공학 연구논문집 제22권 제1호 pp. 85~92, 1991. 울산대학교
Journal of Engineering Research Vol. 22, No. 1, pp. 85~92, 1991.
University of Ulsan.

부분 활성 신경망 분류기의 설계

최원호 · 최재하
전자공학과

〈要 約〉

본 논문에서는 무리의 위치에 적응적으로 동작하고 높은 분류율, 자기조직적 기능을 유지하면서 내부 노드 중 분류에 기여할 가능성이 있는 것 들 만을 활성화하는 PA-AEDC(Partially Activated-Adaptive Euclidean Distance Classifier) 신경망을 설계하였다. 이 신경망은 주신경망과 부신경망으로 구성되어 있으며, 주신경망은 분류의 일을, 부신경망은 주신경망 중 일부분을 선택하여 활성화해 주는 역할을 한다.

성능을 파악하기 위하여 분류 실험을 해본 결과, 분류율은 AMDC나 AEDC와 비슷하게 나왔으나, 분류 시 연산에 참여하는 노드 수는 PA-AEDC의 경우가 대폭 감소되었으며, 따라서 효과적인 동작이 가능하였다.

Design of Partially Activated Neural Net Classifier for Character Recognition

Choi, Won-Ho & Choi, Jae-Ha
Dept. of Electronic Eng.

〈Abstract〉

In this paper, PA-AEDC(Partially Activated-Adaptive Euclidean Distance Classifier) neural network is designed. This one can adaptively operate for the position of clusters and can self-organize the internal nodes and their connections. PA-AEDC consists of two networks:the primary net and the secondary net. The primary net mainly acts as a feature vector classifier, while the secondary net selects and activates the output and internal nodes of the primary net which are supposed to contribute in the classification.

As a result of experiments of classification, we got the expected classifying rates as high as that of AMDC or AEDC. But the number of nodes of PA-AEDC that participate in the arithmetic operation was much reduced.

Ⅰ. 서 론

문자 인식에 있어서 이상적인 특징(feature)을 사용하면 무리(cluster)가 잘 형성되어, 즉 하나의 무리 속에 특징 벡터들이 밀집되어 분류를 해 내는데 큰 어려움이 없겠으나 이상적인 특징을 찾기는 매우 어려운 실정이다. 문자수와 문자체가 다양한 경우에는 더욱 그러하다. [1, 2] 더구나 문자의 구조를 잘 반영하여 주고 잡음에 강한 특징들은 특징 추출과 분류 연산시에 연산량이 너무 많아지는 경향이 있다. 따라서 연산량과 인식률 사이의 적절한 타협을 위하여 불완전한 또는 부분적으로 적합한 특징을 사용할 수 밖에 없는데 이러한 특징을 사용하면 특징공간(feature space) 상에 샘플이 불규칙적으로 퍼져 버리는 문제점이 있다. 고전적인 분류기를 여기에 적용하면 정확도가 심각하게 저하되고 있다. [3]

본 논문에서는 무리화가 잘되지 않는 경우에도 소무리(subcluster)의 개념을 사용하여 적절히 동작할 수 있으며, 무리의 위치에 적응적으로 동작하는 신경회로망 분류기를 설계하고자 한다. 여기서 설계하는 분류기는 AMDC (Adaptive Mahalanobis Distance Classifier) [4]의 높은 분류률, 적응성, 자기조직적(self-organizing) 기능, 학습의 신속성등을 그대로 유지하면서 분류 알고리즘을 더욱 간단히하고 연산량을 줄이기 위하여, 테스팅 시에 내부 노드의 일부 만을 활성화시키며 Mahalanobis거리 대신에 단순한 Euclidean거리를 사용한 PA-AEDC (Partially Activated-Adaptive Euclidean Distance Classifier)이다. 유클리디안 거리를 사용하면 마할라노비스 거리를 써서 분류할 때 보다 분류율이 약간 저하되는 것는 사실이나 연산량에 있어서는 편차 계산이 필요없으므로 대폭 감소될 수 있다.

Ⅱ. AMDC

AMDC는 그림 1 과 같이 입력단, 내부단, 출력단의 3단으로 구성되어있다. 입력단과 내부단의 연결은 완전 연결(full connection)이고 내부단과 출력단은 부분 연결(partial connection)상태이다. 입력단의 노드 갯수는 입력 패턴의 차원 수와 같으며, 내부단은 학습 시 하나씩 발생되므로 내부단의 노드 갯수는 샘플의 모양과 학습 조건에 따라 달라진다. 출력단에서는 자신과 연결된 내부 노드에서 계산된 값을 만아서 그 중 가장 작은 값을 내보내는 클래스로 입력 패턴을 귀속시킨다.

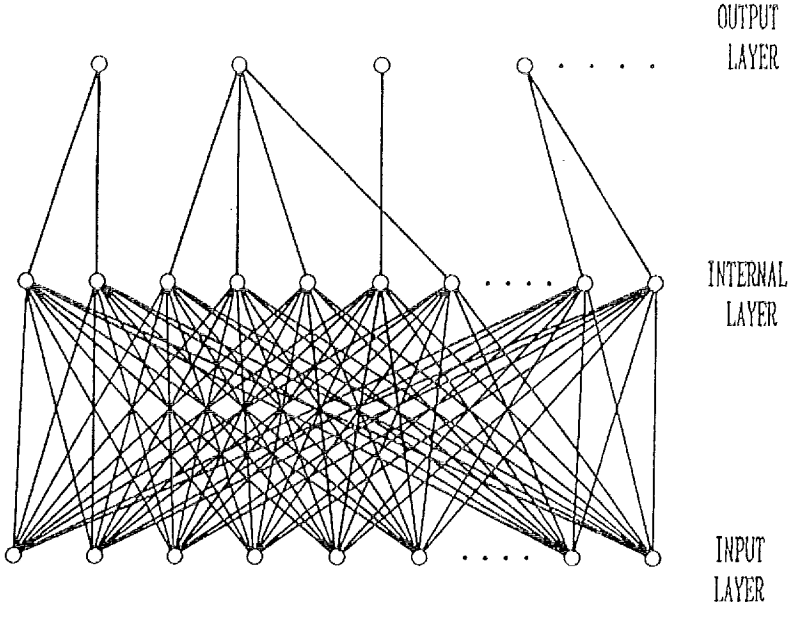


그림 1. AMDC의 구조

이 분류기는 입력 패턴이 어느 클래스에 속하는가를 결정하는 판단의 기준으로서 마할라노비스 거리를 사용한다.

내부 노드와 입력 노드 간 연결 값들을 대표 벡터 M이라 부른다. 입력 패턴 벡터와 출력노드에 의해 지정된 내부 노드들 사이의 마할라노비스 거리를 계산하여 최소거리가 vigilence parameter α 보다 작으면 식(1)에 의하여 대표 벡터 M을 계산한다.

$$M_i(t+1)=M_i(t)+\frac{1}{t+1}[X_i(t+1)-M_i(t)] \quad (1)$$

$$M_i(1)=X(1) \quad (2)$$

여기서 $M_i(t)$는 i번째 내부 노드로 학습된 학습 패턴 벡터가 t개 일때의 대표 벡터를 뜻한다. 따라서 $M_i(t+1)$은 이 노드에 포함되는 학습 패턴이 하나 더 들어왔을 때의 대표 벡터이다. $X_i(t)$은 i번째 내부 노드에 t+1번째로 포함된 입력 패턴 벡터이다.

AMDC는 마할라노비스 거리를 계산하기 위하여 다음 식과 같은 편차 벡터가 필요하다. 편차 벡터는

$$\sigma_i=[\sigma_{i1}\ \sigma_{i2}\ \sigma_{i3}\ \ldots\ \sigma_{in}]^T \quad (3)$$

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

$$\sigma_{ij}^2=\frac{1}{t+1}\sum_{r=1}^{t+1}x_j^2(r)-m_{ij}^2(t+1) \quad (4)$$

의 식을 이용한다. 이 식에서 $x_j(r)$은 i번째 내부 노드로 r번째 입력한 j축의 입력 값, 즉 j번 째 입력 노드의 입력 값이고 $m_{ij}(t+1)$은 i번째 내부 노드에서 (t+1)번째까지 입력된 학습 패턴들의 대표 벡터의 j번 째 원소이다. 내부 노드가 처음 발생했을 때 대표값은 입력 패턴과 같고 그 노드의 편차는 0 이다. 이 경우에 마할라노비스 거리를 계산할 수 없으므로 σ 대신에 d라는 벡터를 구해서 계산한다.

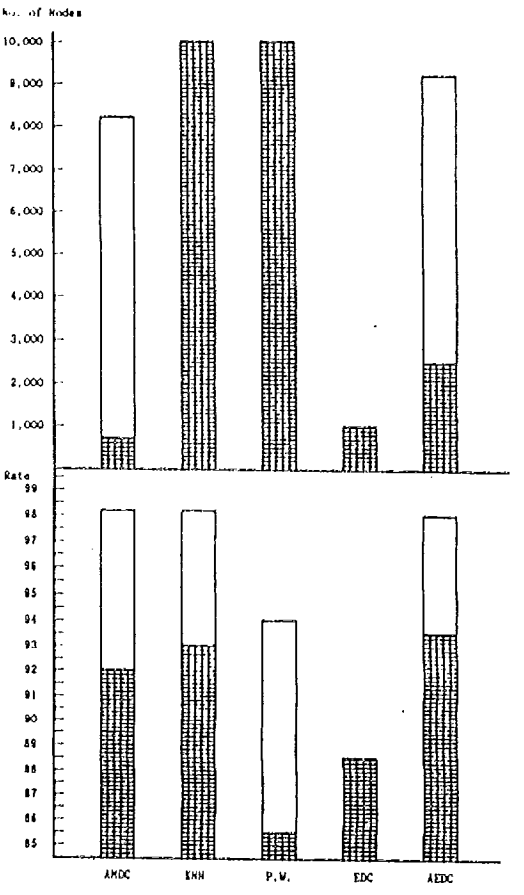


그림 2. 여러 분류기의 분류율과 내부노드 수의 비교

$$d_i=[d_{i1}\ d_{i2}\ d_{i3}\ \ldots\ d_{in}] \quad (5)$$

$$d_{ij}=\begin{cases}\theta & \text{if } \sigma_{ij}\le\theta\\ \sigma_{ij} & \text{if } \sigma_{ij}>\theta\end{cases} \quad (6)$$

이 식에서 θ 는 σ 의 초기값으로 생각하면 된다. 따라서 마할리노비스의 거리는

$$D_i^2=\sum_{j=i}^{n}\frac{(x_{ij}-m_{ij})^2}{d_{ij}^2} \quad (7)$$

이 된다.

이 AMDC를 500자 23종(총 11500자)의 한글 인식에 적용하여 다른 분류기와 비교하여 보면 그림(2)와 같다. 이 그래프에서 보면 분류율은 92~98.5%, 내부 노드 갯수는 800~8200개로 다른 분류기 보다 분류율이 우수하며, 발생되는 노드의 갯수도 작다는 것을 알 수 있다.

Ⅲ. PA-AEDC의 설계

복합 문자체, 복합 크기의 문자 인식에서 다양한 형태의 문자 패턴을 보다 잘 인식하려면 형태 변화를 잘 인식해야 하는데, 이를 위해서는 적응성을 갖고 있는 분류기로 인식하는 것이 효과적이다. AMDC는 패턴 변형을 효율적으로 흡수하여 적응성을 나타내는 분류기이다. 그러나 분류기의 layer가 두 개로서 입력 노드와 내부 노드 사이에서 임의의 입력 패턴에 가장 가까운 내부 노드를 선택하기 위해서는 모든 내부 노드의 대표 벡터 M과 D를 계산하여야 한다. 그러나 모든 내부 노드에 대하여 계산을 하지 않고 가능성이 있다고 인정되는 내부 노드만을 활성화 시킬 수 있다면 연산량은 대폭 감소될 수 있을 것이다. 본 논문에서는 전분류(pre-classification)의 개념으로 내부 노드를 부분적으로 활성화할 수 있는 PA-AEDC를 설계하고자 한다.

3.1 구조

PA-AEDC는 AMDC와 같은 형태의 신경회로망을 주신경망(primary net)으로하고 신경회로망을 선택적으로 활성화하여 주는 부신경망(secondary net)의 두 부분으로 구성되어있다. 주신경망이 AMDC와 다른 점은 내부 노드와 출력 노드와의 연결이 단일 방향이 아닌 양방향으로 구성되어 있어서 내부 노드에서 계산된 거리가 출력 노드로 전달되는 한편 부신경망에 의해서 선택된 출력 노드는 자신에게 소속된 출력 노드에 활성화 신호를 보내준다. 이 신호에 의해서 활성화된 내부 노드만이 연산에 참여하게 된다.

부신경망은 입력 벡터 값을 받아서 벡터의 차원 수를 축소 시키고 이를 선택/활성 노드로 전달한다. 벡터의 차원 수를 축소시키는 방법은 그림 4 에서 보는 바와 같이 특징 벡터를 추출할 때 receptive field를 8×4, 4×8로 하여 특징 벡터의 크기가 수평 방향 연산자에 32, 수직 방향에서 32, 모두 64개로 구성하였다면, receptive를 16×8, 8×16으로 키우면 벡터 크기는 16으로 줄어들 수 있다. mesh 또는 방향성 에지 연산자를 특징으로 사용하는 경우에는 인근 receptive field의 원소 값을 단순히 더함으로써 receptive를 키울 수 있고 따라서 벡터 크기를 줄일 수 있다. 축소 크기 벡터를 만드는데 인근의 값 만을 더하므로 입력 노드에서 부수 특징(secondarg feature)

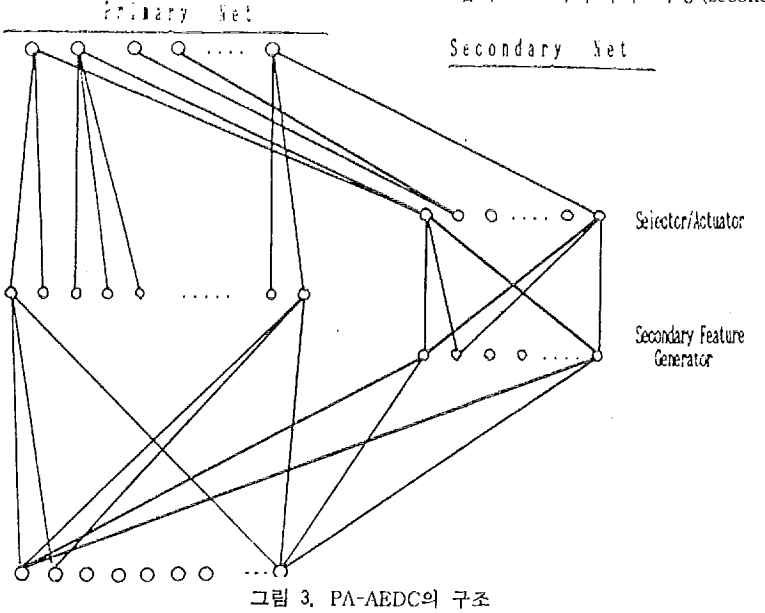
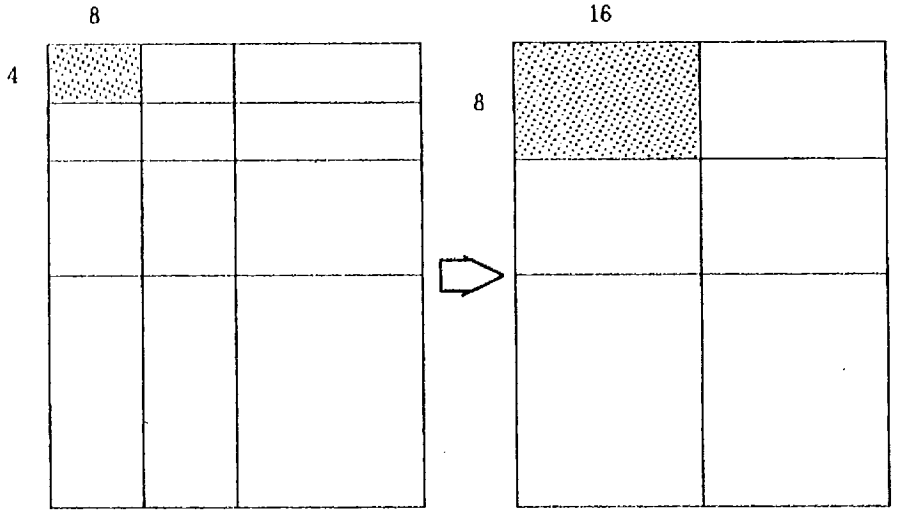
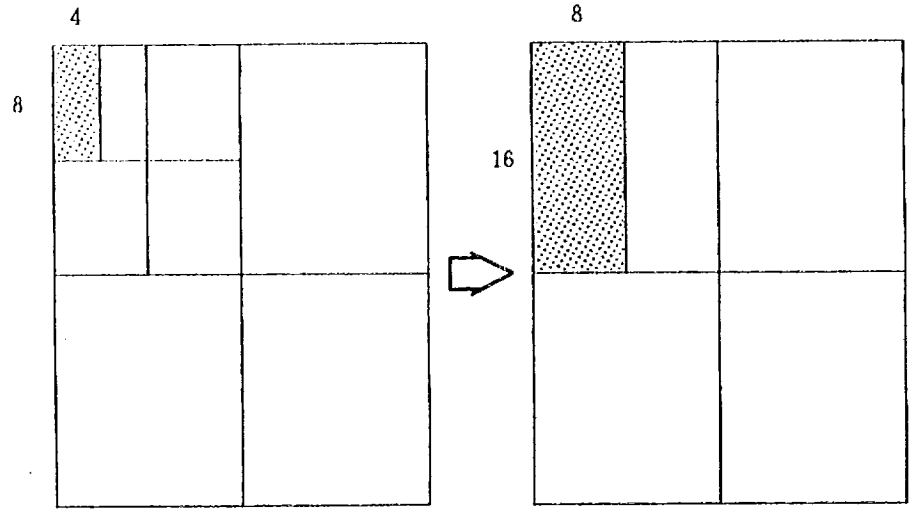


그림 3. PA-AEDC의 구조



(a) 수평



(b) 수직

그림 4. 특징 벡터의 축소

발생 노드로의 연결은 완전 연결일 필요가 없이 부분 연결만으로도 가능하다. 부신경망에서는 입력 패턴이 귀속될 가능성이 있는 출력 노드를 선택하는 역할을 하므로 굳이 특징 벡터를 모두 쓰지않고 축소된 벡터만을 사용하여도 동작에 무리가 없다. 선택/활성(select/actuating) 노드는 학습 중에 입력되는 패턴들로 계산된 대표 값을 갖고 있다가, 테스팅 중에 입력되는 패턴들과 거리를 계산하여 가장 작은 거리와 그 다음 작은 거리를 갖는 두 개의 선택/활성 노드가 자신에게 연결된 출력 노드들을 활성화 시킨다.

이와같이 두 개의 선택/활성 노드가 선택되는 이유는 분류시에 가능성이 있는 클래스를 넓히기

위함이다. 대개 하나의 선택/활성 노드에 여러 개의 주신경망 출력노드가 연결되므로 이때의 활성화 되는 출력 노드의 수는 여러 개가 된다. 가장 작은 거리와 두번째로 작은 거리를 갖는 선택/활성 노드를 선정하기 위해서는 Hopfield Net[5, 6, 7] 와 같이 동일 layer에 있는 다른 노드들의 출력 값이 입력되어야 한다

주신경망의 내부 노드가 갖는 대표 벡터 M은 AMDC의 대표 벡터, 즉 식(1), (2)와 같다. 본 논문에서 구성하는 PA-AEDC는 연산량을 줄이기 위하여 마할라노비스 거리 대신에 유클리디안 거리를 사용하므로 식 (3)-(6)과 같은 편차 계산이 필요 없다. 유클리안 거리는 다음 식과 같다.

$$D_i^2 = \sum_{j=1}^{n} (x_j - m_{ij})^2 \qquad (8)$$

여기서 x_j와 m_{ij}는 각각 입력 패턴과 대표 벡타의 j번째 원소이다.

부신경망에 의해 출력 노드를 통하여 활성화된 내부 노드는 식(8)과 같은 거리를 계산하여 출력 노드로 전달한다. 출력 노드는 자신에게 입력되는 값 중 가장 작은 값을 선택하여 출력시킨다. 출력 노드에 연결된 내부 노드가 하나 밖에 없다면 그 값을 그대로 출력시킨다. 출력된 값 중 작은 순서에 따라 1, 2, 3, 4 등의 후보를 결정하면 된다.

3.2 학습 및 테스팅

학습시에 입력 패턴을 넣고 해당 출력 노드를 지정하면 이 노드가 자신에게 연결된 부신경망의 선택/활성 노드들과 주신경망의 내부 노드 들을 활성화시켜서 주, 부신경망 양쪽에서 모두 학습이 된다.

학습 순서는 다음과 같다.

1. 입력 패턴 X_j가 입력되고 해당 출력 노드를 지정한다.
2. 부신경망의 부수 특징 발생 노드에서 축소 입력 벡터 Z_j가 생성된다.
3. 지정된 출력 노드는 자신에게 연결된 주신경망의 내부 노드와 부신경망의 선택/활성 노드를 활성화 시킨다.
4. 출력 노드에 의하여 활성화된 주신경망의 내부 노드들은 자신이 갖고는 대표 벡터 M_i와 입력 벡터 X_j와의 거리를 (8)식에 의하여 계산한 후, 출력 노드에 계산 값을 보낸다. 출력 노드는 이 계산 값을 받아서 가장 작은 값을 내보내는 노드의 거리가 주신경망의 vigilence parameter α_p 보다 작으면 그 노드를 활성화시키고 나머지는 비활성 상태로 한다. 이 때 내부 노드의 출력 값이 모두 α_p 보다 크면 새로운 내부 노드를 발생시켜서 자신과 연결한다.
5. 다시 출력 노드에 의해서 활성화된 내부 노드는 대표 벡터 M_i를 (1)식에의해서 수정하고 새로 발생된 노드는 대표 벡터 값을 (2)식처럼 입력 벡터와같개 놓는다.
6. 출력 노드에 의하여 활성화된 부신경망의 선택/활성 노드들은 축소 입력 벡터 Z_j와 자신의 대표 벡터 N_k와의 거리를 계산하여 출력 노드로 보낸다. 출력 노드는 이 계산 값을 받아서 가장 작은 값을 내보내는 노드의 거리가 부신경망의 vigilence parameter α_p 보다 작으면 그 노드를 활성화시키고 나머지는 비활성화 시킨다. 이때 선택/활성 노드들의 출력 값이 모두 α_p 보다 크면 새로운 선택/활성 노드를 발생시켜서 자신과 연결하고, 대표벡터 $N_k(1)$을 다음과 같이 잡는다.

$$N_k(1) = X_k(1) \qquad (9)$$

여기서 N_k는 k번째 선택/활성 노드의 대표 벡터 값이고 X_k는 그 노드에 처음으로 포함되는 입력 패턴이다.

7. 출력 노드에 의하여 다시 활성화된 부신경망의 선택/활성 노드는 Z_j를 이용하여 자신의 대표 벡터 N_k를 수정한다. 즉

$$N_k(t+1) = N_k(t) + \frac{1}{t+1}\{X_k(t+1) - N_k(t)\} \qquad (10)$$

의 식에 의하여 N_k를 다시 계산한다. 이 식들은 식 (1), (2)와 모양이 같다.

앞의 방법대로 학습을 하면 특징 공간 상의 학습 패턴 분포에 따라 내부 노드와 선택/활성 노드를 적절히 발생시켜 나갈 수 있는 적응성을 갖게 된다. Vigilence parameter α_p와 α_s를 작게하면 노

드 수가 많아지는 대신 인식률은 높아지며 반대로 크게하면 노드 수가 적어지며 인식률은 저하된다.

학습을 마치게되면 다른 패턴들을 넣어서 분류를 해보는 테스팅을 하게 되는데 그 과정은 다음과같다.

1. 입력 패턴 X_j를 입력시킨다.
2. 부신경망의 부수 특징 발생 노드에서 축소 입력 벡터 Z_j가 생성된다.
3. 선택/활성 노드에서 Z_j와 자신의 대표 벡터 N_k와의 거리를 계산한다. 계산된 거리를 다른 선택/활성 노드의 값과 비교하여 가장 작거나 두번째로 작으면 자신에게 연결된 출력 노드를 활성화 한다.
4. 활성화 신호를 받은 출력 노드들은 자신에게 연결된 내부 노드들을 활성화시킨다.
5. 활성화된 내부 노드들은 입력 패턴 벡터 X_j와 자신의 대표 벡터 M_i와의 거리를 계산하여 출력 노드로 전달한다.
6. 출력 노드는 자신에게 입력되는 값 중 가장 작은 것을 선택하여 자신의 출력 값으로 삼는다.
7. 출력 노드의 출력 값 중 작은 순서대로 후보 1, 2, 3, 4로 정한다. 이와같이 후보를 정하는 이유는 필요한 경우 후처리에서 최종 분류를 하기 위함이다.

Ⅳ. 실험 및 결과

본 논문에서 설계한 PA-AEDC의 성능을 파악해 보기 위하여 한글의 분류 실험을 하였다. 실험에 사용한 특징은 수평, 수직 에지 연산자로서 수평 방향 4×8, 수직 방향 8×4의 receptive field의 것이다.

먼저 한글을 200자에 대하여 실험을 하였다. 6종(1200자)의 한글을 학습시켰으며 12종(2400자)의 데이타로 테스팅을 하였다. 실험한 결과는 표 1과 같다.

이 실험에서 발생하는 노드 수는 앞의 것이 주신경망의 내부 노드 수이고 뒤의 것이 부신경망의 선택/활성 노드 수이다. 하나의 문자를 인식할 때에 연산에 참여하는 노드의 갯수는 평균적으로 식(10)과 같다.

표 1. 여러 α_P, α_S에 대한 분류율과 노드 수의 실험 결과

α_P	α_S	후보 갯수				노드 갯수	연산 참여 노드 수
		1	2	3	4		
100	50	93.0	96.0	99.0	99.0	988/140	154
	60	92.0	95.5	98.0	98.0	994/ 67	107
	70	93.0	97.0	99.0	99.0	1004/ 35	92
200	50	91.5	94.0	98.5	99.0	541/140	148
	60	89.5	93.5	98.5	98.5	533/ 67	83
	70	93.0	96.0	99.0	99.0	545/ 35	67
300	50	93.0	95.5	98.5	99.0	500/140	148
	60	90.5	94.0	98.5	98.5	478/ 67	81
	70	93.0	96.5	99.0	99.0	485/ 35	63

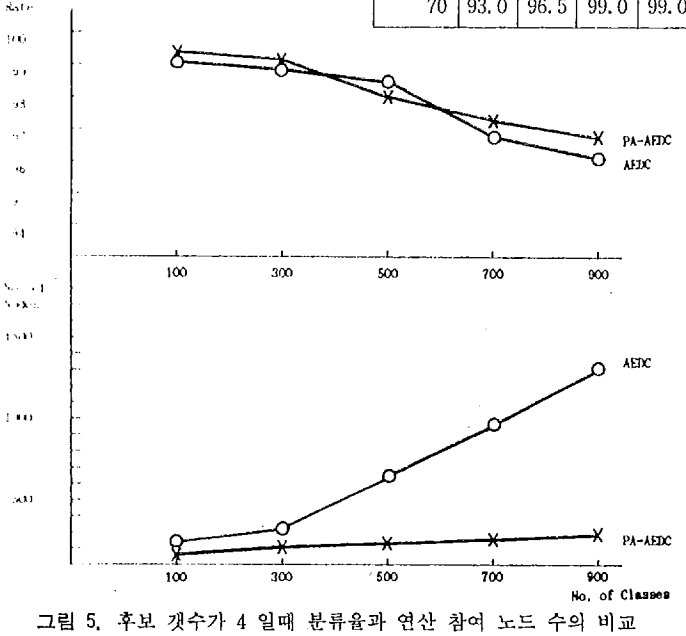


그림 5. 후보 갯수가 4 일때 분류율과 연산 참여 노드 수의 비교

연산참여노드수=선택/활성 노드수

$$+\frac{\text{주신경망 내부노드수}}{\text{선택/활성노드수}}\times 2 \qquad (10)$$

이 식에서 2를 곱한 이유는 두개의 선택/활성 노드가 선택되기 때문이다.

표 1 에서보면 AEDC를 그대로 사용하여 연산을 하는 경우에 연산에 참여하는 내부 노드 수보다 PA-AEDC의 연산에 참여하는 노드 수가 훨씬 작다는 것을 알 수 있다. 또한 선택/활성 노드의 연산은 차원이 작은 벡터를 갖고 행하여 지므로 연산량은 더욱 감소하게 된다.

다음에 분류 대상 문자 수를 100에서 900까지 증가시켜가면서 분류율과 연산 참여 노드 수의 변화를 AEDC와 비교하여 보았다.

이 실험에서 사용한 AEDC는 α_P = 250의 분류기이고 PA-AEDC도 α_P는 250으로 같고 α_S는 65로 실험하였다. 결과 그림에서보면 분류율은 서로 비슷하지만, 연산에 참여하는 노드 수는 PA-AEDC가 훨씬 작다는 것을 알 수 있다.

V. 결 론

본 논문에서 설계한 PA-AEDC는 같은 클래스에 소속되는 문자일지라도 문자의 모양이 판이하게 다른 경우에 적응적으로 서브클러스터를 만들어 주어 하나의 클래스로 묶어주는 AMDC의 기능을 그대로 갖고있다. 이와 같이 서브클러스터를 발생시키면 문자와 같이 여러 모양의 글자가 존재하는 패턴들을 분류할 때에 효과적으로 동작할 수 있다.

신경회로망은 또한 신속한 학습과 정확한 동작을 할 수 있도록 구성하는 것이 중요하다. Back-propagation 모델과 같이 학습에 너무 많은 시간이 소요되고 local minima[8, 9]에 빠지는 경향이 있으면 실용화가 거의 불가능하다. PA-AEDC는 신속한 학습이 가능하고 분류 동작에 있어서도 필요한 노드 만을 활성화하므로 AMDC나 AEDC보다 연산량이 대폭 감소되어 효율적으로 동작할 수 있다.

실험 결과에서 알 수 있듯이 PA-AEDC는 AEDC의 분류율과 거의 비슷하지만 분류할 클래스 수가 많을수록 활성화되는 노드 수의 차이가 더 벌어지는 것을 알 수 있다. 실제로 900자를 분류할 때 PA-AEDC의 활성화 노드 수는 AEDC의 1/6-1/7 수준이 된다. 그럼에도 불구하고 PA-AEDC는 AMDC나 AEDC가 갖고 있는 적응성, 자기조직적 기능 (self-organization)을 그대로 유지하므로 문자 인식 시스템을 구성하는 경우에 이 PA-AEDC를 사용하는 것이 더 유리할 것으로 판단된다.

참 고 문 헌

1. J.T. Too and R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley, 1974.
2. R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis, Wiley-Interscience, 1973.
3. I. Guyon, I. Poujaud, L. Personnaz, G. Dreyfus, "Comparing different neural network archtectures for classifying handwritten digits," in International Joint Conference on Neural Networks, pp. II-127-II-132, Washington D.C., 1989.
4. 최원호, 최동혁, 이병래, 박규태, "한글 인식을 위한 신경망 분류기의 응용," 대한전자공학회 논문지, 제27권 8호, pp. 93-103, 1990.
5. J.J. Hopfield, "Neural networks and physical system with emergent collective computational abilities," Proceedings of the National Academy of Sciences, U.S.A., vol. 79, pp. 2554-2558, April 1982.
6. J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," Proceedings of the National Academy of Sciences, U.S.A., vol. 81, pp. 3088-3092, May 1984.
7. Y.H. Pao, Adaptive Pattern Recognition and Neural Network, Addison-Wesley, 1988.
8. P. Baldi and K. Hornik, "Neural networks and principal component analysis : learning from examples without local minima," Neural Networks vol. 2, pp. 53-58, 1989.
9. J.L. Mclelland, D.E. Rumelhart, Parallel Distributed Processing, vol. I, II, MIT Press, 1986.