論 文
40~2~11

델타 규칙을 이용한 다단계 신경 회로망 컴퓨터 : Recognitron Ⅲ

Multilayer Neural Network Using Delta Rule : Recognitron Ⅲ

金 春 錫* · 朴 忠 圭** · 李 奇 翰*** · 黃 熙 隆§
(Chun-Suk Kim · Chung-Kyu Park · Ki-Han Lee · Hee-Yeung Hwang)

요 약

한 단계 *NN*(신경 회로망 컴퓨터, Neural Network)으로 했을 때 패턴을 학습시키지 못하는 linear separability문제 *XOR*경우를 해결하기 위하여 다단계 *NN*으로의 확장이 필요하게 되었다. 다단계 *NN*은 *EBP*(Error Back Propagation) 학습 방식으로 학습을 많이 했는데 이 *EBP NN*에는 많은 문제점이 있었다 : 첫째, D. Rummelhart가 *DR*(Delta Rule)을 확장시켰다고 하지만 입력부와 은닉부 사이에 존재하는 Weight, Weight Matrix *M*, 과 입력부와의 Linear combination한 중간 결과인 *H*가 다시 은닉부와 출력부 사이에 존재하는 Weight, Weight Matrix *N*, 과 linear combination이 이루어져서 계산된 결과 C_a를 생성하므로 문제가 발생하고, 둘째, 출력부에서 생성된 사이의 Weight *N*을 줄이는 것은 옳지만, 이 차이를 가지고서 입력부와 은닉부 사이의 Weight *M*을 고친다는 것은 정말로 옳지 못하다.

Recognitron Ⅲ가 이들 문제점들을 해결하기 위해서 제안되었다. 시뮬레이션 결과 Recognitron Ⅲ는 3단계 *NN* 그 자체를 학습하는 것이 아니라 한 단계씩으로 분할하여서 학습하므로 인해서 *EBP NN*보다 학습속도가 최소 35배에서 최대 72배가 빨리 학습됨을 알수가 있었고, 패턴 일반화(Pattern Generalization)의 경우는 EBP *NN*보다 Recognitron Ⅲ가 좋다. 그러나, 학습되는 패턴의 수는 입력과 출력 단계에서 *n*개의 셀과, 은닉부에서는 n+1의 셀의 존재하는 경우에 *EBP NN* 은 2^n개이지만 Recognitron Ⅱ는 n개이다.[5].

Abstract - The multilayer expanson of single layer NN (Neural Network) was needed to solve the linear seperability problem as shown by the classic example using the XOR function. The EBP (Error Back Propagation) learning rule is often used in multilayer Neural Networks, but it is not without its faults: 1) D. Rimmelhart expanded the Delta Rule but there is a problem in obtaining C_a from the linear combination of the Weight matrix N between the hidden layer and the output layer and H, wich is the result of another linear combination between the input pattern and the Weight matrix M between the input layer and the hidden layer. 2) Even if using the difference between C_a and D_a to adjust the values of the Weight matrix N between the hidden layer and the output layer may be valid is correct, but using the same value to adjust the

Weight matrix M between the input layer and the hidden layer is wrong.
Recognitron III was proposed to solve these faults. According to simulation results, since Recognitron III does not learn the three layer NN itself, but divides it into several single layer NNs and learns these with learning patterns, the learning time is 32.5 to 72.2 time faster than EBP NN one. The number of patterns learned in a EBP NN with n input and output cells and n+1 hidden cells are 2^n, but n in Recognitron III of the same size.[5] In the case of pattern generalization, however, EBP NN is less than Recognitron III.

1. 서 론

현재 신경 회로망 컴퓨터(Neural Network, *NN*)는 여러가지 규칙을 이용해서 학습을 하고 있는데 먼저 이들을 간략하게 살펴보고, 이들의 문제점 및 고쳐야 할 방법들에 대해서 논해보자.

1.1 델타 규칙(Delta Rule)을 이용한 한 단계신경 회로망 컴퓨터(Single Layer Neural Network)

한 단계 신경 회로망 컴퓨터(Single Layer Neural Network, *SNN*)은 그림 1과 같다.

그림 1과 같은 *SNN*을 학습시키는 방식은 Hebbian학습 규칙과 델타 규칙(Delta Rule, *DR*)이 있는데 *DR*은 Hebbian학습 규칙보다 많은 패턴을 학습시킬 수 있는 학습방식이다. 예를 들면, (1, 1, 1, 1), (1, 1, 1, −1), (−1, −1, 1, 1)과 같은 3개의 패턴이 있을 때, Hebbian학습규칙으로는 이들 패턴을 기억시키기가 어렵지만 *DR*로는 가능한 것이다. *DR*에 의한 학습규칙은 Appendix B에 나타나 있다. 이 *DR*은 Widrow-Hoff의 학습 규칙[1]과 매우 비슷하지만, 오차를 구하는 방식이 틀리다.

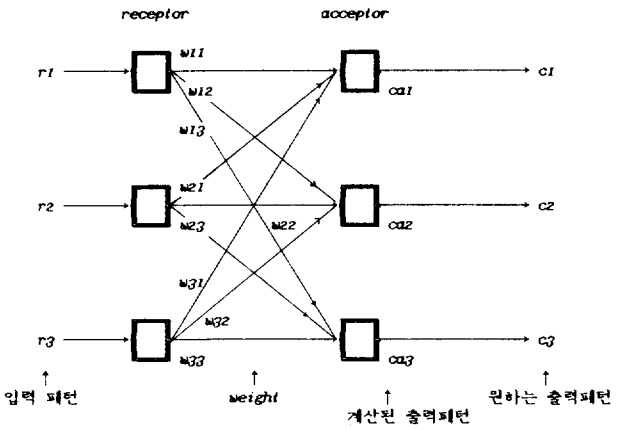

그림 1 한 단계 신경 회로망 컴퓨터
Fig. 1 Single Layer Neural Network, SNN

하지만, 이러한 *SNN*으로써 패턴을 학습시키지 못하는 경우가 있는데 이는 다음 절에서 살펴 보자.

1.2 다단계 신경 회로망 컴퓨터(Multi-Layer Neural Network)로의 전환

*SNN*으로 패턴을 학습시키지 못하는 경우를 살펴보고, 왜 다단계 신경 회로망 컴퓨터(Multi-Layer Neural Network, *MNN*)으로 이를 확장해야 하는지 그 이유를 살펴 보자.

1.2.1 다단계 신경 회로망 컴퓨터(Multi-Layer Neural Network)의 필요성

M. Minsky와 S. Papert는 그들의 저서인 Perceptron[2]에서 *SNN*으로 풀지 못하는 그림 2와 같은 linear separability문제를 제시하였다.

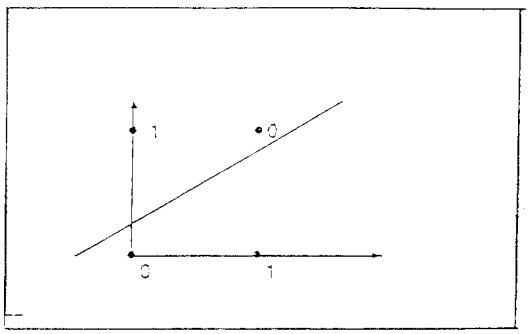

그림 2 linear separability 문제
Fig. 2 Linear Separability Problem

*正 會 員 : 崇實大 大學院 電氣工學科 卒業 · 工博
**正 會 員 : 崇實大 工大 電氣工學科 敎授 · 工博
***正 會 員 : 서울大 大學院 컴퓨터工學科 博士課程
§正 會 員 : 서울大 工大 컴퓨터工學科 敎授 · 工博
接 受 日 字 : 1990年 7月 11日
1 次 修 正 : 1991年 1月 3日

이 linear separability문제는 *XOR*로써 *SNN*으로는 풀 수가 없다. 따라서, 이를 풀기 위해서는 *MNN*이 필요하게 되었다.

1.2.2 다단계 신경 회로망 컴퓨터(Multi-Layer Neural Network) : 확장된 델타 규칙(Extended Delta Rule)을 이용한 다단계 신경 회로망 컴퓨터 EBP(Error Back Propagation Multi-Layer Neural Network)

앞 절에서 *SNN*으로 풀지 못하는 linear separablility문제를 풀기 위해서 *NN*을 다단계로 확장시켰다. 이렇게 *NN*을 다단계로 확장시킴으로써 linear separability와같은 문제는 쉽게 풀 수가 있었다. 또한, Kolmogorov는 *NN*을 그림 3과 같이 3단계 *NN*으로 하고 중간층(Hiden Layer)에 있는 뉴우런의 수를 입력층(Receptor Layer)에 있는 뉴우런의 수의 2배에 한 개를 더 추가한 3단계 *NN*으로는 모든 문제를 풀 수 있다고 증명하였다[5].

그러면, 이들 3단계 *NN*을 학습시키기 위한 어떤 규칙이 있어야 하는데, D. Rummelhart가 EBP (Error Back Propagation)이라는 새로운 학습 규칙[3]을 개발했는데, 이것이 확장된 델타 규칙 (Extended Delta Rule, *EDR*)이다. 이 *EDR*은 *SNN*의 학습 규칙인 *DR*을 확장한 개념이지만, Convergence하다는 것을 증명하지는 못했다. D. Rummelhart가 사용한 3단계 *NN*은 그림 3과 같다. 그리고 이 3단계 *NN*을 학습시키기 위한 *EDR*은 Appendix *C*에 나타나 있다.

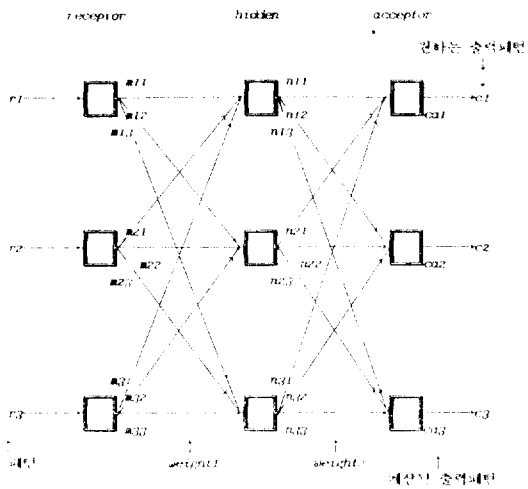

그림 3 3단계 신경 회로망 컴퓨터
Fig. 3 3(Multiple) Layer Neural Network, MSNN

1.3 확장된 델타 규칙(Extended Delta Rule)을 이용한 다단계 신경 회로망 컴퓨터 EBP (Error Back Propagation Multi-Layer Neural Network)의 한계점

D. Rummelhart가 그림 3과 같이 3단계 *NN*을 구축하고 *EDR*학습 규칙을 개발했지만 여러 가지 한계점이 있다.

첫째, D. Rummelhart가 *DR*을 확장시켰다고 하지만 입력부와 은닉부 사이의 Weight M과 입력부와의 linear combination한 결과인 H가 다시 은닉부와 출력부 사이의 Weight matrix N과 linear combination이 이루어져서 계산된 결과 C_a를 생성하므로 문제가 발생한다. 왜냐하면, 입력부와 Weight M과의 linear combination결과인 H가 이미 오차를 내포하고 있으므로, 이 H를 가지고 다시 C_a를 생성한다면 H의 오차가 C_a의 오차를 또 생성한 결과이므로 *EDR*을 사용할 경우에 D_a와 원하는 결과 C_a의 차이값이 오차를 줄일 수 있는 —패턴을 학습시킬 수 있는— 유일한 방법인데 C_a가 오차의 오차를 포함하고 있으므로 원하는 차이값이 정확할 수가 없다. 또한, 출력부에서 생성한 C_a와 D_a의 차이로 은닉부와 출력부 사이의 Weight N을 줄이는 것은 옳지만, 이 차이를 가지고서 입력부와 은닉부 사이의 Weight M을 고친다는 것은 정말로 옳지 못하다.

본 논문은 이러한 한계점을 타파하기 위한 방법과 목적등을 제시하고 있다. 2장은 3단계 *NN*인 Recognitron Ⅲ의 구조와 학습 방식등과 이 학습 방식의 Convergence 정리등을 보였고, 3장은 Recognitron Ⅲ의 시뮬레이션과 분석 및 결과를 보이고 4장은 결론을 맺었다.

2. 델타 규칙(Delta Rule)을 이용한 다단계 신경 회로망 컴퓨터(Multi-Layer Neural Network) : Recognitron III

앞에서 언급한 *EDR*을 이용한 *MNN*인 *EBP NN*의 한계점을 극복하기 위한 Recognitron Ⅲ의 구조 및 학습 규칙을 살펴 보자.

2.1 확장된 델타 규칙(Extended Delta Rule)을 이용한 다다계 신경 회로망 컴퓨터 *EBP* (Error Back Propagation Multi-Layer Neural Network)의 한계점의 타계

1.3절에서 설명한 *EDR*을 이용한 *EBP NN*의 한계점은 *SNN*의 학습 규칙인 *DR*을 무리하게 확장시킨데 있다. 이 *EBP NN*의 한계점을 타계하기 위한 Recognitron Ⅲ는 *MNN*을 여러개의 *SNN*으로 변형해서 각각의 *SNN*에 DR을 적용시킴으로써 이들 문제를 해결할 수가 있다.

2.2 Recognitron Ⅲ의 구조

*DR*을 이용하기 위해서 *MNN*을 여러개의 *SNN*으로 변형한다는 것은 그림 4와 같은 D. Rummelhart가 개발한 *EBP* 3단계 *NN*을 그림 4와 같은 형태의 *NN*인 Recognitron Ⅲ로 변형하면 된다.

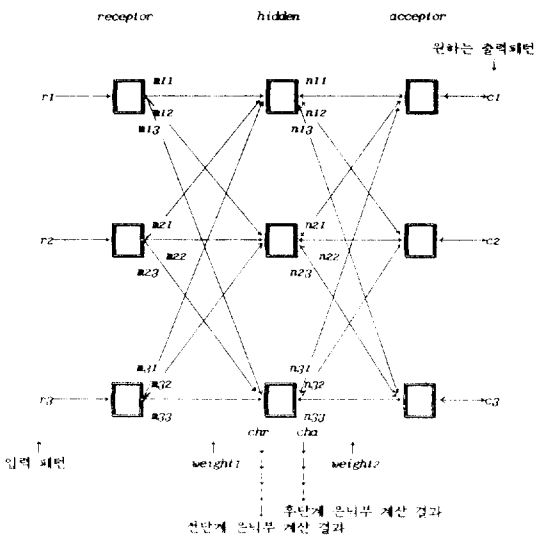

그림 4 3단계 Recognitron Ⅲ신경 회로망 컴퓨터
Fig. 4 Layer Recognitron Ⅲ neural network

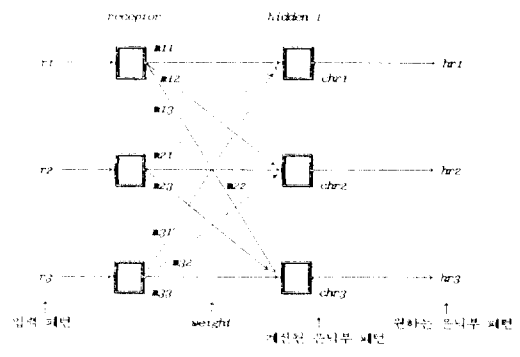

그림 5 전 단계 신경 회로망 컴퓨터
Fig. 5 First Layer Neural Network, FLNN

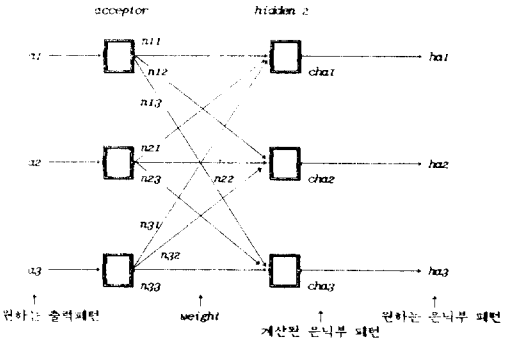

그림 6 후 단계 신경 회로망 컴퓨터
Fig. 6 Later layer Neural Network, LLNN

그러면 그림 5와 6과 같은 두개의 *SNN*으로 고칠 수가 있는 것이다.

*EBP NN*과 Recognitron Ⅲ의 차이점은 *EBP NN*은 입력부의 패턴과 입력부와 은닉부 사이의 Weight를 이용해서 은닉부의 패턴 값 H를 계산하고, 이 *H*와 은닉부와 출력부 사이의 Weight를 가지고서 C_a를 계산해서 출력부에서의 원하는 결과 D_a와의 오차를 구하는 연속적인 전방향(forward) 계산과 계산된 오차를 이용해서 Weight들을 조정하는 연속적인 후방향(backward)계산으로 오차들을 줄임으로써 학습이 이루어진다. 이들 계산과정을 살펴보면 한 방향의 연속적인 계산으로 멀티 타스킹을 사용하더라도 계산 시간의 절약을 얻을 수가 없다.

하지만, Recognitron Ⅲ는 입력부의 패턴 *R*과 출력부의 패턴 C_a를 모두 입력부로 사용하여서 입력부와 은닉부 사이의 Weight를 가지고 H_1패턴 값을 구하고, 출력부와 은닉부 사이의 Weight를 가지고서 H_2를 구한 다음에 이들 두 계산된 H_1과 H_2를 적당한 값으로 함으로써 학습을 하는데, 즉 구한 H_1과 H_2가 같아지면 학습이 끝나는 것이다.

*EBP NN*의 학습 방식이 단지 한 방향으로의 흐름 전방향, 후방향인데 비해서 Recognitron Ⅲ는 양방향 흐름-전후방향으로 함으로써 *DR*을 이용한 학습을 가능하게 할 뿐더러 Convergence정리등이 증명됨으로써 3단계 *NN*의 위치를 확실하게 하고 있다.

2.3 Recognitron Ⅲ의 학습 규칙

그림 4와 같은 Recognitron Ⅲ는 그림 5와 그림 6과 같은 두 개의 *SNN*으로 구할수가 있다. 따라서, Recognitron Ⅲ의 학습은 은닉부 *H*의 결정과

각각의 *SNN*을 *DR*을 이용하는 것으로 이루어진다. 그러면, Recognitron Ⅲ의 학습 규칙을 살펴보자. 그림 5와 같은 *SNN*인 경우에, 입력부 패턴 *R*과 Weight *M*과의 linear combination 결과를 Transfer Function을 취한 후의 값, H_1을 구할 수가 있다. 또한, 그림 6과 같은 출력부 *A*의 패턴과 Weight *N*과의 linear combination에 의한 결과를 Transfer Function을 취한 H_2를 얻을 수 있다. 여기서 Transfer Function은 Sigmoid Function을 택했다.

그러므로, Recognitron Ⅲ는 입력부 R과 출력부 *A*의 패턴 모두를 입력으로 하는 Pseudo Supervised 학습형태라고 부를 수도 있다. 이 방식은 입력부 *R*을 데이타로, 출력부 *A*를 키값으로 하는 Kohonen의 Unsuperviese학습 방식[3]과 아주 흡사하다. 하여간, 이렇게 구한 H_1과 H_2는 은닉부의 값 *H*를 구하는데 아주 중요한 단서가 된다. 은닉부의 값 H를 구하는 방식은

1) $H=(H_1^2+H_2^2)^{1/2}$,
2) $H=(H_1+H_2)/2$,
3) $H=H_1$,
4) $H=H_2$,

그 외에도 여러 가지가 있겠지만, Reconitron Ⅲ는 2)의 방식을 택하였는데 이는 계산의 간편성 때문이다. 여기서 3), 4)의 학습 방식을 취하면 계산은 더욱 간편하나 이는 Multilayer Perceptron의 학습 규칙과 비슷하게 됨으로 이를 피하였다.

이렇게 결정한 *H*값을 가지고서 그림 5와 그림 6과 같이 *SNN*을 *DR*을 이용해서 학습을 시킬 수가 있는데 자세한 것은 Appendix *D*에 있다.

또한 *DR*에서는 학습비를 실험에 의해서 얻은 고정값 0.9를 사용했는데 Recognitron Ⅲ에서는 실험에 의하지 않고 계산에 의해 구했다. Recognitron Ⅲ에서의 학습비는 전체 오차를 생성한 Weight Space와 수정해야 할 Weight의 비로써 구하는데, 즉,

$$\text{학습비}=\frac{\text{오차를 생성한 Weight 값}}{\text{오차를 생성한 전체 Weight Space}}$$

으로 하였다. 이를 수식으로 표현하면,

$$\text{학습비}=\frac{M_{ij}*R_j}{(\sum_j \text{std}*\text{sign value}-M_{ij}*R_j)}$$

여기서 std=11.25129로써 이는 Transfer Function의 하나인 Sigmoid Function의 Bound값이다.

sign value=+1오차가 0보다 클 경우
sign value=−1오차가 0보다 작을 경우

이렇게 가변적인 학습비를 사용함으로써 오차를 많이 생성한 Weight *M*는 변화를많이 주고, 오차를 적게 생성한 Weight *M*는 적은 변화를 주는 것이다. 그리고 이 오차가 갖는 아주 중요한 의미는 Weight의 증감 방향을 지정해 주는 것이다. 왜냐하면, 학습비의 sign-value는 현재의 Weight를 조정할 방향, 즉 Weight를 증가할 것인지, 아니면 감소할 것인지를 결정하기 때문이다. 이는 4장 수렴 증명에서 자세히 다룰 것이다.

전체적인 Recognitron Ⅲ의 학습 규칙을 수학적으로 알아보자. 앞에서도 정의한바와 같이 입력 패턴을 *r*, 은닉부 패턴을 *h*, 그리고 원하는 출력 패턴을 *a*인 벡터라고 하고, 입력부와 은닉부 사이의 Weight를 *M*, 그리고 은닉부와 출력부 사이의 Weight를 *N*인 행렬이라고 하자. 그리고 벡터는 *n*차고, 행렬은 *nXn*차라고 하자. 또한 C_a는 계산된 결과이고, *n*차의 벡터인 것이다.

*EDR*에 의한 *EBP NN*의 학습을 표시하면 아래와 같다.

$h=rM$으로써 *n*차의 벡터가 구해지고, $C_a=hN$으로써 *n*차의 벡터가 구해지므로 $C_a=rMN$으로써 *n*차의 벡터가 구해진다. 그러면 C_a와 *a*의 차이를 오차로 해서 이 오차를 이용해서 *EBP NN*을 학습시키는 것이다.

Recognitron Ⅲ는 은닉부의 패턴을 *h*1, *h*2인 *n*차의 벡터라고 하면,

$h1=rM$으로써 *n*차의 벡터가 구해지고, $h_2=N^{-1}a$으로써 *n*차의 벡터가 구해진다. 여기서 N^{-1}은 $N_{n\times n}$의 역행렬이다. 다시 $h=(h1+h2)/2$로 하고 이렇게 구한 *h*와 *h*1의 차이를 오차로 해서 전 단계의 *NN*을 학습시키고, 역시 구한 *h*를 이용해서 $C_a=hN$으로써 *n*차의 벡터가 구해지므로 C_a와 *a*의 차이를 오차로 해서 후 단계 *NN*을 학습시키는 것이다.

2.4 Recognitron Ⅲ의 수렴 증명

3단계 *NN* Recognitron Ⅲ의 학습 형태는 *SNN*을 *DR*을 이용한 학습 규칙과 비슷하지만, 틀린점은 Recognitron Ⅲ는 입력부와 출력부의 패턴이 모두 입력부 형태로 쓰이므로 그림 5와 그림 6과 같은 *SNN*인 경우에는 출력부의 개념이 없고, 이를 은닉부가 대신하는데 우리는 은닉부의 값을 정확히 모르기 때문에 그림 5에서 생성한 은닉부의 값 H_1과 그림 6에서 구한 은닉부의 값 H_2를 이용

해서 은닉부의 값 H를 구한다고 앞절에서 말했다. 그러므로 우리는 그림 5에서 구한 은닉부의 값 H_1과 H의 차이를 이용해서 그림 5의 전 단계 NN을 DR을 써서 학습시키고, 그림 6에서 구한 H_2와 H의 차이를 이용해서 그림 6의 후단계 NN을 DR을 써서 학습시키는 것이다.

그러면 Recognitron Ⅲ가 수렴한다는 것은 그림 5와 그림 6에서 구한 $H_i^{(t)}$가 최종적으로 어떤 값으로 된다는 것을 보이면 될 것이다. 그림 5에서의 Weight의 변화와 그림 6에서의 Weight의 변화는 아래와 같다.

$$M_{ij}^{(t+1)}=M_{ij}^{(t)}+L1*(H_i^{(t+1)}-H_{i1}^{(t)})*R_j^{(t)}$$
$$N_{ij}^{(t+1)}=N_{ij}^{(t)}+L2*(H_i^{(t+1)}-H_{i1}^{(t)})*A_j^{(t)}$$

여기서,

$$L1=[M_{ij}^{(t)}]/\{[\sum_j \text{std}*\text{sign value}-M_{ij}^{(t)}*R_j^{(t)})]\}$$
$$L2=[N_{ij}^{(t)}]/\{[\sum_j \text{std}*\text{sign value}-N_{ij}^{(t)}*A_j^{(t)})]\}$$

std=11.25129로써 이는 Transfer Function의 하나인 Sigmoid Function의 bound 값이다.

sign value=+1오차가 0보다 클 경우
sign value=−1오차가 0보다 작을 경우

$E_i^{(t)}=H_i^{(t)}-H_{i1}^{(t)}$라고 하자. 그러면 여기서 증명해야하는 것은 $\Delta E_i^{(t)}=E_i^{(t+1)}-E_i^{(t)}$가 0보다 작은 것을 증명하면 된다.

(증명)

$$H_i^{(t+1)}=[H_{i1}^{(t+1)}+H_{i2}^{(t+1)}]/2,$$
$$H_i^{(t)}=[H_{i1}^{(t)}+H_{i2}^{(t)}]/2$$

임을 알 수가 있다.

$$\begin{aligned}\Delta E_i^{(t)}&=E_i^{(t+1)}-E_i^{(t)}\\&=(H_i^{(t+1)}-H_{i1}^{(t+1)})-(H_i^{(t)}-H_{i1}^{(t)})\\&=[H_{i2}^{(t+1)}-H_{i1}^{(t+1)}/2]-[(H_{i2}^{(t)}-H_{i1}^{(t)})/2]\\&=[(H_{i2}^{(t+1)}-H_{i2}^{(t)})/2]-[(H_{i1}^{(t+1)}-H_{i1}^{(t)})/2]\\&=\{[\sum A_j^{(t+1)}N_{ij}^{(t+1)}-\sum A_j^{(t)}N_{ij}^{(t)}]/2\}\\&\quad-\{[\sum R_j^{(t+1)}M_{ij}^{(t+1)}-\sum R_j^{(t)}M_{ij}^{(t)}]/2\}\\&=(1/2)\{\sum A_j^{(t)}[N_{ij}^{(t+1)}-N_{ij}^{(t)}]\\&\quad-\sum R_j^{(t+1)}[M_{ij}^{(t+1)}-M_{ij}^{(t)}]\}\\&=(1/2)\{\sum A_j^{(t)}*L2*(-E_i^{(t)})*A_j^{(t)}\\&\quad-\sum R_j^{(t+1)}*L1*E_i^{(t)}*R_j^{(t)}\}\\&=(1/2)[-\sum A_j^{(t)}*L2*E_i^{(t)}*A_j^{(t)}]\\&\quad-(1/2)[\sum R_j^{(t+1)}*L1*E_i^{(t)}*R_j^{(t)}]<0\end{aligned}$$

(왜냐하면 $-\sum A_j^{(t)}*L2*E_i^{(t)}*A_j^{(t)}<0$이고, $-\sum R_j^{(t+1)}*L1*E_i^{(t)}*R_j^{(t)}<0$이므로)

$$\therefore \Delta E_i^{(t)}<0$$

따라서 $E_i^{(t)}$는 0으로 수렴할 것이고, $H_i^{(t+1)}$과 $H_i^{(t)}$는 $E_i^{(t)}$가 0으로 수렴할 때 어떤 $H_i^{(n)}$의 값으로 수렴할 것이다.

그림 7은 Recognitron III의 은닉부의 값 $H=(H_1+H_2)/2$의 수렴 그래프이다. 이때 TSS(Total Sum of Square, 출력부의 오차들의 제곱의 합)는

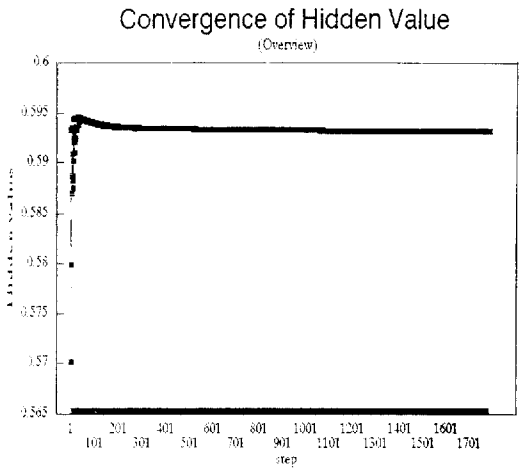

그림 7 a) 은닉부 값의 수렴 그래프(전체)
Fig. 7 a) The Convergence Graph of The Hidden Layer (Overview)

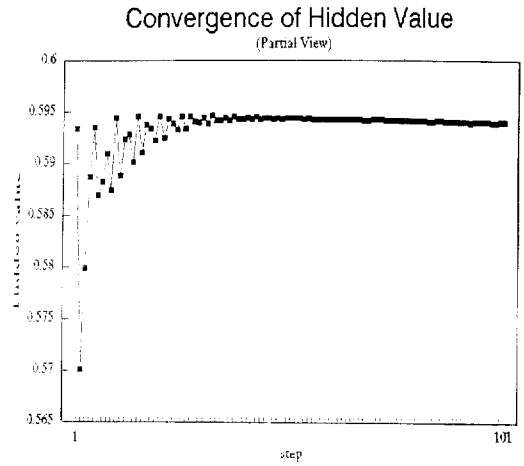

그림 7 b) 은닉부 값의 수렴 그래프(부분)
Fig. 7 b) The Convergence Graph of The Hidden Layer (Partial-view)

따라서 $E_i^{(t)}$는 0으로 수렴할 것이고, $H_i^{(t+1)}$과 $H_i^{(t)}$는 $E_i^{(t)}$가 0으로 수렴할 때 어떤 $H_i^{(n)}$의 값으로 수렴할 것이다.

그림 7은 Recognitron III의 은닉부의 값 $H=(H_1+H_2)/2$의 수렴 그래프이다. 이때 TSS(Total Sum of Square, 출력부의 오차들의 제곱의 합)는

0.00001로 주졌을 때의 은닉부의 값이 수렴하는 것을 보여주는 것이다. 이는 5장 시뮬레이션의 결과이다.

3. Recognitron Ⅲ의 시뮬레이션 분석 및 검토

Recognitron Ⅲ를 간단히 시뮬레이션해보고 *EBP NN*와 비교 검토해 보자.

3.1 Recognitron Ⅲ의 시뮬레이션 환경

Recognitron Ⅲ와 *EBP* 신경 회로망 컴퓨터는 IBM PC i286PC 상에서 학습이 이루어졌고, Turbo-C로 프로그래밍을 했다. 물론 이때 데이타들은 모두 같았다.

3.2 Recognitron Ⅲ의 시뮬레이션 및 분석

입력 패턴은 아래 그림 8과 같이 주고 출력패턴은 그림 9와 같이 주었다.

3.3 Recognitron Ⅲ와 *EBP NN*의 Pattern Generalization기능, 학습 속도, 기능한 학습 패턴 수의 비교 분석

3.3.1 Recognitron Ⅲ와 *EBP NN*의 학습 속도 비교 분석

Recognitrin Ⅲ의 학습은 전 단계에서 구한 은닉부의 값과 후 단계에서 구한 은닉부의 값을 이용해서 적당한 은닉부의 값을 고정하고, 이를 이용해서 전 단계와 후단계를 각각 독립적으로 학습한

[입력 패턴 벡터]
패턴 1 : -1 -1 -1 -1 -1 1 1 1 1 1
패턴 2 : -1 -1 -1 -1 1 1 1 1 1 -1
패턴 3 : -1 -1 1 1 1 1 1 -1 -1
패턴 4 : -1 -1 1 1 1 1 -1 -1 -1 -1
패턴 5 : -1 1 1 1 1 -1 -1 -1 -1 -1

그림 8 입력 패턴
Fig. 8 Input Pattern

[원하는 출력 패턴]
패턴 1 : 1 -1 -1 -1 -1 -1 -1 -1 -1 -1
패턴 2 : -1 -1 1 -1 -1 -1 -1 -1 -1 -1
패턴 3 : -1 -1 -1 1 -1 -1 -1 -1 -1 -1
패턴 4 : -1 -1 -1 -1 1 -1 -1 -1 -1 -1
패턴 5 : -1 -1 -1 -1 -1 1 -1 -1 -1 -1

그림 9 출력 패턴
Fig. 9 Output Pattern

표 1 시스템들의 학습 속도 비교
Table 1 Comparision with Learning Speed of *NN*s

내용 / 시스템	TSS	학습비	학습시간
Recognitron Ⅲ	0.01	가변적	17.5초
	0.000001	가변적	3분50초
EBP NN	0.01	0.3	31분54초
	0.01	0.6	15분47초
	0.01	0.9	10분17초

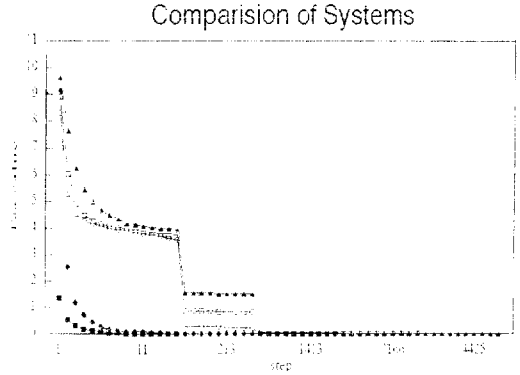

그림 10 각 시스템들의 TSS 변화
Fig. 10 TSS Convergence of Neural Network Systems

다고 했다. 표 1은 Recognitrin Ⅲ와 *EBP NN*의 학습 속도를 나타낸 것이다.

그림 10은 각 시스템의 *TSS*가 감소할 때, 즉 학습되어질 때 *TSS*의 감소를 나타낸것이다.

이 표 1로부터 *TSS*가 같을 때 Recognitron Ⅲ는 *EBP NN*보다 최소 35.3배에서 최대 72.2배가 빠를 것을 알 수가 있다. 이를 분석해보면 첫째, Recognitron Ⅲ의 초기 *TSS*가 *EBP NN*의 초기치보다 훨씬 작다는 것이다. 이는 은닉부의 값을 설정할 때 Recognitron Ⅲ는 *EBP NN*보다 수학적으로 보다 정확한 값을 얻기 때문이다. 둘째, 입력부와 은닉부 사이의 Weight *M*과 은닉부와 출력부 사이의 Weight *N*을 조정할때 2.3절에서말한 바와 같이 Weight가 많은 오차를 생성한 것은 많은 Weight변화를 시켜주고, Weight가 적은 오차를 생성한 것은 적은 Weight변화를 시켜주기 때문이라고본다.

3.3.2 Recongnitron Ⅲ와 *EBP NN*의 가능한 학습 패턴의 수 비교 분석

Weight M과 Weight N은 같은 초기치로하고, 학습 패턴은 그림 8과 같이 주고, Recognitron Ⅲ와 *EBP NN*를 각각 표 1과 같이 *TSS*와 학습비를 주었을 때의 학습 속도는 표 1에서 보였다. 이 절에서는 이들 테스트 패턴을 그림 11과 같이 했을 때 표2는 이들과 학습 패턴들과의 Hamming Distance를 구한 것이다.

표 2에서 보면 될 수 있으면 각 테스트 패턴은 학습 패턴과 Hamming Distance가적은 것으로 수

[테스트 패턴]
패턴 1 : −1 1−1 1−1 1−1 1−1 1
패턴 2 : −1 1 1−1 1 1−1 1 1−1
패턴 3 : −1 1 1 1−1 1 1−1 1−1
패턴 4 : −1−1−1−1−1−1−1 1 1 1
패턴 5 : −1−1−1−1−1−1 1 1 1 1

그림 11 테스트 패턴
Fig. 11 Test Pattern

표 2 테스트 패턴과 학습 패턴들과의 Hamming Distance
Table 2 Hamming Distance of Test Pattern and Learning Input Pattern

패턴		학습패턴				
		1	2	3	4	5
테	1	4	6	**4**	6	5
스	2	5	3	5	5	**3**
트	3	5	5	5	3	**2**
패	4	**3**	5	7	9	9
턴	5	**2**	4	6	8	10

표 3 각 시스템을 이용한 테스트 패턴의 재생(Recall)결과
Table 3 Recall result with Test Pattern on NN_s

재생한 결과 시스템		테스트 패턴				
		1	2	3	4	5
Recognitron Ⅲ	수렴 학습 패턴	3	5	5	5′C	5′C
EBP NN		3	5	5	1	1

여기서, 5′C는 학습 패턴 5의 반전을 의미한다.

렴해야 할 것이며, Hamming Distance가 10이라는 것은 이 시스템의 경우에는 학습 패턴과 테스트 패턴이 반전(Complement)관계라는 것이다. 표 2에서 검은 부분은 테스트패턴이 수렴하는 학습패턴이다.

이 테스트 패턴을 이용해서 각 시스템이 재생(Recall)한 결과는 표 3에 나타나 있다.

표 3에서 살펴보면 Recognitron Ⅲ의 테스트 패턴 4와 5의 재생의 경우에 Hamming Distance가 가장 낮은 학습 패턴 1로 수렴해야 하는데 그렇치 못하고 학습 패턴5의 반전패턴으로 수렴함을 알 수가 있는데 이는 Recognitron Ⅲ가 *DR*을 그대로 사용했기 때문으로 분석된다. 사실은 이러한 결과는 안좋다. 왜냐하면 이러한 반전된 패턴은 학습을 시키지 않았기 때문이다. *NN*에서는 이러한 경우가 될 수 있으면 안나타나야 좋은 것이다. 표 3의 결과로 미루어 가능한 학습 패턴의 수는 *EBP NN*이 Recognitron Ⅲ보다 많다는 것을 알 수가 있다.

3.3.3 Recognitron Ⅲ와 *EDP NN*의 Pattern Generalization의 기능 비교 분석

*EBP NN*과 Recognitron Ⅲ의 Pattern Generalization기능을 살펴보면, 그림 12에서와 같이 학습이 안된 패턴의 재생을 수행한 결과는 표 4에서 보는 바와 같이 *EBP NN*은 이미 학습된 패턴으로 나타나지만 Recognitron Ⅲ는 새로운 패턴을 생성해냈다.

물론 이 새롭게 생성된 패턴은 여러번 Error Correction을 수행해도 안정된 상태(stable state)로 고정됨을 볼 수가 있었다.

학습 않된 패턴 : −1 1−1−1 1 1 1 1 1−1

그림 12 패턴의 일반화
Fig. 12 Pattern Generalization

표 4 패턴 일반화 비교
Table 4 Comparison of recall result with unlearned Pattern on NN_s

재생한 결과 시스템		패턴 일반화 재생 결과
학습 않된 패턴		−1 1−1−1 1 1 1 1 1−1
Recognitron Ⅲ	수렴된 재생 패턴	−1 1−1−1 1 1 1 1 1−1
EBP NN		−1−1−1−1 1 1 1 1 1−1

4. 결 론

*EBP NN*의 불합리성을 앞에서 논했고, 이를 어떻게 하면 타계 할 것인가를 알기 위해서 Recognitron Ⅲ가 본 논문에서 제안되었고, Recognitron Ⅲ는 *EBP NN*가 가지는 한계점을 타계했다.

*EBP NN*이 3단계 *NN*에서 수렴한다는 것을 수학적으로 증명을 못했는데 Recognitron Ⅲ는 이를 증명했다.

Recognitron Ⅲ의 학습 속도는 *EBP NN*의 학습 속도보다 무려 최소 35.3배에서 최대 72.2배가 빠르다.

Pattern Generalization의 경우에는 *EBP NN*보다 Recognitron Ⅲ이 좋다. 하지만, 학습 가능한 패턴의 수는 *EBP NN*가 Recognitron Ⅲ보다는 좋다.

그리고 *EBP NN*는 4단계, 5단계로 확장할 수가 있는데——물론 그 결과는 아주저조하지만——Recognitron Ⅲ는 이렇게 4단계 이상인 경우는 적용하기가 힘들다.

Appendix a : Transfer Function의 종류들

1. Hard Limit Function

   $F(z)=0$, if $z<$ Threshold
   $F(z)=z$, if $z>=$ Threshold

2. Sigmoid Function

   $F(z)=1/(1+e^{-z})$

3. Hyperbolic Tangent Function

   $F(z)=(e^{z}-e^{-z})/(e^{z}+e^{-z})$

Appendix b : 델타 규칙(Delta Rule)을 이용한 한 단계 신경 회로망 컴퓨터 (Single Layer Neural Network)의 학습규칙

[학습 규칙]

1. Random값을 가지고 Weight W를 초기화
2. 출력부(Acceptor)의 각 뉴우런 a_j의 Weighted Sum을 구한다.
   ; $a_i=\sum_j r_j * w_{ji}$, $j=1, 2, 3$
   ; r_j는 입력 패턴
3. Weighted Sum$_i$을 Transfer Function을 통과시켜서 Ca_i을 구한다.
   ; Transfer Function은 Hard Limit Function, Sigmoid Fuction
4. 뉴우런 i에서의 Error e_i을 구한다.
   ; $e_i=a_i-Ca_i$
5. 만약 Error e_i이 원하는 Threshold 값보다 작으면 끝낸다.
   아니면, 뉴우런 i에 연결된 Weight w_{ji}을 고치고
   ; $w'_{ji}=w_{ji}+cl * e_i * a_i$
   ; $cl=0.9$(학습비)
2. 로가서 다시한다. $i=1, 2, 3$

Appendix c : 확장된 델타 규칙(Extended Delta Rule)을 이용한 다 단계 신경 회로망 컴퓨터 *EBP*(Error Back Propagation Multi-Layer Neural Network)의 학습규칙

[학습 규칙]

가정 : F'(Weighted Sum_j) $=Ca_j * (1-Ca_j)$
F'(Weighted Sum_j)은 Sigmoid Function F의 미분 값이다.
F(Weighted Sum_j) $=1/(1+e^{-\text{Weighted Sum}_j})$

1. Random값을 가지고 Weight M과 N를 초기화
2. 출력부(Acceptor)의 각 뉴우런 a_i에 연결된 Weighted Sum_i을 구한다.
   ; Weighted $Sum_i=\sum_j(F(\sum_j r_j * M_{ji}) * N_{ji})$
3. Weighted Sum_i을 Transfer Function을 통과시켜서 Ca_i을 구한다.
   ; Transfer Function은 Sigmoid Function.
4. 뉴우런 a_i에서의 Error e_i을 구한다.
   ; $e_i=a_i-Ca_i$
5. 만약 e_i이 원하는 Threshold값보다 작으면 끝낸다.
6. 아니면 은닉부와 출력부 사이의 Weight N을 수정하고,
   ; $N'_{ji}=N_{ji}+cl * \delta_j * Ca_i$
   ; $\delta_j=(a_j-Ca_j) * Ca_j * (1-Ca_j)$
   ; $cl=0.9$(학습비)
   또한 입력부와 은닉부 사이의 Weight M을 수정한다.
   ; $M'_{ji}=M_{ji}+cl * \delta_j * h_i$
   ; $\delta_j=(\sum_k \delta_k * N_{kj}) * Ca_j * (1-Ca_j)$
   ; $cl=0.9$(학습비)
   ; δ_k=출력부에서 구한 δ_j와 같다. $k=j$
   그리고 2로 돌아간다.

Appendix d : 델타 규칙(Delta Rule)을 이용한

다 단계 신경 회로망 컴퓨터 (Multi-Layer Neural Network) : Recognitron Ⅲ의 학습 규칙

[학습 규칙]

가정 : Transfer Function은 Hyperbolic Tangent Function이고, 그 값은 [−1, 1]사이 값이다.

입력과 출력 패턴 백터는 {−1, 1}이다.

1. Weight M과 N을 Random값으로 초기화 시킴.

《전 단계》

2. Delta Rule을 이용해서 은닉부의 값 hr_i를 구함.

; $hr_i = F(\sum_j r_j * M_{ji})$

3. 은닉부의 값 $h_i = (hr_i + ha_i)/2$를 구함.

4. Error er_i를 구함.

; $er_i = h_i - hr_i$

5. er_i가 Threshold보다 작으면 끝내고, 아니면 6으로

6. 입력부와 은닉부 사이의 Weight M을 수정한다.

; $M'_{ji} = M_{ji} + (r_j * M_{ji}/$전체 Error Weight Space$) * er_i$

; 전체 Error Weight Space $= std * sgn - r_j * M_{ji}$

, $std = 11.25129$ (Sigmoid Function의 최대값으로 이보다 크면 항상 1)

$sgn = +1$, if $er_j \geqq 0$

$sgn = -1$, if $er_i < 0$

《후 단계》

2. Patrial Pivoting을 이용한 Gauss-Jordan법으로 은닉부의 값 ha_i를 구함.

; 출력부의 값을 Arc Hyperbolic tangent Function에 통과시켜서;

za_i를 구하고, Partial Pivoting을 이용한 Gauss-Jordan법에 의한;

Weighted Sum_i을 구한다.

; Weighted $Sum_i = \sum_i za_i * N_{ji}$

; Weighted Sum_i를 Transfer Function에 통과시켜 ha_i를 얻는다.

; $ha_i = F$(Weighted Sum_i)

3. 은닉부의 값 $h_i = (hr_i + ha_i)/2$를 구함.

4. Error ea_i를 구함.

; $ea_i = h_i - ha_i$

5. ea_i가 Threshold보다 작으면 끝내고, 아니면 6으로.

6. 은닉부와 출력부 사이의 Weight N을 수정한다.

; $N'_{ji} = N_{ji} + (a_i{}^* N_{ji}/$전체 Error Weight Space$) * ea_i$

; 전체 Error Weight Space $= std * sgn - a_i * N_{ji}$, $std = 11.25129$ (Sigmoid Function의 최대값으로 이보다 크면 항상 1)

$sgn = +1$, if $ea_i \geqq 0$

$sgn = -1$, if $ea_i < 0$

참 고 문 헌

[1] B. Widrow, and M. Hoff, "Adaptive Switching Circuits", 1960 IRE WESCON Connvention Record, Part 4, pp. 96~104, August 23~26, 1960

[2] M. Minsky, and S. Papert, Preceptron, The *MIT* Press, 1969

[3] D. Rumelhart, and J.L. McClelland, Parallel Distributed Processing, Chap. 8, 1986.

[4] T. Kohonen, "Correlation Matrix Memories", IEEE. Trans. Com. Vol. C-21, No. 4, 1972.

[5] R.H. Nielsen, Nuerocomputing, Addison-Wesley Pub. Comp., 1990