Journal of Korea Multimedia Society Vol. 9, No. 5, May 2006(pp. 649-657)

영향력 분포도를 이용한 Q-학습

성연식[†], 조경은[††]

요 약

강화학습이란 환경에 대한 정보가 주어지지 않았을 때 현재의 상태에서 가능한 행동들을 취한 후 얻어지는 보상값이 가장 큰 행동을 최적의 행동 전략으로 학습하는 것이다. 강화학습에서 가장 많이 사용하는 Q-학습은 환경의 특정 상태에서 가능한 행동 중에 하나를 선택해서 취한 행동으로 얻어지는 보상값으로 구성되는데 실세계 상태를 이산값으로 표현하기에는 많은 어려움이 있다. 상태를 많이 정의하면 그만큼 학습에 필요한 시간이 많아지게 되고 반대로 상태 공간을 줄이면 다양한 환경상태를 한 개의 환경상태로 인지를 하고 그 환경에 맞는 한 가지의 행동만 취하도록 학습하기 때문에 행동이 단순해진다. 본 논문에서는 학습 시간을 단축하기 위해 상태 공간을 줄이는 데서 발생하는 행동의 단순화의 단점을 보완하기 위한 방법으로 영향력 분포도를 이용한 Q-학습 방법을 제안한다. 즉, 영향력 분포도와 인접한 학습 결과를 이용해서 학습하지 못한 중간 상태에 적합한 행동을 취하게 하여 동일한 상태 개수에 대해서 학습 시간을 단축하는 것이다. 동일한 학습 시간 동안에 일반적인 강화학습 방법으로 학습한 에이전트와 영향력 분포도와 강화학습을 이용해서 학습한 에이전트의 성능을 비교해 보았을 때 영향력 분포도와 강화학습을 이용해서 학습한 에이전트가 단지 일반적인 강화학습에 필요한 상태공간의 4.6%만 정의를 하고도 성능 면에서는 거의 비슷한 효과를 볼 수가 있음을 확인하였다. 이는 영향력 분포도와 강화학습을 이용한 학습이 일반적인 강화학습에 비해서 학습 속도가 2.77배정도 빨리 이루어지고 실제 학습해야 할 상태 공간의 개수가 적어져서 발생되는 문제를 영향력 분포도를 이용해서 보완을 하기 때문이다.

Q-learning Using Influence Map

Yunsick Sung[†], Kyungeun Cho[††]

ABSTRACT

Reinforcement Learning is a computational approach to learning whereby an agent take an action which maximize the total amount of reward it receives among possible actions within current state when interacting with a uncertain environment. Q-learning, one of the most active algorithm in Reinforcement Learning, is consist of rewards which is obtained when an agent take an action. But it has the problem with mapping real world to discrete states. When state spaces are very large, Q-learning suffers from time for learning. In constant, when the state space is reduced, many state spaces map to single state space. Because an agent only learns single action within many states, an agent takes an action monotonously. In this paper, to reduce time for learning and complement simple action, we propose the Q-learning using influence map(QIM). By using influence map and adjacent state space's learning result, an agent could choose proper action within uncertain state where an agent does not learn. When this paper compares simulation results of QIM and Q-learning, we show that QIM effects as same as Q-learning even thought QIM uses 4.6% of the Q-learning's state spaces. This is because QIM learns faster than Q-learning about 2.77 times and the state spaces which is needed to learn is reduced, so the occurred problem is complemented by the influence map.

Key words: Reinforcement Learning(강화학습), Q-learning(Q학습), Influence Map(영향력 분포도)

※ 교신저자(Corresponding Author) : 조경은, 주소 : 서울시 중구 필동 3가 26 동국대학교(100-715), 전화 : 02)2260-3834, FAX : 02)2265-8742, E-mail : cke@dongguk.edu
접수일 : 2005년 12월 20일, 완료일 : 2006년 3월 6일

[†] 정회원, 모바일잼 대표, 동국대학교 일반대학원 컴퓨터공학과 (E-mail : sung@dongguk.edu)
[††] 종신회원, 동국대학교 정보산업대학 컴퓨터멀티미디어공학과

1. 서 론

게임 세계에는 수많은 에이전트들이 서로에게 영향을 주고 받으면서 새로운 환경에 적응한다. 하지만 각각의 에이전트는 다양한 상태를 가지고 있어서 수많은 에이전트들의 상태를 예측하고 최적의 행동을 결정하는 것은 쉽지 않다. 또한 현재 상태에서 최적의 행동을 취해도 다른 에이전트의 행동에 의해서 그 결과가 달라지고 취한 행동으로 일정 기간 동안에 발생된 보상을 적용하거나 일정 기간이 지난 후에 받은 보상으로 행동에 대한 평가를 내리기가 어렵다. 그래서 게임 세계에서 에이전트가 학습하기 위해서는 수많은 에이전트의 상태를 고려하고 불투명한 변화를 다루는 능력이 필요하다. 그리고 일정 기간 동안에 발생된 보상을 적용하거나 일정 기간이 지난 후에 얻은 보상에 기반한 학습능력을 가진 알고리즘으로 개발하는 것이 적합하다. 학습 알고리즘에는 다양한 방법이 있지만 이중에 강화학습이 이러한 특징을 가장 많이 포함한다[1].

강화학습을 적용하기 위해서 에이전트의 수많은 상태들을 고려하면 이로 인해 학습에 필요한 시간이 길어진다. 강화학습의 성능은 학습시간에 비례해서 상태의 수가 많아지면 필요한 학습시간을 확보하기에는 많은 어려움이 발생하기 때문에 결국 성능 저하로 이어진다. 따라서 본 연구에서는 게임 시스템과 같이 다수의 에이전트가 각각의 판단에 따라 선택하는 행동들을 예상하고 자신의 행동을 선택해야 하는 게임 응용분야에 적합한 강화학습 알고리즘을 제안한다.

이 논문에서 사용한 강화학습은 상태를 줄여서 학습 시간을 단축하고 이로 인해서 감소되는 학습 효과를 영향력 분포도를 이용해서 증진시키는 방법이다.

또한, 실세계 상태값은 대부분 연속적인값을 가지게 되지만 학습을 위해 이산 상태값으로 표현하면서 많은 상태를 한 가지 상태로 인지하고 학습하기 때문에 학습 효과가 떨어지게 된다. 학습된 결과를 이용해서 행동을 취할 때 이산 상태 중간값에 해당하는 상태에서 행동을 취할 때 적절하지 못한 행동을 취하는 경우가 빈번하게 발생된다. 따라서 본 논문에서는 인접한 이산 상태값과 영향력 분포도를 이용하여 학습하지 못한 중간 상태에 가장 적절한 행동을 계산해서 최상의 행동을 선택하여 학습효과를 높일 수 있음을 보였다.

본 논문의 구성은 다음과 같다. 2장에서는 강화학습과 영향력 분포도에 대한 소개한다. 강화학습의 대표적인 알고리즘인 Q-학습에 관한 간략한 이론을 정리하고 Q-학습이 가지고 있는 문제점과 그것을 해결하기 위한 기본연구들을 소개한다. 그리고 영향력 분포도에 관련된 이론들을 정리하고 적용하기 위한 간단한 방법을 기술한다. 3장에서는 영향력 분포도를 이용하여 이산 상태값의 중간값을 생성하는 방법에 대해서 기술한다. 그리고 일반적인 Q-학습과 영향력 분포도를 이용한 Q-학습의 차이점을 들어서 비교한다. 4장에서는 본 논문의 실험 방법과 결과에 대해 기술한다. 강화학습을 위한 상태와 행동을 정의하는 방법과 실험 방법을 설명하고 실험 결과를 정리하고 분석한다. 마지막으로 5장에서는 실험 결과에 대한 결론을 내리고 향후 연구 과제를 제시한다.

2. 관련 연구

본 논문에서는 Q-학습의 학습 효과를 증가시키기 위해서 영향력 분포도를 이용하고 있는 데 이번 장에서는 본 논문에서 사용한 강화학습의 대표적인 알고리즘인 Q-학습과 영향력분포도를 소개한다.

2.1 강화학습

강화학습이란 환경에 대한 정보가 주어지지 않았을 때 현재의 상태에서 가능한 행동들을 취한 후 얻어지는 보상값이 가장 큰 행동을 최적의 행동 전략으로 학습하는 것이다.

이 중 Q-학습[2]은 대표적인 강화 학습으로 Watkins가 처음으로 제안했다. Q-학습은 취한 행동을 평가하기 위해서 가치함수를 사용하는데 수식으로 표현하면 다음과 같다.

$$V^{\pi}(s_t) \equiv r_t + \gamma r_{t+1} + \cdots = \sum \gamma^i r_{t+i} \quad (1)$$

식 (1)에서 가치함수 $V^{\pi}(s_t)$는 t시간에 발생되는 상태s_t에서 행동Π를 이용했을 때 평가되는 값으로 정의를 할 수가 있다. 가치함수 공식을 살펴보면 t는 시간을 나타내고 γ은 시간이 지나감에 따른 보상의 감소율을 나타낸다. γ의 값 범위는 $0 \le \gamma < 1$인데 1보다 작은 이유는 만약에 감소율이 1인 경우에는 발생된

보상값이 계속 영향을 미치기 때문이다. r은 해당하는 시간t에 평가되는 보상값이 된다. 결국 가치함수는 보상값의 합으로 이루어지는데 보상값은 시간에 따라 계속 감소하게 된다. 이 공식을 이용해서 구한 보상값은 Q-값을 계산하기 위해서 사용된다. 일반적인 Q-값은 다음과 같이 갱신된다.

$$Q(s,a)\leftarrow Q(s,a)+\alpha[\ r + r \max Q(s',a') - Q(s,a)\ ] \quad (2)$$

식 (2)에서 α는 학습률을 나타내고 0에서 1사이의 값을 가진다. 학습률이 0이 되면 전혀 학습을 하지 않는다. r은 시간이 지나감에 따른 보상의 감소율을 나타내고 r은 보상값을 나타낸다. Q(s',a')은 s상태에서 a행동을 취할 때 s'상태로 전이가 되는데 이 상태에서 a'행동을 취할 때의 Q-값이다. Q-학습은 다음에 이어질 행동을 고려하지 않고 독립적으로 Q-값이 결정되기 때문에 알고리즘이 단순해지고 빨리 원하는 값에 수렴하게 된다[3]. 하지만 다양한 문제들도 제시된다. 첫째, 상태를 이산 상태로 구분하므로 실세계 상태를 이산 상태로 표현하기 때문에 발생하는 문제점이 있다. 이러한 문제점을 해결하기 위해 H. Aljibury는 국소 최소 자승법(Locally Weighted Regression)[4]을 이용한 Q-학습 성능향상 방법을 제시하였다. 국소 최소 자승법은 두 이산 상태 중간 Q-값을 추정할 수 있는 함수를 만들어서 실세계를 이산 상태로 표현하기 때문에 발생되는 문제를 보완한다. 이 함수를 이용하여 서로 인접한 두 Q-값의 중간값을 세밀하게 구할 수 있다. 그러나 추정할 수 있는 함수를 만들 때 지형적인 요소를 함수에 반영하지 않고 전체 환경을 하나로 일반화하여 함수를 만들기 때문에 게임처럼 환경이 하나의 전략요소가 될 만큼 중요한 경우에는 적합하지 않다. 또한 중간값을 추정하기 위해서 좀 더 많은 요소들을 고려하고 값을 결정하는 것이 좋겠지만 그러한 요소들을 모두 고려해서 하나의 공식으로 만들기에는 어렵고 복잡한 공식을 매번 계산해서 값을 얻어내는 것도 시스템에 부담이 될 수 있다. 이 논문에서는 학습이 끝나고 모든 Q-값을 일괄적으로 각각 계산한 후 이용한다. 그렇기 때문에 학습된 결과물을 이용하는 데 있어서 매번 복잡한 계산을 하고 그 값을 이용하는 것보다 훨씬 빠른 시간을 기대할 수 있다. 또한 Q-값을 각각 계산해서 추후에도 더 많은 요소들을 고려할 수 있다.

둘째, Q-학습은 모든 상태와 행동에 대해서 올바른 의사 결정을 할 때까지 학습되어야 하기 때문에 학습에 걸리는 시간이 길어지게 된다. 이를 보완하기 위해서 분포 기여도[5]를 이용해서 학습시간을 단축하기 위한 방법이 제시되었다. 보통 Q-학습에서 하나의 상태에 대한 하나의 행동값만 학습한다. 분포 기여도에서는 학습 결과가 반영된 Q-값의 주변 Q-값에도 영향을 미치게 하여 학습에 필요한 시간을 줄여주고 있다. 보완하기 위한 다른 방법은 상태를 처음에 적게 정의하고 상태 개수를 학습할 때 동적으로 증가하는 방식[6]이 제기되기도 하였다. 하지만 실세계에서는 많은 상태들이 존재하기 때문에 위의 방법보다 학습이 필요한 상태-행동의 경우를 좀 더 효율적으로 줄일 방법이 필요하다.

학습에 걸리는 시간 문제는 상태 공간을 줄여서 쉽게 해결 할 수 있지만 상태 공간을 줄일수록 학습 효과가 떨어지는 현상이 발생된다. 그래서 이 논문에서는 상태 공간을 줄여도 학습효과가 떨어지지 않는 방법을 제시한다. 각 환경상태에는 최적의 행동이 존재한다. 그런데 상태를 줄이게 되면 다양한 환경상태를 한 개의 환경상태로 인지하고 그 환경에 맞는 한 가지 행동만 취하게 된다. 그렇기 때문에 상태공간을 줄이면 행동이 단순해져서 결국 학습 효과가 낮아지게 된다. 하지만 3장에서 기술할 영향력 분포도와 주변의 학습된 정보를 이용한다면 학습된 최적의 행동보다 연속적인 값을 가지는 실세계에 더욱 적합한 행동을 찾아서 취하기 때문에 향상된 성능을 보여주게 된다.

2.2 영향력 분포도

영향력 분포도란 지형에서 존재하는 여러 사물이나 에이전트들이 지형에 미치는 요소를 모두 표현해서 각각의 지형 위치에 따른 전략이나 전술적인 위치를 결정하는 방법이다.

이 논문에서는 두 개의 Q-값의 중간값을 유추하기 위해서 영향력 분포도를 사용했다. 중간의 새로운 Q-값을 생성하기 위해서 주위에 있는 각각의 Q-값이 미치는 영향력 정도를 가지고 값을 결정하게 된다.

일반적으로 영향력 분포도를 적용하기 위해서는

다음과 같은 과정이 수행되어야 한다. 첫째, 지형을 격자 무늬로 나눈다. 이렇게 나누어진 하나의 격자 단위가 지형을 분석하기 위한 기본 단위로 사용된다. 둘째, 지형에 영향을 미치는 사물이나 에이전트를 선정한다. 물론 지형에 영향을 미친다면 무형의 어떠한 것도 가능하다. 지형 위에는 다양한 요소들이 존재하겠지만 의사결정을 위해서는 이 많은 요소들 중에 필요한 요소들을 선별해야만 한다. 셋째, 두 번째에서 결정한 요소들이 얼마나 지형에 영향을 미치는 지 결정하고 마지막으로 각각의 영향력이 어떻게 주변에 영향을 미치는 지를 결정해야 한다. 이렇게 만들어진 영향력 분포도를 이용하면 데이터 안에 감춰져 있는 어떠한 유형이나 경향성을 파악할 수 있게 된다. 게임 장르 중에서는 전략 게임에 주로 사용이 되는 데 특히 영향력 분포도를 이용해서 지형 및 에이전트의 배치 유형을 파악하고 이를 이용해서 에이전트가 이동할 위치를 결정하거나 공격할 대상을 선택하게 된다. 현재 영향력 분포도에 관련된 연구는 영향력 분포도 값을 의사 결정에 사용에 하기 위한 방법론, 영향력 분포도 값을 갱신하기 위한 방법론 그리고 지형을 고려한 영향력 분포도 적용 방법론에 관한 것이 진행되고 있다.

3. 영향력 분포도를 이용한 Q-학습

이 논문에서는 학습을 마친 Q-테이블을 이용해서 두 개 이상의 Q-값을 가지고 중간값을 생성해서 학습 효과를 높일 수 있는 방법을 제안한다.

실세계를 잘 표현하기 위해서 많은 상태로 나누어서 학습하면 좋지만 상태가 많아지면 학습에 필요한 시간이 길어지게 된다. 그래서 이 논문에서는 학습이 끝난 후에 학습된 결과를 가지고 영향력분포도를 이용해서 상태를 확장하는 방법을 소개하려고 한다.

영향력 분포도를 이용해서 Q-학습 성능을 높이기 위해서는 일반적인 Q-학습 방법에 비해서 추가적으로 몇 가지 고려해야 한다. 영향력 분포도를 이용해서 Q-테이블이 확장되기 때문에 Q-테이블의 크기를 결정하는 상태와 행동의 개수는 확장된 이후의 개수를 고려해서 정의한다. 추가적인 작업으로는 정의한 Q-테이블을 가지고 실험하는 방법은 똑같지만 그림 1에서 나타난 것처럼 영향력 분포도를 적용해서 Q-테이블의 크기를 확장하는 작업이 필요하다.

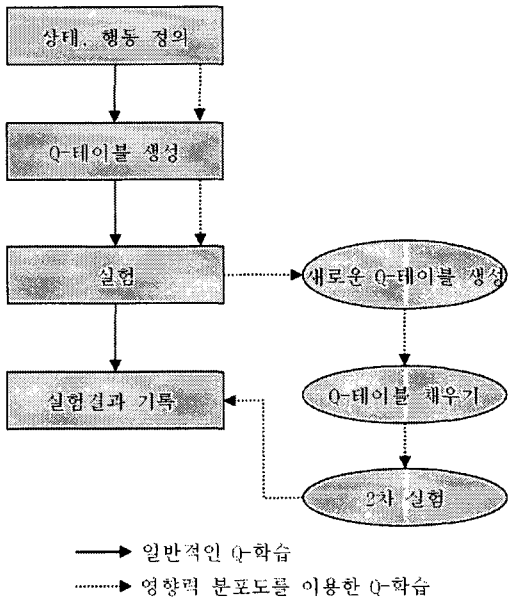


그림 1. 영향력 분포도를 이용한 Q-학습 순서도

확장하는 첫 번째 단계로 만들어진 Q-테이블을 어느 정도의 크기로 확장시킬 것인지를 결정하고 생성한다. 둘째, 학습한 Q-테이블을 새로 결정한 크기의 Q-테이블에 복사하고 중간값을 영향 분포도로 채운다. 이렇게 만들어진 Q-테이블을 에이전트의 의사결정에 사용함으로써 일반 Q-학습과 성능이 얼마나 차이가 나는 지 비교해 보았다.

그림 2는 Q-테이블의 Q-값을 영향력 분포도를 이

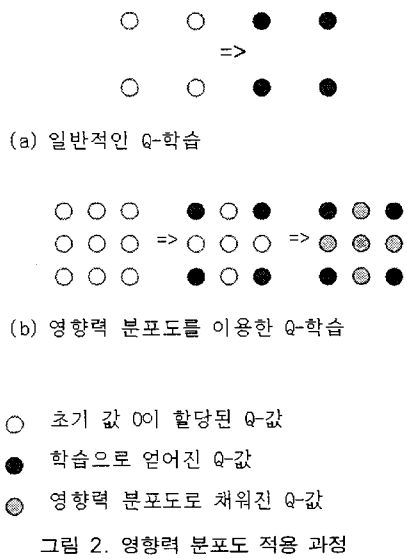


그림 2. 영향력 분포도 적용 과정

용해서 확장하는 과정을 일반적인 Q-학습과 비교해서 전체 Q-테이블 중에 일부만 나타낸 것이다. 일반적인 방법으로 학습한 (a)에서는 학습을 통해서 모든 Q-값이 결정되지만 영향력 분포도를 이용한 (b)에서는 Q-값의 일부는 학습으로 얻어지고 일부는 영향력 분포도를 이용해서 채워진 것을 볼 수 있다. (b)의 2번째 단계는 일반적인 Q-학습에서 얻은 Q-값을 새로 만든 Q-테이블에 복사하는 과정을 나타내 것이고 3번째 단계는 채우지 못한 Q-값을 영향력 분포도를 이용해서 채운 것이다.

4. 실 험

이번 장에서는 본 논문에서 제시한 방법의 성능을 알아보기 위한 실험에 대해서 소개한다. 실험을 위한 시스템 설계 방법과 이를 구현한 방법에 대해서 기술한다. 마지막으로 실험 결과를 일반적인 Q-학습과 영향력 분포도를 이용한 Q-학습의 성능을 비교한다.

4.1 시스템 설계

실험에서 총 3개의 에이전트를 이용하는데 2개의 에이전트가 한 팀이고 남은 한 개의 에이전트 혼자가 다른 2개의 에이전트와 싸운다. 에이전트의 목적은 상대편 에이전트를 공격해서 상대편의 모든 에너지를 소멸시키는 것이다.

각 에이전트는 9 개의 행동을 취할 수가 있는데 8 개의 방향으로 이동하는 것과 공격 행동이다. 에이전트는 행동을 결정하기 위해서 다음과 같이 3 가지를 고려한다.

① 상대 팀 에이전트의 위치
② 같은 팀 에이전트의 위치
③ 자기 자신의 위치

에이전트가 좀 더 나은 의사 결정을 하기 위해서는 좀 더 많은 상태를 Q-학습에 적용하는 것이 좋겠지만 표 1에서 보듯이 고려해야 할 상태 변수 개수가 증가할 때마다 Q-테이블의 크기는 급격히 증가하게 된다. 첫번째 열은 에이전트의 위치를 25 가지로 표현한 경우에 상태 변수 개수 증가에 따른 Q-테이블의 크기를 나타낸 것이고, 두번째 열은 에이전트의 위치를 9 가지로 표현한 경우에 상태 변수 개수 증가에 따른 Q-테이블의 크기를 나타낸 것이다. 이 논문에서는 상태변수를 3 개만 선택해서 에이전트가 의사결정을 한다.

표 1. 상태 변수 개수와 Q-테이블 크기의 관계

상태 변수 개수	25개의 위치상태	9개의 위치상태	비율
1	225	81	36.00%
2	5625	729	12.96%
3	140625	6561	4.67%
4	3515625	59049	1.68%
5	87890625	531441	0.60%
6	2197265625	4782969	0.22%
7	54931640625	43046721	0.08%

Q-학습에서 각각의 에이전트 위치를 표현하기 위해서 픽셀단위를 이용하면 상태가 너무 많아져서 Q-테이블의 크기가 많이 커지게 된다. 그래서 전체 경기장을 25개의 동일한 크기로 분할했다. 그렇게 되면 에이전트의 위치는 25가지 상태 중 하나의 상태로 정의되기 때문에 필요한 Q-테이블의 크기는 25 * 25 * 25 * 9 = 140625개가 된다. 반면에 영향력 분포도를 이용한 Q-학습은 상태를 좀 더 줄여서 경기장을 9개의 동일한 크기로 분할해서 필요한 Q-테이블의 크기는 9 * 9 * 9 * 9 = 6561개가 된다.

그림 3에서 볼 수 있듯이 구장을 세부적으로 나누어서 상태가 늘어남에 따라 Q-테이블의 크기는 급격하게 증가하게 된다. 상태를 25 가지로 정의한 것이 9 가지로 정의를 한 것에 비해서 이번 실험에서는 약 21 배 많은 Q-테이블의 크기가 필요하다. 즉 9 가지 상태로 정의를 한 Q-테이블이 25개의 상태로 정의한 Q-테이블에 비해 4.6%정도의 공간만 차지한다.

각각 행동의 보상값은 자신이 상대 에이전트에게 준 타격과 자신이 입은 타격으로 계산되는데 공식으로 표현하면 다음과 같다.

$$R_i = H_{i1}^3 - D_{1i} * 2 + \cdots + H_{in}^3 - D_{ni} * 2$$
$$= \sum H_{in}^3 \quad \sum D_{ni} * 2 \qquad (3)$$

여기서, H_{in}는 i번째 에이전트가 n번째 에이전트에 준 타격값이고 D_{ni}는 n번째 에이전트가 i번째 에이전트에게 준 타격값이다. 즉 자신이 입은 타격값이다.

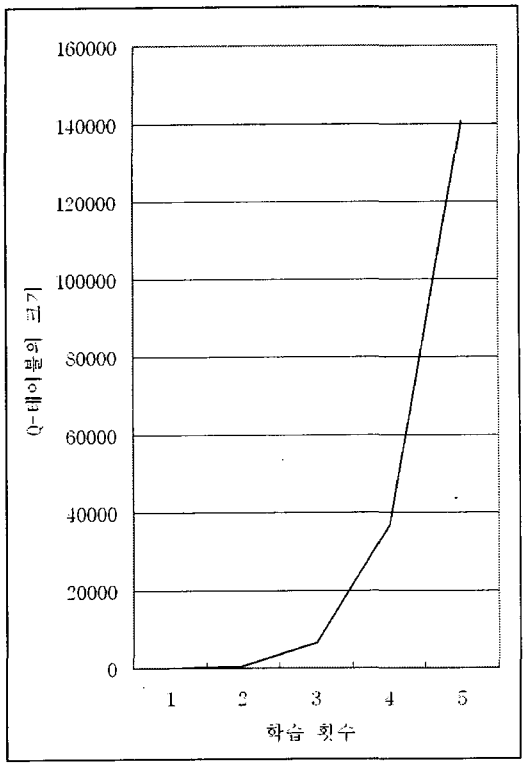


그림 3. 상태공간의 증가에 따른 Q-테이블의 크기

공식을 만들 때 에이전트가 공격적인 성향을 보이도록 하기 위해서 타격을 준 값이 타격을 입은 값보다 많을 때 더 많은 보상을 받도록 했다.

표 2에서 보듯이 보상값을 계산을 할 때 두 개의 에이전트가 동시에 상대 에이전트를 공격할 때 가장 높은 보상값이 나오도록 했다. 서로 타격을 1씩 주고받을 때에 0보다 작은 보상값이 나오도록 해서 에이전트가 동시에 상대 에이전트를 공격하도록 공식을 만들었다.

부과적으로 에이전트의 현실성을 반영하기 위해서 일정 영역 안에 들어오는 에이전트만 인지 할 수 있도록 했다.

표 2. 공격력과 보상값의 관계

사용자 팀 공격력	컴퓨터 팀 공격력	보상값
0	1	-2
0	0	0
1	1	-1
1	0	1
2	1	6
2	0	8

4.2 시스템 구현 및 실험 방법

본 논문은 Java 1.4로 테스트 환경을 구현하였다. 각 에이전트는 매 턴마다 자신이 취할 행동을 결정하고 행동한다. 경기장은 100*100 크기를 가지도록 구성하였고 에이전트는 40*40의 크기를 가진다.

실험은 크게 2가지 방법으로 진행했다. 첫째, 일반적인 Q-학습 방법을 이용해서 실험했다. 에이전트의 위치 정보를 25 개의 상태로 표현하고 에이전트가 취한 행동에 대한 보상값을 Q-테이블에 저장한다. 실험은 한 경기를 진행하는데 시간이 적게 걸리기 때문에 한번의 경기로 충분한 학습을 하기에는 부족하다. 그래서 1900번의 경기를 진행하고 이를 10단계로 나누어서 각 단계별로 학습 효과를 관찰했다.

둘째, 에이전트의 위치 정보를 9 개의 상태로 표현하고 Q-학습을 시작한다. 위와 동일하게 10 단계로 학습시키고 각 단계 끝마다 만들어진 Q-테이블을 영향력 분포도를 이용해서 확장하고 새로 만들어진 Q-테이블을 이용해서 성능을 평가한다. 새로운 Q-테이블은 실험을 통해서 얻은 Q-테이블의 Q-값과 영향력 분포도로 얻은 새로운 값을 이용해서 채운다. 그래서 2가지 방법을 이용한 실험 결과를 비교하였다.

그림 5는 영향력 분포도를 이용해서 Q-테이블을 새로 만들기 위한 구현 코드이다. 변수 QTable은 학습을 통해서 얻은 Q-값을 저장한 Q-테이블이고 변수 NewQTable은 영향력 분포도를 이용해서 값을 채울 Q-테이블이다. 변수 NewQTable의 크기는 변수 QTable의 크기에 중간값을 채우기 위한 크기를

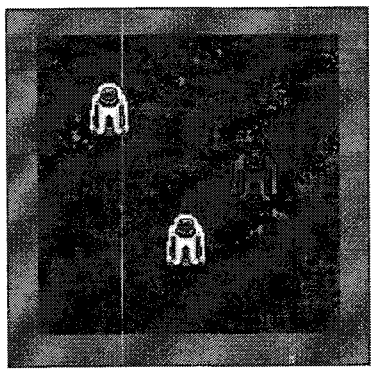


그림 4. 실험화면

```
public void makeInfluenceMap() {

   double NewQTable[];

   SetSize( NewQTable,//새로운 Q-테이블 크기 결정
      QTABLE_SIZE+INTERMEDIATE_POINT_COUNT );

   InitQTable( NewQTable );//새로운 Q-테이블 초기화

   for( int i=0;//새로운 Q-테이블의 모든 Q-값에 대해서
      i< QTABLE_SIZE+INTERMEDIATE_POINT_COUNT;
      i++ )
   {
      if( IsIntermediatePoint(i) )
      {//중간값을 계산해야 할 경우
         //영향력 분포도를 적용
         NewQTable[i] = GetIntermediateValue( QTable ,i );
      }
      else
      {//Q-값을 그대로 적용할 경우
         NewQTable[i] = GetQValue( QTable,i );
      }
   }
   QTable = NewQTable;
}
```

그림 5. 영향력 분포도 코드

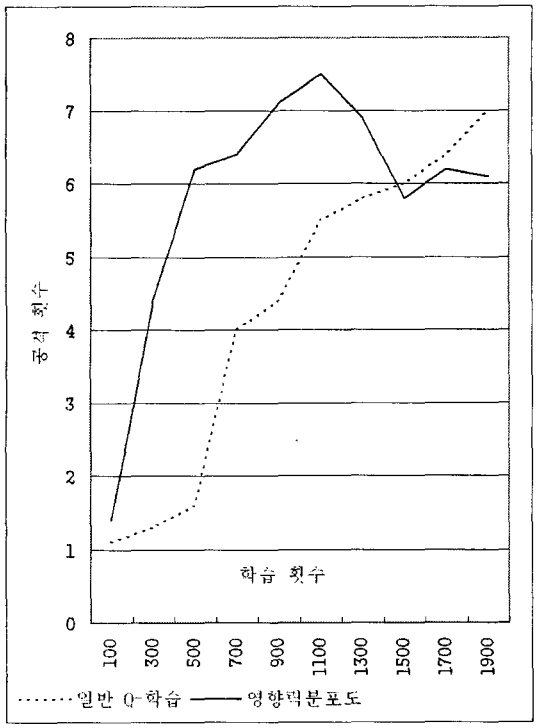


그림 6. 학습 횟수에 따른 평균 공격 횟수

합한 크기로 생성한다. 소스를 보면 새로운 테이블에 값을 채우기 위해서 2개의 함수를 이용하는 데 첫번째는 GetIntermediateValue함수를 이용해서 영향력 분포도를 이용한 중간값을 계산해서 가져오는 것이고 두번째는 기존 Q-테이블에 저장된 값을 그대로 가져오는 것이다.

4.3 실험 결과 및 분석

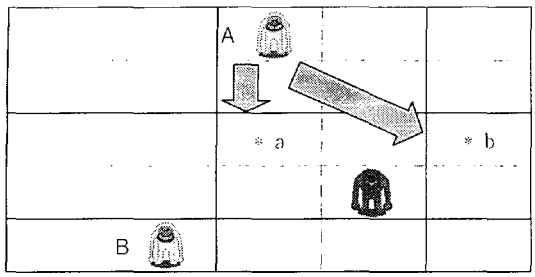
두 개의 실험 조건을 다시 정리해보면 실험은 동일한 시간 동안 에이전트의 학습이 이루어 졌고 영향력 분포도를 이용한 Q-학습이 일반적인 Q-학습에 비해4.6% 크기의 Q-테이블을 가지고 학습했다.

그림 6과 표 3은 학습 횟수에 따른 평균 공격 횟수를 실험한 결과이다. 그래프를 보면 초반에는 비슷한 효과를 보이지만 1500 번 이상의 학습이 이루어졌을 때에는 상태공간을 더 세분적으로 구분한 일반적인 Q-학습이 좀 더 좋은 성과를 보였다. 반면에 영향력 분포도를 이용한 Q-학습에서는 일정 수준의 성능에 도달하면 더 이상 성능이 향상되지 않는 것을 볼 수가 있었다.

표 3. 평균 공격 횟수

학습 횟수	영향력 분포도	일반
100	1.1	1.4
300	1.3	4.4
500	1.6	6.2
700	4.0	6.4
900	4.4	7.1
1100	5.5	7.5
1300	5.8	6.9
1500	6.0	5.8
1700	6.4	6.2
1900	7.0	6.1

그림 7을 보면 A위치에 있는 에이전트 입장에서 B에이전트와 동시에 상대 에이전트를 공격하기 위해서 학습한 결과, 2곳이 동시 공격이 가능하고 각각의 Q-값이 43.054와 156.859이 되었다. 표 4는 일반적인 Q-학습을 할 때 Q-테이블의 Q-값을 표현한 것인데 이를 이용해서 에이전트가 움직인다면 에이전트는 A위치에서 두 번 내려와서 동시 공격 가능 a지점에서 B에이전트와 동시에 공격을 할 것이다. 이는



* 협동 공격이 가능한 위치

그림 7. Q-테이블 크기 비교

표 4. Q-학습으로 얻은 Q-테이블 값

-1	-1	-1
0	43.054	156.8594
0	0	0

표 5. 영향력 분포도를 적용한 Q-테이블 값

-1.000	-1.000	-1.000	-1.000	-1.000
-0.500	10.264	21.027	49.478	77.930
0.000	21.527	**43.054**	99.957	**156.859**
0.000	21.527	21.527	49.978	78.430
0.000	0.000	**0.000**	0.000	**0.000**

동시 공격하기에 더 좋은 b지점이 있음에도 불구하고 Q-테이블의 크기가 작아서 생기는 현상이다. 하지만 표 5를 보면 영향력 분포도를 이용해서 확장한 Q-테이블을 이용할 경우에 A위치에서 큰 Q-값을 따라 이동할 경우에 동시 공격 가능 b지점에서 동시 공격을 할 수 있게 되어서 더 좋은 결과가 나타내게 된다.

실험 결과를 통해서 볼 수 있듯이 영향력 분포도를 이용해서 Q-테이블의 크기를 늘린다면 동일한 학습 시간 동안에 학습을 수행할 때 기존 방법에 비해서 4.6%의 상태공간만을 차지하지만 성능면에서는 거의 비슷한 효과를 볼 수가 있었다. 이는 표 6와 그림 8에서 볼 수 있듯이 하나의 상태 공간이 일반적인 방법에 비해서 영향력을 이용한 학습이 2.77배 정도 빨리 학습하기 때문이다.

표 6은 실험이 모든 상태 공간에서 균일하게 학습이 진행된다고 가정할 때 학습 횟수가 증가함에 따라 각 상태 공간의 Q-값이 갱신되는 횟수를 나타낸 것이다. 그림 8는 표 6를 다이어그램으로 나타낸 것이다. 이처럼 한 개의 Q-값에 대해서 평균 학습 횟수가

표 6. 한 개의 Q-값에 대한 평균 학습 횟수

실험	학습 횟수	일반	영향력 분포도
1	225	9	25
2	450	18	50
3	900	36	100
4	1800	72	200
5	3600	144	400
6	7200	288	800
7	14400	576	1600
8	28800	1152	3200
9	57600	2304	6400
10	115200	4608	12800

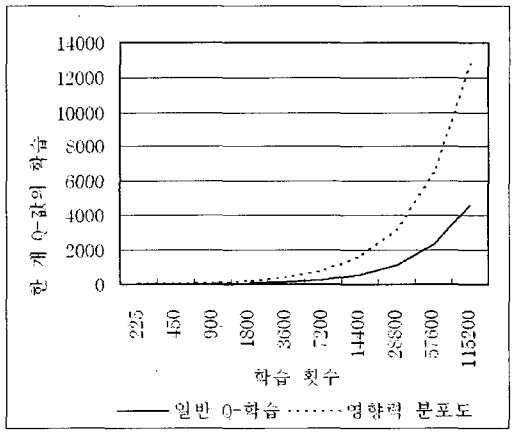


그림 8. Q-값의 학습량

차이가 나는 것은 일반적인 Q-학습의 Q-테이블 크기가 영향력 분포도를 이용한 Q-학습의 Q-테이블 크기의 2.77배 정도 크기 때문에 그에 따라 학습에 필요한 실험횟수도 증가되기 때문이다.

5. 결론 및 향후 연구방향

본 논문에서는 실험을 통해서 얻은 Q-테이블을 영향력 분포도를 이용해서 확장시킬 때 에이전트의 성능을 향상시킬 수 있음을 보였다. 실험을 통해서 동일한 학습 시간에 일반적인 Q-학습에 비해서

4.6%의 상태공간만 정의하고도 영향력 분포도를 이용해서 그 성능을 향상시켜 비슷한 효과를 나타낼 수 있음을 보였다.

본 논문에서 제시한 방법으로는 학습시간이 많이 걸리는 경우에 적합하고 기존에 학습된 결과물의 성능을 향상 시킬 수 있는 방법으로도 사용이 될 수 있다. 특히 게임처럼 수많은 에이전트가 다른 에이전트와 전투가 일어날 때의 상황을 학습하기 위해서는 무수히 많은 실험이 이루어져야 하는 데 이러한 경우에 영향력 분포도를 이용한다면 학습 시간을 단축할 수 있기 때문에 더 많은 상황에서의 학습이 가능해진다.

Q-학습의 중간 Q-값을 생성하기 위한 많은 연구들이 있었는데 이 논문을 통해서 영향력 분포도와 주위의 학습된 Q-값을 이용해서 중간값을 생성하는 방법을 제시했다.

학습된 Q-테이블을 영향력 분포도를 이용해서 무한대로 증가 시킬 수는 없을 것이라고 생각된다. 향후 논문에서는 학습된 Q-테이블과 확장된 Q-테이블의 크기에 대한 성능 분석이 이루어져야 할 것이다. 그리고 지형을 고려한 영향력 분포도의 확장 방법에 관한 연구도 진행되어야 할 것이다.

Q-학습은 게임 시스템에 적합한 많은 요소를 가지고 있지만 앞에서 언급한 단점 때문에 아직까지도 상용화 게임 시스템에 널리 사용되지 못하고 있다. 이 논문을 시작으로 Q-학습을 게임에 적용하기에 부적합한 요소들을 하나씩 해결해 나아가고 마지막으로 게임에 적합한 Q-학습을 제시하고자 한다.

참 고 문 헌

[1] Peter Stone, *Layered Learning in Multi-Agent System*, PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1998.

[2] C. J. C. H. Watkins and P. Dayan, "Technical Note : Q-learning," *Machine Learning*, Vol. 8, pp. 279-292, 1992.

[3] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning : An Introduction*, The MIT Press, Cambridge, MA., 1998.

[4] H. Aljibury, *Using Locally Weighted Regression to Enhance Q-learning*, University of Florida, 2002.

[5] 정석일, 이연정, "분포 기여도를 이용한 퍼지 Q-learning," 퍼지 및 지능시스템학회 논문지 2001, Vol. 11, No. 5 pp. 388-394.

[6] Yasutake Takahashi, Minoru Asada, and Koh Hosoda, "Reasonable Performance in Less Learning Time by Real Robot Based on Incremental State Space Segmentation," *Proc. of IEEE/RSJ International Conf. on Intelligent Robots and Systems*, pp. 1518-1524, 1996.

[7] Paul Tozour, *Influence Mapping*, in Game Programming Gems 2, Mark Deloura (ed), Charles River Media, 2003.

성 연 식

1998. 1~2001. 5 (주)케이원시스템 대리
2004. 4~2005. 2 (주)포켓스페이스 팀장
2004. 8 부산대학교, 전자전기정보컴퓨터공학(공학사)
2004. 9 ~현재 동국대학교 일반대학원 컴퓨터공학과 석사과정
2005. 9~현재 : 모바일잼 대표
관심분야 : 게임 인공지능, 소프트웨어공학

조 경 은

1993. 2 동국대학교, 전자계산학과(공학사)
1995. 2 동국대학교, 컴퓨터공학과 대학원(공학석사)
2001. 8 동국대학교, 컴퓨터공학과 대학원(공학박사)
2003. 3~2003.8 영산대학교 게임공학과 전임강사
2003. 9~현재 동국대학교, 정보산업대학 컴퓨터멀티미디어공학과 조교수
관심분야 : 컴퓨터 게임 알고리즘, 게임 인공지능, 컴퓨터 그래픽스, 멀티미디어 정보처리