Proceedings of the 11th
KACC, October 1996

연관사상 메모리를 이용한 로봇 머니퓰레이터의 학습제어기 설계

Design of a Robot Learning Controller Using Associative Mapping Memory

정 재욱*, 국 태용*, 이 택종*

*성균관대학교 전자공학과 (Tel: 0331-290-7202; Fax: 0331-290-7191; E-mail: *skechy@contr.skku.ac.kr*)

Abstracts In this paper, two specially designed associative mapping memories, called Associative Mapping Elements(AME) and Multiple-Digit Overlapping AME(MDO-AME), are presented for learning of nonlinear functions including kinematics and dynamics of robot manipulators.
The proposed associative mapping memories consist of associative mapping rules(AMR) and weight update rules(WUR) which guarantee generalization and specialization of input-output relationship of learned nonlinear functions. Two simulation results, one for supervised learning and the other for unsupervised learning, are given to demonstrate the effectiveness of the proposed associative mapping memories.

Keywords MDOAME, AME, CMAC, LUT, learning control

1. 서론

학습제어기를 설계하는 방식중 look-up table(LUT) 방식은 제어에 필요한 신호인 학습제어입력을 저장하여 반복되는 학습시점에서 꺼내어(recall phase) 사용하고 학습규칙에 따라 새로운 신호로 대체하여(training phase) 저장하므로써 학습횟수가 계속 진행됨에 따라 공정의 출력이 원하는 값으로 수렴해 가도록 하는 방법이다[2][4]. 이 방식은 메모리에 저장하는 기법이 매우 단순하여 실제 적용이 매우 편리한 장점이 있지만 제어신호의 샘플링 속도가 증가함에 따라 필요한 메모리의 양이 과도하게 증가하고, 특정 명령이나 공정만을 대상으로 반복적인 학습을 통하여 제어목적을 이룸으로써 소위 학습의 일반화 성질을 얻기가 어렵다.

한편 연관사상 신경망의 일종인 CMAC[1][5]은 기타 여러 신경망 보다 계산이 단순하고 빨라서 제어기 설계시 현재의 마이크로 프로세서로도 구현인 가능할 뿐만 아니라 학습규칙 또한 매우 단순하여 적용하기가 용이하다. 이것은 기존의 LUT 방식을 hashing 기법[3]이라는 특별한 입력 addressing을 통하여 메모리 양를 현저히 줄일 수 있는 기법이고 학습규칙이 기존의 LMS(least mean square)규칙을 사용하여 다른 신경망에서 나타나는 local minima 현상을 피할 수 있는 장점을 가진다. 그러나 CMAC은 입력 addressing에서 나타나는 설계변수인 일반화 인자와 양자화 레벨은 적용하는 공정에 따라 일의적으로 결정되는 것이 아니라 시행착오를 통하여 정해야하며 아울러 hashing 기법에 의한 메모리 충돌문제와 복잡성은 이 방식의 응용을 어렵게 하는 요인이 된다.

그리하여 이 논문에서는 기존의 LUT 방식에서 나타나는 메모리 증가 문제와 CMAC의 입력 addressing 문제를 동시에 해결할 수 있는 AME[4]를 도입하여 제어기를 설계하고자 한다.
그러나 이 논문의 AME는 기존의 기법에서 나타나는 메모리 충돌문제와 학습의 일반화 문제를 개선시키기 위해 입력 addressing을 변경하였고, 메모리 수정 규칙을 종래의 반복횟수에 따른 규칙뿐만 아니라 시간에 따른 규칙을 함께 적용시켜 수렴 속도를 향상시키고자 한다. 또한 개선된 구조와 학습 규칙의 성능검증을 위해 컴퓨터 시뮬레이션을 통하여 광범위한 입력공간을 갖는 비선형 함수의 근사화문제와 로봇제어에서 원하는 관절신호를 발생시키기 위한 inverse kinematics문제를 다루기로 한다. 우선 앞의 예에서는 원하는 입출력쌍이 존재하는 경우 제안한 연관사상 메모리 구조의 입력신호 패턴분류 성능을 알 수 있고, 다음으로 로봇의 inverse kinematics문제에서는 원하는 출력을 학습구조에 직접사용하지 못하는 경우, 즉, 자율학습(unsupervised learning)에 대한 제어성능을 살펴볼 수 있다.

2. Multiple-Digit Overlapping AME

CMAC이나 AME 같은 연관사상 메모리를 사용하는 목적은 비선형 함수의 입출력 관계를 표현하는 막대한 양의 데이터를 연상(recall phase)하고 저장(training phase)하는데 사용되는 메모리의 양을 줄이는데 있다. 메모리에 저장할 때 소위 "가중치(weight)"라 불리는 훈련된 메모리의 내용은 학습한 함수의 입출력 변환을 나타내는 특별한 양상(specialization)을 띄게되며, 학습한 함수의 연속성이 보장된다면 메모리 속에 분산 저장된 가중치는 학습한 함수와 유사한 함수들을 근사적으로 표현할 수 있는 일반화 성질(generalization)을 갖게된다[4].
이를 위해 AME는 연관사상 규칙(associative mapping rule : AMR)과 가중치 수정 규칙(weight update rule:WUR)을 이용하여 분산 저장된 가중치를 입력에 따라 지정하고 그 내용을 수정해 간다.

2.1 AMR

먼저 n개의 입력변수 $x_i(t) \in R\ (1 \le i \le n)$와 출력 $y(t) \in R$ 를 시간 $t \in [0, t_f]$에서 정의하고 이들이 이산 시간열 $1 \le k \le n_f$에서 이산 데이터 열로 주어진다고 가정하자. 그러면 입력 데이터는 다음과 같은 L개의 자리를 갖는 부동 소수점 형태로 표현할 수 있다.

$$\begin{aligned} x_1(k) &= s_a^k\ \overline{a_1}^k\ \overline{a_2}^k \cdots\ \overline{a_L}^k, \\ x_2(k) &= s_b^k\ \overline{b_1}^k\ \overline{b_2}^k \cdots\ \overline{b_L}^k, \\ &\vdots \\ x_n(k) &= s_c^k\ \overline{c_1}^k\ \overline{c_2}^k \cdots\ \overline{c_L}^k, \end{aligned}$$

여기서 $\{ s_a^k, s_b^k, \cdots, s_c^k \}$는 $\{x_1(k), x_2(k), \cdots, x_n(k)\}$의 부호를 나타낸다. 위와 같이 표현된 입력변수를 AMR에 적용하기 위해서 각 자리를 M진수 계로 변환하여 모두 양의 값을 가지도록 천이시켜서 해당 메모리에 addressing할 수 있는 번지로 표현한다. 즉, $1 \le r \le L$의 경우

$$\begin{aligned} a_r^k &= s_a^k\ \overline{a_r}^k + (M-1), \\ b_r^k &= s_b^k\ \overline{b_r}^k + (M-1), \\ &\vdots \\ c_r^k &= s_c^k\ \overline{c_r}^k + (M-1). \end{aligned} \tag{1}$$

식(1)과 같이 표현되는 입력변수의 값으로 해당되는 메모리로의 사상과 이를 통한 출력은 다음의 식으로 기술할 수 있다.

$$\begin{aligned} f^j &= [f_1^j f_2^j \cdots f_L^j]^T \\ w_r^j(k) &= f_r^j(a_r^k, b_r^k, \cdots, c_r^j) \\ y^j(k) &= \sum_{r=1}^{L} w_r^j(k) \end{aligned} \tag{2}$$

여기서 f는 (1)식으로 나타난 addressing 번지로 해당 메모리로 사상하는 관계를 나타내며, w는 해당 메모리의 값(가중치)을 나타내고 y는 입력번지에 의해 사상된 모든 가중치의 합으로 계산된 출력을 나타낸다. 아울러 j는 반복횟수에 대한 번호이며 k는 샘플열의 순서를 가리킨다. 그림[1]은 두 변수에 대한 식(1)과 (2)의 예를 제시한 것이다. 그림에서 알 수 있듯이 두 변수의 경우 하나의 AME 블록에는 $(2M-1)^2$의 메모리 장소가 존재하며, 인접한 데이터는 그렇지 못한 데이터에 비해 많은 메모리 장소를 공유할 수 있음을 알 수 있다. 결국 이러한 사상특성은 AME가 학습이 진행될 때 일반화 특성을 갖는 중요한 요인으로 작용한다.

2.2 MDO-AMR

앞 절에서 기술한 AMR은 CMAC이 가지는 입력의 사상규칙에 비해 매우 단순하며, 특히 메모리에 대한 번지지정 방법이 입력 데이터의 개수와 설계할 공정에서 필요한 정밀도에 따라 쉽게 정할 수 있는 장점이 hashing 기법을 이용하는 CMAC에 비해 설계자가 이해하기 쉬운 구조를 가지고 있다. 그러나 CMAC이 가지는 메모리 충돌문제는 전술한 AMR 역시 내포하고 있는 문제로, 입력공간에서 서로 떨어진 곳에 위치한 데이터가 같은 메모리로 사상될 때 그 출력결과가 심각한 영향을 받을 수 있다. 그리하여 이 논문에서는 기존의 AMR을 확장하여 그 규칙의 단순성은 유지하면서 이러한 메모리 충돌문제를 완화시킬 수 있도록 사상규칙을 새롭게 구성하였다. 즉, 기존의 방법에서는 하나의 메모리를 사상하기위해 각 변수에서 같은 위치(r)에 있는 값으로 addressing을 구성하였지만 제안한 방법은 인접한 자리를 몇 개 묶어 중첩시키므로써 메모리 addressing을 구성하였다. 이 방법을 사용하면 서로 떨어진 곳에 있는 데이터가 같은 메모리 장소를 할당하여도 중첩된 번지 구조로 인하여 충돌번지의 상위나 하위번지에서 이들값을 확산하여 저장할 수 있으므로 출력에 미치는 영향을 줄일 수 있을 뿐만 아니라 각 변수에 대해 두자리 이상으로 구성된 번지는 한자리 만으로 구성된 기존의 방법에 비해 충돌 가능성 자체가 훨씬 줄어들게 된다. 이를 식으로 나타내면

$$\begin{aligned} w_r^j(k) = f_r^j(\ & a_r^k, a_{r+1}^k, \cdots, a_{r+v}^k, \\ & b_r^k, b_{r+1}^k, \cdots, b_{r+v}^k, \\ & \vdots \\ & c_r^k, c_{r+1}^k, \cdots, c_{r+v}^k,). \end{aligned} \tag{3}$$

여기서 v는 중첩된 자리수를 나타낸다. 그림[2]는 자리수 중첩(Multiple-Digit Overlapping:MDO) AMR을 두자리 중첩의 경우에 적용시킨 예를 보여준다.

2.3 학습규칙: WUR

AME의 가중치 수정규칙(weight update rule:WUR)은 분산된 가중치로 저장된 비선형 함수의 입출력에 대한 연속성을 이용하여 구성하는데, 만약 $\{x_1(k), x_2(k), \cdots, x_n(k)\}$가 $\{x_1(k-1), x_2(k-1), \cdots, x_n(k-1)\}$의 다음에 나타나는 입력이라면 이들간에 많은 유효자리는 서로 같은 값을 가지게 된다. 그리하여 많은 경우 다음과 같은 영이 아닌 수 $\{m_a^k, m_b^k, \cdots, m_c^k\}$가 존재한다.

$$\begin{aligned} a_r^k &= a_r^{k-1}, \quad 1 \le r \le m_a^k \\ b_r^k &= b_r^{k-1}, \quad 1 \le r \le m_b^k \\ &\vdots \\ c_r^k &= c_r^{k-1}, \quad 1 \le r \le m_c^k. \end{aligned}$$

모든 입력변수에 대한 중첩자리를

$$m_d^k = \min\{m_a^k, m_b^k, \cdots, m_c^k\} \tag{4}$$

으로 정하고, k=1인 경우는 $m_d^1 = 0$으로 하여 가중치 수정 신호 $e^j(k)$로 가중치를 학습시킨다. 영이 아닌 $m_d^k (> 0)$가 존재하면 출력 $y^j(k)$와 $y^j(k-1)$은 m_d^k에 해당하는 메모리를 서로 공유하게 되어 메모리의 절감과 함께 수정규칙의 계산을 그 만큼 줄일 수 있는 이점이 있다. 이러한 개념을 바탕으로 수정 규칙은 다음과 같다.

$m_d^k = 0$ 또는 L인 경우,

(5) $w_r^{(j+1)}(k) = w_r^j(k) + \eta_{1r} e^j(k) \quad 1 \le r \le L,$

여기서 $\sum_{r=1}^{L} \eta_{1r} = 1.$

$m_d^k > 0$ 인 경우,

(6)

$$w_{r(j+1)}(k) = \begin{cases} w_r^{(j+1)}(k-1) & \text{for } 1 \le r \le m_d^k \\ w_r^j(k) + \eta_{2r}\left(e^j(k) - \sum_{q=1}^{m_d^k}\{w_q^{(j+1)}(k-1) - w_q^j(k-1)\}\right) & \text{for } m_d^k + 1 \le r \le L \end{cases}$$

여기서 $\sum_{r=m_d^k+1}^{L} \eta_{2r} = 1$ 이고, 초기값은 $w_r^1(k) = 0$ 으로 둔다. 식(5)와 (6)에서 나타나는 분포 파라미터인 η_{1r}, η_{2r}은 다음과 같이 데이터의 분포 특성에 따라 달리 선택할 수 있다.

(7)

$$\eta_{1r} \equiv \begin{cases} \frac{1}{L} & \text{(equal distribution)} \\ \frac{2(L-r+1)}{L(L+1)} & \text{(reversely weighted distribution)} \end{cases}$$

$$\eta_{2r} \equiv \begin{cases} \frac{1}{L-m_d^k} & \text{(equal distribution)} \\ \frac{2(L-r+1)}{(L-m_d^k)(L-m_d^k+1)} & \text{(reversely weighted distribution)} \end{cases}$$

이상의 수정규칙은 반복횟수에만 적용되므로 시간적으로 변하는 샘플열에서 발생하는 오차신호에 대해서는 아무런 수정이 발생하지 않게 된다. 그리하여 제안한 MDO-AME에서는 입력신호가 연속임을 가정한다면 (5)식으로 주어진 수정규칙을 샘플열 k에 대해서도 적용하여 시간적인 오차보정이 함께 일어나도록 하여 전체적인 수렴시간을 줄일 수 있도록 하였다.

(8) $w_r^j(k+1) = w_r^j(k) + \eta e^j(k)$

여기서 η는 입력의 분포 특성에 따라 (7)식에서 나타낸 분포에 따른 정의중 하나를 선택하여 사용할 수 있다.

3. 시뮬레이션

제안한 MDO-AME의 성능 검증을 위해 우선 광범위한 입력 공간을 가지는 비선형 함수의 근사화에 대해 적용하였고, 둘째로는 로봇제어에서 필수적인 inverse kinematics(역기구학) 문제에 적용하였다. 앞의 예제는 원하는 입출력쌍이 존재하며 또한 수정신호인 오차신호 발생에 직접 적용이 가능한 경우이고 두 번째 예는 제어기 설계시 종종 발생하는 원하는 입출력쌍을 쉽게 또는 직접 얻을 수 없는 경우에 대한 것이다.

3.1 비선형 함수의 근사화

근사화 해야할 함수는 두 개의 입력을 가진 함수로 다음과 같다.

$$g(k) = \exp(\sin(x) - \cos(y)) + \alpha$$

이 함수의 모양은 그림[3]과 같고 30회 학습후의 결과는 그림[4]에 제시하였고, 30회때의 결과는 MSE가 0.0031정도로 만족할 만한 결과를 보였다.

3.2 역기구학 문제

전술한 바와 같이 역기구학 문제는 원하는 입력에 대해 출력을 직접 알 수 없다는 난점이 있다. 이것은 로봇제어시 Cartesian 좌표계에서 기술된 원하는 제어명령을 역기구학 방정식을 통하여 관절좌표계로 변환하여 로봇에 명령을 내려야 하고, 로봇의 궤적을 다시 Cartesian 좌표계로 바꾸어 원했던 결과와의 차를 기초로 제어기를 설계해야한다. 그러나 역기구학 방정식은 로봇의 자유도가 높아짐에 따라 가능한 해의 개수가 이에따라 증가하므로 쉽게 정하지 못하는 난점이 있다. 아울러 역기구학 방정식을 풀기 위해서는 로봇의 inverse Jacobian이 역행렬을 가져야만 하는 조건을 만족해야 한다. 그리하여 이 문제의 해결을 위해 우리는 기존의 학습제어기 설계에서 널리 이용하는 다음과 같은 수정신호를 WUR에 적용하였다.

(9) $e_i^j(k) = \beta J_i^{jT}(k) \Gamma^{-1} \tilde{d}^j(k)$

여기서 β는 학습률, J_i^j는 i번째 관절에서 j번째 학습의 forward Jacobian, Γ는 Jacobian의 과도한 영향을 줄이기 위한 정규화 상수행렬, $\tilde{d}^j$는 j번째 학습시의 원하는 궤적과 실제궤적의 차를 각각 나타낸다. 이러한 오차신호를 이용하여 2-link 머니퓰레이터에게 지름이 1m인 원궤적을 그리는 명령을 내려 그 결과를 제시하였다. 그림[5]는 이 문제를 풀기 위한 블록다이어그램을 나타내었고, 원하는 원궤적에 대한 500번 학습후의 결과는 그림[6]에 보였다. 이때의 MSE 오차는 평균 0.00001정도로 나타났고 실제로 300번 정도에서 x축의 오차는 0.0004, y축은 0.0005 정도의 결과를 얻을 수 있었다.

4. 결론

이 논문에서는 새로운 입력 사상기법을 도입한 MDO-AME를 제안하였다. 이 방식은 기존의 LUT 기법이 가지는 샘플링 속도에 따른 메모리 증가문제를 이와 무관하게 바꾸어 원하는 정밀도에만 그 크기가 달라지도록 하였으며, 이와 유사한 CMAC 방식이 갖는 사상규칙의 복잡성을 입력값에 따른 사상방식으로 바꾸어 제어기 설계가 용이하도록 만들었다. 그리고 컴퓨터 시뮬레

이션을 통하여 그 결과가 우수함을 검증하였다.

이후과제로는 제안한 기법을 로봇의 동력학 제어문제에 확장적 용하는데 있고 이에대한 연구가 현재 진행 중에 있다.

참고 문헌

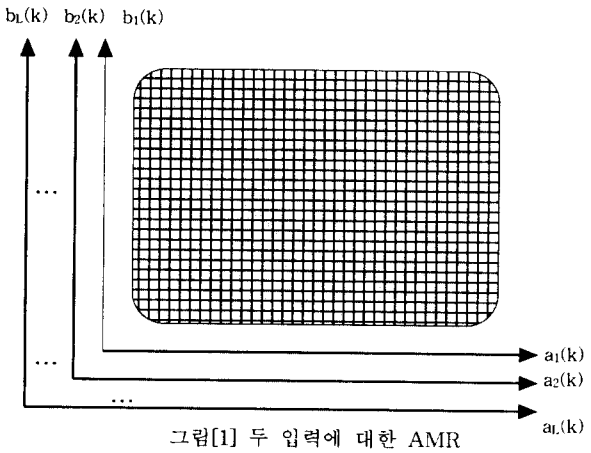
[1] J. Albus, Brain, Behavior, and Robotics, N.H. Byte Books, 1981

[2] S. Arimoto et al., "Bettering operation of robots by learning," J. Robotic Syst., pp. 123-140, 1984.

[3] D. Knuth, "The Art of Computer Programming," Sorting and Searching, vol.3, Addison Wesley.

[4] Tae-yong Kuc, A Learning Control Theory with Application to Robotic Systems, Ph.D. Thesis, Dept. of EE Eng., POSTECH, 1993.

[5] W. T. Miller et al., "Application of a general learning algorithm to the control of robotic manipulators," Int. J. Robotics Res., vol.6, no.2, Summer, 1987.



그림[1] 두 입력에 대한 AMR

Fig.[1] AMR with 2 inputs

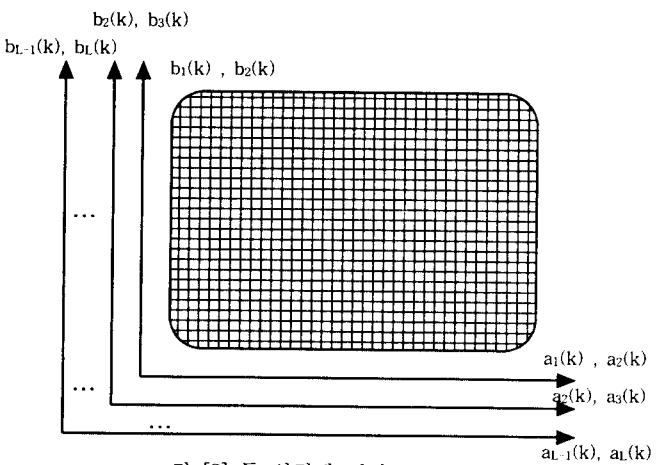


그림 [2] 두 입력에 대한 MDO-AMR

Fig. [2] MDO-AMR with 2 inputs

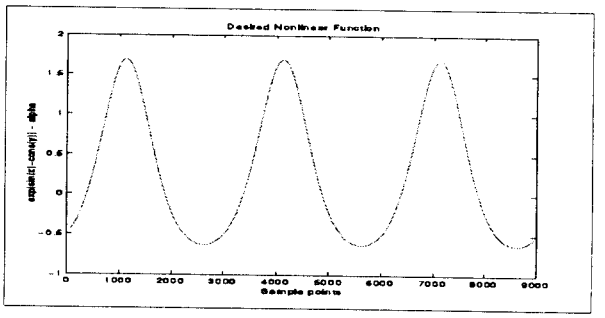


그림 [3] 함수 근사화를 위한 비선형 함수

Fig. [3] Desired Nonlinear Function for Function Approximation

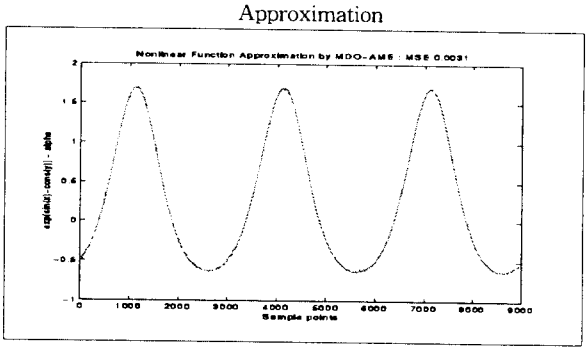


그림 [4] 30회 학습 후의 근사화 결과

Fig. [4] Actual Function Approximated after 30th Iterations

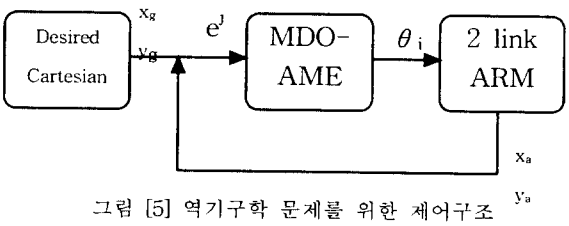


그림 [5] 역기구학 문제를 위한 제어구조

Fig. [5] Learning Control Scheme for Inverse Kinematic Problem

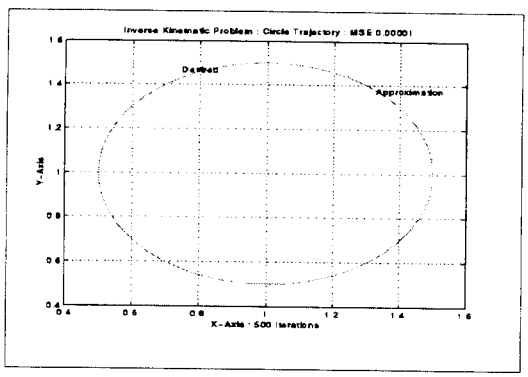


그림 [6] 역기구학 제어시 사용된 기준궤적과 근사궤적

Fig. [6] Desired Trajectory and Approximation Trajectory for Inverse Kinematic Control