 나타낼 수 있다.

$$\begin{aligned} J(h) &= \frac{1}{2}\varepsilon(h)^2 = \frac{1}{2}\{y_p(h) - y_m(h)\}^2 \\ &= \sum_{i=1}^{M} \frac{1}{2}\{y_{p,i}(h) - y_{m,i}(h)\}^2 \end{aligned} \quad (14)$$

여기서, y_p는 플랜트의 출력벡터, y_m는 모델의 출력벡터이고 $y_{p,i}$는 플랜트의 i번째 출력, $y_{m,i}$는 모델의 i번째 출력이다. 이때 최적의 규칙을 구하기 위해서는 이 값을 최소화하도록 퍼지규칙을 수정해 나가야 한다. 만일 퍼지멤버쉽 함수를 가우스함수로 나타내고 규칙의 전제부를 Product-sum방법으로 추론한다면

$\mu A_{j,k}(x_j) = e^{-\frac{1}{2}a_{j,k}(x_j - b_{j,k})^2}$ 이고

$\mu A_{1,k}(x_1) \wedge \mu A_{2,k}(x_2) \wedge \ldots \wedge \mu A_{N,k}(x_N)$

$= e^{-\frac{1}{2}\{a_{1,k}(x_1 - b_{1,k})^2 + a_{2,k}(x_2 - b_{2,k})^2 + \ldots + a_{N,k}(x_N - b_{N,k})^2\}}$ 이므로

$$\phi_k = \frac{e^{-\frac{1}{2}\{a_{1,k}(x_1-b_{1,k})^2+a_{2,k}(x_2-b_{2,k})^2+\ldots+a_{N,k}(x_N-b_{N,k})^2\}}}{\sum_{l=1}^{L} e^{-\frac{1}{2}\{a_{1,l}(x_1-b_{1,l})^2+a_{2,l}(x_2-b_{2,l})^2+\ldots+a_{N,l}(x_N-b_{N,l})^2\}}}$$

$$= \frac{\prod_{j=1}^{N} e^{-\frac{1}{2}a_{j,k}(x_j-b_{j,k})^2}}{\sum_{l=1}^{L}\prod_{j=1}^{N} e^{-\frac{1}{2}a_{j,l}(x_j-b_{j,l})^2}} \quad (15)$$

이때 가우스 멤버쉽 함수의 중심값($b_{j,k}$)과 분산($1/a_{j,k}$)은 다음과 같이 조정될 수 있다[10,11].

$$\Delta a_{j,k} = -\frac{\partial J(h)}{\partial a_{j,k}}$$

$$= \frac{\partial}{\partial a_{j,k}} \sum_{i=1}^{M} \frac{1}{2}(y_{p,i} - y_{m,i})^2$$

$$= \sum_{i=1}^{M} (y_{p,i} - y_{m,i}) \frac{\partial y_{m,i}}{\partial a_{j,k}}$$

여기서 모델의 i번째 출력은 $y_{m,i} = \sum_{k=1}^{L} \phi_k \cdot f_{k,i}(x)$ 이고

$$\frac{\partial \phi_k}{\partial a_{j,k}} = -\frac{1}{2}(x_j - b_{j,k})^2(1-\phi_k)\phi_k,$$

$$\frac{\partial \phi_k}{\partial b_{j,k}} = a_{j,k}(x_j - b_{j,k})\ (1-\phi_k)\phi_k$$

이므로

$$\Delta a_{j,k} = \sum_{i=1}^{M} (y_{p,i} - y_{m,i})$$

$$f_{k,i}(x)\left\{-\frac{1}{2}(x_j - b_{j,k})^2(1-\phi_k)\phi_k\right\} \quad (16)$$

마찬가지로

$$\Delta b_{j,k} = -\frac{\partial J(h)}{\partial b_{j,k}}$$

$$= \sum_{i=1}^{M} (y_{p,i} - y_{m,i})\ f_{k,i}(x)\ \{a_{j,k}(x_j - b_{j,k})\ (1-\phi_k)\phi_k\} \quad (17)$$

따라서 모델링 에러를 줄이기 위한 퍼지 멤버쉽함수의 기울기와 중심값은 다음과 같이 수정된다.

$$a_{j,k(new)} = a_{j,k(old)} + \Delta a_{j,k} \quad (18)$$

$$b_{j,k(new)} = b_{j,k(old)} + \Delta b_{j,k} \quad (19)$$

본 연구에서 사용된 퍼지 멤버쉽 함수는 조정이 쉽도록 하기 위해서 전구간에 대해 미분 가능한 가우스 함수를 사용하고 있지만 실제로 이러한 멤버쉽 함수를 사용한 기여도는 식 (15)에 의해서 계산되므로 이를 그림으로 나타내면 Fig. 4에서와 같이 실제 가우스 함수와는 전혀 다른 형태를 나타낸다. 그리고 비록 가우스 함수가 그 성질상의 이유로 어떤 영역에서도 값이 제로가 되는 경우는 없지만 Fig. 4의 멤버쉽 함수에서 보는 바와 같이 모든 규칙이 전 영역에 골고루 적용이 되어야 한다는 것은 규칙으로도 문제가 있을 뿐만 아니라 대상 영역을 할당하므로써 학습의 효과를 극대화하고자 하는 본 논문의 취지와도 어긋난다. 일반적으로 규칙이라고 할 때에 어쩔수 없는 일부의 예외는 인정해도 가능한 한 해당되는 경우에 대해서는 지배적인 것이 더욱 효과적이므로 여기서는 가우스 함수의 기울기(분산) 값을 매우 크게하여 규칙의 추론 결과 기여도가 Fig. 5에서 보는 바와 같이 사다리꼴 함수로 나타나게 함으로써 조건에 대한 규칙의 기여도를

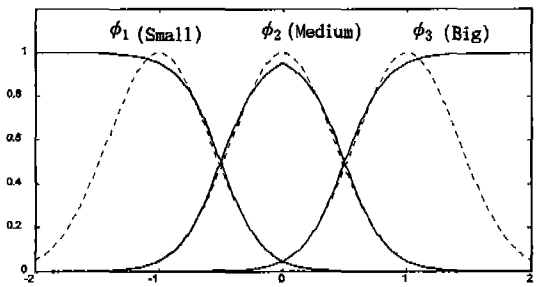


Fig. 4 Gaussian membership function & Rule's fitness

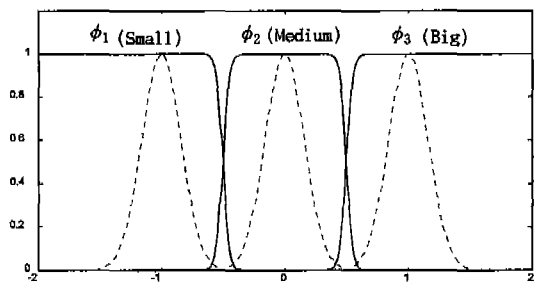


Fig. 5 Fuzzy membership function for fuzzy division

1 또는 0의 경우만으로 양분할 수 있다. Fig. 5의 기여도 함수 ϕ_k 가 전 영역에서 제로의 값은 아니지만 Fig. 4와는 달리 자신의 규칙이 적용될 때에는 거의 100% 기여하고 다른 규칙이 적용되는 영역에서는 기여도가 제로에 가까우므로 규칙에 대한 역할 분담이 확실하게 된다. 그리고 만약 기여도가 아주 작은 부분을 무시하지 않고 고려하게 되면 전건부의 규칙이 모든 경우에 대해서 적용되어야 하므로 후건부의 모든 뉴럴네트웍을 학습해야하는 문제가 발생되어 학습시간이 많이 걸리게 된다. 따라서 규칙의 기여도가 1% 또는 0.1%보다 작은 경우에 대해서는 기여도를 완전히 무시해 버린다면 이러한 부분에서 후건부의 뉴럴네트웍을 학습하는 시간의 낭비를 줄일 수 있게 된다. 따라서 학습에서는 가우스 함수의 기울기 값을 크게하면 할수록 전건부의 조건에 대해 지배적인 규칙만을 학습하므로 효과적으로 분할된 모델을 얻는 것이 가능하게 되어서 모델 전체의 학습시간은 규칙의 개수에 상관없이 거의 일정하게 된다.

4. 시뮬레이션

본 논문에서 제시한 학습 알고리즘의 효과를 살펴보기 위하여 입력과 출력이 각각 2개씩인 2축 로봇의 역기구학 문제에 대해 적용해 본다. 이것은 학습의 결과를 2 차원 평면상에서 도식적으로 나타내는 것이 가능하기 때문이다. Fig. 6에서와 같이 로봇 팔의 길이가 각각 l_1, l_2 이고 기준축에 대한 각도가 θ_1, θ_2 일 때 끝점의 위치를 나타내는 기구학 식은 다음과 같다.

$$x = l_1\cos\theta_1 - l_2\cos(\theta_1+\theta_2) \quad (20)$$

$$y = l_1\sin\theta_1 - l_2\sin(\theta_1+\theta_2) \quad (21)$$

$$0 < \theta_1,\ \theta_2 < \pi,\ l_1 = 0.6m,\ l_2 = 0.4m$$

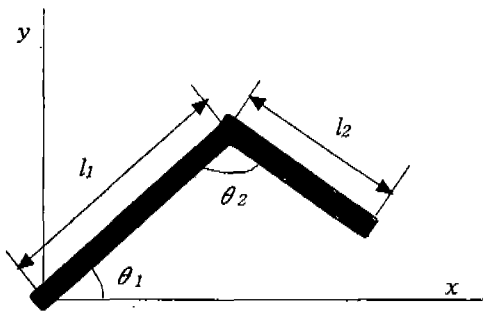


Fig. 6 Diagram of 2-axis manipulator

여기서 만일 임의의 좌표값(x, y)을 나타내는 로봇의 형상 (θ_1,θ_2)를 구하려고 한다면 다음과 같은 로봇의 역기구학 식을 구해야 한다.

$$(\theta_1,\theta_2) = F(x,y) \quad (22)$$

먼저 학습의 범위를 정하기 위해 각각의 축이 0도에서 180도까지 회전 가능하다고 하면 작업영역은 Fig. 7의 회색부분이 된다. 따라서 작업영역 전체에 대해 2개의 은닉층을 가지며 각각의 은닉층 뉴런의 개수가 5개씩인 뉴럴네트웍 모델을 구성하고 전 영역에 걸쳐 임의로 선택한 360개의 훈련쌍들에 대해 $(\theta_1,\theta_2) = F(x_i,y_i),\ i = 1,\ldots,360$ 과 같은 관계를 갖도록 학습하므로써 2축 로봇의 역기구학 문제를 해결하고자 한다. 이때 학습이 이루어지기 전인

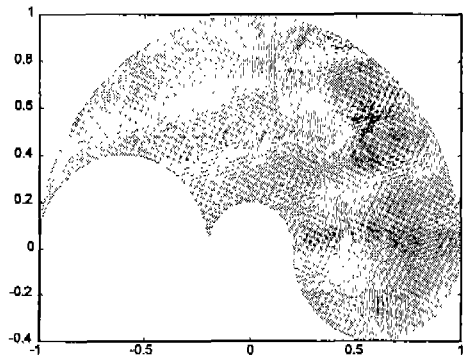
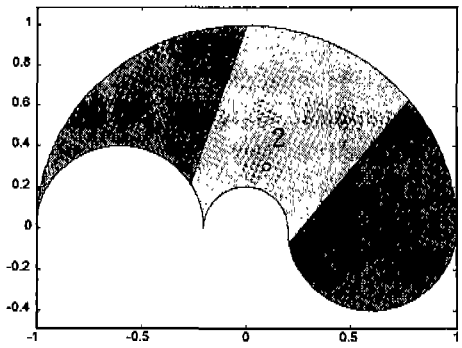
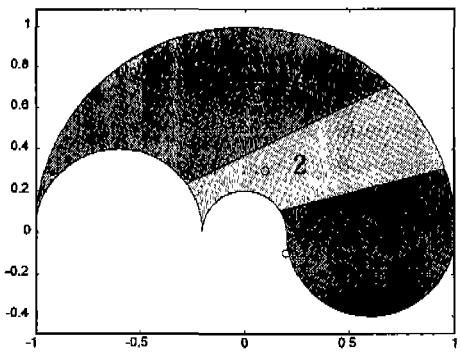


Fig. 7 Working area of 2-axis manipulator

초기의 에러는 가로 세로의 합이 400 ~ 500m가량이 되며 학습의 완료조건을 각각의 훈련쌍에 대해 평균 0.02m 이내로 두고자 한다면 전체 에러의 합이 15m 이내면 된다. 이러한 학습조건에 대하여 종래의 방법인 뉴럴네트웍만으로 학습을 수행했을 때에 300,000회 이상의 학습에 대해서도 에러의 합이 20 이하로 줄어들지 않았다. 그러나 만일 3개의 규칙으로 퍼지분할한 다음 각각의 뉴럴네트웍에 대해 학습하고자 한다면 먼저 Fig. 7의 작업영역에서 모델의 입력 변수 x, y를 사용해서 3개의 영역으로 적절하게 나누는 것이 필요하다. 따라서 제안한 방법에 의해 멤버쉽 함수의 중심위치를 작업영역내에 잘 배치하면 Fig. 8의 (Good division)에 나타난 세점 (0.6, -0.1), (0.1, 0.3), (-0.5, 0.5)으로 제시할 수 있고 이에 따른 규칙의 영역은 Fig. 5와 같은 멤버쉽 함수를 사용하여 두 멤버쉽 함수간의 적합도가 0.5인 점들을



(Good division)



(Bad division)

Fig. 8 Fuzzy division before training

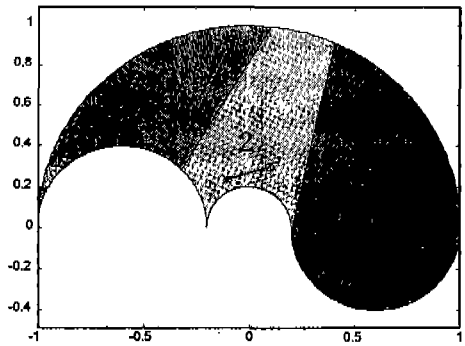


Fig. 9.1 Fuzzy division after training(I)

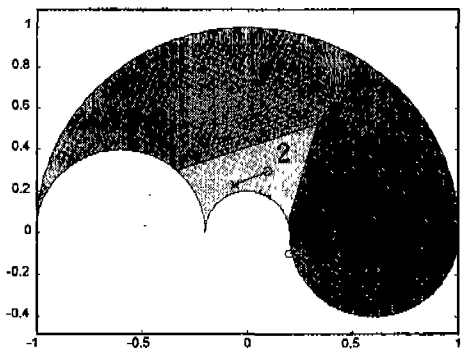


Fig. 9.2 Fuzzy division after training(II)

연결하면 그림과 같이 나눌 수 있다. 이와 같이 처음부터 퍼지분할이 잘된 경우에는 분할영역은 고정시킨 채로 후건부의 뉴럴네트웍에 대해서만 학습을 수행해도 약 50,000회의 학습 후에는 원하는 에러 범위안에 들어 갈 수 있었으나 Fig. 8의(Bad division)과 같이 임의의 세 점을 (0.2, -0.1), (0.1, 0.3), (0.0, 0.5)로 취했을 때에는 멤버쉽 함수에 의한 초기분할이 너무 한쪽에 치우쳐진 관계로 분할이 제대로 이루어지지 않았고 학습에서도 같은 조건의 에러 범위 안에 들어가는데 약 2.5배(약 125,000회)의 시간이 소요되었다. 그러나 뉴럴네트웍의 학습시에 임의로 퍼지분할한 영역에 대한 조정과정을 병행했을 때에는 각각 32,000회와 35,000회에서 학습이 완료되었고 이때 함수의 중심값은 Fig. 9.1과 9.2에서 보는 바와 같이 o에서 *의 위치로 이동되었음을 알 수 있고 또 그림에서는 이동후의 퍼지분할영역이 어떻게 바뀌었나를 보여주고 있다. 여기서 처음에 제시한 멤버쉽 함수의 위치는 서로 많이 달랐지만 학습후의 위치는 점 1과 2는 거의 일치하고 있고 점 3의 경우는 다소 큰 차이를 보이고 있지만 초기 위치와 비교하면 서로 많이 접근했음을 알 수 있고 접근하는 방향으로 보아 만일 학습의 조건을 좀 더 강화한다면 점 3의 위치도 거의 비슷하게 나타날 것이라고 예상할 수 있다. 보다 더 간단한 뉴럴네트웍 모델을 사용하기 위해서는 규칙의 개수를 늘리면 되므로 이를 위해서는 모델의 입력영역에 대한 멤버쉽 함수의 중심값을 임의로 더 제공하는 것 만으로 쉽게 해결할 수 있다. 따라서 본 시뮬레이션은 뉴럴네트웍의 모델링에서 일차적으로 퍼지분할을 병행하는 것이 모델링 에러를 빠르게 최소화시키는 효과적인 방법임을 보여주고 있으며 비록 처음에 퍼지영역을 잘못 분할했을지라도 멤버쉽 함수의 조정 과정을 병행한다면 결과적으로 분할영역을 최적화할 수 있으므로 퍼지분할의 효과가 극대화되는 결과를 얻을 수 있음을 보여주었다.

5. 결 론

뉴럴네트웍을 이용한 시스템의 모델링은 일반적으로 그

이론은 간단하지만 실제 적용에 있어서는 몇 개의 은닉층을 사용해야 할지 또는 각각의 은닉층에서 노드의 개수를 몇 개로 하는 것이 적당한지 하는 등의 판단이 쉽지 않고 한번 학습하는데 걸리는 시간이 길기 때문에 시행착오를 통해서 적절한 모델을 찾는 것도 용이한 일은 아니다. 그리고 처음부터 너무 많은 은닉층과 노드를 가진 모델에서 학습을 시작할 경우, 학습시간이 많이 걸리는 것은 물론이고 모델의 일차 미분 값이 심하게 변하는 현상을 보이므로 이 또한 바람직하지 않다. 따라서 본 논문에서는 임의의 시스템에 대한 뉴럴네트웍 모델을 퍼지분할법을 이용하여 여러개의 부모델로 나눈 다음 이들을 기울기가 큰 가우스 함수를 사용해 거의 독립적으로 학습함으로써 전체에 대한 모델을 빠르게 학습할 수 있는 방법을 제안하였다. 시뮬레이션 결과에 의하면 뉴럴네트웍 모델에서 학습 시간이 많이 걸리거나 원하는 에러의 범위안에 수렴이 잘 안되는 경우에는 이와 같이 대상 영역을 여러 개의 영역으로 나누어서 학습하는 것이 아주 효과적임을 알 수 있었다. 다음으로 퍼지영역의 분할방법에 관해서는 학습과 동시에 스스로 최적의 영역을 찾아내는 방법을 제안하였다. 제안한 방법은 입력변수가 여러개 일 때에도 사용이 편리하도록 입력 영역에 대해 가우스 멤버쉽 함수의 중심점의 위치만을 임의로 제시하면 등간격으로 영역분할을 한 다음 뉴럴네트웍의 학습과 동시에 이의 위치를 조정하게 함으로써 최적의 영역으로 퍼지분할을 할 수 있음을 보였다.

6. 참 고 문 헌

1. Gracia, C. E., and Morari, M., "Internal model control. 1. A unifying review and some new results", Ind. Eng. Chem. Process Des. Dev., Vol. 25, pp.403-411, 1982.
2. Psaltis, D., Sideris, A., and Yamamura, A. A., "A multi-layered neural network controller", IEEE Control System Mag., Vol. 8, pp. 17-21, 1988.
3. Hunt, K. J. and Sbarbaro, D., "Neural networks for nonlinear internal model control", IEE Proceedings-D, Vol. 138, pp. 431-438, 1991.
4. Economou, C. G., Morari, M., and Palsson, B. O., "Internal model control. 5. Extension to nonlinear systems", Ind. Eng. Chem. Process Des. Dev., Vol. 25, pp.403-411, 1986.
5. Hornik, K., "Approximation capabilities of multi-layer feedforward networks",Neural Networks, Vol. 4, pp. 251-260, 1991.
6. Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning internal representations by error propagation", Parallel distributed processing, Vol. 1, pp. 318-362, 1986. Cambridge, MA: MIT Press.
7. Stornetta, W. S., and Huberman, B. A., "An improved three-layer backpropagation algorithm", Proceed. of IEEE 1st Inter. conf. on Neural Networks, San Diego, CA,1987.
8. Pineda, F. J., "Generalization of backpropagation to recurrent and higher order networks", Neural information processing systems, ed. Dana Z. Anderson, pp. 602-611, 1988. New York: American Institute of Physics
9. Takagi, T. and Sugeno, M., "Fuzzy identification of systems and its application to modeling and control", IEEE Trans. on Systems, Man, and Cybernetics, Vol. 15, , No. 1, pp. 116-132, 1985.
10. Chung, B. M. and Oh, J. H., "Autotuning method of membership function in a fuzzy learning control", Journal of Intelligent & Fuzzy Systems, Vol. 1, No. 4, pp. 335-349, 1993.
11. Lee, S., and Kil, R. M., "A gaussian potential function network with hierarchically self-organizing learning", Neural Networks, Vol. 4, pp. 207-224, 1991.