CMAC 신경회로망을 기반으로 한 학습제어
(Learning Control Based on CMAC Neural Networks)

유종준*정태진**최종수***
(J. J. Yoo, T. J. Chung** J. S. Choi**)

CMAC 신경회로망은 다차원 비선형 함수를 학습을 통하여 발생되는 많고 복잡한 데이터들을 퍼셉트론과 같이 집합시켜 메모리를 구성하고 처리하는 분야이다. 일반적으로 학습알고리즘은 소수의 반복으로써 수렴한다. 본고에서는 CMAC의 메카니즘 및 CMAC의 특성을 기술하고, CMAC의 학습가능성을 예시하였다. CMAC의 학습성능을 시험하기 위해서 3관절 로봇의 squatting 문제에 적용하였다.

I. 서 론

제어분야에 적용되는 신경회로망은 시스템의 학습능력과 성능에 주요한 영향을 미치는 한 분야이다. 최근에, 다층 퍼셉트론(multilayer perceptrons)과 같은 다층 신경회로망은 은닉층(hidden layer)에 매우 큰 unit들로 구성되어 있다면 임의의 매핑을 통해서 표현할 수 있다. 모든 네트워크 weight들은 각각의 훈련 데이터로 갱신되기 때문에, 이러한 네트워크들은 데이터 포인트 집합을 통해서 low-order polynomial을 맞추는 것과 유사한 방식으로 multi-input/multi-output 함수 데이터를 전체적 근사화(global approximation)로 구성할 수 있다. 그러나 전체 weight를 갱신하는 특성은 국부적 구조(local structures)의 상세함을 얼룩지게 하는 경향이 있고, 학습속도를 떨어뜨리며, 또한 함수근사화의 정확성이 훈련데이터의 표현순서에 민감하게 된다. 훈련 데이터의 표시순서에 대한 감도는 온라인 학습을 필요로 하는 문제에서 전체 신경회로망 구조의 유용성을 상당히 제한한다. 뇌의 sensorymotor 제어구조는 국부적으로 동조된(locally-tuned) 중복 receptive field를 갖는 뉴론을 이용하여 조직된다는 사실은 잘 알려져 있다. CMAC (Cerebellar Model Articulation Controller) 및 RBF

* 산업기술지도실 연구원
** 산업기술지도실 실장
*** 전북대학교 전기공학과 박사과정

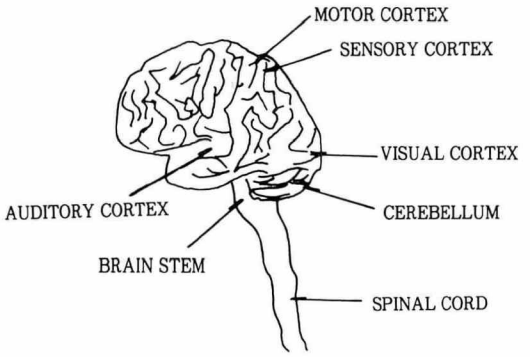
(Radial Basis Function)와 같은 신경회로망 계산기법들은 이와 동일한 조직원리를 이용하여 복잡한 비선형 함수를 표현하는 데에 매우 성공적이었다. 이러한 네트워크들은 multi-input/multi-output 함수 데이터의 구성은 국부적으로 근사화(local approximation)되어 있다. 각 포인트에 대하여 일부분의 weight 만이 입력공간에서 활동하기 때문에, 개개의 weight들은 국부적 출력 오차의 부분을 나타낸다. 국부적 접근법의 유효성을 결정하는 네트워크 파라미터들은 receptive field 함수의 위치, 폭, 수 등이다. 복잡한 비선형 함수를 표현하는 데 국부적 근사화 기법을 이용하는 주요한 장점은 전체적 접근법(global approaches)에 비해 더욱 빠른 학습능력과 멀리 떨어져 있는 영역에서 학습된 네트워크를 흐트러뜨리지 않고 입력공간의 일부에서 네트워크을 훈련시키는 능력이 있기 때문이다. CMAC는 Albus에 의해서 제안된 신경회로망 모델로서 [1][2], 기능적으로는 일반화된 look-up table 형태로 다변수 함수를 근사화시키며, 구조적으로는 복잡한 비선형 함수를 표현하는데 3-layer feedforward network 또는 look-up table 방식으로 구성한다. CMAC는 매우 빠른 학습능력뿐만 아니라 보간(interpolation)과 근사화(approximation) 특성을 갖고 있다. 따라서, 최근에 CMAC에 대한 관심이 증가하였고, 몇몇 연구자들은 다양한 문제들을 해결하기 위해서 CMAC를 적용하였다. 예를들면, Moody[3]는 chaotic time series를 예측하기 위해서 수정된 형태의 CMAC를 이용하였다. Carter등 [4]은 CMAC 네트워크의 fault tolerence를 조사하였고, Kraft와 Campagna[5]는 두 가지 적응제어기와 CMAC를 기반으로한 제어기를 비교하였다. 또한, Miller [6][7]는 vision을 이용한 로봇의 실시간 제어를 위해서 CMAC을 이용하였으며, Lin과 Kim[8]은 CMAC를 이용한 self-learning control 기법을 제안하였다.

본고는 CMAC 신경회로망에 대해서 간략히 기술하였고, CMAC의 학습능력을 시험하기 위해서 로봇 squatting 문제에 적용하였다.

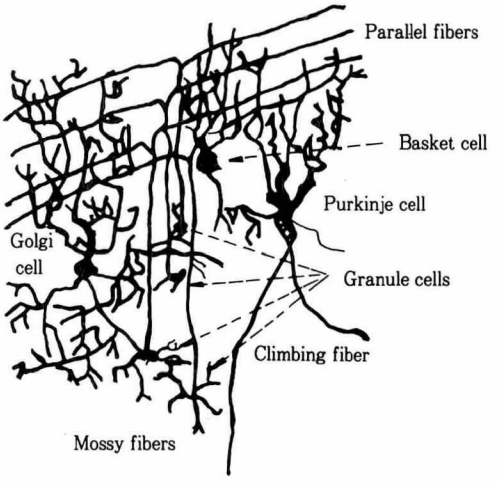
Ⅱ. CMAC 신경회로망

1. 소뇌와 CMAC

소뇌(cerebellum)는 중뇌(midbrain)에 붙어 있으며 (그림 1)에서 처럼 visual cortex 아래에 위치해 있다. 소뇌의 기능은 사람의 손과 발, 눈 등의 움직임을 빠르고 정확하게 제어하는 것이다. 소뇌에는 많은 cell과 fiber들이 서로 섞여 있다. (그림 2)에서 보는 바와 같이 Mossy fibers라 불리는 작은 뿌리들이 많이 있는데, 이것은 외부세계 또는 고등명령부(higher command center)로부터 정보를 수집한다. 이 두 개의 fiber집합(외부세계와 고등명령부에 대한 Mossy fibers)이 소뇌에 들어오는데 이것들은 사실상 구별하기가 어렵다. 이러한 Mossy fibers는 여러 개의 가지로 퍼지고 각 가지는 Granule cell이라 불리는 신경세포와 접촉하여 흥분(excitatory(+))한다. 각 Granule cell은 12개의 Mossy fibers에 1개가 접촉되고 한 개의 출력 axon을 만들어낸다. Granule cell로 부터의 각 axon들은 함께 덩어리를 만들고 서로 병렬로 동작한다. 이러한 axon 덩어리들을 Parallel fibers라 부른다. Granule cell과 Parallel fiber의 옆에는 Golgi cell이라 불리는



(그림 1) A side view of human brain showing the cerebellum.



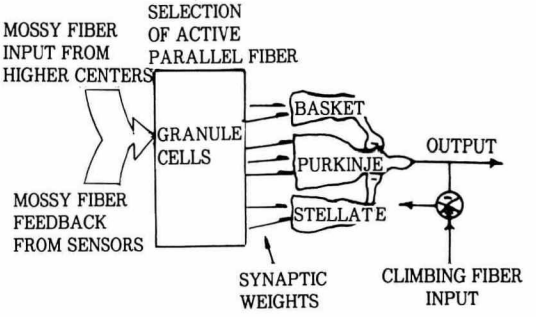
(그림 2) The principal cells and fiber systems of the cerebellar cortex.

세포와 dendrite들이 있다. Golgi cell은 Parallel fiber의 이득을 조절하고, Mossy fiber 입력의 활성레벨에 관계없이 Granule cell의 일부만이 threshold값보다 큰 값을 갖도록 Granule cell을 조절한다. 결과는 Parallel fiber의 일부만이 선택되고 출력을 만드는데 공헌하게 된다.

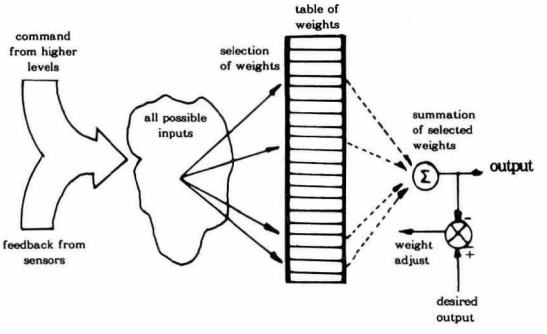
Granule cell로부터의 정보를 갖고 있는 Parallel fiber는 weight를 갖는 Purkinje cell, Basket cell, Stellate cell과 같은 세 개의 다른 세포와 접촉하여 흥분한다. Purkinje cell은 셋 중에서 가장 중요한 역할을 하며 Parallel fiber들로 부터의 입력을 더하는 기능을 수행하고 cerebellar cortex의 출력을 만들어낸다. 반면에 Basket cell과 Stellate cell들은 Parallel fiber로부터의 positive weight들과 함께 합해진 negative weight들을 Purkinje cell에 공급한다.

Cerebellar cortex에 들어가는 또 하나의 fiber는 Climbing fiber이다. Climbing fiber는 Purkinje 출력을 변경시 synaptic weight의 강도를 조절하는 역할을 한다.

(그림 3)은 소뇌에 대한 Albus의 이론적 모델이다. Albus는 이론적 모델을 CMAC라 부르는 실제적인 소뇌모델로 수정하였다. (그림 4)가 소뇌의 기능을 인공적으로 모방한 CMAC 모델이다.



(그림 3) A theoretical model of the cerebellum.



(그림 4) A CMAC model.

2. CMAC의 메카니즘

가. CMAC mapping

CMAC에서 상태에 대한 정보는 메모리에 분산적으로 저장되고 address key로서 상태변수를 이용하여 검색한다. 데이터 검색을 위해서 상태변수들은 중간변수들로 mapping되고, 출력값을 검색하기 위해서 물리적 메모리 어드레스로 mapping된다. 즉, CMAC는 다음과 같은 직렬 mapping으로 표현된다.

$$S \rightarrow M \rightarrow A \rightarrow P$$

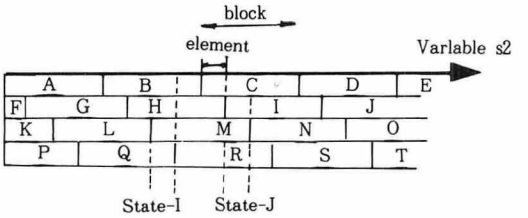
여기에서, **S**:입력벡터(input vectors)

M:중간변수(intermediate variables 또는 conceptual memory)

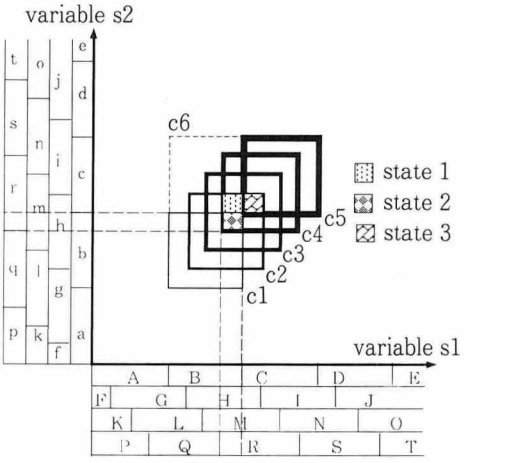
A:메모리 어드레스(physical memory address)

P:출력벡터(output vectors)이다.

첫번째 mapping(**S→M**)에 대하여, 각 상태변수 s는 (그림 5)와 같이 몇개의 block으로 분류되고, 각 block은 다시 몇개의 요소로 분류된다. (그림 5)에서는 변수에 대한 block division이 4개의 층으로 구성되어 있다. 각 층에서 각 block은 인식명(identification name)으로 배치된다. 그러므로 양자화된 상태는 각 층에 대한 block 이름들의 집합으로 특성화된다. 예를 들면, (그림 5)에서 상태 *i*와 *j*는 각각 {B, H, M, Q}와 {C, I, M, R}에 의해서 특성화된다. Mapping은 다변수인 경우로 확장할 수 있는데, (그림 6)은 변수가 2개인 경우에 대한 예이다. 사각형 c1은 변수 *s1*의 B와 변수 *s2*의 b에 의해서 정의되고, c2는 H와 h에 의해서 정의된다. 고차원인 경우의 중간변수들은 hypercube가 된다.



(그림 5) The block division of the CMAC for single input variable.



(그림 6) The block division of the CMAC for two input variables.

두번째, mapping은 중간변수(intermediate variable)들이 메모리 어드레스로 mapping되는데(**M→A**), 이 mapping 과정에서 hash coding이 이용된다. Hash coding은 큰 메모리 어드레스 공간을 보다 적은 메모리 공간으로 압축하기 위한 메모리 어드레스 기법으로, hash coding을 사용함으로써 메모리를 효율적으로 이용할 수 있다.

마지막으로 상태에 대한 메모리 어드레스의 모든 내용은 출력을 만들기 위해 합해지는데 이 과정이 **A→P** mapping이다. 결과적으로 한 상태를 포함하는 모든 hypercube들은 상태의 출력을 계산하는데 공헌하게 된다. 각 hypercube는 여러 개의 상태가 공유하게 된다. 즉, (그림 6)에서 state 2는

hypercube c1, c2, c3 및 c4를 포함한다. 이와 같이 메모리를 공유하는 것은 CMAC의 일반화(generalization) 특성이며, hypercube의 수는 block 수로 결정되기 때문에 일반화의 정도를 조절할 수 있다. 위에서 설명한 CMAC의 정보검색은 수학적으로 다음과 같이 기술할 수 있다.

s(k)를 시간 k에서의 상태, $c_j(s(k))$를 상태 s(k)에 포함된 j번째 hypercube, N_c를 상태에 포함된 hypercube의 수, $ID_{c_j}(s(k))$를 $c_j(s(k))$의 ID(identification) 수라 하자. 벡터 U(s(k))는 다음과 같이 정의한다.

$$U(s(k))=[u_1(s(k)),\ u_2(s(k)),\cdots u_i(s(k)),\ \cdots u_i(s)k))]^{\tau} \quad (1)$$

여기에서

$$u_i(s(k))=\begin{cases}1, \text{ if } i \in \{ADDR(ID_{c_j}(s(k)))\,;1\le j\le N_c\\ 0, \text{ otherwise}\end{cases} \quad (2)$$

이고, P_m은 메모리의 크기이다.

벡터 M은 식(3)으로 표시되며 메모리의 모든 내용을 나타낸다하고

$$M=[m_1,\ m_2,\cdots,\ m_{pm}]^T \quad (3)$$

상태 s(k)에 대하여 저장된 정보를 INFORM(s(k))라 하면, INFORM(s(k))는 다음과 같이 계산된다.

$$INFORM(s(k))=M\cdot U(s(k)) \quad (4)$$

한 개의 상태에 대한 정보는 그 상태에 포함된 hypercube에 할당된 여러 개의 메모리로부터 검색된다.

Hash coding으로 인한 메모리의 충돌을 피하기 위해서는 필요한 메모리를 추정해야 한다. 모든 변수들이 동일한 수의 block과 element로 나누어진다고 가정하고, 한 block내의 element 수를 N_e, 한 변수에 대한 block의 수를 N_b, 변수의 수를 N_v라하면, 중간변수를 나타내는 메모리(conceptual memory)의 크기, 즉 양자화된 입력 상태의 수 N_{ts}는 다음과 같다.

$$N_{ts}=(N_e\times N_b)^N \quad (5)$$

각 hypercube는 물리적 메모리(physical memory) 할당을 해야하므로 필요한 물리적 메모리 P_m은

$$P_m\cong N_e\times(N_b)^N \quad (6)$$

이다. 식(6)의 P_m은 $N_v>1$에 대하여 식(5)의 N_{ts}보다 작아야 한다. 즉, 필요로 하는 물리적 메모리의 크기는 전체 상태들의 수(conceptual memory)보다 작아야 한다.

이론적으로 만일 물리적 메모리가 P_m보다 작다면 메모리 충돌은 피할 수가 없다. 더우기 필요한 메모리를 감소시키기 위해서 **M→A** mapping을 이용하는데, 많은 실제 시스팀들은 입력상태중 일부분만을 이용하므로, P_m보다 작은 크기의 메모리도 때로는 충분하다.

나. Hash coding

CMAC 메모리 할당방법에서 가장 중요한 기능중의 하나는 hash coding방법이다. Hash함수 h(k)는 hypercube k에 대한 어드레스로써 계산되고 이용된다. h(k)는 P_m보다 크지 않아야 한다. Hash coding 기법은 여러가지가 있으나 random number generator 방법이 많이 이용된다. 이 방법은 식(7)과 같이 seed로써 key를 갖는 random number generator를 이용한다.

$$h(k)=RAND(k)\cdot P_m \quad (7)$$

여기에서 RAND(k)의 범위는 [0.0, 1.0]이다.

다. CMAC의 학습

CMAC에서 training data가 주어지면, 학습은 모

든 패턴에 대하여 목표값과 출력값간의 오차를 최소한으로 줄이는 것이다. i번째 목표값과 출력값을 각각 T_i, O_i라 하면, 오차 E는 다음과 같다.

$$E=(T_i-O_i)^2 \tag{8}$$

여기에서 O_i는 식(3)의 INFORM(•)이다.

메모리는 LMS 규칙을 이용하여 다음과 같이 갱신된다.

$$\triangle M=\frac{\alpha(T_i-M \cdot U_i)U_i}{N_e} \tag{9}$$

여기에서 $\alpha(\alpha>0)$는 학습비(learning rate)이다.

CMAC 학습은 양자화 기법(quantization technique)을 이용한다. 동일한 CMAC element 내의 모든 상태는 동일한 상태로 취급된다. 따라서 양자화로 인한 오차를 피할 수가 없는데, 만일 양자화 레벨이 미세하다면 오차는 더욱 감소할 것이다. 즉, N_b, N_e가 오차에 영향을 미치는 주요소이다.

3. CMAC의 특징

위에서 설명한 CMAC 신경회로망의 특성을 몇 가지로 요약하면 다음과 같다.

• CMAC는 일반화(generalization) 특성이 좋다. 많은 메모리 cell들은 하나의 정보를 표현하는 데 사용된다. 이것은 사람 뇌의 기능과 유사하다. 출력은 어드레스의 key를 이용함으로써 메모리 cell의 연상을 통해서 검색된다. 그러므로 CMAC는 불완전한 입력 패턴 key를 이용하여 일반화 및 패턴 검색과 같은 연상 메모리 특징을 갖고 있다.

• CMAC는 실제 입력을 받아 출력을 만들어낸다.

• CMAC는 학습성능이 좋다. CMAC는 신경회로망중의 하나로서 국부적으로 분포한 정보를 지니고 있다. 메모리에서 입력-출력 관계는 인접한 영역간에서 결정된다. 그러므로 학습을 용이하게 할 수 있다.

• CMAC는 추가학습(increamental learning) 특성이 좋다. Training을 위한 전체 training 데이터를 미리 준비하는 것은 상당히 어렵다. 학습 데이터는 학습과정에서 본래의 데이터에 추가시킬 수 있다. 따라서 weight들은 새로 확장된 데이타 집합으로 갱신된다.

• CMAC는 메모리를 효과적으로 이용한다. 많은 수의 상태들은 hash coding을 이용하여 보다 적은 물리적 메모리로 mapping되기 때문에 메모리의 크기를 줄일 수 있다.

• CMAC를 이용하면, 상세한 수학적 모델링 또는 해석, 학습에 대한 복잡한 실시간 계산이 필요치 않다. 이것은 CMAC 신경회로망에서 입력/출력 mapping시 training 샘플로부터 학습되기 때문이다.

• CMAC는 logic cell array를 이용하여 실제적인 hardware로 구현할 수 있다. CMAC의 시뮬레이션을 수행하기 위해서 과거에는 컴퓨터를 이용하였지만, 최근에 Miller 등은 CMAC를 hardware로 구현하였다 [6][7]. 따라서 앞으로 더욱 많은 weight의 문제해결과 더욱 거대한 시스팀에 실질적으로 이용할 수 있을 것이다.

위와 같이 CMAC 신경회로망은 기존의 신경회로망 모델들에 비해 많은 장점들을 갖고 있지만 다음과 같은 피할 수 없는 단점이 있다.

• 일반화 특성이 전체적이지 못하다. 입력 공간에서 서로 인접한 상태들은 입력이 training되지 않았더라도 출력이 서로 인접하여 나타난다. 이것은

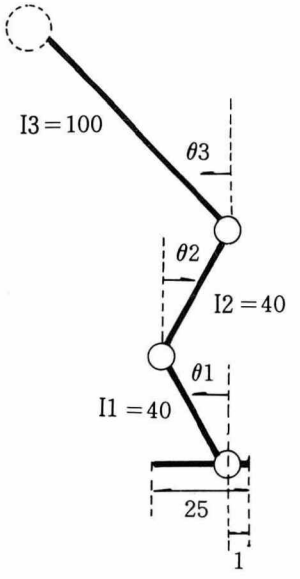
국부적인 일반화 특성을 갖게되고 다층 perceptron과 같은 전체적인 일반화 특성을 나타내는 신경회로망보다 더 적은 학습 충돌을 갖지만, 전체적 일반화 특성은 다층 신경회로망에 비해 다소 떨어진다.

• Hash coding으로 인한 메모리 충돌이 발생한다. 메모리 크기를 줄이기 위해 필요한 hash coding 때문에 메모리 충돌이 발생하게 되며 이것이 학습과정에서 잡음의 원인이 된다.

Ⅲ. CMAC의 제어문제 적용

1. 문제 정의

CMAC 신경회로망의 학습성능을 시험하기 위해서 3-joint squatting 문제를 이용한다. 학습 목표는 로봇을 넘어뜨리지 않고 원하는 위치까지 앉게 하는 것이다. 모의실험에 이용된 로봇은 (그림 7)에서와 같이 foot, leg, thigh 및 body로 구성되어 있고, 4개의 link를 갖는다. 각 joint의 각도는 수직축을 기준으로 하여 시계방향(+)으로 측정된다. 각 link의 길이와 질량, link 각도의 한계는 〈표 1〉과 같다.



(그림 7) A configuration of a robot for squatting.

〈표 1〉 A specification of 3-joint squatting robot.

Link length(cm)	Mass(kg)	Angle limits
foot : 25	foot : 0	$-45^\circ \le \theta 1 \le 45^\circ$
leg : 40	leg : 3	$-180^\circ \le \theta 2 \le 0^\circ$
thigh : 40	thigh : 5	$-40^\circ \le \theta 3 \le 140^\circ$
body : 100	body : 10	

2. 학습과정 및 시뮬레이션

3-joint 로봇에 대한 입력변수는 각 joint에 대한 각도($\theta 1$, $\theta 2$, $\theta 3$)이고, CMAC에 이용된 block과 element의 수는 각각 3, 4개이다. 따라서 물리적 메모리 Pm은 식 (6)에 의해서 108개이다. 로봇은 서있는 자세에서 출발한다. 즉, $\theta 1 = 2\theta = \theta 3 = 0^\circ$ 이다. 학습과정에서, link의 무게중심이 foot의 범위를 벗어나게 되면 학습은 실패한 것으로 간주하고 다시 시도하게 된다. 즉, 무게중심이 foot의 범위내에 위치하면 reward(r=1.0)로 주어지고 foot의 범위를 벗어나게 되면 penalty(p=−1.0)로 주어진다.

입력 상태변수를 $s_i(k)$ (i=1, 2, 3)라 하면, 상태 $s_i(k)$에 대하여 저장된 정보(CMAC의 출력)는 식 (4)에 의해서 다음과 같이 계산된다.

$$\mathrm{INFORM}(s_i(k)) = M \cdot U(s_i(k)) \tag{10}$$

메모리는 다음의 학습규칙에 의해서 갱신된다.

$$\triangle M = \frac{\alpha(\mathrm{Critic} - M \cdot U_i)}{N_e} \tag{11}$$

여기에서, 학습률 α는 0.1이다. Critic은 reward 또는 penalty값이 0보다 크면 1이고 그렇지 않으면 0의 값을 갖는다. 또한, 각도의 변화량($\Delta\theta$)은 다음과 같이 계산된다.

$$\Delta\theta = \begin{cases} 1^\circ, \text{ if } \dfrac{\partial f(w,\theta_i)}{\partial f(\theta_i)} + \sin(\theta_{id} - \theta_i) > 0 \\ -1^\circ, \textit{ otherwise} \end{cases} \quad (12)$$

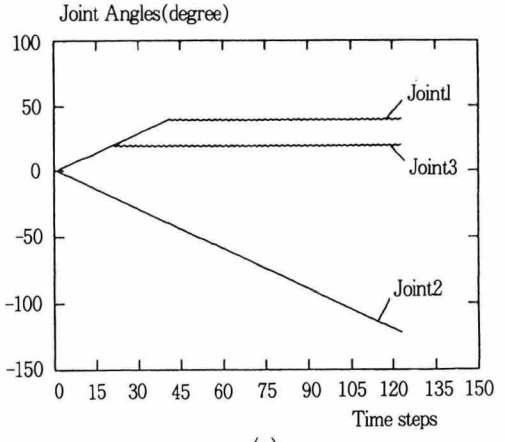
시뮬레이션의 학습과정은 다음과 같이 요약할 수 있다.

Step 1. 메모리를 초기화한다.

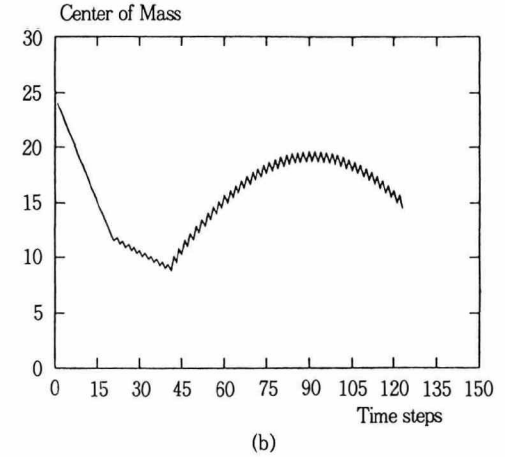
Step 2. 초기 각도(θ_i(t))와 목표 각도(θ_d(t))를 설정한다.

Step 3. 학습을 통해서 $\Delta\theta$(t)를 결정한다.

Step 4. θ(t)가 결정되면, 이 때의 무게중심이 안정한 영역이면 reward값을, 안전한 영역이 아니면 penalty값을 주고 학습한다.

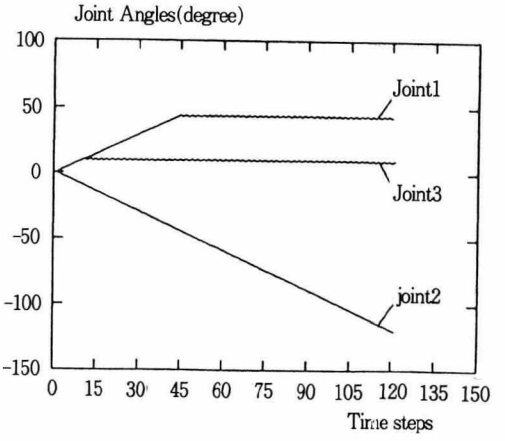


(a)

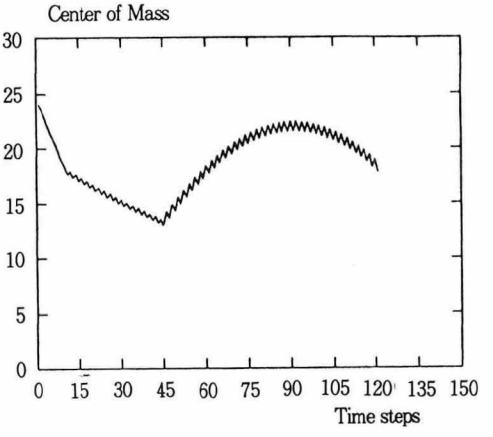


(b)

(그림 8) Joint paths and the center of mass for squatting example.(target angles : $\theta 1 = 40^\circ$, $\theta 2 = -122^\circ$, $\theta 3 = 20^\circ$)



(a)



(b)

(그림 9) Joint paths and the center of mass for squatting example.(target angles : $\theta 1 = 44^\circ$, $\theta 2 = -130^\circ$, $\theta 3 = 10^\circ$)

Step 5. $\theta(t)=\theta_d(t)$이면, Step 1에서부터 반복한다.

(그림 8)과 (그림 9)는 두 squatting 예제에 대한 시뮬레이션 결과이다. 두 예제 모두 초기상태의 각도는 $\theta1=\theta2=\theta3=0$ 이고, 이 때의 무게중심은 24.0이다. (그림 8)은 목표각도가 $\theta1=40°$, $\theta2=-120°$, $\theta3=20°$ 이고, 무게중심이 14.5인 경우이다. (그림 9)는 목표각도가 $\theta1=44°$, $\theta2=-130°$, $\theta3=10°$ 이고, 무게중심이 15.0인 경우이다. 두 예제 모두 무게중심의 이동이 [0, 25]의 범위를 벗어나지 않고 안정하게 목표각도까지 squatting한다.

Ⅲ. 결 론

본 고에서는 CMAC라 불리는 신경회로망의 메카니즘을 간략하게 기술하였고, CMAC 신경회로망의 특징들을 설명하였다. 또한 CMAC의 학습성능의 유효성을 보이기 위하여 3-joint robot의 squatting 문제에 적용하였다. CMAC 학습을 통해서 메모리에 저장된 정보는 robot의 squatting을 시험하는 데 이용하였고, 목표각도까지 쓰러지지 않고 잘 squatting함을 입증하였다. 앞에서 설명한 바와 같이 CMAC 신경회로망은 학습속도가 빠르고 국부적 일반화 특성이 좋으며 메모리를 효율적으로 이용하는 등 많은 장점이 있지만, 반면에 CMAC 신경회로망은 일반화 특성이 전체적이지 못하고 hash coding으로 인한 메모리 충돌이 학습과정에서 noise로 작용하게 되는 단점이 있다. 따라서 기존의 다층 신경회로망의 장점인 전체적 일반화 특성을 향상시키고 hashing nosie를 제거하는 방법이 연구되어져야 할 것이다.

참 고 문 헌

[1] Albus, J. S., "A new approach to manipulator control : The Cerebellar Model Articulation Controller (CMAC)," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, pp. 220−227, September 1975.

[2] Albus, J. S., "Data storage in the Cerebellar Model Articulation Controller (CMAC)," *Journal of Dynamic Systems, Measurement and Control, Transactions of ASME*, pp. 228−233, September 1975.

[3] Moody, J., "Fast learning in multi-resolution hierarchies," in *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, Ed. Los Altos, CA : Morgan Kaufmann, 1989, pp. 29−39.

[4] Carter, M. FJ., Rudolph, and A. J, Nucci, "Operational fault tolerance of CMAC networks," in *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, Ed. Los Altos, CA : Morgan Kaufmann, 1990, pp.340−347.

[5] Kraft, L. G. and D. P. Campagna, "A comparison between CMAC neural network control and two traditional adaptive control systems," *IEEE Control System Magazine*, pp.36−43, Apr. 1990.

[6] Miller, W. T., "Real-time application of neural networks for sensor-based control of robots with vision," *IEEE Transactions on Systems, Man and Cybernetics*, vol.19, pp.825−831, 1989.

[7] Miller, W. T. F. H. and L. G. Kraft, "CMAC : An associative neural network alternative to backpropagation," *Processdings of the IEEE*, vol. 78, no.10, pp. 1561−1567, Oct. 1990.

[8] Lin, C.−S. and H. Kim, "CMAC−based adaptive critic self-learning control," *IEEE Transactions on Neural Networks*, vol. 2, no.5, pp.530−533, 1991.

[9] Cotter, N. E. and T. J. Guillerm, "The CMAC and a theorem of Kolmogorov," *Neural Networks*, vol. 5, pp. 221-228, 1992.

[10] Wong, Y. and A. Sideries, "Learning convergence in the Cerebellar Model Articulation Controller," *IEEE Transactions on Neural Networks*, vol.3, no.1, pp.115-121, Jan. 1992.