퍼지 및 지능시스템학회 논문지 2001, Vol. 11, No. 3, pp. 223-230

유전자 알고리즘을 사용한 구조적응 자기구성 지도의 최적화

Optimization of Structure-Adaptive Self-Organizing Map Using Genetic Algorithm

김현돈 · 조성배
Hyun-Don Kim and Sung-Bae Cho

연세대학교 컴퓨터과학과
Department of Computer Science, Yonsei University

요 약

자기구성 지도는 주어진 입력에 대해 올바른 출력 값이 제공되지 않는 비교사 방식으로 학습된다. 또한, 반응하는 순서나 위치를 통해 위상이 보존(topology preserving)되는 특성을 가지고 있어 많은 분야에 응용되고 있다. 그러나, 자기 구성지도는 학습이 되기 전에 위상을 미리 고정시켜야 하기 때문에 실제 문제에 적용하기 어렵다는 단점을 가지고 있다. 구조 적응형 자기구성 지도는 자기구성 지도의 고정된 구조 때문에 발생하는 문제를 해결하기 위해 지도의 구조를 학습 중에 적절하게 변경시킨다. 이때, 변화된 구조의 가중치를 어떻게 초기화시킬 것인가 하는 것이 또한 중요한 문제이다. 이 논문에서는 구조 적응형 자기구성 지도 모델에서 유전자 알고리즘을 이용하여 분화된 노드의 가중치를 결정하는 방법을 제안한다. 이 방법은 기존의 구조 적응형 자기구성 지도보다 다소 높은 인식률을 보였고, 숫자 별 인식률 편차를 줄일 수 있었다. 오프라인 필기 숫자 데이터로 실험한 결과, 제안한 방법이 유용함을 알 수 있었다.

Abstract

Since self-organizing map (SOM) preserves the topology of ordering in input spaces and trains itself by unsupervised algorithm, it is used in many areas. However, SOM has a shortcoming: structure cannot be easily determined without many trials-and-errors. Structure-adaptive self-organizing map (SASOM) which can adapt its structure as well as its weights overcome the shortcoming of self-organizing map: SASOM makes use of structure adaptation capability to place the nodes of prototype vectors into the pattern space accurately so as to make the decision boundaries as close to the class boundaries as possible. In this scheme, the initialization of weights of newly adapted nodes is important. This paper proposes a method which optimizes SASOM with genetic algorithm (GA) to determines the weight vector of newly split node. The learning algorithm is a hybrid of unsupervised learning method and supervised learning method using LVQ algorithm. This proposed method not only shows higher performance than SASOM in terms of recognition rate and variation, but also preserves the topological order of input patterns well. Experiments with 2D pattern space data and handwritten digit database show that the proposed method is promising.

Key Words : 구조 적응형 자기구성 지도, 자기구성 지도, 신경망, 유전자 알고리즘

1. 서 론

자기구성 지도는 주어진 입력에 대해 올바른 출력 값이 제공되지 않는 비교사 방식으로 학습된다. 또한, 반응하는 순서나 위치를 통해 위상이 보존(topology preserving)되는 특성을 가지고 있어 많은 분야에 응용되고 있다. 그러나, 자기구성지도는 학습이 되기 전에 위상을 미리 고정시켜야 하기 때문에 패턴 분류 등의 실제 문제에 적용하기 어렵다는 단점을 가지고 있다. 이러한 결점을 해결하기 위해 동적으로 구조를 변화시키고자하는 연구들이 수행되어 왔다[1]. 한편, 자기구성지도의 노드들의 초기 가중치값 설정은 자기구성지도의 성능에 직접적인 영향을 미치는데, 최적의 초기 조건을 자동으로 찾아주는 방법이 필요하다. 이를 위하여 유전자 알고리즘을 사용하였는데, 주어진 문제에 대한 최적의 해를 효과적으로 찾는 유전자 알고리즘의 특징 때문에 신경망을 최적화하는데 종종 응용되어왔다[2, 3, 4].

유전자 알고리즘이 신경망에 결합되는 방식에는 크게 두 가지 방법이 있다. 먼저, 염색체의 구조를 결정할 때, 가중치들을 염색체로 암호화하는 방법이 있고[4], 위상 구조를 염색체로 암호화하는 방법이 있다[2, 3]. 이 방법들 중에서 가중치를 염색체로 암호화하는 방법을 이용하여 노드를 효과적으로 초기화하고자 하였다. 본 논문에서는 동적으로 노드가 분화되어 구조를 스스로 변화시키는 구조 적응형 자기구성 지도 모델에 유전자 알고리즘을 적용하였다. 분화된 자식 노드들의 가중치 값을 염색체로 암호화하여 진화시켜서 구조적응 자기구성 지도의 가중치를 초기화하였다. 실험의 결과로 구조적응 자기구성 지도 분류기의 인식률이 향상되었음을 알 수 있었다.

접수일자 : 2001년 5월 1일
완료일자 : 2001년 5월 7일

본 논문의 구성은 다음과 같다. 2장에서는 자기구성 지도에 관해서 전반적으로 설명하고, 위상보존을 평가 기준 등에 대해 살펴본다. 3장에서는 본 논문에서 제안하는 구조 적응형 자기구성 지도, 유전자 알고리즘을 통한 신경망의 최적화를 살펴본다. 4장에서 실험을 통한 제안한 방법들의 유용성을 검증하였고 5장에서 결론 및 향후 연구에 대해 언급한다.

2. 관련연구

2.1 자기구성 신경망

지도 학습방법을 사용하는 신경망은 입력값과 그에 상응하는 출력값을 가지고 학습을 수행한다. 학습은 각 입력값에 대해 올바른 출력값이 나올 수 있도록 가중치를 변화시키는 것이다. 신경망은 생물학적 근거에 기반을 두고 있지만, 학습 도중에 올바른 출력값이 주어져야 한다는 것은 이러한 생물학적 의미에 맞지 않다. 이 같은 문제에 답이 될만한 것을 제안한 사람은 튜보 코호넨(Teuvo Kohonen)[5]이다. 그가 제안한 자기 구성 지도는 어떻게 신경망이 자기 스스로 그 구조를 생성해 낼 수 있는 지에 대한 답을 비교적 간단하게 제시해 주고 있다. 자기 구성이라는 말은 신경망이 주어진 입력에 대해 올바른 출력값이 제공되지 않고도 학습됨을 의미한다. 뿐만 아니라, 반응하는 순서나 위치를 통해 위상을 보존(topology preserving)한다는 특성을 가지고 있다.

자기구성 지도는 그림 1에서 보는 바와 같이 입력층과 출력층, 두 개의 층으로 구성되어 있다. 출력층은 N개의 노드로 구성되며 각 노드는 입력 벡터와 같은 차원을 가지는 가중치 벡터로 구성되어있다. 신경망의 각 노드는 가중치 벡터에 임의의 실수를 할당함으로써 초기화된다. 이 후, 신경망에 하나의 입력 벡터를 가하고, 신경망의 노드들로부터의 거리 값을 구하고, 이로부터 가중치 벡터를 갱신하는 세 단계로 학습이 이루어진다.

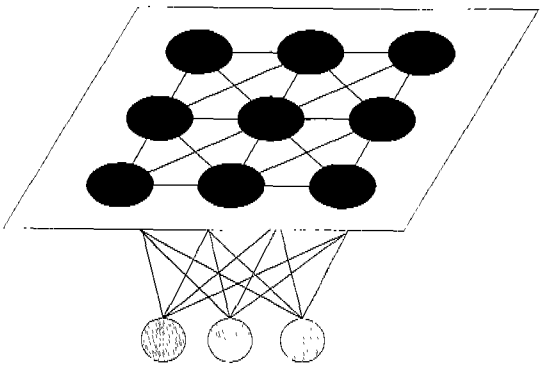

그림 1. 자기구성 지도의 구조
Fig 1. Overview of self-organizing map

입력 벡터가 신경망에 들어오면, 입력 벡터와 모든 노드들과의 유클리드 거리를 계산한다. 최소 거리를 갖는 노드가 승자로 선택되고 수식에서처럼 mc로 나타낼 수 있다.

$$\|x-m_c\| = \max_i \{\|x-m_i\|\}$$

그 다음에 승리자 노드를 중심으로 이웃 노드들과 승리자 노드에 대한 가중치 벡터 갱신이 일어난다. 그 과정은 다음의 수식과 같다.

$$m_i(t+1) = m_i(t) + \alpha(t) \times n_{ci}(t) \times \{x(t) - m_i(t)\}$$

여기서 $\alpha(t)$는 학습률을 나타내는 함수, nci(t)는 이웃 함수, mi(t)는 노드의 가중치, x(t)는 입력 벡터 값이다[5, 6]. nci(t)에서 c는 승리자 노드의 인덱스이다. 일반적으로 학습률과 이웃 함수는 학습이 반복되면서 감소되어야 한다.

2.2 위상보존 평가척도

자기구성 지도의 가장 대표적인 특징은 위상보존이다. 위상보존이란 입력 벡터의 위상 구조를 출력층의 노드들이 유클리드 거리에 따라 반영함을 의미한다. 즉, 유사한 벡터들이 근접하게 위치하여 위상학적 구조가 반영되는 것이다. 이러한 위상보존의 척도에 관한 연구는 다음과 같다.

Topographic 오류[7]는 입력 벡터 스페이스의 연속성을 계산하는 기준이다. 입력 벡터 x에 대해서 x의 가장 가까운 출력층의 가중치 벡터를 wi라고 하고, 두 번째로 가까운 가중치 벡터를 wj라고 하자. 이 때, 입력 벡터 스페이스의 모든 벡터들이 wi 혹은 wj에 매핑될 때, wi와 wj에 해당하는 노드들이 근접하면 벡터 x에 대해서 위상이 연속하고, 그렇지 않은 경우 위상은 불연속하게 되어 오류가 발생하게 된다. 수식으로 표현하면 다음과 같다.

$$\varepsilon_t = \frac{1}{N} \sum_{k=1}^{N} u(x_k)$$

여기서, u(xk)값은 위상이 연속할 경우 1이 되고 연속하지 않을 경우 0이 된다.

Topographic 함수[8]는 근접한 수용 영역을 가진 출력층 노드들 중에서 실제 두 노드 사이의 거리가 특정 s 값 이상되는 경우를 오류로 설정하고 그 오류 값을 누적시킨 값이다. 여기서 수용 영역 Ri 는 Vi ⋂M 과 같다. 여기서 M은 입력 스페이스를 나타내고 Vi는 Voronoi 다면체를 말한다. Topographic 함수는 다음과 같다.

$$\Phi_L^M(s) = \sum_{i=1} \#\{n_j \mid j \in L, \|n_i - n_j\| > s,\ n_i \text{와 } n_j \text{는서로 이웃함}\}$$

Koenig의 위상보존 평가 함수[9]는 n개의 최근접 이웃(nearest neighbor), NNji (i ∈[1,n], j ∈[1,N])를 이용한다. 여기서 N은 노드의 수를 말한다. 입력 스페이스 X와 출력 스페이스 Y에 대하여 입력 벡터 vj에 대한 위상보존은 다음과 같이 평가된다.

$$\begin{cases} 3, & \text{if } NN_{ji} \text{ on } X = NN_{jt} \text{ on } Y \\ 2, & \text{if } NN_{ji} \text{ on } X = NN_{jl} \text{ on } Y \quad l \in [1,n] \quad i \neq l \\ 1, & \text{if } NN_{ji} \text{ on } X = NN_{jt} \text{ on } Y \quad t \in [n,k] \quad n < k \\ 0, & \text{else.} \end{cases}$$

이러한 방식으로 입력 벡터 vj에 대한 평가값 qmj를 구하여 합한 qm이 위상보존의 정도를 나타내게 된다. qm은 다음과 같다.

$$q_m = \frac{1}{3n \times N} \sum_{j=1}^{N} q_{mj}$$

완벽한 위상보존이 이루어 졌을 때, qm 값은 1.0이 된다. 그리고 일반적으로 n은 4로 k는 10으로 둔다. 본 논문에서는 이 척도를 통해서 위상보존 정도를 평가하였다.

3. 구조 적응형 자기구성 지도

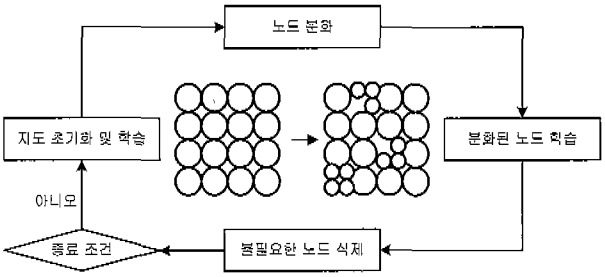

그림 2. 구조 적응형 자기구성 지도
Fig 2. Structure adaptive self-organizing map

이 장에서는 자기구성 지도의 성능 향상 방법인 구조 적응형 자기구성 지도에 대해서 살펴본다. 구조도는 그림 2와 같고 기본 알고리즘은 표 1과 같다.

표 1. SASOM의 기본 알고리즘
Table 1. Algorithm of SASOM

① 지도를 4×4 크기로 초기화한다.
② SOM 알고리즘으로 학습시킨다.
③ 지도의 노드들 중 여러 클래스의 데이터가 섞인 노드를 찾는다.
④ 찾아낸 노드들을 2×2 크기의 노드로 분화시킨다.
⑤ 분화된 노드들을 LVQ 알고리즘으로 학습시킨다.
⑥ 분화된 노드들 중, 학습에 참여하지 않는 노드를 삭제한다.
⑦ ③~⑥의 과정을 종료 조건이 만족될 때까지 반복한다.

이 알고리즘은 각각 지도 초기화 및 초기 학습 단계, 노드 분화 단계, 분화된 노드 학습 단계의 세 부분으로 나뉘어진다[1]. 여기서 두 가지 학습이 필요한데, 첫 번째는 일반적인 SOM 알고리즘을 이용하여 학습하는 것이고, 두 번째는 교사 학습 방법을 혼합한 LVQ 방식의 학습이다.

3.1 지도 초기화 및 초기학습

이 단계에서는 지도의 크기를 임의로 설정하여 초기화하고 코호넨 알고리즘에 의해서 학습시킨다. 지도는 4×4의 크기로부터 시작하여 2.1 절에 나온 수식에 의해 학습된다.

3.2 노드 분화

이 단계는 초기화된 지도를 토대로 분화되어야 할 노드를 찾아내는 역할을 한다. 분화되어야 할 노드를 찾아내는 방법은 다음과 같다. 먼저 지도의 모든 노드들에 대해서 최상의 매칭 클래스를 구한다. 초기에는 대부분, 하나 이상의 최상의 매칭 클래스를 가지는 노드가 발생할 것이다. 노드가 하나의 클래스에 대해서 반응하는 것이 아니라 다수의 클래스에 대해서 반응하게 되면 잘못된 결과를 산출하게 된다. 그러므로 이러한 노드들을 찾아서 분화시킨다.

이 모델에서, 분화되어야 할 노드를 찾아내기 위해서 hit ratio 값을 이용하는데, 이것은 i번째 노드에서 빈도수가 가장 높게 매칭되는 클래스의 빈도수를 i번째 노드에 매칭되는 클래스들의 빈도수 합으로 나눈 것이다.

여기에서는, hit ratio가 100% 미만을 나타내는 노드들을 찾아서 분화시키는 방식을 사용하였다. 분화되어야 할 노드들은 2×2 의 노드로 분화시킨다. 이때, 분화된 하위 노드들의 가중치는 분화되기 전 부모 노드의 가중치 값을 기반으로 이웃한 노드들의 가중치 값들을 고려하여 다음의 식과 같이 산출된다.

$$C=\frac{(P\times 2)+\sum N_c}{S}$$

여기서, C는 자식 노드의 가중치, P는 부모 노드의 가중치, Nc는 자식 노드의 이웃 노드의 가중치, S는 (Nc의 개수 +2)를 나타낸다. 즉, C는 이웃노드와 부모 노드의 평균값으로 결정된다.

- **경우 1** : 가장 일반적인 경우인데, 그림 3에서 P4 노드가 분화 될 경우를 예로 들면,

$$C_i=\frac{(P_i\times 2)+P_i+P_{i+1}}{4} \qquad (\text{여기서 } , i=i \bmod 4)$$

이 된다.

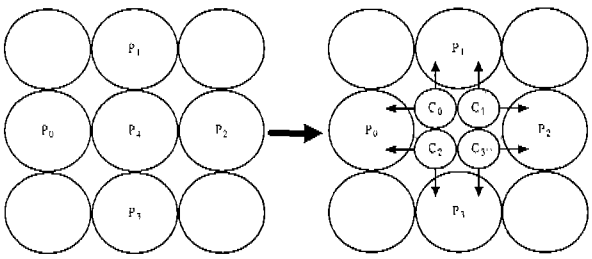

그림 3. 분화된 노드의 가중치 결정 (경우 1)
Fig 3. Determination of the weight of split node (case 1)

- **경우 2** : 이웃 노드의 수가 경우 2보다 작은 경우이다. 그림 4에서 자식 노드 C1의 경우, 오른쪽에 노드가 없으므로, 이웃 노드로 고려해야 할 노드는 P1밖에 없다. 그러므로 C1의 값은 {(P4×2)+P1}/3이 된다. 그리고 C3 노드의 경우, 이웃 노드로 고려해야 할 노드가 없으므로, C3의 값은 {(P4×2)}/2가 된다. 즉, 부모 노드의 가중치와 같은 값을 가지게 된다.

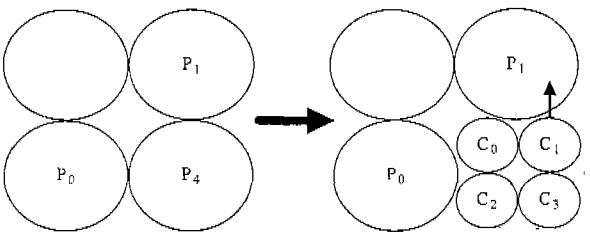

그림 4. 분화된 노드의 가중치 결정 (경우 2)
Fig 4. Determination of the weight of split node (case 2)

- **경우 3** : 이웃 노드의 수가 경우 2보다 많은 경우이다. 그림 5의 노드 P4의 경우를 살펴보자. 이 경우 자식 노드 C1은 위쪽 방향으로 1개의 이웃 노드를 가지고, 오른 쪽 방향으로 3개의 이웃 노드를 가진다. 이때, 노드 C1의 가중치는 부모 노드의 가중치와

두 배에 모든 이웃 노드의 가중치의 합을 합하여 평균을 구한 값이 된다. 이렇듯, 이웃 노드의 분화된 정도에 따라서 고려해야할 이웃 노드의 숫자는 달라지게 되는 데, 정확한 초기화를 위해서 존재하는 모든 이웃 노드들을 찾아내어야 한다.

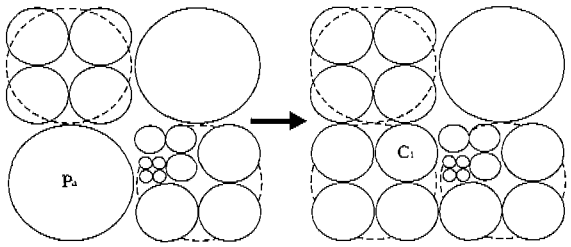

그림 5. 분화된 노드의 가중치 결정 (경우 3)
Fig 5. Determination of the weight of split node (case 3)

3.3 분화된 노드 학습

일단 분화가 일어나면 그 노드들에 대해서 학습을 시켜야 한다. 이 단계에서의 학습은 순수한 코호넨 알고리즘에 LVQ 알고리즘을 결합한 형태의 교사 학습 형태를 가진다. 학습식은 다음과 같다.

$$m_i(t+1) = m_i(t) + \alpha(t) \times n_{ci}(t) \times h_{ci}(t) \times \{x(t) - m_i(t)\}$$

여기서 hci(t)는 다음과 같이 계산된다.

$$\begin{cases} h_{ci}(t) = 1, & \text{if } x(t)\text{가 이전 단계 학습에서} \\ & m_i(t)\text{나 } m_j(t)\text{의 부모 노드에 할당된 경우} \\ h_{ci}(t) = 0, & \text{if } x(t)\text{가 이전 단계 학습에서} \\ & m_i(t)\text{나 } m_j(t)\text{의 부모 노드에 할당 안된 경우} \end{cases}$$

원래 LVQ 알고리즘은 최상 매칭 클래스에 대해서는 가중치를 증가시키고, 그 이외에는 가중치 값을 오히려 감소시켜서(즉, hci(t) = -1), 학습이 빠르게 진행 되도록 하였다. 그러나, 이 경우 오히려 최적한 해를 나타내지 못하는 경우가 발생할 수 있기 때문에 위에서 보는 것과 같이 페배 노드들에 대해서는 학습을 시키지 않았다. 또한 기존의 LVQ 알고리즘에서는 이웃 함수(nci(t))가 없이 전체의 지도에 대해서 학습을 시킨다. 그러나, 본 논문에서는 이웃 함수를 사용하여 가까이 있는 최상 매칭 클래스들에 대해서 학습을 시켰는데, 이것은 자기구성 지도의 특성인 위상보존이 손상되는 것을 방지하기 위해서이다. 또한, 과도한 노드 분화를 막고 학습 시간을 향상시키기 위하여 노드가 상한 임계치 이상의 깊이 만큼은 분화되지 않도록 설정하였다.

일단 위의 학습 알고리즘으로 학습되면, 불필요한 노드를 삭제한다. 삭제 평가 역시 기존의 hit ratio를 이용한다. 분화와는 달리 노드 삭제 경우에는 불필요한 노드를 찾아야 하므로 hit ratio가 0, 즉 어떤 데이터에 대해서도 승리자로 할당되지 못한 노드를 삭제하게 된다. 그러나, 무조건 hit ratio가 0인 노드를 삭제하게 되면, 다음과 같은 번거로움이 발생할 수 있다. 다음 단계 학습에서 삭제된 노드를 필요로 하게 되면, 그 노드가 존재하지 않으므로 다시 주변 노드를 분화하여 그 노드와 비슷한 가중치를 가지도록 학습시킨다. 이 경우, 하나의 노드가 필요해서 다시 3개의 노드를 더 삽입한 결과가 된다. 이러한 잘못된 삭제를 피하기 위해서 각 노드마다 age를 두었다. age는 hit ratio가 0이어도 삭제되지 않는 횟수를 말한다. 즉, age가 3일 경우, 3번까지는 hit ratio가 0이어도 그 노드를 삭제하지 않는다.

4. 유전자 알고리즘을 이용한 자식 노드 가중치의 초기화

이 장에서는 유전자 알고리즘을 이용한 분화된 자식 노드의 가중치를 초기화하는 방법에 대해서 살펴본다. 전체 구조도는 그림 6과 같다.

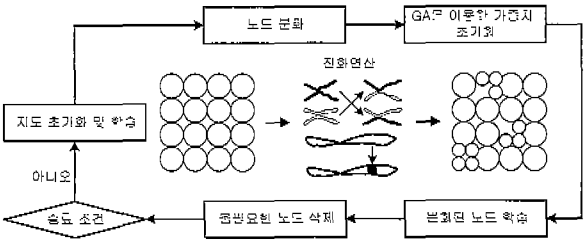

그림 6. 유전자 알고리즘을 이용한 신경망 최적화
Fig 6. Optimization of neural network using genetic algorithm

4.1 유전자 알고리즘을 이용한 분화된 자식 노드의 초기화

표 2. GASASOM의 기본 알고리즘
Table 2. Algorithm of GASASOM

① 지도를 4×4 크기로 초기화한다.
② SOM 알고리즘으로 학습시킨다.
③ 지도의 노드들 중 여러 클래스의 데이터가 섞인 노드를 찾는다.
④ 찾아낸 노드들을 2×2 크기의 노드로 분화시킨다.
⑤ 분화된 노드들의 초기 가중치 값을 유전자 알고리즘으로 구한다.
⑥ 분화된 노드들을 LVQ 알고리즘으로 학습시킨다.
⑦ 분화된 노드들 중, 학습에 참여하지 않는 노드를 삭제한다.
⑧ ③~⑦의 과정을 종료 조건이 만족할 때까지 반복한다.

이 방법은 기존의 구조 적응형 자기구성 지도의 노드 분화 단계와 분화된 노드 학습 단계사이에 분화된 노드의 진화 단계를 추가하여 네 부분으로 이루어진다. 학습 방법은 구조 적응형 자기구성 지도와 동일하다. 알고리즘의 내용은 표2와 같다.

일단 분화가 일어나면 유전자 알고리즘을 통하여 진화가 일어나게 된다. 진화 정도는 적합도 평가 함수에 임계치를 따로 주어 않았고, 특정 세대 만큼 항상 진화가 수행 되도록 하였다. 진화는 각각 분화된 노드들의 자식들만이 참여하여 이루어진다.

4.2 염색체 구조

염색체는 2×2로 분화된 자식 노드들의 가중치 값으로 인코딩 된다. 인코딩 에는 그림 7과 같다. 자식 노드들의 가중치 값은 n 차원을 가지는 실수들의 값이다. 그러므로, 염색체의 각 세그먼트는 자식 노드 가중치의 실수 세그먼트 값을

가지게 된다. 그러므로, n 차원인 자식 노드의 가중치에 대해서, 염색체의 크기는 $4\times n$ 의 실수가 된다.

각각의 염색체에 대해서 초기화 과정이 필요한데, 초기화는 일반적으로 수행하는 것처럼 임의로 모든 집단의 염색체를 초기화하지 않았다. 자기구성 지도의 특징인 위상보존을 파괴하지 않게 하기 위해서, 먼저 하나의 염색체를 선택하여 앞 단계의 노드 분화 단계에서 초기화된 4개의 가중치 값들을 인코딩 하였다. 즉, 부모 노드와 부모 노드들의 이웃 노드의 가중치 값이 반영된 값을 개체로 초기화한 것이다. 그리고, 4개의 가중치 값의 각 세그먼트의 최대 값과 최소 값을 구하여, 나머지 집단들의 염색체를 초기화하는 데 이용하였다. 각 세그먼트의 값이 세그먼트 단위의 최소 값과 최대 값 사이의 값을 가지도록 임의로 값을 초기화하였다. 그래서, 첫 번째 염색체와 값의 변화를 줄여 빠른 진화가 이루어지도록 하였다.

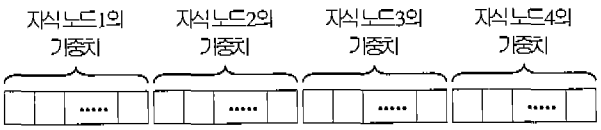

그림 7. 염색체의 구조
Fig 7. Structure of Chromosome

4.3 유전자 연산자

유전자 연산자는 일반적인 유전자 알고리즘에서처럼 선택 연산자, 교차 연산자, 돌연변이 연산자를 수행한다. 먼저 선택 연산자의 경우, 적합도 정도가 높은 개체들을 계속해서 살려나가는 형태의 선택을 수행하였다. 즉, 집단들 중에서 상위 적합도 값을 가지는 절반의 개체들을 우선적으로 선택을 하고, 나머지 개체들은 선택된 상위 개체들의 값을 그대로 복사하는 형태를 취하였다.

그리고 교착 연산자의 경우, 일점 교차을 사용하였고, 교차율은 0.3으로 하였다. 돌연변이 연산자의 경우, 돌연변이율이 0.01이 되도록 하였다. 돌연변이가 발생할 경우, 발생한 세그먼트 값을 초기화 단계에서 구한 최소 값과 최대 값 사이의 임의의 값으로 할당하였다.

4.4 적합도 평가

적합도 평가는 양자화 오류 함수(quantization error) 값을 이용하였다. 즉, 부모 노드에 할당된 데이터 벡터들을 따로 모아서, 염색체의 4개의 가중치와의 유클리드 거리를 구하여 합을 구하였다. 오류 함수 값이 클수록 적합도는 낮게 된다. 수식으로 나타내면 아래와 같다.

$$Error = \sum (m_c - v_i)^2$$

여기서 mc 는 염색체의 가중치를 나타내고 vi는 노드에 할당된 입력 벡터를 나타낸다.

5. 실험 결과

5.1 구조 적응형 자기구성 지도

실험은 데이터에 따라 두 부분으로 나누어진다. 첫 번째, 실험은 선형 일차 함수와 비선형 일차 함수로 나누어지는 2차원 패턴 공간 데이터에 대한 실험이고 두 번째, 실험은 구조 적응형 자기구성 지도 실험에서 사용되었던 오프라인 필기 숫자 데이터이다.

(1) 2차원 패턴 공간 데이터에 대한 실험

이 실험을 통해 구조 적응형 자기구성 지도가 얼마나 문제에 따라 입력 공간을 잘 반영하는 지를 살펴보고자 하였다. 실험의 데이터는 다음의 두 식에 의해서 얻을 수 있다.

$$\begin{cases} if\ (2y+4x-3)>0\ then\ \ class\ 0 \\ if\ (2y+4x-3)<0\ then\ \ class\ 1 \end{cases} \quad (1)$$

$$\begin{cases} if\ (x-0.75)(y-0.5)>0\ then\ \ class\ 0 \\ if\ (x-0.75)(y-0.5)<0\ then\ \ class\ 1 \end{cases} \quad (2)$$

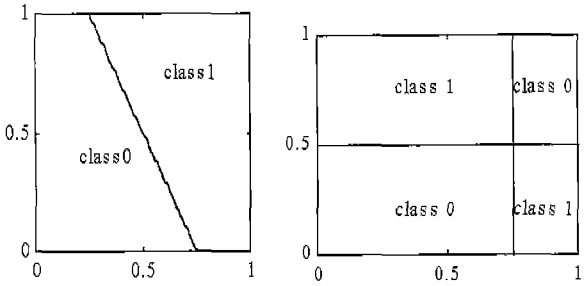

그림 8. 수식(1), (2)의 평면도
Fig 8. Graph of equation(1), (2)

표 3. 수식(1)의 실험 결과
Table 3. Experimental result on equation (1)

		SOM	SASOM
인식률	학습 데이터	99 %	100 %
	테스터 데이터	97 %	99 %
노드의 수		100	30

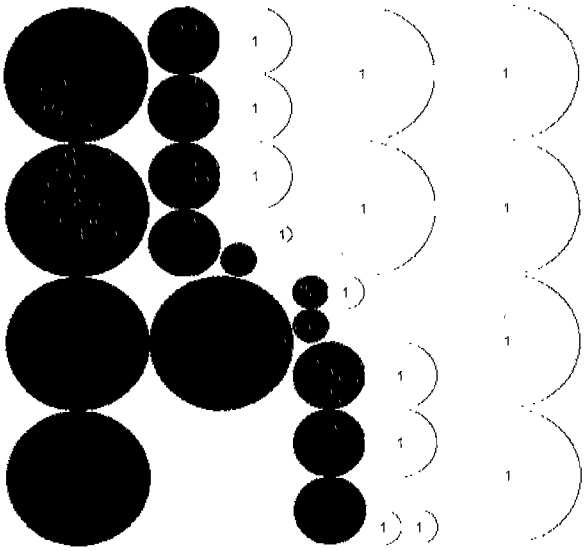

그림 9. 함수 (1)에 대한 SASOM의 지도
Fig 9. Map structure of SASOM on equation (2)

수식 (1)의 경우 선형 일차 함수로 x, y 모두 [0, 1]인 실수 공간을 하나의 경계로 두 부분으로 나눈다. 그리고 수식 (2)의 경우 실수 공간을 두 개의 경계로 4 부분으로 나눈다. 이 실험을 통해서 구조 적응형 자기구성 지도가 얼마나 경계에서 잘 분화되는 지를 보이고자 하였다. 두 식에 대한 실험 결과는 각각 표 4와 표 5에 나타나 있다. 구조적응형 자기구성 지도(SASOM)가 SOM 보다 인식률 뿐만 아니라, 노드 수에서 효율적임을 알 수 있다. 또한 그림 9와 그림 10은 SASOM이 결정 경계(Decision Boundary)부분에서 분화를 통해 더 많은 노드를 가지고 이 노드들을 통해 세밀하게 인식함을 보여 주고 있다.

표 4. 수식(2)의 실험 결과
Table 4. Experimental result on equation (2)

		SOM	SASOM
인식률	학습 데이터	99 %	100 %
	테스터 데이터	90 %	95 %
노드의 수		100	39

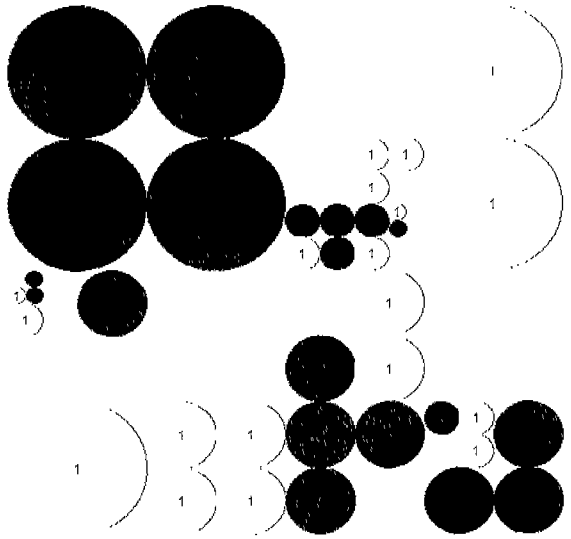

그림 10. 함수 (2)에 대한 SASOM의 지도
Fig 10. Map structure of SASOM on equation (2)

(2) 필기 숫자 데이터에 대한 실험

실험은 Concordia대학의 필기숫자 데이터 중 A(2000개)와 C(2000개)에서 Gradiant특징을 추출하여 사용하였다. 학습 횟수는 100000번, 학습률은 0.02로 하여 코호넨 알고리즘과 구조 적응형 자기구성 지도에 대한 실험을 수행하였다. 실험 결과는 그림 11과 같이 나타났다. 노드 수는 기존 코호넨 신경망이 900(30×30)인 반면에, 제안한 신경망은 710개다.

데이터에 대해서 Koenig의 함수를 이용해서 위상보존을 측정하였다. Koenig는 입력 벡터로부터 출력 공간에서 n개의 승리자를 구하고, 출력 노드로부터 일정한 거리 안에서 역시 n개의 승리자를 구하여 각각의 n개의 승리자의 값을 비교하여 위상보존 정도를 평가하였다. 2.2절에 수식으로 표현되어있다. 사각형 구조를 이용하였기 때문에 n(최근접 이웃의 수)은 4로 하였고, k 는 10으로 하였다. 위상보존 정도는 1에 가까울수록 완벽함을 의미한다. 결과는 표 5와 같다. 표에서 보듯이 4개의 최근접 이웃에 대한 위상보존 정도는 구조 적응형 자기구성 지도가 월등히 높음을 알 수 있다.

표 5. 위상보존 평가
Table 5. Evaluation of topology preservation

	SOM	SASOM
$\sum q_m$	8801	16600
위상보존 정도	0.3667	0.692

그림12는 반응 클래스와 그 빈도수에 따라서 노드가 어떻게 분화 되었는 지를 보여 주고 있다. 그림에서 원은 노드를 나타내고, 원 안의 굵은 글씨의 숫자는 반응 클래스를 나타내고, 괄호 안의 숫자는 빈도수를 나타낸다. 첫 번째 노드는 많은 클래스에 대해 반응하였으나, 분화가 진행될 수록, 하나나 두개의 클래스에 반응하였다.

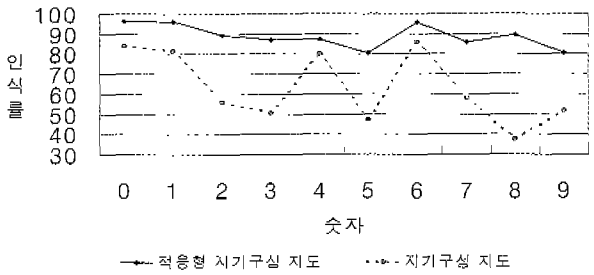

그림 11. 숫자 데이터에 대한 인식률 비교
Fig 11. Comparison of recognition rate of SASOM and SOM

표 6. SASOM 실험 결과
Table 6. Experimental results on handwritten digits

숫자	인식률 (%)	
	SOM	SASOM
0	84	98.5
1	81.5	100
2	56	93.5
3	50.5	92.5
4	80	93.5
5	47.5	85
6	86	96.5
7	58	95
8	37.5	90
9	51.5	96
종합	63.25	94.05

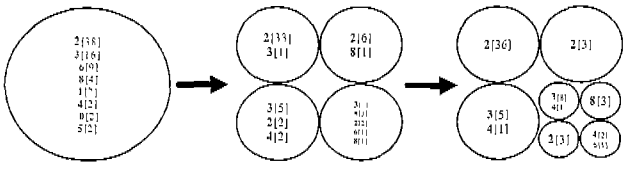

그림 12. 실제 분화의 예
Fig 12. Example of node splitting

5.2 유전자 알고리즘을 이용한 분화된 자식 노드의 초기화

데이터와 환경은 모두 4.1 절의 실험 (2)와 동일하다. 유전자 알고리즘에서는 집단의 크기를 20으로 하였고, 최대 세대는 1000으로 고정하였다. 실험은 기존의 구조 적응형 자기구성 지도(SASOM)와 유전자 알고리즘과 결합된 구조 적응형 자기구성 지도(GASASOM)에 대해 수행하였다. 각각의 실험 결과는 표 7과 같다.

먼저 신경망의 전체 노드 수를 비교해 볼 경우, 각각 710개와 694개를 나타내었다. 이것은 SASOM보다 GASASOM이 효율적으로 노드를 분화하고 삭제했음을 의미한다. 즉, 유용하지 않은 노드를 계속 분화해서 분화된 노드들이 실제 적으로는 사용되지 않았음을 알 수 있다. 그만큼 SASOM의 학습 정도가 낮다는 것을 반영한다.

표 7. GASASOM의 실험 결과
Table 7. Experimental results of GASASOM

	SASOM	GASASOM
실험 환경	최대 세대 : 1000 집단 크기 : 20	학습 회수 : 100000 학습률 : 0.02
인식률	94.05 %	94.95%
분화된 노드의 수	710	694

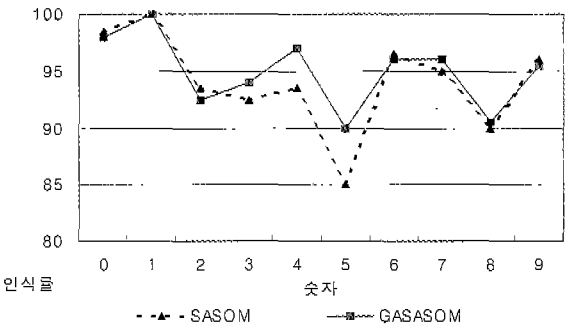

그림 13. GASASOM의 인식률 비교
Fig 13. Comparison of recognition rate of SASOM and GASASOM

인식률에 대한 실험 결과는 그림 13과 같이 나타났다. 인식률은 GASASOM이 94.95%를, SASOM이 94.05%를 나타냈다. 전체적인 인식률만 본다면 거의 비슷한 인식률을 보였는데, 그림 13에서 보듯이 GASASOM이 SASOM보다 편차가 작은 인식률을 보이고 있다. 즉, 특정 숫자에 대해서 편중되어 학습되지 않았다. 특히 숫자 5의 경우, SASOM에서는 유일하게 잘 인식되지 않았으나, GASASO에서는 다른 숫자만큼 인식되고 있음을 보이고 있다.

그림 14는 하나의 노드에 대한 적합도 평가 함수의 값의 변화를 나타낸다. 초기의 0세대에서의 적합도 평가 함수의 값은 알고리즘의 두 번째 단계에서 초기화된 가중치에 대한 값이고, 999세대에서의 값이 최종적으로 학습에 참여한 가중치 값을 나타낸다.

그림 15는 GASASOM 방법의 위상보존을 나타내는 그림이다. 그림에서 보듯이, 임의로 집단의 가중치 값을 할당하였어도, 단계 2에서의 가중치 값을 기본으로 하였기 때문에 위상보존 역시 잘 이루어졌다.

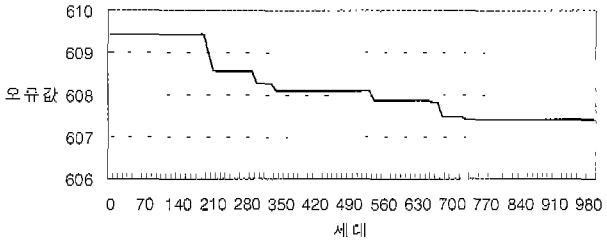

그림 14. 양자화 오류 함수 값의 변화
Fig 14. Change of Quantization error value

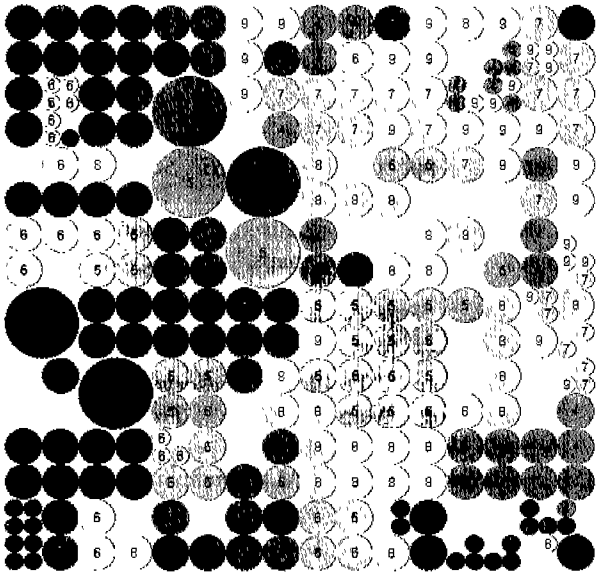

그림 15. 학습중의 위상
Fig 15. Topology of GASAOM map during training

6. 결 론

본 논문에서는 자기구성 지도의 성능 향상 방법에 대해서 살펴보았다. 전체적인 내용을 요약하면 다음과 같다. 먼저, 문제에 따라 지도 구조가 스스로 변화하는 구조 적응형 자기구성 지도 모델을 제안하였다. 여기에 기존의 알고리즘에 LVQ를 이용한 교사 학습 알고리즘을 추가하여, 교사 학습과 비교사 학습의 장점을 취하여 분류기로서 더 나은 결과를 얻었고, 학습 시간도 줄였다. 둘째로, 구조 적응형 자기구성 지도의 분화된 자식 노드의 가중치 초기화를 위해서 유전자 알고리즘을 적용하였다. 인식률에서 측면에서는 크게 성능 향상을 시키지는 못했지만 각 오프라인의 필기 숫자들의 인식률 편차를 크게 줄였다. 특히, 특정 숫자가 제대로 인식되지 않는 문제를 해결하였다. 그러나, 유전자 알고리즘의 수행 속도가 느리다는 단점 또한 노출하였다. 그래서 문제에 따라 적절하게 SASOM과 GASASOM을 선택할 필요가 있다.

참 고 문 헌

[1] S. B. Cho, "Self-organizing map with dynamical node splitting: Application to handwritten digit recognition," *Neural Computation*, vol.9, 1345~1355, 1997.

[2] D. Polani, and T. Uthmann, "Adaptation of Kohonen Feature Map Topologies by Genetic Algorithms," *Parallel Problem Solving from Nature*, 2, p. 421-429, 1992.

[3] D. Polani, and T.Uthmann,"Training Kohonen Feature Maps in different Topologies: an Analysis using Genetic Algorithms," *Proceedings of the Fifth International Conference on Genetic Algorithms*, p. 326-333, 1993.

[4] M. Mclnerney and A. Dhawan, "Training the Self-Organizing Feature Map using Hybrids of Genetic and Kohonen Methods," *Proceedings of ICNN' 94, International Conference on Neural Networks,* pp. 641-644, *IEEE Service Center*, 1994.

[5] T. Kohonen, "Self -organized formation of topologically correct feature maps," *Biol. Cyb.*, 43:59-69, 1982.

[6] T. Kohonen, Self-Organizing Maps, Springer, Berlin Heidelberg, 1995.

[7] K. Kiviluoto, "Topology Preservation in Self-Organizing Maps," *IEEE Transactions on Neural Networks*, 1996.

[8] T. Villmann, R. Der, M. Herrmann, and T. M. Martinetz, "Topology Preservation in Self-Organizing Feature Maps : Exact Definition and Measurement," *IEEE Transactions on Neural Networks*, vol 8, no. 2, 1997.

[9] A. Konig, "Interactive Visualization and Analysis of Hierarchical Neural Projections for Data Mining," *IEEE Transactions on Neural Networks*, vol 11, no. 3, 2000.

저 자 소 개

김현돈

1998년 : 연세대학교 컴퓨터과학과 (학사)
2001년 : 연세대학교 컴퓨터과학과 (석사)
E-mail : neoace@candy.yonsei.ac.kr

조성배

1998년 : 연세대학교 전산학과 (학사)
1990년 : 한국과학기술원 컴퓨터과학 (석사)
1993년 : 한국과학기술원 컴퓨터과학 (박사)
1993년~1995년 : 일본 ATR 인간정보통신 연구소 객원연구원
1995년~1998년 : 연세대학교 컴퓨터과학과 조교수
1998년 : 호주 University of New South Wales 초청연구원
1999~현재 : 연세대학교 컴퓨터과학과 부교수

관심분야 : 신경망, 패턴인식, 인공생명, 에이전트 등
E-mail : sbcho@csai.yousei.ac.kr