1816 大韓機械學會論文集 第16卷 第10號, pp. 1816~1823, 1992.

〈論 文〉

CMAC을 위한 이웃간訓鍊 方法

권 성 규*

(1991년 12월 12일 접수)

Neighborhood Sequential Training Technique for CMAC

Sunggyu Kwon

Key Words: Manipulator Control(머니퓰레이터 제어), Training(훈련), Learning(학습), CMAC (Cerebellar Model Articulation Controller), Table Look-Up(表順覽)

Abstract

In order to develop general CMAC training technique applicable to any CMAC, characteristics of CMAC learning algorithm and training problems of CMAC are studied. Neighborhood Sequential Training technique which is general and free from CMAC learning interference is proposed. The technique is used to generate mathematical functions and found to be effective.

기 호 설 명

S_i : i번째 入力變數(input variable)
$\vec{S}$: 入力狀態벡터(input state vector) $\vec{S}=(S_1,S_2,\cdots,S_I)$
s_{in} : i번째 입력변수의 n번째 훈련 입력변수 값
$\vec{s}_i$: i번째 훈련 입력점
$\vec{A}_i$: i번째 훈련 입력점에 의해 지정되는 記憶細胞들의 집합
r_i : i번째 入力變數 軸(input variable axis)의 解像度(resolution)
${}^iC_j$: i번째 입력변수의 j번째 量子化層(quantizing level)
K : 입력변수에 대한 양자화층의 갯수
N_{ij} : ${}^iC_j$의 中間變數(intermediate variable)의 개수

1. 서 론

1972년 J.S. Albus는 動物體의 運動을 담당하는 中樞 神經系의 조직과 기능의 원리[1]에 바탕을 둔 頭腦 모델[2]을 개발하였으며, 이 모델을 발전시켜 1975년에 CMAC(cerebellar model articulation controller)이라는 머니퓰레이터 制御器(manipulator controller)를 제안하였다.[3]

CMAC을 제안하면서 Albus는 로봇 팔의 제어에 CMAC을 이용하는 제어방법을 제안하였다.[3] 그 이후 컴퓨터비젼(computer vision)[16]이나 工程 自動化 시스템 개발[5]에 이용된 예도 있지만, CMAC이 이용된 대부분의 연구는 로봇 머니퓰레이터의 제어에 관한 것들이었다.[6~9] 1987년 Miller는 작업대 위에 놓여 있는 3차원 공간의 특정한 물체에 대해서 로봇손을 일정한 거리에 위치시키기 위해서 로봇의 손목에 裝着된 비디오 카메라를 CMAC을 이용하여 제어했을 뿐 아니라,[10] 로봇 손이 움직이는 콘베어 벨트 위에 놓여있는 어떤 물체를 일정한 거리를 유지하면서 추적할 수 있도록 CMAC을 이용한 제어계를 구성하였다.[4] 또한 Miller는 2축 로봇 머니퓰레이터의 동적제어를 위한 제어계에서 로봇 관절의 토오크를 담당하는 제어기로서 CMAC을 이용하였으며, 1990년에는 상당한 속도로 움직이는 산업용 5축 로봇의 동적 제어를 담당하는 CMAC 제어기를 구성하였다.[11]

*정회원, 한국원자력연구소 원격장치기술실

CMAC을 이용하는 제어방법을 종래의 제어방법들과는 전혀 다르다. 즉, 제어계의 動特性을 분석하여 그 수학적 모델을 나타내는 복잡한 수식을 푸는 방법이 아니라, 제어기가 담당할 제어함수를 미리 記憶表(memory table)에 심어 두었다가 실제의 작업상황에서는, 그 상황에 상응하는 적절한 기억세포들에 기억된 제어함수 값의 算術合을 제어기의 출력값으로 사용한다.

그렇기 때문에 CMAC을 이용하는 제어방법에서는 CMAC의 기억표에 제어함수를 어떻게 저장할 것인가 하는 것이 무엇보다 중요한 문제이다. 이것은 CMAC의 측면에서는 제어함수의 학습이고 CMAC 설계자의 측면에서는 CMAC이 제어함수를 학습하도록 훈련하는 일이다.

CMAC의 훈련을 위한 몇몇 방법들이 제안되었지만[10,12,17], 대부분이 특정한 제어문제만을 위한 훈련방법들이었으며, 그런 훈련방법을 구현하기 위해서 CMAC을 정의하는 여러 매개변수들을 어떻게 운용할 것인지에 대해서는 명확하게 설명되지 않았다.

본 논문에서는 CMAC의 훈련에 관련된 문제점뿐만 아니라 효율적인 CMAC 훈련방법의 개발에 관한 문제를 연구하였으며, 학습간섭의 영향을 전혀 받지 않으면서 CMAC의 學習一般化(learning generalization) 특성을 살린, 일반적으로 응용될 수 있는 이웃간훈련방법을 제안하였다. 이 훈련 방법을 2변수 연속함수를 위한 2차원 CMAC의 훈련 모사에 적용하여 전체 입력점 수의 1.3% 정도의 훈련 회수로 그 연속함수의 최대 함수값 1.0에 대해 0.0025의 제곱 평균 제곱근 오차(root mean square error, 이하 RMS error라 함)를 갖는 수준의 훈련성과를 거둘 수 있다.

2. CMAC의 訓鍊

CMAC의 응용될 수 있는 제어함수는 입력변수에 의해서 정의되는 제어함수 값이 입력변수 값의 변화에 따라 부드럽게 변하는 連續函數이어야 한다. 그 값이 급격하게 변화하는 제어함수에 대해서는 CMAC이 응용될 수 없다. 그러나, 대부분의 제어함수는 그 값이 부드럽게 변하는 연속함수의 특성을 갖고 있다.[12]

CMAC이 제어계 내에서 어떤 제어함수를 담당하는 제어기가 되도록 하기 위해서는 CMAC의 기억표에 제어함수를 심어주는 훈련과정을 부여하여야 한다. 이 과정은 반복학습에 의한 훈련으로서, CMAC이 원하는 제어함수를 충분히 학습할 수 있도록 시행되어야 하는데, 일반적으로 그 반복 훈련 회수가 증가할수록 CMAC의 기억표에 저장되는 함수는 원하는 제어함수로 收斂한다.[3] Albus는 이 반복학습에 의한 CMAC 학습 알고리즘의 수렴성을 경험적으로 증명하였지만,[12] 1989년 Parks와 Militzer는 수학적으로 증명하였다.[13]

CMAC에서 어떤 制御狀態(control state)는 I개 입력변수로 이루어지는 I차원 入力空間(input Space) 내의 한 점(入力點)으로 표시되고, 그 위치는 I개의 입력변수 값으로 정해지며, 그 I개 입력변수 값을 성분으로 하여 입력상태벡터가 정의된다. 입력공간 내의 모든 입력점에 대해서 이웃(neighborhood)이라는 영역이 각 입력점 주위에 정의되는데, 한 이웃에 있는 입력점들에 대한 입력변수는 서로 비슷한 값을 갖는다. 입력변수 값은 어떤 제어상태에 대한 CMAC 출력값을 저장하고 있는 기억세포의 番地를 지정하는 指標로 사용되는데, 그 번지는 CMAC의 記憶細胞 指定 알고리즘(memory addressing algorithm)에 의해 지정된다. 이 알고리즘에 의하면, 한 입력점에 의해 지정되는 기억세포들 중의 일부는 그 입력점 이웃에 있는 다른 입력점들에 의해서도 지정된다. 이 특성에 의하면 한 제어상태 대한 CMAC의 훈련은, 그 제어상태를 표시하는 입력공간의 한 입력점 이웃에 있는 다른 입력점들에 의해 표시될 유사한 제어상태에 대해서도 일부의 훈련효과를 남긴다. 이것을 CMAC의 학습일반화라고 부른다.[6]

다른 한편으로 어떤 제어상태에 대해서 CMAC이 그 제어함수의 값을 학습하였다고 할 때, 뒤이어 주어지는 새로운 훈련이 먼저 번과 유사한 제어상태에서 주어진다면, 이미 주어진 훈련에 의해서 그 값이 조정되었던 기억세포들 중의 일부가 새로운 훈련에 의해서 그 값이 다시 조정된다. 이렇게 되면, 이미 주어졌던 훈련에 의한 학습효과가 떨어지게 된다. CMAC에서 이런 현상은 학습간섭이라 한다.[6]

학습간섭이 CMAC의 훈련에 미치는 나쁜 영향을 이해하기 위해 다음의 연속함수를 학습할 1차원 CMAC을 훈련하는 예를 들어 보자.

$$f(x)=2[\mathrm{Sin}(x)-1/2\mathrm{Sin}(2x)+1/3\mathrm{Sin}(3x)]$$

$$[\mathrm{Cos}(x)-1/2\mathrm{Cos}(2x)+1/3\mathrm{Cos}(3x)]$$

이 함수는 $-180°\leq x\leq 180°$의 범위에서 그 값이 부드럽게 변하는 연속함수이다. 이 CMAC을 훈련하기 위해 입력변수 x를 위한 $r_1=1.0$으로 하면 전체 입력점은 361개가 된다. 양자화층의 수는 $K=7$로 하고, 훈련 회기마다 선택하는 x값을 다른 훈련 회기에서 선택했던 x 값들과 상관없이 임의로 입력변수 축을 따라 선택한다고 하자. 입력변수가 x 한 개 뿐 이므로, 예를 들어, 첫 번째 훈련 입력점을 $s_1=-180$으로 하고 $s_2=-178$, $s_3=-175$, …, 하여 마지막 훈련을 $s_{150}=180$에서 한다면 $N_{train}=150$이 된다. 이렇게 훈련하면, s_2에서의 훈련으로 s_1에서 이미 그 값이 조정된 7개 기억세포 중의 일부가 그 값이 다시 조정되어, CMAC은 s_1에 대해서는 틀린 함수값을 학습하게 되는 결과를 낳게 된다.

그래서 Fig. 1에 점선으로 그려진 식(1)의 그래프와 CMAC의 기억표에 심어진 함수를 나타내는 실선으로 그려진 그래프를 비교하면 CMAC의 출력값에 큰 誤差가 있음을 알 수 있다.

이런 잘못된 훈련방법에 의한 학습간섭의 나쁜 영향은 피하고, CMAC의 학습일반화 특성을 살릴 수 있는 효율적인 훈련방법을 개발하기 위해서는 다음의 세 가지 문제점을 동시에 고려하여야 한다.

(1) CMAC을 정의하는 입력공간의 어디에서 훈련을 위한 입력점(訓鍊 入力點, training input point)을 선택할 것인가?

(2) 한 훈련 회기가 끝난 뒤 어떤 기준에서 다음 훈련 입력점을 선정하며, 그 훈련 입력점에 대한 각 입력변수 값을 어떻게 정할 것인가?

(3) 원하는 제어능력을 CMAC이 보유할 수 있기까지는 얼마나 많은 훈련과정이 필요한가?

이런 문제점들을 고려해 볼 때 가장 단순한 일반적인 CMAC의 훈련방법은, 각 훈련 회기마다 가장 큰 오차를 가진 출력값을 내놓는 제어상태에 해당하는 입력점을 입력공간 내에서 찾아 그 점에서 훈련하는 방법이다.[3] 이 방법은 간단하기는 하지만 각 훈련 회기마다 입력공간 내의 모든 입력점에 대해서 출력값을 조사해야 되기 때문에 그에 필요한 시간이 많이 소요되는 큰 단점이 있다.

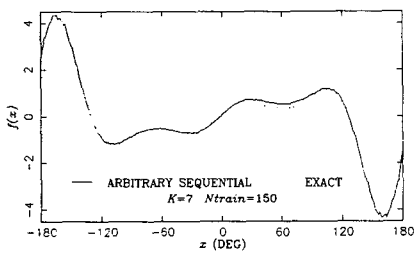

Fig. 1 $f(x)$ of Eq. (1) generated by arbitrary sequential training

또 다른 일반적인 CMAC 훈련방법으로는 훈련 입력점을 無作爲로 선택하되, 모든 훈련 입력점간의 무작위성이 입력공간 전체에서 均一하게 유지되도록 훈련 입력점을 선택하는 무작위훈련(random training)[10,14] 방법이 있다. 이 방법은 선택된 훈련 입력점간에 균일한 무작위성이 달성될 때 까지 상당히 많은 입력점에서 훈련을 해야 하기 때문에 오랜 학습 시간이 소요되는 단점이 있다. 그렇지만 훈련 회수가 증가할 수록 훈련의 성과가 향상되기 때문에 충분히 많은 입력점들에 대해서 훈련을 실시하고, 선택된 훈련 입력점간의 무작위성이 수학적으로 정의된 균일한 무작위성에 접근하게 하면 CMAC이 달성할 수 있는 최고의 훈련 성과를 거둘 수 있다.[15]

3. 이웃간訓鍊(Neighborhood Sequential Training)

이웃간훈련은 CMAC의 학습간섭 문제는 피하면서 학습일반화 특성을 살리기 위해 입력공간의 이웃 개념에 기초를 두고 있다. CMAC의 학습일반화 특성에 의하면, 한 훈련 입력점에서의 훈련은 그와 近接한 위치에 있는 여러 입력점에 의해서 정의되는 제어상태에 대해서도 부분적인 학습효과를 남기게 된다. 그러나 학습일반화의 以面에는 학습간섭이라는 逆機能도 있음은 이미 관찰한 바이다. 그래서 이웃간훈련에서는, 한 훈련 입력점의 이웃에서는 또 다른 훈련 입력점이 선택되지 않도록 하고, 각 기억세포의 값은 훈련 기간이 끝날 때까지 단 한번 조정된다.

$\bar{A}_m$을 m번째 훈련 입력점에 의해서 지정되는 기억세포들의 집합이라고 하고, $\bar{A}_n$을 n번째 훈련에 대한 것이라고 할 때, 이웃간훈련에서는 다음의 관계가 성립한다. 여기서, $m\neq n$.

$$\bar{A}_m\cap\bar{A}_n=0 \tag{2}$$

그러므로 학습간섭의 문제가 전혀 나타나지 않는다. 또한 CMAC을 구성하여, 기억표의 모든 기억세포의 값을 0으로 初期化(initialization)하고 나면, CMAC의 훈련기간 중 어떤 훈련 입력점에 대한 CMAC의 출력값도 초기값 0이기 때문에 훈련회기마다 CMAC 학습 알고리즘을 적용할 때 CMAC의 출력값을 원하는 출력값과 비교할 필요가 없다. 그러므로 그런 비교에 소요되는 시간을 벌 수 있다.

I개의 입력변수를 갖는 I차원의 CMAC에 대해서, S_i를 i번째 입력변수라 하자. 그러면 이웃간훈련을 위한 S_i 입력변수 축에서 n번째 훈련 입력변수 값 s_{in}는 다음 식에 의해 정해진다.

$$s_{in}=s_{i0}+(n-1)Kr_i \quad \begin{cases} i=1,2,\cdots,I \\ n=1,2,\cdots,N_{i1} \end{cases} \tag{3}$$

여기서 s_{i0}는 S_i 입력변수 축에서 가장 작은 입력변수 값이고, K는 각 입력변수를 위한 양자화층의 갯수, r_i는 S_i 입력변수 축의 해상도, 그리고 N_{i1}은 S_i의 첫 번째 양자화층의 중간변수의 개수이다.

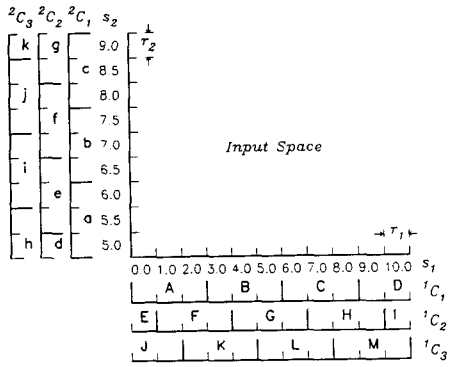

Fig. 2 Quantization of the input space of a two-dimensional CMAC with $K=3$ quantizing levels for each input variable

이웃간훈련에 있어서 필요한 훈련 회기의 수(N_{train})는 입력공간의 규모, 각 입력변수 축의 크기와 해상도 그리고 K에 의해서 결정된다. 예를 들어, 1차원 CMAC의 경우에 필요한 훈련 회기의 수는 첫 번째 양자화층의 중간변수의 개수와 동일하다. I차원의 CMAC에 대해서는,

$$N_{train}=\prod_{i=1}^{I}(N_{i1}) \tag{4}$$

이 된다.

Fig. 2에 두 입력변수 $0.0\leq S_1\leq 10.0$와 $5.0\leq S_2\leq 9.0$로 된 2차원 CMAC의 입력공간이 그려져 있다. 이 CMAC에서 $r_1=1.0$, $r_2=0.5$이고 $K=3$이다. 예를 들어, 첫 번째 입력변수 S_1에 대한 첫 번째 양자화층 $^1C_1$은 임의로 A,B,C,D라고 이름 붙힌 4개의 중간변수로 되어 있어 $N_{11}=4$이다. 이 CMAC을 이웃간훈련 방법으로 훈련하면, $N_{11}=4$ $N_{21}=3$이므로 $N_{train}=12$이다. 그리고 각 입력변수에 대한 훈련 입력변수 값은 순차적으로 다음과 같이 된다.

$i=1$, $N_{11}=4$, $s_{10}=0.0$: $s_{11}=0.0$,
$s_{12}=3.0$, $s_{13}=6.0$, $s_{14}=9.0$
$i=2$, $N_{21}=3$, $s_{20}=5.0$: $s_{21}=5.0$,
$s_{22}=6.5$, $s_{23}=8.0$

Table 1은 훈련 회기 i의 훈련 입력점 $\bar{s}_i$와 그 훈련 회기에 지정된 기억세포들의 집합 $\bar{A}_i$를 나타낸다. 예를 들어, 첫 번째 훈련 회기의 훈련 입력점은 $\bar{s}_1=(0.0,\ 5.0)$이고, CMAC의 기억세포 지정 알고리즘에 의해서 3개의 기억세포 Aa, Ed, Jh가 지정되며, CMAC 학습 알고리즘에 따라 이 훈련 회기에 기억세포의 값이 초기값 0에서 어떤 값으로 조정된다.

Fig. 3은 식(1)을 학습할 $K=10$, $r_1=1.0$을 갖는

Table 1 Training input points for the CMAC of Fig. 2

i	$\bar{s}_i$	$\bar{A}_i$	i	$\bar{s}_i$	$\bar{A}_i$
1	(0.0, 5.0)	{Aa, Ed, Jh}	7	(6.0, 5.0)	{Ca, Gd, Lh}
2	(0.0, 6.5)	{Ab, Ee, Ji}	8	(6.0, 6.5)	{Cb, Ge, Li}
3	(0.0, 8.5)	{Ac, Ef, Jj}	9	(6.0, 8.0)	{Cc, Gf, Lj}
4	(3.0, 5.0)	{Ba, Fd, Kh}	10	(9.0, 5.0)	{Da, Hd, Mh}
5	(3.0, 6.5)	{Bb, Fe, Ki}	11	(9.0, 6.5)	{Db, He, Mi}
6	(3.0, 8.0)	{Bc, Ff, Kj}	12	(9.0, 8.0)	{Dc, Hf, Mj}

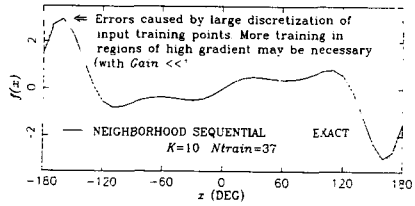

Fig. 3 $f(x)$ of Eq. (1) generated by neighborhood sequential training

1차원 CMAC을 이웃간훈련 방법으로 훈련했을 때의 결과를 나타낸다. 전체 입력점의 개수는 361개인데, 이웃간훈련에서는 전체 입력점 수의 10% 정도인 N_{train}=37이 소요되는 사실에 주의할 필요가 있다. Fig. 3을 Fig. 1과 비교해 보면 이웃간훈련 방법이 훨씬 적은 수의 훈련 회기(N_{train})로 비교가 안 될 정도의 훈련성과를 올린 것을 알 수 있다. 물론 Fig. 3의 어떤 구간에서 보이는 큰 오차는, 利得因子(gain factor) g[6]를 조정하거나 입력변수를 不連續化(discretization)할 때 그 부분에서는 해상도를 달리하는 등의 방법으로 줄일 수 있다.

Table 2는 x와 y를 입력변수로 갖는 다음과 같은 함수를 위해 서로 다른 K로 구성된 여러 CMAC을 이웃간훈련 방법으로 훈련했을 때, 각각

$$f(x,y)=\mathrm{Sin}(x)\,\mathrm{Sin}(y)\quad\begin{cases}0^\circ\le x\le 360^\circ\\ 0^\circ\le y\le 180^\circ\end{cases}\tag{5}$$

CMAC에 소요되는 기억세포의 수와 훈련 회수 그리고 정의된 2차원 입력공간 전체에 대해서 CMAC의 출력값이 갖는 RMS Error를 보여 준다. CMAC의 r_1과 r_2는 1.0이었다. CMAC을 사용하지 않고 보통의 表順覽(table look-up) 방법을 사용하면 65,341(=181×361)개의 기억세포가 필요하지만, 이 도표에서 보듯이 최대의 기억표가 요구되는 K=2로 구성되는 CMAC은 32,942개의 기억세포를 필요로 한다.

이 도표에서 M은 어떤 CMAC이 요구하는 기억세포의 개수를 나타내는데, M은 다음과 같이 결정된다.

$$M=\sum_{j=1}^{K}\left(\prod_{i=1}^{I}N_{ij}\right)\tag{6}$$

여기서, N_{ij}는 i번째 입력변수의 j번째 양자화층이 갖는 중간변수의 개수이다. CMAC에서, 일정한 해상도에 의해 불연속화된 입력변수에 대한 양자화 알고리즘에 의하면, K 값이 커지면 N_{ij}는 작아지고 결과적으로 요구되는 기억표의 크기는 작아진다.[15] Table 2에서는 또한, 작은 K 값을 갖는

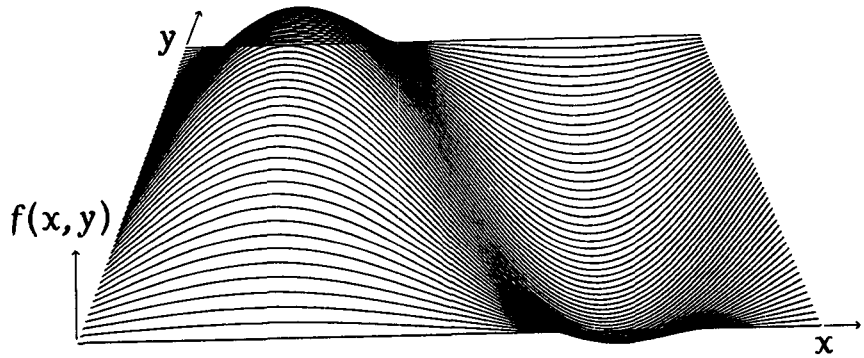

Fig. 4 $f(x,y)$ of Eq. (5) generated by CMAC with K =9 trained by neighborhood sequential training.

Table 2 Performance comparison of various CMACs with different K for Eq.(5)

K	2	4	6	9	12	15	20
M	32,942	16,744	11.346	7.749	5.952	4.875	3,800
N_{train}	16,471	4,186	1,891	861	496	325	190
RMS Error	0.0001	0.0005	0.0011	0.0025	0.0044	0.0069	0.0123

CMAC이 큰 K 값을 갖는 CMAC보다 더 많은 훈련 회기와 더 큰 기억표를 요구하지만 작은 K 값을 갖는 CMAC의 성능이 더 좋다는 사실을 확인할 수 있다.

Fig. 4는 Table 2의 CMAC 중, K=9를 갖는 CMAC을 이웃간훈련 방법으로 전체 입력점 65,341 개의 1.3% 정도인 861 개 훈련 입력점에서 훈련했을 때, 그 CMAC의 기억표에 심어진 함수를 x와 y를 두 축으로 하는 직교공간 좌표계(cartsian coordinate)의 3번째 축에 그린 그래프이다. 이 CMAC의 입력공간 전체에 대한 출력값의 RMS Error는 최대 함수값 1.0에 대해 0.0025이다.

4. 結果 및 討議

이웃간훈련 방법에 의하면, CMAC 기억표를 구성하는 전체 기억세포중에서 일부의 기억세포는 전체 훈련과정 동안에 한 번도 그 값이 조정되지 않고 초기화된 값을 그대로 가지고 있는 경우가 생길 수 있다. 예를 들어, Fig. 2에 있는 중간변수들에 의한 일부 기억세포 Id, Ie, If, Ig, Eg, Fg, Gg, Hg, Jk, Kk, Lk, Mk는 전체 훈련과정 동안에 한 번도 지정되지 않았고, 그래서 Table 1에 낱열되어 있지 않다. CMAC의 출력값이 이런 기억세포들을 포함해서 산출될 때, 그 출력값의 오차가 커지게 되고 그로 인해 CMAC의 성능이 전체적으로 저하된다.

이런 문제는 어떤 한 입력변수에 대한 여러 양자화층의 중간변수의 개수가 같지 않을 때 발생한다. 한 예로서, 식(5)를 학습할 2차원 CMAC을 훈련한다고 가정하자. Table 3은 K 값이 다른 여러 CMAC의 각 양자화층에 대한 중간변수의 개수를 보여준다. 이 도표에서 K=7인 CMAC에 대해서 N_{11}=52 N_{12}=53이고 N_{21}=26 N_{22}=27이며, K=8인 CMAC에 대해서는 N_{21}=23 N_{22}=24이다. Table 4에서 볼 수 있듯이, K=7과 K=8로 된 CMAC들의 RMS Error는 K=9를 갖는 CMAC의 RMS Error보다 더 크다. 이 현상은 Table 2에서 이미 지적되었듯이 작은 K 값을 갖는 CMAC의 성능이 큰 K 값을 갖는 CMAC보다 더 좋다는 일반적인 원칙[12]에 배치되는 것이다.

이 문제는 K 값과 입력변수의 해상도를 적당히 조정하여, 각 입력변수의 양자화층에 대한 중간변

Table 3 The number of intermediate variables for various quantizing levels

K	6		7		8		9	
J	N_{1j}	N_{2j}	N_{1j}	N_{2j}	N_{1j}	N_{2j}	N_{1j}	N_{2j}
1	61	31	52	26	46	23	41	21
2	61	31	53	27	46	24	41	21
3	61	31	53	27	46	24	41	21
4	61	31	53	27	46	24	41	21
5	61	31	52	27	46	24	41	21
6	61	31	52	27	46	23	41	21
6			52	26	46	23	41	21
8					46	23	41	21
9							41	21

Table 4 The effect of varying K on M, N_{train}, and RMS Error

K	6	7	8	9
M	11,346	9,805	8,648	7,749
N_{train}	1.891	1.352	1.058	861
RMS Error	0.0011	0.0028	0.0032	0.0025

Table 5 N_{train} for neighborhood sequential training(NST) and random training(Random) for the same RMS Error

K	4	6	10	15	20	30
NST	4,186	1,891	703	325	190	91
Random	800,000	300,000	150,000	35,000	8,000	2,000

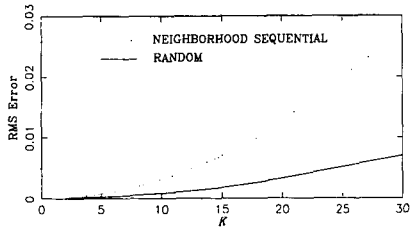

Fig. 5 RMS Error levels achieved by the neighborhood sequential training and random training

수의 개수가 같아지도록 하면 생기지 않는다. 이 조정도 물론 CMAC에 필요한 기억표의 크기가 이용 가능한 한도 내에 있을 때 가능한 것이다. 만약 그렇게 조정이 가능하다면, 모든 기억세포가 훈련 회기 동안에 꼭 한 번씩만 지정되도록 할 수 있으며, CMAC을 이웃간훈련 방법에 충실하게 훈련할 수 있을 것이다.

이 훈련방법에 있어서, 훈련 입력점들이 반드시 이미 훈련이 주어진 어떤 입력점의 이웃에 속하지만 않는다면, 각 입력변수 축을 따라서 꼭 순차적으로 훈련 입력변수 값을 정할 필요는 없지만 순차적으로 값을 정하는 것이 전자계산의 측면에서 볼 때 유리할 뿐 아니라 용이하다.

이웃간훈련과 무작위훈련을 비교해 보면 무작위훈련에서 소요되는 훈련 회수가 엄청나게 많다. Fig. 5는 식(5)를 위해 이웃간훈련과 무작위훈련에 의해서 훈련된 여러 CMAC의 훈련성과를 비교하기 위해서, 다른 K 값을 갖는 CMAC들에 대한 RMS Eror를 나타낸다. 이 그림에서 두 훈련 방법을 비교해 보면, 비록 훈련 횟수가 훨씬 많기는 하지만, 무작위훈련에 의해서 훈련된 CMAC들이 이웃간훈련에 의한 CMAC들 보다 더 나은 출력값을 산출하는 사실을 알 수 있다.

Table 5는 Fig. 5에 나타난 이웃간훈련에 의해서 훈련된 CMAC 수준의 출력값을 내기 위해서, 같은 CMAC을 무작위훈련 방법으로 훈련할 때 요구되는 훈련 회기의 수를 보여준다. 이를 보면, 이웃간훈련이 무작위훈련 보다 훨씬 적은 수의 훈련 회기로 상당한 수준의 학습능력을 갖는 CMAC을 구성할 수 있게 해주는 훈련방법임을 알 수 있다.

5. 結論

학습간섭의 문제가 전혀없으며 CMAC의 학습일반화 특성을 잘 살린 이웃간훈련 방법은 CMAC이 이용할 수 있는 어떤 제어함수의 훈련에도 이용할 수 있으며, 각 입력변수의 양자화층에 대한 중간변수의 개수가 같기만 하면, 짧은 시간내에 입력공간 전체에 대해서 일정 수준의 훈련성과를 기대할 수 있다. 2변수 연속함수를 위한 2차원 CMAC의 훈련 모사에 이 방법을 적용하여, 전체 입력점 수의 1.3% 정도의 훈련 회수로 그 연속함수의 최대 함수값 1.0에 대해 0.0025의 RMS Error를 갖는 수준의 훈련성과를 거둘 수 있었다.

CMAC으로 이룩할 수 있는 최고의 성능 한도 이내의 어떤 정해진 수준의 성능을 갖는 CMAC을 개발하기 위해서는 먼저 이웃간훈련을 적용하고 난 뒤 무작위훈련을 적용하면, 전체 훈련기간을 단축시킬 수 있을 뿐 아니라 CMAC의 출력값 오차도 최대한으로 줄일 수 있을 것이다.

참고문헌

(1) Albus, J. S., 1971, "A Theory of Cerebellar Function," Mathematical Biosciences, Vol. 10, pp. 25~61.

(2) Albus, J. S., 1972, Theoret ical and Experimental Aspects of a Cerebellar Model, Ph.D. Dissertation, Department of Biomedical Engineering, Univrsity of Maryland.

(3) 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.

(4) Miller, W. T. Ⅲ., Glanz, F. H. and Kraft, L. G. Ⅲ., 1987, "Application of a General Learning Algorithm to the Control of Robotic Manipulators," The International Journal of Robotics Research, Vol. 6, No. 2, pp. 84~98.

(5) 정재문, 김기엽, 정광조, 1990, "CMAC 메모리에 의한 연마공정자동화," '90 한국자동제어학술회의 논문집 국내학술편, pp. 186~189.

(6) 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.

(7) Camana, P.C., 1977, A Study of Physiologically Motiviated Mathematical Models for Human Postural Control, Ph.D. Dissertation, Department of Electrical Engineering, Ohio State University.

(8) Rajadhyaksha, J., 1985, A 2-D Adaptive Multilink CMAC Manipulator Simulation, MSME Thesis, Department of Mechanical Engineering, Louisiana State University.

(9) Nam, K. and Kuc, T., 1988, "An Application of the CMAC to Robot Control," '88 한국자동제어학술회의 논문집 국제학술편, pp. 999~1005.

(10) Miller, W. T. Ⅲ., 1987, "Sensor-Based Control of Robotic Manipulators Using a General Learning Algorithm," IEEE Journal of Robotics and Automation, Vol. RA-3, No. 2, pp. 157~165.

(11) 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.

(12) 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.

(13) Parks, P.C. and Militzer, J., 1989, "Convergence Properties of Associative Memory Storage for Learning Control Systems," Automation and Remote Control, A Translation of Avtomatika i Telemekhanika, Vol. 50, No. 2, Part 2, pp. 254~286.

(14) Kwon, S., 1987, Training An Adaptive Robotic CMAC Controller, MSME Thesis, Department of Mechanical Engineering, Louisiana State University.

(15) Kwon, S., 1990, An Adaptive Control System for Biological and Robotic Simulations, Ph.D. Dissertation, Department of Mechanical Engineering, Louisiana State University.

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

(17) Albus, J.S., 1979, "Mechanisms of Planning and Problem Solving in the Brain," Mathematical Biosciences, Vol. 45, pp. 247~293.