퍼지 및 지능시스템학회 논문지 2002, Vol. 12, No. 1, pp. 7-13

삼각형 소속함수로 구성된 퍼지시스템의 고속 퍼지추론 알고리즘

Fast Fuzzy Inference Algorithm for Fuzzy System constructed with Triangular Membership Functions

유병국

Byung-Kook Yoo

한려대학교 멀티미디어정보통신공학과

요 약

퍼지이론의 응용은 대부분 퍼지추론을 이용하는 것이다. 그러나 퍼지추론은 입력변수의 수가 많아지거나 각 입력변수에 많은 수의 퍼지라벨을 설정할 경우 그 추론에 필요한 계산시간이 많아지게 되며 이러한 것은 컴퓨터 연산의 대수곱(arithmetic product)의 수에 의해 결정된다. 더구나 퍼지추론의 응용이 가장 활발한 퍼지제어분야에서는 이러한 추론시간은 실제 시스템에 적용 시 가장 큰 제약조건이 된다. 특히, 마이크로프로세서를 이용하거나 PC-based 제어기를 설계할 때 이러한 추론시간은 매우 중요한 문제가 된다. 본 논문에서는 이러한 추론시간을 효율적으로 줄이기 위해, 즉 추론 시 필요로 하는 곱 연산의 수를 줄이기 위하여 삼각형 소속함수를 이용하는 퍼지시스템에 적용 가능하며 정보의 손실이 발생되지 않는 간단한 고속 퍼지추론 알고리즘을 제안한다. 이것은 퍼지 추론 시 입력상태공간의 분할과 간단한 기하학적 해석을 통해 얻어지는 것이며 결과적으로 퍼지규칙의 수를 줄이는 것과 같다.

Abstract

Almost applications using fuzzy theory are based on the fuzzy inference. However fuzzy inference needs much time in calculation process for the fuzzy system with many input variables or many fuzzy labels defined on each variable. Inference time is dependent on the number of arithmetic product in computation process. Especially, the inference time is a primary constraint to fuzzy control applications using microprocessor or PC-based controller. In this paper, a simple fast fuzzy inference algorithm(FFIA), without loss of information, was proposed to reduce the inference time based on the fuzzy system with triangular membership functions in antecedent part of fuzzy rule. The proposed algorithm was induced by using partition of input state space and simple geometrical analysis. By using this scheme, we can take the same effect of the fuzzy rule reduction.

Key Words : Fuzzy system, Fuzzy inference, Fuzzy rule reduction, Fuzzy control

1. 서 론

퍼지이론의 응용은 대부분 퍼지추론[1-3]을 이용하는 것이며 이것은 제어분야와 신호처리분야, 소프트웨어분야 등의 전자공학뿐만 모든 공학분야와 인문사회과학까지도 이러한 퍼지추론은 이용될 수 있다. 이러한 것은 퍼지이론을 이용함으로써 전문가의 지식이나 언어적으로 표현된 모호한 지식이나 정보를 좀더 타당하고 객관적으로 처리할 수 있다는 장점이 있기 때문이다. 여러 가지 응용 중 가장 활발한 연구가 이루어지는 것은 제어분야이며 이것은 기존의 제어이론이 복잡하고 고차인 시스템에 대하여 그 수학적 모델을 정확히 찾을 수 없고 또한 제어기를 설계하기가 어렵기 다는 문제점을 해결할 수 있는 것이 퍼지제어방식이기 때문이다. 그러나 초기의 퍼지제어방식은 그 안정도를 보이기 어려웠다. 이러한 이유로 많은 제어이론연구자로부터 주목을 받지 못하였으나 최근 퍼지시스템의 근사성질(Universal Approximation Theorem[4-8])의 증명 이후로부터 다시 퍼지제어방식의 연구가 활발히 진행 중에 있다. 그러나 이러한 퍼지추론에서 가장 크게 대두되는 문제가 퍼지규칙의 수이다. 즉, 퍼지시스템의 입력변수가 늘어남에 따라, 그리고 각 입력변수 상에 설정된 퍼지라벨의 수가 늘어남에 따라 필요로 하는 퍼지규칙의 수는 기하급수적으로 증가하게 된다. 더구나 퍼지추론의 제어분야 응용은 추론에 필요한 계산시간이 빨라야 된다는 제약조건이 있다. 즉, 디지털컴퓨터를 이용한 제어기 설계 시 Sampling 시간이내에 제어입력을 계산해야 한다. 따라서 이러한 문제를 해결하기 위해 최근 들어 퍼지규칙의 수를 줄이고자하는 연구들이 행해지고 있다. 수치해석적 방법인 보간법을 이용하는 방식[9]이 있었으며 여기에 단일값 분해법(SVD:singular value decomposition)을 이용한 방식[10]과 퍼지규칙의 잉여(redundancy) 부분을 제거하려는 방식[11]이 있었다. 또한 뉴로-퍼지에서의 방식으로 iteration algorithm을 이용한 방식[12]과 직교변환방법을 이용한

접수일자 : 2001년 7월 9일
완료일자 : 2001년 12월 28일

방식[13]이 제안되기도 하였으나 아직 뚜렷한 방법이 제시되고 있지 않고 있으며 이러한 방식들의 적용은 퍼지시스템의 형태, 즉 단일 값 결론부(SVCT:Signle-valued consequence type)를 갖는 타입, Sugeno-type(ST), Takagi-Sugeno- type(TST)에 따라 그 적용이 제한된다는 단점을 가지고 있다. 또한 이러한 퍼지시스템의 형태는 결론부의 설정에 따라 여러 가지 방식이 존재할 수 있다. 그리고 특히 [9-11]의 방식은 퍼지규칙을 미리 설계한 후에 그 규칙의 내용을 분석하여 퍼지규칙을 줄이는 방식으로서 실시간으로 퍼지시스템의 파라미터를 적응시키는 적응방식에의 응용[4-8]에는 사용될 수 없는 방식이다.

따라서 본 논문에서는 SVCT-퍼지시스템이나 ST-퍼지시스템, 그리고 TST/TSKT-퍼지시스템에 관계없이 적용할 수 있으며 또한 실시간 적응 퍼지시스템을 이용하는 응용에서 사용가능한 방식으로서 퍼지시스템의 전제부(antecedent)에 사용된 퍼지집합의 형태가 삼각형 퍼지집합인 경우에 대하여 곱 연산의 수를 줄이는 방식을 제안한다. 이것은 결론적으로 퍼지규칙의 수를 줄이는 효과와 같으며 기존의 모든 퍼지추론을 이용하는 응용에서 이용될 수 있으며 또한 앞에서 제안된 여러 가지 퍼지규칙 줄임 알고리즘을 적용 한 후에 다시 그 퍼지추론의 계산시간을 줄이기 위해서도 적용될 수 있는 알고리즘이다.

본 논문의 구성은 2장에서 삼각형 퍼지집합을 이용한 $max \cdot min$퍼지추론 과정을 단일값 결론부를 갖는 형태의 퍼지시스템을 이용하여 간단히 살펴보고 3장에서는 $max \cdot min$추론과정을 좀더 분석하고 또한 입력상태공간의 분할과 삼각형 퍼지집합의 기하학적 성질을 이용하여 한번의 추론과정에서 적용되는 퍼지규칙의 수가 줄어들 수 있음을 보인다. 4장에서는 $max \cdot product$추론에 대하여 간단히 살펴보고 두 가지 추론방식에 대하여 곱 연산의 수를 본 논문에서 제안된 FFI알고리즘을 적용하지 않은 경우와 적용한 경우에 대하여 곱의 수를 비교 분석한다. 마지막으로 5장에서는 결론과 향후 연구계획에 대하여 논한다.

2. 퍼지시스템의 추론과정 분석

일반적인 퍼지시스템은 그림 1과 같은 기본 구조를 가진다. 퍼지시스템의 구분은 퍼지추론에 사용되는 합성연산에 따라 $max \cdot min$방식과 $max \cdot product$방식으로 크게 나눌 수 있다. 또한 퍼지규칙 결론부의 형태에 따라 SVCT, ST, TST/TSKT와 그 외의 여러 가지 방식으로 구분될 수 있으나 본 논문에서 제안한 고속 퍼지추론알고리즘(Fast Fuzzy Inference Algorithm:FFIA)은 퍼지시스템 결론부의 형태와는 관계가 없으므로 간단하게 SVCT-퍼지시스템의 추론과정을 이용하여 알고리즘을 보인 후 ST나 TST/TSKT-퍼지시스템에의 적용은 Remark 1에서 논의한다. 먼저 알고리즘을 쉽게 설명하기 위하여 $max \cdot min$추론방식을 이용하며 단일값 퍼지화와 무게중심법 비퍼지화, 그리고 삼각형 퍼지집합을 이용하는 SVCT-퍼지시스템을 고려한다.

그림 1과 같이 주어진 퍼지시스템에 대하여 SVCT-퍼지시스템의 l번째 퍼지규칙은 다음과 같다.

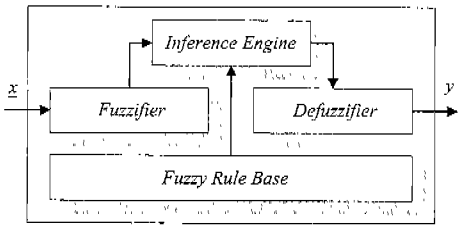

그림 1. 퍼지시스템의 기본구조
Fig. 1. Basic structure of fuzzy system

$$R^l: If\ x_1\ is\ F_1^l \text{ and } x_2\ is\ F_2^l \text{ and } \cdots x_n\ is\ F_n^l,\ then\ y\ is\ Q^l \quad (1)$$

여기서 x_i는 퍼지시스템의 입력변수이며 F_i^l와 Q^l은 각각 l번째 규칙의 x_i변수 상에 설정된 퍼지집합과 출력 변수 y 상에 정의된 퍼지집합이다. 그리고 본 논문에서는 퍼지시스템의 전체 규칙 수를 $M(l=1,\cdots,M)$, 그리고 각 입력변수 x_i상에 정의된 퍼지라벨의 수는 각각 k_i개로 나타낸다. 따라서 전체 퍼지규칙이 입력상태공간상의 상태 값에 대하여 커버(cover)가 되기 위해서는 순열의 곱셈법칙에 의해 $k_1 \times k_2 \times \cdots \times k_n$개의 퍼지규칙이 필요하게 된다.

이 퍼지시스템에서 단일 값 퍼지화와 $max \cdot min$합성연산을 사용하는 추론과정중 l번째 규칙에 의한 과정을 그림으로 살펴보면 그림 2와 같다. 단, 여기서 사용되는 퍼지집합의 형태는 삼각형 퍼지집합이며 입력변수는 x_1과 x_2 두 개인 경우이다.

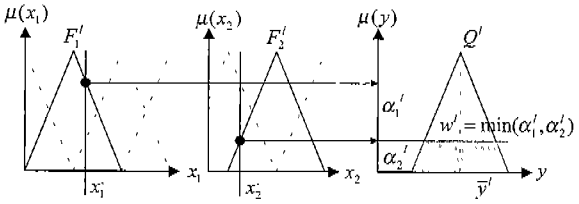

그림 2. 삼각형 퍼지집합, 단일값 퍼지화를 사용한 $max \cdot min$ 추론과정
Fig. 2. $max \cdot min$ inference procedure using triangular fuzzy sets and singleton fuzzification

여기서 α_1^l, α_2^l은 각각 $\mu_{F_1^l}(x_1')$과 $\mu_{F_2^l}(x_2')$이다. 결국 l번째 규칙의 접화정도(firing rate)는 min연산에 의해 결정되고 이 때 얻어지는 퍼지집합은 그림 2의 음영부분이 된다. 이렇게 추론된 M개의 퍼지집합에 대하여 max연산을 통해 하나의 퍼지집합으로 만들고 이에 대한 무게중심법 비퍼지화과정을 통하여 y값을 얻어내며 이를 하나의 식으로 표현하면 다음과 같다.

$$y(\underline{x}) = \frac{\sum_{l=1}^{M} \bar{y}^l \cdot min(\alpha_1^l, \alpha_2^l, \cdots, \alpha_n^l)}{\sum_{l=1}^{M} min(\alpha_1^l, \alpha_2^l, \cdots, \alpha_n^l)} \quad (2)$$

여기서 $\bar{y}^l$은 Q^l퍼지집합의 소속정도 중 그 소속정도가 1.0에 해당되는 y공간상의 값이며 $\mu_{Q^l}(y)$는 그 종

류에는 관계없이 좌우 대칭이면서 모두 같은 형태의 퍼지집합이어야 한다. 이를 선형의 형태로 퍼지베이시스(전제함수벡터)함수와 파라미터벡터로 나누어 쓰면 다음과 같이 표현될 수 있다.

$$y(\underline{x}) = \theta^T \xi(\underline{x}) \tag{3}$$

여기서 $\theta=(\bar{y}^1, \bar{y}^2, \cdots, \bar{y}^M)^T$인 $M\times 1$ 파라미터벡터이며 $\xi(\underline{x})=(\xi^1(\underline{x}),\xi^2(\underline{x}),\cdots,\xi^M(\underline{x}))^T$는 $M\times 1$인 전제함수 벡터이다. θ를 파라미터벡터라 하는 것은 대부분의 퍼지시스템 응용에서 θ를 적응시킬 수 있는 퍼지시스템의 파라미터(adaptable parameters)로 설정하여 사용하기 때문이며 전제함수벡터는 다음과 같다.

$$\xi^l(\underline{x}) = \frac{min(\alpha_1^l, \alpha_2^l, \cdots, \alpha_n^l)}{\sum_{l=1}^{M} min(\alpha_1^l, \alpha_2^l, \cdots, \alpha_n^l)} = \frac{w^l}{\sum_{l=1}^{M} w^l} \tag{4}$$

여기서 $w^l = min(\alpha_1^l, \cdots, \alpha_n^l)$이며 이것은 $max\cdot min$추론방법에 대한 것이며 $max\cdot product$추론방식의 표현식은 4장에서 다룬다.

3. 고속 퍼지추론 알고리즘

3.1 전제부 추론 과정

고속 퍼지추론 알고리즘, 다시 말해 추론과정에서 곱계산의 수를 줄이는 알고리즘을 논하기 위해 다음과 같은 정의를 필요로 한다.

정의 1 : 각 입력변수 상에 설정된 퍼지집합의 형태가 그림 3과 같을 때 퍼지집합 A_i^j의 표현은 다음과 같다.

$$A_i^j = (C_i^j, L_i^j, R_i^j),\quad i=1,2,\cdots,n,\quad j=1,2,\cdots,k_i \tag{5}$$

여기서 A_i^j는 i번째 입력변수 상에 정의된 j번째 퍼지집합을 나타내는 것이다. 정의 1과 같이 정의된 퍼지집합에 대한 소속정도 값은 (6)과 같이 계산되어진다.

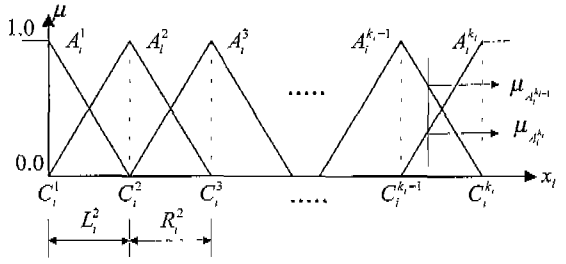

그림 3. 입력변수 x_i상에 설정된 퍼지집합

Fig. 3. Fuzzy sets defined on input variable x_i

$$\mu_{A_i^j}(x_i) = \begin{cases} \dfrac{x_i+(L_i^j-C_i^j)}{L_i^j}, & C_i^j-L_i^j \le x_i \le C_i^j \\ -\dfrac{x_i-(R_i^j+C_i^j)}{R_i^j}, & C_i^j \le x_i \le C_i^j+R_i^j \\ 0, & Otherwise \end{cases} \tag{6}$$

여기서 $i=1,2,\cdots,n,\ j=1,2,\cdots,k_i$ 이다. 따라서 (3)과 같은 퍼지시스템의 전제부가 전체 입력상태공간 상의 상태 값에 대하여 커버(cover)가 되기 위해서는 F_i^l를 각각의 A_i^j에 대한 순열의 곱셈법칙으로 설정한다면 $k_1\times k_2\times\cdots\times k_n$개의 퍼지규칙이 필요하게 된다. 따라서 퍼지추론의 과정에서 삼각형퍼지집합을 사용하기 때문에 $\mu_{F_i^l}(x_i')$, $i=1,2,\cdots,n$ 값을 계산하기 위해 (6)과 같이 한번의 곱(product)연산을 해야 하며 결국 각 규칙에 대하여 n번의 곱과 전체적으로 M개의 규칙에 대한 계산이므로 퍼지추론을 위한 전제부에서의 곱계산은 $n\times M = n\times k_1\times k_2\times\cdots\times k_n$번의 곱 계산을 하게된다. 그러나 실제 퍼지시스템의 추론과정을 프로그래밍 할 때에는 다음과 같은 과정으로 전제부의 추론계산이 이루어진다. 여기서 알고리즘은 pseudo-코드로 나타낸다.

알고리즘 1 : 전형적인 퍼지추론 알고리즘 - 전제부

Algorithm 1 : Conventional algorithm for fuzzy inference - antecedent part

```
/* initialize */
1. k[0]← k_1, k[1]← k_2, ···, k[n-1]← k_n
2. c[0][0]← C_1^1, c[0][1]← C_1^2, ···, c[n-1][k_n-1]← C_n^{k_n}
3. l[0][0]← L_1^1, l[0][1]← L_1^2, ···, l[n-1][k_n-1]← L_n^{k_n}
4. r[0][0]← R_1^1, r[0][1]← R_1^2, ···, r[n-1][k_n-1]← R_n^{k_n}

/* inference of antecedent part */
1. x[0]← x_1, x[1]← x_2, ···, x[n-1]← x_n
2. FOR i=0 THRU n
     a. FOR j=0 THRU k[i]
          1. mem_v[i][j] ← cal_m_funct(i, j, x[i]);
3. FOR j1=0 THRU k[0]-1
     a. FOR j2=0 THRU k[1]-1
          ⋮
          1. FOR jn=0 THRU k[n-1]-1
               a. k ← j1*k[0] + j2*k[1] + ··· + jn
               b. w[k] ← min_v_funct(mem_v[0][j1],
                                     ···, mem_v[n-1][jn])
4. GOTO calculation procedure of consequent part

/* cal_m_function */
function cal_m_funct(i, j, x[i])
1. IF(c[i][j]-l[i][j]<x[i]<c[i][j]+r[i][j]) THEN
     a. IF(c[i][j]-l[i][j] < x[i] <= c[i][j]) THEN
          1. RETURN((x[j]+l[i][j]-c[i][j])/l[i][j])
     b. ELSE IF(c[i][j] <= x[i] < c[i][j]+r[i][j]) THEN
          1. RETURN( (r[i][j]+c[i][j]-x[i])/r[i][j] )
2. ELSE IF( ((x[i]>c[i][j])AND(j==k[i]))
        OR ((x[i]<c[i][j])AND(j==0)) ) THEN
     a. RETURN(1.0)
3. ELSE return(0.0)
```

알고리즘 1의 min_v_funct(···)는 요소들 중 최소 값을 반환해주는 서브함수이며 cal_m_funct()내의 2는 입력상태 값이 해당 전제집합(universe of discourse)상의 퍼지라벨 설정 범위를 벗어난 경우 $\mu_{C_i^1}$ 또는 $\mu_{C_i^{k_i}}$의 값을

1.0으로 주기 위한 것이다. 따라서 실제 프로그램에서 사용되는 정수의 곱셈(inference 부분의 3-a-…-1-a)을 무시하고 소속정도 값을 얻기 위한 실수(floating point number)의 곱 연산만을 고려한다면 최대 $\sum_{i=1}^{n} k_i$번 연산을 하게된다. 즉, cal_m_funct()함수가 $\sum_{i=1}^{n} k_i$번 호출된다. 그러나 다음과 같은 가정을 만족하도록 퍼지집합을 설정한다면 그 곱 연산의 수를 줄일 수 있다.

가정 1 :

a. 퍼지규칙의 전제부에 사용되는 퍼지집합의 소속함수는 삼각형이다. 단, 이등변 삼각형일 필요는 없다.

b. 각 입력변수 상에 설정된 퍼지집합 중 임의의 인접한 3개의 퍼지집합이 다음 식을 만족한다.

$$R_i^{a-1} = L_i^a,\ R_i^a = L_i^{a+1},\quad C_i^{a-1} = C_i^a - L_i^a,\ C_i^{a+1} = C_i^a + R_i^a,\quad \text{for } 1 < a < k_i,\ i = 1,2,\cdots,n \tag{7}$$

정리 1 : 퍼지규칙의 전제부를 구성하는 퍼지집합을 가정 1을 만족하도록 설정하고 알고리즘1에서 조건을 먼저 파악하여 소속정도 값을 계산하도록 알고리즘 1을 알고리즘 2와 같이 수정한다면 한번의 추론과정에서 필요한 전제부에서의 실수 곱 연산(arithmetic product)의 수, NAP_A(number of arithmetic product in anteced- ent part)는 다음과 같다.

$$NAP\_A \le n \tag{8}$$

증명 : 알고리즘 1의 cal_m_funct()에서 한 번의 호출에 대하여 참조된 x[i]의 값은 cal_m_funct()내의 1-a, 1-b, 2, 3조건 중 하나의 조건을 만족하게 된다. 그러나 실제 곱 연산은 1-a 와 1-b의 조건을 만족할 때이며 2와 3에 대하여는 곱 연산이 발생되지 않는다. 극단적인 예로 퍼지집합을 입력변수 x_i상에 C_i^j를 설정하고 모든 $L_i^j + R_i^j$, $j = 1,2,\cdots,k_i$값이 전제집합 구간과 같도록 설정한다면 (그림 4) 반드시 알고리즘의 1-a와 1-b중 하나의 조건을 만족하거나 2, 3의 조건을 만족하게 되어 필요한 곱 연산의 수는 cal_m_funct()함수의 호출횟수인 $\sum_{i=1}^{n} k_i$보다 작거나 같다고 할 수 있다. 즉, $NAP\_A \le \sum_{i=1}^{n} k_i$. 그러나 가정 1을 만족하도록 퍼지규칙 전제부의 퍼지집합을 설정하면 두 개의 인접합 퍼지집합에 대한 소속정도는 서로 보수관계(complement- ary relation)가 있다. 즉, 그림 3에서 $\mu_{A_i^j}(x_i) = 1 - \mu_{A_i^{k-1}}(x_i)$이다. 따라서 알고리즘 1

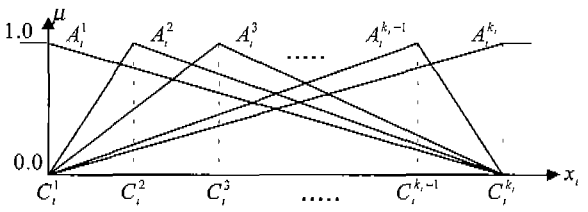

그림 4. $L_i^j + R_i^j$=[x_i에 대한 전체집합] 경우의 퍼지집합 설정

Fig. 4. Fuzzy set distribution of the case of $L_i^j + R_i^j$ =[universe of discourse for x_i]

을 알고리즘 2와 같이 수정하여 조건을 먼저 확인하도록 한다면 하나의 입력변수 x_i에 대한 k_i개의 퍼지 소속정도 값을 계산하기 위해 소속정도 값 계산을 한번만 수행하게 되며 인접한 퍼지집합의 소속정도는 그 값의 보수값을, 나머지는 모두 0이 된다. 또한 $[C_i^1, C_i^{k_i}]$구간 이외의 값에 대하여는 곱 계산이 발생되지 않는다. 결국 한 번의 전제부 추론과정에서 필요로 하는 곱 연산의 수, NAP_A는 (8)을 만족한다. Q.E.D.

알고리즘 2 : 고속 퍼지추론을 위한 개선된 알고리즘 전제부
Algorithm 2 : Modified algorithm for fast fuzzy inference - antecedent part

```
/* initialize */
1. k[0]← k1, k[1]← k2, …, k[n-1]← kn
2. c[0][0]← C1^1, c[0][1]← C2^2, …, c[n-1][kn-1]← Cn^kn
3. l[0][0]← L1^1, l[0][1]← L2^2, …, l[n-1][kn-1]← Ln^kn
4. r[0][0]← R1^1, r[0][1]← R2^2, …, r[n-1][kn-1]← Rn^kn

/* inference of antecedent part */
1. x[0]← x1, x[1]← x2, …, x[n-1]← xn
2. FOR i=0 THRU n-1
   a. FOR j=0 THRU k[i]-2
      1. IF(c[i][j]<= x[i] <= c[i][j+1]) THEN
         a. mem_v[i][j]←(r[i][j]+c[i][j]-x[i])/r[i][j]
      2. ELSE mem_v[i][j] = 0.0 ;
   b. FOR j=0 THRU k[i]-2
      1. IF(mem_v[i][j] ≠ 0.0 ) THEN
         a. mem_v[i][j+1]← 1.0 - mem_v[i][j]
   c. IF (c[i][0] > x[i]) THEN
      1. mem_v[i][0] ← 1.0
   d. IF (c[i][k[i]] < x[i]) THEN
      1. mem_v[i][k[i]] ← 1.0
3. FOR j1=0 THRU k[0]-1
   a. FOR j2=0 THRU k[1]-1
      ⋮
      1. FOR jn=0 THRU k[n-1]-1
         a. k ← j1*k[0] + j2*k[1] + … + jn
         b. w[k] ← min_v_funct(mem_v[0][j1],
                               …, mem_v[n-1][jn])
4. GOTO calculation procedure of consequent part
```

3.2 결론부 추론

전제부와 마찬가지로 결론부의 추론과정에서도 기존의 알고리즘을 수정함으로써 곱 연산의 수를 줄일 수 있다. 구간을 고려하지 않고 (2)만을 이용하여 결론부의 추론을 행한다면 한번의 추론과정 중 결론부 추론에 필요한 곱 연산의 수는 $2M = 2 \times k_1 \times \cdots \times k_n$이다. 그러나 전제부의 추론과정을 통하여 얻은 모든 규칙의 w^l, $l = 1,2,\cdots,M$에 대하여 연산 할 필요가 없다. 이것은 무게중심법을 통해 얻어지는 최종 추론값을 계산할 때 결론부의 퍼지집합에 대하여 각 퍼지규칙에 대한 점화도 w^l

의 값이 *min* 또는 *product* 연산에 의해 0이 되는 경우가 발생하고 이러한 경우에는 결론부의 계산에서 제외할 수 있기 때문이다.

정리 2 : 가정 1을 만족하도록 전제부의 퍼지집합을 설정하고 결론부의 퍼지집합을 좌우 대칭이면서 동일한 모양의 퍼지집합으로 설정하고 알고리즘 3과 같이 점화도가 유효한지에 대한 조건문(2-a, 4-a)을 삽입하므로써 한번의 추론과정 중 결론부에서 행해지는 곱 연산의 수, NAP_C(number of arithmetic product in conseq- uent part)는 다음과 같다.

$$NAP\_C = 2^{n+1} \tag{9}$$

증명 : 가정 1을 만족하도록 전제부 퍼지집합을 설정하고 각 x_i를 포함하는 구간이 정해지면 이 때 $w^l \neq 0$, $l = 1, 2, \cdots, M$인 경우는 전체 규칙 중 일부분에 해당된다. 만약 $w^l = 0$이면 해당되는 규칙은 점화되지 않는다. 따라서 (2)를 다음과 같이 다시 쓸 수 있다.

$$y(\underline{x}) = \frac{\sum_{l=1, w^l \neq 0}^{M} \bar{y}^l \cdot w^l}{\sum_{l=1, w^l \neq 0}^{M} w^l} \tag{10}$$

이와 같이 점화되는 규칙의 수를 알아보기 위하여 그림 5를 고려해본다. 여기서 Box된 퍼지집합은 현재 입력변수가 해당 퍼지집합의 소속정도 $a_i^l \neq 0$인 것이며 인접한 두 개의 퍼지집합이 이에 속한다. 따라서 n개의 입력변수에 대하여 각 규칙에서 추론된 점화도가 $w^l \neq 0$인 퍼지규칙의 수는 2^n이다. 결국 n개의 입력변수에 대한 한번의 추론 과정에서 y에 영향을 미치는 퍼지규칙의 수는 2^n개이며 결과적으로 결론부 추론과정에서 필요로 하는 곱 연산의 수는 $2 \cdot 2^n$개가된다. Q.E.D.

가정 1을 만족하도록 퍼지시스템의 전제부 퍼지집합을 설정하고 *max*·*min*알고리즘 2를 적용함으로써 한번의 추론 과정에서 필요로 하는 실수 곱 연산의 수는 $NAP \leq (n+2^{n+1})$개가된다. 이것은 알고리즘 1을 적용한 경우의 $\sum_{i=1}^{n} k_i + 2 \cdot M$보다 작은 수가된다. 이러한 결론부의 추론과정은 알고리즘 3과 같다.

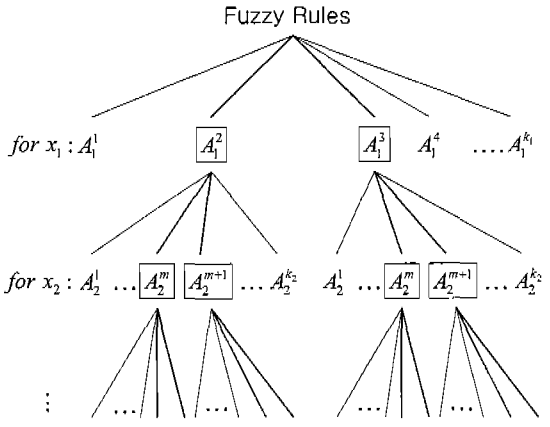

그림 5. 점화되는 퍼지규칙 트리
Fig. 5 The tree of the fired fuzzy rules

알고리즘 3 : 고속 퍼지추론을 위한 개선된 알고리즘- 결론부
Algorithm 3 : Modified algorithm for fast fuzzy inference - consequent part

```
/* consequent part */
1. total_w ← 0.0
2. FOR k=0 THRU M-1
     a. IF (w[k] ≠ 0.0) THEN
          1. total_w = total_w + w[k]
3. y ← 0.0
4. FOR k=0 THRU M-1
     a. IF (w[k] ≠ 0.0 ) THEN
          1. xi[k] ← w[k]/total_w
          2. y = y + (xi[k] * theta[k])
```

여기서 xi[k]와 theta[k]는 각각 (2)의 $\xi(\underline{x})$와 θ를 나타내며 y는 $y(\underline{x})$를, total_w은 $\sum_{l=1}^{M} \min(a_1^l, a_2^l, \cdots, a_n^l)$를 나타낸다. 결국 한번의 추론과정에서 수행되는 곱 연산의 수는 2^{n+1} 개이다.

Remark 1. FFIA의 결론부에 추론과정을 ST와 TST/TSKT-퍼지시스템에 대하여 적용할 수 있음을 보이기 위하여 각각의 타입을 고찰한다.

퍼지시스템의 R^l번째 규칙은 다음과 같다.

$$R^l: \textit{If } x_1 \textit{ is } F_1^l \text{ and } \cdots \text{ and } x_n \textit{ is } F_n^l, \textit{ then Expression} \tag{11}$$

여기서 Expression은 각 타입에 따라 다음과 같이 표현되어진다.

$$\text{(ST)}: y = Z_1^l x_1 + Z_2^l x_2 + \cdots + Z_3^l x_3 \tag{12}$$

$$\text{(TST/TSKT)}: \dot{\underline{x}}(t) = A^l \underline{x}(t) + B^l \underline{u}(t) \tag{13}$$

각 퍼지시스템을 (2), (4)와 같이 표현하면

$$y(\underline{x}) = \frac{\sum_{l=1}^{M} w^l (Z_1^l x_1 + \cdots + Z_n^l x_n)}{\sum_{l=1}^{M} w^l} \tag{14}$$

$$\dot{\underline{x}}(t) = \frac{\sum_{l=1}^{M} w^l (A^l \underline{x}(t) + B^l \underline{u}(t))}{\sum_{l=1}^{M} w^l} \tag{15}$$

이며 여기에 FFIA를 적용하면 (14), (15)의 $\sum_{l=1}^{M}$을 (10)과 같이 $\sum_{l=1, w^l \neq 0}^{M}$으로 대치하면 된다. 즉, 알고리즘 3과 같이 점화도가 유효한지를 먼저 판단하도록 하는 조건 분기문(2-a, 4-a)을 삽입하는 것은 퍼지시스템의 형태와는 관계없이 적용될 수 있다.

4. 곱 추론의 경우에 대한 고찰

본 장에서는 앞에서 *max*·*product*추론에 대한 제안된 FFI 알고리즘의 곱 연산 감소효과에 대하여 고찰하며

*max · min*과 *max · product*추론에 대한 기존 알고리즘(Algorithm 1)과 FFI 알고리즘(Algorithm 2)에 대하여 비교한다.

먼저 곱 추론에 대한 퍼지시스템 식은 다음과 같이 나타낼 수 있다.

$$y(\underline{x}) = \frac{\sum_{l=1}^{M} \bar{y}^{l} \cdot \prod_{i=1}^{n} \alpha_i^{l}}{\sum_{l=1}^{M} \prod_{i=1}^{n} \alpha_i^{l}} \tag{16}$$

따라서 알고리즘 1에 의한 곱 연산의 수는

$$\begin{aligned} NAP\_A &\le \sum_{i=1}^{n} k_i + M \cdot (n-1) \\ NAP\_C &= 2 \cdot M \end{aligned} \tag{17}$$

이며 결국

$$NAP \le \sum_{i=1}^{n} k_i + M(n+1) \tag{18}$$

이다. 그러나 본 논문에서 제안한 FFI 알고리즘(Algorithm 2)에 의한 곱 연산의 수는

$$\begin{aligned} NAP\_A &\le (n + 2^{n}(n-1)) \\ NAP\_C &= 2^{n+1} \end{aligned} \tag{19}$$

이며 결국

$$NAP \le n + 2^{n} \cdot (n+1) \tag{20}$$

이다. 결과적으로 *max · min*나 *max · product* 추론 모두에 대하여 기존 방식의 곱 연산 수는 퍼지시스템의 입력변수와 그 입력변수 각각에 설정한 퍼지라벨의 수에 의해 결정되어지나 본 논문에서 제안된 FFI 알고리즘을 적용하므로서 곱 연산의 수가 단지 입력변수의 수, n에 의해 결정되어진다. 이렇게 곱의 수가 줄어든다는 것을 비교하기 위해 $k = k_1 = k_2 = \cdots = k_n$에 대하여 기존 알고리즘과 FFI 알고리즘을 비교하면 그림 6, 7과 같다.

Remark 2. 퍼지시스템의 비선형 함수에 대한 근사성질을 이용하는 응용에서 여러 가지 학습법칙을 이용하여 퍼지시스템의 파라미터벡터 θ를 적응시킨다. 이것은 $M \times 1$벡터이며 따라서 M의 배수만큼의 곱 연산이 추가된다. 그러나 FFI 알고리즘을 적용하면 이러한 파라미터 적응(parameter update)단계에서도 2^n 배수만큼으로 곱 연산을 줄일 수 있다.

5. 결 론

본 논문에서는 기존의 퍼지추론 알고리즘에 대하여 분석하고 퍼지추론 과정에서 발생되는 실수 곱 연산의 수를 줄이기 위한 퍼지시스템 설계방식으로서 퍼지규칙의 전제부에 삼각형 퍼지집합을 설정하고 추론과정을 수정한 고속 퍼지추론 알고리즘을 제안하였다. 이 알고리즘은 입력상태공간을 구분하고 인접한 삼각형 퍼지집합의 소속정도가 보수관계에 있다는 성질을 이용하며 또한 무게중심법에 의한 비퍼지화 단계에서 전체의 규칙에 대한 계산을 할 필요가 없다는 성질을 이용하였다. 그러기 위해 제안된 알고리즘은 추론과정에서 각 입력변수 값에 대한 퍼지화와 비퍼지화를 수행하기 전에 구간을 나누어 처리하는 방식을 선택하였다. 이 고속 퍼지추론 알고리즘을 적용함으로써 한번의 퍼지추론과정에서 필요로 하는 곱 연산의 수를 각 입력변수상에 설정된 퍼지라벨의 수와는 무관하고 단지 퍼지시스템의 입력변수 n에 의해 결정되어지도록 할 수 있다. 따라서 라벨의 수가 증가하여도 실제 필요로하는 곱 연산의 수를 줄일 수 있으며 이것은 결국 메모리의 사용은 그대로이나 추론과정에서 계산되어지는 퍼지규칙의 수를 줄이는 효과를 얻는다. 이것은 다변수 시스템이나 보다 정밀한 퍼지시스템의 응용을 위한 많은 수의 라벨 수가 필요한 경우 이용되어질 수 있는 알고리즘이다. 앞으로의 연구 계획은 가우시안 소속함수를 이용한 퍼지시스템에 대한 고속 퍼지추론 알고리즘의 연구가 계속되어져야 하겠다.

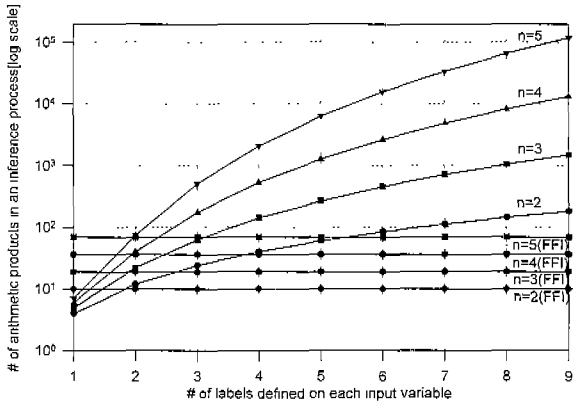

그림 6. *max · min*추론에 대한 기존 알고리즘과 FFI 알고리즘의 곱 연산 수 비교

Fig. 6. Comparison of conventional algorithm with FFI algorithm for *max · min*

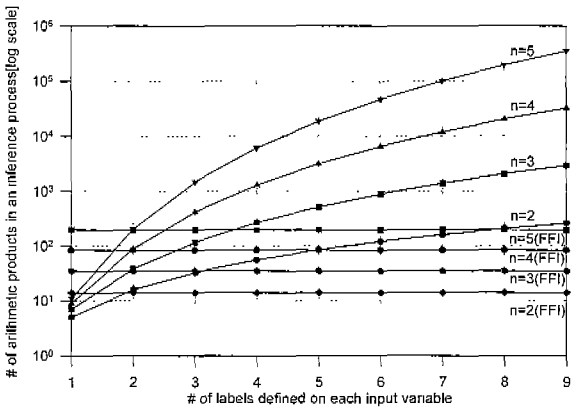

그림 7. *max · product*추론에 대한 기존 알고리즘과 FFI 알고리즘의 곱 연산 수 비교

Fig. 7. Comparison of conventional algorithm with FFI algorithm for *max · product*

참 고 문 헌

[1] C. C. Lee, "Fuzzy logic in control system : Fuzzy logic controller-Part I, II," *IEEE Trans.* on Systems,

Man, and Cybernetics, vol. SMC-20, no. 2, pp. 404-435, 1990.
[2] H. J. Zimmermann, "Fuzzy set theory and its applications," Kluwer Academic Publishers, 1991.
[3] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans.* Syst., Man, Cybern., vol. 15, pp. 116-132, Jan. 1985.
[4] L. X. Wang, "Adaptive fuzzy systems and control : design and stability analysis," Prentice Hall, 1994.
[5] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least squares learning,"*IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 807-814, 1992.
[6] L. X. Wang, "Stable adaptive control of nonlinear system," *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 2, 1993
[7] B. K. Yoo and W. C. Ham, Adaptive fuzzy sliding mode control of nonlinear system, IEEE Trans. on Fuzzy Systems, vol. 6, no. 2, pp. 315-321, 1998.
[8] B. K. Yoo and W. C. Ham, Adaptive control of robot manipulator using fuzzy compensator, *IEEE Trans. on Fuzzy Systems*, vol. 8, no. 2, pp. 186-199, 2000.
[9] L. T. Koczy and K. Hirota, "Size reduction by interpolation in fuzzy rule base," *IEEE Trans. System, Man, and Cybernetics-Part B:Cybernetics*, vol. 27, no. 1, 1997.
[10] Y. Yam, P. Baranyi and C. T. Yang, "Reduction of fuzzy rule base via singular value decomposition," *IEEE Trans. Fuzzy Systems*, vol. 7, no. 2, 1999.
[11] S. Galichet and L. Foulloy, "Size reduction in fuzzy rulebases," Proc. of *IEEE International Conference on Systems*, Man, and Cybernetics, California, USA, vol 3, pp. 2107-2112, 1998.
[12] G. Castellano and A. M. Fanelli, "A neuro-fuzzy model reduction strategy," Proc. of International Conf. on Neural Networks(ICNN'97), Texas, USA, vol. 1, 1997.
[13] J. Yen and L. Wang, "Simplifying fuzzy rule-based models using orthogonal transformation methods," *IEEE Trans. Systems*, Man, and Cybernetics-Part B:Cybernetics, vol. 29, no. 1, 1999.

저 자 소 개

유병국 (Byung Kook Yoo)
1992년 : 전북대학교 전자공학과(공학사)
1995년 : 전북대학교 대학원 전자공학과(공학석사)
1999년 : 전북대학교 대학원 전자공학과(공학박사)
현재 : 한려대학교 멀티미디어정보통신공학과 전임강사

관심분야 : 퍼지제어, 적응제어, 로봇제어, 가변구조제어

Phone : 061-760-1174
Fax : 061-761-6709
E-mail : bkyoo@hlu.hanlyo.ac.kr