# 유전자 알고리즘을 이용한 구조 적응형 자기구성 지도의 자식 노드 가중치 초기화

김 현돈, 조 성배
연세대학교 컴퓨터과학과
e-mail:{neoace,sbcho}@csai.yonsei.ac.kr

# Optimal Weight Initialization of Structure-Adaptive Self-Organizing Map with Genetic Algorithm

Hyun-Don Kim and Sung-Bae Cho
Dept. of Computer Science, Yonsei University

**요 약**

구조 적응형 자기구성 지도는 일반적으로 자기구성 지도의 구조가 초기에 결정되어 학습이 끝날 때까지 변하지 않기 때문에 발생하는 문제를 해결하기 위해 지도의 구조를 학습 중에 적절하게 변경시킨다. 이때, 변화된 구조의 가중치를 어떻게 초기화시킬 것인가 하는 것이 중요한 문제이다. 이 논문에서는 기존의 비교사 학습방법에 LVQ 알고리즘을 이용한 교사 학습방법을 결합한 구조 적응형 자기구

방법은 기존의 구조 적응형 자기구성 지도 알고리즘보다 빠르게 학습되었고, 인식률 면에서도 기존의 방법보다 높은 값을 나타내었으며, 자기구성 지도의 특성인 위상 보존도 잘 이루어졌다. 오프라인 필기 숫자 데이터로 실험한 결과, 제안한 방법이 유용함을 알 수 있었다.

## 1. 서론

자기구성 지도는 비교사 학습방법을 통해서 자신을 스스로 학습시키고, 지도의 구조가 위상을 보존하는 특성이 있다. 이러한 특징으로 많은 분야에서 자기 구성 지도가 이용되고 있지만, 위상은 실험을 통한 데이터의 통계적인 특성을 고려할 경우에 가장 적절하게 선택된다. 그러므로, 자기구성 지도는 학습이 되기 전에 미리 위상을 고정 시켜야 한다는 결점을 가지고 있다. 이러한 결점을 해결하기 위해 동적으로 구조를 변화시키고자하는 연구들이 수행되어 왔고[1], 또한 자기구성 지도와 유전자 알고리즘을 결합한 방법이 제안되기도 하였다[2, 3, 4]. 유전자 알고리즘이 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 중, 가중치를 염색체로 암호화하는 방법을 이용하여 위상 구조를 변화시키는 것이 아닌, 노드의 효과적으로 가중치 초기화하고자 하였다.

전 연구에서 우리는 동적으로 노드가 분화되어 가는 구조를 가지는 구조 적응형 자기구성 지도 모델을 제안하였다[1]. 그리고 LVQ의 교사 학습 방법을 학습 방법에 결합하기도 하였다. 본 논문에서는 기본적인 구조 적응형 자기구성 지도 모델에 유전자 알고리즘을 이용하여 분화된 자식 노드들의 가중치 값을 염색체로 암호화하여 진화시켜서 학습 속도나 인식률에서의 향상을 꾀하였다.

## 2. 구조 적응형 자기구성 지도

이 방법은 각각 지도 초기화 및 초기 학습 단계, 노드 분화 단계, 분화된 노드 학습 단계의 세 부분으로 나뉘어진

다. 여기서 두 가지 학습이 필요한데, 첫 번째는 일반적인 SOM 알고리즘을 이용한 학습을 하는 것이고, 두 번째는 교사 학습 방법을 혼합한 LVQ 방식의 학습이다

.

## 2.1 지도 초기화 및 초기 학습 단계

이 단계에서는 지도의 크기를 임의로 설정하여 초기화하고 코호넨 알고리즘에 의해서 학습시킨다. 지도는 4×4의 크기로부터 시작하여 다음의 식으로 학습된다.

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

여기서 $\alpha$(t)는 학습률을 나타내는 함수, $n_{ci}(t)$는 이웃 함수, $m_i(t)$는 노드의 가중치, $x(t)$는 입력 벡터값이다[4]. $n_{ci}(t)$에서 $c$는 승리자 노드의 인덱스인데, 승리자는 다음의 수식으로 얻을 수 있다[5].

$$\|x - m_c\| = \min_i \{\|x - m_i\|\} \quad (2)$$

## 2.2 노드 분화 단계

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 모델에서는 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 99% 이하를 나타내는 노드들을 찾아서 분화시키는 방식을 사용하였다. 이때, 분화 조건을 99%로 한 이유는 학습데이터에 대해서 과도하게 학습되는 것을 막기 위해서이다. 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.

$$C = \frac{(P \times 2) + \sum N_c}{S} \quad (3)$$

여기서, $C$는 자식 노드의 가중치, $P$는 부모 노드의 가중치, $N_c$는 자식 노드의 이웃 노드의 가중치, $S$는 ($N_c$의 개수+2)를 나타낸다. 즉, $C$는 이웃노드와 부모 노드의 평균값으로 결정된다.

• 경우 1 : 가장 일반적인 경우인데, 그림 1 에서 $P_4$ 노드가 분화 될 경우를 예로 들면,

$$C_i = \frac{(P_4 \times 2) + P_i + P_{i+1}}{4} \quad (\text{여기서, } i = i \bmod 4)$$

이 된다.

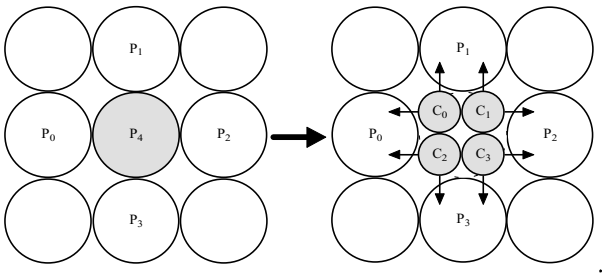


그림 1. 분화된 노드의 가중치 결정 (경우 1)

• 경우 2 : 이웃 노드의 수가 경우 2보다 작은 경우이다. 그림 2에서 자식 노드 $C_1$의 경우, 오른쪽에 노드가 없으므로, 이웃 노드로 고려해야 할 노드는 $P_1$ 밖에 없다. 그러므로 $C_1$의 값은 {($P_4$×2)+$P_1$}/3이 된다. 그리고 $C_3$ 노드의 경우, 이웃 노드로 고려해야 할 노드가 없으므로, $C_3$의 값은 {($P_4$×2)}/2가 된다. 즉, 부모 노드의 가중치와 같은 값을 가지게 된다.

• 경우 3 : 이웃 노드의 수가 경우 1보다 많은 경우이다. 그림 3의 노드 $P_4$의 경우를 살펴보자. 이 경우 자식 노드 $C_1$은 위쪽 방향으로

1개의 이웃 노드를 가지고, 오른 쪽 방향으로 3개의 이웃 노드를 가진다. 이때, 노드②의 가중치는 부모 노드의 가중치의 두 배의 합에 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.

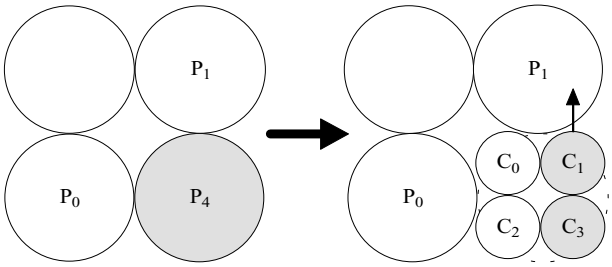


그림 2. 분화된 노드의 가중치 결정 (경우 2)

## 2.3 분화된 노드 학습 단계

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

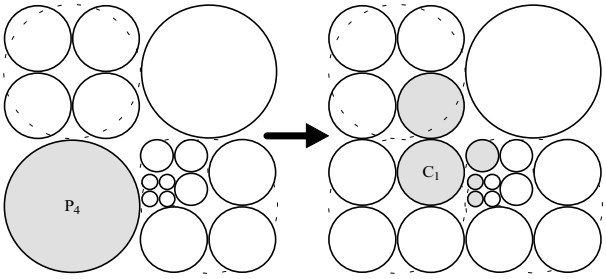


그림 3. 분화된 노드의 가중치 결정 (경우 3)

$$m_i(t+1) = m_i(t) + \alpha(t) \times n_{ci}(t) \times h_{ci}(t) \times \{x(t) - m_i(t)\} \quad (4)$$

여기서 $h_{ci}(t)$는 다음과 같이 계산된다.

$$\begin{cases} h_{ci}(t) = 1, \text{ if } x(t) \text{ 와 } m_i(t)\text{가 같은 클래스에 속할 경우} \\ h_{ci}(t) = 0, \text{ if } x(t) \text{ 와 } m_i(t)\text{가 다른 클래스에 속할 경우} \end{cases}$$

원래 LVQ 알고리즘은 최상 매칭 클래스에 대해서는 가중치를 증가시키고, 그 이외에는 가중치 값을 오히려 감소시켜서(즉, $h_{ci}(t) = -1$), 학습이 빠르게 진행 되도록 하였다[5]. 그러나, 이 경우 오히려 최적한 해를 나타내지 못하는 경우가 발생할 수 있기 때문에 위에서 보는 것과 같이 패배 노드들에 대해서는 학습을 시키지 않았다. 또한 기존의 LVQ 알고리즘에서는 이웃 함수($n_{ci}(t)$)가 없이 전체의 지도에 대해서 학습을 시킨다. 그러나, 본 논문에서는 이웃 함수를 사용하여 가까이 있는 최상 매칭 클래스들에 대해서 학습을 시켰는 데, 이것은 자기구성 지도의 특성인 위상 보존이 손상되는 것을 방지하기 위해서이다.

또한, 과도한 노드 분화를 막고 학습 시간을 향상시키기 위하여 노드가 상한 임계치 이상의 깊이 만큼은 분화되지 않도록 설정하였다.

## 3. 유전자알고리즘을 이용한 가중치 결정

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 표 1과 같다.

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

| |
|---|
| 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 ③~⑥의 과정을 종료 조건이 만족할 때까지 반복한다. |

표 1. 알고리즘

### 3.1 염색체의 구조

염색체는 $2\times2$로 분화된 자식 노드들의 가중치 값으로 인코딩 된다. 인코딩 예는 그림 4와 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 위상 보존을 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.

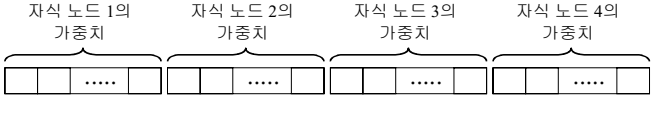


그림 4. 염색체의 구조

### 3.2. 유전자 연산자

유전자 연산자는 일반적인 유전자 알고리즘에서처럼 선

택 연산자, 교착 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 교착을 사용하였고, 교착률은 0.3으로 하였다. 돌연변이 연산자의 경우, 돌연변이 률이 0.01이 되도록 하였다. 돌연변이가 발생할 경우, 발생한 세그먼트 값을 초기화 단계에서 구한 최소 값과 최대 값 사이의 임의의 값으로 할당하였다.

### 3.3 적합도 평가 함수

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

$$F = \sum (m_c - v_i)^2 \qquad (5)$$

여기서 $m_c$ 는 염색체의 가중치를 나타내고 $v_i$ 는 노드에 할당된 데이터 값을 나타낸다.

## 4. 실험 결과

실험은 Concordia대학의 필기숫자 데이터 중 A(2000개)와 C(2000개)에서 Kirsh특징을 추출하여 사용하였다. 학습 횟수는 10000번, 학습률은 0.02로 하여 코호넨 알고리즘과 구조 적응형 자기구성 지도에 대한 실험을 수행하였다. 그리고 분화 정도는 레벨 5까지 되도록 하였다. 그리고, 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(HGSASOM)에 대해 수행하였다. 각각의 실험 결과는 표 2와 같다.

먼저 분화 회수를 비교 해 볼 경우, HGSASOM이 훨씬 작게 일어남을 알수 있는 데, 이것은 HGSASOM이 1.3배 가량 빨리 학습됨을 나타낸다. 또한 노드 수의 경우, 각각 246개와 269개를 나타내었다. 이것은 SASOM이 분화 회수가 많은 데도 불구하고 유용하지 않은 노드를 계속 분화해서 분화된 노드들이 실제 적으로는 사용되지 않았음을 알수 있다. 그만큼 SASOM의 학습 정도가 낮다는 것을 반영한다.

인식률에 대한 실험 결과는 그림 5와 그림 6와 같이 나타났다. 인식률은 학습 데이터에 대해서 HGSASOM이 91.05%를, SASOM이 89.75%를 나타냈다. 학습 데이터에 대해서는 거의 비슷한 인식률을 보였는데, 그림 2에서 보듯이 HGSASOM이 전체적으로 편차 없는 인식률을 보이고 있다. 즉, 특정 숫자에 대해서 편중되어 학습되지 않았다. 그리고 테스터 데이터는 HGSASOM이 86.4%를, SASOM이 84.4%를 나타내었다. 테스터 데이터에 대해서도 거의 비슷한 결과를 나타내었지만 역시, 숫자 별로 편차 없이 높은 인식률을 나타 내었다.

| | | HGSASOM | SASOM |
|---|---|---|---|
| 실험 환경 | | 최대 세대 : 1000<br>집단 크기: 20 | 학습 회수: 10000<br>학습률 : 0.02 |
| 인식률 | 학습 데이터 | 91.05 % | 89.75 % |
| | 테스트 데이터 | 86.4 % | 84.4% |
| 분화 회수 | | 101 | 132 |

표 2. 실험 결과

그림7은 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 가중치값을 나타낸다.

그림 8은 HGSASOM 방법의 위상 보존을 나타내는 그림이다. 그림에서 보듯이, 임의로 집단의 가중치값을 할당하였어도, 단계2에서의 가중치 값을 기본으로 하였기 때문에 위상 보존 역시 잘 이루어 지고 있음을 알 수 있다.

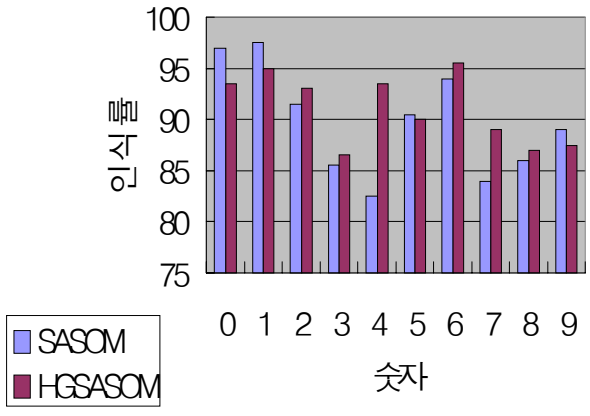


그림 5. 학습 데이터에 대한 인식률

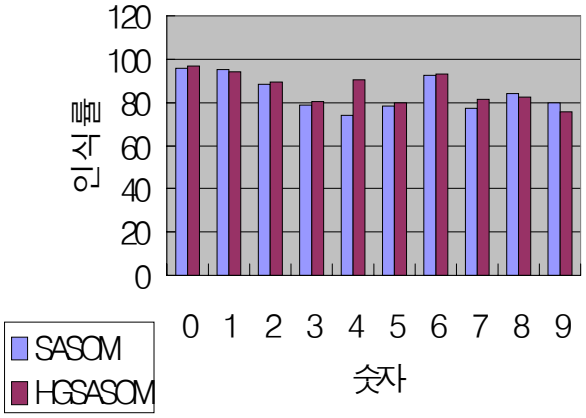


그림 6. 테스터 데이터에 대한 인식률

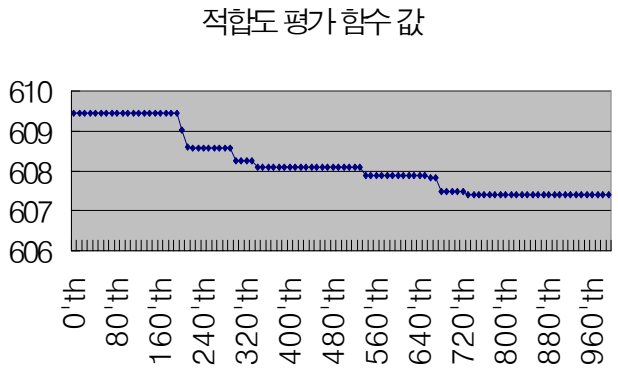


그림 7. 적합도 평가 함수 값의 변화

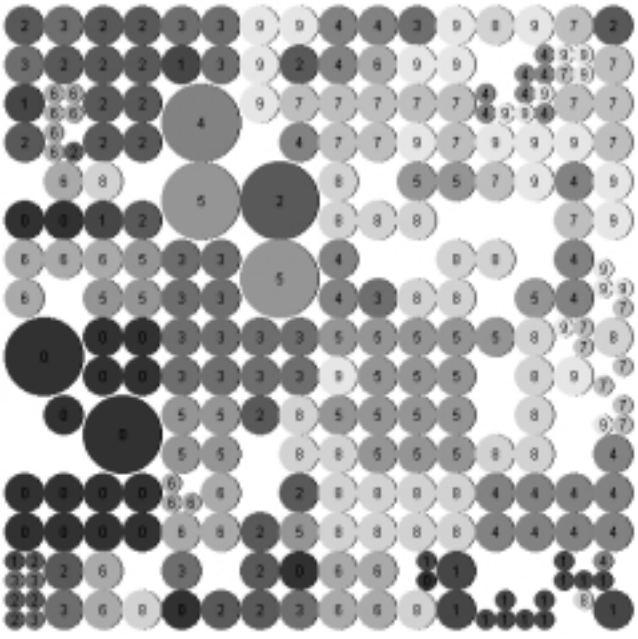

그림 8. 학습중의 위상

## 5. 결론 및 토의

본 논문에서는 구조 적응형 자기구성 지도 알고리즘을 기반으로 유전자 알고리즘을 결합하여 분화된 노드의 가중치값을 효과적으로 초기화하는 모델을 제안하였다. 위의 실험 결과에서 알 수 있듯이, HGSASOM은 기존의 알고리즘보다 근소하게 높은 결과를 보였고, 학습되는 시간이 1.3배 가량 빨라졌고, 학습 편차도 많이 낮아 졌음을 알 수 있다. 또한 자기구성 지도의 특성인 위상 보존이 잘 지켜지고 있음을 알 수 있었다.

## 참고 문헌

[1] S. B. Cho, "Self-organizing map with dynamical node splitting: Application to handwritten digit recognition," *Neural Computation*,Vol. 9, 13451355, 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-Organizing Maps*, *Springer*, Berlin Heidelberg, 1995.