코호넨 신경망을 이용 바둑 사활문제를 풀기 위한 후보 첫 수들

이병두°, 금영욱*
성결대 정보통신공학부°, 성결대 컴퓨터공학부*
blee026@korea.com, ywkeum@sungkyul.ac.kr

Candidate First Moves for Solving Life-and-Death Problems in the Game of Go, using Kohonen Neural Network

Byung-Doo Lee°, Young Wook Keum*
Dept. of Information and Communication Engineering, Sungkyul University,
Dept. of Computer Engineering, Sungkyul University

요 약

바둑에 있어 사활문제는 컴퓨터 바둑을 구현하기 위해 반드시 극복해야 하는 기본적인 문제이다. 사활문제와 같은 국부적인 바둑 문제를 해결하기 위하여 고려해야 될 중요한 사항은 게임 트리의 엄청난 분기수와 그 깊이를 어떻게 처리하느냐이다. 본 논문에서 수행된 실험의 기본 착상은 둘러싸인 돌들을 죽이기 위해 인식된 첫 수들을 찾아내는 인간의 습성을 모방한 것이다. 바둑에 있어, 유사한 사활문제(패턴)들은 자주 유사한 해들을 갖는다. 유사한 패턴을 분류하기 위하여 코호넨 신경망(KNN)을 기반으로 한 군집화를 수행하였으며, 실험 결과는 고무적이며 사활문제를 풀기 위해 신경망으로 통제 학습을 사용하는 패턴 일치와 경쟁할 수 있음을 알아냈다.

ABSTRACT

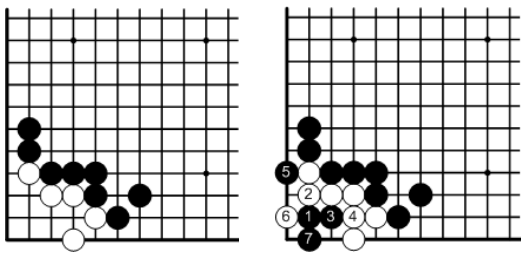
In the game of Go, the life-and-death problem is a fundamental problem to be definitely overcome when implementing a computer Go program. To solve local Go problems such as life-and-death problems, an important consideration is how to tackle the game tree's huge branching factor and its depth. The basic idea of the experiment conducted in this article is that we modelled the human behavior to get the recognized first moves to kill the surrounded group. In the game of Go, similar life-and-death problems(patterns) often have similar solutions. To categorize similar patterns, we implemented Kohonen Neural Network(KNN) based clustering and found that the experimental result is promising and thus can compete with a pattern matching method, that uses supervised learning with a neural network, for solving life-and-death problems.

Keyword : Go, life-and-death problem, Kohonen Neural Network(KNN), pattern clustering, pattern matching, Principal Component Analysis(PCA)

접수일자 : 2008년 11월 07일
심사완료 : 2009년 02월 03일

1. 서 론

현대 컴퓨터 바둑을 구축함에 있어 사활문제는 반드시 극복해야 할 당면 문제인 것이다. 많은 컴퓨터 바둑이 포석, 중반전의 전투에서 상당히 강함에도 불구하고 사활문제에 대해서는 아직까지도 상당한 진전을 보지 못하고 있다. 사활문제에 있어 둘러싸인 돌들의 삶을 위해 수비자가 두 집을 확보하기 위한 두 가지 방책으로는 (1) 돌들의 근간을 넓히거나, (2) 급소 점에 착수를 하는 방법이 있다. 반면에 공격자가 둘러싸인 돌들을 공격하는 두 가지 방법으로는 (1) 돌들의 근간을 좁히거나, (2) 급소 점에 착수를 하는 방법이 된다. 대부분의 바둑 기사는 수 초 내에 직관적으로 자신들이 기억해 왔던 패턴들과 비교하여 급소인 후보 착점들을 선정한 후, 이에 대한 모든 일련의 수순을 검토한 후 최종 첫 수를 둔다. [그림 1]은 흑백 간의 경계가 분명치 않은 사활의 한 예와 흑이 백을 잡기 위한 올바른 수순을 보여주고 있다.



(가) (나)

[그림 1] (가) 사활문제와 (나) 흑의 올바른 수순

상업용 컴퓨터 바둑들은 사활문제를 처리하는 알고리즘이 공개되어 있지 않으며, 그들의 대부분은 선택적 목적 지향 탐색(selective goal-oriented searching)인 전략적 탐색(strategic searching)을 사용하곤 한다.[1] 사활문제를 처리하는 가장 간단한 방법은 게임 트리를 이용하는 방법이 된다. 즉, 모든 단말 노드(terminal node)를 만날 때까지 가능한 모든 착수를 생성한 후에, 휴리스틱 평가 함수(heuristic evaluation function)를 적용하여 최적의 수를 결정하는 것이다. 이러한 맹목적 탐색(brute-force searching)은 엄청난 메모리와 계산 시간을 요하기 때문에 실제 적용할 수가 없다.

게임 트리를 이용하여 사활문제와 같은 국부적인 바둑 문제를 풀기 위해서 가장 중요하게 고려되어야 하는 사항은 게임 트리의 분기 수(branching factor)와 그 깊이(depth)를 어떻게 하면 줄일 수 있느냐는 것이다. 예를 들어, 첫 수로 11개의 착수가 가능하고 깊이가 10인 경우에 가지치기(pruning)를 하지 않는다면 $11^{10}(\approx 2.6\times 10^{10})$개 정도의 노드가 생성되어야 한다. 게임 트리가 클 경우에는 이러한 맹목적 탐색을 이용하여 사활문제를 풀 수가 없다.

Sasaki[2]는 사활문제를 풀기 위한 첫 수를 구해내기 위하여 신경망을 이용한 통제 학습을 구현했으며, 그가 구현한 신경망은 첫 수를 선택하는 데에 있어 프로 1단 정도의 실력을 보였다. 즉, 그는 사활문제를 푸는 데에 있어서 고전적인 탐색법이 아니라 신경망을 이용한 통제 학습법을 제안하였으나, 제한된 첫 수들을 갖고 최적의 수순을 찾아내는 탐색 방법에 대해서는 언급이 없었다.

저자는 사활문제에 있어 착수 가능한 첫 수의 개수와 일련의 수순을 찾아내기 위해 소요되는 계산 시간과의 연관성을 측정하기 위하여, 몇 개의 사활문제에 대해 알파베타($\alpha-\beta$) 가지치기(pruning)로 된 음부호 탐색(Negamax searching)을 적용해 보았다. [표 1]은 초기에 착수 가능한 수가 9 또는 10개가 있는 경우에 노드의 수와 계산 시간이 급격하게 증가됨을 보여주고 있다.

[표 1] 게임 트리에서의 노드 수와 계산 시간

착수 가능 첫 수(개)	계산 시간(초)	노드 수(개)		
		생성	방문	잔여
5	$\ll 1$	218	215	3
6	2	1,266	1,261	5
7	14	10,123	10,116	7
8	74	51,944	51,937	7
9	1,189	4.7×10^5	4.7×10^5	8
10	10,350	4.7×10^6	4.7×10^6	9

[표 1]에 있는 생성 노드 수는 게임 트리 내에 생성된 전체 노드의 수를 말하며, 방문 노드 수는 최적의 수순을 찾기 위해 가지치기된 노드를 포함하여 방문한 노드의 수를 말하며, 잔여 노드 수는 사활문제에 대한 최적의 수순을 나타내는 노드의 개수가 된다.

대단히 간단한 사활문제는 착수 가능한 첫 수의 개수가 10개보다 적으나, 대부분의 사활문제는 착수 가능한 첫 수의 개수가 10개 보다 크기 때문에 최적의 수순을 찾아내기 위해서는 엄청난 계산 시간을 요구하게 된다. 더군다나 대부분의 사활문제에 있어서 흑백간의 경계가 모호하기 때문에 둘러싸인 돌들에 대한 공격과 수비에 있어 최적의 수순을 찾아낸다는 보장도 없다.

본 논문에서는 탐색 방법에 대한 연구보다는 게임 트리내의 분기수를 줄일 수 있는 방법에 초점을 맞추었다. 즉, 사활문제를 위한 후보 첫 수들을 선택하기 위하여 패턴 군집화를 적용하였다.

2. 패턴 군집화

바둑에 있어, 유사한 사활문제(패턴)는 둘러싸인 돌들을 죽이기 위해 유사한 첫 수들을 자주 갖는다. 입력 패턴을 분류하여 유사한 패턴을 군집화하면서 그에 대한 첫 수들을 하나의 집합으로 형성한 후, 이를 군집화된 패턴들을 푸는 해로 사용할 수가 있다.

패턴을 분류하는 방법으로는 패턴 분류(pattern classification)와 패턴 군집화(pattern clustering)가 있다. 일반적으로 패턴 분류가 유사한 패턴을 분류하는데 사용된다. 패턴 분류와 패턴 군집화는 다음과 같은 차이가 있다. 패턴 분류는 새로운 입력 패턴을 이미 정해놓은 군에 등록하는 반면에, 패턴 군집화는 정해 놓은 군이 없는 상태에서 새로운 입력 패턴들을 서로 다른 군으로 분류하는데 사용된다. 패턴 분류의 약점은 새로운 입력 패턴이 이미 정해 놓은 군의 일원이 될 수 없는 경우가 발생될 수도 있는 반면에, 패턴 군집화는 새로운 패턴이 기존 군에 편입이 안 되는 경우에는 신규로 새로운 군을 생성한다. 본 논문에서는 아무런 기존의 군이 없는 관계로 패턴 군집화를 사용했다. 유사한 패턴들은 패턴 군집화를 통해 각 패턴으로부터 추출된 첫 수들의 집합을 갖는 군으로 편입이 된다. 패턴 군집화를 적용하는 근본적인 이유는 군내에 있는 첫 수들이 둘러싸인 바둑돌들을 죽이기 위한 첫 후보 착점으로 사용하기 위함이다. 즉, 게임 트리의 최초 분기 수를 현저히 줄일 수 있다는 것이다.

패턴 군집화를 위하여 통제 학습(supervised learning)이나 비통제 학습(unsupervised learning)이 사용될 수 있다. 코호넨 자기조직화 지도(Kohonen self organization map)가 바람직한 군집화 방법[3]이기 때문에, 본 논문에서는 비통제 학습으로서 코호넨 신경망(Kohonen Neural Network: KNN)을 구축했다.

2.1 변환된 원시 데이터

주성분 분석(Principal Component Analysis: PCA)은 데이터 집합 내에 있는 주성분(Principal Component: PC)을 결정하기 위한 다변량(multivariate) 기술이며, 많은 분야에서 사용되는 통계학적 도구이다. PCA를 사용하게 되면 상관이 있는 변수(correlated variable)의 데이터 집합이 상관이 없는 변수(uncorrelated variable)의 새로운 데이터 집합으로 변환이 되기 때문에, 원시 데이터의 차원을 줄일 수 있다. 다차원 데이터 집합에서, PCA는 군집화에 앞서 중요한 정보를 잃어버림이 없이 원시 데이터 집합의 차원을 줄이는데 가끔씩 사용이 된다. 주성분들은 서로 연관이 없고 첫 몇 개의 주성분들이 가장 큰 변량을 갖고 있기 때문에, 첫 몇 개 주성분들이 군집화 방법을 위해 사용된다.

원시 데이터를 위하여 [4]로 부터 588개의 사활문제를 추출하여 이를 입력 패턴으로 사용하였으며, 각 패턴은 242개의 속성(attribute)을 갖는다. 대부분의 사활문제는 바둑판의 4분의 1(즉, 11×11

행렬)로서 나타낼 수 있기 때문에, 흑백의 착점 여부를 고려하여 242(=2×11×11)개의 속성을 갖도록 하였다. 입력된 패턴 집합을 이용하여 588개의 패턴 벡터들을 생성시켰으며, 각 벡터는 121개의 속성을 갖는 2개의 집합으로 되어있다. 그래서 실험을 위한 최종 입력 행렬은 588개의 행과 242개의 열을 갖는다. MatLab을 이용하여 대부분의 계산이 이루어졌다.

원시 데이터들을 다음과 같은 두 단계로 군집화 시켰다. 첫 번째 단계는 PCA를 적용하여 가능한 한 전체 변도(variation)를 유지하면서, 입력 패턴 벡터들을 저 차원의 벡터(즉, 변환 벡터)로 변환하는 몇 개의 고유벡터를 유도해내는 것이고, 두 번째 단계는 변환 데이터에 대해 군집화를 적용하는 것이다.

한 개의 패턴 P_i는 다음과 같은 242개의 요소로 된 행벡터로 표현된다.

$$(B_{1,1}, B_{1,2} \cdots, B_{11,11}, W_{1,1}, W_{1,2}, \cdots, W_{11,11}) \quad (1)$$

여기서 $B_{i,j}(W_{i,j})$는 바둑판 위의 i번째 행과 j번째 열에 흑(백)돌의 착점 여부를 나타낸다. 만약 돌이 있는 경우에는 1이 되고, 없는 경우에는 0이 된다.

평균 벡터 μ는 588개의 모든 입력 패턴에 대한 행벡터의 평균을 내어 다음과 같이 구해진다.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} P_i \quad (2)$$

각 패턴에 대한 평균 벡터 μ와의 차인 $(P_i - \mu)$는 242×242개의 공분산(covariance)인 C를 구하는데 사용된다.

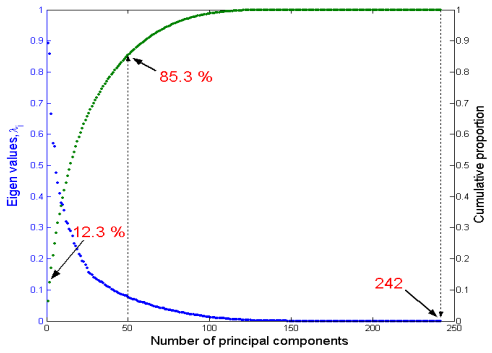
$$C = \frac{(P_i - \mu)^T (P_i - \mu)}{N} \quad (3)$$

여기서 행렬 $(P_i - \mu)^T$는 행렬 $(P_i - \mu)$의 전치 행렬(transpose matrix)이다.

다음으로 직교 고유벡터(orthogonal eigen vector)인 $E_k (k = 1,2,\cdots,242)$와 공분산 행렬 C의 고유치인 λ_k는 다음과 같이 계산한다.

$$CE_k = \lambda_k E_k, \qquad \lambda_k \geq \lambda_{k+1} \quad (4)$$

고유벡터(eigen vector)는 그들의 현저치(significance)에 따라 순서 배열된 직교 기반 집합(orthogonal basis set)이다. 첫 번째 PC들은 고유치 λ_k를 작게 함으로서 내림차순으로 정돈된 공분산 행렬의 고유벡터들이다. 첫 번째 n개의 고유벡터는 입력 패턴 내에 있는 대부분의 분산(variance)을 안고 있다. 차원을 줄이기 위해, 원시 c차원을 n차원 공간으로($n \ll c$) 줄일 수 있는 PC의 수를 제한할 필요가 있다. 첫 몇 개의 PC들을 구하기 위해 자주 사용되는 비공식적인 주먹구구식(rule of thumb)의 방법이 있다. 그림 2의 밑에 있는 곡선을 이루는 점들은 242개의 PC들을 보여주고 있다.



[그림 2] 고유치(eigen value)들과 누적된 비율

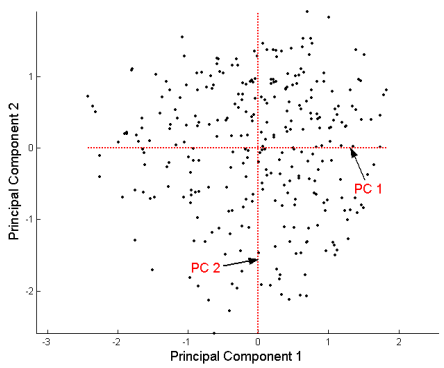
한편 위에 있는 곡선을 이루는 점들은 고유벡터의 수를 잘라낼 수 있는 척도로 쓰이는 전체 변도(total variation)의 누적 비율 σ_n을 보여주고 있다.

$$\sigma_n = \frac{\sum_{i=1}^{n} \lambda_i}{\sum_{i=1}^{c} \lambda_i}, \qquad n = 1,2,\cdots,c \quad (5)$$

여기서 $c\,(242)$는 전체 고유벡터의 수가 되며, n은 선택된 PC들의 개수가 된다.

첫 몇 개의 PC들을 선택하기 위하여 첫 번째 PC들을 잘라내는 한 가지 방법은 전체 변도의 누적 비율을 사용하는 것이다. 만약 σ_n의 값이 임계치(즉, 80%-90%)보다 큰 경우, 첫 번째 n개의 PC들은 많은 정보를 잃지 않고 원시 c개의 속성을 대치할 수 있다.[5] 다른 방법은 전체 변이성(variability)의 $\left(\frac{70}{c}\right)\%$ 보다 작은 모든 PC들을 없애버리는 방법이다.[6] 여기서 c는 주성분의 전체 개수, 즉 $c=242$가 된다. 또 다른 방법은 연속되는 고유치들 사이에 현저한 격차가 없을 때까지 첫 번째 n개의 PC들을 취하는 것이다. 그림 2를 살펴보면 현저한 격차를 보이는 50개의 고유벡터들이 원시 데이터 집합의 변량 대부분(85.3%)을 차지하고 있음을 알 수 있다.

시각적으로 이해를 돕기 위하여, 그림 2에서 보듯이 원시 데이터 집합의 변도 12.3%를 유지하고 있는 첫 두 개의 PC 공간 내에 [그림 3]과 같이 변환된 데이터 집합을 그려 보았다.



[그림 3] 첫 두 개의 PC 공간 내에 있는 변환된 입력 패턴 집합

첫 두 개의 PC(즉, PC 1과 PC 2)로 2차원 상에서 그림을 그린 이유는 주어진 절개(dissection)로 군집을 확정짓는 것보다는, 그 절개가 수긍할 수 있는지를 단순히 살펴보는 것이다. 군집화에 앞서 선택된 고유벡터의 수가 군집화의 정확성을 결정짓기 때문에, 본 논문에서는 242개의 모든 고유벡터를 사용하였다.

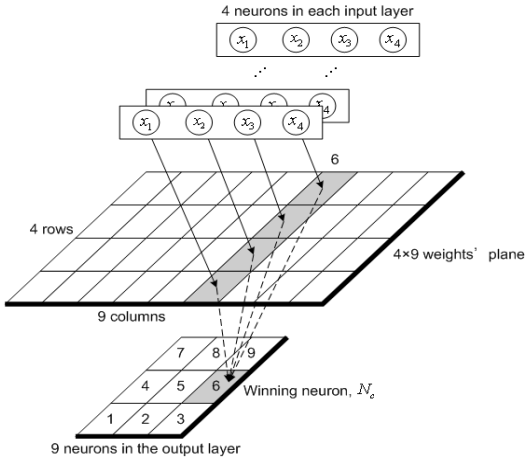
2.2 패턴 군집화

군집화는 데이터를 분류하기 위한 사전 작업이며, 내부적으로 강한 유사성을 갖는 입력 데이터들을 군집화하는데 목적이 있다.[7] 군집화는 데이터 마이닝(data mining), 기계 학습(machine learning), 컴퓨터 비전(computer vision) 및 여타 공학에서 중요한 분야로 강조되어 왔다. 일반적으로 군집화 방법은 계층 군집화(hierarchical clustering) 및 비계층 군집화(non-hierarchical clustering)로 구분이 된다.[8] 비계층 군집화의 한 예가 최근접 이웃 알고리즘(nearest neighbour algorithm)이 되며, 이는 각 요소(instance) 간의 유사성을 측정을 한 후 유사한 요소들을 동일한 군에 편입시키는 것이다. 서로 다른 군집화 방법은 군집 결과가 아주 다를 수가 있으며, 더군다나 어떤 특정 데이터 집합에 잘 작동하는 군집화 알고리즘이 다른 데이터 집합에는 형편없는 결과를 낳을 수도 있다.[9,10] 본 실험에서는 비 통제 학습으로 KNN 기반 군집화를 수행했다.

2.3 KNN 기반 군집화

본 논문에서는 비통제 학습으로 자기조직화 지도(self-organizing maps: SOM)의 개념을 기반으로 하고 2차원 뉴론 배열 내에 다차원 표현의 위상기하학(topology)을 갖는 KNN을 사용했다.[11,12]

KNN의 기본 개념은 [그림 4]에서 보듯이 입력 벡터에 가장 근접한 활성화 출력 계층(active output layer)내에 있는 뉴론(neuron)인 승리 뉴론(winning neuron)을 선택하는 것이다.



[그림 4] KNN의 2차원 계층 예. 망(network)은 입력 계층(input layer)에서 각 패턴 당 4개의 속성을, 출력 계층(output layer)에서 9(=3×3)개의 뉴론을 갖고 있다. 입력 계층에 있는 4개의 모든 뉴론은 가중치 평면(weights' plane)을 통해 출력 계층 내의 9개의 모든 뉴론에 연결되어 있다.

그러고 나서 승리 뉴론의 가중치(weight)와 이웃 뉴론의 가중치는 승리 뉴론으로 부터 거리에 반비례하여 갱신이 된다. 승리 뉴론을 선택하는 방법은 두 가지가 있는데, 첫 번째가 실험에서 사용한 유클리드 거리 계량(Euclidean distance metric)이고, 나머지 하나는 벡터 곱 계량(vector dot product metric)이 된다.

m차원 입력 벡터를 나타내는 1차원 입력 계층과 $r \times c$개의 출력 뉴론을 포함하는 2차원 출력 계층을 고려해 보자. 출력 계층에 있는 각 뉴론은 가중치 평면에 m개의 가중치 벡터와 연결되어 있다. 출력 계층에 있는 승리 뉴론 N_c는 다음과 같은 유클리드 거리 계산에 의거 선택이 된다.

$$\arg\min_{j=1,2,\cdots,r\times c} O_j = \arg\min_{j=1,2,\cdots,r\times c} \sqrt{\sum_{i=1}^{m}(x_i - w_{ij})^2} \quad (6)$$

여기서 O_j는 입력 계층 내에 있는 모든 뉴론과 가중치 평면 내에 있는 관련 가중치와의 유클리드 거리를 나타낸다. x_i는 입력 패턴 내에 있는 i번째 속성값이 되며, w_{ij}는 입력 계층에 있는 i번째 뉴론과 출력 계층 내의 j번째를 연결하는 가중치가 된다.

[그림 4]는 4개의 속성으로 된 1차원 입력 벡터와 출력 계층 내에 2차원 배열(3×3)의 출력 뉴론으로 구성된 아주 간단한 KNN의 구조를 보여주고 있다.

승리 뉴론인 N_c를 선택한 후에, 그와 연관된 가중치인 w_{ic}는 입력 벡터 x_i에 가까운 값으로 갱신이 되며, 승리 뉴론에 대한 이웃 뉴론의 가중치인 w_{ij} 역시 갱신이 된다. 이러한 보정(correction)은 다음과 같은 규칙에 의거 승리 뉴론으로 부터 거리에 따라 감소되어 진다.[13]

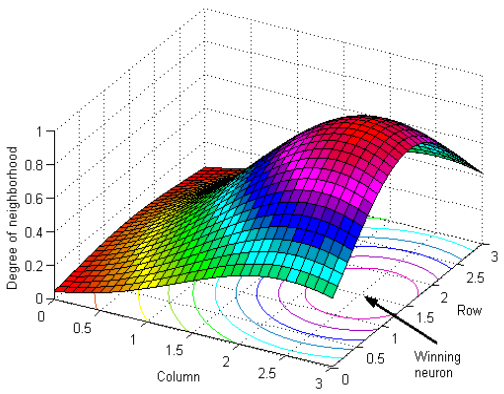
$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t) \cdot D(t) \cdot (x_i - w_{ij}(t)) \quad (7)$$

가중치 보정(alteration)의 정도를 결정짓는 인자는 학습률 $\alpha(t)$와 시간 t에 따른 승리 뉴론의 이웃 함수인 $D(t)$가 된다. 이웃 함수 $D(t)$는 시간 t에 따른 승리 뉴론 N_c와 출력 계층 내의 현재 뉴론인 N_j와의 위상 거리(topological distance)를 나타낸다. 다음과 같은 커널 기반 가우스 함수(kernel-based Gaussian function)가 자주 이웃 함수로 사용된다.

$$D(t) = \exp\left(-\frac{(r_c - r_j)^2 + (c_c - c_j)^2}{2\sigma(t)^2}\right) \quad (8)$$

여기서 r_c와 c_c는 승리 뉴론 N_c의 행과 열을 나타내며, (r_j, c_j)는 출력 계층 내에 현재 대응되는 뉴론의 행과 열이 된다. $\sigma(t)$는 시간 t에 따른 커널 폭을 나타낸다.

[그림 5]는 이웃 함수의 정도가 승리 뉴론 N_c와 현재 뉴론 N_j 사이의 거리에 대해 종속되어 단조 감소하고 있는 커널기반 가우스 함수를 보여주고 있다.



[그림 5] 가우스함수 ($\sigma(t)=1.18$, $r_c=1.5$, $C_c=2.5$)

학습률 $\alpha(t)$와 커널 폭 $\sigma(t)$는 학습되어지는 동안 시간 t에 대한 축소 함수(shrinking function)들이다.[14] 본 실험에서는 다음과 같이 학습률 $\alpha(t)$를 정의했다.

$$\alpha(t)=\alpha(0)\cdot\left(\frac{t_{\max}}{t_{\max}+t}\right) \qquad (9)$$

여기서 $\alpha(0)$는 초기 학습률이고, $t_{\max}$는 최대 회전수(epoch)이며, t는 현재 회전수를 말한다. 실험을 위해 $\alpha(0)$은 0.5, $t_{\max}$는 1,000으로 고정했다.

커널 폭 인자 $\sigma(t)$는 시간 t에 대해 위상학적으로 종속되는 척도 함수(scaling function)이다. 이 실험에서는 커널 폭 $\sigma(t)$를 다음과 같이 설정하였다.

$$\sigma(t)=\sigma(0)\cdot\left(\frac{t_{\max}}{t_{\max}+t}\right) \qquad (10)$$
$$=\sqrt{\left(\frac{r_w}{2}\right)^2+\left(\frac{c_w}{2}\right)^2}\cdot\left(\frac{t_{\max}}{t_{\max}+t}\right)$$

여기서 $\sigma(0)$은 출력 계층 내 지도의 지름에 종속되는 초기 커널 폭이 되며, r_w와 c_w는 각각 출력 계층 내 지도의 행과 열의 폭이 된다. 실험에서는 $\sigma(0)$을 $\frac{30}{\sqrt{2}}$로 설정하였다. 즉, $r_w=c_w=30$이 된다.

KNN을 적용하기 위하여, 입력 계층과 출력 계층을 갖는 2계층 신경망을 구축하였다. 입력 계층에서 흑과 백을 표현하는 121개의 속성을 갖는 두 개의 집합을 제공하였다. 출력 계층을 위해 900(=30×30 격자) 뉴론을 구축하여, 최악의 경우에 각 입력 패턴이 서로 다른 각각의 군에 편입될 수 있도록 충분히 설정하였다. Kohonen은 신경망을 훈련시키기 전에 표준화(normalization)하기를 권하기 때문에[15], 모든 입력 벡터에 대해 다음과 같이 표준화를 하였다.

$$x'=\frac{x}{\sqrt{\sum_{j=1}^{242}x_j^2}} \qquad (11)$$

여기서 x는 입력 패턴을 가리키며, $\frac{1}{\sqrt{\sum_{j=1}^{242}x_j^2}}$는 표준화인자(normalized factor)로 불리우며, x'는 표준화된 입력 패턴을 가리킨다.

KNN을 훈련하는 계산 절차는 다음과 같다.

(1) 588개의 원시 패턴을 500개의 훈련용 패턴(training patterns: 실제 훈련된 패턴 수는 흑과 백의 색상을 바꾸어 적용하였기 때문에 1,000개가 된다)이며, 교차 유효성(cross validation)을 수행하기 위하여 88개의 유효성 패턴(validation patterns)으로 분리하였다.

(2) 1,000개의 훈련용 입력 패턴이 교차 유효성을 위하여 표준화되었다.

(3) -0.01부터 +0.01지의 사이의 값에서 무작위로 초기 가중치를 부여하였으며, 학습률 $\alpha(0)$은 0.5로 초기 커널 폭 $\sigma(0)$는 $\frac{30}{\sqrt{2}}$로 초기화하였다.

(4) 가중치 행렬(weight matrix)에 과다 영향(over influencing)을 받는 것을 피하기 위해, 각 회전수에 훈련용 입력용 패턴을 무작위로 순차화시켜 훈련시켰다.

(5) 훈련용 입력 패턴과 관련 가중치를 유클리드 거리 척도를 이용하여 계산한 후에, 승리 뉴론을 선택했다.

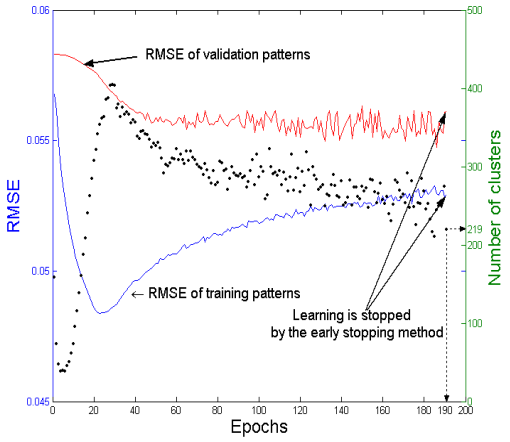
(6) 다음으로 승리 뉴론과 이웃 뉴론에 대한 가

중치를 갱신했다.

(7) 훈련을 계속할지 여부를 결정하기 위해, 초기 정지 방법(early stopping method)을 사용하였으며, 훈련 시 수렴률(convergence rate)을 확인하면서 신경망을 일반화시켰다.

훈련용 패턴 집합의 과도적합(overfitting)을 피하기 위해 교차 유효성을 이용한 조기 정지 방법으로 KNN을 훈련 및 일반화시켰다. 조기 정지 방법의 기본 개념은 훈련용 데이터로 훈련 중에 신경망에 유효 에러(validation error)가 현저하게 증가할 경우 훈련을 멈추는 것이다. 망을 훈련시키기 위한 회전수는 1,000번으로 제한을 두었으며, 정지 계수(stopping parameter)는 1.2로 정했고, 훈련용 띠(training strip)의 길이인 임계치 θ는 5로 정했다.

[그림 6]은 훈련용 패턴과 유효성 패턴에 대해 시간에 따른 제곱평균 오차(root-mean-square error: RMSE) 및 생성된 군의 개수를 보여주고 있다.



[그림 6] 훈련 회수 1,000회전 및 $\sigma(0)=0.5$가 적용된 조기 정지 방법으로 훈련된 훈련용 패턴과 유효성 패턴의 RMSE. 검은 점은 시간 t회전에 따른 생성된 군의 수.

망에 정지 계수를 적용한 후에, 훈련은 190회전만에 끝났다. 유효성 패턴에 대해서는 RMSE가 0.055에 수렴하는 것을 알 수 있으며, 반면에 훈련용 패턴에 대한 RMSE는 30회전에서 증가하는 것을 볼 수가 있으며, 그 때 군집화된 군의 수는 200에서 300개 범위가 되었다. 190회전을 끝낸 후, 1,000개의 훈련용 패턴은 219개의 서로 다른 군에 군집화가 되었다.

88개의 유효성 패턴을 시험용 패턴(test pattern)으로 재사용 하였으며, 사활문제를 풀기 위한 첫 수에 대한 집합을 얻기 위해 패턴 분류를 하였다. 88개의 시험용 패턴 중에서, 78개(88.7%)의 패턴들은 이미 생성된 219개의 서로 다른 군에 분류가 되었고, 10개(11.3%)의 패턴은 훈련용 패턴이 속한 군에 편입이 안 된 미분류 상태가 되었다. 미분류가 된 시험용 패턴이 생긴 주요 원인은 영역 크기(domain size)가 다소 크기 때문이다.

다음으로, 패턴 분류된 78개의 패턴들이 사활문제를 푸는 해, 즉 첫 수를 각 군에 어느 정도 정확하게 저장하고 있는지를 확인하였다. 78개 시험용 패턴에 대한 올바른 첫 수들을 보유하고 있는 정확도는 표 2에서 보듯이 62.8%가 되었다. 표 2에서 보듯이 이 결과를 Sasaki에 의해 수행된 실험 결과와 비교를 하였다.

[표 2] Sasaki에 의해 수행된 신경망으로 된 통제 학습과 KNN으로 된 비통제 학습과의 성능 비교

망 형태	역전파망 (Back-propagation network)	코호넨 신경망 (KNN)
도메인 규모	9×9	11×11
학습 형태	통제	비통제
적용 방법론	패턴 일치	패턴 군집
훈련용 패턴 수	2,000	1,000
시험용 패턴 수	1,000	88 (미분류 10개 포함)
올바른 첫 수를 뽑아내는 정확도	34.9% (첫 번째 답만 적용) 또는 65.0% (첫 5개의 답을 적용한 경우)	분류된 78개 패턴에 대해 62.8% (군 내의 첫 수들의 집합 적용)

Sasaki는 2,000개의 패턴을 통제된 신경망으로 훈련시켰으며, 훈련된 신경망을 이용하여 1,000개

의 시험용 패턴에 대한 첫 수들을 추출해냈다. 표 2에 있는 34.9%의 정확도는 출력 계층 내에 있는 가장 큰 값 하나를 적용한 경우가 되고, 65.0%는 가장 큰 값 5개를 적용한 경우이다.

[표 2]는 사활문제를 풀기 위한 첫 수를 선택함에 있어 KNN 기반 군집화가 고무적인 방법임을 보여주고 있으며, 본 실험을 통해 다음과 같은 사실을 발견했다.

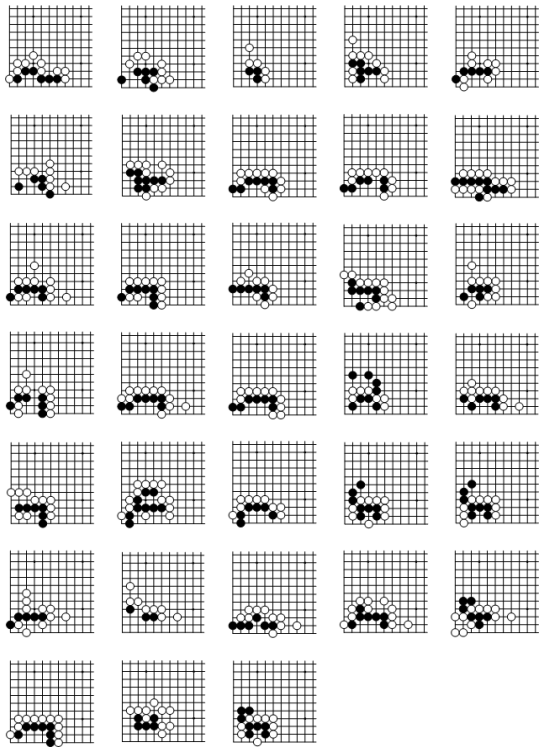
- 만약 훈련용 패턴으로 상당한 정도의 개수가 망에서 훈련이 된다면 비통제 학습으로서의 KNN은 상당히 높은 적중률로 올바른 첫 수들을 발견한다는 사실과,
- KNN으로 된 패턴 분류(pattern classification)가 신경망으로 된 통제학습인 패턴 일치(pattern matching)와 경쟁할 수 있음을 확인했다.

[그림 7]은 KNN 기반 군집화에 의해 생성된 어느 한 군의 군집화된 패턴들의 유사성을 보여주고 있다. 실제로 군집화된 패턴들 간의 선명한 유사성을 찾아볼 수가 없다. 동일한 군에 군집화가 되지 말아야 할 패턴들이 다소 있기 때문에, 사활문제를 풀기 위한 첫 수를 구하는데 있어 KNN 기반 군집화 하나로만 문제를 푸는 것인 한계가 있음을 알 수 있다.

3. 결 론

게임 트리를 이용하여 사활문제와 같은 국부적인 문제를 풀기 위해 중요한 고려 사항은 어떻게 게임 트리에서의 엄청난 초기 분기수와 깊이를 처리하느냐 하는 것이다. 이러한 문제를 처리하기 위해서는 휴리스틱(heuristic) 방법을 적용해야 한다.

유사한 패턴은 사활문제에서 유사한 첫 수를 갖고 있기 때문에, 패턴 군집화(또는 패턴 분류)는 사활문제를 풀기 위한 첫 수들을 얻어내는 유용한 방법이다. 이렇게 얻어진 첫 수들은 게임 트리를 이용해 사활문제를 푸는 데에 있어 초기 분기수를 줄이는데 상당한 기여를 하게 된다. 실험의 목적은 패턴 군집화가 실제의 컴퓨터 바둑에서 어느 정도 적용될 수 있는가를 확인하는 것이다.



[그림 7] 25×25 출력 뉴론으로된 KNN에 의해 군집화된 어느 한 군내의 패턴들

저자는 원시 데이터(사활문제)의 차원을 줄이기 위해 PCA를 적용해 보았으나, 사활문제는 비교적 작은 차원(예: 11×11)이기 때문에 굳이 PCA를 적용하지 않아도 됨을 확인하였다. 그러나 모든 고유벡터를 갖고 PCA에 생성된 변환 데이터는 원시 데이터를 군집화하는데 유용하다는 사실 역시 발견했다. 마지막으로 (비 통제 학습의 하나인) 코호넨 신경망을 사용한 패턴 군집화의 성능은 매우 고무적이며, (신경망으로 된 통제 학습을 사용하는) 패턴 일치와 경쟁할 수 있음도 확인되었다.

참고문헌

[1] J. Burmeister, “Studies in Human and Computer Go: Assessing the Game of Go as a Research

Domain for Cognitive Science", PhD thesis, University of Queensland, Australia, 1995.
[2] N. Sasaki, Y. Sawada, J. Yoshimura, "A Neural Network Program of Tsume-Go", Computer and Games 1998, pp. 167-182, Springer-Verlag, Germany, 1999.
[3] T. Kohonen, "The Self-Organizing Map", Springer, Germany, 2nd Edition, 1997.
[4] M. Harada, "Tsumego - Life and Death Problem of Go", http://www.hitachi.co.jp/Sp/tsumego/ index-e.html.
[5] T. Thireou, L. Strauss, G. Kontaxakis, S. Pavlopoulos, A. Santos, "Principal Component Analysis in Dynamic Position Emission Tomography", IEA/AIE 1998, vol. 2, pp. 612-618, 1998.
[6] S. Raychaudhuri, J. M. Stuart, R. B. Altman, "Principal Components Analysis to Summarize Microarray Experiments: Application to Sporulation Time Series", Pacific Symposium on Biocomputing 2000, pp. 452-463, USA, 2000.
[7] A. Kishimoto, "Transposition table driven scheduling for two-players game", Master's thesis, University of Alberta, Canada, 2002.
[8] H. Liu, H. Motoda, Feature Extraction, Construction and Selection: A Data Mining Perspective, Kluwer Academic Publishers, USA, 1998.
[9] J. Mao, A. K. Jain, "A Self-Organizing Network for Hyperellipsoidal Clustering (HEC)", IEEE Transactions on Neural Networks, 7(1):16-29, 1996.
[10] K. Yeung, W. Ruzzo, "An Empirical Study on Principal Component Analysis for Clustering Gene Expression Data", Bioinformatics, 17(9): 763-774, 2001.
[11] K. Das, W. Perrizo, Q. Ding, "Using Neural Networks for Clustering on RSI Data and Related Spatial Data", ISCA International Conference Reuse and Integration, pp. 111-116, USA, 2000.
[12] Z. Zupan, Z. Gasteiger, Neural Networks in Chemistry and Drug Design, Wiley-VCH, 1999.
[13] S. Lawrence, C. L. Giles, A. C. Tsoi, A. D. Back, "Face Recognition: A Hybrid Neural Network Approach", Technical report UMIACS-TR-96-16 and CS-TR-3608, Institute for Advanced Computer Studies University of Maryland, USA, 1996.
[14] T. Reutterer, M. Natter, "Segmentation Based Competitive Analysis with MULTICLUS and Topology Representing Networks", Computers and Operations Research 2000, 27(11-12): 1227-1247, 2000.
[15] T. Kohonen, "The Self-Organizing Map", IEEE, vol. 78, pp. 1464-1480, 1990.

이병두(Byung-Doo Lee)

1982 한양대 원자력공학 학사
1991 서강대 정보처리학 석사
2005 오클랜드대 컴퓨터공학 박사
2007~현재 성결대 정보통신학부 강사

관심분야 : 컴퓨터공학, 인공지능, 컴퓨터바둑

금영욱(Young Wook Keum)

1978 서울대 수학과 학사
1992 서강대 전산학과 석사
1997 서강대 전산학과 박사
1982~1993 IBM 시스템즈 엔지니어
1997~현재 성결대 컴퓨터공학부 부교수

관심분야 : 수치해석, 병렬/분산처리,
컴포넌트 S/W 개발