論文 90-27-8-11

한글 인식을 위한 신경망 분류기의 응용

(A Neural Net Classifier for Hangeul Recognition)

崔 源 昊* 崔 東 赫* 李 炳 來* 朴 圭 泰*

(Won Ho Choi, Dong Hyuk Choi, Byung Rae Lee, and Kyu Tae Park)

要 約

본 논문에서는 마할라노비스 거리를 분류의 기준으로 삼고 적응성을 갖는 신경 회로망 분류기(AMDC : adaptive Mahalanobis distance classifier)를 설계하였다.

이 분류기는 입력단, 내부단, 출력단으로 구성되어 있으며, 입력단과 내부단 사이는 완전 연결되어 있고, 내부단과 출력단 사이는 부분 연결 방식으로 되어 있다. 출력 노드의 갯수는 분류할 클래스의 수와 같으며, 내부 노드의 갯수는 학습시에 형성된 서브클래스의 갯수와 같다.

학습시에 입력패턴이 들어오면 현재의 대표 벡터와 편차 벡터값을 이용하여 학습하며, 학습이 완료된 후 추가 학습이 필요하더라도 학습을 처음부터 다시할 필요가 없어 현 상태에서 추가되는 부분만 학습한다.

AMDC는 특징공간 상에서 서브클래스를 형성하고, 특징 분포 모양을 고려하므로 패턴 변형 흡수가 용이하다.

초기 편차θ와 threshold α를 변화시켜가며 한글 500자(multi font한글 인쇄체, 각 자당 23종, 총 11500자)를 분류하는 실험을 수행한결과, 이들 값이 학습 패턴의 분포에 적합한 부분(G 영역)에서 성능이 향상되었으며, 분류율, 내부 노드수를 검토하여 기존 분류기와 비교하였다.

Abstract

In this paper, using the neural network design techniques, an adaptive Mahalanobis distance classifier (AMDC) is designed. This classifier has three layers: input layer, internal layer and output layer. The connection from input layer to internal layer is fully connected, and that from internal to output layer has partial connection that might be thought as an Oring.

If two ormore clusters of patterns of one clsss are laid apart in the feature space, the network adaptively generate the internal nodes, which are corresponding to the subclusters of that class.

The number of the putput nodes in just same as the number of the classes to classify, on the other hand, the number of the internal nodes is defined by the number of the subclusters, and can be optimized by itself. Using the method of making the subclasses, the different patterns that are of the same class can easily be distinguished from other classes.

If additional training is needed after the completion of the training, the AMDC does not have to repeat the trainging that has already done.

To test the performance of the AMDC, the experiments of classifying 500 Hangeuls were done. In experiment, 20 print font sets of Hangeul characters (10,000 characters) were used for training, and with 3 sets (1,500 characters), the AMDC was tested for various initial variance θ and threshold α, and compared with other statistical or neural classifiers.

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

接受日字 : 1990年 6月 25日

Ⅰ. 서 론

신경 회로망의 응용 분야로는 음성 합성및 인식, 패턴 인식, 영상처리등의 많은 분야가 있지만 그중에서도 일반적으로 많이 응용되고 있는 분야는 패턴인식이다.

패턴 인식은 컴퓨터 응용의 실용성이 매우 높이 기대되어 이미 오래전 부터 연구되어 왔으며 많은 연구 성과가 있었다. 기존의 패턴 인식에 관한 연구에서는 원형 정합(template matching), 구조 분석, 통계적 방법 등이 주로 사용되었다. 원형 정합 방법은 패턴 변형시 성격이 급격히 저하되며, 구조적 방법은 구조를 파악하여 인식하므로 인식률이 높지만, 자소 접촉과 잡음에 취약하다.[1] 통계적 방법에서 이상적인 특징(feature)을 사용하면 분류가 잘되어 인식률이 높아 지지만 패턴의 변형을 흡수할 수 있는 특징 추출이 힘들다.[10-11]

신경 회로망은 학습을 통해 문자를 인식하는 동적(dynamic) 시스템이므로 잡음에 강한 신경망의 구성이 가능하여[12] 입력 패턴이 어느 정도 변형되더라도 인식하는 데에 큰 문제가 없으며 병렬 처리가 가능하다.

신경망 회로의 학습은 지도 학습(supervised learnning)과 자율 학습(unsupervised learning)의 두 가지 방법이 있다. 지도 학습은 주어진 입력에 대하여 원하는 출력을 정하여 훈련시키는 방법이고, 자율학습은 출력을 지정하여 주는 teacher가 없이 학습을 진행한다. Hamming 신경회로 모델, Adaline, BP, 최적 연상 변환(optimal associative mapping)등이 지도 학습 방법을 취하고 있으며, ART, 자율 특징 변환(self-organizing feature maps)등은 자율 학습의 학습 방법을 갖는다.

한글과 같이 클래스 갯수가 방대할 때, BP를 사용하여 많은 클래스를 학습시키려면 학습 시간이 길어지며 수렴의 보장이 없다. 따라서 학습의 소요시간이 짧고 확실하게 수렴해 주는 모델이 필요하다. 그리고, 문자를 인식함에 있어서 많은 클래스에 부분적으로 적합한 특징을 사용할 수 밖에 없는데, 이렇게 되면 특징 공간(feature space)상에 샘플이 불규칙적으로 퍼져 버리게 된다. 분류의 성능을 높이기 위해서는 이러한 샘플들의 모양에 적응적으로 동작하는 분류기가 필요하다.

본 논문에서는 문자인식의 정확도를 높이기 위하여 샘플의 무리 모양에 적응적, 자기 조직적(self-organizing)으로 동작하여 소무리(subcluster)를 형성하여 주고 마할라노비스 거리(Mahalanobis distance)와 같이 무리의 모양을 고려한 거리 측정법을 사용한 분류기를 설계하였다. 본 논문에서 제안하는 분류기는 지도학습 방법으로 훈련시켜서 하나 또는 두개의 루프(loop)로 학습이 되므로 매우 빠른 시간내에 학습이 완료되며, 테스트 알고리듬이 간단하고 명료하므로 구현하기가 용이하다.

Ⅱ. 분류기 설계

1. 분류기 구조

본 논문에서 설계하고자 하는 분류기는 적응성을 갖는 마할라노비스 거리분류기(AMDC;adaptive Mahalanobis distance classifier)이다. AMDC는 입력단, 내부단, 출력단의 3단으로 구성되어 있다. 입력단과 내부단의 연결은 완전 연결(full connection)상태이다. 입력단의 노드갯수는 입력패턴의 차원수와 같으며, 내부단은 학습시 하나씩 발생되므로 내부단의 노드갯수는 샘플들의 모양과 학습조건에 따라 달라진다.

BP에서는 출력 노드와 내부 노드의 갯수가 고정된 상태로 학습을 하며, 내부 노드의 갯수에 따라 전체 신경망의 성능이 영향을 받는다. 내부 노드수를 많게하면 구조가 복잡해지는 대신 수렴을 잘 하며, 반대로 내부 노드수를 적게하면 구조가 간단해지지만 수렴하기가 어렵다. 따라서 적절한 내부 노드의 갯수를 설정하기란 매우 어렵다.[7]

반면에 AMDS는 학습과정에서 출력과 내부 노드를 를 적응적으로 발생시켜 주므로 BP와 같은 비 수렴에 의한 제한성이 없으며, 신경망 자체가 적응적으로 동작하여 주목 영역(attention area)을 넓혀 가므로 내부 노드의 갯수가 적당한 선에서 제거될 수 있다. 내부단의 노드에서는 입력과 각 노드들이 갖고

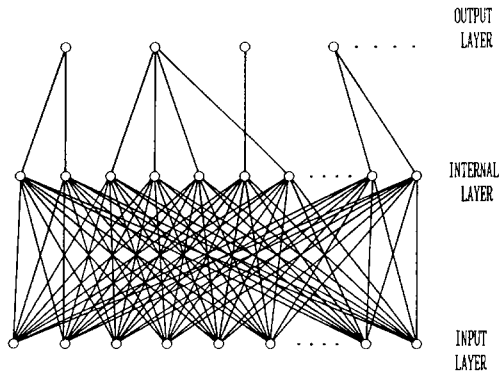


그림 1. AMDC의 구조
Fig. 1. Structure of the AMDC.

있는 중심점과의 마할라노비스 거리를 계산해 준다. 계산된 값을 자신과 연결된 출력 노드로 보내준다. 하나의 클래스를 작은 클래스들의 집합으로 생각해서 작은 클래스에 해당하는 내부 노드를 소속된 클래스에 해당하는 출력단에 연결하여 준다. 이러한 작은 클래스(subclass)를 필요로 하는 경우가 문자 인식의 경우에 특히 많이 발생한다. 본 논문에서 제안하는 신경 회로망은 이러한 소무리를 형성하여 준다. 소무리의 형성이 필요한가, 불필요한가를 판단하여 소무리 형성의 명령을 주는것이 아니라, 감독자가 학습 패턴과 소속 클래스만을 지정하여 주면 신경망은 소무리가 필요한가, 아니면 기존의 무리에 같이 넣어 줄 것인가를 판단하여 스스로 동작하게 된다. 소무리를 많이 갖고 있는 클래스는 그 클래스에 해당하는 출력노드가 많은 내부 노드와 연결이 되고, 소무리를 하나만 갖고 있는 출력 노드는 하나의 내부 노드와 연결이 된다.

출력단에서는 자신과 연결된 내부 노드에서 계산된 값을 받아서 그중 가장 작은 것을 출력시킨다. 하나의 내부 노드만이 연결된 출력 노드는 받은 값을 그대로 출력시키면 된다. 출력 노드의 출력 중에서 가장 작은 값을 내보내는 클래스로 입력 패턴을 귀속시킨다.

본 논문에서 제안하는 분류기는 입력 패턴이 어느 클래스에 속하는가를 결정하는 판단의 기준으로서 다음과 같이 정의 내린 거리를 사용하는데 이는 마할라노비스 거리의 단순화된 형태이다.

$$D_i^2 = \sum_{j=1}^{n} \frac{(x_j - m_{ij})^2}{d_{ij}^2} \qquad (1)$$

x_j : 패턴 벡터

m_{ij} : 클래스 i의 중심 벡터

d_{ij} : 클래스 i의 Covariance matrix의 대각선 성분

$d_{ij} < \theta$인 경우, $d_{ij} = \theta$

아니면, d_{ij}

2. 대표 벡터 **m**

생성된 내부 노드는 신경망이 입력 패턴을 학습할 때 대표 벡터를 만드는데, 이 벡터는 그 노드에 귀속된 입력 패턴들의 평균값이다. 학습시에 해당 클래스를 지정하고 입력 패턴을 넣으면 지정된 출력 노드는 자신에게 속한 내부 노드들을 활성화 시킨다. 활성화된 내부 노드들은 입력되고 있는 패턴 벡터와 자신이 갖고있는 대표 벡터와의 거리를 계산해서 주목 영역(attention area)내에 입력 패턴이 있는가 또는 밖에 있는가를 판단한다. 만약 주목 영역안에 있으면 다음 식에 의하여 대표 벡터 **m**을 이동시킨다.

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \frac{1}{t+1}\{\mathbf{x}_i(t+1) - \mathbf{m}_i(t)\} \qquad (2)$$

$$\mathbf{m}_i(1) = \mathbf{x}_i(1) \qquad (3)$$

여기서 $\mathbf{m}_i(t)$는 i번째 내부 노드에 소속된 학습 패턴이 t개 일 때의 대표 벡터를 뜻한다. 따라서 $\mathbf{m}_i(t+1)$은 이 노드에 포함되는 학습패턴이 하나 더 들어왔을 때의 대표 벡터이다. $\mathbf{x}_i(t+1)$은 i번 째 내부 노드에 t+1번 째로 포함된 입력패턴 벡터이다. 활성화된 모든 내부 노드의 주목 영역 안에 입력 패턴이 없을 때에는 새로운 내부 노드를 하나 생성시켜 지정된 출력 노드와 연결하고 식(3)과 같이 그 내부 노드의 대표 벡터는 입력패턴과 같게 한다.

학습 입력패턴이 지정된 출력 노드에 의해 활성화된 내부 노드 중 여러개의 주목 영역 안에 포함될수도 있다. 그림 2에서 처럼 X점에 입력패턴이 존재하면 서브클래스 1, 2, 3 모두의 주목 영역 안에 들어오게 된다. 이런 경우에는 출력 노드에서 입 패턴과 가장 가까운 거리에 대표점을 갖고 있는 서브클래스를 선정하여 그 서브클래스로 입력패턴을 귀속시켜서 대표점을 이동시킨다. 다른 서브클래스들은 비록 입력패턴이 자기의 주목 영역 안에 있더라도 출력노드로 부터 선정되지 못하면 아무런 동작을 하지 않는다. 여기서 거리라 함은 마할라노비스 거리를 의미한다.

기존의 분위기는 학습 입력벡터를 모두 받은 후 평균을 계산하거나 입력이 완료된 후에 학습 단계로 들어가므로 학습이 끝난 다음 추가로 학습이 더 필요한 경우에는 처음부터 다시 학습을 해야한다. 그러

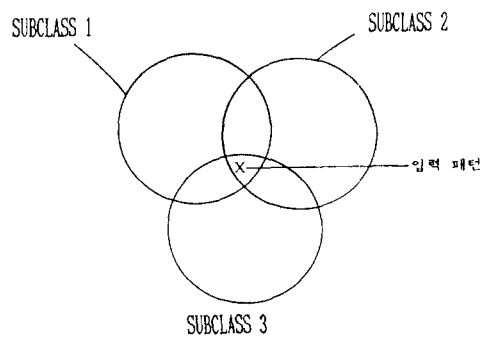


그림 2. 입력 패턴이 여러개의 서브클래스의 주목 영역안에 있는 경우

Fig. 2. A input pattern can be put in attention areas of several subclasses.

나 본 논문에서 제안하는 분류기는 학습 입력벡터를 모두 받은 후에 학습단계로 들어 가는 것이 아니고 식(2)에서 보는 바와 같이 하나의 학습패턴이 들어오면 단일 루프(loop)에 의해 대표 벡터의 계산이 끝날 수가 있으며 학습전체는 하나 또는 두개의 루프로 완료된다. 또한 차기 대표점 $\mathbf{m}(t+1)$의 계산은 현 대리점 $\mathbf{m}(t)$에서 학습 입력벡터 $\mathbf{x}(t+1)$을 받으면 곧바로 계산될 수 있으므로, 학습이 완료된 상태에서 추가 학습이 필요한 경우에는 추가되는 학습패턴을 그대로 넣어 주어도 앞서 학습된 결과에 자동적으로 연결되어 학습될 수 있다.

학습에 의해 형성된 대표점은 분류된 클래스의 중심점은 아니며 그림 3에서 보는 바와 같이 서브클래스의 중심점이 된다. 그림에서 내부 노드와 출력 노드와의 연결은 ORing의 개념으로서

(subclass 1) OR (subclass 2) (4)

가 되고 클래스 2는 하나의 내부 노드만을 점유하므로 서브클래스 3의 중심점이 바로 클래스2의 중심점이 된다.

각 클래스에 속하는 학습패턴의 분포 모양에 따라 그 클래스가 점유하는 내부 노드, 즉 서브클래스의 갯수가 달라진다. 학습패턴이 널리 분포되어 있어서 하나의 서브클래스의 주목 영역 안에 모두 들어올 수가 없을때는 새로운 내부 노드를 발생시키게 된다.

새로 발생된 내부 노드의 초기 주목 영역(initial attention area)은 서브클래스의 갯수와 밀접한 관계가 있다. 초기 주목 영역을 넓게 잡으면 서브클래스의 갯수가 줄어들고, 좁게 잡으면 서브클래스의 갯수가 많아진다. 좁은 영역에 밀집된 패턴을 갖는 클래스는 하나 또는 몇 개만의 내부 노드를 갖게된다.

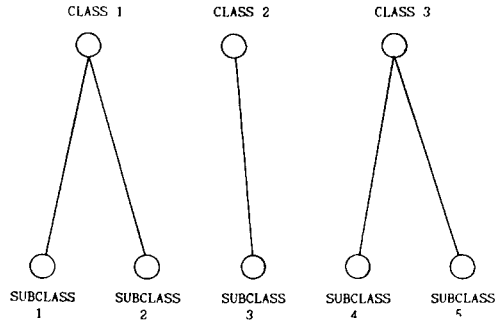


그림 3. 내부 노드와 출력 노드의 연결
Fig. 3. The connection between internal and output nodes.

3. 편차 σ와 주목 영역

본 논문에서 제안하는 AMDC는 내부단의 노드들이 자신에게 소속되는 학습 패턴들의 분포 모양에 적응하는 주목 영역을 갖는다. 즉 학습 패턴들의 편차가 크면 넓은 주목 영역을 갖게되며 편차가 작으면 좁은 주목 영역을 갖는다. 이와 같이 학습패턴들의 분포모양에 적응적으로 동작하는 것은 매우 중요한 의미를 갖는다. 같은 성능을 갖는 분류기라면 내부 노드의 갯수가 적을수록 좋다. 내부 노드의 갯수가 적음면 그만큼 PE(processing element)의 갯수가 줄어들고 연결(connection)도 간단해지기 때문이다. 대부분의 기존 분류기는 학습패턴의 분포 모양을 무시하고 내부 노드의 숫자를 고정시켜서 구성하고 있다. 이렇게 하면 필요 이상으로 내부 노드가 많이 사용되기도 하며, 몇 개의 내부 노드를 쓰는 것이 적절한지 파악하기가 어렵다. 그러나 AMDC는 주어진 초기값을 갖고 내부 노드를 적절히 만들어 가면서 패턴모양에 따라 주목 영역을 변화시키므로 학습자가 특별히 신경을 쓰지 않더라도 적합한 숫자의 내부 노드를 구성시킬 수 있다.

편차 벡터 σ는 대표점을 중심으로 하여 각 축의 편차를 나타내는 벡터이다. 즉

$$\sigma_i=[\sigma_{i1}\ \sigma_{i2}\ \sigma_{i2}\cdots\sigma_{in}]^T \tag{5}$$

로 표시된다. 여기서 σ_{ij}는 i번째 내부 노드의 j축에 대한 편차이다. 편차의 계산은

$$\sigma_{ij}^2=\frac{1}{t+1}\sum_{r=1}^{t+1}x_j^2(r)-m_{ij}^2(t+1) \tag{6}$$

의 식을 이용하면 된다. 이 식에서 $x_j(r)$은 i번째 내부 노드로 r번 째 입력한 j축 입력의 값, 즉 j번 째 입력 노드의 입력값이고 $m_{ij}(t+1)$은 i번째 내부 노드에서 $(t+1)$번 째 까지 입력된 학습 패턴들의 대표벡터의 j번째 원소이다.

학습시에 출력 노드에 의해 활성화된 내부 노드는 다음 식을 계산한다.

$$g_{ij}=\frac{(x_{ij}-m_{ij})^2}{d_{ij}^2}-\alpha \tag{7}$$

이 식은 내부 노드의 중심점에서 입력패턴과의 j방향에 대한 거리에서 threshold α를 뺀 값이다. 모든 j에 대하여 g_{ij}가 음수 또는 0이라면, 즉

$$g_{ij}\leq 0 \text{ for all } j,\ j=1,2,\cdots,n \tag{8}$$

의 조건이 만족되면 입력패턴은 내부 노드i의 주목 영역 안에 들어가게 된다. 이 식에서 α가 비록 상수이기는 하지만 각 내부 노드 별로 크기가 다른 주목

영역을 갖는다. 왜냐하면 식(7)의 제수 $(d_{ij})^2$은 내부 노드에 소속된 샘플의 분포 모양에 따라 그 값이 달라지기 때문이다. 주목 영역과 외부와의 경계선은

$$\frac{(x_{ij}-m_{ij})^2}{d_{ij}^2} \tag{9}$$

를 만족하여야 하고 이 식을 고쳐 쓰면

$$(x_{ij}-m_{ij})^2=\alpha\cdot d_{ij}^2 \tag{10}$$

가 된다. 따라서 유클리디안 거리의 개념으로 보았을 때 i번 째 내부 노드의 j축에 대한 주목 영역은 $\sqrt{\alpha\cdot d_{ij}}$가 된다. 그러므로 각 노드마다 또 각 축 별로 거리가 다른 주목 영역을 형성하게 된다.

내부 노드에서 출력 노드로 전달하는 값은 식(1)의 D_i를 그대로 출력 노드로 전달하되, 모든 방향에 대해 주목 영역 안에 입력이 존재하면 음부호를 붙이고 그렇지 않으면 그대로 출력 노드로 전달한다.

$$D_i'=\begin{cases}-D_i^2 & \text{if } \mathbf{x} \text{ is in attention area}\\ +D_i^2 & \text{otherwise}\end{cases} \tag{11}$$

출력 노드에서는 Z_i'값을 받아서 부호를 이용하여 어느 노드의 주목 영역 안에 입력패턴이 존재하는가를 판단하여 $|Z_i'|$값으로 어느 노드와 가장 가까운가를 결정할 수 있다.

초기 주목 영역은 α와 θ에 의해 결정된다. $\alpha\cdot\theta$가 크면 넓은 초기 주목 영역을 갖고, 작으면 좁은 영역을 갖는다. θ를 작게하면 내부 노드가 많이 발생하고 θ를 크게하면 내부 노드 수가 줄어드는 대신 무리의 분포 모양을 고려하기가 어렵다. α도 작으면 내부 노드가 많이 발생하고 너무 크면 서브클래스로 나누어져야 할 부분이 나누어지지 못하고 같은 서브클래스로 합쳐져 버리는 문제점이 있다.

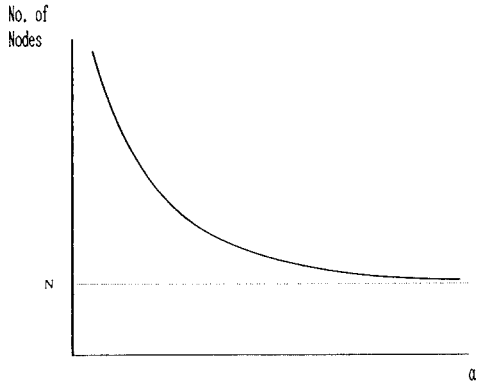


그림 4. 노드 수와 α와의 관계
Fig. 4. The relation between the number of nodes and α.

그림 4에서 보는 바와 같이 θ를 고정시켰을 때 α에 대한 노드의 갯수는 단조 감소의 형태가 된다. 노드의 갯수는 ν가 증가함에 따라 클래스의 갯수 N에 수렴한다. 노드의 갯수가 N보다 작을 수는 없는데, 그이유는 학습 시에 클래스가 처음으로 지정되면 새로운 내부 노드를 발생시키므로 모든 클래스, 다시 말해서 모든 출력 노드는 최소한 하나의 내부 노드를 갖게되기 때문이다.

4. 학습

학습시에는 각 노드 별로 갖고있는 대표 벡터 **m**과 편차 벡터 σ를 입력패턴에 따라서 수정해 나가며, 새로운 노드가 필요할 경우에 노드를 만들어 주고 이 노드와 출력 노드를 연결해 주는 일을 하게 된다.

학습 순서는 다음과 같다.

1. 입력패턴 **x**를 입력단에 넣고 이 입력패턴이 소속된 클래스에 해당하는 출력 노드를 지정한다. 지정된 출력 노드가 내부 노드를 갖고 있지 않는 새로운 노드라면 내부 노드를 발생시켜서 연결한다. 연결을 한다는 의미는 내부 노드와의 연결 가중치를 "1"로 한다는 뜻이다.

2. 새로 연결된 내부 노드의 대표 벡터 **m**을 현 입력패턴과 같게 놓고 편차는 초기치 θ로 한다. 즉,

$$\mathbf{m}_k=\mathbf{x} \tag{12}$$

$$\mathbf{d}_k=[\theta,\theta,\theta,\cdots,\theta]^T \tag{13}$$

로 놓는다. 여기서 k번째 내부 노드가 새로 연결된 것으로 가정한다. 대표 벡터와 편차 벡터의 차원은 입력 벡터의 차원과 같다.

3. 지정된 출력 노드가 이미 내부 노드를 갖고있는 경우에는 내부 노드에 적절한 신호를 내줌으로써 내부 노드를 활성화시킨다. 활성화된 내부 노드는 식 (11)과 같은 출력식을 만들어서 출력 노드로 보낸다. 내부 노드로 부터 Z_k'값은 받은 출력 노드는 받은 값들 중에 음수값이 존재하는가를 확인한다. 받은 값이 모두 양수이면 1번과 2번에서와 같이 새로운 노드를 발생하여 대표 벡터와 편차 벡터를 만들어준다. 음수가 존재할 때는 가장 작은 음수를 보내주는 내부 노드 만을 활성화시키고 나머지 내부 노드는 활동을 중지시킨다.

4. 다시 활성화된 내부 노드는 (1)식의 **m**과 (5), (6)식의 σ를 계산하여 새로운 **m**과 σ로 삼는다.

학습에 대한 흐름도를 그림 5에 그려 놓았다. 이 흐름도에서 보듯이 하나의 패턴에 대학 학습이 반복

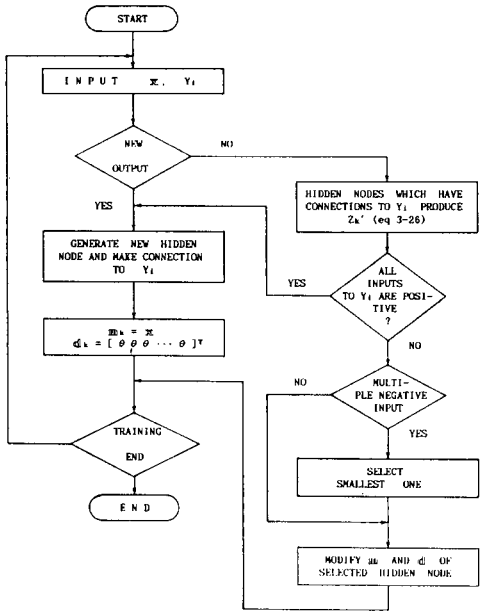


그림 5. 학습 과정의 흐름도
Fig. 5. The flow chart of the training procedure.

루프가 없이 끝나기 때문에 학습시간이 매우 짧게 소요되는 장점을 갖고 있다.

5. 테스팅

학습시에 입력패턴의 위치와 편차에 의해 계산된 대표 벡터 **m**과 초기치가 고려된 편차 벡터 **d**는 학습이 끝나면 내부 노드에 보존한다. 테스팅시에 모든 내부 노드는 입력 **x**에 대해 (11)식의 Z_i'을 계산한다.

$$\begin{cases} -\sum_{j=1}^{n} \dfrac{(x_{ij}-m_{ij})^2}{d_{ij}^2} & \text{if } (x_{ij}-m_{ij})^2 \le \alpha \cdot d_{ij}^2 \text{ for all } j \\ +\sum_{j=1}^{n} \dfrac{(x_{ij}-m_{ij})^2}{d_{ij}^2} & \text{if } (x_{ij}-m_{ij})^2 > \alpha \cdot d_{ij}^2 \text{ for any } j \end{cases} \quad (14)$$

식 (14)의 Z_i'은 D_i^2이 아닌 D_i로 해야 엄밀한 거리가 계산될 수 있지만 D_i의 크기를 비교하는 것이나 D_i^2의 크기를 비교하는 것이 마찬가지이고 D_i를 구하기 위해서는 제곱근을 구해야 하는 번거러움이 있으므로 그대로 D_i^2를 계산하도록 한다.

앞 식에서 구한 Z_i'을 출력 노드로 보내면 출력노드는 자신에게 연결된 내부 노드로 부터 오는 모든 Z_i'을 받아서 크기를 비교하여 가장 작은 값을 출력시킨다. 이 때에 부호는 관계없이 그 절대값을 비교한다. 출력 노드들이 내보낸 출력 값을 서로 비교하여 가장 작은 값을 출력하는 노드가 승자가 되어 입력 패턴은 그 출력 노드가 나타내는 클래스에 귀속된다.

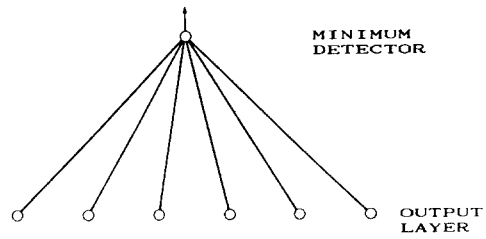


그림 6. 출력 노드의 최소값 선정
Fig. 6. The selection of an output node whose output value is the minimum.

Ⅲ. 실험 및 결과 고찰

1. 실험방법

본 논문에서 제안한 AMDC의 성능을 파악하기 위하여 한글패턴을 분류해 보았다. 금성, HP, 큐닉스사의 레이저 프린타로 인쇄된 각종 크기의 한글 폰트(font)를 입력으로 사용하였다.

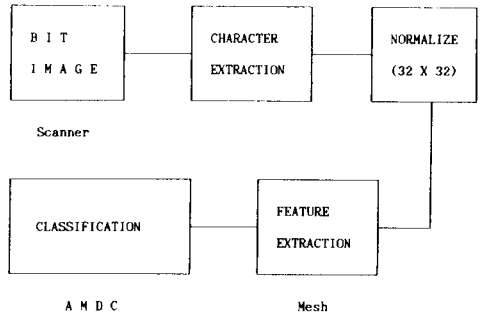


그림 7. 문자 인식 과정
Fig. 7. The procedure of the pattern recognition.

한글을 사용 빈도수 순서로 정리하여 레이저 프린터를 이용해서 인쇄한 후 스캐너로 입력한다. 이 중에서 공백은 버리고 문자만을 추출한다. 추출된 문자를 32×32 화소로 정규화(normalize)한다. 정규화된 문자로 부터 특징을 추출한다. 실험에 사용된 특징은 가장 단순한 메쉬(mesh)이다.

32×32로 정규화된 문자패턴을 8×8영역으로 나눈다. 하나의 문자를 이와 같이 나누면 모두 64개의

영역이 생긴다. 하나의 영역 안에 있는 문자를 구성하는 점의 화소수를 세어서 그 영역의 특징으로 삼는다. 이렇게 하면 한 문자의 특징은 64개 영역의 화소수를 센 값들을 원소로 한 벡터가 된다. 즉

$$\mathbf{F}_i=[C_{i1}\ C_{i2}\ C_{i3}\cdots\ C_{i64}]^T \qquad (15)$$

이다. $\mathbf{F}_i$는 i번째 문자의 특징 벡터이고, C_{ij}는 i 번째 문자의 j번 째 영역에서 문자를 구성하는 화소수를 센 값이다.

특징의 성능이 좋으면 분류기가 뛰어나지 않더라도 문자가 구분이 잘 되므로 인식을 하는데에 어려움이 없다. 그러나 특징이 그다지 좋지 않을 때는 분류기의 역할이 중요하다. 본 논문에서 매쉬 특징을 사용한 이유는 이 특징이 매우 단순하여 실험이 용이할 뿐 아니라 성능이 그다지 좋지를 않기 때문에 분류 능력을 실험하기에는 적절하기 때문이다.

특징 추출의 앞단에 있는 정규화 작업은 메쉬 특징을 사용하기 때문에 필요한 단계로서 다른 특징을 사용할 때에는 굳이 정규화를 시킬 필요가 없다. 추출된 특징은 AMDC의 입력이 된다. 식 (15)에서 보듯이 입력이 64차원이므로 AMDC의 입력단에 64 개의 입력 노드가 존재한다.

실험에 사용된 문자 패턴의 일부분을 그림8에 수록하였다. 실험에는 모두 23종의 문자 패턴이 사용되었으며 한 종류 당 500자 씩의 문자가 사용되었다. 그 중 20종의 문자 패턴을 갖고 학습시켰으며, 3종의 문자 패턴으로 테스팅을 하였다. 따라서 모두 10,000자가 학습에 사용되었으며, 1,500자로 테스팅을 하였다. 학습과 테스팅에 사용된 문자는 고딕체와 명조체가 섞여 있으며 회사 별로 분류해 보면 표 1과 같다. 실험 결과에 나타나는 분류율은 테스팅을 한3종의 문자에 대한 분류율의 평균이다.

표 1. 실험에 사용된 문자의 회사별 분류
Table 1. Character fonts used in the experiment.

회사명 \ 구분 / 자형		학 습	테 스 팅
금 성	명 조 체	6	1
	고 딕 체	5	1
H P	명 조 체	5	1
	고 딕 체	2	•
큐 닉 스	명 조 체	1	•
	고 딕 체	1	•
계		20	3

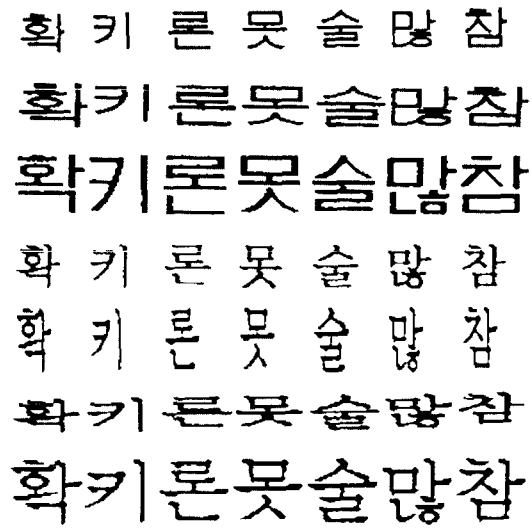


그림 8. 실험에 사용된 문자의 예
Fig. 8. The example of characters used in experiments.

성능의 비교를 위하여 몇 가지의 분류기를 실험하였다. AMDC와 함께 AMDC에서 마할라노비스 거리가 아닌 유클리디안 거리를 이용한 분류기를 실험함으로써 마할라노비스 거리를 사용하는 것의 장점을 알아 보았으며, KNN분류기, Parzen Windows 분류기와 성능을 비교하였다. 또한 클래스 내에서 서브클래스를 형성하지 않고 유클리디안 거리를 바로 비교하여 분류하는 유클리디안 거리 분류기(EDC:euclidean distance classifier)도 실험하였다. EDC를 실험할 때, 같은 클래스의 고딕체와 명조체 한글을 하나의 클래스로 하면 성능이 너무 저하되므로 서로 별도이 클래스로 하여 실험하였다. 즉 500자의 한글에 대하여 1000개의 클래스로 분류토록 하였다. 실험에 사용된 기종은 IBM PC-AT이며 여기에 FAST9 transputer board를 부착하여 실험하였다. FAST9은 9개의 32-bit transputer를 갖고 있으며 이 9 개의 transputer 모두가 실험에 사용되었다. 이때의 연산능력은 대략 90MIPS이다. 시뮬레이션에 사용된 언어는 Parallel C이다.

2. 실험 결과-분류율

AMDC를 한글 500자에 대하여 분류해 본 결과, 분류율은 threshold α(식 7)가 커짐에 따라 그림9와 같은 모양을 갖는다. α가 작을 때는 높은 분류율을 갖고 있다가 α가 커짐에 따라 분류율이 점점 낮아진다. 낮아지던 분류율은 어느 점($\theta=1.2$일 때는 $\alpha=2.6$)

에서 다시 약간 높아지고 그 상태가 어느 정도 유지되다가 다시 떨어지기 시작한다. 이와같이 내려가던 분류율이 다시 올라가는 이유는 α와 θ의 초기값이 특징 공간 상에서 학습패턴의 모양에 잘 들어 맞는 영역이 있기 때문이다. 그림에서 G영역이 그 경우이다.

G영역은 θ를 크게 잡으면 좌측에 위치하며 θ가 작으면 우측에 있게 된다. 다시 말해서 큰 θ에서는 α값이 작은 곳에서 G영역이 생기며, θ를 작게 하면 그림 8의 그래프에서 처럼 오른쪽에 G영역이 있게 된다.

α가 작을때는 분류율이 높지만 내부 노드가 많이 발생하는 문제점이 있다. 또한 α를 크게 하면 내부 노드가 적게 발생하여 신경 회로망이 간단해 지지만 분류율이 낮아지게 된다. 그러므로 분류율과 노드 갯수 사이에 적절한 타협이 필요하며, α와 θ값의 선택은 G영역내의 것을 사용하는 것이 좋다.

G영역의 분류율도 θ값에 따라 조금씩 다르다. 그림 8에서 $\theta=1.2$일 때 S영역의 분류율은 대략 94.5% 정도이지만 $\theta=1.8$인 경우에는 95.5% 정도이다. θ에 대하여 G영역의 분류율을 그래프로 그려보면 그림 9와 같다. 그림에서 보듯이 G영역의 분류율은 θ가 증가함에 따라 증가하다가 일정한 값을 유지한다. 실험한 결과에 따르면 메쉬 특징을 사용한 한글의 경우 θ가 1.8 이상의 것을 사용하는 것이 좋다고 판단된다. 물론 이때에 G영역 내에 들어가도록 θ에 맞는 α값이 선정되어야 한다.

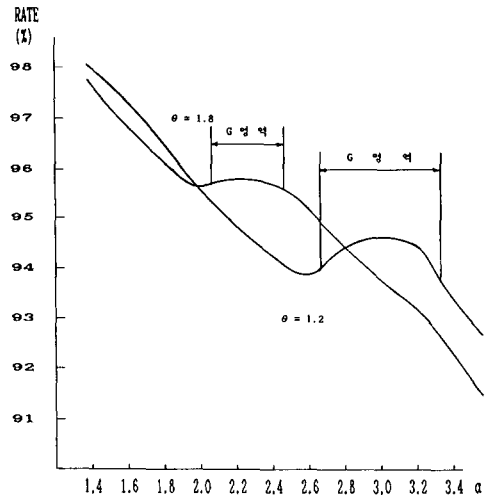


그림 9. α에 대한 분류율의 변화
Fig. 9. Relation between classifying rate and α.

G영역에서 내부 노드 갯수는 약 1500개 정도로서 명조체와 고딕체가 각 각 500개의 내부 노드를 만들어 주고, 나머지 500개 정도의 서브클래스가 더 형성된 셈이다. 이 서브클래스는 학습 패턴 중 같은 글자인데도 모양이 판이하게 다르거나, 잡음으로 인하여 변형된 글자에 의하여 생성된 것이다. 표2에서 보면 θ가 작은 경우에는 α가 비교적 큰 곳에서 G영역이 있게되고, θ가 크면 작은 α값에서 G영역이 형성된다. 내부 노드의 갯수는 대략 1500개 정도이며, $\theta=1.8$ 이상에서는 분류율이 비슷한 것을 알 수 있다.

회로의 복잡도에 관계없이 높은 분류율이 필요한 경우에는 α와 θ값을 작게하면 되고, 그다지 높지 않은 분류율로 만족할 때에는 내부 노드수가 작은것을 선택한다. 예로 표 3에 분류율이 94%일 때 θ에 대한 노드수를 살펴보았다. 이 표에서 보듯이 94%일 때는 $\theta=2.4, \alpha=2.6$으로 하는 것이 노드수가 가장 작으며, 경제적인 회로가 공성될 수 있다.

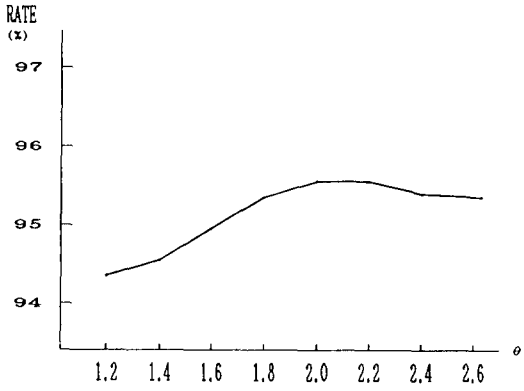


그림 10. G영역의 분류율
Fig. 10. Classifying rate of G regions.

표 2. θ에 대한 G영역의 분류율, 내부 노드수와 α
Table 2. Variations of the classifying rate, the number of internal nodes, and α for some values of θ.

θ / 구분	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6
분 류 율	94.31	94.64	94.91	95.30	95.57	95.50	95.40	95.30
내부노드수	1522	16275	1566	1500	1518	1500	1470	1470
α	3.0	2.8	2.6	2.5	2.4	2.3	2.2	2.1

표 3. 분류율 94%일 때의 내부 노드수
Table 3. The number of internal nodes in case that the classifying rate is 94%.

θ / 구분	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8
노드수	1308	1258	1218	1170	1150	1122	1112	1142	1129
α	3.3	3.1	3.0	2.9	2.8	2.7	2.6	2.5	2.4

3. 실험 결과-내부 노드수

AMDC의 내부 노드수는 α와 θ가 커짐에 따라 감소한다. 그림11에서 보듯이 α가 클수록, 또 θ가 클수록 내부 노드의 갯수가 줄어든다. α나 θ가 계속 커지면 이들은 클래스의 갯수에 수렴하게 된다. 실험에서 500자를 분류하기 위하여 500개의 클래스를 구성하였으므로 내부 노드는 500개에 수렴한다. 내부 노드가 500개에 가까와지면 명조체와 고딕체의 구분없이 출력 노드가 하나의 서브클래스를 갖게되므로 분류율이 급격히 저하된다.

앞 절에서 설명한 G 영역이 1500개의 노드 수 근처에서 형성되므로 그래프에서 이 때의 θ와 α 값을 알 수 있다. 본 논문에서는 문자의 구조를 고려하지 않은 메쉬 특징을 사용하였기 때문에 특징 공간 상에서 명조체와 고딕체의 위치가 많이 떨어지게 된다 따라서 서브클래스의 갯수, 즉 내부 노드의 갯수가 최소한 1000개 이상이 되는 것이 바람직하다. 만약 Peripheral이나 LDCD, GDCD[8,9] 와 같이 문자의 구조를 고려한 특징을 사용하면 G영역에서의 노드 갯수가 훨씬 줄어들 수 있다.

4. 분류기의 성능 비교

AMDC의 성능을 KNN, Parzen Windows, EDC (euclidean distance classifier), AEDC (adaptive euclidean distance classifier)와 비교하기 위하여 한글 500자에 대하여 같은 조건으로 실험하였다. KNN은 K값을 1에서 11까지 변화시켜 보았으며, Parzen Windows는 window의 크기를 300에서 700까지 변화시키며 실험하였다. KNN과 Parzen Windows는 하나의 학습 샘플당 하나의 내부 노드를 갖게된다. 따라서 모두 1000개의 내부 노드를 갖는다.

EDC는 하나의 출력 노드가 두 개의 내부 노드를 갖는데, 두 개의 내부 노드는 각각 명조체와 고딕체에 해당한다. 500개의 클래스에 대하여 학습시키려면 EDC 신경망은 명조체 500개, 고딕체 500개의 내

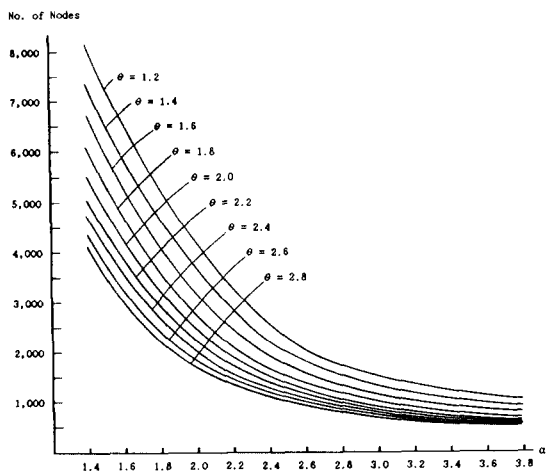


그림11. 노드 수와 α, θ와의 관계
Fig. 11. The variation of the number of nodes.

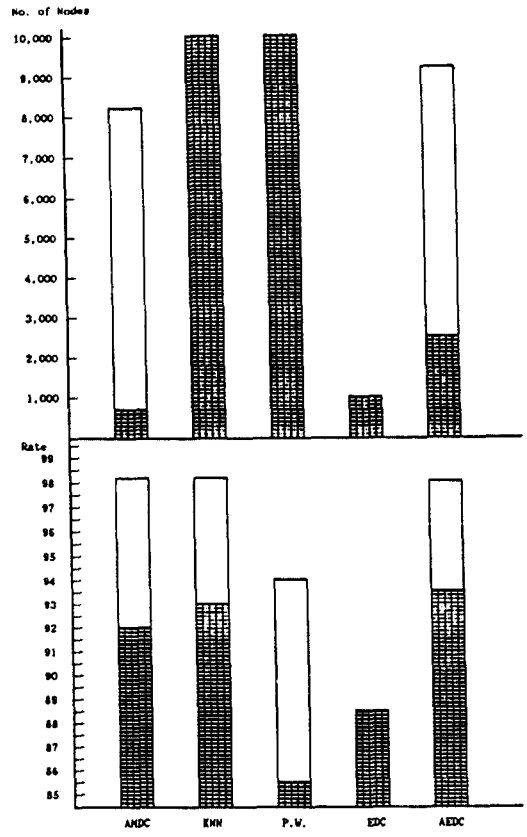


그림12. 여러 분류기의 분류율과 내부 노드수의 비교
Fig. 12. The comparison of the different networks.

부 노드가 발생하므로 모두 1000개의 내부 노드를 갖게 된다.

AEDC는 AMDC와 구조와 같으며, 학습시에 편차를 고려하지 않고 초기치로 주어진 α값을 반지름으로 한 주목 영역을 갖는다. AEDC에서 거리의 계산은 유클리디안 거리로 한다. AEDC의 분류 성능은 RCE와 매우 유사한 것으로 볼 수 있다.

이들 각 분류기의 봉류율과 내부 노드 갯수를 그림12에 그렸다. 이 그래프에서 AMDC의 분류율은 $\alpha=\theta=2.8$일 때 92% 부터 시작하여 $\alpha=\theta=1.2$일때 98.5% 까지의 분류율을 나타냈으며, 이 때 내부 노드의 갯수는 700에서 8200개 까지 변화한다.

KNN은 K=1일 때 98,5%를 나타냈으며, K=11에서 93% 정도의 분류율을 보였고 내부 노드 갯수는 10000개로 고정되어 있다. Parzen Windows는 window 크기가 300일 때 94% 정도이며 700일 때는 85.5% 정도의 분류율을 갖는다. 내부 노드의 갯수는 KNN과 마찬가지로 10000개로 고정되어 있다. EDC는 1000개의 내부 노드에 88.5%의 분류율을 나타내었다. AEDC는 $\alpha=3.0$일 때 분류율 93.5%, 노드 수 2550에서 $\alpha=1.4$일 때 분류율 98%, 노드 수 9200 정도의 성능을 갖는다.

Ⅳ. 결 론

본 논문에서 제안한 AMDC는 같은 클래스임에도 불구하고 패턴 모양이 판이하게 다른 경우에 적용적으로 서브클래스를 만들어 주어 하나의 클래스로 묶어 주므로, 문자와 같이 여러 모양의 같은 글자가 존재하는 패턴들을 분류할 때에 효과적으로 동작한다. 예를 들어서 "4"와 "4", "0"과 "0", "7"과"7"또는 "7"등은 같은 글자이지만 그 모양이 확연하게 다르며, 한글에서는 "ㅊ"과 "ㅊ", "ㅎ"과 "ㅎ", "ㅅ"과"ㅅ" 등과 같이 하나의 글자가 여러 모양을 갖고 있는 경우가 많다. 이러한 글자들은 모양이 많이 다르기 때문에 좋은 특징(feature)을 사용한다 하더라도 특징 공간상에서 상당한 거리로 떨어져있기 마련이다. 사람이 여러글자 모양에 대하여 모두 프로그램하여 처리할 수도 있겠지만 글자 모양의 다양함에 비추어볼 때 많은 노력과 시간이 필요할 것은 자명한 사실이다. 그러나 AMDC를 사용하면 글자 모양 별로 스스로 서브클래스를 생성시키고 하나의 출력 노드가 이들을 묶어서 하나의 클래스로 만들어 주므로, 글자 변형에 융통성이 있다. 하나의 클래스 또는 서브클래스가 갖는 특징 공간상의 샘플 분포 모양에 대해서도 적응적으로 동작하는 것이 중요하다 AMDC는 이러한 무리(cluster)모양에 적절하게 동작하기 위하여 서브클래스의 편차 벡터를 내부 노드가 계산하여 유지하며, 분류시에 이 편차 벡터를 이용하여 거리를 계산한다.

본 논문에서는 인쇄체의 한글 인식에 대해서만 실험을 하였지만 AMDC의 이러한 장점으로 인하여 인쇄체 문자 인식뿐만 아니라 필기체 문자 인식에서도 성능이 우수할 것으로 기대된다. 필기체 문자인식을 실험하기 위해서는 다른 분류기와 비교하기 위한 데이타 베이스의 구성이 필요하다.

參 考 文 獻

[1] 강현철, 최동혁, 이완주, 박규태, "원소 변환을 이용한 한글 패턴의 구조분석, 대한전자공학회 논문지, 제26권 제12호, 1989, pp. 61-69.

[2] V. Vemuri, "Artificial neural networks: an introduction," in *Artificial Neural Networks: Theoretical Concepts, IEEE*, 1988.

[3] T. Kohonen, *Self-organization and Associative Memory*, Springer-Verlag, 1984.

[4] T. Kohonen, "An Introduction to neural computing," *Neural Networks*, vol. 1, pp. 3-16, 1988.

[5] T. Kohonen, R. Chrisley, and G. Barma, "Statistical pattern recognition with neural networks: benchmarking studies," *in Neural Networks from Models to Applications*, I.D. S.E.T., Paris, pp. 160-167, 1989.

[6] R.O. Lippmann, "An introduction to computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4-22, April 1987.

[7] M. Bichsel, P. Seitz, "Minimum class entropy: A mximum information approach to layered networks," *Neural Networks*, vol. 2, pp. 133-141, 1989.

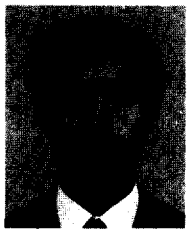
[8] N. Hagita, S. Naito, I. Masuda, "Recognition of handprinted Chinese characters by global and local direction contributivity desity feature,." *J66-D, vol. 6, pp. 722-729, 1983.*

[9] Y. Mori, K. Yokosawa, "Neural Networks that learn to discriminate similar Kanji characters," in *Advances in Neural Information* Processing Systems I,D. Touretzky ed., Morgan Kaufmann Pub., pp. 332-339, 1989.

[10] J.T. Tou and R.C. Gonzalez, *Patiern Recognition Principles*, Addison-Wesley, 1974.

[11] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley-Interscience, 1973.
[12] W.Y. Huang and R.P. Lippmann, "Neural net and traditional classifiers," *in Conference on Neural Information Processing Systems-Natural and Synthetic*, IEEE, pp. 387-396, Nov. 1987.

著 者 紹 介



李 炳 來(正會員)
1963年 10月 29日生. 1985年 2月 연세대학교 공과대학 전자공학과 졸업. 1987年 2月 연세대학교 대학원 전자공학과 공학석사. 현재 연세대학교 대학원 전자공학과 박사과정. 주관심분야는 영상처리 및 Computer vision 등임.

●

崔 源 昊 (正會員) 第27卷 第7號 參照
현재 울산대학교 전자및 전산기공학과 조교수

崔 東 赫 (正會員) 第26卷 第12號 參照
현재 연세대학교 대학원 전자공학과 박사과정

●

朴 圭 泰 (正會員) 第26卷 第7號 參照
현재 연세대학교 전자공학과 교수