論文94-31B-1-11

병렬 자구성 계층 신경망(PSHNN)의 구조

(Architectures of the Parallel, Self-Organizing Hierarchical Neural Networks)

尹 寧 佑*, 文 部 鉉*, 洪 大 植*, 康 昌 彦*

(Young Woo Yun, Tae Hyun Moon, Dae Sik Hong and Chang Eon Kang)

要 約

병렬 자구성 계층 신경망(Parallel, Self-Organizing, Hierarchical Neural Networks : PSHNN)은 기존의 다층 신경망(Multi-layer Neural Networks)과 같이 여러개의 층으로 이루어져 있다. PSHNN이 기존의 다층 신경망과 다른 점은 기존의 다층 신경망의 경우 각 층이 직렬로 연결되어 있는 점에 반하여 각 층이 병렬로 구성되어 있다는 것이다. 또한 각 층의 끝 부분에는 에러감지 기법을 이용해서 신호를 축출하는 기능을 가지고 있으며, 이렇게 축출된 신호는 비선형 변환을 거쳐서 다음 층으로 보내지게 된다. PSHNN의 성능을 기존의 다층 신경망과 실험을 통해서 살펴보면, 비슷한 정도의 시스템 복잡도를 가질때, 인식면에서 비슷한 정도의 결과를 주며 학습시간은 훨씬 짧은 것으로 나타난다.

Abstract

A new neural network architecture called the Parallel, Self-Organizing Hierarchical Neural Network (PSHNN) is presented. The new architecture involves a number of stages in which each stage can be a particular neural network (SNN). The experiments performed in comparison to multi-layered networks with backpropagation training and indicated the superiority of the new architecture in the sense of classification accuracy, training time, parallelism.

Ⅰ. 서 론

신경 회로망 연구 분야 중 가장 중요한 내용들은 시스템의 성능의 향상, 시스템의 복잡도, 학습시간과 검사시간, robustness, 시스템의 범용성 등이 있다. 이 논문에서는 새로운 형태의 신경망인 병렬 자구성 계층 신경망 (Parallel, Self-Organizing Hierarchical Neural Networks: PSHNN)을 소개한다.

현재 신경 회로망 중 가장 각광을 받고 있는 것은 역전파 기법을 이용한 다층 신경망(Multi Layer Neural Networks : MNN)이다. 이 신경망은 Werbors에 의해 처음 제안되었고 그 이후에 많은 사람들에 의해 새롭게 개발되어 왔으며, 또한 여러 응용분야에 쉽게 적용시킬 수 있음을 알 수 있었다.[10-12, 13] 하지만 MNN은 학습시간이 길다는 점과 또한 국부 최소값에서 벗어나지 못할 가능성이 있다는 등의 단점이 있다.

새로 제안되는 PSHNN은 MNN과 유사한 구조를

*正會員, 延世大學校 電子工學科
(Dept. of Elec. Eng., Yonsei Univ.))

接受日字 : 1993年 6月 29日

가지고 있다. 즉 PSHNN은 여러 작은 신경망을 복합해서 큰 신경망을 이루며 그 작은 신경망은 MNN에서 1 계층의 신경망과 같은 개념으로 사용된다. 하지만 PSHNN이 기존의 MNN과 다른 점은 PSHNN에서는 각 계층 신경망(Single- layer Neural Networks : SNN) 끼리 서로 무관하다는 것이다. 즉 MNN에서는 각 계층이 서로 직렬로 연결되어서 신호들이 서로 유관하게 처리 되지만 PSHNN에서는 병렬로 구성되기 때문에 SNN 끼리 서로 무관하게 동작하게 되는 것이다. 또다른 특색은 각 SNN 사이에 비선형 변환(nonlinear transform) 기능을 두고 사용하며, 각 SNN의 끝단에는 신호를 축출하는 기능을 갖도록 하는 것이다.

이 논문에서는 PSHNN의 구조를 설명하고 특히 에러 축출방법에 대해 상세히 논하고, PSHNN의 성능을 MNN과 비교해 본다. 먼저 Ⅱ장에서는 신경회로망에서 숫자를 표현하는 여러가지 기법에 대해서 논한다. Ⅲ장에서는 PSHNN의 구조와 축출방법에 대해 보여주며 Ⅳ장에서는 축출방법에 따른 PSHNN의 성능을 분석하고 MNN의 성능과 비교 분석하며 마지막 Ⅴ장에서 결론을 맺는다.

Ⅱ. 정수의 표현

신경회로망 구성시 신경 소자나 가중치의 상태들을 나타내는 방법은 매우 중요한 부분중의 하나이다. 특히 인식 분야에서는 입력 신호나 출력 신호의 표현 방법에 따라 똑같은 시스템에서도 여러가지 다른 형태의 성능을 보여주는 것을 알 수 있다. Handcock [9], Taked와 Goodman [11] 등이 여러가지 숫자표현 방법에 대해 제시하고 있다. 이 장에서는 신경회로망에서 입력 신호로 많이 사용하는 이진수의 표현법에 관해 논하기로 한다. 먼저 gary 코드의 성능을 이진수와 비교하고 이진수의 비선형 변환에 관해 논한다.

정수를 표현하는데 가장 많이 쓰이는 방법은 기존의 이진수를 그대로 사용하는 것이다. 이렇게 이진수로 표현하게 되면 다른방법(예로서 simple sum 방법, temperature 방법)에 비해 경제적으로는 매우 유리하지만 에러가 발생했을 경우 매우 치명적인 타격을 입게 되는 단점이 있다. [9] 또 하나의 중요한 단점은 정수를 사용할 때 이진수를 사용하면 비슷한 두 값이 매우 다르게 표현될 수 있다는 것이다. 예로서 정수 127과 128을 생각해 보자. 두 값은 실제로 gray level에서 1의 차이를 보여주고 있다. 이 두 정수를 이진수로 표현하면 01111111과 10000000로 각각 표현되며, 이 경우는 Hamming 거리로 최고인 8의 차이를 보여주게 된다.

이러한 현상은 특히 패턴인식 등에서 시스템의 성능을 저하시키는 요소가 될 수 있다.

Gray 코드를 사용하면 위와 같은 이진수가 갖고 있는 문제점을 비트 수를 증가시키지 않고 어느정도 해결할 수 있다. Gray 코드는 다음과 같이 정의된다. [2]

$$g_1 = b_1,\ g_k = b_k \oplus b_{k-1},\ k \geq 2 \tag{1}$$

여기서 g_i는 gray 코드의 각 비트를 의미하며, b_i는 이진수의 각 비트를 의미한다. 또한 ⊕는 exclusive-or 연산을 의미한다.

이제 정수를 이진수와 gray 코드로 각각 바꿀때 어느 것이 원래의 정보를 더 잘 표현하고 있는지를 class separability의 개념으로 비교해 보기로 하자. 즉, 이진수와 gray 코드를 각각 사용해서 정수를 표현하는 경우, 어느 방법이 원래의 정수가 가지고 있는 정보를 잘 표현해 주는가를 살펴보는 것이다.

먼저 L개의 class가 있다고 하고 각각의 class는 N개의 샘플로 구성되어 있으며, 각 샘플은 n 비트의 이진수로 표현된다고 가정하며, 각 class는 A_i, i=1, …, L로 나타낸다. x와 y를 각각 하나의 샘플들이라고 하면 두 샘플간의 거리는 Hamming 거리로 다음과 같이 표현된다.

$$\begin{aligned} d(x,y) &= |x-y| \\ &= |x_0 - y_0| + \cdots |x_n - y_n| \end{aligned} \tag{2}$$

이때 각 class의 With-In Class-Distance를 다음과 같이 정의한다.

$$DI(A_i) = \frac{2}{N(N-1)} \sum_k \sum_l d(x_k, x_l), \qquad x_k x_l \in A_i \tag{3}$$

여기서 $k < l$이다. 또한, 비슷하게 Between Class Distance는 다음과 같이 정의된다.

$$DI(A_i, A_j) = \frac{1}{N^2} \sum_k \sum_l d(x_k, y_l), \qquad x_k \in A_i, y_l \in A_j \tag{4}$$

위의 거리로 class seperability meajure J를 다음과 같이 정의하자.

$$J = \frac{2}{L-1} \frac{\sum_k \sum_l DB(A_k, A_l)}{\sum_i DI(A_i)}, \qquad k < l \tag{5}$$

J가 큰 값을 나타내면 낼수록 좋은 class seperability를 나타낸다고 볼 수 있다.

〈표1〉에서는 2개의 class가 있을때, gray 코드와 이진수의 class seperability를 여러가지 경우에서 보여주고 있다. 첫번째의 경우는 이항 분포에서 각 class 당 200개의 데이타를 뽑아서 사용한 것이고, 두번째는 400개씩 뽑았을때, 그리고 3번째와 4번째는 정규 분포에서 각각 200개, 400개 씩 뽑아서 binary로 변환시키고 gray 코드로 변환시켜서 seperability를 구한 것이다.

표 1. 이진수와 gray 코드의 class seperability
Table 1. Comparison between the binary and the gray code.

데이타		Within Distance(DI)	Between Distance(DB)	Seperability
이항분포 200	binary	3.38072	4.48310	1.32608
	Gray	2.51590	3.72440	1.48035
이항분포 400	binary	3.13564	4.71947	1.50510
	Gray	2.35675	4.49084	1.90034
정규분포 200	binary	3.47043	4.50125	1.29703
	Gray	3.40225	4.41360	1.29726
정규분포 400	binary	3.49197	4.50159	1.28912
	Gray	3.11378	4.12045	1.32330

위의 결과를 살펴보면 DB와 DI 모두 gray 코드의 경우가 이진수의 경우보다 작게 나오고 있으며, 궁극적으로 class seperability J는 gray 코드의 경우가 이진수의 경우 보다 항상 크게 나타나고 있음을 알 수 있다. 즉 gray 코드를 사용하면 이진수를 사용한 경우 보다 정수로 표현된 것을 좀 더 정확히 표현해 줄 수 있다는 것을 알 수 있다.

신경회로망 분야나 패턴인식 분야 등에서 이진수를 이용해서 정보를 표현할 때 발생하는 문제점들 중 또 다른 하나는 서로 매우 다른 정보들이 비슷하게 표현될 수 있다는 것이다. 실제로 이러한 문제를 해결하기 위해서는 temperature 코딩이나 simple sum 코딩 방법을 사용하는 방법이 있다. 하지만 이러한 방법들은, 비트 수의 면으로 봤을 때, 경제적으로 매우 비효율적인 방법이다.

우리는 여기서 이진수의 비트 수를 그대로 둔 채 위에서 언급한 문제를 해결하기 위한 방법으로 비선형 변환 방법을 제안한다. 즉 이진수로 표현된 정보들 중 서로 매우 구분하기가 힘든 정보들이 있을 때 그것들을 비선형 변환을 통해서 쉽게 구분할 수 있도록 해주자는 것이다.

비선형 변환은 여러가지 방법들이 있지만 가장 간단한 방법중의 하나는 이진수로 표현된 벡터를 기존의 선형 변환-즉, DFT, DCT, 혹은 RDFT [6] 등-을 이용해서 변환 시킨뒤 그 결과를 hard limiter 등을 통해서 다시 이진수로 바꾸어 주는 방법이다. 기존의 여러가지 방법들 중 이 논문에서는 RDFT를 선형변환으로 사용하기로 한다.

패턴 인식 문제 등에서 하나의 신호 영역에서 인식률이 안좋은 경우, 변환을 사용해서 신호를 다른 영역의 값으로 바꾸어 준 뒤 다시 인식할 경우 종종 좋은 결과를 얻는 것을 알 수 있다. 즉 시스템의 가장 적합한 feature를 찾는 것이 패턴 인식 문제등에서는 매우 중요하며, 이때 다른 영역으로 신호를 변환시키는 방법은 가장 적합한 feature를 찾는 방법들 중 매우 유용한 방법이다. 이 논문에서는 비선형 변환을 이용해서 신경 회로망의 입력 신호들의 영역을 임의로 변환 시키고자 하는 것이다.

Ⅲ. PSHNN

병렬 자구성 계층 신경망(Parallel, Self-Oraganizing, Hierarchical Neural Networks)을 제안하는 목적은 신경망 연구분야에서 도출되는 여러가지 문제점들을 해결하기 위한 것이다.

먼저 전체 시스템의 복잡도를 줄여 보자는데 목적이 하나있다. 또한 성능을 향상시키며, 다른 많은 신경망의 문제점으로 대두되고 있는 국부 최소값에서 벗어나고자 하며, 학습시간과 검사시간을 줄여 보고자 하며, 마지막으로 다른 계층 신경망과 비교하여 좀 더 병렬적으로 시스템을 구성해 보자는데 목적을 두고 있다.

PSHNN의 계층 수는 환경에 따라 스스로 적절히 변화시킨다. 이것이 PSHNN의 가장 큰 특징이 되며 여기서 각 계층은 임의의 신경회로망을 이용하여 구성할 수 있다. 일반적으로 계층 수는 복잡한 데이터를 사용할 경우 많아지고, 비교적 간단한 데이터를 사용할 경우 적어진다.

이 논문에서 각 계층을 구성하기 위해서 실제로 사용한 신경회로망은 가장 간단한 형태의 Delta-Rule을 이용한 신경망을 사용하여 구현한다. 그림 1에서 PSHNN과 일반적인 다층신경망(Multi-Layer Neural Networks :MNN)의 구조적 차이를 보여주고 있다.

그림 1에서와 같이 PSHNN의 두 계층(SNN) 사이에는 비선형 변환과정이 들어 있으며 여기서는 윗단의 SNN에서 축출된 신호들이 비선형 변환되어 다음단의 SNN으로 보내지게 된다. 다음 단의 SNN에서 신호를 받아들일 것인지 혹은 축출할 것인지를 결정하는 방법은 여러가지가 있다. 이 논문에서 사용하는 방법은 먼저 SNN을 어느 정도 학습 시킨후 제대로 학습된 것과 제대로 학습되지 않는 것들을 분류한 뒤에 그것으로 축출범위를 결정한다. 그 후 신호를 다시 그

SNN에 집어 넣어서 그 결과가 받아들이는 부분에 속하면 통과시키고 그렇지 않으면 축출하는 방법을 택한다.

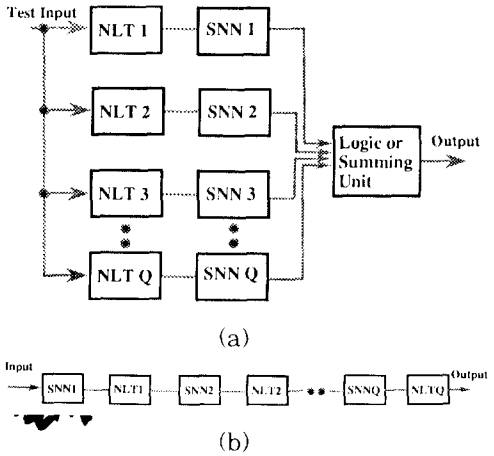


그림 1. PSHNN과 MNN의 구조 비교
(a) PSHNN (b) MNN

Fig. 1. Block Diagram for PSHNN and MNN.
(a) PSHNN Networks,
(b) Cascaded Multistage.

실제로 그림 2에서 PSHNN을 학습시키는 방법에 관해 보여주고 있으며 그림 3에서는 알고리듬으로 보여주고 있다.

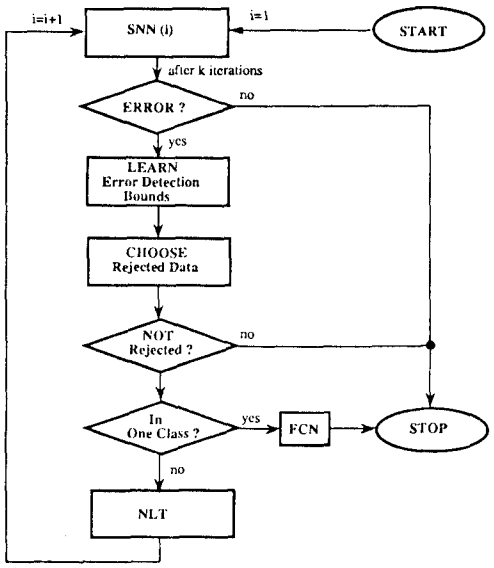


그림 2. PSHNN의 학습과정의 Flow Chart

Fig. 2. Flow chart of the training procedure of the PSHNN.

<PSHNN의 학습과정>

가정 : SNN의 최대 반복 학습 수 (iteration 수) = k
초기화 : i=1

1. SNN(i)를 최고 k번까지 학습시킨다.

2. SNN(i)에 들어가는 각 입력 신호에 대해 출력을 검사한다.
   1) 출력에 에러가 없는 경우 → 학습을 멈춘다.
   2) 출력에 에러가 있는 경우 → 에러 감지 구역을 설정하고 에러가 있는 신호를 축출한다.

3. 과정 2에서 축출된 신호들을 검사한다.
   1) 만일 과정 2에서 축출된 신호들이 전부 1 class에 포함되어 있으면 final class number(FCN)를 정하고 학습을 멈춘다.
   2) 그렇지 않으면 다음 과정으로 간다.

4. 1) 과정 2에서 축출된 신호를 비선형 변환을 취한다.
   2) i=i+1
   3) 과정 1로 간다.

그림 3. PSHNN의 학습과정

Fig. 3. Training procedure of the PSHNN.

여기서 최대학습 수를 k로 고정하는 것은 신경회로망이 과학습되는 것을 방지하기 위한 것이며 또한 국부 최소값에 빠지는 것을 막는 효과를 보기 위한 것이다. 그럼으로 인해 학습시간도 단축할 수 있다. 이와 같이 PSHNN이 구성되면 각 SNN은 weight matrix, 축출 영역, 그리고 비선형 변환 시스템을 가지고 있게 된다.

학습과정이 모두 끝이나면 PSHNN은 신호의 특성에 따라 여러 개의 SNN으로 구성된다. 검사과정은 기존의 MNN의 검사과정과 같은 형태로 진행된다. 그림 4에서 그 과정을 보여준다.

<검사 과정>

초기화 : i=1

1. 검사 신호를 SNN(i)에 입력 시킨다.

2. SNN(i)의 출력단에서 출력된 신호가 축출지역에 들어 있는지를 판단한다.
   1) 축출지역에 들어 있으면,
      a) SNN(i)가 마지막 SNN이면, PSHNN의 출력값을 FCN으로 한다. 검사과정을 마친다.
      b) SNN(i)가 마지막 SNN이 아니면, 과정 3으로 간다.
   2) 축출지역에 들어 있지 않으면, PSHNN의 출력값을 SNN(i)의 출력값으로 하고, 검사과정을 마친다.

3. 검사 신호를 비선형 변환 시킨다.

   i=i+1로 한 후 과정 1로 간다.

그림 4. PSHNN의 검사과정

Fig. 4. Testing procedure of the PSHNN.

즉 학습된 신경회로망에 신호를 넣으면 그 신호가 각 계층을 거쳐서 마지막 계층의 출력이 입력 신호에 대한 결과가 된다. 즉 신호가 순서적으로(sequential) 계층을 통과하게 된다. 하지만 PSHNN에 간단한 논리회로를 부가하면 검사과정을 완전히 병렬로 처리할 수 있다. 이것은 학습과정에서 각 SNN을 독립적으로 학습을 시켰기 때문에 검사 신호를 모든 SNN에 동시에 입력 시킨 후 그 각 SNN들의 출력값들 중에서 타당한 것 하나만 선택하면 된다. 그림 5에서는 그렇게 구성된 PSHNN의 구조를 보여주고 있으며 그림 6에서는 논리회로 부분을 보여주고 있다. 이 논리회로의 기본적인 동작 개념을 간단히 설명하고 상위 SNN의 출력값이 그 SNN의 축출 지역에 속해 있지 않으면 그 SNN의 출력이 PSHNN의 값이 된다. 만일 그 SNN의 출력값이 축출지역에 속하게 되면 그 다음 단의 SNN의 출력 값을 대신 사용하게 한다.

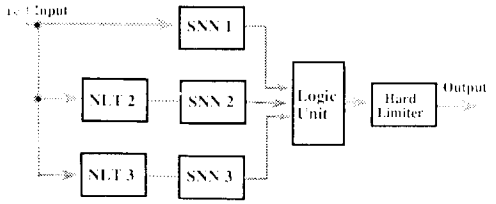


그림 5. PSHNN의 검사과정의 구조
Fig. 5. block diagram of the testing procedure with the PSHNN.

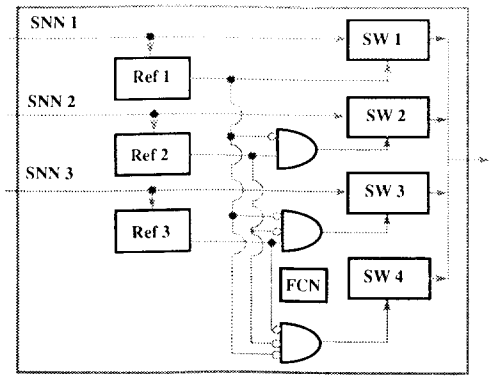


그림 6. PSHNN의 검사 구조의 출력을 조절해 주는 논리 회로
Fig. 6. Logic unit of the PSHNN.

2. 축출방법

축출 영역을 결정하는 원래의 목적은 한 SNN을 학습시킬때 학습을 방해하는 신호들을 골라내기 위한 것이다. 즉 하나의 SNN을 어느 정도 학습시킨뒤 그 SNN을 학습시키는 신호들을 검토해서 그 신호가 "학습에 도움이 되는 경우"는 그대로 두고 "방해가 되는 경우"는 축출한뒤 축출된 신호들만을 이용해서 새로운 SNN을 구성해 학습을 시키고자 하는 것이다. 이 경우 어떻게 "학습에 도움을 주는 경우"와 아닌 경우를 구분하느냐가 매우 중요한 문제로 대두된다. 실제로 구분하는 방법은 SNN으로 쓰이는 신경망의 종류에 따라 또한 신호의 종류에 따라 여러가지로 나타날 수 있다.

여기서는 SNN을 Delta Rule을 이용한 한 계층의 신경망으로 사용했을 때 신호의 축출 방법을 고려하기로 한다. 이때 SNN의 끝 단에는 Sigmoid함수를 사용하며, 출력값은 (0, 1) 사이의 값을 갖도록 한다.

한 계층의 신경망을 학습시키는 경우, 우리는 제대로 학습된 신호와 제대로 학습이 되지 않는 신호를 구별할 수 있다. 하지만 학습이 끝난 뒤에 검사 과정 중에는 실제로 그 신호의 출력값이 제대로 된 것인지 아닌지를 판단하기가 매우 어렵다. 왜냐하면 학습과정 중에는 원하는 출력값을 알기 때문에 신호의 출력값의 타당성을 판단할 수 있지만, 검사과정에는 그럴 수 없기 때문이다. 그래서 우리는 학습과정과 검사과정 모두에서 일관성 있게 신호의 타당성을 검토할 수 있는 방법을 마련해야 한다. 그 한가지 방안은 학습과정 중에 학습 신호의 출력값들을 이용해서 신호의 타당성을 검토할 수 있는 영역을 설정하는 것이다. 그렇게 설정된 영역에 따라서 학습과정과 검사과정 모두에서 신호의 출력을 얻은 뒤, 그것의 타당성을 검토한 후 축출하거나 그대로 사용하거나 한다.

이제 이러한 영역을 얻는 몇가지 방법에 관해 논해보기로 하자.

1) Matching 방법

학습과정과 검사과정 모두에서 에러를 검사하는 가장 간단한 방법은 바로 table-look-up 방법이다. 즉 이미 결정되어 있는 원하는 출력값들과 신호의 출력값을 비교해서 일치되면 그대로 사용하고 일치되지 않으면 축출하는 방법이다. 이 방법을 사용하면 제대로 학습된 신호는 절대로 축출되지 않는다. 하지만 잘못 학습된 신호도 축출되지 않을 가능성은 있게 된다. 이 방법을 사용하기 위해서는 각 SNN에서 원하는 출력값들 모두를 기억하고 있어야 한다. 그리고 학습이 끝난 뒤 기억된 값과 신호출력값을 비교해서 축출여부를 판단하게 된다. 실제 많은 경우, 특히 패

턴인식 문제에서, n개의 class가 있을 때 원하는 출력값을 R^n의 base 벡터로 이용을 하고 있다. 즉 각 class의 출력값을 다음과 같이 표현해서 사용하고 있다.

$$\begin{aligned}&\text{class } 1 \rightarrow (1,\ 0,\ 0,\ \cdots\ 0)^t\\&\text{class } 2 \rightarrow (0,\ 1,\ 0,\ \cdots\ 0)^t\\&\qquad\vdots\\&\qquad\vdots\\&\qquad\vdots\\&\text{class } n \rightarrow (0,\ 0,\ 0,\ \cdots\ 0,\ 1)^t\end{aligned}$$

이런 경우 우리는 각 SNN에서 모두 원하는 출력값을 다 기억하지 않아도 쉽게 출력값의 타당성을 판단할 수 있다. 즉 신호의 출력 중에 "1"이 1개 이면 올바른 출력이고 그렇지 않으면 잘못된 신호로 판단하면 된다.

2) 에러 영역

에러 영역은 출력값이 이 구역에 들어 갔을 경우 잘못된 신호라고 간주하고 축출하는 구역이다. 즉 에러 구역을 통해서 잘못된 신호를 가능하면 많이 축출하고 잘된 신호는 축출되지 않도록 구성하는 것이 목적이다. 에러 구역은 학습시에 잘못 학습된 신호들을 모아서 그 신호들로부터 잘못 학습된 신호들의 구역을 설정하는 방법이다.

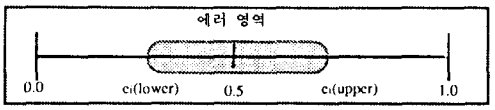


그림 7. 에러 영역
Fig. 7. Error region.

그림 7에서 하나의 출력 노드에 대한 에러 구역의 한 예를 보여준다. 즉 어떤 신호의 출력이 그림 7의 빗금친 부분에 속하게 되면 그 신호는 축출된다. 이것은 주로 출력값이 1이 아니면 0에 가까운 값이기 때문에 0.5에 가까운 값은 에러일 가능성이 높기 때문이다. 출력노드의 갯수가 n일 때, 에러 구역은 다음과 같이 정의된다.

$$\text{에러 영역 (EB)}=\left[\begin{pmatrix}e_1(\text{upper})\\e_1(\text{lower})\end{pmatrix},\begin{pmatrix}e_2(\text{upper})\\e_2(\text{lower})\end{pmatrix},\cdots,\begin{pmatrix}e_n(\text{upper})\\e_n(\text{lower})\end{pmatrix}\right] \quad (6)$$

위의 에러 영역을 이용해서 신호의 축출 여부를 결정할 경우 학습시에는 모든 잘못 학습된 신호가 축출되며, 제대로 학습된 신호가 축출될 가능성도 꽤 높다. 또한 검사시에는 잘못된 신호가 축출되지 않을 가능성도 물론 존재한다. 그림 8에서는 에러 영역을 정의 해주는 알고리듬을 보여준다.

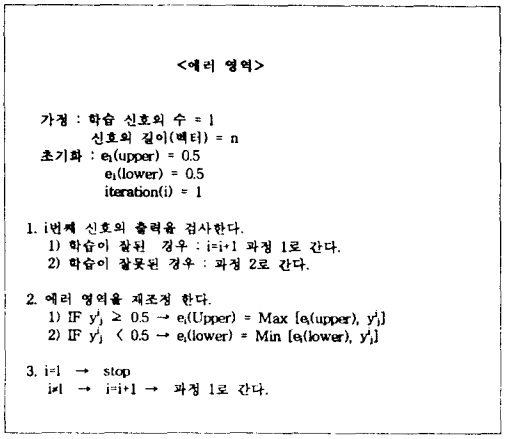


그림 8. 에러 영역을 설정하는 방법
Fig. 8. The method of deciding error region.

3) 무-에러 영역

앞 절에서는 학습시에 잘못 학습된 신호들을 이용해서 에러영역을 만들어서 에러를 축출하였다. 비슷한 방법으로 학습이 잘된 신호들을 이용해서 에러가 없는 영역을 구성할 수 있다. 에러 영역과는 반대로 이 경우에는 무 에러 영역에 시스템의 출력값이 들어오게 되면 그 신호는 그대로 통과 시키고, 그렇지 않을 경우는 축출하는 방법을 사용한다. 이 영역을 사용하게 되면 학습시에는 제대로 학습된 모든 신호가 그대로 통과하게 되지만 잘못 학습된 신호도 역시 통과될 가능성이 있으며, 검사시에는 제대로 학습된 신호가 축출되는 경우도 물론 발생하게 된다.

그림 9의 빗금친 부분이 하나의 출력 노드에 대한 무-에러 영역을 표시하며 출력 노드의 갯수가 n 일때 무-에러 영역은 식(7)에서 정의된다. 또한 그림 10에서는 무-에러 영역을 구하는 알고리듬을 보여주고 있다.

$$\text{무-에러 영역 (CB)}=\left[\begin{pmatrix}c_1(\text{upper})\\c_1(\text{lower})\end{pmatrix},\begin{pmatrix}c_2(\text{upper})\\c_2(\text{lower})\end{pmatrix},\cdots,\begin{pmatrix}c_n(\text{upper})\\c_n(\text{lower})\end{pmatrix}\right] \quad (7)$$

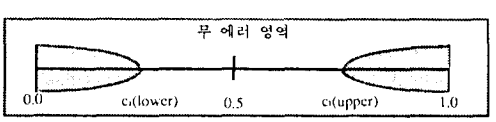


그림 9. 무-에러 구역
Fig. 9. No-error region.

<무-에러 영역>

가정 : 학습 신호의 수 = l
신호의 길이(벡터) = n
초기화 : c_i(upper) = 1.0
c_i(lower) = 0
iteration(i) = 1

1. i번째 신호의 출력을 검사한다.
1) 학습이 잘못된 경우 : i=i+1 과정 1로 간다.
2) 학습이 잘된 경우 : 과정 2로 간다.

2. 무-에러 영역을 재조정 한다.
1) IF $y_j^i \geq 0.5 \rightarrow c_i$(Upper) = Max [$c_i$(upper), y_j^i]
2) IF $y_j^i < 0.5 \rightarrow c_i$(lower) = Min [$c_i$(lower), y_j^i]

3. i=l → stop
i≠l → i=i+1 → 과정 1로 간다.

그림 10. 무-에러 영역을 구하는 방법
Fig. 10. The method of deciding no-error region.

4) 여러 영역들의 동시 사용

앞에서 에러 영역과 무-에러 영역을 정의하였다. 각각은 잘못 학습된 신호와 학습이 잘된 신호들을 이용해서 영역 설정을 하였다. 즉 에러 영역에서는 그 영역으로 출력 신호가 들어오면 신호를 축출하는 방법으로, 무 에러 영역에서는 그 영역으로 신호가 들어오면 신호를 통과하는 방식으로 신호의 타당성을 감별하고 있다. 우리는 위의 두 영역을 동시에 사용함으로써 좀 더 효과적으로 신호를 구분할 수 있다.

그림 11에서는 하나의 출력 노드에 대해 두 영역을 동시에 사용하는 모양을 보여주고 있다.

또한 3.2.1절에서 사용한 matching 방법을 위의 두 방법과 동시에 사용할 경우 좀 더 정확하게 신호의 타당성을 검사할 수 있다.

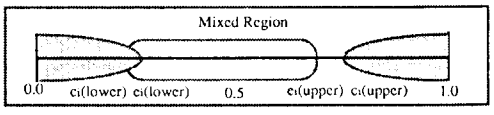


그림 11. 에러 영역과 무-에러 영역의 동시사용
Fig. 11. Simultaneous use of error and no-error bound.

5) 최대 거리 영역

앞의 에러 영역과 무 에러 영역은 시스템의 출력 벡터 값들의 축출 영역을 결정할 때 벡터의 각 비트들을 개별적으로 고려해서 했다. 이 절에서 얘기하고자 하는 방법은 시스템의 출력 벡터를 하나의 신호로 생각해서 그것들의 관계로 부터 축출 영역을 구분하고자 하는 것이다.

이 방법은 앞 절의 무 에러 영역을 결정해 주는 것과 마찬가지로 제대로 학습된 신호들의 출력을 이용해서 구축한다. 그림 12에서 처럼 제대로 학습된 신호들의 출력값들을 다 모은 뒤에 그 값들의 평균값을 구하고 그 평균값으로부터 가장 멀리 떨어진 신호의 출력값 까지의 거리를 이용해서 원을 그린다. 그렇게 해서 그 시스템의 출력값이 그 영역으로 들어오면 통과시키고 그렇지 않으면 축출한다.

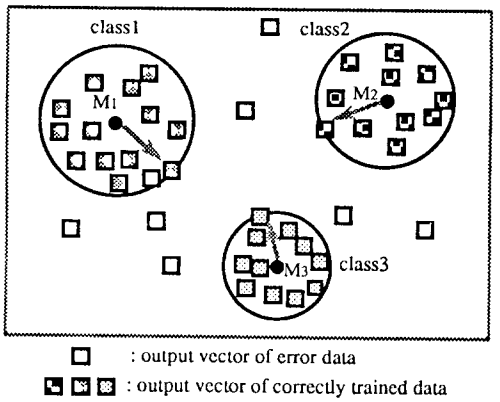


그림 12. 최대 거리 영역 방법
Fig. 12. Maximum distance scheme.

여기서 Mi를 총 n개의 class 중 i 번째 class의 출력값이라고 하면

$$M_i = \sum_{j \in class\ i} Y_j \qquad i = 1,\ldots n \tag{8}$$

이 된다. 즉 M_i는 i 번째 class의 제대로 학습된 신호의 출력값의 평균 값이다. 그리고 M_i에서부터 그 class의 출력값들 중 제대로 학습된 신호들까지의 거리중 최대값을 취하면

$$MAX_i = Max\left[\left|M_i - Y_j\right|\right] \qquad j \in classi, i = 1,\ldots,n \tag{9}$$

이 된다.

즉 벡터 M_i와 Max_i 두 요소를 가지고 신호의 축출 여부를 결정하면 된다.

Ⅳ. 결과 및 고찰

이 장에서는 앞에서 설계한 PSHNN의 성능을 컴퓨터 시뮬레이션에 의해 분석해 본다. 먼저 여러가지 축출 방법에 따른 PSHNN의 성능을 비교 검토한다.

그리고 PSHNN의 성능을 기존의 역전파 기법을 적용시켜 구성한 MNN의 성능과 비교한다.

실험들에 사용된 데이터는 원격탐측에 의해 얻은 영상 신호이며, Flight Line C1 (FLC1)이라는 이름을 갖고 있다. 이 데이타는 미국 인디아나 주의 농토를 비행기에서 찍어서 만든 것이다. 실험에서는 8개의 주파수 대역을 갖는 신호를 사용하고, 4개의 class 만을 사용한다. [8]

각 주파수 대역에 해당하는 영상은 0~255까지의 영상정보를 나타내며, 그 영상의 한 점을 나타내는 데는 이진수로 8 비트가 필요하다. 우리는 8개의 주파수 대역을 사용하므로, 영상의 한 점을 나타내는 데는 총 64 비트가 사용된다. 그러므로 신경망의 입력 노드수는 64개이고 출력 노드수는 class가 4이므로 4개이다.

시스템 구성시 학습시에 사용된 신호 샘플은 class 당 200개씩, 검사시에는 375개의 샘플을 사용해서 수행했다. 그러므로 학습시에 사용된 총 데이타 갯수는 800이고, 검사시에 사용된 총 신호 갯수는 1500이다.

구성되는 PSHNN의 각 SNN은 Delta rule을 이용한 1 계층 신경망에 출력단에 Sigmoid 함수를 사용하여 구성하며, 각 SNN을 학습시킬 때 학습률 η는 0.05로 사용하고, 학습 가중치의 초기값은 -0.5~0.5 사이의 값중 임의로 정해서 사용한다.

1. 축출 방법에 따른 성능 분석

이 절에서는 3.2 절에서 제안한 축출 방법에 따른 PSHNN의 성능을 비교 분석한다. 에러 구역을 사용한 경우와 그리고 matching 방법, 에러 영역, 무 에러 영역 세가지 모두를 동시에 사용한 경우 또한 최대 거리 영역 방법을 사용한 경우 각각을 PSHNN을 구성해서 성능을 알아본다. 모든 경우, 비선형 변환은 RDFT를 이용해서 구성한다.

표 2에서 에러 영역만을 사용했을 경우(PSHNN1) SNN의 갯수와 실제 학습시와 검사시의 인식률을 나타내 준다.

표 3에서는 동시에 Matching 방법, 에러 영역 방법, 무 에러 영역 방법을 사용했을 경우(PSHNN2)의 성능을 보여주며, 표 4에서는 최대거리영역 방법을 사용한 경우(PSHNN3)의 성능을 보여준다.

세가지 방법의 성능을 비교해 보면 PSHNN2의 경우 사용되어진 SNN의 갯수가 제일 적고 판별 능력은 제일 뛰어남을 알 수 있다. 즉 PSHNN1의 경우 최대의 검사과정시 인식률은 89.27%이며, 이때 사용된 SNN의 갯수는 4개 이다. 또한 PSHNN3의 경우는 91.66%이고 SNN의 갯수도 역시 4개이다. 반면에 PSHNN2의 경우는 92.13%를 나타내며 사용되어진 SNN의 갯수는 단 2개 뿐이다. 즉 PSHNN2의 경우가 가장 좋은 성능을 보여줌을 알 수 있다. 즉 matching 에러 영역, 무-에러 영역을 동시에 사용한 경우의 PSHNN 성능이 가장 뛰어남을 알 수 있다. 이것의 이유는 앞으로 이론적으로 좀 더 규명해야 할 과제이다.

또한 특이할 만한 것은 세 경우 모두 공통적으로 나타나는 현상으로 신경망의 특성이 처음에는 학습 횟수를 늘이면 늘일수록 성능이 나아지지만 어느 정도 이상의 되면 성능이 오히려 저하되는 현상을 볼 수 있다. 예를 들면, PSHNN2의 경우 학습 횟수가 90일때 학습 신호에 대해서는 98.75%의 인식률을

표 2. 4-class 문제에서 PSHNN1의 인식률과 반복횟수에 따른SNN의 갯수 (η=0.05, FLC1)

Table 2. The classification accuracy of the PSHNN1 and the total number of SNNs as a function of the number of iterations in the 4-class problem (η=0.05, FLC1).

허용된 학습 횟수	SNN의 수	인식률	
		학습신호(%)	검사신호(%)
50	3	99.98	89.07
70	3	99.98	89.27
100	4	99.98	87.87
300	3	99.98	87.27

표 3. 4-class 문제에서 PSHNN2의 인식률과 반복횟수에 따른 SNN의 갯수 (η=0.05, FLC1)

Table 3. The classification accuracy of the PSHNN2 and the total number of SNNs as a function of the number of iterations in the 4-class problem (η=0.05, FLC1).

허용된 학습횟수	SNN의 수	인식률	
		학습신호(%)	검사신호(%)
50	2	97.88	91.20
90	2	98.75	92.13
100	2	98.50	92.03
200	2	99.00	90.80
400	2	98.50	90.33

주지만 검사 신호에 대해서는 92.13%의 인식률을 준다. 그리고 학습횟수를 더 늘이게 되면 학습 신호에 대해서는 거의 99%의 인식률을 보이지만 검사 신호에 대해서는 90% 정도로 떨어지는 것을 볼 수 있다. 이것은 모든 신경망 구조에서 나타나는 현상으로서 과 학습에 기인한 것이다. 그러므로 PSHNN2의 경우 학습 횟수를 약 90회 정도로 하면, 가장 좋은 신경망을 얻을 수 있다.

표 4. 4-class 문제에서 PSHNN3의 인식률과 반복횟수에 따른 SNN의 갯수 (η=0.05, FLC1)

Table 4. The classification accuracy of the PSHNN3 and the total number of SNNs as a function of the number of iterations in the 4-class problem (η=0.05, FLC1).

허용된 학습횟수	SNN의 수	인식률	
		학습신호(%)	검사신호(%)
50	6	97.50	91.60
70	6	97.37	91.26
100	4	97.87	91.60
180	4	98.12	91.66
200	5	98.25	91.33

2 PSHNN과 MNN의 비교

이 절에서는 Matching 방법과 에러 영역 방법, 무 에러 영역 방법을 축출 방법으로 사용한 PSHNN과 역전파 방법을 이용한 MNN의 성능을 비교한다.

구성된 MNN은 3NN과 4NN으로서 두 경우 모두 은닉층의 노드의 갯수를 64로 고정시키고 학습률은 0.01로 하며 초기 가중치 값은 (-0.5, 0.5)의 값들 중 무작위로 골라서 한다.

표 5, 표 6에서 3NN과 4NN의 성능을 보여준다. 4NN의 성능이 3NN의 성능 보다 뛰어남을 알 수 있다. 4NN의 경우 학습 횟수가 800 일때 검사시의 인식률의 최고값 92.80%를 기록했다. 그러나 3NN의 경우는 약 4000번의 학습이 지난뒤 최대로 89.07%를 기록했으며, 두 경우 모두 그 이상 학습이 되면 오히려 성능이 저하됨을 알 수 있다. 이러한 현상도 역시 앞절에서 서술한 과 학습에 기인한 것으로 판단된다.

PSHNN2와 MNN을 비교하면 PSHNN2의 인식률은 4NN과 비슷하여 3NN 보다는 훨씬 좋음을 알 수 있다. 이때 시스템의 복잡도는 PSHNN2의 경우 2 계층만을 사용한 것이고 4NN은 3 계층을 사용한 것이다. 즉 PSHNN2와 4NN이 비슷한 성능을 나타내지만 복잡도에서는 PSHNN2가 훨씬 뛰어남을 알 수 있다.

또한 신경회로망의 중요한 요소중의 하나인 학습시간을 비교해 보기 위해 먼저 SNN 하나를 계산하는데 걸리는 시간을 1로 하고 PSHNN2의 학습시간과 비슷한 성능을 주는 4NN의 학습시간을 비교해 보기로 한다. PSHNN2를 학습시키는데 걸리는 시간은 먼저 SNN1에 대해 800×90이 걸리고 48×90이 SNN2에 걸린다. 그래서 총 76,320 만큼의 시간이 걸린다. 반면에 4NN의 경우는 800×800×3 = 1,920,000의 시간이 걸리며, 이 시간은 PSHNN2 보다 약 25배 가량 더 걸리는 시간이다.

즉 PSHNN과 MNN의 성능을 비교하면 비슷한 성능을 주는 경우 PSHNN의 시스템이 훨씬 간단하고 학습시간도 훨씬 더 짧음을 알 수 있다.

표 5. 4-class 문제에서 반복횟수에 따른 3NN의 인식률 (η=0.01, FLC1)

Table 5. The classification accuracy of 3NN as a function of the number of iterations in the 4-class problem (η=0.01, FLC1).

학습횟수	인식률	
	학습신호(%)	검사신호(%)
100	80.75	73.60
1000	92.62	87.87
3000	93.50	88.93
4000	93.50	89.07
5000	93.74	88.93

표 6. 4-class 문제에서 반복횟수에 따른 4NN의 인식률 (η=0.01, FLC1)

Table 6. The classification accuracy of 4NN as a function of the number of iterations in the 4-class problem (η=0.01, FLC1).

학습횟수	인식률	
	학습신호(%)	검사신호(%)
100	87.75	84.74
500	99.63	92.47
700	99.75	92.80
900	99.88	92.80
1000	99.88	92.60

V. 결 론

새롭게 제안된 신경망인 PSHNN은 특히 시스템의 복잡도를 줄이고, 학습시간을 줄이며, 성능을 향상시켜 보자는데 목적을 두고 구성되었다.

PSHNN이 기존의 신경망의 MNN과 가장 크게 틀린 점은 각 SNN이 직렬이 아니고 병렬로 연결되며 서로 무관하게 동작한다는 점이다. 또한 각 SNN마다 잘못 학습된 신호를 축출하는 방법을 이용해서 그 다음 SNN의 학습에 부담을 줄여 줌으로써 학습시간을 단축할 수 있고, SNN의 갯수를 여러가지 경우에 따라 스스로 조절해 줄 수 있는 능력도 있다.

실제로 컴퓨터 시뮬레이션을 통해 PSHNN의 성능을 MNN과 비교한 결과, 비슷한 성능을 주는 경우 PSHNN의 복잡도가 훨씬 덜하며, 학습시간은 매우 빠름을 알 수 있었다.

앞으로는 PSHNN의 성능을 높이기 위해 SNN의 성능을 높이기 위한 기초 연구와 또한 좀 더 이론적으로 PSHNN의 성능을 분석하는 방향의 연구가 진행되어야 하리라 판단된다.

※이 논문은 '92 통신 학술 연구 과제 지원과 '92 연세 대학교 학술 연구비 지원에 의한 결과임

參 考 文 獻

[1] E.R Berlekamp, *Algebraic Coding Theory*, McGraw-Hill Book Company, 1968.

[2] L.W. II, Couch, *Digital and Analog Communication Systems*, Mac Millan Pub. Co., 1983.

[3] R. O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis*, A Wiley-Interscience, 1973.

[4] O. K. Ersoy and D. Hong, "Neural Network Learning Paradigms Involving Nonlinear Spectral *Processing*", *Proceedings of ICASSP 89*, Glasgow, Scotland, U. K., May, 1989.

[5] O.K. Ersoy and D. Hong, "The Pararllel, Self-Organizing, Hierarchical Neural Networks", *IEEE Trans. on Neural Network*, vol. 1, no. 2, June 1990.

[6] O.K. Ersoy, "Real Discrete Fourier Transform", *IEEE Tran. ASSP, ASSP-33*, no. 4, p.880-882, Aug. 1985.

[7] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 1972.

[8] H. Ghassemian and D. Landgrebe, "On-Line Object Feature Extraction for Multispectral Scene Representation," *Technical Report no. TR-EE 88-34*, Purdue University, August 1988.

[9] Peter J. B. Hancock, "Data representation in neural nets : an empirical study", *Proceedings of the 1988 Connectionist* Models Summer School, Morgan Kaufmann Publishers Inc., p. 11-20, 1988.

[10] D. E. Rumelhart and J. L. McClelland, PDP Research Group, *Parallel Distributed Processing*, The MIT Press, Cambridge Massachusetts, 1988.

[11] M. Takeda and J.W. Goodman, "Neural Networks for Computation : Number Representations and Programming Complexity," *Applied Optics*, vol. 25, no. 18, September 1986.

[12] P. J. Werbos, "Learning How the world works : specifications for predictive networks in robots and brains", *Proceedings of ICSMC 87*, p.302-310, 1987.

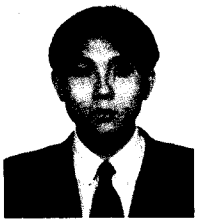
[13] DARPA, *DARPA Neural Network Study*, AFCEA International Press, 1988.

[14] Paolo Antognetti Veljko Milutinovic, *Neural Networks*, PRENTICE HALL, 1991.

[15] 홍 대식, "병렬 자구성 계층 신경망에 관한 연구", '92 통신학술 연구과제 결과 보고서, 1993

[16] 문 태현, 홍 대식, 강 창언, "병렬 자구성 계층 신경회로망", 제 4회 신호처리 합동학술 대회, 1991년 9월

著 者 紹 介



尹 寧 佑(準會員)

1992年 2月 연세대학교 전자공학과 졸업(공학사). 1992年 3月 ~ 현재 연세대학교 대학원 전자공학과 석사 과정 재학중.

洪 大 植(正會員)

1983年 2月 연세대학교 전자공학과 졸업(공학사). 1985年 2月 연세대학교 대학원 전자공학과 졸업(공학석사). 1990年 8月 Purdue University 졸업(Ph.D). 1990年 9月 ~ 1991年 7月 Purdue University Post-Doctoral Resear Associate. 1991年 8月 ~ 현재 연세대학교 전자공학과 조교수.

文 部 鉉(準會員)

1986年 2月 연세대학교 전자공학과 졸업(공학사). 1986年 2月 ~ 1991年 3月 (주) 대우전자 중앙연구소 연구원. 1993年 8月 연세대학교 대학원 전자공학과 졸업(공학석사). 1993年 9月 ~ 현재 연세대학교 대학원 전자공학과 박사과정.

康 昌 彦(正會員)

1960年 연세대학교 전기공학과 졸업(공학사). 1965年 연세대학교 전기공학과 졸업(공학석사). 1969年 미국 미시간 주립 대학교 대학원 전기공학과 (공학석사). 1973年 미국 미시간 주립 대학교 대학원 전기공학과(공학박사). 1967年 ~ 1973年 미국 미시간 주립 대학교 공업 연구소 선임연구원. 1973年 ~ 1981年 미국 노던 일리노이 대학교 전기공학과 조교수, 부교수. 1982年 ~ 현재 연세대학교 전자공학과 교수.