論 文

역전파 알고리듬과 사전을 이용한 필기체 영문자 인식

正會員 金 應 成* 正會員 趙 成 桓** 正會員 李 根 泳*

A Recognition of Handwritten English Characters Using Back Propagation Algorithm and Dictionary

Eung Sung Kim,* Seong Hwan Cho,** Keun Young Lee* *Regular Members*

要 約

본 논문에서는 역전파 알고리듬으로 학습된 신경회로망과 사전을 이용하여 필기체 영문자 인식을 수행하였다. 스캐너를 이용하여 입력된 영상화일로부터 불필요한 데이터 부분을 제거하고 문자의 다양성을 최소화하기 위해서 여러가지 전처리과정, 즉 문자분리, 중심변환, 잡음제거, 배율조정과 세선화과정을 거쳤다. 다음으로 세선화된 문제 패턴으로부터 문자의 특징이 추출되고, 신경회로망에 시험데이터에 대한 특징들을 학습시켰다. 그리고 테스트할 영문자에 대해서도 특징들을 추출하여 이미 학습된 신경회로망에 의해 분류하였다. 마지막으로 학습시간을 줄이고 인식율을 향상시키기 위한 방법과 학습시간과 은닉층의 노드수에 대해 고찰하였다.

실험 결과로서 이와같은 시스템으로 필기체 영문제에 대하여 학습후에 약 93%의 높은 인식율을 얻을 수 있었고 사전을 이용했을 경우 인식율이 약 97%였다.

ABSTRACT

In this paper, it is shown that neural networks trained with back propagation algorithm and dictionary can be applied to recognize handwritten English characters. To eliminate the useless data part and to minimize the variety of characters from the scanned image file, various preprocessings ; that is, segmentation, centering, noise filtering, scaling and thinning are performed. After these, characteristic features are derived from thinned character pattern. The neural network is trained by using the extracted features for sample data, and all test data are classified into English alphabets according to their features through the neural network. Finally, the ways of reducing learning time and improving recognition rate, and the relationship between learning time and hidden layer nodes are considered. As a result of this study, after successful training, a high recognition rate has been obtained with this system for the trained patterns and about 93% for test patterns. Using dictionary, the recognition rate was about 97% for test pattern.

*成均館大學校 電子工學科
Dept. of Electronic Engineerinh, Sung Kyun Kwan Univ.
**大宥工業專門大學 電子計算機科
Dept. of Computer Science Dae Yeu Technical Junior College
論文番號 : 93-17

Ⅰ. 서 론

기존의 많은 양의 문서 데이터와 늘어나는 정보를 데이터 베이스화하여 원하는 정보를 신속히 찾아내기 위해서 수작업에 의한 입력 방법보다 신속하고 정확한 입력 장치의 개발이 요구되었다. 이런 시스템을 구성하는데 있어서 문자 인식은 중요한 문제로 대두되어 컴퓨터를 사용하여 인간의 글씨 그대로를 인식하게 하는 여러 가지 방법, 즉 알파벳을 구문론적으로 해석[1]하거나 패턴의 발생을 통계적으로 분석하여 인식하는 방법[2], 또는 문자 패턴을 분류하여 근접하게 매칭하는 방법[3]등이 연구되었으나 이러한 방법들은 많은 산술계산으로 인한 처리속도가 느리며 특히 필기체 문자와 같이 동일한 문자라도 여러 형태와 다양한 기울기, 크기를 가지는 패턴의 경우 잡음에 민감하여 인식율이 낮은 단점이 있다.[4,5] 그래서 본 연구에서는 필기체 영문자에 맞는 전처리 과정에 대해서 고찰하며 인식에 있어서는 신경회로망의 특징, 즉 데이터가 인식과정에서 학습시킨 신경회로망에 의해 병렬처리됨으로써 기존의 방법보다 처리속도가 빠르며, 잡음에 의한 손상된 패턴에 대해서도 인식가능한 특성을 이용하여 필기체 영문제 인식 시스템을 구현한다.

세부과정으로서 먼저 스캐너로부터 필기체 영문자를 입력하고 인식과정에서 실제 신경망을 적용하기 전에 전처리 단계로 문자분리와 중심변환, 잡음제거, 배율조정, 세선화과정을 거치는데, 이 단계는 필기체 영문자의 다양성을 일반화 시키기 위하여 필요하다. 전처리 과정을 거친 각 문자에 대해 특징추출을 한다. 문자의 학습과 인식에 사용하는 신경회로망은 Falman이 제안한 수정된 역전파 알고리듬(back propagation algorithm)[6]을 이용한 다층 퍼셉트론(multi-layer perceptron)을 사용한다. 인식시에 인식율을 높이고 오류 문자를 수정하기 위한 방법으로서 사전을 이용하여 단어별로 문자를 인식하는 시스템을 제안하고, 또한 은닉층을 이루는 노드의 수와 학습율, 모멘텀이 학습시간과 인식율에 미치는 영향에 대해 고찰한다.

Ⅱ. 본 론

1. 전처리 과정

문자 데이터를 입력하여 실제 신경회로망의 입력으로 들어가기 전에 같은 문자라도 쓰는 사람에 따라 일정한 모양을 갖지 못하므로 그 다양성을 최소화시키고 불필요한 데이터부분과 알고리듬 수행시간을 줄이기 위해서 전처리 과정을 거친다.

1.1 문자 분리

입력 데이터 화일로부터 각각의 한 문자로 분리하는데, 이는 기본적으로 각 글자사이의 공백을 이용하여 수행한다.[7] 즉 각 글자와 글자사이에는 공간이 있으므로 일정 크기의 1(문자 부분)의 화소가 공백으로 둘러싸여 있을때 한 문자로 간주하여 그 문자에 대한 하나의 화일을 만든다. 각 문자에 대해 후에 인식과정에서 사용될 다섯가지의 정보를 문자 분리시에 함께 검출한다. 첫번째와 두번째는 단어와 단어사이의 공백의 존재와 그 단어가 그 줄의 마지막 단어인가를 개행문자로 문자 화일 끝에 저장하게 되는데, 이는 신경회로망으로 문자인식후에 사전을 이용한 단어인식을 쉽게 하기 위해서 이다. 그 다음은 대문자와 소문자의 모양이 같은 문자(c, k, o, p, s, v, w, x, z)의 구별을 위해서 글자의 세로(행) 크기를 저장한다. 그러나 K와 k, P와 p는 글자의 세로 크기가 같기 때문에 그것으로는 대문자 구분의 척도가 될 수 없어 먼저 K와 k의 구별위해 문자부분에 해당하는 제일 윗줄로부터 6번째행의 글자선의 수를 조사하여 이를 저장한다. 이는 K와 k의 가장 큰 특징이 글자의 두번째 획인 "<"를 위에서 부터 쓰는가 중간에서 부터 쓰는가에 있기 때문이다. P와 p의 구별은 다음 문자의 위치와 단어열 윗줄과의 공간으로 한다. 즉 그림 1.에서 대문자 P ①, ②번의 P와 P다음 문자 a, y의 위치관계를 보면, P의 크기에 해당하는 부분-a, y의 윗 부분-이 공백으로 되어 있음을 알 수 있다. 마찬가지로 소문자p ①, ③번에서는 a, h의 아랫 부분이 공백으로 되어있다. 이와 같이 4가지의 경우는 공백위치로 구별하고, 대문자 P의 ③번과 소문자p의 ②번은 그 방법으로는 구별이 불가능하므로 이 경우는 P 또는 p의 문자부분에 해당하

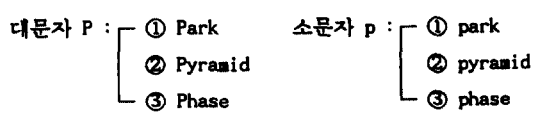

그림 1. P와 p의 구별
Fig. 1. Classification of P and p

는 제일 윗행과, 각 단어의 문자부분에 해당하는 제일 윗줄과의 공백 크기가 12화소보다 크면 소문자로, 작으면 대문자로 구분하도록 하였고 이에 대한 정보를 문자화일 끝에 저장한다.

1.2 잡음제거

문자에서 점이나 선에 나타나는 한 화소씩 벗어난 점들 즉 잡음들은 차후의 세선화와 특징추출을 하는데 있어서 불필요한 정보로 작용하므로 제거한다.

1.3 배율 조정

입력문자들의 불균일한 크기를 일정하게 하기위한 과정으로서 이 배율 조정에 대한 알고리듬들[8, 9, 10]의 문제점, 예를 들어 "d"나 "o"와 같은 경우 확대 조정을 하면 가운데의 공백이 메워지는 경우를 없애기 위해서 본 논문에서는 배율조정에 있어 개선된 알고리듬을 제안하는데 먼저 문자의 중심을 찾아 중심으로부터 방사형으로 원하는 만큼 배율조정을 하고, 배율이 1보다 큰 경우에는 채우기과정을 거쳐 완전한 확대 또는축소된 문자를 얻는다. 배율 조정에 대한 블록 다이어그램이 그림 2.에 나타나 있다. 배율 조정은 먼저 문자에 대해 방사형의 확대, 축소를 위해서 문자의 중심점을 검출한다. 그리고 확대 또는

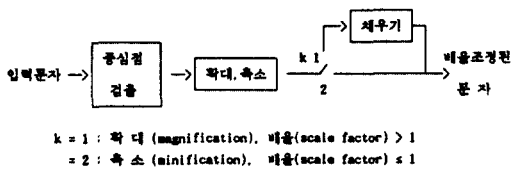

그림 2. 배율 조정 블록 다이어 그램
Fig. 2. Scaling block diagram

축소할 배율값을 계산한다. 이는 배율 조정전의 문자 화일에서 문자의 가로 세로크기를 각각 측정하여 그중 큰 값에 대한 확대 또는 축소 문자 크기의 비로 계산한다. 즉,

$$\text{배율(scalefactor)} = \frac{\text{확대, 축소 문자의 크기}}{\text{배율 조정 전의 문자 크기}} \quad (1)$$

이다. 배율 조정전 문자의 가로 세로크기중 큰 값을 택하는 이유는 작은 쪽이 배율 조정으로 너무 비대해지는 것을 막기 위해서이다. 배율을 계산한 후에 그 값이 확대에 해당하면 문자에 해당되는 화소(1)와 화소(1)사이의 배경 화소(0)를 배경과 문자와는 다른 임의의 값으로 표시하는데 이는 전술한 공백이 메워지는 문제점을 없애기 위해서이다. 그 다음 배율 조정 변환을 하는데 이는 문자의 각 화소(i, j)들을 배율만큼 중심(M, N)을 중심으로하여(m, n)으로 이동시킨다. 그림 3.로부터

$$M = \text{scalefactor} \times (M - i) + m$$
$$N = \text{scalefactor} \times (N - j) + n \quad (2)$$

이고, 식(2)를 m과 n에 대해 정리하면,

$$m = \text{scalefactor} \times i + M \times (1 - \text{scalefactor})$$
$$n = \text{scalefactor} \times j + N \times (1 - \text{scalefactor}) \quad (3)$$

이다. 그러므로 위 식(3)에 따라 문자 화일중의 문자 부분에 해당하는 점(i, j)를 점(m, n)으로 확대 또는 축소한다.

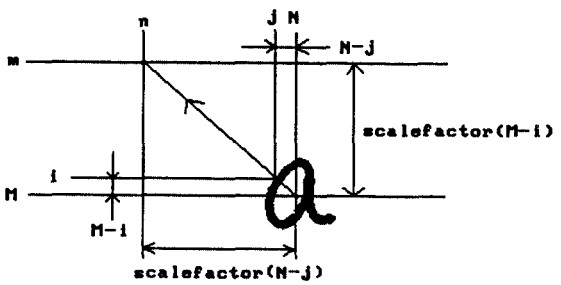

그림 3. 배율 조정 변환
Fig. 3. Scaling transform

확대의 경우, 배율 조정 변환된 후 화소들이 연결되지 않는 경우가 생기는데 이를 연결하기 위한 작업으로서 채우기를 한다. 채우기는 그림 4.와 같은 3×5 마스크를 사용하여 표 1.에 적용시켜서 효과적으로 수행할 수 있다. 즉 그림 4.에서(i, j)의 화소값이 1인 경우, 그리고 P5가 1인 경우 표 1.로부터 P3가 1이 된다. (i, j)의 화소값이 임의의 값(×)인 경우도 마찬가지이다. 수행 순서는 먼저 임의의 값(×)에 대하여 채우기를 하고, 그 다음 문자(1)에 대해서 수행한다. 그리고나서 임의의 값(×)을 원래의 배경인 0으로 만든다. 배율이 2보다 큰 경우는 4×7 마스크로, 3보다 큰 경우 5×9 마스크로 그림 4.와 표 1.

을 확장시킨다.

표 1. 채우기 조건
Table 1. Filling condition

Pi=1(x)	Pi=1(x)
P1	P1′
P2 또는 P3 또는 P4	P2′
P5	P3′
P6 또는 P7 또는 P8	P4′
P9	P5′

P1	P1′	(i, j)	P5′	P9
P2	P2′	P3′	P4′	P8
P3	P4	P5	P6	P7

그림 4. 채우기 마스크(3×5)
Fig. 4. Filling mask(3×5)

1.4 세선화

세선화 과정은 필기도구 및 사람에 따라서 다른 글자의 굵기를 일정하게 하고 특징추출을 쉽게 하기 위한 것으로서 두꺼운 글자의 일부를 제거하여 한 두 화소의 가는 글자를 만드는 것이다. 이때 제거해야 하는 점들은 경계선위에 존재해야 하며종결점이 아니어야 하고 글자의 연결성을 파괴해서는 안된다. 단 너무 지나치게 제거하므로서 선의 붕괴가 일어나지 않게 해야 한다.[11,12] 이와 같은 기준에 따라 다음과 같은 알고리듬을 수행한다.

P5	P4	P3
P6	P1	P2
P7	P8	P9

그림 5. P1을 중심으로 한 8개의 이웃
Fig. 5. 8 Neighbor of center point P1

먼저 문자 화일의 한 화소를 P1이라 할 때 8개의 이웃을 그림 5.와 같이 정의하고, P1의 8개 이웃중 화소 값이 1인점의 수를 B(P1)이라 하고, P_2, P_4, P_6, P_8중 적어도 하나가 0인 점을 경계점, 8개의 이웃중에 하나만의 1인점을 갖는 점을 끝점, P1의 8개 이웃중 제거 후보점이 아닌 화소수를 N(P1), 그리고 교차점 X(P1)은 식(4)와 같이 정의한다.

$$X(P1)=\sum_{i=2}^{5} bi \qquad (4)$$

여기서 bi는 P(2i−1)이 1이거나 P(2i)가 1이고, P(2i−2)가 0일때는 1로, 그외의 경우는 0으로 한다. Xn(P1)은 Pn을 1로 가정한 X(P1)이라고 할 때, 세선화 단계는 표 2.와 같다.

표 2. 세선화 단계
Table 2. Thining steps

단계 1	P2+P4+P6+P8≤3……P1이 경계점
단계 2	B(P1)≥2……P1은 끝점
단계 3	N(P1)≥1……세선의 끝에 있는 1의 화소가 반복적으로 제거되는 것 방지
단계 4	X(P1)=1……P1은 분리점이 아님
단계 5	P4가 제거 후보점이 아니거나 $X_4(P1)=1$……연결성을 유지
단계 6	P6가 제거 후보점이 아니거나 $X_6(P1)=1$……연결성을 유지

한 문자의 모든화소에 대해 위와 같은 6가지 조건을 순차적으로 적용한다. 한 화소에 대해 6가지 조건이 모두 만족되면 그 점을 제거후보점으로 지정하고, 모든 화소에 대해 위 조건을 수행하여 마지막 단계에서 지정된 제거후보점들을 제거(1에서 0)으로)한다. 다시 제거후보점이 존재하지 않을 때까지 위 과정을 반복 수행한다.

2. 특징 추출

특징 추출은 문자분류에 필요한 문자의 특징을 추출하여 특징 백터를 만드는 과정으로 이에 따라 인식할 수 있는 문자의 변형정도가 결정된다.

(1) 특징 마스크 적용

세선화를 거친 문자에 어느 특징이 포함되어 있는가를 알아내기 위해 5×5의 특징 추출 마스크를 적용한다. 마스크적용 결과는 어떠한 특징이 데이터의 어느 부분에 존재하는가를 나타내주는 특징 맵이다. 사용되어진 특징 추출 마스크는 그림 6.에 나타나 있는데, 본 논문에서 필기체 영문자에 맞는 마스크를 제안하였다. 그림의 각 마스크의 값과 그에 해당하는 데이터값을 곱하고 각 마스크에 대한 바이어스를 더해준 값이 0보다 크면 마스크의 가운데에 해당하는

데이터는 그 마스크에 대한 특징으로 간주하여 1로, 0보다 작으면 0으로 만든다. 그림 6.의 (d)와 (e)에서 ▨부분은 45°나 135° 방향의 위, 아래 끝점을 추출하기 위해서 부가적으로 취한 것이다. 이와 같이 한 문자 데이터에 대해서 20개의 특징 마스크를 적용하여 20개의 특징 맵을 만든다.

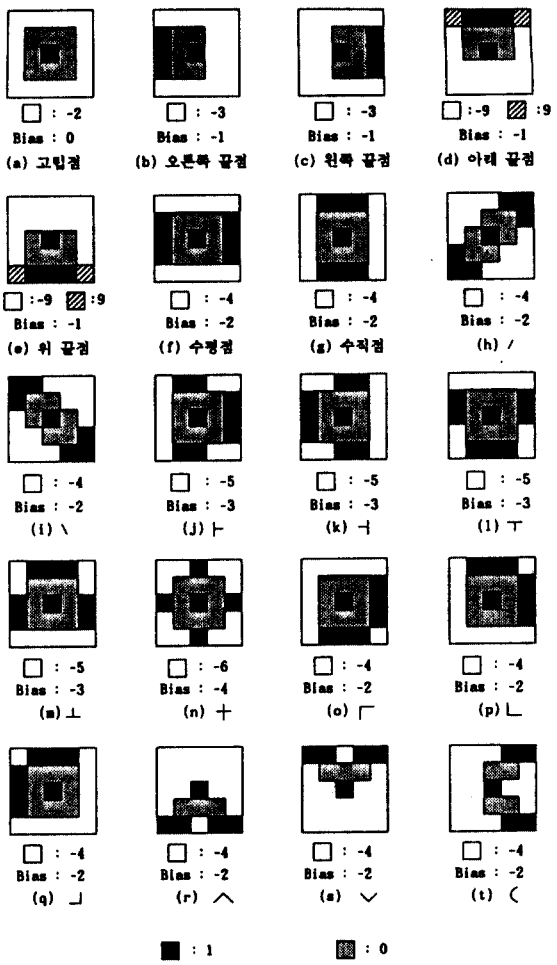

그림 6. 특징 추출 5×5 마스크
Fig. 6. Feature extraction 5×5 mask

(2) 특징 맵 블록화

문자 데이터를 그림 6.의 5×5 마스크로 스캐닝하면서 마스크 적용하면 20개의 각 특징마다 특징 맵을 얻게 되어, 이 특징 맵을 그대로 사용하면 특징 벡터의 크기가 매우 크므로 분류과정에서 처리할 데이터 양이 너무 많고, 각 특징은 실제 문자인식에 있어서는 대략적인 위치만이 중요하기 때문에 26×26배열의 데이터를 3×3배열로 블록화 하여 전체 특징 벡터를 20×26×26바이트에서 20×3×3바이트, 즉 180바이트로 축소한다.

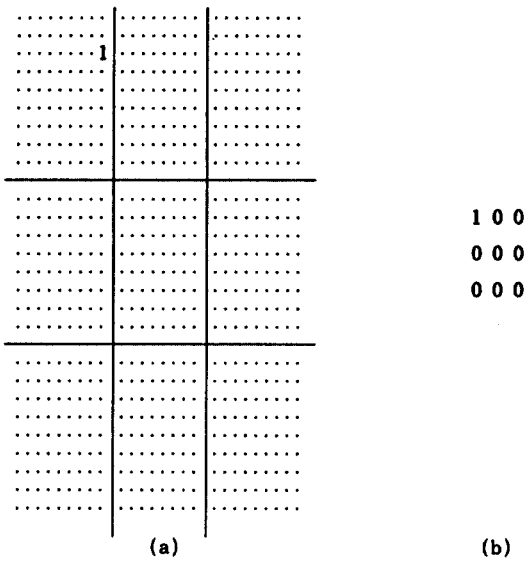

그림 7. 특징 맵 블록화 결과
(a) 위 끝점 특징 맵
(b) 3×3으로 블록화한 결과
Fig. 7. Result of feature map blocking
(a) Upper end point feature map
(b) Result of 3×3 blocking

그림 7.의 경우 (a)의 위 끝점 특징 맵인 26×26 배열 데이터를 3×3블록으로 나눈 다음, 각 블록안에 "1"의 갯수가 하나 이상이면 3×3 배열의 해당 위치를 "1"로 만들고 그 결과가 (b)이다. 그림 7.과 같은 특징 맵 블록화 과정을 20개의 특징 맵에 모두 적용한다.

(3) 부가 특징 맵 추출

위 (2)과정으로부터 얻어진 특징 맵 블록화 결과인 180개 특징 벡터를 사용할 경우 한 화소 차이로 특징 영역이 벗어나는 경우가 있다. 이와 같은 경우 추후의 인식 단계에서 오인식의 원인이 되기 때문에 이점을 개선하기 위한 방법을 제안하는데, 즉 20개의 특징에 대한 180개의 특징 맵으로써 각 특징의 존재

여부를 알 수 있고, 부가적으로 3가지의 특징 맵을 더하여 위와 같은 오인식의 원인을 줄일 수 있다. 실제로 180개의 특징 벡터를 사용했을 경우 인식율이 89.5%였고, 부가 특징 맵을 더한 192개의 특징벡터를 사용했을 경우는 93.9%로 약 4%의 인식율 증가를 보였다.

그림 8.(a)에서와 같이 세선화된 문자 데이터로부터 ① 5, 6번째 행, ② 20, 21번째 행, ③ 11, 12번째 열에 대해서 4부분의 블록화를 하는데 각 블록안에 "1"이 하나라도 있으면 그 부분의 특징맵을 "1"로 한다. 각 3가지의 블록에 대해 4바이트의 특징 맵이 추출되므로 12바이트의 부가 특징 맵을 180바이트의 특징 벡터에 더한다. 그림 8.(b)은 "b"에 대해 부가 특징 맵을 더한 192바이트의 특징 벡터를 나타내는데, 그림중에서 역상부분이 추가된 12바이트의 부가 특징 맵부분이다.

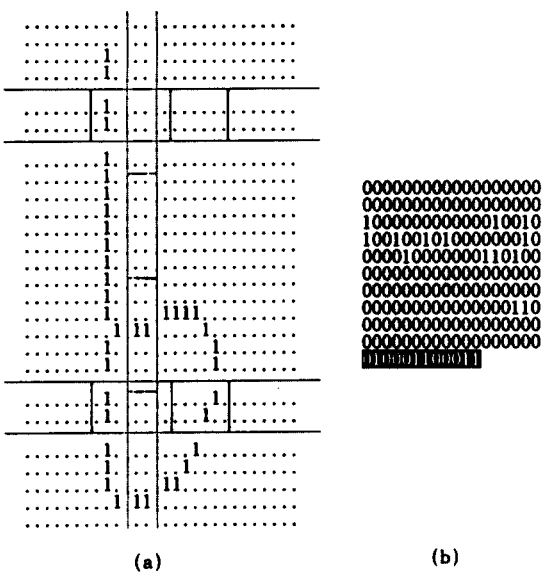

그림 8. 부가 특징 맵 추출
(a) 12부분의 블록 나누기
(b) (a)에 대해 20개 특징 마스크와 부가 특징 맵 추출을 한 결과

Fig. 8. Additional feature map extraction
(a) Block seperation of 12 parts
(b) Feature extraction and Additional feature map extraction result for(a)

(4) 대·소문자 구별 정보

문자 분리시에 화일 끝에 저장됐던 다섯 가지의 문자 정보를 읽어 현재의 문자에 대해서 대소를 구별한 다음, 192바이트의 특징벡터의 끝에 1바이트의 대·소문자 구별 정보를 저장한다. 대·소문자 구별 알고리듬은,

정의 : Char-대·소문자 구별 변수

① 문자에 해당하는 행(row)의 크기가 18화소보다 크면 Char=1(대문자), 작으면 Char=0(소문자)로 한다.(18화소의 기준은 실험결과이다.)

② 위 ①에 의해 Char=1(대문자)일 경우, K가 대문자이면 Char=2, P가 대문자이면 Char=Char+2

로 한다. 이와 같이 할 경우 Char값이

0일때 : 소문자　　1일때 : 대문자

2일때 : 대문자이고, 문자가 K일 경우 대문자

3일때 : 대문자이고, 문자가 P일 경우 대문자

4일때 : 대문자이고, 문자가 K, P일 경우 대문자

가 된다. 이 Char의 4가지 값중에 하나를 대·소문자 구별정보로 저장한다.

(5) 공백과 개행문자의 특징 벡터

공백과 개행문자의 특징 벡터를 다른 문자들의 특징 벡터와 다르게 하기위해 공백은 192개의 특징 벡터중 앞 3개만 1로, 나머지는 0, 대·소문자 구별 정보는 0으로 하고, 개행문자는 192개의 특징 벡터중 뒤 3개만 1로, 나머지는 0, 대·소문자 구별 정보는 0으로 한다.

3. 신경회로망의 학습과 인식

3.1 신경회로망의 학습

특징 추출된 192개의 특징 벡터가 그림 9.와 같은 역전파 알고리듬을 이용한 2층 퍼셉트론의 각 입력 노드에 들어간다. 학습시에는 대·소문자 구별 바이트는 사용하지 않는다. 출력노드의 갯수는 45개로 원하는 출력에 대한 문자분류 코드값은 45개의 출력 노드에서 각각 한 문자에 대해 한 노드만 1로 하고 나머지는 0으로 하였다. 문자분류 코드에 공백과 개행문자를 포함시켜 추후의 단어 인식때에 각 단어의 분류를 쉽게 하였고, 영문자의 종류가 43개인 것은 소문자와 모양이 같은 대문자는 분류 코드에 포함시키지 않았기 때문이다.

필기체 영문자중에서 각 문자에 대해 문자 발생

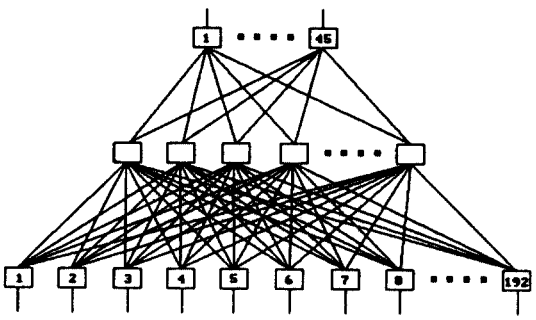

그림 9. 학습을 위한 2층 퍼셉트론
Fig. 9. Two layer perceptron for learning

빈도에 따라 학습 문자를 선택하여 공백과 개행문자를 포함하여 신경회로망에 학습시킨다. 학습은 원하는 출력과 실제 출력과의 오차를 측정하기 위한 전향단계와 오차를 줄이기 위하여 연결강도를 조정하는 후향단계로 나눌 수 있으나[13] 신경회로망의 단점인 긴 학습 시간을 줄이기 위해서 Falman이 제안한 수정된 학습 알고리듬을 사용하였다.[6]

(Ⅰ) 전향 단계

1) 192개의 특징점을 가진 학습 데이터와 그에 따르는 45개의 출력(target) 데이터를 결정한다.

2) 연결 강도 W_{ji}와 W_{kj}를 −0.5와 +0.5 사이의 임의의 값으로 초기화 한다.

3) 현재의 W_{ji}와 W_{kj}값에 따라 모든 뉴런의 실제 출력값을 계산한다.

(Ⅱ) 후향 단계

1) 각각의 입력에 대해 모든 뉴런에 대한 오차를 계산한다.

출력층 k의 오차

$$\delta_k=(t_k-O_k)O_k(1-O_k) \tag{5}$$

이고, 은닉층 j의 오차

$$\delta_j=O_j(1-O_j)\sum_k \delta_k W_{kj} \tag{6}$$

가 된다. 이때 출력층 k의 오차를

$$\delta_k=(\tanh^{-1}(t_k-O_k))(O_k(1-O_k)+0.1) \tag{7}$$

로 바꿔주는데, 역전파 알고리즘은 실제 출력값 O_k가 0또는 1에 가깝고 이때 원하는 출력과 실제출력과의 차인 (t_k-O_k)가 1에 근접하는 경우 실제오차값 δ_k는 0에 가까운 값이 되므로 학습시간이 매우 길어지는 단점이 있다. 그래서 이점을 해결하기 위해 식(5)에서 시그모이드함수를 미분한 항인 $O_k(1-O_k)$에 식(7)과 같이 0.1을 더해 주었고 (t_k-O_k)에 hyperbolic arctangent함수를 취하여 (t_k-O_k)가 클수록 그 값을 더 크게 δ_k에 반영 시키므로서 학습시간을 크게 감축시켰다. hyperbolic arctangent함수는 1과 −1에서 +∞와 −∞가 되므로 이를 방지하기 위해 (t_k-O_k)가 +0.9999999보다 크면 +20으로, −0.9999999보다 작으면 −20으로 함수값을 정하였다. 그림 10.은 출력층 k의 오차에 대해 식 (5)를 사용했을 때와 식(7)을 사용했을 때의 총 오차와 학습시간과의 관계를 나타낸다. 이 그림에서 식(7)을 사용했을 때의 수렴학습시간이 훨씬 빠른 것을 알 수 있다.

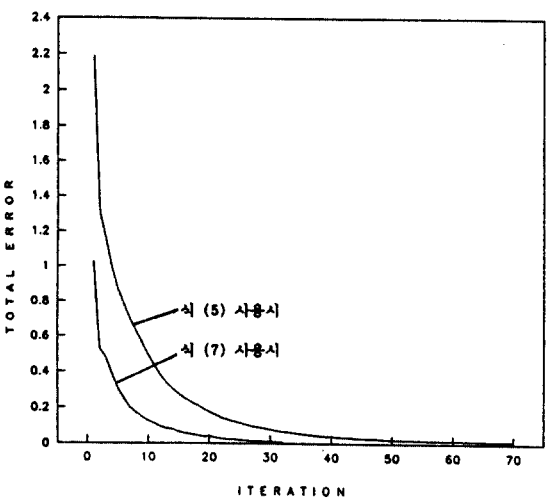

그림 10. 출력층 오차 계산법에 따른 학습시간 비교
Fig. 10. Learning time according to output layer error calculation

2) 각각의 연결강도를 조정한다.

즉 출력층과 은닉층의 연결강도(weight)의 변화량

$$\Delta W_{kj}(n+1)=\eta\delta_k O_j+\alpha\Delta W_{kj}(n) \tag{8}$$

이고, 은닉층과 입력층의 연결강도 변화량

$$\Delta W_{ji}(n+1)=\eta\delta_j O_i+\alpha\Delta W_{ji}(n) \quad (9)$$

이다. 여기서 η : 학습율, α : 모멘텀

이다. 모멘텀은 바로 이전의 연결강도 변화량을 얼마만큼 적용시킬 것인가를 나타내며, 연결강도를 변화시킴과 동시에 바이어스를 변화시킴으로써 국부최소점(local minima)에 빠지는 것을 어느 정도 방지하고 학습시간을 줄이는데 유용하다. 학습 시간을 짧게하고 인식율을 높이기 위해서 학습율η와 모멘텀α의 최적치를 결정한다.

3) 계산된 총 오차가 오차의 한계(0.01)에 이를 때까지 전향 단계부터 과정을 되풀이 하고, 작으면 학습이 완료된 것으로 한다.

3.2 신경회로망의 인식

학습을 마친 후 학습 패턴과, 학습 패턴을 포함한 시험 패턴을 인식 회로망에 입력으로 주어 각각의 출력뉴런의 값을 얻는다. 이 값들 중에서 가장 큰 값을 1로 하고 나머지는 0으로 하여 어느 부류에 속하는 문자인지를 분류한다. 분류된 결과 문자가 c, k, o, p, s, v, w, x, z일 경우 대·소문자 구별 정보를 이용하여 다음과 같이 문자 분류를 한다.

① 분류된 문자가 c, o, s, v, w, x, z일때,
대·소문자 구별 정보가 0이면 소문자, 1이면 대문자.

② 분류된 문자가 k이고, 대·소문자 구별 정보가 2이거나 4이면 k는 대문자, 아니면 소문자.

③ 분류된 문자가 p이고, 대·소문자 구별 정보가 3이거나 4이면 p는 대문자, 아니면 소문자.

4. 사전을 이용한 단어 인식

문자별로 신경회로망을 이용하여 인식을 한 후 오인식이 됐을 경우의 오류 문자 수정을 위해 단어별로 인식후에 사전과 비교하여 사전안의 단어와 같거나 가장 오차가 적은 단어를 찾아 그것을 최종 인식 단어로 한다. 즉 처음 문자 입력시에 영문자로 된 한 단어를 입력하므로 그 단어의 각각의 한 문자에 대해서 신경회로망의 인식 과정을 거친 후 인식된 단어를 사전의 단어들과 비교하여 그 결과의 사전단어를 출력한다.

(1) 사전 구축

먼저 사전은 단어만을 써 넣는 것으로 하여 단어의 문자수별로 화일을 만든다. 단어의 끝에는 사전의 기본어휘에 따른 1에서 4까지의 빈도수를 주었다. 빈도수 4는 중학교 기본어휘이고, 3은 고등학교 기본어휘, 2는 대학 교양어휘, 1은 일반어휘로 주었다. 이 빈도수는 인식단어와 사전과 비교후 가장 오차가 적은 단어들이 여러 개일 경우 빈도수가 가장 큰 단어를 최종 인식 단어로 하기 위함이다.

(2) 사전과 비교

인식 단어와 사전속의 단어와의 비교 알고리즘은 다음과 같다.

① 인식된 단어의 문자수를 계산한다.

② 문자수에 해당하는 사전 화일을 선택한다.

③ 인식 단어에 대한 사전 검색(searching)을 수행한다. 즉 인식 단어와 사전의 각각의 한 단어를 비교하는데 이때 오차가 가장 적은 단어들도 함께 검색한다.

④ 위 ③의 결과 완전히 정합(matching)되는 단어가 있을시 그 단어를 최종 인식 단어로 한다.

⑤ 위의 ③의 결과 정합 문자가 없을시, 오차가 가장 적은 단어가 1개일 경우는 그 단어를, 2개 이상이면 단어 끝의 빈도수가 가장 큰 것을, 가장 큰 빈도수의 단어가 1개이상일 경우는 제일 앞의 단어를 최종 인식 단어로 한다.

이와 같은 방법으로 최종 인식 단어를 출력한다.

Ⅲ. 실험 결과 및 고찰

1. 데이터 입력

본 논문은 영문자 데이터들을 얻는데 있어서 일반성과 신뢰성을 주기위해 영어를 쓸줄 아는 다양한 계층의 사람들, 즉 14세부터 50세까지의 남녀로부터 50개의 단어별로 2,000자의 필기체 영문자를 데이터 용지에 쓰게 하였다. 문자를 쓸때 약간의 제한 요소를 두었는데, 이는 문제를 간략화하고 불필요한 처리 과정을 생략하기 위해서 필요하다. 즉 필기도구는 연필이나 볼펜을 사용하도록 하였고, 가로, 세로 각각 1Cm 크기의 네모 안에 쓰도록 하였다. 데이터 용지로부터의 문자 데이터 입력방법으로 주변 밝기나 데이터 입력 공간에 영향을 거의 받지 않고 비교적 입력 데이터의 형상이 원글자와 거의 같은 스캐너를 사용하였는데, 이는 본 연구가 문자 데이터 입력시의 상태에 따라 인식율에 큰 영향을 주기 때문이다. 그림 11.은 스캐너를 통하여 입력된 2,000자의 영문자

데이터중 일부이다.

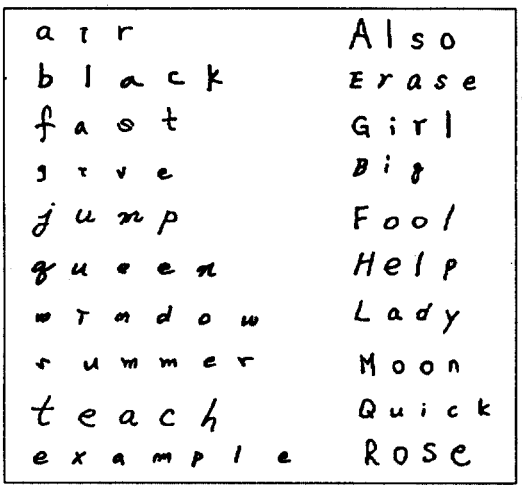

그림 11. 입력된 필기체 영문자 데이터
Fig. 11. Handwritten english input data

2. 문자 인식

2,000개의 필기체 영문자 데이터 중에서 532개의 학습패턴을 선택하여 2층 퍼셉트론을 학습시켰고, 인식은 문자별로 출력한 결과이다. 모든 시뮬레이션은 IBM-PC / 486 33MHz의 환경에서 수행되었다.

1) 은닉층(hidden layer) 노드수에 따른 인식율의 변화

은닉층은 주어진 문제를 해결하기 위한 국부적인 정보를 저장하는데, 이 은닉층의 노드수는 학습시간에 많은 영향을 준다. 일반적으로 학습속도는 은닉층의 노드수가 많을수록 연결 강도에 대한 계산량이 많아지므로 느리게 된다. 신경회로망의 수렴은 이론적으로는 오차가 0이 될 때이나 이러한 경우는 거의 존재하지 않으므로 계산된 오차가 미리 정해준 기준치보다 작을 때 수렴됐다고 본다. 본 연구에서는 오차 기준을 0.01로 놓았다. 학습율은 0.9 모멘텀은 0.2로 하고 은닉층의 노드수를 10개에서 60개까지 10단계씩 나누어 변화시키면서 반복횟수와 학습패턴과 시험패턴에 대한 인식율을 조사하였고 그 결과가 표 3.에 나타나 있다. 각각에 대해 반복횟수를 200번으로 제한하여 이때까지 학습을 완료하지 못하면 현재의 학습정보를 가지고 학습을 끝내게 했다. 표 3.에서 노드수가 늘어날수록 반복횟수는 급격히 감소하다가 35개부터는 28번 전후로 거의 같게 되는데, 이로부터 은닉층의 노드수가 늘어날수록 계산해야 할 연결강도의 수는 많아지지만 그 수가 어느 정도 이상이 되면 수렴시간에만 영향을 줄뿐 수렴되는 전체 오차는 거의 같음을 알 수 있다. 학습시간은 30개일 때가 가장 빨랐으며 수렴속도의 변화에 대한 그림이 그림 12.에 나타나 있다. 이는 은닉층의 노드수가 적당한 수까지 증가할수록 정보의 분산효과가 커져 각 노드가 배워야 할 부분이 적어지므로 쉽게 학습을 할 수 있으며, 그 수가 많아지면 필요 이상의 정보 분산을 가져와 학습이 어려워지기 때문이다.

인식율은 시험 패턴에 대해 은닉층의 노드수가 30개일때 가장 큰 값인 93.9%를 보였고 노드수가 증가함에 따라 90~92.5%로 거의 변화를 보이지 않았다. 노드수가 10개일때 인식율이 낮은데 이는 회로망이 입력 패턴에 대한 정보를 모두 수용할 수 없어서 더이상 원하는 값에 수렴하지 못하여 적응능력이 떨어지기 때문이다.

표 3. 은닉층의 노드수에 따른 반복횟수와 인식율의 변화
Table 3. The vaiation of iteration and recognition rate based on hidden layer number.

은닉층 노드수	반복 횟수	인식율 (%)	
		학습 패턴	시험 패턴
10	157	97.6	87.6
20	41	99.9	89.9
30	32	100	93.3
40	28	100	91.5
50	27	99.9	91.1
60	26	99.5	90.1

(2) 학습율(η)에 따른 인식율의 변화

학습시에 전체 오차의 최소값에 도달하기 위해서는 각각의 오차의 변화폭을 작게 해야 하지만 실제로 계산상의 반복횟수가 늘어나 학습시간이 길어진다. 그래서 변화폭을 크게 하여 반복횟수를 줄이기 위해서는 학습율을 크게 해야 하나 진동하게 되는 경우가 발생한다. 본 연구에서는 은닉층의 노드수를 30개, 모멘텀을 0으로 하고 학습율을 0.1에서 0.9까지 변화시키면서 각각의 반복횟수와 인식율을 조사하였다. 표 4.에 그 결과가 나타나 있다. 이로부터 학습율이 클수록 빨리 수렴한다는 것을 알 수 있고, 인식율의 변화는 0.6과 0.9에서 92.8%로 같았다.

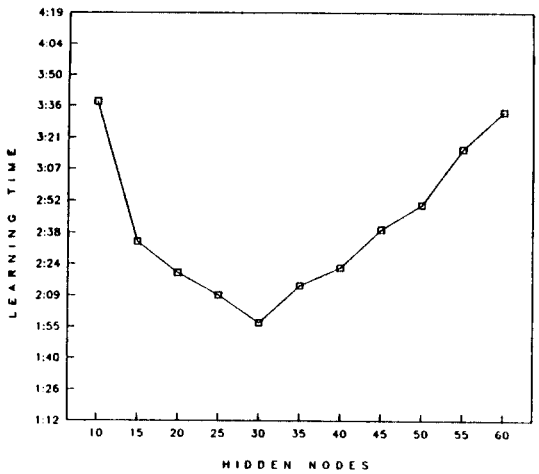

그림 12. 은닉층의 노드수에 따른 수렴속도의 변화
Fig. 12. The variation of learning time based on hidden layer number

(3) 모멘텀(α)에 따른 인식율의 변화

모멘텀은 오차 수정시에 현재의 오차량 뿐만 아니라 이전의 수정량도 고려하여 오차로부터의 수정량에 가속도를 주어 학습을 빠르게 하는 작용을 한다. 본 연구에서는 모멘텀의 최적값을 구하기 위하여 은닉층의 노드수를 30개, 학습율을 0.9로 하고 모멘텀을 0.1부터 0.9까지 변화시키면서 인식율을 조사하였고 그 결과가 표 5.에 나타나 있는데 모멘텀이 0.2이서 수렴속도가 가장 빨랐고 그 값이 커질수록 늦어지다가 0.7이상에서는 진동하여 수렴하지 못했다. 이로부터가중치 변화에 있어서 이전값의 반영율이 적은 쪽에서 수렴이 잘 된다는 것을 알 수 있고, 진동

표 4. 학습율에 따른 반복횟수와 인식율의 변화
Table 4. The variation of iteration and recognition rate based on learning rate

학 습 율	반 복 횟 수	인 식 율 (%)	
		학습 패턴	시험 패턴
0.1	200	98.9	89.9
0.2	141	99.8	91.0
0.3	101	99.9	90.9
0.4	82	100	90.9
0.5	70	100	91.3
0.6	62	100	92.8
0.7	56	100	92.2
0.8	51	100	92.3
0.9	48	100	92.8

발생 이유는 이전의 연결 강도의 반영율이 너무 크게 현재의 연결 강도에 작용을 했기 때문이다. 인식율 역시 모멘텀이 0.2일때 93.3%로 가장 좋았으나 전체적으로 거의 변화가 없었다.

표 5. 모멘텀에 따른 반복횟수와 인식율의 변화
Table 5. The variation of iteration and recognition rate based on momentum

모 멘 텀	반 복 횟 수	인 식 율 (%)	
		학습 패턴	시험 패턴
0.1	46	100	93.2
0.2	41	100	93.3
0.3	43	100	92.8
0.4	49	100	92.7
0.5	55	99.9	92.7
0.6	65	99.9	92.0
0.7	진동		

3. 단어 인식

위 2.의 결과를 근거로 하고 사전을 이용하여 문자가 아닌 단어별로 인식을 수행했는데, 사전과의 비교 방법은 4절의 (2)를 따랐다. 사전의 단어는 글자수가 1자에서 11자인 500개 단어(문자수는 2,310자)로 하였고 데이터 입력시 받아들인 50개의 단어(문자수는 2,000자)에 대해 단어 인식 과정을 수행하였다. 그에 대한 결과가 표 6.에 나타나 있다. 표 6.의 문자별 인식의 인식율은 위 1.(은닉층노드의 수 30개, 학습율 0.9, 모멘텀 0.2일때)의 결과이고 단어별 인식의 인식율은 같은 조건하에서 사전을 이용해 인식한 결과이다. 단어별 인식을 했을 경우 문자별 인식때보다 인식율이 93.3%에서 97.1%로 약 4% 높아졌는데 이로부터 오류 문자 수정을 할 수 있다.

표 6. 50개 단어(2,000자)에 대한 문자별 인식과 단어별 인식의 결과비교
Table 6. The result of character recognition and word recognition about 50 words(2000 characters)

	인 식 율 (%)
문자별 인식	93.3
단어별 인식	97.1

Ⅳ. 결 론

본 연구에서는 필기체 영문자를 인식하는데 있어

서 인간의 시각 시스템을 모방한 신경회로망을 이용하는 인식 시스템을 제시하였다. 제안된 시스템은 필기체 영문자를 데이터 용지로부터 받아 들여 인식율을 높이기 위한 방법으로서 전처리 과정을 거쳤는데 본 논문에서는 전처리의 여러 부분에서 필기체영문자에 맞는 여러 알고리듬들을 제안하였다. 학습과 인식을 위해 2층 퍼셉트론으로 구성된 신경회로망을 구성하였으며 학습은 역전파 알고리듬을 이용하였다. 학습시간을 줄이기 위한 방법으로 수정된 출력층의 오차계산법을 이용하였으며 이 방법은 수렴속도를 높이는데 상당한 효과를 얻었다. 신경회로망의 입력 노드수는 192개로 하였으며 출력 노드는 45개, 은닉층의 노드수는 30개로 하여 532개의 학습패턴에 대해 학습시켰다. 인식 과정에서는 은닉층의 노드수와 학습율, 모멘텀이 학습시간과 인식율에 미치는 영향을 조사하였고 본 논문에서는 인식율을 높이는 한 방법으로서 신경망을 통한 인식 단어와 사전의 단어를 비교하여 오류 문자를 수정하는 단어별 인식을 수행하였다. 인식율은 전체 패턴 50개 단어의 2,000자에 대하여 문자별 인식은 93.3%를 나타냈으며, 사전을 이용하여 단어별 인식을 했을 경우 97.1%로서 인식 오류 문자의 수정에 이 방법이 큰 효과가 있음을 알 수 있었다. 이와 같이 신경회로망과 사전을 이용하여 필기체 영문자의 인식 시스템을 설계하는데 있어 본 연구에서 사용된 알고리듬들을 사용하면 영문자 정보 문서의 인식에 큰 도움을 줄 것으로 기대된다. 제안된 방법의 앞으로의 연구 과제로서는 필기체 영문자에 맞는 빠른 전처리 과정의 개발과 학습시간을 빠르게 하고 인식율을 높일 수 있는 새로운 학습 알고리듬과 신경망의 구조에 대한 연구가 필요하다. 그리고 제안된 시스템의 실시간 처리와 신경망의 병렬처리를 위해서 하드웨어의 구현이 필요하다.

참 고 문 헌

1. K. S. Fu and S. Y. Lu, "A Clustering Procedure for Syntatic Patterns," IEEE Trans. on SMC, Vol.SMC-7, No.10, pp.732-742, Oct. 1977.
2. W. H. Abdulla, et al., "A Preprocessing Algorithm for Handwritten Character Recognition," Pattern Recognition Letters, Vol 7, pp.13-18, Jan. 1988.
3. R. L. Kashyap and B. J. Oommen, "An Effective Algorithm for String Correction using Generalized Edit Distance," Information Science, Vol.23, pp.123-142, 1981.
4. Bernard Widrow, et al., *DARPA Neural Network Study*, 1987-Feb., AFCEA Int. Press, 1988.
5. T. Kohonen, *Self-organization and Associative Memory*, 2nd Ed, Berlin, Springer-Verlag, 1988.
6. S. E. Falman, "Faster-Learning Variations on Backpropagation : An Empirical Study," Proc. Connectionist Models Summet School, Carnegie Mellon University, pp.38-51, 1988.
7. Bor-Shenn Jeng, "Optical Chinese Character Recognition using Accumulated Stroke Features," Optical Engineering, Vol.28, No.7, pp.793-799, July 1989.
8. A. Namane and M. A. Sid-ahmed, "Character Scaling by Contour Method," IEEE Trans, PAMI, Vol. 12, No.6, pp.600-606, June 1990.
9. Robert A. Ulichney and Donald E. Troxel, "Scaling Binary Image with Telescoping Template," IEEE Trans. on PAMI, Vol.PAMI-4, No. 3, May 1982.
10. S. Y. Lee, S. Yalamanchili and J. K. Aggarwal, "Parallel Image Normalization on A Mesh Connected Array Processor," Pattern Recognition, Vol.20, No.1, pp.115-124, 1987.
11. N. J. Naccache and R. Shinghal, "An Investigation into the Skeletonization Approach of Hilditch," Pattern Recognition, Vol.17, No.3, pp. 279-284, 1984.
12. Satoshi Suzuki, "Binary Picture Thinning by An Iterative Parallel Two-subcycle Operation," Pattern Recognition, Vol.20, No.3, pp.297-307, 1987.
13. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representation by Error Propagation," Parallel Distribution Processing, Vol.1, pp.318-362, MIT Press, 1987.

金 應 成(Eung Sung Kim) **正會員**
1967年 2月 22日生
1989年 2月 : 성균관대학교 전자공학과 졸업(공학사)
1992年 2月 : 성균관대학교 대학원 전자공학과 졸업(공학석사)
1992年 3月～現在 : 성균관대학교 대학원 전자공학과 박사과정 재학중

李 根 泳(Keun Young Lee) **正會員**
1947년 12월 30일생
1973年 : 전남대학교 전기공학과 졸업
1975年 : 한양대학교 대학원 전자공학과 석사학위 취득
1978年 : 한양대학교 대학원 전자공학과 박사학위 취득
1979年 3月～1980年 2月 : Denmark 공과대학(연구)
1987年 9月～1988年 8月 : 영국 Loughborough대학(연구)
1977年 3月～1981年 8月 : 광운공대 조교수
1981年 9月～現在 : 성균관대학교 전자공학과 교수
※관심분야 : 스윗칭이론, 마이크로 프로세서, 영상 처리, 신경회로망

趙 成 桓(Seong Hwan Cho) **正會員**
1980年 2月 : 成均館大學校 電子工學科 卒業
1982年 2月 : 成均館大學校 大學院 電子工學科(工學碩士)
1991年 8月 : 成均館大學校 大學院 電子工學科(工學博士)
1982年 9月～1985年 7月 : 海軍士官學校 教授部 專任講師
1985年 9月～現在 : 大宥工業專門大學 電子計算機科 助教授
※관심분야 : 神經回路網, 映像處理, 패턴認識