비교사 학습과 교사 학습 알고리즘을 결합한 구조 적응형 자기구성 지도

김현돈, 조성배
연세대학교 컴퓨터과학과

A Structure-Adaptive Self-Organizing Map with Combination of Supervised and Unsupervised Learning Algorithms

Hyun-Don Kim and Sung-Bae Cho
Computer Science Department, Yonsei University

요 약

일반적으로 자기구성 지도에서는 구조가 초기에 결정되어 학습이 끝날 때까지 변하지 않기 때문에 각 문제에 대한 구조를 반복된 실험을 통해서 최적화 시켜야 한다. 그러나, 지도의 구조가 학습 중에 적절하게 변경된다면, 해당 문제에 가장 알맞은 구조의 지도를 생성할 수 있을 것이다. 이 논문에서는 기존의 적응형 자기구성 지도의 비교사 학습방법에 LVQ 알고리즘을 이용한 교사 학습방법을 결합한 구조 적응형 자기구성 지도 모델을 제안한다. 이 방법은 일반적인 자기구성 지도 알고리즘보다 작은 수의 노드를 가지고 높은 성능을 보일뿐만 아니라, 자기구성 지도의 특성인 위상 보존도 잘 이루어진다. 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. 자기구성 지도는 이러한 특성 때문에 데이터 시각화(visualization)가 필요하거나 위상 보존매핑(topology-preserving mapping)이 필요한 부분에서 이용되고 있다. 그러나 위상은 실험을 통한 데이터의 통계적인 특성을 고려할 경우에 가장 적절하게 선택됨에도 불구하고, 자기구성 지도는 학습이 되기 전에 미리 위상을 고정 시켜야 한다는 결점을 가지고 있다. 이러한 결점을 해결하기 위해 동적으로 구조를 변화시키고자하는 연구들이 수행되어 왔다. 트리 형태로 노드를 확장하는 방식이나 점증적으로 구조를 변화시키는 방법 등이 있다[2]. 본 논문에서는 점증적으로 구조를 변화시키는 방식을 통해 자기구성 지도의 구조를 적응시켜 보았다.

Fritzke는 지도의 구조를 표현하는데 용이한 사각형 구조의 이점을 살리기 위해서 사각형의 위상을 계속 유지하면서 구조를 변화시키는 방법을 사용하였다[1]. 그러나, 이 경우는 실제 문제에서 필요한 노드보다 많은 노드가 추가되는 단점이 있다. 따라서, 필요한 노드만을 분화시키는 동적 노드 분화 방법이 필요하다. 본 논문에서는 LVQ의 교사 학습 방법과 SOM의 비교사 학습 방법을 결합하여 위상을 보존하면서 데이터에 따라 적절하게 그 구조를 변화시키는 적응형 자기구성 지도 신경망을 제안한다.

2. 구조 적응형 자기구성 지도의 구조

이 모델은 다음의 알고리즘을 기본으로 하여 완성되었다.

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 ③~⑤의 과정을 종료 조건이 만족할 때까지 반복한다.

이 알고리즘은 각각 지도 초기화 및 초기 학습 단계, 노드 분화 단계, 분화된 노드 학습 단계의 세 부분으로 나뉘어진다. 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 c는 승리자 노드의 인덱스인데, 승리자는 다음의 수식으로 얻을 수 있다[4].

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

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.

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

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

• 경우 1 : 가장 일반적인 경우인데, 그림 1 에서 P_4 노드가 분화될 경우를 예로 들면,

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

이 된다.

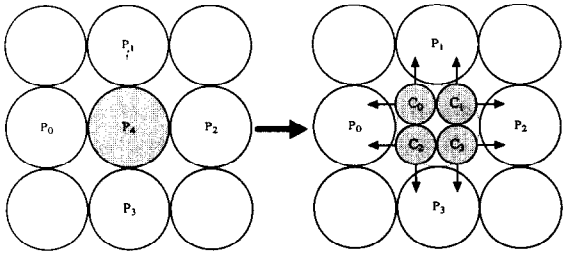

그림 1. 분화된 노드의 가중치 결정 (경우 1)

• 경우 2 : 이웃 노드의 수가 경우 2보다 작은 경우이다. 그림 2에서 자식 노드 C_1의 경우, 오른쪽에 노드가 없으므로, 이웃 노드로 고려해야 할 노드는 P_1 밖에 없다. 그러므로 C_1의 값은 $\{(P_4 \times 2)+P_1\}/3$이 된다. 그리고 C_3 노드의 경우, 이웃 노드로 고려해야 할 노드가 없으므로, C_3의 값은 $\{(P_4 \times 2)\}/2$가 된다. 즉, 부모 노드의 가중치와 같은 값을 가지게 된다.

• 경우 3 : 이웃 노드의 수가 경우 1보다 많은 경우이다. 그림 3의 노드 P_4의 경우를 살펴보자. 이 경우 자식 노드 C_1은 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.

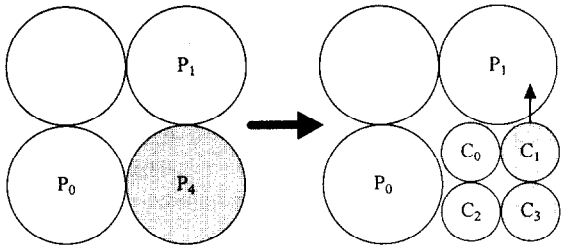

그림 2. 분화된 노드의 가중치 결정 (경우 2)

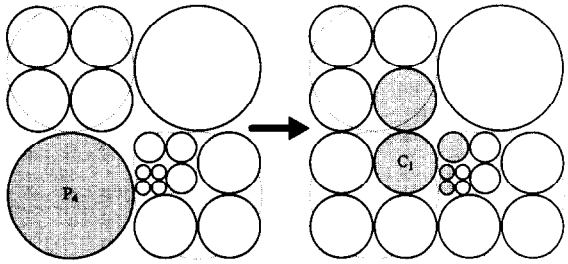

그림 3. 분화된 노드의 가중치 결정 (경우 3)

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.

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

여기서 $h_{ci}(t)$는 다음과 같이 계산된다.

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

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[3]. 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. 실험 결과

실험은 Concordia대학의 필기숫자 데이터 중 A(2000개)와 C(2000개)에서 Kirsh특징을 추출하여 사용하였다. 학습 횟수는 50000번, 학습률은 0.02로 하여 코호넨 알고리즘과 구조 적응형 자기구성 지도에 대한 실험을 수행하였다. 실험 결과는 그림 4와 같이 나타났다. 노드 수는 기존 코호넨 신경망이 900(30×30)인 반면에, 제안한 신경망은 324개다.

그림5는 반응 클래스와 그 빈도수에 따라서 노드가 어떻게 분화되었는 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 6은 노드가 분화된 전체 지도를 보여주는데, 원 안의 숫자는 각각 노드가 반응하는 클래스를 나타낸다. 또한, 그림에서 공백으로 나타나는 부분은 노드가 없는 부분이다. 그림에서 보듯이 비슷한 형태를 한 숫자들이 근접한 곳에 위치하고, 같은 숫자에 대해서 반응하는 노드들은 서로 뭉쳐있는 것을 볼 수 있다. 따라서, 제안한 방법은 신경망의 구조를 변화시키면서도 위상을 잘 보존시키는 것을 알 수 있다.

또한, 반응한 노드의 이웃을 고려하여 후보 클래스를 계산한 결과, 표1과 같이 이웃의 크기에 따라 다른 오류율을 얻을 수 있었다. 이웃의 크기는 노드 사이의 거리를 유클리드 거리로 계산한 값이다. 이때, 이웃의 크기에 따라 오류율이 줄어듦은 지도가 위상 보존이 잘 이루어지고 있다는 것을 보여 준다.

이웃의 크기	0	1	2	3	4	5
오류율(%)	12.4	8.95	4.8	3	1.9	1.1

표 1. 오류율

4. 결론 및 토의

본 논문에서는 코호넨의 자기구성 지도 알고리즘을 기반으로 문제에 따라 지도 구조가 스스로 변화하는 구조 적응형 자기구성 지도 모델을 제안하였다. 또한 기존의 알고리즘에 LVQ를 이용한 교사 학습 알고리즘을 추가하여, 더 나은 결과를 얻고자 하였고 학습 시간도 줄이고자 하였다. 위의 실험 결과에서 알 수 있듯이, 구조 적응형 자기구성지도는 기존의 코호넨 알고리즘보다 높은 결과를 보였고, 특히 구조가 변화했음에도 자기구성 지도의 특성인 위상 보존이 잘 지켜지고 있음을 알 수 있었다.

참고 문헌

[1] B. Fritzke, "Growing- grid - A self-organizing netwroks with constant neighborhood range and adaptation strength," *Neural Processing Letters*, Vol. 2, No 5, 9~13, 1995.

[2] B. Fritzke, "Growing self-organizing netwroks - Why?," *ESANN'96*, p.61~72, 1996.

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

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

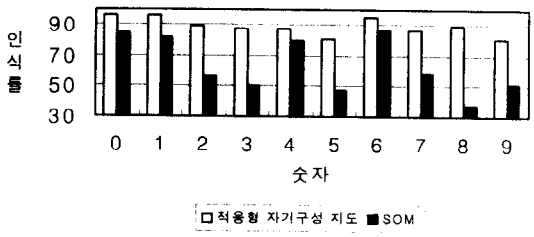

그림 4. 테스트 데이터에 대한 인식률

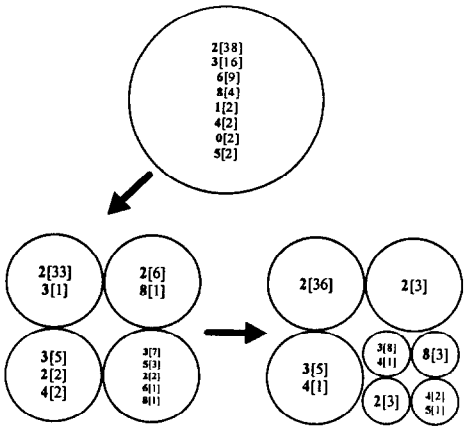

그림 5. 실제 분화의 예

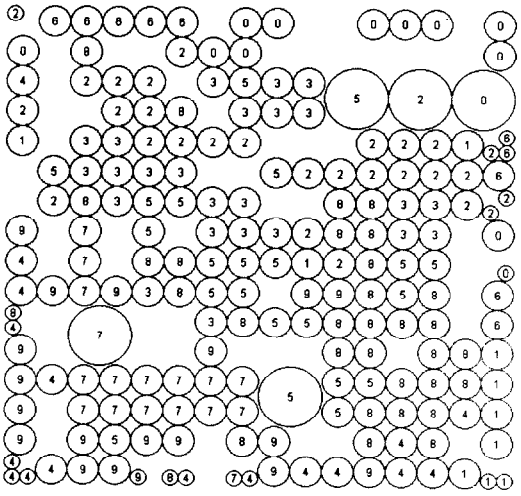

그림 6. 분화된 자기구성 지도