논 문 논문번호 17-3-9

다단계 신경망 모델에 대한 다중 처리기 상으로의 사상 전략과 분산 역전파 알고리즘†

(A Mapping Strategy of Multilayered Neural Networks on Multiprocessor and Its Distributed Backpropagation Algorithm)

낭 종 호* 최 선 민 이 상 훈 윤 현 수** 맹 승 렬**
(Jong Ho Nang) (Seon Min Choe) (Sang Hoon Lee) (Hyun Soo Yoon) (Seung Ryoul Maeng)

요 약

본 논문에서는 완전히 연결된 다단계 신경망 모델을 분산 메모리 다중 처리기 시스템에 사상시키는 방법과, 이 사상 방법에 기초한 분산 역전파 학습 알고리즘을 제안하고 그 성능을 분석하였다. 제안한 방법에서는 신경망의 각 단계에 있는 뉴론들을 p개의 서로 다른 집합으로 나누고, 이렇게 나눈 부신경망을 p개의 처리기에 할당하여 학습을 시키게 된다. 분석에 의하면 제안된 사상 방법과 분산 역전파 학습 알고리즘의 p-처리기 수행 시간은 하나의 처리기를 사용하는 경우의 $\frac{3}{4}\cdot p$ 만큼의 시간만이 필요하며, 또한 한 처리기는 하나의 처리기를 사용하여 학습 시키는 경우에 비하여 $\frac{p}{2}$만큼의 메모리만을 가지고 있으면 되기 때문에 큰 신경망을 빠른 속도로 학습시킬 수 있다. 이런 분석은 어떤 신경망 모델을 학습시키는데 필요한 가장 적당한 처리기의 갯수를 결정하는데 사용될 수 있다.

ABSTRACT

In this paper, we propose and analyze a parallel learning algorithm of a fully connected multilayered feedforward neural network using the backpropagation one on a distributed-memory multiprocessor system. In our system, the neurons on each layer are partitioned into p disjoint sets and each set is mapped on a processor of a p-processor system The p-processor speed-up ratio of the backpropagation algorithm over a single processor is $\frac{3}{4}\cdot P$, and space-reduction ratio over a single processor is $\frac{P}{2}$ This analysis can be used as a basis in determining the most cost-effective or optimal number of processors.

† 본 연구는 1989년도 한국과학재단 연구비 지원(KOSEF 891-1105-003-1)으로 이루어진 것임.

* 정 회 원 : 한국과학기술원 전산학과
** 종신회원 : 한국과학기술원 전산학과

접수일자 : 1990년 3월 13일

1. 소 개

신경망 모델의 가장 큰 특징으로는 학습(learning)기능과 고도의 병렬성(parallelism)을 들 수 있는데, 현

재까지 여러 신경망 모델이 개발되어 많은 응용 분야에서 그 효용성을 증명하였다. 현재 발표된 신경망 모델은 망의 형태, 각 뉴론의 특징, 그리고 학습 방법에 따라서 50여개의 신경망 모델로 나눌 수 있다[Trel89]. 이런 신경망 모델은 인간의 두뇌를 모방한 계산 모형을 사용하기 때문에 기존의 계산방식으로는 풀기 어려웠던 많은 문제를 쉽게 풀수 있지만, 신경망 모델의 크기가 커지면(즉, 뉴론의 갯수가 많아지면)학습을 시키거나 원하는 결과를 얻어내는데 많은 양의 계산을 필요로 한다. 그러므로 새로운 신경망 모델을 개발하거나 새로운 응용 분야에 대한 실험을 할 때 시간이 많이 걸린다는 문제점을 가지고 있다. 이런 문제점을 해결하기 위하여 신경망을 효율적으로 구현하는 방법에 관한 연구가 많이 있어 왔는데, 이런 연구는 크게 하나의 물리적 처리기가 하나의 뉴론을 담당하도록 하는 직접 구현(physical implementation)방법과, 하나의 처리기가 시분할 방식으로 여러 뉴론을 처리하는 가상 구현(virtual implemenation)방법으로 나눌 수 있다.

직접 구현 방법은 VLSI나 광학 등을 사용하는 방법인데, 대부분의 경우 아날로그 연산기에 의하여 처리되기 때문에 연속적인 수치의 계산을 기반으로 하는 신경망 모델의 성격에 부합되어 효과적인 연산이 가능하다. 반면에 이런 구현 방법은 현재의 기술로는 많은 수의 뉴론과 연결선으로 이루어진 신경망의 구현이 어렵고, 신경망 모델의 변경이 어렵기 때문에 범용성을 기대할 수 없다는 문제점을 가지고 있다. 이와는 달리 가상형 구현 방법은 하나의 처리기가 시분할 방법에 의하여 여러개의 뉴론을 담당하여 처리하는 방법인데, 신경망의 형태나 각 뉴론의 행동 방식을 프로그램에 의하여 변경할 수 있는 범용성을 가지고 있어서 새로운 신경망 모델을 개발하거나 실제 응용 시스템을 개발하는데 많이 사용되고 있다.

신경망 모델의 가상 구현에 관한 연구는 다시 크게 DSP(Digital Signal Processor)와 같은 전용 프로세서를 개발하는 연구[Work 88]와, Systolic Array에 기초한 신경망 구현에 관한 연구[Kung 88, Depr89, Forr87, Mill89], 그리고 기존의 다중 처리기 시스템의 병렬 처리 기법을 사용한 구현에 관한 연구[Pome88, Ghos89, Cook 88]등으로 나눌 수 있다. 그러나 DSP와 같은 전용 프로세서는 그 성능에 한계가 있고, Systolic Array를 이용한 구현 방법은 신경망 모형이 하드웨어적으로 고정되어 있기 때문에 범용성이 부족하다는 문제점을 가지고 있다. 반면에 다중 처리기에 기초한 병렬 시뮬레이션 방법은 처리기의 숫자에 비례하여 성능 향상을 기대할 수 있고, 소프트웨어에 의하여 여러 신경망 모델을 시뮬레이션할 수 있는 범용성을 가지고 있기 때문에 많은 연구들이 이런 방법을 사용하고 있다.

본 논문에서는 분산메모리 다중 처리기(Distributed-Memory Mutiprocessor DMM)를 이용한 신경망 병렬 시뮬레이션 방법을 제안하였다. 특히 병렬 학습 시스템의 출발점으로서 가장 보편적으로 사용되고 성능이 우수한 신경망 모델인 다단계 역전파(*multi-layered backpropagation*)모델을 DMM에 사상(mapping)시키는 방법과, 이런 사상 방법에 기초한 병렬(혹은 분산)역전파 학습 알고리즘을 제안하였으며, 이 방법의 예상되는 성능 향상을 수행 시간과 필요한 총 메모리양의 관점에서 분석하였다. 이런 사상 방법과 병렬 학습 알고리즘은 다른 유사한 신경망 모델에도 쉽게 적용할 수 있을 것이다.

2장에서는 다단계 신경망 구조와 역전파 학습 알고리즘에 대하여 설명하고, 3장에서는 이 신경망 모델을 DMM 시스템에 사상하는 방법과, 이런 사상 방법에서의 분산 역전파 학습 알고리즘을 설명하고, 4장에서는 예상되는 성능향상 정도를 수행 시간과 필요한 총 메모리 양의 관점에서 분석하도록 한다. 5장에서는 본 연구의 결론과 앞으로의 연구 방향에 대하여 언급하도록 한다.

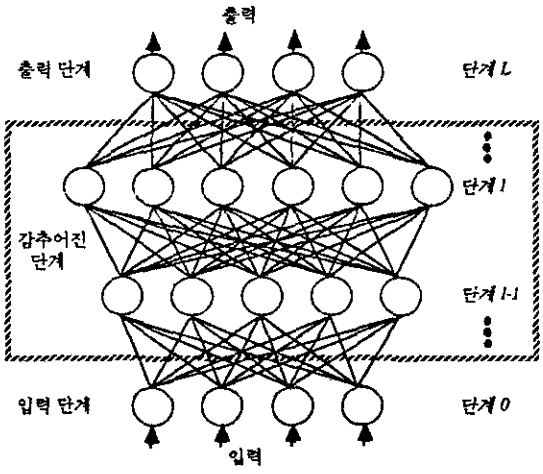
2. 다단계 역전파 학습 알고리즘

본 논문에서는 병렬화(parallelizing)하고자 하는 신경망 모델은 역전파 학습 알고리즘을 사용하는 완전히 연결된 다단계 신경망(fully connected multi-layered neural network)이다. 이 신경망 모델은 가장 보편적인 신경망 모델로서 바젼이나 음성 인식, 음파 탐지, 레이다, 신호 처리, 그리고 로보트 응용 분야 등에서 많이 사용되고 있다[Darp 88]. 본 장에서는 이 신경망 모델의 구조와 학습 알고리즘에 대하여 알아보도록 한다.

2.1 완전히 연결된 다단계 신경망 구조

완전히 연결된 다단계 신경망 구조는 (그림 1)에 나타낸 것과 같이 $L+1$개의 단계로 이루어져 있다. $l=0$

에 있는 첫번째 단계는 입력단계로서 n_0개의 뉴론으로 이루어져 있으며, 다음에 있는 l($1 \le l \le L$)단계들은 각각 n_l개의 뉴론으로 이루어져 있다. 각 단계에 있는 뉴론들은 다음 단계의 모든 뉴론들과 연결되어 있으며, 각 단계 l의 i번째 뉴론($n_i(l)$)은 활성값(activation value) $a_i(l)$를 가지고 있으며, 뉴론 $n_i(l)$와 뉴론 $n_j(l+1)$를 연결하는 연결선(link)은 가중치(weight value) $w_{ji}(l, l+1)$를 가지고 있다.



(그림 1) 완전히 연결된 다단계 신경망 구조

2.2 역전파 학습 알고리즘

감독 학습(supervised learning) 알고리즘은 학습시키고자 하는 입/출력 패턴에 대응되는 연결선의 가중치를 찾는 프로시쥬어로서, 어떤 입력을 보여주면서 신경망이 생성하는 결과가 원래의 원하는 출력 패턴과 일치하도록 가중치를 바꾸어 주는 것이다. 역전파 학습 알고리즘[Rume 87]은 감독 학습 알고리즘의 일종으로서 다음과 같은 3패스로 이루어져 있다.

- 전 방향 수행
- 오류의 역전도
- 연결선 가중치 수정

전 방향 수행에서는 주어진 입력 패턴에 대한 뉴론 $n_i(l)$의 활성값 $a_i(l)$을 이용하여 구하게 된다. 여기서 f는 nonlinear sigmoid함수로서 $f(x)=\frac{1}{1+e^{x}}$ 의 형태를 갖는다.

$$a_i(l) = f\left(\sum_{j=1}^{n_{l-1}} w_{ij}(l-1,l)\cdot a_j(l-1)\right),\ l=1,\ldots,L \text{ and } i=1,\ldots,n_l \qquad (1)$$

두번째 패스에서는 신경망의 실제 생성한 값과 원하는 값과의 차이(즉, 오류)를 계산하여 이 차이를 식-(2)을 이용하여 입력 단계까지 역전파하는 패스이다. 이 식에서 $\delta_i(l)$는 뉴론 $n_i(l)$의 오류값이고 $t_i(L)$는 출력 단계에 있는 뉴론 $n_i(L)$의 원하는 활성화 값이다.

$$\delta_i(l) = \begin{cases} [t_i(l)-a_i(l)]\,[a_i(l)\,(1-a_i(l))], & l = L \\ \left[\sum_{k=1}^{n_{l+1}} \delta_k(l+1)\, w_{ki}(l,l+1)\right][a_i(l)\,(1-a_i(l))], & l = L-1,\ldots,1 \end{cases} \qquad (2)$$

세번째 패스는 2번째 단계에서 구한 $\delta_i(l)$을 이용하여 연결선의 가중치를 식-(3)을 이용하여 바꾸는 단계인데, 식-(3)에서 η은 얼마나 학습을 빠르게 할 것인가를 결정하는 학습 속도이다.

$$\Delta w_{ij}(l-1,l) = \eta\cdot\delta_i(l)\cdot a_j(l-1) \qquad (3)$$

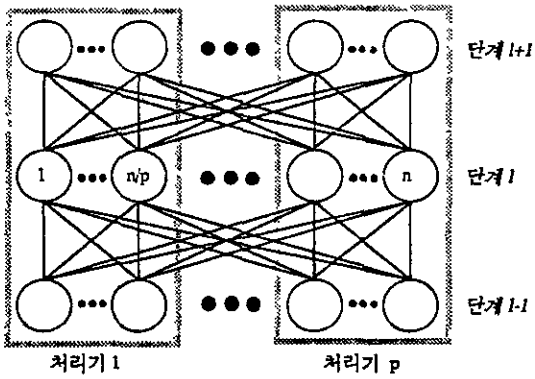
3. 다중 처리기 시스템으로의 사상 방법과 분산 역전파 학습 알고리즘

신경망 모델을 다중 처리기 시스템을 이용하여 시뮬레이션하는 가장 구현 방법에서 가장 중요한 이슈로는 병렬 시뮬레이션의 성능을 최대로 하기 위하여 어떤 방식으로 신경망을 분할하여 다중 처리기 시스템의 각 처리기에 할당하느냐 하는 사상 방법(mapping strategy)이다. 본 장에서는 완전히 연결된 다단계 신경망을 p개의 처리기로 이루어진 DMM(Distributed-Memory Multiprocessor)로 사상하는 방법과, 이런 사상 방법을 이용한 분산 역전파 학습 알고리즘에 대하여 설명하도록 한다.

3.1 다중 처리기 시스템으로의 사상 방법

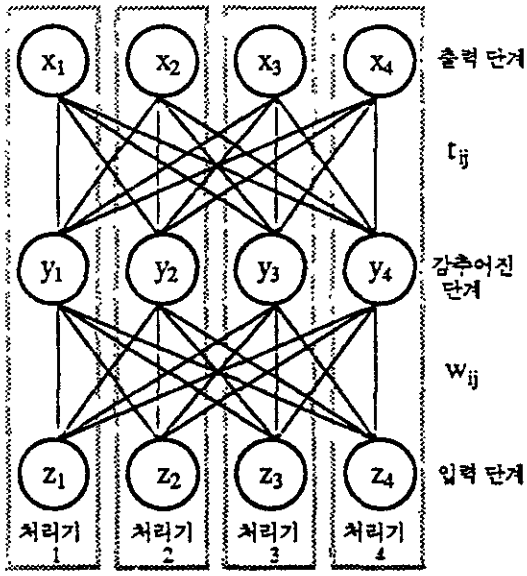
앞에서 언급한 것과 같이 본 논문에서 목표로 삼고 있는 다중 처리기 시스템 DMM은 공유 메모리가 전혀 없이, 처리기 사이의 통신이 포인트와 포인트를 연결하는 링크를 통하여 이루어지는 다중 처리기 시스템이다. 이런 처리기에 적합하도록 연결된 다단계 신경망을 분할하는 방법은 (그림 2)에 나타낸 것과 같이 신경망의 각 l단계에 있는 뉴론들 n_l을 처리기의 갯수 만큼 나누어서 각 처리기에 할당하는 것이다.

각 처리기는 자신의 지역 메모리에 자기가 맡은 뉴론들에 대한 활성값과 오류값, 그리고 할당된 뉴론의 입력과 출력 연결선의 가중치를 저장하게 된다. 그런데

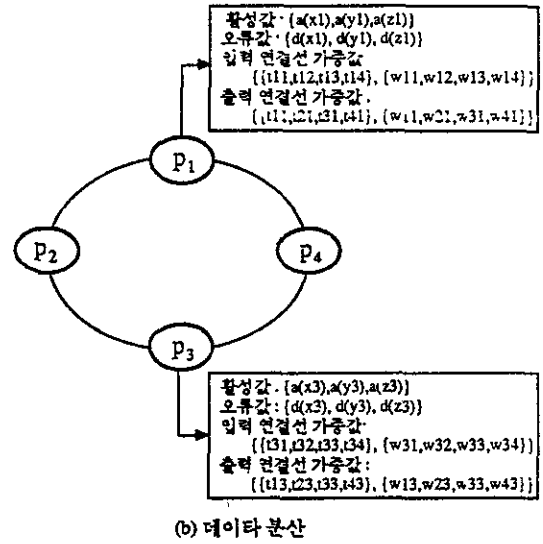


(그림 2) 완전히 연결된 다단계 신경망 분할 예제

단계 l에 있는 어떤 뉴론의 입력 연결선은 단계 l-1에 있는 뉴론의 출력 연결선이 되기 때문에 같은 연결선의 가중치가 두 처리기의 지역 메모리에 저장되게 된다. 이런 분할 방법은 비록 연결선의 가중치를 중복하여 저장함으로서 메모리를 많이 사용하게 되지만, 3.3절에 나타낸 것과 같이 분산 역전파 알고리즘의 수행중에 필요한 복잡한 통신을 피할 수 있게 하여 준다. 하지만 각 뉴론에 대한 활성화 값이나 오류값은 완전히 p개로 나누어져서 저장되게 된다. (그림 3)의 (a) 는 각 단계마다 4개의 뉴론을 가지고 있고, 총 3단계로 이루어진 완전히 연결된 신경망 구조를 링(ring)의 형태로 연결된



(a) 신경망 분할 예제



(b) 데이타 분산

(그림 3) 신경망의 분할과 데이타 분산 예제

4 처리기-DMM에 사상하는 예제를 나타낸 것이고, (b) 는 이런 경우에 신경망의 여러 데이타들이 각 처리기의 지역 메모리에 저장되는 형식을 나타낸 것이다.

3.2 분산 역전도 학습 알고리즘

3.1절에서 언급한 것과 같이 본 논문에서 제안한 사상 방법에서는 하나의 신경망이 p개의 부신경망으로 나누어져 각각의 처리기에 할당되므로, DMM의 각 처리기는 전체 신경망을 시뮬레이션하기 위해서는 서로 협력을 하여야 한다. 그러므로 DMM의 각 프로세서는 기본적으로 식 (1)-(3)을 이용하여 역전파 학습 알고리즘을 수행하지만, 단계 l의 뉴론에 대한 계산을 하기 위해서는 다른 처리기의 지역 메모리에 저장되어 있는 단계 l-1과 단계 l+1의 뉴론의 대한 데이타가 필요하기 때문에 수행중에 처리기 사이의 통신이 필요하게 된다. 그러므로 식(1)-(3)의 역전파 학습 알고리즘의 대한 분산 학습 알고리즘의 각 패스는 개념적으로 통신 단계와 계산 단계로 나눌 수 있다. 앞의 사상 방법에서 연결선의 가중값은 학습 알고리즘의 세번째 패스에서 조정이 되고 각 처리기는 자신의 지역 메모리에 입력 연결선과 출력 연결선에 대한 가중치를 모두 가지고 있기 때문에, 첫번째 패스와 두번째 패스를 연결선 가중치의 교환 없이도 수행할 수 있다. 하지만 두 곳에 저장되어 있는 연결선 가중값이 항상 같도록 유지하기 위해서는

어떤 coherence방법이 필요하게 되는데, 이 방법에 대해서는 3.2.3절에서 설명하도록 한다.

이제 역전도 학습 알고리즘을 여러개의 처리기가 협력하면서 수행하는데 필요한 통신 패턴과, 각 패스에 대한 분산 알고리즘에 대하여 설명하도록 한다. 여기서 $N_t(l)$는 처리기 p_t에 할당된 단계 l의 뉴론들의 집합을 나타내기 때문에 $|N_t(l)| = \frac{n_l}{p}$이다.

3.2.1 첫번째 패스 : 전 방향 수행

식(1)을 이용하여 $a_i(l)$을 계산하기 위하여 처리기 p_t는 다른 처리기에 저장되어 있는 모든 $a_j(l-1)$을 알아야 한다. 이런 요구 조건은 각각의 처리기가 자기에게 할당된 단계 $l-1$의 모든 뉴론 $j(j \in N_t(l-t))$들의 활성화 값 $(a_j(l-1))$을 방송(broadcasting)하고, 다른 뉴론 $j(j \notin N_t(l-1))$들의 활성화 값 $a_j(l-1)$을 받음으로서 충족될 수 있다. 이런 통신 패턴을 all-to-all 방송[John 89]이라고 하는데, 각 처리기에 저장된 메세지를 각 처리기가 방송하여 결과적으로 모든 처리기에 모든 메세지가 전달되도록 하는 방법이다.

이런 all-to-all 방송이 끝나게 되면, 각 처리기는 첫번째 단계에서 필요한 모든 활성값$(a_j(l-1))$을 알 수 있기 때문에 식(1)을 독립적으로 수행할 수 있게 된다. 여기서 유의할 것은 앞에서 언급한 것과 같이 이미 분할 방법에 의하여 연결선의 가중값$(w_{ij}(l-1,l))$은 각 처리기가 지역적으로 입/출력 연결선의 가중값을 저장하고 있기 때문에 서로 교환될 필요가 없다. 〈알고리즘-1〉은 앞에서 설명한 분산 역전도 학습 알고리즘의 첫번째 패스를 나타낸 것인데, 여기서 all-to-all-broadcast()는 $a_j(l-1)$을 방송하고 받기 위한 프로시쥬어이다(이 프로시쥬어는 4장에서 자세히 설명하도록 한다.)

〈알고리즘-1〉 분산 전 방향 수행 알고리즘

```
for l = 1 to L do
    /* Broadcast and Receive a_j(l-1) */
    for each neuron j∈N_t(l-1) do
        all-to-all-broadcast(a_j(l-1));
    end_for
    /* Compute a_i(l) */
    for each neuron i∈N_t(l) do
        a_i(l) = f( Σ_{j=1}^{n_{l-1}} w_ij(l-1,l)·a_j(l-1) );
    end_for
end_for
```

3.2.2 두 번째 패스 : 오류의 역전파

오류의 역전파 패스에 대한 분산 알고리즘은 활성값을 방송하는 대신에 오류값$(\delta_k(l+1))$을 방송하는 것을 제외하고는 첫번째 패스와 비슷하다. 즉, $j \in N_t(l)$인 뉴론 $n_i(l)$의 오류값$(\delta_i(l))$을 계산하기 위해서는 다른 처리기에 저장되어 있는 모든 $\delta_k(l+1)$이 필요한데, 이런 요구 조건은 앞의 패스에서와 마찬가지로 각각의 처리기 P_t가 $k \in Nt(l+1)$인 뉴론 $n_k(l+1)$의 오류값$(\delta_k(l+1))$을 방송하고, $k \notin N_t$인 각 뉴론 $n_k(l+1)$의 오류값$(\delta_k(l+1))$을 받는 all-to-all 통신을 수행함으로서 만족될 수 있다. 〈알고리즘-2〉는 위에서 설명한 처리기 p_t에 대한 오류 역전파 패스에 대한 분산 알고리즘을 나타낸 것이다.

〈알고리즘-2〉 분산 오류 역전파 알고리즘

```
/* Compute Error Values for Output Layer */
for each neuron i∈N_t(L) do
    δ_i(L) = [t_i(L) - a_i(L)][a_i(L)·(1 - a_i(L))]
end_for

for l = L - 1 to 1 do
    /* Broadcast and Receive δ_k(l+1) */
    for each neuron k∈N_t(l+1) do
        all-to-all-broadcast(δ_k(l+1));
    end_for
    /* Compute δ_i(l) */
    for each neuron i∈N_t(l) do
        δ_i(l) = [ Σ_{k=1}^{n_{l+1}} δ_k(l+1)·w_ki(l,l+1) ][a_i(l)·(1 - a_i(l))]
    end_for
end_for
```

3.2.3 세번째 패스 : 연결선 수정

앞에서 언급한 것과 같이 연결선에 대한 가중값이 두개의 처리기에 각각 저장되어 있기 때문에 이런 값들이 항상 같은 값을 가지도록 하는 coherence 방법이 필요하다. 이런 요구 조건을 만족하는 가장 간단한 방법은 통신에 의한 것이다. 즉, 하나의 처리기가 연결선의 가중치 새로운 값을 계산한 다음 이 값을 다른 처리기에게 통신을 통하여 알려주는 방법이며, 이런 통신은 각각의 뉴론들을 연결한 연결선마다 필요하다. 이런 종류의 통신 패턴을 all-to;all 개인통신[John 89]이라고 하는데, 이런 통신 형태에서는 각각의 처리기가 각각의 다른 처리기에게 고유한 메세지를 보내는 작업을 한다. 하지만 이런 형태 통신의 복잡도(complexity)는 각 처리기를 연결한 상호 연결망마다 틀리고, 또한 연결선의 수에 비례하여 커지기 때문에 많은 수의 뉴론으로 이루어진 신경망의 경우에는 매우 시간이 걸리는 작업이다.

본 연구에서 제안한 연결선 가능값의 일치(consistency)를 이루는 또 다른 방법은 재계산방법

(recomputation method)이다. 이 방법에서는 한번의 부가적인 all-to-all방송과 계산을 통하여 가중값의 일치를 유지한다. 이 방법에 대하여 좀더 자세히 설명하면 다음과 같다.

단계 $l+1$에 있는 뉴론 $n_k(l+1)$은 처리기 P_x에 사상되어 있고, 단계 l에 있는 뉴론 $n_i(l)$은 처리기 P_t에 사상되어 있다고 가정하자. 계산하고자 하는 연결선 가중치의 차이인 $\Delta W_{ki}(l, l+1)$는 처리기 P_t가 처리기 P_x에 저장되어 있는 $\delta_k(l+1)$값만을 알면 독립적으로 계산할 수 있다. 그 이유는 이미 $W_{ki}(l, l+1)$값이 처리기 P_t의 지역 메모리에 출력 연결선 가중값으로 저장되어 있기 때문이다. 마찬가지로 연결선 가중치의 차이인 $\Delta' W_{ki}(l, l+1)$는 처리기 P_x에서 처리기 P_t에 저장되어 있는 $a_i(l)$값을 알면 역시 독립적으로 계산할 수 있다. 그 이유는 앞의 경우와 마찬가지로 P_x는 지역 메모리에 $W_{ki}(l, l+1)$값을 입력 연결선 가중값으로 보관하고 있기 때문이다. 그런데 처리기 P_t가 계산한 $\Delta W_{ki}(l, l+1)$값과 처리기 P_x가 계산한 $\Delta' W_{ki}(l+1)$값은 같은 데이타를 이용하여 계산하였기 때문에 항상 같을 수 밖에 없다. 그러므로 두 처리기에 저장되는 연결선의 가중값은 항상 같은 값이 유지될 수 있다.

앞의 설명에서 처리기 P_t가 $a_i(l)$을 보낼때나 처리기 P_x가 $\delta_k(l+1)$을 전송할 때는 이 값을 all-to-all 방송의 형태로 전송하여야 한다. 그 이유는 단계 l에 있는 각 뉴론은 단계 $l+1$에 있는 모든 뉴론들과 연결되어 있기 때문이다. 〈알고리즘-3〉은 앞에서 설명한 연결선 가중값 수정 알고리즘을 나타낸 것이다.

〈알고리즘-3〉 분산 연결선 가중값 수정 알고리즘

```
for l = 1 to L do
    /* Updates Output Weight Vector */
    /* Broadcast and Receive δk(l+1) */
    for each neuron k∈Nt(l+1) do
        all-to-all-broadcast(δk(l+1));
    end_for
    /* Update wki(l,l+1) */
    for each neuron i∈Nt(l) do
        for k = 1 to nl+1 do
            Δwki(l,l+1) = η·δk(l+1)·ai(l);
            wki(l,l+1) = wki(l,l+1) + Δwki(l,l+1);
    end_for

    /* Updates Input Weight Vector */
    /* Broadcast and Receive aj(l-1) */
    for each neuron j∈Nt(l-1) do
        all-to-all-broadcast(aj(l-1));
    end_for
    /* Update wij(l-1,l) */
    for each neuron i∈Nt(l) do
        for j = 1 to nl-1 do
            Δwij(l-1,l) = η·δi(l)·aj(l-1);
            wij(l-1,l) = wij(l-1,l) + Δwij(l-1,l);
    end_for
end_for
```

그런데 출력 연결선 가중값의 수정을 2번째 패스에서 직접할 수 있기 때문에 〈알고리즘-3〉에 나와있는 오류의 all-to-all 방송은 생략될 수 있다. 〈알고리즘-4〉는 〈알고리즘-2〉와 〈알고리즘-3〉을 합쳐서 필요없는 all-to-all 방송을 생략한 향상된 알고리즘이다.

〈알고리즘-4〉 분산 오류 역전파와 연결선 가중값 수정 알고리즘

```
for l = L - 1 to 1 do
    /* Broadcast and Receive δk(l+1) */
    for each neuron k∈Nt(l+1) do
        all-to-all-broadcast(δk(l+1));
    end_for

    /* Compute δi(l) */
    for each neuron i∈Nt(l) do
        δi(l) = [Σ_{k=1}^{nl+1} δk(l+1)·wki(l,l+1)]·[ai(l)·(1-ai(l))]
    end_for

    /* Update Output Weight Vector wki(l,l+1) */
    for each neuron i∈Nt(l) do
        for k = 1 to nl+1 do
            Δwki(l,l+1) = η·δk(l+1)·ai(l);
            wki(l,l+1) = wki(l,l+1) + Δwki(l,l+1);
        end_for
    end_for

    /* Updates Input Weight Vector wij(l-1,l) */
    /* Broadcast and Receive aj(l-1) */
    for each neuron j∈Nt(l-1) do
        all-to-all-broadcast(aj(l-1));
    end_for
    /* Update wij(l-1,l) */
    for each neuron i∈Nt(l) do
        for j = 1 to nl-1 do
            Δwij(l-1,l) = η·δi(l)·aj(l-1);
            wij(l-1,l) = wij(l-1,l) + Δwij(l-1,l);
        end_for
    end_for
end_for
```

4. 성능 평가

본 장에서는 본 논문에서 제안한 신경망 구조를 분산 메모리 다중처리기 시스템에 사상하는 방법과 분산 역전파 알고리즘에 대한 성능을 예상되는 수행 속도와 필요한 메모리 사용량의 관점에서 분석하도록 한다.

4.1 시간 복잡도와 수행 속도 향상 비율

하나의 처리기를 사용하여 식(1)-(3)에 표현된 알고리즘을 이용하여 n개의 뉴론을 가지고 있는 어떤 신경망을 학습시키기 위하여 필요한 시간(T_1)은, t_i를 역전파 알고리즘의 i번째 패스를 수행하는데 필요한 시간이라고 하면 $T1=t_1+t_2+t_3$로 표시할 수 있다. 또한 M_a를 두 부동 소수점 숫자에 대한 한번의 곱셈과 한번의

덧셈을 하는데 걸리는 시간이라 가정하고, F를 한번의 sigmoid 함수를 계산하는데 걸리는 시간이라고 가정하면 1 패스에 대한 예상 수행 시간(t_1)은 식(4)와 같이 나타낼수 있다.

$$
\begin{aligned}
t_1 &= n\cdot(n\cdot M_a + F)\\
t_2 &= n\cdot(n\cdot M_a)\\
t_3 &= n\cdot(n\cdot M_a)\\
T_1 &= t_1 + t_2 + t_3\\
&= n\cdot(3\cdot n\cdot M_a + F)
\end{aligned}
\qquad (4)
$$

여기서 예상 수행 시간의 계산을 쉽게하기 위하여 신경망의 각 단계는 모두 n개의 뉴론으로 이루어 있다고 가정하도록 한다.

n개의 뉴론으로 이루어진 신경망 단계에 대하여 p-처리기 DMM상에서의 분산 역전파 알고리즘 수행시간(T_p)는 식(5)와 같이 나타낼 수 있다. 식(5)에서 t_1은 알고리즘-(1)을 수행하기 위하여 필요한 수행 시간이며, t_2은 알고리즘-(4)를 수행하는데 필요한 수행 시간이며, AAB(p)는 p-처리기 DMM상에서 all-to-all방송을 하기 위한 시간이다.

식(5)에서 $\frac{n}{p}$이 들어간 이유는 각각의 처리기가 신경망 어떤 단계에 있는 n개의 뉴론을 모두 같은 갯수 만큼 나누어서 계산하기 때문이다.

$$
\begin{aligned}
t_1' &= \frac{n}{p}AAB(p) + \frac{n}{p}\cdot(n\cdot M_a + F),\\
t_2' &= time\_to\_compute\_errors + time\_to\_update\_weights\\
&= \left(\frac{n}{p}\cdot AAB(p) + \frac{n}{p}\cdot(n\,M_a)\right) + \left(\frac{n}{p}\cdot(n\,M_a) + \frac{n}{p}AAB(p) + \frac{n}{p}\cdot(n\cdot M_a)\right)\\
&= \frac{n}{p}\cdot(2\,AAB(p) + 3\cdot n\,M_a)\\
T_p &= t_1' + t_2'\\
&= \frac{n}{p}\cdot(3\,AAB(P) + 4\,n\cdot M_a + F)\ .
\end{aligned}
\qquad (5)
$$

다음은 p-처리기 DMM상에서 all-to-all 방송을 하는 알고리즘과 예상되는 수행시간에 대하여 알아보도록 한다. 이러한 분석을 하기위해서는 각 처리기가 가지고 있는 통신 능력에 대한 가정이 먼저 있어야 하는데, 본 논문에서는 처리기가 어떤 한 순간에 한번의 송신과 수신을 할 수 있는 one-port 통신[John 89] 기능을 가정하도록 한다. 이때 처리기가 송신을 하고 수신을 하는 통신 포트는 서로 다를 수 있으며, 만약 한 순간에 송신이나 수신 중에서 하나의 동작만을 허용한다면 예상 수행 시간은 2배가 될 것이다. 한 처리기가 하나의 단위 메시지를 송신하거나 수신을 하는데 걸리는 시간을 C라고 가정하자. 이때 C는 필요한 통신 패스를 설치하는 시간과 단위 메시지를 실제 전송하는데 걸리는 시간을 모두 포함한 시간이며, 단위 메시지는 활성값이나 오류값을 나타내는 부동 소숫점 숫자를 나타내는 하나의 단어라고 가정한다.

이러한 one-port 통신을 하는 p-처리기 DMM에서 all-to-all 방송을 하는데 필요한 최소한의 시간은 각 처리기가 단위 시간에 하나의 메시지만을 받을 수 있기 때문에 (p-1) C이다. 이런 최소치는 ring 상호 연결망에서도 이룰 수 있기 때문에, one-port 통신을 가정하는 경우에는 ring보다 연결선을 더 가지고 있는 비싼 상호 연결망(예를 들면 complete connection나 hypercube, 혹은 mesh 등)을 사용할 필요가 없다. 만약 one-port가 아니라 한 순간에 여러개의 포트를 통하여 통신을 하는 d-ports 통신 처리기인 경우에는 많은 연결선을 가지고 있는 다른 상호 연결망이 더욱 좋은 성능을 낼 수 있을 것이다. 하지만 현재 발표된 거의 대부분의 처리기는 모두 one-port 통신 밖에 할수 없어서, d-ports 통신 가정은 현실 가능성이 없기 때문에 본 논문에서는 one-port 통신을 가정하였다.

〈알고리즘-5〉은 링(ring) 상호 연결망을 사용하는 p-처리기 DMM의 한 처리기 P_i에서 all-to-all 방송을 하기 위한 알고리즘이다. 이 알고리즘에서 p-처리기 DMM의 각 처리기는 0부터 p-1까지 고유의 처리기 번호를 가지고 있으며, 처리기 p_i는 처리기 $P_{(t+1) \bmod p}$와 처리기 $P_{(t-1) \bmod p}$와 직접 연결되어 있다고 가정한다. 〈알고리즘-5〉 링 상호 연결망으로 연결된 p-처리기 DMM에서의 all-to-all방송

```
/* Sends an element(A[0]) and Receives p-1 elements(A[1]-A[p-1]) */
for i = 0 to p-2 do
    parbegin
        send A[i] to p(t+1) .
        receive A[i+1] from p(t-1) .
    parend
end_for
```

식(4)와 식(5)을 사용하면 하나의 처리기를 사용할 때에 대한 p-처리기 역전파 학습 알고리즘 수행 시간 비율 $S(P)=T_1 / T_p$은 식-(6)와 같이 나타낼 수 있다.

$$
\begin{aligned}
S(p) &= \frac{T_1}{T_p}\\
&= \frac{n\,(3\,n\,M_a + F)}{\frac{n}{p}\cdot(3\cdot AAB(P) + 4\cdot n\cdot M_a + F)}\\
&= \frac{p\cdot(3\,n\,M_a + F)}{3\,C\cdot(p-1) + 4\,n\,M_a + F} \quad (\text{by } AAB(p) = C\cdot(p-1))\\
&= \frac{p\cdot(3\,n\,M_a + \theta\cdot M_a)}{3\cdot\Delta\,M_a\,(p-1) + 4\,n\,M_a + \theta\cdot M_a} \quad (\text{by } C=\Delta\,M_a\,,\ F=\theta\cdot M_a)\\
&= \frac{p\,(3\,n + \theta)}{3\cdot\Delta\,(p-1) + 4\,n + \theta}
\end{aligned}
\qquad (6)
$$

일단 어떤 응용에 대한 학습이 끝나면 그 응용에 대한 원하는 신경망 연결선의 가중값을 알 수 있기 때문에, 신경망은 전 방향 수행만을 하게되며, 이때의 하나의 처리기에 대한 p처리기 수행 시간 비율 S'(p)는 식-(7)과 같다.

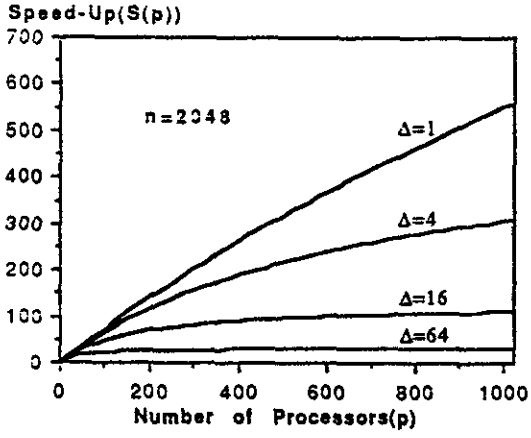
$$S'(p) = \frac{t_1}{t'_1} = \frac{p(n+\theta)}{\Delta(p-1)+n+\theta} \qquad (7)$$

p-처리기 DMM에서 만약 신경망의 크기가 p보다 매우 크면 S(p)는 (3/4)·이 되고 S'(p)는 p가 된다. 즉,

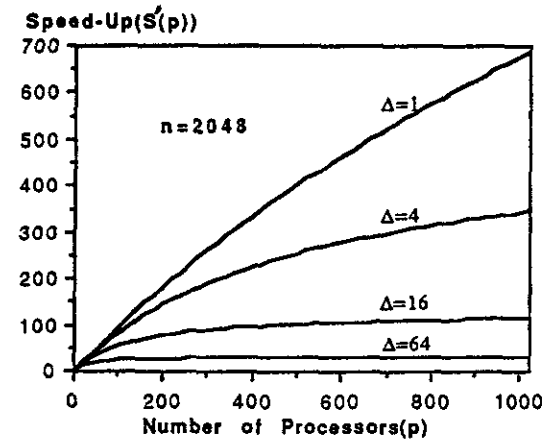
$$\lim_{n\to\infty} S(p) = \frac{3}{4}\cdot p$$
$$\lim_{n\to\infty} S'(p) = p$$

위의 식에서 p개의 처리기를 사용하는 경우에 S(p)이 p가 아니라(3/4)·p가 되는 이유는 같은 연결선 가중치에 대한 계산이 수정 패스에서 2번 수행되기 때문이다.

식 (6)-(7)에서 가장 중요한 파라메터는 통신/계산 비율인 Δ인데, 이 값은 보통 0.5에서 256사이의 값을 가지게 된다[Anna 89]. 신경망의 각 단계에 2048개의 뉴론이 있고 θ=40인 경우에 여러 종류의 Δ 값의 대한 예상되는 전체 학습 수행 속도 향상 비율(S(p))을 (그림 4) (a)에 나타냈으며, 같은 환경에서 전 방향 수행 속도 향상 비율(S'(p))를 (그림 4) (b)에 나타내었다.



(a) 분산 Backpropagation 알고리즘의 전체 속도 향상 비율

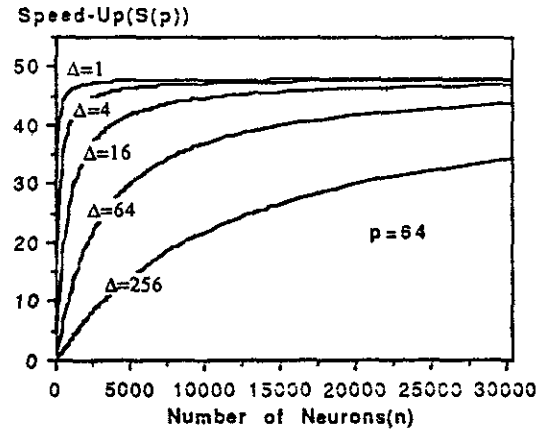


(b) 분산 Backpropagation 알고리즘의 전 방향 수행 속도 향상 비율

(그림 4) n=2048일때 수행 속도 향상 비율

(그림 4)를 통하여 알 수 있는 것은 본 연구에서 제안한 사상 방법과 분산 학습 알고리즘은 다단계 신경망 분산 시뮬레이션의 경우에 Δ 값에 대응하여 가장 가격-성능비가 좋은(cost-effective) 처리기의 숫자가 있다는 것이다. 그러므로 전체 처리기의 수가 이 숫자보다 작은 경우에는 처리기를 첨가할 때마다 처리기의 숫자에 거의 비례하여 성능이 향상되지만, 이 숫자보다 많은 경우에는 첨가한 처리기의 숫자에 비례하여 성능이 증가하지 않는다.

(그림 5)는 처리기의 숫자가 64로 고정되어 있는 경



(그림 5) 처리기의 숫자가 64로 고정된 경우의 수행 속도 향상 비율

우에 신경망의 크기에 대한 수행 속도 향상 비율이 나타낸 것이다. 이 그림에서 알수 있듯이 신경망의 크기가 작으면 수행 속도 향상 비율이 Δ 값에 민감한 반응을 받지만 신경망의 크기가 커지면 이 비율은 (3/4)·64=48로 접근함을 알 수 있다.

4.2 필요한 메모리 양 감소 비율

분산 메모리를 가지고 있는 다중 처리기 시스템들의 각 처리기는 보통 적은 양의 메모리만을 지역적으로 가지고 있기 때문에, 필요한 각 처리기의 지역 메모리의 양도 시뮬레이션 할 수 있는 신경망의 크기에 많은 영향을 미친다.

각 단계마다 n개의 뉴론을 가지고 있고 L개의 단계로 이루어져 있으며 완전히 연결된 다단계 신경망 구조를, 하나의 처리기에서 역전파 학습 알고리즘을 이용하여 학습시킬때 필요한 총 메모리양(M_1)은 다음과 같이 계산할 수 있다.

$$M_1 = space\_for\_a_i(l) + space\_for\_\delta_i(l) + space\_for\_w_{ij}(l-1,l)$$
$$= n \cdot L + n \cdot L + n^2 \cdot (L-1)$$

또한 같은 신경망을 본 논문에서 제안한 사상 방법과 분산 역전파 알고리즘을 이용하여 학습시킬 때 필요한 총 메모리 양(M_p)은 다음과 같이 계산할 수 있다. 이 때 연결선 가중값은 두곳에 저장되어 있기 때문에 좀더 많은 메모리를 필요로 한다.

$$M_p = space\_for\_a_i(l) + space\_for\_\delta_i(l) + space\_for\_w_{ij}(l-1,l) + space\_for\_received\_datum\_per\_a\_processor$$
$$\approx n \cdot L + n \cdot L + 2 \cdot n^2 \cdot (L-1) + p \cdot n$$
$$= n \cdot (2 \cdot L + p) + 2 \cdot n^2 \cdot (L-1)$$

그런데 필요한 메모리양 M_p은 사상 방법에 의하여 p-처리기 DMM에서 완전히 p로 나뉘어서 저장된다. 그러므로 하나의 처리기를 사용하는 경우에 필요로 하는 메모리양에 대한 p-처리기 DMM의 한 처리기에 필요한 메모리 양의 비율(M(p))은 다음과 같이 계산할 수 있다.

$$M(p) = \frac{M_1}{\frac{1}{p} \cdot M_p} = \frac{p \cdot (2 \cdot n \cdot L + n^2 \cdot (L-1))}{n \cdot (2 \cdot L + p) + 2 \cdot n^2 \cdot (L-1)}$$

만약 신경망의 크기가 매우 커지면, 이 비율은 p-처리기 DMM의 경우에 p/2가 된다. 즉,

$$\lim_{n \to \infty} M(p) = \frac{p}{2}$$

그러므로 완전히 연결된 다단계 신경망을 분산 역전파 알고리즘을 사용하여 p-처리기 DMM 학습시키면, 학습 속도도 빨라질 뿐만 아니라 DMM의 각 처리기는 작은 양의 메모리만을 가지고도 큰 신경망을 시뮬레이션할 수 있음을 알 수 있다.

5. 결론 및 앞으로의 연구 방향

인간 뇌의 정보 처리 방법을 모방하는 신경망 모델은 지금까지 인공 지능 분야에서 쉽게 풀수 없었던 여러 문제를 쉽게 해결할 수 있는 방법을 제공하기 때문에 최근에 많은 연구가 이루어지고 있다. 이런 신경망 모델은 구조가 단순하고 전문가 시스템과 같이 지식을 명시적으로 입력시킬 필요가 없다는 장점을 가지고 있는 반면에 많은 뉴론과 연결선을 가지고 있는 신경망 모델의 경우, 학습 자체도 많은 시간이 소요되기 때문에 새로운 신경망 모델을 개발하거나 응용 분야를 개발하는 것이 어렵다는 단점도 가지고 있다.

본 논문에서는 이런 문제점을 해결하기 위하여 현재 가장 많이 사용하고 있는 다단계 신경망 모델을 분산 메모리 다중 처리기 시스템에 사상시키는 방법과, 이 사상 방법에 기초한 분산 역전도 학습 알고리즘을 제안하고 그 성능을 계산에 의하여 분석하였다. 이 분석에 의하면 제안된 사상방법과 역전도 학습 알고리즘은 p개의 처리기를 사용하여 많은 수의 뉴론을 가지고 있는 신경망을 분산 학습시키는 경우에, 수행 시간은 하나의 처리기를 사용하는 경우의 $\frac{3}{4} \cdot p$만큼의 시간만이 필요하며, 또한 한 처리기는 하나의 처리기를 사용하여 학습 시키는 경우에 비하여 $\frac{p}{2}$만큼의 메모리만을 가지고 있으며 되기 때문에 큰 신경망을 빠른 속도로 학습 시킬 수 있다.

현재 본 논문에서 제안한 방법을 386 PC를 호스트로 하고 여러개의 트랜스퓨터를 연결한 다중 처리기 시스템을 이용하여 구현하고 있다. 특히 이 시스템은 사용자 인터페이스를 원도우에 기초한 메뉴 처리 방식으로 하였기 때문에 신경망 모델의 여러 파라메터를 대화형

으로 쉽게 바꿀수 있는 기능도 가지고 있으며, 역전파 알고리즘을 병렬로 수행할 수 있기 때문에 빠른 신경망 학습을 제공할 수 있다.

앞으로 더 연구하여야 할 방향으로는 본 논문에서 제안한 방법을 실제 실험을 통하여 성능을 분석하는 것과, 역전파 학습 알고리즘 이외의 다른 학습 알고리즘에 대한 병렬 학습 방법을 개발하는 것 등을 들 수 있다.

참 고 문 헌

[Anna89] M. Annaratone, C. Pommerell, and R. Ruhl, "Interprocessor Communication Speed and Performance in Distributed-Memory Parallel Processors," Proc. of the 16th Annual Int'l Symp. on Computer Architecture, pp. 315-324, 1989.

[Cook88] J. Cook and J. Gilbert, "Parallel Neural Network Simulation using Sparse Matrix Techniques," Microprocessing and Microprogramming 24, pp. 621-626, 1988.

[Darp88] DARPA Neural Network Study, AFCEA International Press, 1988.

[Depr89] E. Deprit, "Implementing Recurrent Back-Propagation on the Connection Machine," Neural Networks, vol. 2, pp. 295-314, 1989.

[Forr87] B. M. Forrest et al., "Implementing Neural Network Models on Parallel Computer," The Computer Journal, Vol. 30, No. 5, pp. 413-419, 1987.

[Ghos89] J. Ghosh and K. Hwang, "Mapping Neural Networks onto Message-Passing Multicomputer," Journal of Parallel and Distributed Computing, April 1989.

[John89] S. L. Johnsson and C. T. Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes," IEEE Trans. on Computers, Vol. 38, No. 9, pp. 1249-1268, 1989.

[Kung88] S. Y. Kung, J. N. Hwang, "Parallel Architecture for Artificial neural Nets," Proc. of IEEE 2nd Int'l Conf. on Neural Networks, Vol. II, pp. 165-172, 1988.

[Mill89] J. Millan and P. Bofill, "Learning by Backpropagation: A Systolic Algorithm and Its Transputer Implementation," Neural Networks, Vol. 1, No. 3, pp. 119-137, 1989.

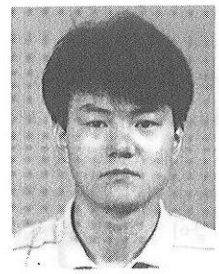
[Pome88] D. A. Pomerleau et al., "Neural Network Simulation at Warp Speed: How We Got 17 Million Connections Per Second", Proc. of IEEE 2nd Int'l Conf. on Neural Networks, Vol. II, pp. 143-150, 1988.

[Rume87] D. E. Rumehalt, J. L. McClelland et al., Parallel Distributed Processing, vol. 1., MIT Press.

[Trel89] P. Treleaven, Neurocomputers, Research Note 89/8, Dept. of CS, Univ. College Londen, 1989.

[Work88] G. A. Works, "The Creation of Delta: A New Concept in ANS Processing," Proc. of IEEE 2nd Int'l Conf. on Neural Networks, Vol. II, pp. 159-164, 1988.

[Yoon90] H. Yoon and J. H. Nang, "Multilayer Neural Networks on Distributed-Memory Multiprocessor," Proc. of Int'l Neural Network Conference, Paris France, July 1990.



낭 종 호

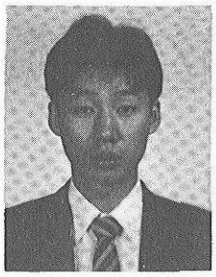
1986년 서강대학교 전자계산학과 졸업

1988년 한국과학기술원 전산학과 석사학위 취득

1988년 제 7회 정보과학 논문 경진 대회 대학원부 우수상 수상

1988년 현재 한국과학기술원 전산학과 박사과정 재학중

주 관심분야 : Neural Network, Functional Ligic Language, AI Machine, Multiprocessor Architecture

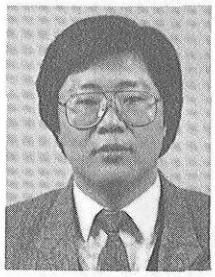


최 선 민

1989년 홍익대학교 전자계산학과 졸업
1989년 현재 : 한국과학기술원 전산학과 석사과정 재학중
주 관심분야 : Neural Network

이 상 훈

1989년 부산대학교 전자계산기공학과 졸업
1989년 현재 : 한국과학기술원 전산학과 석사과정 재학중
주 관심분야 : Neural Network



윤 현 수

1979년 서울대학교 전자공학과 졸업
1981년 한국과학기술원 전산학과 석사학위 취득
1981－1984 삼성전자연구원
1988년 오하이오 주립대학 전산학 박사학위 취득
1988－1989 AT & T Bell Labs. 연구원
1989년－현재 한국과학기술원 전산학과 조교수
주 관심분야 : Parallel Computer Architecture, Interconnection Network, Protocol Enginnring, Neural Network

맹 승 렬

1977년 서울대학교 전자공학과 졸업
1979년 한국과학기술원 전산학과 석사학위 취득
1984년 한국과학기술원 전산학과 박사학위 취득
1984년 현재 한국과학기술원 전산학과 조교수
주 관심분야 Parallel Processing, Data Flow Machine, Neural Netwokr, Vision Architecture, Object-Oriented Language Architecture