대한기계학회논문집(A) 제 20 권 제 10 호, pp. 3135~3141, 1996

〈논 문〉

본 연구는 1995년도 계명대학교 비사연구기금으로 이루어졌음.

CMAC의 S-to-M 변환을 위한 알고리즘

권 성 규*

(1996년 1월 27일 접수)

An Algorithm for S-to-M Mapping in CMAC

Sunggyu Kwon

Key Words: Training(훈련), Table Look-Up(표순람), Mapping(변환), CMAC(Cerebellar Model Articulation Controller)

Abstract

In order to develop an efficient algorithm for S-to-M mapping in CMAC, characteristics of CMAC mappings is studied and conceptual mapping procedure is physically described. Then, careful observations on the mapping procedure and experience reveal a simple algorithm of the S-to-M mapping. The algorithm is described and compared with other procedures for S-to-M mapping. It is found very efficient in terms of computational operations and processing time.

기호설명

S_i : i번째 입력변수
S : 입력벡터
N : 입력공간의 차원(입력벡터를 구성하는 입력변수의 개수)
r_i : i번째 입력변수의 해상도
K : 한 입력변수에 대한 중간변수의 개수
Si_n : Si의 n번째 입력변수 위치값
${}^iM_K$: i번째 입력변수에 대한 K번째 중간변수축
${}^iM_{K1}$: ${}^iM_K$의 1번째 마디
${}^iN_K$: ${}^iM_K$를 분할한 마디의 개수

1. 서 론

Albus, J.S.는 1975년 CMAC(cerebellar model articulation controller)을 제안하면서, 로봇 팔의 제어에 CMAC을 이용하였다.[1] 그 이후 컴퓨터비전(computer vision)[2]이나 공정자동화 시스템 개발에 이용된 예도 있지만, CMAC이 이용된 대부분의 연구는 매니퓰레이터의 제어에 관한 것들이었다.[3~5] CMAC의 응용분야가 넓어지고 다양한 연구 결과가 발표되면서,[6~10] CMAC 훈련의 문제점을 인식하게 되고,[11] CMAC의 훈련을 위한 방법들이 제안되었다.[12,13] 최근에는, CMAC의 변환 알고리즘, 훈련 및 여러 매개변수들이 CMAC의 성능에 미치는 영향 등이 여러 연구들에서 보고되고 있다.[14,15]

CMAC을 이용하는 제어에서는, 제어계의 동특성을 분석하여 그 수학적 모델을 나타내는 복잡한 수식을 푸는 방법이 아니라, 제어기가 담당할 제어함수를 미리 기억표(memory table)에 심어 두었다가, 실제의 작업상황에서는 그 제어 상태가 기억표의 어느 곳에 해당하는지를 계산하여 그곳에 기억된 제어함수 값들의 산술합을 제어기의 출력값으로 내어놓는다. 그렇기 때문에, CMAC을 이용하는 제어방법에서는, CMAC이 그 기억표에 제어함수를 저장할 수 있게 하는 훈련을 어떻게 할 것인지가 중요한 문제중의 하나이다. 물론, 훈련상태나 제어상태에 따라 제어 함수값을 어디에 저장하고

*회원, 계명대학교 기계설계학과

어디서 불러내는지는 CMAC에서 일련의 변환(mapping) 과정에 의해 정의되어 있다. 그 일련의 변환 과정은, 연속적인 입력변수를 적절한 해상도로 불연속화하고, 입력벡터를 한 무리의 CMAC 중간변수벡터로 변환한다. 결국 연속적인 변수를 불연속화된 변수로 또 중간변수벡터로 변환하는 것이다.

훈련뿐 아니라 제어기로서 작동할 때에도, CMAC의 입력정보나 출력정보는 항상 CMAC의 정의에 따른 일련의 변환과정을 거치게 된다. 비록 CMAC이 Look-Up Table 방식에 의한 제어기라고 하나, 변환과정에서 정보의 흐름이 지체되거나 오차가 발생한다면, 훈련이나 제어시간의 엄청난 지연을 넘어, CMAC이 잘못 설계될 수 도 있다.

이 연구에서는, CMAC의 변환과정을 이해하고, 그 과정을 물리적으로 기술하여, CMAC 변환과정을 관찰한다. 그리고, 변환과정중 첫번째 단계인 S-to-M 변환의 단순한 변환규칙을 발견하여, 그 규칙에 대한 알고리즘을 제시하고, CMAC에서의 입·출력 흐름의 가속화를 통한 CMAC의 훈련 및 제어성능 향상을 목표로 한다.

2. CMAC에서의 변환

CMAC은 세 단계의 연속된 변환(Figs. 1~3)으로 정의된다. 이 논문에서는, 변환을 위한 계산 알고리즘을 설명하는데 도움이 되도록 변환과정을 물리적으로 묘사하기 위해 노력하였다.

먼저 S→M 변환(SM변환), N개의 연속적인 입력변수로 구성된 입력벡터(S)는 CMAC에서 각 입력변수에 해당하는 N개의 입력연수축(S_i)들로 구성되는 N차원 입력공간(receptive fields,[10] 벡터공간, S)에서 정의된다. $S=(S_1, S_2, \cdots, S_N)$, 각 입력변수축은 CMAC을 설계할 때 적당한 해상도(resolution), r_i에 따라 불연속화(不連續化)되기 때문에, 입력변수축은 입력변수값 범위, 즉 입력변수축의 크기를 해상도로 나눈 수만큼의 마디(knot)[14]로 분할된다. 그에 따라 입력공간은 유한한 수의 입력점들로 구성된다.

CMAC에서는 각 입력변수가 K개의 중간변수로 변환되므로, 각 입력변수축은 K개의 중간변수축(quantizing function)[1]으로 변환된다. Fig. 1은 2차원 입력공간이 $K=4$로 SM변환되는 개략도이다. 이 K개의 중간변수축들은 K개의 변환 층(層, level)으로 구분된다고 간주한다. 각 중간변수축도 불연속화되는데, 중간변수축의 해상도(quantizing interval)는 r_i의 K배로 정의된다. 같은 입력변수축에 대한 K개의 중간변수축들은 서로 입력변수축의 해상도만큼 전위(offset)되어 있다. 그래서, 해상도에 따라, 같은 입력변수축에 대한 K개의 중간변수축들이 분할된 마디의 개수는 보통 서로 다르다.

$$
\begin{aligned}
M = &((^1m_1,\ ^1m_2,\ \cdots,\ ^1m_K),\\
&(^2m_1,\ ^2m_2,\ \cdots,\ ^2m_K),\\
&\cdots,\\
&(^Nm_1,\ ^Nm_2,\ \cdots,\ ^Nm_K))\\
S \rightarrow M &: S_i \rightarrow (^im_1,\ ^im_2,\ \cdots,\ ^im_K)
\end{aligned}
$$

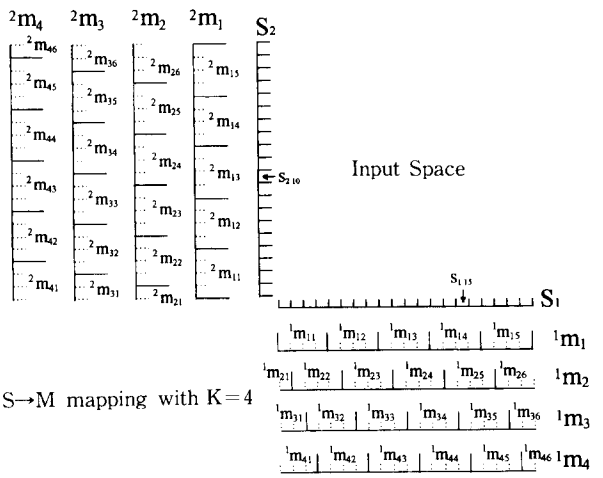

Fig. 1 S→M mapping with K=4 of a two-dimensional input space

여기서, S_i는 i번째 입력변수축, ${}^{i}m_K$는 S_i에 대한 K번째 중간변수축.

다음은, M→A 변환(MA변환)인데, A 공간은 바로 기억공간이다. 전 단계의 변환에서 정해진, 각 입력변수에 대한 K개의 중간변수축들 중에서, 변환 층이 같은 중간변수축들에 의해, 입력공간과 같은 차원의 기억공간이 구축된다. 그러므로, K개의 변환 층에 의해 구축된 K개의 기억공간이 있다. Fig. 2는 Fig. 1의 AM변환을 보여준다. 기억공간의 차원은 입력벡터의 차원과 같고, 크기는 각 층별 중간변수축의 크기에 의해 결정된다. 기억공간을 구축하는 각 축의 크기는 중간변수축이 분할된 마디의 개수와 같다. $A=(A_1, A_2, \cdots, A_K)$. Fig. 2에서는 $K=4$이므로 네 개의 2차원 기억공간이 있다.

$$M \rightarrow A : ({}^{1}m_j, {}^{2}m_j, \cdots, {}^{N}m_j)$$
$$\rightarrow A_j({}^{1}N_j, {}^{2}N_j, \cdots, {}^{N}N_j)$$

여기서, ${}^{N}m_j$는 N번째 입력변수축에 대한 j번째 중간변수축이고, ${}^{N}N_j$는 ${}^{N}m_j$를 분할한 마디의 개수이다.

마지막 변환단계인, A→p 변환(Ap변환)에서는,

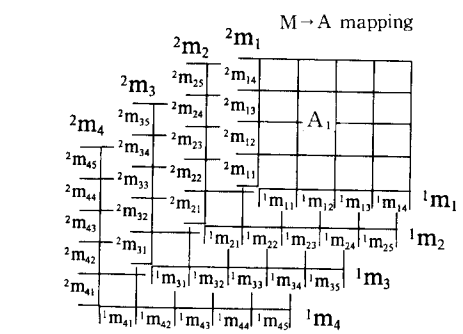

Fig. 2 M→A mapping with K=4

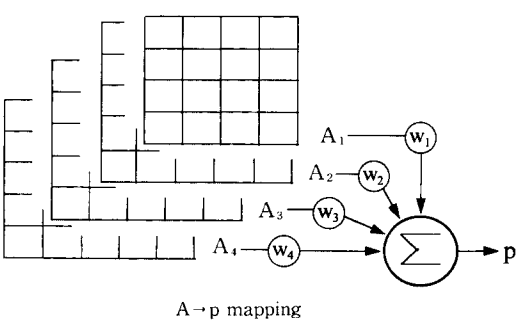

Fig. 3 A→p mapping with K=4

K개의 기억공간 각각에서 입력벡터에 해당하는 기억장소의 기억값이 복사되고, 그 값들이 더해져 제어기의 출력값이 된다. $p=w_1+w_2+\cdots+w_K$. 여기서, w_K는, K번째 기억공간인 A_K의 어느 번지에서 복사되어 나온 기억값이다. Fig. 3은 Fig. 2의 Ap변환을 보여준다. $K=4$이므로 네 개의 기억값이 더해진다.

$$A \rightarrow p : \sum_{j=1}^{K} w_j = p$$

여기서, w_j는 A_j에서 복사되어 나온 기억값이다.

3. 변환규칙

CMAC 설계에 필요한 매개변수들의 값을 결정한 뒤, 제일 먼저 입력변수축을 불연속화한다. 그러면, 한 입력변수값은 불연속화된 입력변수축에서의 위치의 의미로 변환된다. 그 변화를 변환과정에서 나타내는 수단으로써, 그 위치에 어떤 수를 지정하여, 그 수를 입력변수 위치값이라 하자. 그러면, 어떤 입력벡터는 N개의 입력변수 위치값으로 구성된다. 한 입력변수축에 대한 위치값의 개수는 그 축이 분할된 마디의 개수, 즉 입력변수값 범위가 해상도 r_i로 나누어진 값이다. Fig. 1에서 보면, 2차원 입력공간을 구성하는 두 입력변수축은 각 각 해상도 r_1과 r_2로, 스무 개씩의 마디로 분할되었다.

이제, 어떤 입력변수값(s_i)에 해당하는 입력변수 위치값을 고려해 보자. 한 입력변수축의 하한보다 크거나 같으며 그 하한에 그 축의 해상도를 더한 값보다는 작은 입력변수값들이, 분할된 첫번째 마디인 첫번째 입력변수 위치값($s_{i\ 1}$)으로 나타내진다고 하자. 그 입력변수축의 하한과 s_i의 차이를 해상도만큼 나눈 수가 첫 번째 위치값에 대한 상대적인 위치를 나타내는 수가 된다. 그 수가 n이면, s_i에 해당하는 입력변수 위치값은 $s_{i\ n}$이다.

입력변수 위치값은, SM변환에 의해, K개의 중간변수 위치값들로 변환된다. 한 입력변수 위치값이 어떻게 K개의 중간변수 위치값으로 변환되는지를 살펴보자. 첫 번째 중간변수축에 대해서는, s_i에 대한 $s_{i\ n}$를 찾을 때와 같되 단지 해상도를 $r_i \cdot K$로 하면 된다. 두 번째 중간변수축부터는 진위를 고려해야 한다. 두 번째 중간변수축은 첫 번째 축에 대해서 r_i만큼 입력변수값이 커지는 방향,

양의 방향으로 전위하므로, 첫 마디의 해상도는 입력변수축의 해상도와 같게 하고 나머지 범위에 대해서는 해상도를 $r_i \cdot K$로 하면 된다. 세 번째 중간변수축은 두 번째 중간변수축에 대해서 양의 방향으로 입력변수축의 해상도만큼 전위하므로, 첫 째 마디는 해상도가 r_i의 두 배가 된다. 이런 방법으로 나머지 중간변수축들에 대해서도 중간변수 위치값을 나타내는 마디로 분할할 수 있다.

한 중간변수축(^im_j)이 분할되는 마디의 개수(^iN_j)는, 대략 입력변수축의 크기를 $r_i \cdot K$로 나눈 수이다. 그러므로, 하나의 중간변수축에서의 위치값의 개수는 입력변수 위치값의 개수에 비해 K배 정도로 줄어들지만, 그런 중간변수축이 각 입력변수축마다 K개 있다. Fig. 2에서 첫 번째 입력변수축(S_1)에 대한 네 개의 중간변수축, $^1m_1$, $^1m_2$, $^1m_3$, $^1m_4$들은 각각 다섯 개, 여섯 개의 마디로 분할되었다. 그래서, $^1N_1=5$, $^1N_2=6$, $^1N_3=6$, $^1N_4=6$.

Fig. 1에서, 입력벡터 $s=(s_{1\ 15},\ s_{2\ 10})$는 SM변환을 거치면, $s_{1\ 15}$는 ($^1m_{14}$, $^1m_{25}$, $^1m_{35}$, $^1m_{44}$), $s_{2\ 10}$은 ($^2m_{13}$, $^2m_{24}$, $^2m_{33}$, $^2m_{43}$)으로 변환되므로, $m=(($^1m_{14}$, $^1m_{25}$, $^1m_{35}$, $^1m_{44}$), ($^2m_{13}$, $^2m_{24}$, $^2m_{33}$, $^2m_{43}$))가 된다.

MA변환에서, 각 입력변수축에 대한 K개의 중간변수축들은, 다른 입력변수축들에 대한 K개의 중간변수축들 중에서 서로 같은 층의 중간변수축들과 함께 입력공간과 같은 차원의 K개의 기억공간을 형성한다. 예로, Fig. 1에서, 세 번째 층의 중간변수축들, $^1m_3$과 $^2m_3$으로, 첫 번째 축($^1m_3$)의 크기는 6이고 두 번째 축($^2m_3$)의 크기는 6인, 2차원 배열의 기억공간 $A_3(6,\ 6)$이 형성된다. 그래서 이 기억공간의 크기는 36이다. (Fig. 2 참고) 또한, 각 중간변수 위치값은 기억공간에서의 기억 장소를 나타내는 번지의 역할을 한다. 앞에서 예를 든 $s=(s_{1\ 15},\ s_{2\ 10})$에 대해서는, $m=(($^1m_{14}$, $^1m_{25}$, $^1m_{35}$, $^1m_{44}$), ($^2m_{13}$, $^2m_{24}$, $^2m_{33}$, $^2m_{13}$))이, MA변환에 의해, 기억공간에 대한 번지 $a=(A_1($^1m_{14}$, $^2m_{13}$), $A_2($^1m_{25}$, $^2m_{24}$), $A_3($^1m_{35}$, $^2m_{33}$), $A_4($^1m_{44}$, $^2m_{43}$))로 변환된다.

Ap변환에서는, 한 기억공간에서 하나씩 선택된, K개의 값이 합해져서 출력값이 된다. 위의 예에 대해서, $a=(A_1($^1m_{14}$, $^2m_{13}$), $A_2($^1m_{25}$, $^2m_{24}$), $A_3($^1m_{35}$, $^2m_{33}$), $A_4($^1m_{44}$, $^2m_{43}$))가 $p=w_1+w_2+w_3+w_4$로 변환되는데, 여기서 $w_1=A_1($^1m_{14}$, $^2m_{13}$), $w_2=A_2($^1m_{25}$, $^2m_{24}$), $w_3=A_3($^1m_{35}$, $^2m_{33}$) 및 $w_4=A_4($^1m_{44}$, $^2m_{43}$).

4. S → M 변환 계산

지금까지는 SM변환과정을 그림으로 확인하였다. 이제는, 어떤 입력변수값이 중간변수 위치값으로 변환되는 과정, 즉 주어진 어떤 입력변수값이 각 중간변수축의 어느 마디에 해당하는지를 계산하는 문제를 고려해 보자. 이 문제를 단순하게 만드는 알고리즘을 찾아서, 구현하는 계산 공식을 설정하는 것이, 이 연구의 주제이다.

알고리즘을 찾기위해 Fig. 4의 예를 들어 보자. 이 그림에서 입력변수축, S_i는 해상도 r_i로 나뉘어 열 다섯 개의 마디로 분할되어 있다. $K=5$이므로, S_i에 대해서 다섯 개의 중간변수축 $^im_1$, $^im_2$, $^im_3$, $^im_4$, $^im_5$가 있다. 그리고, 각 중간변수축들이 분할된 마디의 수는, $^iN_1=3$, $^iN_2=4$, $^iN_3=4$, $^iN_4=4$, $^iN_5=4$. 여기서, $^iN_1=3$인 반면, 다음 것들에 대해서는 그 수가 하나 늘어난 $^iN_2=^iN_3=^iN_4=^iN_5=4$인 사실에 주목할 필요가 있다. 이것은 중간변수축이 설정되는 CMAC의 특성에 의해, 해상도가 $r_i \cdot K$인 중간변수축들이 서로 입력변수축의 해상도 r_i만큼 양의 방향으로 전위되어 있기 때문이다. 그림에서 보는 것처럼, 두 번째 층의 중간변수축 $^im_2$는 첫 번째 층의 중간변수축 $^im_1$에 대해서 입력변수축의 해상도 r_i만큼 전위(轉位)되어 있기 때문에, 마디의 개수가 하나 늘어나서, $^iN_2=4$이다. $^im_3$도 $^im_2$에 대해서 전위되어 있으나, 그 전위량이 $r_i \cdot K$에는 미치지 못하여 여전히 $^iN_3=4$이다. 나머지에 대해서도 같은 이유로 $^iN_4=^iN_5=4$. 이 그림에서, 입

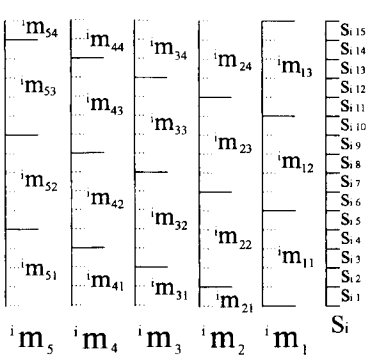

Fig. 4 An example of S → M mapping with K=5 of an input variable, S_i

력변수값 $s_{i\ 8}$은 SM변환에 의해 ($^{i}m_{12}$, $^{i}m_{23}$, $^{i}m_{33}$, $^{i}m_{42}$, $^{i}m_{52}$)로 변환된다.

입력변수값을 SM변환하기 위한 계산공식을 설정하기 위해서는, 관계되는 모든 계산에서 나머지는 무시되는, Fortran에서 integer 계산과 같은 계산이 되도록 환경을 만드는 것이 좋다.[14,16] 그러므로, 우선, 입력변수의 해상도 r_i를 1로 둘 수 있도록 계산 환경을 만들어야 한다. 어떤 입력변수값 s_i가 입력변수 위치값 $s_{i\ n}$으로 변환되었다고 가정하자, $n\geq 1$, 즉 입력변수값 범위의 하한을 포함하는 첫 번째 마디, 즉 $n=1$인 입력변수 위치값을 $s_{i\ 1}$이라 할 때, n번째 마디에 해당하는 입력변수값이 주어졌다고 가정하자. 해상도가 1이므로, 사실은, s_i와 $s_{i\ n}$가 같다.

그리고, 아무 입력변수 위치값 $s_{i\ n}$에 대한 j 번째 중간변수축에서의 중간변수 위치값 $^{i}m_{j\ l_j}$는, j 번째 중간변수축에서 l_j 번째 중간변수 위치값임을 나타낸다. 첫번째 중간변수 위치값은 $^{i}m_{1\ l_1}$, 여기서 l_1은 첫 번째 중간변수축에서 l_1 번째 중간변수 위치값임을 나타낸다. 그러면, j 번째 중간변수축에서, 입력변수의 하한값을 포함하는 입력변수축의 첫 번째 마디인, 입력변수 위치값 $s_{i\ 1}$을 변환하는 중간변수 위치값은 $^{i}m_{j1}$로 표시되어, $l_j=1$. $s_{i\ n}$에 대한 l_j를 계산해 보자. 다음부터 나오는 계산은 모두, 나머지가 무시되는 정수 계산임을 주의한다.

l_1과 l_j는 다음 두 식으로 계산할 수 있다.

(1) $l_1=1+(n-1)/K$,

(2) $l_j=[2K+\{n-(j-1)-1\}]/K$, $j\geq 2$.

여기서, n은 $s_{i\ n}$의 n이다. 이 공식으로 Fig. 3의 $n=6$에 대해서, $l_1=1+(6-1)/5=2$. $l_2=[2\cdot 5+\{6-(2-1)-1\}]/5=2$. $l_3=l_4=l_5=2$. $n=9$에 대해서는, $l_1=1+(9-1)/5=2$. $l_2=[2\cdot 5+\{9-(2-1)-1\}]/5=3$. $l_3=l_4=3$. $l_5=[2\cdot 5+\{9-(5-1)-1\}]/5=2$. 다른 n 들에 대한 것들과 함께 Table 1에 l_j 계산결과가 있다.

위의 계산결과를 잘 관찰해 보면, n, j 및 l_j 사이에 어떤 일정한 규칙이 있음을 발견할 수 있다. 즉, 각 중간변수축마다 위의 공식 (2)를 적용하지 않고, $^{i}m_1$에 대한 l_1만 (1)로 계산하고, 그 나머지 축들에 대해서는 새로운 규칙을 이용해서 l_j들을 결정할 수 있다.

일반적으로 한 중간변수 위치값 $^{i}m_{j\ l_j}$로 나타내지는 중간변수축의 마디에 대해서, 입력변수축의 어떤 입력변수 위치값에서부터 K 개의 연이은 위치값들이 그 중간변수 위치값으로 표시되는 마디로 변환된다. $^{i}m_{1\ l_1}$로 변환되는 입력변수 위치값들 중에 제일 작은 위치값의 그 마디에서의 위치를 1로 할 때, $s_{i\ n}$의 $^{i}m_{1\ l_1}$ 마디에서의 상대위치는,

(3) $l_r=n-K\cdot(l_1-1)$

로 정의하여 계산할 수 있다. $^{i}m_{1\ l_1}$의 l_1을 (1)로 계산하고, l_r을 (3)으로 계산한 뒤, 나머지 l_j들은 다음과 같이 계산할 수 있음을 발견하였다.

$$(4)\quad l_j=\begin{cases} l_1, & \text{if } l_r=1 \\ l_1+1, & \text{if } 2\leq j\leq l_r \\ l_1 & \text{if } l_r<j\leq K \end{cases}$$

Fig. 4에서, 예로, $s_{i\ 5}$에 대해서, (1)에 의해, $^{i}m_1$에 대한 $l_1=(5+5-1)/5=1$. 그래서 $s_{i\ 5}$에 대해서 $^{i}m_{1\ 1}$. 이 마디에 대한 $s_{i\ 5}$의 상대위치는, $l_1=1$이므로, $l_r=5-5\times(1-1)=5$. (4)에 의해, $l_2=l_3=l_4=l_5=2$. $s_{i\ 11}$에 대해서, (1)에서 $l_1=3$, (3)에서 $l_r=11-5\times(3-1)=1$이고, (4)에서 $l_2=l_3=l_4=l_5=3$.

Table 1 l_j of intermediate variable knot $^{i}m_{j\ l_j}$ for $s_{i\ n}$

$s_{i\ n}$	n	l_1	l_2	l_3	l_4	l_5
$s_{i\ 5}$	5	1	2	2	2	2
$s_{i\ 6}$	6	2	2	2	2	2
$s_{i\ 7}$	7	2	3	2	2	2
$s_{i\ 8}$	8	2	3	3	2	2
$s_{i\ 9}$	9	2	3	3	3	2
$s_{i\ 10}$	10	2	3	3	3	3
$s_{i\ 11}$	11	3	3	3	3	3

5. 토의 및 관찰

우선, SM변환과정을 계산하는데 있어서, 모든 입력변수의 해상도를 1로 두어 정수 계산 환경으로 만드는 것이 긴요하다. 실수 계산의 경우에는, 소수이하의 계산에서 계산과 표현의 오차가 생길 수 있다. 그로 인해 변환 계산에 오차가 생기면, CMAC에서는 틀린 번지의 기억장소에 기억값을 저장하고, 또 저장할 때와는 다른 번지에서 기억값을 복사해 내는 결과를 가져오게 된다.

SM변환과정을 단순한 알고리즘으로 계산할 수 있게 해주는 간단한 규칙은, 경험과 관찰에 의해서

발견할 수 있었으나, 분명히 CMAC이 갖는 몇 가지 특징에 기인한 것임을 알 수 있다.[17] 우선, 중간변수축의 중간변수 위치값으로 표현되는 대부분의 마디들이 입력변수축의 해상도보다 정확히 K배 크게 CMAC이 설계된다는 사실이고, 두 번째로는, 각 층별 중간변수축들이 입력변수축의 해상도만큼 서로 전위되어 있는 사실이다.

$K=5$인 Fig. 4에서, 보통의 한 중간변수 위치값들은 어떤 범위의 다섯 입력변수 위치값들에 의해서 그 역할을 할 수 있다. 그러나, 그 전위 때문에, ${}^{i}m_{2}$의 ${}^{i}m_{21}$은 첫 번째 입력변수 위치값 $s_{i\ 1}$에 의해서만 그 역할을 한다. 즉, 그 해상도가 입력변수 해상도와 같다고 볼 수 있다. 또한, ${}^{i}m_{3}$은 ${}^{i}m_{31}$은 입력변수 위치값 $s_{i\ 1}$과 $s_{i\ 2}$에 의해서만 역할을 하여, 그 해상도가 $2r_i$이다. 이렇게, 각 중간변수축들의 첫 중간변수 위치값으로 표현되는 마디들의 해상도가 다른 것은, ${}^{i}m_{41}$과 ${}^{i}m_{51}$뿐 아니라, ${}^{i}m_{54}$, ${}^{i}m_{44}$, ${}^{i}m_{34}$ 및 ${}^{i}m_{24}$에 대해서도 마찬가지이다. 그래서, $s_{i\ 2}$에 대해서 ${}^{i}m_{1}$에서는 l_j가 1이고, ${}^{i}m_{2}$에서는 l_j가 2였다가, ${}^{i}m_{3}$에서는 l_j가 다시 1이 되어, ${}^{i}m_{14}$, ${}^{i}m_{5}$에서는 l_j가 1이다. 이 관찰은 전위가 4절에서 논의된 양의 방향뿐 아니라 음의 방향에 대해서도 마찬가지이다.

이제 공식 (1), (3), 및 (4)에 의한 SM변환 계산의 효과를 계산시간의 관점에서 비교해 보자. 한 입력변수에 대해서, (1)로 첫 번째 중간변수축에 대한 l_1을 계산하고, (2)로 나머지 $K-1$ 개의 중간변수축에 대해서 l_j를 계산한다면, 나눗셈을 $K-1$번 해야 한다. 그러나, 나머지 $K-1$ 개의 중간변수축에 대해서 (3)과 (4)를 사용하면, 곱셈 한 번과 조건 판별을 위한 계산이 소요된다. 곱셈과 나눗셈의 계산 시간을 D로 같다고 보고 (4)의 조건 판별에 걸리는 시간을 d로 보면, (2)에 의한 계산시간은

(5) $D\cdot(K-1)$.

(3)과 (4)에 의해서는

(6) $D+d$.

(5)와 (6)의 차이는, 하나의 입력변수에 대해서, $D\cdot(K-2)-d$이므로, 일반적으로 입력 공간의 차원이 N이라 할 때,

(7) $N[D\cdot(K-2)-d]$.

D와 d의 차이로 인한 (7)의 크기는, N과 K가 커질수록 커진다. CMAC 문제에서, 훈련 및 출력 시간이 N의 크기에 기하학적으로 비례해서 커지는 사실을 감안하면, N의 크기에 의한 (7)의 효과는 훨씬 더 큼을 알 수 있다.

6. 결 론

CMAC 설계에서, SM변환 계산을 위해 모든 입력변수 해상도를 1로 하여 정수 계산 환경을 만들어, 이 논문에서 기술한 입력변수 위치값과 중간변수 위치값의 개념을 활용한 (3) 및 (4)를 이용하면, CMAC의 변환과정이 단순해지고, 변환에 소요되는 계산시간은 CMAC 문제의 N과 K에 따라 (7)만큼 줄어든다.

이 논문에서 제시한 알고리즘은, 응용문제 이전의 CMAC 변환 과정에 대한 것이므로, 이론적으로는, 어떤 CMAC의 설계에서도 이용될 수 있다. 그러나 실제의 현장 문제에 구현하기 위해서는 응용 문제에 맞도록 시스템을 구축하여야 할 것이다. 제대로 구현이 된다면, CMAC 제어기 자체만의 계산 시간 단축과 변환 과정의 단순화 효과가 달성될 수 있을 것으로 기대한다.

본 논문에서 제시한 알고리즘은 CMAC의 정의에 기인한 특성을 관찰하여 찾아낸 것이므로, 이 알고리즘을 적용하는데 있어서, 안정성과 같은, CMAC의 제어기로서의 본질적 특성에는 아무 영향을 미치지 않는다. 그러나, 훈련을 거쳐 CMAC의 기억표에 심어질 제어함수가 충분히 부드럽지(smooth) 않으면[18] 어떤 훈련기법도 CMAC의 안정성[19]을 확보할 수 없을 것이다.

참고문헌

(1) Albus, J.S., 1975, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control, Transactions of the ASME*, Series G. Vol. 97, No. 3, pp. 220~227.

(2) Manglevhedakar, S., 1986, *An Adaptive Hierarchical Model for Computer Vision*, MSME Thesis, Department of Mechanical Engineering, Louisiana State University.

(3) Nam, Kwanghee, 1989, "A Computed Torque Method Incorporating an Iterative Learning

Scheme," *Proceedings of '89 KACC,* pp. 1097~1112.

(4) Kim, Hyo-Gyu and Oh, Se-Young, 1994, "Stable Neural Controller DEsign Based on Composite Adaptation," *Proceedings 1994 IEEE International Conference on Robotics and Automation,* pp. 3174~3179.

(5) Nam, K. and Kuc., T., 1988, "An Application of the CMAC to Robot Control," *Proceedings of '88 KACC,* pp. 999~1005.

(6) Park, Y.S., Jung, W.T., Kwon, S., Yoon, J.S., Lee, J.S. and Kim, H.S., 1993, "Neural Network Application to Camera-Robot Kinematic Conversion," *Proceedings on '93 ICAR, the 6th International Conference on Advanced Robotics,* Nov. 1993, Tokyo, Japan, pp. 329~332.

(7) Saito, F. and Fukuda, T., 1994, "Learning Architecture for Real Robotic Systems-Extension of Connectionist Q-Learning for Continuous Robot Control Domain," *Proceedings 1994 IEEE International Conference on Robotics and Automation,* May, 1994, pp. 27~32.

(8) Janabi-Sharifi, E., Fakhry, H. H. H. and Wilson, W. J., 1994, "Integration of a Robust Trajectory Planner with a Feedforward Neural Controller for Robotic Manipulators," *Proceedings 1994 IEEE International Conference on Robotics and Automation,* May, 1994, pp. 3192~3197.

(9) Larsen, G.A., Cetinkunt, S. and Donmez, A., 1995, "CMAC Neural Network Control for High Precision Motion Conbtrol in the Presence of Large Friction," *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control,* September 1995, Vol. 117, pp. 415~420.

(10) Shelton, R.O. and Peterson, J.K., 1992, "Controlling a truck with an adaptive critic CMAC design," *SIMULATION,* Vol. 58, No. 5, May, 1992, pp. 319~326.

(11) 권성규, 1992, "CMAC을 위한 이웃간訓鍊方法," 대한기계학회논문집, 제16권, 제10호 pp. 1816~1823.

(12) Thompson, D.E. and Kwon, S., 1995, "Neighborhood Sequential and Random Training Techniques for CMAC," *IEEE Transactions on Neural Networks,* Vol. 6, No. 1, pp. 196~202.

(13) Miller, W.T. III., Hewes, R.P., Glanz, F.H. and Kraft, L.G. III., 1990, "Real-Time Dynamic Control of An Industrial Manipulator Using A Neural-Network-Based Learning Controller," *IEEE Transactions on Robotics and Automation,* Vol. 6, No. 1, pp. 1~9.

(14) Brown, M. and Harris, C., 1994, *Neurofuzzy Adaptive Modelling and Control,* Prentice Hall, 1994.

(15) Miller, W.T. III, 1994, "Real-Time Neural Network Control of a Biped Walking Robot," *IEEE Control Systems,* February, 1994, pp. 41~48.

(16) Brodie, R. and Forth, Inc., 1987, *Starting FORTH,* 2nd Edition, Prentice-Hall, Inc., p. 102.

(17) Albus, J.S., 1979, "A Model of the Brain for Robot Control, Part 2: A Neurological Model," *BYTE,* Vol. 4, No. 7, pp. 54~95.

(18) Albus, J.S., 1975, "Data Storage in the Cerebellar Model Articulation Controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control, Transactions of the ASME,* Series G. Vol. 97, No. 3, pp. 228~233.

(19) Parks, P.C. and Militzer, J., 1989, "Convergence Properties of Associative Memory Storage for Learning Control Systems," *Automation and Remote Control, a translation of Automatika i Telemekhanika,* Vol. 50, No. 2, February, Part 2, pp. 254~286.