퍼지 및 지능시스템학회 논문지 2005, Vol. 15, No. 1, pp. 1-5

강화학습을 이용한 n-Queen 문제의 수렴속도 향상

The Improvement of Convergence Rate in n-Queen Problem Using Reinforcement learning

임수연 · 손기준 · 박성배 · 이상조

SooYeon Lim, KiJun Son, SeongBae Park and SangJo Lee

경북대학교 컴퓨터공학과

요 약

강화학습(Reinforcement-Learning)의 목적은 환경으로부터 주어지는 보상(reward)을 최대화하는 것이며, 강화학습 에이전트는 외부에 존재하는 환경과 시행착오를 통하여 상호작용하면서 학습한다. 대표적인 강화학습 알고리즘인 Q-Learning은 시간 변화에 따른 적합도의 차이를 학습에 이용하는 TD-Learning의 한 종류로서 상태공간의 모든 상태-행동 쌍에 대한 평가 값을 반복 경험하여 최적의 전략을 얻는 방법이다. 본 논문에서는 강화학습을 적용하기 위한 예를 n-Queen 문제로 정하고, 문제풀이 알고리즘으로 Q-Learning을 사용하였다. n-Queen 문제를 해결하는 기존의 방법들과 제안한 방법을 비교 실험한 결과, 강화학습을 이용한 방법이 목표에 도달하기 위한 상태전이의 수를 줄여줌으로써 최적 해에 수렴하는 속도가 더욱 빠름을 알 수 있었다.

Abstract

The purpose of reinforcement learning is to maximize rewards from environment, and reinforcement learning agents learn by interacting with external environment through trial and error. Q-Learning, a representative reinforcement learning algorithm, is a type of TD-learning that exploits difference in suitability according to the change of time in learning. The method obtains the optimal policy through repeated experience of evaluation of all state-action pairs in the state space. This study chose n-Queen problem as an example, to which we apply reinforcement learning, and used Q-Learning as a problem solving algorithm. This study compared the proposed method using reinforcement learning with existing methods for solving n-Queen problem and found that the proposed method improves the convergence rate to the optimal solution by reducing the number of state transitions to reach the goal.

Key words : 강화학습, Q-Learning, 에이전트, 보상, 상태전이

1. 서 론

최근의 전자화된 정보량의 폭발적인 증가는 사람들에게 다양한 정보에 대한 충족감은 제공하지만, 처리해야 할 정보의 양이 과다하게 늘어남으로써 원하는 정보를 찾아내는데 많은 부담을 안겨주고 있다. 또한 다양하고 복잡한 정보의 형태는 원하는 정보를 빠르고 정확하게 찾는 기술을 요구하고 있다. 이 때, 과거의 경험이나 환경의 변화를 감지하고 이로부터 기존의 지식을 변경하고 환경에 적응하는 학습능력은 가장 중요한 기술 중의 하나이다. 지능적인 인간 에이전트(agent)는 사회 환경 속에서 시행착오 뿐 아니라 즉각적인 정보와 학습된 지식, 특별한 경험들을 통하여 학습하게 된다[1].

강화학습(reinforcement learning)은 사전 지식이 필요 없고, 예제가 아닌 경험과 관찰을 통해서 학습한다. 환경에 대한 입력은 단지 지연(delay)된 스칼라 보상(scalar reward)만인 상태 환경에서 시행착오를 통하여 효율적인 결정 전략을 학습할 수 있다. 목적에 도달하기 위한 임무는 각 행동에 대한 장기적인 할인된(discounted) 보상을 최대화 시키는 것이며, 비록 초기 학습 속도는 느리지만 복잡한 임무를 수행하는데 있어 효율적이다[2].

본 논문에서는 강화학습을 적용하기 위한 예를 n-Queen 문제로 정하고 강화학습을 이용한 문제풀이 알고리즘을 제안하고자 한다. 제안한 알고리즘이 주어진 환경에서 빠르고 효율적임을 보이기 위하여 기존의 방안들과 비교하는 실험을 수행하고 이를 평가하고자 한다.

접수일자 : 2004년 11월 10일
완료일자 : 2004년 12월 7일
감사의 글 : 본 연구는 2004년 한국과학재단의 "신진연구자연수지원"사업에 의하여 지원받았음. (M02-2004-000-20577-0)

2. 관련연구

n-Queen 문제는 정방형 체스 판에 n개의 여왕 말을 위치시키는 문제인데, 이 때 어느 두 여왕도 같은 행, 같은 열, 같은 대각선상에 놓여있으면 안된다는 제한 조건이 있다. 이 때 해의 후보의 개수는 n !개가 된다.

2.1 백트래킹(Backtracking)이용 방법

백트래킹은 어떤 집합에서 어떤 기준을 만족하면서 그 집합에 속한 대상의 순서를 선택하는 문제를 푸는데 사용되는

트리의 수정된 깊이우선탐색(depth first search : DFS)이다.

백트래킹은 *n*-Queen 문제를 해결하기 위해 이용하는 가장 일반적인 방법으로 상태 공간 트리를 깊이우선탐색하여 각 마디가 유망한지를 검사하고, 만약 유망하지 않으면 그 마디의 부모마디로 되추적한다. 이는 마디가 유망하고 그 마디에 해답이 없는 경우에만 그 마디의 자식 노드를 방문하는 것을 제외하고는 깊이우선탐색과 같다.

2.2 몬테칼로기법을 도입한 백트래킹이용 방법

2.1의 방법보다 더 효율적인 *n*-Queen 문제 해결 방법 중의 하나는 몬테칼로 알고리즘(Montecarlo Algorithm)을 사용하는 것이다. 몬테칼로 알고리즘은 확률적 알고리즘으로 다음 실행할 명령이 때로는 무작위(random)로 결정되며, 표본 공간의 무작위 평균치를 가지고 표본공간에서 정의된 무작위 변수의 기대치를 추정한다[3].

즉, 초기의 여러 행에 대해서 무작위(random)로 여왕말을 배치하고 남아있는 행에 대해서만 백트래킹을 적용하는 것이다.

2.3 강화학습이용 방법

상태 공간 트리를 탐색하여 문제의 해를 찾기 위하여 쓰는 일반적인 방법은 깊이우선탐색(DFS)이며, 탐색 중에 방문하는 각 노드에서 어떤 특정한 검사를 하여 그 노드의 상태가 유망한지 즉, 해가 될 가능성이 있는지를 결정한다. 강화학습을 이용하여 *n*-Queen 문제를 해결하기 위해서는 각각의 노드에서 최선의 다음 노드를 선택할 필요가 있다. 이를 위해서는 각 노드가 얼마나 좋은 상황인지를 추정할 필요가 있는데, 이러한 추정값을 내는 함수를 평가함수(evaluation function)라고 한다. 대부분의 평가함수는 휴리스틱(heuristic)을 도입해서 노드를 평가한다.

가장 간단한 알고리즘은 현재 노드로부터 선택가능한 모든 노드들에 대해서 평가함수를 적용하고, 그 중에서 가장 큰 값을 가지는 노드로 분기하는 것이다. 그러나 평가함수는 기본적으로 한 노드에만 국한된 평가를 하므로, 선택한 노드가 현재는 최선이라고 할지라도 결과적으로 승리를 이끌 수 있는 노드인지는 알 수가 없다.

따라서 현재 선택된 노드의 가치를 평가하여 다음 상태의 가치를 추정하여 학습하는 것이 유용하며, 어떤 상태에서 어떤 행동을 선택했을 때의 가치를 계산하는 것을 무한히 반복하게 되면 결국 최종 목적에 수렴하게 될 것이다.

3. 강화학습(reinforcement learning)

어떤 상태에 있는 에이전트가 한 행동을 수행하고 환경으로부터 보상을 받는 것을 결정 프로세스(decision process)라고 하며, 새로운 상태가 단지 현재 상태와 행동으로 결정되는 결정 프로세스를 MDP(Markov decision process)라 한다. 특히 환경으로부터 받는 보상을 기반으로 수행할 행동을 배우는 학습을 강화학습이라고 한다.

3.1 강화학습모델

강화학습은 동적 프로그래밍과 교사학습을 혼합한 형태의 학습으로서 학습을 수행하는 에이전트는 에이전트의 외부에 존재하는 환경(environment)과 시행착오(trial and error)를 통해 상호작용(interaction)하면서 학습한다. 즉, 학습을 수행하는 동안 주어진 환경에서 취할 수 있는 행동을 시도하며, 외부 환경으로부터 에이전트가 선택한 행동에 대한 평가로서 스칼라형의 강화값을 받아 강화(reinforce)된다. 이는 자신이 수행한 행동에 대하여 보상값을 받아서 조금씩 좋은 방향으로 행동을 강화시키는 학습방법으로 MDP방식에 기반하고 있다.

MDP에서는 현재 상태에서 최적의 행동을 결정하기 위해 목표까지의 모든 가능한 행동들을 계산하여 가장 높은 값을 가지는 행동을 결정한다. 하지만 상태의 개수가 커지거나 정확한 확률값이나 보상값을 알 수 없을 경우 불가능하다는 단점이 있다. 이에 반해 강화학습은 동적 프로그래밍이라 불리는 전통적인 최적화 기법에 접근하는 온라인 기법이며, 현재 상태에서 최적의 행동을 계산을 통해 결정하지 않고 여러 번의 시행착오에 기반한 경험에 의하여 각 상태에서의 최적의 행동을 조금씩 학습해 나간다[4].

이 때, 환경은 상태(state)들로 이루어져 있으며, 행동(action)을 취함으로써 상태의 변경이 이루어진다. 특히 목표에 도달하기 위해 취한 행동들을 정책(policy)이라고 하며 에이전트의 임무는 이 제어정책(control policy)을 학습하는 것이다. 이 에이전트의 스케쥴, 즉 제어정책을 π로 나타내었을 때 우리가 풀 문제는 가장 좋은 π를 찾는 것이며 이는 가장 많은 획득 축적된 값(accumulated reward)을 얻게 하는 π를 의미한다.

에이전트는 현재의 상태에서 다음 행동을 선택하기 위하여 정책 $\pi : S \rightarrow A$를 학습하게 된다. 임의의 초기상태 st에서 임의의 정책 π를 취함으로써 획득 축적된 값은 다음과 같이 정의된다.

$$V^{\pi}(s_t) \equiv r_t + \gamma\, r_{t+1} + \gamma^2 r_{t+2} + \ldots \equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

이 때 s_t는 초기 상태, r_t는 시간 t에서의 보상이다. 보상을 합산하는 여러 가지 방법이 있는데, 여기서는 하나의 정책을 수행함으로써 얻을 수 있는 각 보상에 시간에 따른 차등 비율을 적용하여 합산하는 방법을 사용한다.$\gamma \in [0,1]$는 즉각적인 보상값(immediate reward)이 아닌 지연된 값들을 결정하는 할인상수(discount factor)로 미래의 보상값보다는 현재 즉각적으로 받는 보상값을 더 중요하다는 것을 나타낸다.

MDP에서 에이전트의 목적은 모든 상태 s에 대하여 축적된 보상을 최대로 하는 정책 π를 학습하는 것이다. 그러한 정책을 최적 정책(optimal policy)라 하고 π^*로 나타내며, 축적된 보상은 $V^*(s)$로 나타낸다.

$$\pi^* \equiv \arg\max_{\pi} V^{\pi}(s), (\forall s)$$

$$V^{\pi^*}(s) \Rightarrow V^*(s)$$

그러면 최적의 정책을 에이전트는 어떻게 얻을 수 있을까? 직접적으로 최적의 정책인 π^*를 얻는 것은 어려운 일이다. 왜냐하면 학습 예들이 직접적으로 $<s,a>$의 순서쌍으로 제공되어지지 않고 대신 학습자(learner)는 학습에 필요한 보상들을 받기 때문이다.

따라서 에이전트들은 이러한 함수를 몰라도 환경과의 상호작용을 통해 최적의 정책을 학습할 수 있는 방법이 필요하게 되었다. 어떠한 구체적인 정책을 배우는 것보다는 수치적인 평가함수를 학습하는 것이 쉬우며, 학습할 수치적인 평가함수로 V^*가 있다.

에이전트의 목표는 보다 많은 보상을 얻어내는 것이다. 모든 상태들에 대한 V^*를 알게 되면, V^*에 따라 상태를 변환하게 될 것이며 이러한 변환들이 순차적으로 이루어지면 최적의 정책에 도달하게 된다[5]. 즉, 어떤 $s \in S$에 대하여 다음과 같은 최적의 정책 π^*가 존재할 수 있으며, 상태 s에서의 최적의 행동은 즉각적인 보상값과 뒤따르는 상태들의 V^* 값을 합한 것을 최대화시키는 행동 a로 정의할 수 있다.

$$\pi^*(s) \equiv \arg\max_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

여기서 $\delta(s,a)$는 상태 s에서 행동 a를 취했을 때 옮겨가는 상태를 나타내는 상태 전이 함수(state transition function)이며, r은 즉각적인 보상값을 주는 보상함수(reward function)를 나타낸다. 그러나 위의 학습 규칙은 상태전이 함수인 δ와 보상함수 r을 모두 알아야 유용하다.

이를 해결하기 위한 환경 모델이 불필요한 비모델 강화학습 방법 중의 하나가 Q-Learning이다.

3.2 Q-Learning

Watkins가 제안한 Q-Learning[6]은 가장 널리 이용되고 있는 강화학습 방법으로서 통계적 동적 프로그래밍(stochastic dynamic programming)에 근거한 강화학습이다

Q-Learning은 시간 변화에 따른 적합도 차이를 학습에 이용하는 TD-Learning의 한 종류이며, 환경 모델에 대한 정보가 없이 행동의 적합성을 나타내는 Q값만 학습하므로 실제로 적용할 때 좋은 결과를 보이고 있다. Q값이란 어떤 상태에서 에이전트가 한 행동을 선택했을 때, 발생되는 최적의 누적 보상값이다. 따라서 한 에이전트가 Q함수만 갖는다면, 어떠한 상태에서도 각 행동에 대한 Q값을 통해 수행해야 할 행동을 알 수 있다.

Q-Learning 에이전트는 학습 과정에서 최적의 정책을 결정하는 Q값의 수렴을 위하여 일련의 상태-행동을 선택하는 것이 필요하다. 이 알고리즘의 학습 결정 전략은 각각의 상태-행동 쌍에 대한 장기적인 할인된 보상을 측정하는 상태-행동 가치 함수 Q에 의하여 결정된다.

따라서 상태 공간(state space)에 있는 모든 상태-행동(state-action)쌍을 반복 경험하면서, 평가값인 Q값을 기반으로 하여 환경(environment)이 주는 보상(reward)값을 각각의 상태-행동 쌍에 대한 전략(policy)을 학습한다. 이 때, 모든 상태-행동 쌍의 값을 저장하기 위하여 룩업 테이블(lookup table)을 이용한다. 강화학습에서의 가치 함수는 각 노드가 승리로 이끌어주는 데 얼마나 가치가 있는지를 직접 평가해준다. 물론 이 가치함수는 시행착오를 통해서 만들어 나가야 한다.

Q-Learning의 기본 개념은 다음과 같이 표현될 수 있다.

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$

이 때, $Q(s,a)$는 상태 s부터 시작해서 첫 번째 행동으로 a를 적용하고, 그 후에 최적의 정책을 따라가면서 획득할 수 있는 할인 축적된 보상을 의미한다. 만일 $Q(s,a)$가 구해진다면 최적의 정책을 발견할 수 있게 되는 것이다.

$$\pi^*(s) = \arg\max_a Q(s,a)$$

Q-Learning의 가장 큰 장점은 이렇게 숫자 하나로 에이전트가 처한 상황과 취할 수 있는 행동을 기술할 수 있다는 것이다. Q함수의 값들은 테이블형식의 메모리에 저장하는데, 이런 경우 상당히 많은 상태와 행동들이 존재하는 문제에서 불합리하다. 따라서 신경망(Neural Network) [7]이나 클러스터링(Clustering)을 가미한 다양한 방법들이 사용되기도 한다.

4. Q-Learning을 이용한 n-Queen알고리즘

본 논문에서는 문제풀이 알고리즘으로 비 모델 강화학습 방법 중의 하나인 Q-Learning 알고리즘을 사용한다.

n-Queen 문제에서 가능한 모든 경우의 노드에 대하여 트리를 형성하고, 가치 함수는 모든 노드에 대해 룩업 테이블처럼 대응되어 있다. 가치 함수의 결과인 Q-테이블의 초기 값은 0으로 세팅한 후, 스스로 플레이 과정을 시뮬레이션하기 시작한다. 첫 행에 놓을 여왕은 가치함수와 상관없이 무작위로 선택하고, 다음 행에 놓을 여왕의 자리를 유망한 노드들 중 하나를 선택한다. 이를 반복함으로써 해답을 찾게 되는데 중간에 더 이상 유망한 노드를 발견하지 못하는 경우에는 탐험을 중지하고 Q-테이블 값을 갱신한다. 이렇게 에이전트가 경험하는 지각, 행동, 보상으로 이루어지는 사건 열을 에피소드(episode)라고 하며, 이는 학습을 수행하는 에이전트와 외부 환경과의 상호작용이 자연스럽게 끝나는 것을 의미한다. 여러 번의 에피소드들을 통한 Q값의 갱신은 최적해에 수렴함은 이미 증명되어 있다[8].

Q-Learning 알고리즘은 더 이상 진행할 수 없는 노드에 이른 다음에는 플레이 결과를 플레이 과정에 반영시켜서 가치 함수를 갱신하는 것으로 최종 방문한 노드의 가치 함수 값을 일정 수준(전파 상수)만큼 바로 이전의 노드들로 전파시키는 방법이다.

본 논문에서는 유망한 노드인 경우의 즉각적인 보상값(immediate reward)은 1로 주었으며 할인 상수(discount factor)값은 0.9로 설정하였다.

강화학습은 이와 같이 스스로 작동하게 내버려두지만 원하는 행동을 하면 보상(1)하고 그렇지 않으면 벌(0)을 준다. 이는 많은 시행착오를 시뮬레이션하는 학습 과정이 필요하지만, 점진적으로 성능을 향상시킬 수 있다는 장점을 가지고 있다. 각각의 Q값에 대해 학습하기 위해, Q-Learning 에이전트들은 그 자신의 Q값들을 위해 n개의 Q-테이블($n \times n$)을 유지하며, Q-Learning된 에이전트는 장기적인 행동들에 대한 결과를 예상할 수 있는 능력을 가지게 된다.

다른 종류의 학습에서는 일어나지 않지만 강화학습에서 발생하는 문제들 중 하나는 탐색(exploration)과 이용(exploitation) 사이의 균형 문제이다. 탐색은 새로운 정보를 모으기 위하여 알려지지 않은 상태와 행동을 선택하는 것을 말하며 이용은 이미 학습된 상태와 행동을 취하는 것을 말한다. 이용은 모든 가능한 상태-행동들이 Q-Learning 수렴법칙을 만족하기 위하여 충분히 탐색되는 것을 보장하며, 탐색은 탐욕(greedy) 정책을 적용하고 있다[9]. 보상을 얻기 위해서는 이미 알고 있는 것을 이용해야 하지만, 미래에 더 좋은 행동을 선택하기 위해서는 탐색도 중요하다. 탐색과 이용의 균형은 강화학습에서 매우 중요한 문제 중의 하나이며 여러 가지 요인에 의해 판단되어진다.

본 논문에서는 강화학습에서의 탐색과 이용의 균형 문제를 해결하기 위하여 횟수테이블에 Queen이 놓여진 횟수를 별도로 저장하는 방법을 택하였다.

n-Queen 문제에서는 *n*개의 Queen이 놓여지는 에피소드가 발생함과 동시에 해답을 찾게 된다. n보다 적은 Queen을 놓고도 에피소드가 끝나는 경우에는 실패한 경로가 되므로 다음의 에피소드에서는 이를 피해서 탐색하는 것이 효율적이며, 이는 경로설정을 무작위로 정함으로써 이미 방문한 노드를 계속 방문하는 문제도 또한 피할 수 있을 것이다. 따라서 회수테이블에 저장된 값이 적은 곳으로 탐색하도록 유도함으로써 이 문제를 해결할 수 있다.

다음의 그림 1은 지금까지 설명한 알고리즘을 자바를 이용하여 표현한 것이다.

```
// 해당 행의 queen 의 적당한 위치 찾기
// 찾았을 경우 다시 다음행의 값을 queens (다음행 숫자) 로 다시 재 호출
 public static void queens (int i)
{
  int j;   int jp = 0;  // 이동할 행 위치
  for (j = 1; j <= n; j ++)  // 다음 행에 Queen 을 놓을 위치 찾기
  {
    if (promising (i, j)) {     // 해당 열에 놓을 수 있는지 확인
       if (jp == 0)       // 처음 비교시 기존 위치 기억 값이 0이면
          jp = j;            현재 찾은 행으로 기준을 바꿈

       if (PT [Ts][i][jp] > PT [Ts][i][j])  // 배치 횟수가 적은쪽 선택
          jp = j;
    }
  }
  if (jp != 0) {     // 이동할 위치를 찾았을 거나 마지막 위치까지 왔다면
    coln ++;       // 진행한 행의 값 증가
    col [coln] = jp;     // Queen 배치
    PT [Ts][coln][jp] ++;     // 해당 행, 열의 Queen 배치 횟수 증가
    runcount1 ++;     // Queen 배치 횟수 증가
    if (coln != n)       // 마지막 줄이 아니면, Queen 배치가 덜 끝났을 경우
       queens (coln + 1);     // 다음 행의 Queen 배치
    else {      // 마지막 위치 까지 왔을 경우
       runcount5 ++;
       QTcount ();
    }
  }
  else    // 마지막 위치 까지 오지 못했을 경우. Q 계산
  {
    QTcount ();
  }
 }
```

그림 1. Q-Learning을 이용한 n-Queen 알고리즘

5. 실험 및 평가

본 논문에서 *n*-Queen 문제를 풀기 위하여 제안한 Q-Learning을 이용한 알고리즘은 JDK 1.4.2를 사용하여 자바 언어로 구현하였다.

n-Queen 문제에서의 성능을 평가하는 기준은 n개의 Queen을 놓기 위하여 얼마나 많은 상태전이를 하였는 가이다. 가능한 모든 상태에 룩업 테이블을 이용하여 Q-Learning을 수행하였다. 실험에서 매 번의 실행은 시행의 순열로 이루어져 있다. 각 실행의 첫 번째 시행에서 에이전트는 무작위로 위치를 갖게 된다. 시행 이후에는 보상을 받은 값들을 전파하며 Q-테이블 값들을 갱신하며 시행을 계속한다. 각 시행은 n번째 행에 도달하는 순간 끝나게 된다.

예를 들어 *n*=10일 경우, 정답을 찾기 위해서는 평균 22번의 에피소드가 발생하였으며 평균 143회의 상태 전이가 발생하였다.

본 논문에서는 일반적인 백트래킹과 몬테칼로 기법을 이용한 백트래킹, 그리고 Q-Learning을 이용한 세 가지 방법들에 대하여 최적해를 찾기 위한 상태 전이 횟수를 비교하였다. 그 결과 세 번째 방법을 이용한 실험에서 가장 적은 횟수의 상태전이로 해답을 찾을 수 있었으며, 많은 탐색이 필요한 큰 *n*의 경우 보다 큰 성능의 향상을 보임을 알 수 있었다.

표 1은 각각의 *n*에 대한 세 가지의 실험 결과 발생한 상태 전이의 수를 비교해서 보여주고 있다.

표 1. 상태 전이 횟수의 비교

N	Backtracking-1	Backtracking-2 (Monte Carlo)	Q-learning
4	26	16	8
5	15	16	13
6	171	66	53
7	42	43	39
8	876	171	33
9	333	383	65
10	975	942	143
11	517	1,039	300
12	3,066	1,539	428
13	1,365	5,434	852
14	26,495	16,846	764
15	20,280	14,869	621
20	3,992,510	1,384,727	7,007

다음의 표2는 *n*=10일 때 생성된 Q-테이블의 값들을 보여주고 있다. 예를 들면, 테이블에서 Q(1,1)의 값은 *n*-Queen 테이블의 (1,1)위치에서 2행으로 전이 가능한 모든 상태들에 대한 Q-테이블 값들 중의 최대치를 나타낸다. 표를 분석한 결과, 최대 Q값들의 위치(표에서 색칠한 부분)가 주어진 문제의 해에 수렴함을 알 수 있었다.

이는 Q-Learning의 결과로 얻을 수 있는 근사값이 문제를 해결하는 최적해에 수렴함을 나타낸다. 이에 대한 증명은 [8]에 구체적으로 나타나있다.

표 2. n=10일 때 생성된 Q-테이블의 값

	1	2	3	4	5	6	7	8	9	10
1	4.095	4.686	4.095	4.095	4.686	5.695	6.126	5.695	5.695	4.686
2	5.217	4.686	5.217	5.217	5.695	5.217	5.217	3.439	4.095	5.217
3	4.686	4.686	4.686	4.095	4.686	4.095	3.439	5.217	4.686	0.000
4	3.439	4.686	4.095	4.095	4.095	3.439	4.095	4.095	4.095	4.095
5	3.439	3.439	3.439	1.900	2.710	0.000	3.439	2.710	4.095	0.000
6	1.900	1.000	3.439	2.710	1.900	1.000	1.000	2.710	2.710	2.710
7	0.000	1.000	1.000	0.000	1.900	2.710	1.900	1.900	0.000	1.900
8	1.000	1.000	1.000	1.900	1.000	0.000	1.000	0.000	0.000	0.000
9	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	Goal

6. 결 론

본 논문에서는 *n*-Queen 문제의 해결을 위하여 강화학습 기법을 이용한 알고리즘을 제안하였다. 기존의 알고리즘들과 비교한 결과, 목표에 도달하기 위한 상태전이의 수를 줄여줌으로써 결론적으로는 최적 해에 수렴하는 속도가 상당히 빠름을 알 수 있었다. 이는 *n*의 크기가 커질수록 더욱 효과적이었다.

그러나 일반적인 강화학습 방법들의 가장 큰 문제점은 큰 상태공간을 갖는 보다 복잡한 문제들에 그대로 적용하기 어렵다는 것이다. Q-Learning과 같은 기본적인 강화학습은 문제의 크기가 커짐에 따라 성능이 매우 떨어지게 된다. 왜냐하면, 보상이 원래의 행동이 행해진 상태까지 역전파되어야 하기 때문이다. 또한 문제의 성격에 따라 상태공간이 커지는 문제점이 발생하였다. 큰 상태공간은 보다 많이 탐험을 해야 하며 현재 경험한 상태-행동 쌍의 Q값만을 갱신하므로 학습 속도가 느려진다는 것을 의미한다. 따라서 룩업 테이블에 의한 표현 방식은 적절한 방법이라고 볼 수 없으며, 에이전트가 학습을 위하여 작은 행동 공간을 사용하는 방안에 대한 연구가 필요하고, 또한 속도 개선에 대한 연구와 효율적인 함수 근사 방법에 대한 연구가 이루어져야 할 것이다.

또한 실세계와 비슷한 모형들에 대한 학습을 위하여 주어진 환경에 대한 지식을 학습과정에서 적절히 이용하는 방법과 함께 다양한 형태의 강화학습 알고리즘에 대한 연구가 필요할 것이다.

참 고 문 헌

[1] 장병탁, 이종우, 서영우, "학습 에이전트," 한국정보과학회지, Vol. 18, No. 5, pp. 26-35, 2003.

[2] 이영아, 홍석미, 정태충, "함수근사와 규칙추출을 위한 클러스터링을 이용한 강화학습," 정보과학회 논문지(B), Vol. 30, No. 11, pp. 1054-1061, 2003.

[3] R. E. Neapolitan and K. Naimipour, Foundations of Algorithms, 2nd Ed, Jones and Bartlett Publisher, 1998.

[4] 박찬건, 양성봉, "강화 학습에서의 탐색과 이용의 균형을 통한 범용적 온라인 Q-학습이 적용된 에이전트의 구현," 정보과학회 논문지(B), Vol. 30, No. 7, pp. 672-680, 2003.

[5] 이승준, 장병탁, "탐색 강화 계층적 강화 학습," 정보과학회 제 28회 추계학술대회, Vol. 28, No. 2, pp. 151-153, 2001.

[6] C. J. Watkins and P. Dayan, "Technical note : Q-Learning," Machine Learning, 8, pp .279-292, 1992.

[7] S. Haykin, Neural Network, 2nd Ed, Prentice- Hall, 1999.

[8] T. M. Mitchell, Machine Learning, McGraw-Hill, 1997.

[9] R. S. Sutton and A. G. Barto, Reinforcement Learning : An Introduction. The MIT Press, 1998.

[10] A. Mccallum, K. Nigam, J. Rennie and K. Seymore, "Building domain-specific search engines with machine learning techniques," In AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace, pp. 135-141, 1999.

[11] B. T. Zang and Y. W. Seo, "Personalized Web-Document Filtering Using Reinforcement Learning," Applied Artificial Intelligence, vol. 15, pp. 665-685, 2001.

[12] J. Rennie and A. McCallum, "Using Reinforcement Learning to Spider the Web Efficiently," In proceedings of the 16th International Conference on Machine Learning(ICML-99), pp. 335-343, 1999.

저 자 소 개

임수연(SooYeon Lim)
1988년 : 경북대 전자공학과 (공학사)
1991년 : 경북대 컴퓨터공학과 (공학석사)
2004년 : 경북대 컴퓨터공학과 (공학박사)
2004년~현재 경북대학교 컴퓨터공학과 연구원
관심분야 : 기계학습, 정보검색, 시멘틱웹, 자연어처리

손기준(KiJun Son)
2000년 : 상주대 컴퓨터공학과 (공학사)
2003년 : 경북대 컴퓨터공학과 (공학석사)
2003년~현재 경북대학교 컴퓨터공학과 박사과정

관심분야 : 자연어처리, 기계학습, 정보검색

박성배(SeongBae Park)
1994년 : 한국과학원 전산학과 (공학사)
1996년 : 서울대 컴퓨터공학과 (공학석사)
2002년 : 서울대 컴퓨터공학과 (공학박사)
2004년~현재 경북대학교 컴퓨터공학과 교수

관심분야 : 자연언어처리, 기계학습, 정보검색, 바이오인포매틱스

이상조(SangJo Lee)
1974년 : 경북대 수학교육과 (이학사)
1976년 : 한국과학운 전산학과 (공학석사)
1993년 : 서울대 컴퓨터공학과 (공학박사)
1976년~현재 경북대학교 컴퓨터공학과 교수

관심분야 : 언어처리, 지식처리, 정보검색, 기계학습 시멘틱웹, 온톨로지