論文94-31A-6-19

확률 연산을 이용한 볼츠만 머신

(Boltzmann machine using Stochastic Computation)

李 日 完*, 蔡 洙 翊*

(Eel Wan Lee and Soo Ik Chae)

要 約

확률 연산은 VLSI로 신경망을 구현할 때에 곱셈기의 면적을 줄이기 위해 많이 쓰이고 있다. 이와 같은 장점 외에도 확률 연산은 볼츠만 머신을 구현하는 데에 필요한 고유한 랜덤 오차를 가지고 있다. 이 오차는 볼츠만 머신에서 전역 최소점을 찾기 위해 쓰이는 시뮬레이티드 아닐링에 유용하다. 본 논문에서는 이러한 확률 연산을 이용하여 볼츠만 머신을 구성하는 방법론을 제안하고 특히 문제가 되는 덧셈기 문제와 시뮬레이티드 아닐링에 대해 좀더 상세한 분석을 하였다. 이 분석에 의하면 확률연산을 이용한 볼츠만 머신은 패턴 인식/완성 문제에 적합한 구조임을 알 수 있었다. 이와 같은 결과는 패턴 인식/완성 문제의 전형이라고 할 수 있는 XOR, full adder, 숫자 인식 문제등의 모의 실험을 통하여 확인하였다.

Abstract

Stochastic computation is adopted to reduce the silicon area of the multipliers in implementing neural network in VLSI. In addition to this advantage, the stochastic computation has inherent random errors which is required for implementing Boltzmann machine. This random noise is useful for the simulated annealing which is employed to achieve the global minimum for the Boltzmann Machine. In this paper, we propose a method to implement the Boltzmann machine with stochastic computation and discuss the addition problem in stochastic computation and its simulated annealing in detail. According to this analysis, Boltzmann machine using stochastic computation is suitable for the pattern recognition/completion problems. We have verified these results through the simulations for XOR, full adder and digit recognition problems, which are typical of the pattern recognition/completion problems.

Key Words

stochastic computation, Boltzmann machine, simulated annealing

Ⅰ. 서론

확률 연산은 집적회로로 신경 회로망을 구현할 때에 곱셈기의 면적을 줄이기 위해 쓰이는 수 표현 방법의 하나이다.[1] [2] [3] 이같은 구현상의 장점외에도 확률 연산은 볼츠만 머신을 구현하는 데에 필요한 랜

* 正會員, 서울大學校 半導體共同研究所 및 電子工學科
(Dept. of Elec. Eng., Seoul Nat'l Univ.)

※ 본 연구는 과학재단(KOSEF)의 핵심전문연구 과제(931-0900-059-2)의 연구비로 이루어졌음

接受日字 : 1993年 6月 5日

덤 오차를 내포하고 있다. 이 랜덤 오차는 볼츠만 머신에서 전역 최소점을 찾기 위해 사용되는 시뮬레이티드 아닐링에서 이용된다. 그러나, 이 둘을 바로 결합시켜 볼츠만 머신을 만드는 데에는 본문에서 살펴보는 바와 같이 확률연산에서 덧셈기의 입력이 늘어날 수록 점점 덧셈기를 구현하는 것이 어려워지는 문제와 기존의 수체계를 이용한 시뮬레이티드 아닐링에서의 온도를 바로 확률연산에서의 시뮬레이티드 아닐링에 도입할 수 없다는 문제들이 있다. 본 논문에서는 이와 같은 문제들에 대한 해결 방법을 제시하고 실험으로 그 결과를 확인하여 확률연산으로 볼츠만 머신을 구현할 수 있음을 확인하였다.

본 논문은 다음과 같이 구성되었다. 서론에서는 그동안 별개로 연구되어져 온 확률연산과 볼츠만 머신에 대해 각각 설명을 한다. 본문인 2장에서는 확률연산을 이용한 볼츠만 머신을 구현할 때 문제가 되는 덧셈기 문제와 아닐링 문제에 대해 언급하고 모의 실험결과를 살펴보도록 한다. 3장에서는 이러한 문제들을 고려하여 구성한 확률연산을 이용한 볼츠만 머신의 벤치마크로 XOR, 덧셈기, 숫자 인식문제의 시뮬레이션 결과를 살펴보고 토의한다

1. 확률 연산

확률 연산이란 수를 표현함에 있어서 한 노우드의 신호가 '1'이 될 확률을 이용하여 나타내며 또 이러한 수들을 간단한 논리 소자와 지연 소자만을 이용하여 계산하는 방법을 일컫는다.[2] 가령 AND소자의 한 입력단자를 관측하였을 때, 절반 정도의 시간에는 '1'을 관측하고 나머지 절반 정도의 시간에는 '0'을 관측하였다면 단극성 표현방법으로는 '1/2'를 나타내고 양극성 표현으로는 '0'을 나타낸다. 그리고 AND 소자의 나머지 입력단자에서 '1'을 관측할 확률이 역시 1/2이었다면 이 AND게이트의 출력단자에서 '1'을 관측할 확률은 1/4이며 이것은 단극성 표현에서는 두 수를 곱한 결과이다. 이와 같은 확률 연산은 제한된 면적의 집적회로에 보다 많은 곱셈기를 넣어 병렬처리를 시도하기 위하여 이용되고 있다.[1], [3]

확률 연산에서의 오차는 기존의 이진 연산 방법과는 다르게 랜덤하게 나타난다. 이 랜덤 오차는 결국 신호에 더해지는 잡음으로 모델링할 수 있다. 랜덤 오차와 표현하고자 하는 수 x와의 관계는 표 1에 요약되어 있다. 확률 연산에서는 노우드에서 '1'을 관측할 확률을 어떻게 해석하는가에 따라 수 표현 영역과 곱셈 및 덧셈의 구현 방법이 달라지게 된다. 본 논문에서는 그 중에서 양극성 수 표현 방법을 썼다. 표 1에서 잡음에서의 L은 확률을 추정하는 데에 쓰이는 필스의 관측 길이이고, P 는 관측하는 신호가 '1'로 관측될 확률을 나타낸다.

표 1. 확률 연산에서의 수 표현 방법
Table 1. Number representation in the stochastic computation.

	수 해석 방법	표현 영역	곱셈 방법	덧셈 방법	랜덤 오차(잡음)
단극성	x=P	0<= x <=1	AND	OR	$\sqrt{x(1-x)/L}$
양극성	x=2*P-1	-1<= x <=1	XNOR	Majority Logic	$\sqrt{(1-x*x)/L}$

2. 볼츠만 머신

볼츠만 머신은 되먹임(feedback) 연결이 있는 신경망 구조를 가진다.[4] 또한 뉴런간의 연결 강도에 의해 정의된 에너지를 최소화하는 문제에 홉필드(Hopfield) 망의 경우와 같이 국부 최소점에 빠지는 것을 방지하기 위해 시뮬레이티드 아닐링(Simulated Annealing)이라는 최적화 기법을 사용하는데 이 알고리즘은 다음과 같이 기술할 수 있다.

시뮬레이티드 아닐링

(A.1) 하나의 뉴런이 주위의 다른 뉴런들에 의해 정해지는 내적에 따라 식 (1)과 같은 확률에 의해서 자신의 다음 상태를 결정한다. 내적을 달리 표현하면, 자신의 상태가 변했을때 생기는 에너지 차이라고 할 수 있다.

$$P(X=+1)=\frac{1}{1+e^{-(\sum_j W_{ij} X_j)/T}}$$
$$P(X=-1)=\frac{1}{1+e^{+(\sum_j W_{ij} X_j)/T}} \quad (1)$$

(A.2) 위의 식에서 확률을 결정할 때 도입되는 변수 T를 [온도] 라 부르며 처음에는 이 온도를 높게 유지하다가 차츰 이 온도를 낮추어 가면서 하나의 뉴런의 상태가 +1 또는 -1의 값을 가질 확률을 0 또는 1이 되게 한다.

위의 방법 (A.1)과 (A.2)는 시뮬레이티드 아닐링을 신경망에 적용하여 에너지(또는 비용 함수)의 전역 최소점을 찾는 과정이다. 가중치와 주위 환경이 주어졌을 때 이 전역 최소점을 찾는 시뮬레이티드 아닐링을 외부에서 불러 쓸 수 있는 하나의 과정으로 보고 이 과정에 학습 알고리즘을 추가할 수 있다. 패턴 인식을 예를 들어 설명하면 다음과 같다.

볼츠만 머신에서의 학습 알고리듬

(L.1) 뉴런을 되먹임이 없는 순방향 연결만이 있

는 망 구조와 같이 입력 뉴런, 은닉 뉴런 ,출력 뉴런 세 가지로 나눈다.

(L.2) [Teacher Phase] 라 불리는 단계에서는 입력과 출력을 모두 원하는 값으로 고정을 시킨 뒤 은닉뉴런의 상태를 미리 정의된 에너지가 최소가 되도록 정한다. 이때 위의 (A.1)과 (A.2)에서 기술된 시뮬레이티드 아닐링을 사용한다.

(L.3) [Student Phase] 라 불리는 단계에서는 입력만을 원하는 값으로 고정시킨 뒤 은닉 뉴런과 출력 뉴런의 상태를 이미 정의한 에너지가 최소가 되도록 정한다. 이 때에도 마찬가지로 위의 (A.1)과 (A.2)에서 기술된 시뮬레이티드 아닐링이 쓰인다.

(L.4) 이 두개의 과정에서 결정된 값으로 뉴런사이의 연결 세기를 다음의 학습식에 의해 갱신한다.

$$\Delta W_{ij} = \eta \cdot \left(< S_i \cdot S_i >_{teacher} - < S_i \cdot S_j >_{student} \right) \qquad (2)$$

식 (2)에서 S_i는 i번째 뉴런의 상태를 나타낸다. W_{ij}를 계산하기 위하여 S_i와 S_j만을 이용하는 이 학습식은 오차 역전파 학습방법을 채용한 순방향 신경회로망과 같이 국부성(locality)을 지니고 있다. 이러한 성질은 신경망을 만들 때에 연결에 소요되는 면적을 줄일 수 있는 좋은 성질이다.

Ⅱ. 볼츠만 머신에의 확률 연산 도입

앞의 I.2 절에서 살펴본 바와 같이 볼츠만 머신은 뉴런의 상태가 '1'에 있을 확률을 모사하기 위해서는 난수 발생기가 필요하다. 그러나, 볼츠만 머신에서 내적을 구하는 연산에 모두 확률 연산을 적용하게 되면, I.1절에서 언급한 확률 연산이 가지고 있는 랜덤 오차때문에 굳이 따로 난수 발생기를 만들지 않더라도 계산된 내적값 자체에 잡음 즉, 랜덤 오차가 들어가 이를 바로 시뮬레이티드 아닐링에 쓸 수 있다. 그러나, 이와 같은 기본 개념을 그대로 확장을 할 때 문제가 되는 것은

(1) 확률연산에서는 모든 입력에 대해 선형 덧셈의 조건을 만족시키는 덧셈기를 찾기가 어렵고,

(2) 에너지 최소점으로의 수렴을 보장하는 시뮬레이티드 아닐링에서는 난수의 발생확률 분포를 조정할 수 있어야 하는데 확률연산에서는 이미 연산 구조가 결정이 되면 이 난수의 발생확률 분포를 바꾸기가 어렵다는데에 있다.

따라서 본 장에서는 이러한 문제들이 실제로는 어떻게 나타나는지 그 경향을 분석하고 해결책을 제시한다.

1. 덧셈기

확률 연산에서 많은 수의 입력을 가지는 덧셈기를 구현하는 것은 그리 간단한 문제가 아니다. 이를 반드시 확률적으로 계산할 필요를 느끼지 않으면 축전기에 전하를 계속적으로 축적하는 방식을 써서 처리를 하는 것이 보통이다. [5] [6] [7] 등의 논문에는 이러한 덧셈 방법을 이용한 펄스 연산형 신경망 회로들이 제시 되어 있다.

이와 같은 방법은 어느 정도 제한된 범위내에서는 덧셈의 선형성을 보장한다. 그러나, 이와 같은 방식으로 덧셈을 하면 덧셈기의 동작 영역(dynamic range)이 펄스열의 관측길이에 비례하여 제한을 받고, 이는 곧 덧셈기의 선형 동작을 보장하는 최대 입력의 갯수가 바로 관측 길이에 의해 제한을 받는 것을 의미한다. 이에 대한 해결책으로 본 논문에서 제시하는 덧셈방식은 표 2에 요약되어 있는 것처럼 매번 커패시터에 더한 값에서 바로 '1' 또는 '0'의 결과를 얻어내어 그 결과를 축전기에 더해주는 것이다. 이와 같은 덧셈 방식은 축전기에 단지 매번 비교 결과의 부호 정보만을 기억할 것임을 알 수 있다. 또한 이 때의 결과는 '1' 또는 '0'의 값만을 가지므로 누산기로 축전기대신 계수기(counter)를 이용해 전체 뉴런을 구현하는 것도 가능하다. 이 두 방법들에 대한 개념도는 그림 1과 같다. 두번째 덧셈 방법의 핵심은 공간 상에서의 덧셈과 시간상에서의 덧셈을 두개의 단계로 나누어 처리해 줌으로써 동작 영역을 첫번째 덧셈방법과는 달리 관측길이에 무관하도록 만들 수 있다는 것이다.

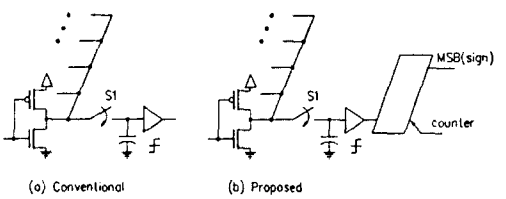


그림 1. 확률연산에서의 두가지 덧셈 방법

Fig. 1. Two schemes for the summation in the stochastic computation.

확장 가능한 최대 시냅스 수의 상한은 확률연산의 주기가 시작되기 이전에 우선은 축전기의 전압을 VDD/2로 충전을 하고 그 뒤에 확률 연산을 해주었을 때, 계속 충전만이 되거나 또는 계속 방전만이 되는 경우에 축전기(capacitor)에 충전되는 전하가 VDD나 GND로 포화되지 않도록 하는 조건에서 얻어질 수 있으며 표 2에 그 값을 정리하였다. 단, 여

기에서 C는 전하를 누적시키는 커패시터의 값, △V는 선형조건이 제한하는 전압의 유효영역, I는 하나의 시냅스가 켜졌을 때 흐르는 전류의 크기, △t는 트랜지스터가 전류를 축전기에 공급하는 시간이다.

표 2. 두 덧셈 방법에 의해 최대로 확장 가능한 시냅스의 갯수

Table 2. The maximum allowable number of the synapses in each scheme for the summation.

	덧셈 방법의 간단한 기술	확장 가능한 최대 시냅스의 수
방법 (a)	키르히호프의 전류 법칙에 의해 더해진 전류로 일정시간동안 스위치 S1을 열어 축전기의 전하를 더하거나 뺀다. 이는 일반 덧셈기로 보아도 된다.	$\frac{C\cdot\Delta V}{I\cdot L\cdot\Delta t}$
방법 (b)	한 묶음동안 전류법칙에 의해 더해진 전류를 보고 이를 크기 정보는 버리고 단지 1또는 -1의 부호정보만을 남긴채 이 정보를 계수기에 더한다.	$\frac{C\cdot\Delta V}{I\cdot\Delta t}$

그러나, 방법 (b)는 그림 2의 도표에서 알 수 있듯이 원래의 덧셈과 완전히 그 값이 일치하지 않을 뿐 아니라 부호까지 일치하지 않는 영역이 있다. 그림 2는 입력의 갯수가 작은 경우에 대해서는 비교적 선형 덧셈을 충실히 하는 방법(a)와 최대 확장 가능한 덧셈기의 입력을 늘이기 위해 제시한 방법 (b)의 결과의 차이를 보기 위해 그린 그래프이다. 그림 2에서 가로 축은 그림 1에서의 방법 (a)에 의한 덧셈 결과를 나타내므로 그 범위가 입력의 갯수가 증가할 수록 계속 커지는 반면, 그림 1에서의 방법 (b)의 결과를 나타내는 세로 축은 계속 -1과 +1의 사이에 있게 된다. 따라서 이 영역의 크기는 그림 2에 나타나 있는 것과 같이 내적을 계산하는 입력의 갯수와 그 입력의 패턴에 영향을 받는다. 그러나, 서로 다른 입력 패턴에 대해서 같은 내적의 결과를 얻을 수 있으므로 방법 (a)와 방법 (b)는 다음과 같은 예에서 볼 수 있듯이 경우에 따라서는 두 덧셈의 결과의 부호가 달라지는 경우도 발생하게 된다.

(예제) 입력 패턴에 따라 결과의 부호가 달라지는 경우로는 3개 입력을 가진 경우의 한 예로 (0, 0, 0) (1/2, -1, 1/2)을 들 수 있다. 이 둘은 서로 다른 입력 패턴이지만 그림 1에 제시된 방법 (a)에 의하면 둘 다 내적값으로는 0을 가진다. 그러나, 방법 (b)에 의하면 (0,0,0)은 0을 출력하지만 (1/2,-1,1/2)는 1/8(=(1/2 -1 + 1/2 + 1/2 * 1 * 1/2)/2)을 출력한다

그러나, 이 두 덧셈의 방법이 서로 완전히 상관이 없는 것은 아니므로 그림 2에서 비록 그 구간의 길이가 0은 아닐지라도 어느 한쪽의 덧셈 결과가 관측되면 나머지 한쪽의 덧셈 결과가 일정한 범위로 제한되는 것을 알 수 있다.

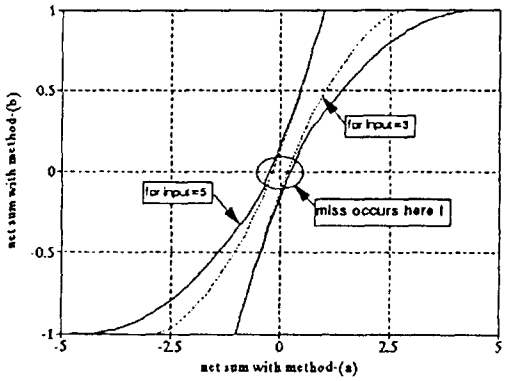


그림 2. 두가지 덧셈방법의 차이

Fig. 2. Difference between two schemes for the summation.

위의 두 방법을 수식으로 표현하면 다음과 같이 나타낼 수 있다.

$$\text{방법 } (a) : Net_i = \sum_j w_{ij} \cdot x_j \tag{3}$$

$$\text{방법 } (b) : x_i = \sum_{l.n(B_{1,l})>n(B_{0,m,l})}^{2^n} [\prod_j^n \frac{1-(-1)^{b_j} w_{ij} \cdot x_j}{2}] \tag{4}$$

단, 여기에서

$$B_{1,l} = \left\{ b_i \in \{0,1\} | \sum_{i=1}^{n} b_i \cdot 2^i = i, b_i = 1 \right\} \tag{5}$$

$$B_{0,l} = \left\{ b_i \in \{0,1\} | \sum_{i=1}^{n} b_i \cdot 2^i = i, b_i = 0 \right\} \tag{6}$$

로 정의된다.

예를 들면, 입력의 갯수가 3인 경우 두 덧셈 방법을 보면

$$\text{방법 } (a) : x_1 + x_2 + x_3 \tag{7}$$

$$\text{방법 } (b) : \frac{x_1 + x_2 + x_3 - x_1 x_2 x_3}{2} \tag{8}$$

과 같이 나타난다.

그러나, 이보다 큰 수의 입력에 대해서는 현재 연구가 진행되고 있기는 하나 아직 명확하게 수식으로 풀기가 어렵기 때문에 표 3에서와 같이 각 덧셈기의 입력이 [-1,+1] 에서 균일한 확률 분포를 갖는다고 가정을 하여 [-1,+1] 사이에서 무작위로 수를 뽑아

두 덧셈의 결과에서 부호 차이가 날 경우의 확률을 알아보았다. 이 실험으로 두개의 덧셈 방법이 완전히 일치하지는 않지만 두 덧셈결과의 부호가 일치하지 않는 확률은 각 입력에 인가되는 신호의 값이 [-1, +1] 에서 균일할 때, 일정 범위 이하가 됨을 확인하였다.

표 3. 서로 다른 방법으로 덧셈을 수행하였을 때 두 결과의 부호가 다를 확률

Table 3. Probability of having different signs between the summation results based on two addition schemes.

입력의 수	1	3	5	7	9	11	13
부호가 다를 확률(%)		1.91	1.17	0.74	0.56	0.68	0.51

2. 상태 전이 및 시뮬레이티드 아닐링

I.2절에서 언급한 시뮬레이티드 아닐링은 온도의 제어 방법에 따라 전역 최소점으로의 수렴을 보장할 수 있는 좁은 의미의 '시뮬레이티드 아닐링(Simulated Annealing)'과 단지 경험에 의해 온도를 낮추어 실제 응용에 적용하기는 하되 전역 최소점으로의 수렴을 보장할 수는 없는 '시뮬레이티드 퀜칭(Simulated Quenching)'으로 분류된다.[8] 시뮬레이티드 아닐링은 전역 최소점을 보장할 수는 있으나 이름에서 알 수 있듯이 전역 최소점으로의 수렴속도가 매우 느리며 따라서 매우 많은 연산량을 요구하게 된다. 따라서 볼츠만 머신의 응용에서는 시뮬레이티드 퀜칭을 쓰는 경우가 많은데, 본 논문의 경우도 '시뮬레이티드 퀜칭'을 써서 온도를 제어하였다. 이후로 '시뮬레이티드 아닐링'이라함은 넓은 의미의 시뮬레이티드 아닐링을 언급하는 것이다.

확률연산으로 시뮬레이티드 아닐링을 구현하려면 확률연산이 잡음을 내포한다고 하더라도 이 잡음의 크기를 제어할 수 있어야 한다. 특히, 볼츠만 머신에 적용하는 시뮬레이티드 아닐링은 또한 뉴런의 상태전이 방법에 따라 달라지게 된다. 따라서 확률연산으로 효과적인 시뮬레이티드 아닐링을 구현하기 위해서는 온도 조절과 상태전이 방법, 이 두가지에 대한 논의가 필요하다.

볼츠만 머신에서의 온도를 결정하는 많은 것들 가운데 가장 직접적으로 관계되는 것은 바로 펄스의 관측길이 L이므로 본 논문에서는 이를 조절을 함으로써 온도를 조절하고자 한다. 확률 연산에서의 온도는 다음과 같이 정의된다. 길이가 L인 펄스 열의 분산은 $\sigma_x^2 = \frac{p(1-p)}{L} = \frac{1-x^2}{L}$ 이므로, 높은 온도에서 오차가 크도록 볼츠만 머신에서 랜덤 오차의 분산은 온도에 비례한다고 가정한다. 따라서 $\frac{1-x^2}{L} = k_B \cdot T$ 가 되며, 결국 온도 T는 $T = \frac{1-x^2}{k_B} = \frac{1}{L}$ 로 정의 될 수 있다. 여기에서 T는 표현하고자 하는 수 x에도 영향을 받기 때문에 L만으로 완전히 온도를 조절하는 것이 불가능함을 알 수 있다. 그러나 표현하고자 하는 수 x는 주어지는 문제에 의해 정해지는 것이므로 이를 마음대로 바꿀 수 없다. 따라서 전체적으로 비례상수 ξ 를 곱해 온도 T가 유한한 펄스열의 관측 길이 L에 대해서는 항상 0이 되는 경우를 방지한다.

$$|x'| = |\xi \cdot x| \le \xi \tag{9}$$

$$\frac{1-\xi^2}{k_B \cdot L} \le T \le \frac{1}{k_B \cdot L} \tag{10}$$

온도를 식 (10)과 같이 정의를 하여 이 온도를 $1/(k_B \cdot 1)$로 만들어 주는 것은 단지 한 펄스열만을 관측한다는 것이고, 이를 0에 가깝게 떨어뜨린다는 것은 관측길이를 매우 길게 한다는 것이므로 뉴런의 상태를 제대로 결정할 확률이 더욱 높아지므로 이는 1차적인 의미에서 기존의 볼츠만 머신에서의 온도에 해당함을 확인할 수 있다.

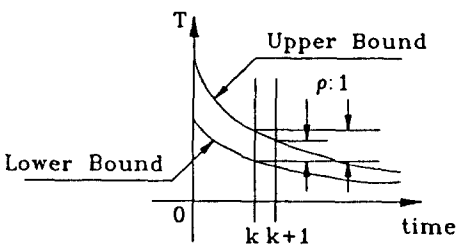


그림 3. 아닐링 스케줄의 도해

Fig. 3. Illustration of the annealing schedule.

그림 3에서 보듯이 확률연산을 이용하여 시뮬레이티드 아닐링을 할 때에는 온도가 정확히 정의되지 않고 그래프에 표시한 것처럼 범위가 주어지게 된다. 이 때 온도를 천천히 낮추면 그림 3에서와 같이 추정되는 온도의 영역이 겹치는 부분이 항상 있게 된다. 본 논문에서 ρ 를 이전의 온도 구간에 대한 겹쳐진 온도구간의 비로 정의하고 이 비를 항상 일정하게 유지하도록 하면서 아닐링을 하였다. 이 방법은 아래의 식에서 보이는 바와 같이 기존의 수체계를 사용하는 볼츠만 머신에서 많이 쓰인 아닐링 스케줄인 지수적인 아닐링 스케줄에 해당하게 된다.

$$(\frac{1}{k_b \cdot L_k} - \frac{1-\xi^2}{k_B \cdot L_k}) \cdot \rho = (\frac{1}{k_b \cdot L_{k+1}} - \frac{1-\xi^2}{k_B \cdot L_k}) \quad (11)$$

$$\frac{1-(1-\rho)\cdot\xi^2}{L_k} = \frac{1}{L_{k+1}} \quad (12)$$

$$L_{k+1} = \frac{L_k}{1-(1-\rho)\cdot\xi^2} \quad (13)$$

이 매개변수 ρ 는 실제 시뮬레이티드 아닐링에서 각 뉴런이 한 온도구간에 머무는 횟수 또는 각 뉴런이 한 온도구간에서 상태전이할 수 있는 횟수 n_T로 표시할 수 있다. 즉, n_T번 전이를 하였을때의 온도가 더이상 겹치지 않는 다음 온도 구간으로 옮기도록 하려면

$$\frac{(1-(1-\rho)\cdot\xi^2)^{n_T}}{L_k} = \frac{1-\xi^2}{L_k} \quad (14)$$

$$n_T = \frac{\log(1-\xi^2)}{\log(1-(1-\rho)\cdot\xi^2)} \quad (15)$$

$$\rho = 1 - \frac{1-(1-\xi^2)^{1/n_T}}{\xi^2} \quad (16)$$

로 항상 일정하게 유지해야 할 온도구간의 겹침비 ρ 를 구할 수 있다. 그러나, 주의할 것은 이와 같이 지수적으로 온도를 낮추는 것이 실제 응용에서 많이 쓰이고 있는 아닐링 방법이기는 하나 단지 '더 나은 해(better solution)'의 확률을 높이는 것이지 반드시 전역최소점을 찾는 것을 보장하지는 않는다는 것이다.

상태 전이 방법은 매 온도에서 상태전이할 수 있는 뉴런의 수를 온도에 비례하도록 만들어 줌으로써 초기에는 병렬전이에 의해 잘못 상태전이되는 경우가 있더라도 이는 단지 조정하고자 하는 온도보다 실제 온도가 약간 더 높게 나오는 정도의 영향을 미치나, 아닐링의 말기에는 상태전이가 가능한 뉴런의 평균수 $\langle N_k \rangle$가 1이하로 떨어짐으로서 순차적인 상태전이 방법과 유사하게 동작을 함으로써 이와 같은 잘못된 상태전이가 볼츠만 머신이 전역최소점으로 수렴하는것을 방해하지 않도록 하였다. 그림 4는 뒤의 III장에서 모의 실험한 숫자 인식에서 실제로 각 온도별로 상태전이한 확률을 보여준다.

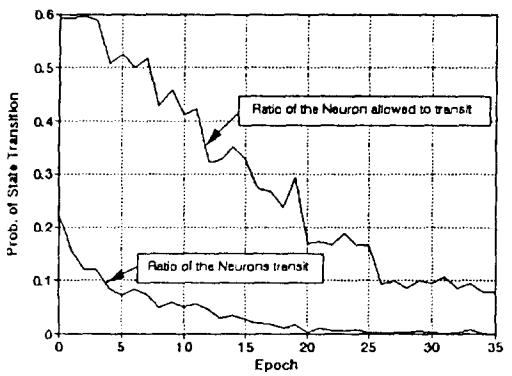


그림 4. 온도 조절의 시간 경과에 따른 각 뉴런의 상태전이 확률

Fig. 4. Probability of the state transition for each neuron.

3. 낮은 온도 구현의 문제

볼츠만 머신을 가지고 부울 대수 문제나 패턴 인식/완성 문제를 풀기 위해서는 온도를 매우 낮추어 수를 표현하는 정확도를 높여야 하며, 이와 같이 낮은 온도를 만들기 위해서는 수를 표현하는 데에 쓰이는 펄스의 길이가 아주 길어야 한다. 그러나, 이는 난수 발생기의 주기성을 이용해 난수 발생기의 주기와 비슷한 펄스의 주기로 만들면 무한히 긴 펄스 열을 사용하지 않더라도 그림 5에서와 같이 랜덤 오차를 매우 작게 만들 수 있다. 이 랜덤 오차및 전체 오차에 대한 식은 아래와 같이 계산된다.

$$\varepsilon_{tot}^2 = \varepsilon_{thermal}^2 + \varepsilon_{offset}^2 \quad (17)$$

$$= \frac{(L/P-n)((n+1)-L/P)}{L/P} \cdot \frac{1-x^2}{L} + \frac{1}{P^2} \quad (18)$$

단, P는 LFSR(Linear Feedback Shift Register)이나 Cellular Automata등 난수 발생기의 주기를 나타내고 L은 펄스 열의 관측 길이를 x는 -1에서부터 +1까지의 표현하는 수를 나타내며, n은 $n \le L/P \le n+1$을 만족하는 정수를 나타낸다.

위 식에서 왼쪽의 항은 LFSR(Linear Feedback Shift Register) [9] 이나 Cellular Automata [10] 등 난수 발생기의 주기성에 의해 커지거나 작아지는 항이며 오른 쪽 항은 난수 발생기의 펄스열의 길이가 유한하여 생기는 양자화 오차에 해당한다. 난수 발생기의 주기가 매우 긴 경우 이상적인 경우와 255인 경우 그리고 63인 경우에 대해서 오차가 어떻게 변하는지는 그림 5에 있다. 그림 5와 식 17, 18에서 알 수 있듯이 같은 펄스열의 길이를 가지고서도 LFSR등의 특성방정식을 바꾸어줌으로써 LFSR의 주기를 관측하는 펄스열의 길이에 가까이 가져갈 수 있게 되고 이러면 그림 6와 같이 랜덤 오차가 급격히 줄어들게 되어 전역 최소점을 찾기 위해 필요한 매우 낮은 온

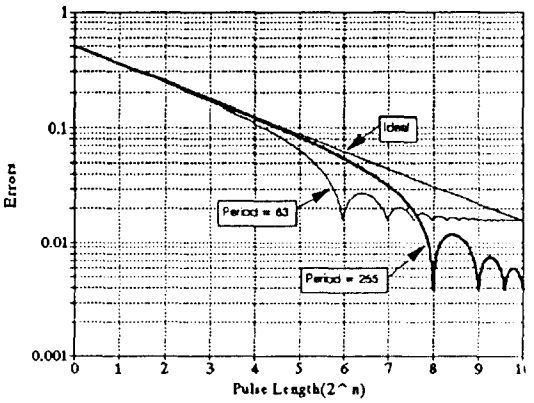


그림 5. 펄스의 주기와 난수발생기의 주기, 특성 방정식에 의해 정해지는 잡음의 크기
Fig. 5. The error represented with pulse length and the period of PRNG and its characteristic equation.

도를 그리 길지 않은 펄스열을 가지고서도 만들어 줄 수 있다.

4. 학습 방법을 고려한 전체 시스템의 구성 및 타이밍

본 논문에서의 볼츠만 머신의 학습 방법은 I.2에서 언급한 Hebbian 방식의 학습 방법을 이용하였다. 이 Hebbian 학습 방법을 확률연산을 이용하는 볼츠만 머신에 그대로 구현한 회로가 그림 6에 있다. 그림 6에서 시냅스라고 블랙박스로 처리된 것은 아래 그림 7에 보다 상세하게 나타내었다. 이와 같이 학습 회로가 붙어 있는 각 시냅스가 그림 8에 있는 하나의 뉴런을 위해 여러개가 들어간다. 그림 8에 있는 것은 그림 1에서 살펴본 방법 (a)를 이용하여 덧셈부분을

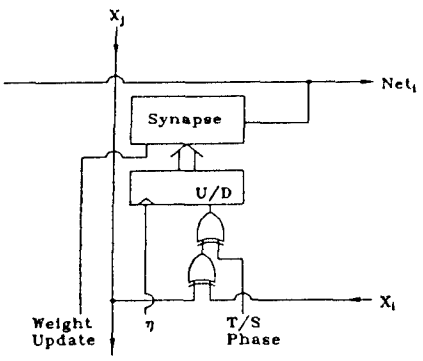


그림 6. 각 시냅스마다 있는 학습회로
Fig. 6. Learning circuit in each wynapse.

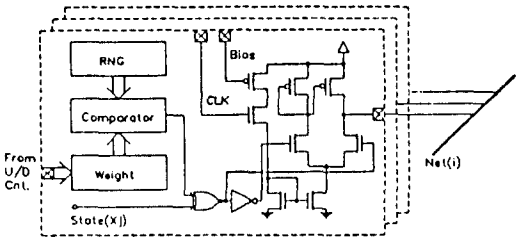


그림 7. 확률연산을 이용한 신경망의 시냅스
Fig. 7. Synapses in Neural Network using stochastic computation.

구현한 뉴런이다. 그림 7과 그림 8은 그림 1의 덧셈기를 제어 신호까지를 포함해 보다 자세히 그린 것이다.

또한 이와 같은 Hebbian 학습 방법까지를 고려한 전체적인 타이밍은 그림 9에 있다. 그림 9에서 비스듬히 올라가는 부분은 확률연산을 하기 위해 난수 발생기가 필요한 부분 -즉, 순방향 연산을 하는 부분- 이고, 수직으로 올라가고 내려가는 부분에서는 확률연산을 하지 않고 바로 계수기를 이용하여 값을 변경하는 것을 나타낸다. 시냅스에 갱신(Update) 신호를 주는 것은 확률 연산을 쓰지 않고 바로 덧셈을 한다.

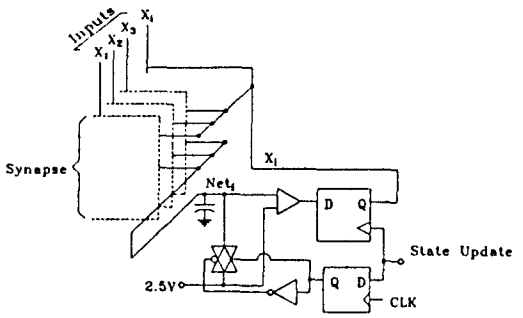


그림 8. 전체적인 시스템 블럭도
Fig. 8. Block diagram of the whole system.

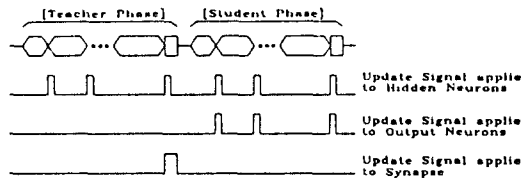


그림 9. 학습까지 포함한 전체적인 시스템의 동작도
Fig. 9. Timing diagram of whole system including learning phases.

예를 들어 학습률 η가 4/256이었다면 256개의 펄스를 이용하여 4/256을 구현하는 것이 아니라 간단히 4번의 덧셈이나 뺄셈을 해주면 보다 정확하고 빠른 연산을 할 수 있다.

Ⅲ. 모의 실험 결과

볼츠만 머신으로 풀 수 있는 문제를 나누면 크게 최적화 문제와 패턴 인식/완성 문제로 나눌 수 있다. 본 논문에서는 이 두가지 유형의 문제중 주로 패턴 인식/완성 문제에 대해 모의 실험을 하였다. 최적화 문제보다 패턴 인식/완성 문제를 모의 실험의 대상으로 선정한 주된 이유는 확률연산에 내재하는 랜덤 오차때문에 가중치의 해상도를 높이기 위해서는 매우 긴 펄스열이 필요한 반면, 패턴 인식/완성 문제는 최적화 문제에 비해 학습이 종료된 후에 요구되는 가중치의 해상도가 4-5비트로 그리 높지 않아도 되기 때문이다.

아래의 세 패턴 인식/완성 문제들에 대해서 학습시에 학습률은 모두 4/256를 적용하였다. 아닐링 스케쥴에서 처음에 온도를 0으로 만들지 않기 위해 ξ는 0.5로 실험을 하였다. 아닐링 스케쥴에서 처음 시작한 펄스의 길이는 32, 최종 펄스의 길이는 256로 선택했으며 길이를 점차로 각각의 뉴런이 4번 정도의 상태전이를 할 수 있을 정도의 시간으로 다음 펄스의 길이로 바꾸어 주었다. 다음 펄스의 길이로 바꾸는 것은 항상 이전 펄스 길이의 두배로 바꾸어 주었다. 최종 펄스의 길이가 256인 것은 II.3절에서 언급한 오차 분석에 따르면 전체적으로 가중치를 대략 4-5비트로 표현할 수 있음을 나타내는 것인데, 이는 다른 논문들에서 볼츠만 머신의 가중치를 표현하는데 필요한 비트수와 일치함을 알 수 있다.

또, 아닐링의 마지막 단계에 가서는 256개의 펄스로도 오차가 생길 수 있기 때문에 온도가 매우 낮은 것을 모사하기 위해 II.3절에서 설명한 방법대로 난수 발생기의 주기를 펄스의 숫자와 일치시켜 뉴런의 상태값을 바꾸었다. 그러나, 위에서 정한 아닐링 스케쥴에 쓰인 값들은 모두 경험적으로 얻어진 값들이기 때문에 이 값들의 올바른 선택에 대해서는 앞으로도 좀 더 많은 연구가 필요하다고 생각된다.

세개의 모의 실험은 모두 SUN 4.1.3/SPARC10 환경에서 이루어졌으며 가장 계산량이 많은 숫자 인식의 경우에는 한 번의 모의 실험을 하는데에 평균 4시간 정도의 계산 시간이 소요되었다. 그러나, 시냅스 전체를 하드웨어로 구성하여 병렬처리를 하였다고 가정을 하여 계산시간을 살펴보면

클럭주(10^7) * 갱신되는 뉴런의 수(20) *
한 주기당 갱신되는 횟수(4) *
평균 펄스 관측렬(128) * 아닐링 기간(16) *
반복 학습 횟수(256) *
Teacher/Student Phase 학습(2)
= 8.3초

정도 소요될 것을 예상할 수 있다.

1. XOR 문제

문제를 푸는 데에 쓰인 뉴런의 수는 입력 뉴런 2개, 출력 뉴런 1개, 은닉 뉴런 6개였다. 실제로 입력과 출력사이에 연결이 있을 때에는 하나의 은닉 뉴런만으로도 XOR 문제는 표현이 가능하고 학습도 가능함이 알려져 있다. 그러나, 본 논문에서 제안한 확률 연산을 이용한 볼츠만 머신에서는 이와 같이 은닉 뉴런이 적게 사용되면 덧셈이 제대로 이루어지지 않아 기본적으로 어느 이상의 은닉 뉴런이 필요하게 되는데, 이를 실험으로 확인해본 결과 은닉 뉴런이 6개 이상일때부터 학습이 제대로 이루어지는 것을 확인할 수 있었다.

그림 10에서 세로 축은 실험에서 전체 학습 패턴중에 아직 배우지 못한 패턴의 평균 갯수를 나타내며 이는 10 회의 반복 실험을 통하여 얻은 평균값이다. 가로축에서 한 칸은 한 패턴의 집합에 대해 모두 학습을 한 것을 나타내며 따라서 XOR문제에서는 한 Epoch를 지나는데 모두 4번의 아닐링을 거치게 된다.

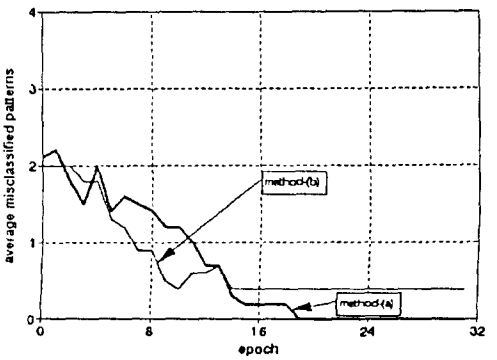


그림 10. XOR 문제 학습 곡선
Fig. 10. Learning curves for the XOR problem.

2. Full Adder 문제

이 문제는 다중 출력 문제의 전형으로서 채택한 문제이다. 이 문제에서 뉴런의 구성은 입력 뉴런 3개, 출력 뉴런 2개, 은닉 뉴런 8개이다. 여기에서도 마찬가지로 은닉 뉴런은 8개를 썼으나 학습이 끝났을

때에는 매 입력 패턴마다 다른 은닉 뉴런의 조합이 나타나지 않고 단지 3 개의 다른 패턴만이 나타났다. 이보다 은닉 뉴런의 갯수가 적을 때에는 Full Adder 문제가 학습하기가 어려웠는데, 역시 은닉 뉴런의 갯수가 적으면 덧셈이 제대로 이루어지지 않은 것이 그 주된 이유로 생각된다.

그림 11에서도 역시 세로 축은 10 회 반복 실험을 하여 평균적으로 배우지 못한 패턴의 수를 나타내고 있다. 이 그래프에서는 두개의 방법이 큰 차이를 보이지 않음을 볼 수 있는데 그 이유는 역시 본론에서 지적하였듯이 입력의 갯수가 늘어나면 두 덧셈의 차이가 별로 나지 않는 가중치로 학습이 될 확률이 높기 때문이다. 이 실험 역시 매우 적은 수의 반복 학습을 통해 학습이 가능함을 볼 수 있다.

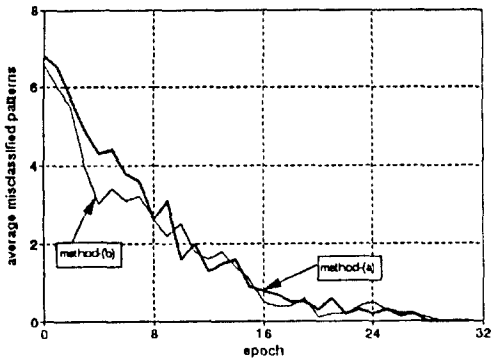


그림 11. Full Adder 문제의 학습곡선
Fig. 11. Learning curve for the Full Adder problem.

3. 숫자 인식 문제

본 숫자 인식 문제에서의 뉴런의 배열은 그림 12와 같다. 그림 12에서와 같이 뉴런들을 배열한 뒤 그 다음에 입력 뉴런으로 들어오는 숫자 패턴들에 대해 출력에 있는 10 개의 뉴런이 각 숫자에 대해 하나씩 반응을 하도록 하는 Winner-Take-All방식의 코우딩을 채용하였다. 그림 12에서 보는 바와 같이 입력 뉴런 20 개, 출력 뉴런 10 개, 은닉 뉴런 10개를 사용하여 숫자 인식을 위한 볼츠만 머신을 만들었다. 이들 사이에는 전부 연결이 있다. 또 바이어스 뉴런은 3 개를 이용하여 문턱값이 -3에서 +3까지의 값을 가질 수 있도록 고려를 하였다.

그림 13은 숫자 인식 문제를 10회 반복하여 풀었을 때, 나올 수 있는 학습 곡선을 평균을 내어 보여준 것이다. 이 문제는 학습이 거의 끝난 상태에서도 평

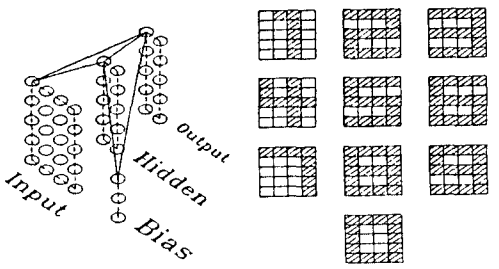


그림 12. 숫자 인식에 쓰이는 신경망의 구조와 패턴들
Fig. 12. Structure of the neural network and training patterns for the digit recognition problem.

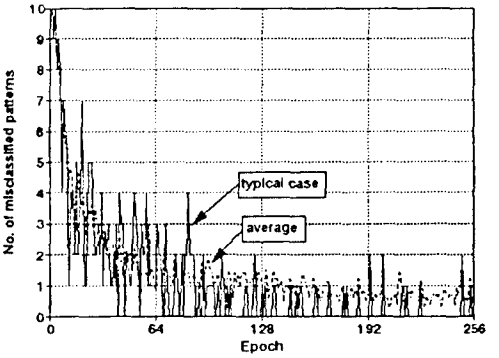


그림 13. 숫자인식 문제의 학습 곡선
Fig. 13. Learning curve for the digit recognition problem.

균적으로는 한개 정도를 제대로 인식하지 못하고 있는 것으로 나타나나 다른 선으로 그려진 그래프를 보면 하나의 학습 곡선들을 보았을 때에는 실제적으로는 학습이 이루어진 이후에는 거의 대부분의 구간에서 인식하지 못하는 패턴이 0이다가 갑자기 2-3개를 인식하지 못하는 경우들이 겹쳐져서 이와 같은 그래프가 나타났다. 이와 같은 결과는 아닐링에서 온도를 충분히 낮추지 않은 것과 또 문자의 입력공간이 4 * 5로 작아서 실제로 문자간의 구별이 그리 용이하지 않았던 것들을 주된 원인으로 꼽을 수 있겠다.

Ⅳ. 결론

본 논문에서는 기존의 볼츠만 머신의 개념을 확률연산을 이용하는 신경망에 적용하였고, 이 때에 생길 수 있는 제반 문제점들을 파악하여 이 문제들이 어떤 경향성을 보이는지에 대해 여러 데이타를 수집하여

이들에 대한 해결 방법을 제시하였다. 또 이렇게 제시한 방법으로 이미 기존의 다른 신경망을 이용하여 푼 몇가지 문제에 적용을 해 봄으로써 새로 제시한 방법이 그 중에서도 그리 높지 않은 가중치의 해상도를 요구하는 패턴 인식/완성 문제에 특히 적합함을 확인하였다. 이와 같이 확률연산을 볼츠만 머신에 적용하는 과정에서 덧셈기의 문제는 신경망의 입력의 갯수가 클 때에는 그 영향이 점점 줄어드는 것을 또한 실험적으로 확인을 하였으나 이 경향에 대해서는 보다 세심한 분석이 필요하다고 여겨진다. 또한 시뮬레이티드 아닐링의 문제도 현재까지 밝힌 것은 본 논문에서 정의에 따른 온도를 낮추면 보다 낮은 에너지를 가지는 상태로 수렴할 확률이 높아진다는 것을 증명한 것에 불과하므로 이 수렴 성질에 대해서도 보다 정량적인 분석도 필요하다고 여겨진다.

參 考 文 獻

[1] M. S. Tomlinson, Jr., D. J. Walkerm and M.A. Sivilotti, "A Digital Neural Network Architecture for VLSI," Proc. IJCNN, vol. 2, pp. 545 - 550, 1990.

[2] P. Mars and W. J. Poppelbaum, *Stochastic and deterministic averaging processors*. Peter Peregrinus, 1981.

[3] S. Park, C. Kim, and S. Chae, "A VLSI Design of the Neural Networks using Random Pulse Streams," Proc. JTC-CSCC '92, Kyungjoo, Korea, pp. 501 - 503, 1992.

[4] Emile Aarts, Jan Korst, Simulated Annealing and Boltzmann Machine, John Wiely & sons, 1989.

[5] Alan F. Murray, "Pulse Arithmetic in VLSI Neural Network," Artificial Neural Networks 1992.

[6] Gyu Moon, VLSI Design of Neural Network using Pulse Coded Weight with On-chip Learning Capability, PhD thesis, George Washington University, 1990.

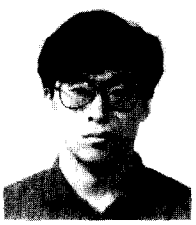
[7] R. B. Allen, J. Alspector and B. Gupta, "Performance of a stochastic learning microchip," in Advances in Neural Information Processing Systems, 1989, vol. 1, pp.

[8] Lester Ingber, "Simulated Annealing : Practice versus theory," Statistics and Computing, Jan. 1993.

[9] David Green, "Modern Logic Design," pp. 165-187, Addison-Wesley, 1986.

[10] P. D. Hortensius, R. D. Mcleod, W. Pries, D. M. Miller and H. C. Card, "Cellular Automata-Based Pseudorandom Number Generators for Built-In Self-Test," IEEE Trans. CAD, vol. 8, pp. 842 - 859, Aug. 1989.

著 者 紹 介



李 日 完(正會員)

1969年 8月 29日生. 1992年 2月 서울대학교 전자공학과(학사). 1994年 2月 서울대학교 전자공학과(석사). 1994年 3月 ~ 현재 서울대학교 대학원 전자공학과 박사과정 재학중. 주관심 분야는 신경회로망 및 집적 회로 설계 등임.

蔡 洙 翊(正會員)

1952年 11月 2日生. 1976年 2月 서울대학교 전기공학과(학사). 1978年 2月 서울대학교 대학원 전기공학과(석사). 1979年 8月 ~ 1982年 3月 공군사관학교 교수부교관. 1987年 (美)Stanford 대학교 전기공학과(박사). 1978年 11月 ~ 1988年 7月 ZyMos Corp. Design Manager. 1988年 8月 ~ 1990年 5月 대우통신 근무. 1990年 7月 ~ 현재 서울대학교 반도체 공동연구소 및 전자공학과 조교수. 주관심 분야는 신경회로망과 집적회로 설계 등임.