Journal of Korea Game Society 2020 Dec; 20(6): 23-30
http://dx.doi.org/10.7583/JKGS.2020.20.6.23

게임 프로그래밍

강화학습에서 점진적인 심화를 이용한 고누게임의 개선

신용우
동아방송예술대학교 창의융합교양학부
ywshin@dima.ac.kr

Improvement of the Gonu game using progressive deepening in reinforcement learning

YongWoo Shin
Division of Creative Convergence Education,
Dong-Ah Institute of Media and Arts

요 약

게임에서는 많은 경우의 수들을 가지고 있다. 그래서 학습을 많이 하여야 한다. 본 논문은 학습속도를 개선하기 위하여 강화학습을 이용했다. 그러나 강화학습은 많은 경우의 수들을 가지므로 학습 초기에 속도가 느려진다. 그래서 미니맥스 알고리즘을 이용하여 학습의 속도를 향상하였다. 개선된 성능을 비교하기 위해 고누게임을 제작하여 실험하였다. 실험결과는 승률은 높았지만, 동점의 결과가 발생하게 되었다. 점진적인 심화를 이용하여 게임트리를 더 탐색하여 동점인 경우를 줄이고 승률이 약 75% 향상되었다.

ABSTRACT

There are many cases in the game. So, Game have to learn a lot. This paper uses reinforcement learning to improve the learning speed. However, because reinforcement learning has many cases, it slows down early in learning. So, the speed of learning was improved by using the minimax algorithm. In order to compare the improved performance, a Gonu game was produced and tested. As for the experimental results, the win rate was high, but the result of a tie occurred. The game tree was further explored using progressive deepening to reduce tie cases and win rate has improved by about 75%.

Keywords : Gonu game(고누게임), Minimax algorithm(미니맥스 알고리즘), Q Learning (Q러닝), Learning speed(학습속도), Progressive deepening(점진적인 심화)

Received: Sep. 20. 2020. Revised: Nov. 13. 2020.
Accepted: Nov. 15. 2020.
Corresponding Author: YongWoo Shin
(Dong-Ah Institute of Media and Arts)
E-mail: ywshin@dima.ac.kr

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

2018년도 국내 게임 시장의 규모는 14조 2,902억원 수준으로, 이는 2017년도의 13조 1,423억원 대비 8.7%가 증가한 것이다. 국내 게임 산업은 전반적으로 성장세를 지속하고 있다[1].

게임 인공지능 기법을 적절히 적용한다면, 게임에서 새로운 기능들을 추가하게 된다. 특정 목적이 달성되면 보상을 하는 과정을 반복하다 보면 에이전트가 학습을 하게 된다[2].

기존의 인공지능에서 지능을 다룬 논문으로는, 오델로 등을 다루었지만[3,4] 고누게임을 연구해 보고자 한다.

논문[5]에서도 Q러닝(Q-learning)을 이용하여 고누 게임을 구현했다. 오델로 게임에서는 말의 위치가 결정되면 고정되지만, 고누게임은 말이 계속 이동한다는 점에서 구현하기가 어렵다. Q러닝 알고리즘을 사용한 논문[5]에서는, 학습의 초기에는 학습된 자료의 부족으로 인하여 학습속도가 느려졌다.

본 논문은 학습 초기에 최선의 값을 산출하지 못할 때 미니맥스 알고리즘(Minimax algorithm)을 이용해서 최선의 값을 부여하도록 하는 방법을 제안하였다. 동점 상황이 발생하였을 때는, 동점으로 기록하게 된다. 그러나 미니맥스 알고리즘을 더 탐색한다면 승리를 추가할 수가 있다. 실험결과, 동점으로 처리할 때보다 승률이 추가되었다.

본 논문은, 1장에서는 서론을 논하며 2장은 관련 연구, 3장은 지능형 보드게임의 구현을 논한다. 4장에서는 실험 및 결과이다. 마지막으로 5장은 결론이다.

2. 관련 연구

2.1 Q러닝

Q러닝의 환경에는 많은 상태가 존재한다. 에이전트는 목적을 달성하려면 상태를 경험해야 한다. 시행착오를 많이 겪으면서, 모든 상태 들을 경험하여야 한다.

Q러닝은 사전에 모델을 설정하지 않고, 상태공간을 충분하게 경험하면 된다.

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_{t+1} + \gamma \max_{a} Q(s_{t+1},a') - Q(s_t,a_t))$$

[Fig. 1] Q-Learning Algorithm

상태 s에서의 행위인 a에 대한 보상으로 r을 받는다[2]. t는 이전의 시간이고 t+1은 이후의 시간을 말한다. max란 최대값이고, α는 학습속도의 매개변수로서 0부터 1사이이며 γ는 할인계수이며 0부터 1사이이다.

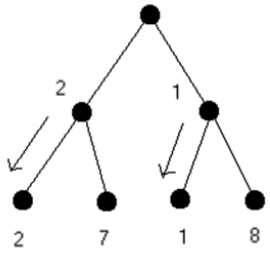
모든 상태공간의 특정한 상태 s의 행위에서 가장 적당한 보상 값인 r을 저장하며 산출한다. 처음에는 모든 상태들 각각에 대해 보상을 얻기 위하여 무작위로 행위를 해서 경험을 쌓는다. 그러나 경험된 상태가 많으면, 보상을 기반으로 불필요한 경험은 줄어든다. 모든 상태를 경험하면 지능적인 동작이 가능하다.

2.2 미니맥스 알고리즘

플레이어(Player) 각각의 하나의 행위를 수(Move) 라 한다. 양의 숫자는 한 플레이어에겐 이익으로, 음의 숫자는 상대방에게 이익이 있는 것으로 가정한다[6].

[Fig. 2]의 아래쪽에 있는 노드와 루트노드(Root node)가 최대레벨(Maximum level)이고, 가운데에 있는 노드는 최소레벨(Minimum level)이다. [Fig. 2]에서 단말노드들의 값 중에서 최소값은 최소레벨에서 택하여야 하고 루트노드에서는 최대값을 택한다.

이렇게 두 수의 앞을 내다봄으로써 플레이어는 적절한 값을 갖는 경기를 하게 된다.



[Fig. 2] Minimax algorithm

2.3 점진적인 심화 (Progressive deepening)

토너먼트(Tournament)에서 플레이어들은 시계가 요구하는 제한된 시간 동안에 몇 번을 움직여야 한다. 어느 일정한 깊이를 탐색하는데 걸리는 시간은 상황에 따라 다르므로 위의 경우 문제가 된다. 진행되는 경기에 관계없이 일정한 깊이까지의 탐색은 그 고정된 깊이에서 보수적인 안전한 선택을 요구하게 된다. 그렇지 않으면 경기는 시간초과로 지게 된다.

이런 문제를 해결하는 방법으로는 깊이 1까지 먼저 각 상황을 분석한 후 깊이 2까지 그리고 깊이 3까지 차례로 시간이 다 될 때까지 탐색해 나가는 것이다. 이 방법에서는 항상 수를 둘 준비가 되어 있다. 시간이 다 소모될 때 진행되어 분석된 깊이보다 한 깊이 얕은 레벨에서의 분석에 의해 선택이 결정된다.

3. 지능형 고누게임의 구현

3.1 알고리즘

상대방말은 개선된말 방향으로 이동한다. 상대방말의 위치에서 유리한 곳으로 한 칸을 이동한다. 즉, 개선된말 하나가 상대방말 옆에 인접되었다면, 상대방말은 개선된말을 공격하려고, 직선이 되는 지점으로 이동한다. 또한, 실수로 상대방말의 직선상에 들어갔다면 상대방말이 승리하게 된다. 상대방말의 경우에도, 개선된말의 직선상에 들어오면 개선된말이 승리하게 된다 [8,9].

학습에서의 상태는 초기에는 0으로 저장되어 모두 같은 값으로 되어 있다. 보상값이 많이 저장되지 않아서 대부분 초기값들을 가지고 있다. 이때는 무작위로 그중에서 하나의 값을 선택한다. 그래서 적당한 위치로는 이동하지 못한다. 상대방에게 공격당하거나 올바르지 않은 방향으로 움직인다.

이런 단점을 없애려고, [Fig .3] 의 알고리즘으로, 같은 값들이 나올 때, 미니맥스를 고려하고 점진적인 심화로 유리한 곳으로 움직이도록 알고리즘을 개선하였다. 강화학습에서의 상태는 계속되는 게임의 결과로, 보상값들이 누적된다. 누적된 값에 의해, 개선된말에게 좋은 값을 선택한다.

```
procedure Q_learning
{
    Discover Max value of QTable
    If Q values are same,
        Put on apply minimax algorithm
            with Progressive deepening
    Choose player's action from QTable
    Move player's character

    Move enemy's character due to e_action()
    If (catch)
        Give plus reward
        If (be captured) Give minus reward
    Update QTable
}
```

[Fig. 3] QMM_learning Algorithm

개선된말이 이동할 방향을 알고리즘이 결정하여 상대방말을 공격하게 된다. 학습의 초창기라면 보상값이 누적되지 않아 동일한 초기값들을 갖고 있다. 그러므로 개선된말이 진행할 방향을 미니맥스 알고리즘에서 정한 곳으로 상대방말을 공격한다.

3.2 평가함수

고누게임을 구현하기 위해서는, 각각의 상황에서, 여러 가지 가능한 수 중에서 몇 수를 내다보는 것이 나에게 유리한 것인지를 평가하여야 한다. 평가된 값 중에서 현재 나에게 유리한 값을 판단하려면 평가함수(Evaluation function)를 만들어야 한다.

이 평가는 게임에 영향이 있는 여러 종류의 특성에 근거한다[7]. 승리할 가능성이 있는 방법으로 평가함수를 제안하여야 한다. 현재 상황에서 이동 가능한 모든 말에서 상대방의 말을 공격 가능한 경우와 그 다음 수에서 공격 가능한 경우를 말한다. 평가함수의 공식은 [Fig. 4] 와 같다.

$$Ef = \sum_{i=1}^{N} (e_i * w_i + t_i) - \sum_{j=1}^{N} (e_j * w_j + t_j)$$

[Fig. 4] Evaluation function

상대방의 말을 공격 가능한 경우는 가중치를 곱한다. 공격할 가능성이 있을 경우는 더한다. 개선된말일 때는 양의 값으로, 상대방말일 때는 음의 값으로 계산한 값을 평가값으로 한다.

Ef 는 평가함수이다. N은 게임에서 가능한 모든 수를 말한다. e_j는 상대방의 말이 공격한 경우이다. e_i는 상대방의 말을 공격한 경우이다. w는 가중치이다. 다양한 값을 가질 수 있고, e를 w에 곱하였으므로 e와 t의 상대적인 중요성으로 결정된다.

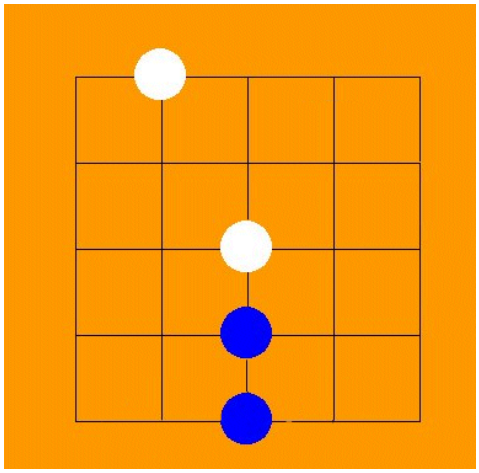
4. 실험과 결과

4.1 실험환경

개선된 말과 상대방말은 각각 둘이다. [Fig. 5]의 경우는, 상대방말 둘이 직선으로 있고, 바로 옆에 개선된말이 인접해 있으면 공격당하게 된다. 직선상을 벗어나면 방어가 된다.

개선된말 또는 상대방말의 위치는 각각의 상황의 학습된 점수가 저장된다. 개선된말이 움직일 때는, 선택 가능한 수 중에서 유리한 수를 선택하게 된다. 그러나 학습된 초기에는 저장된 자료가 적으므로 좋은 선택을 하지 못하게 된다.

상대방말의 경우에는, 초기에 개선된말을 쉽게 공격한다. 학습된 다음에는, 개선된말도 쉽게 공격한다.



[Fig. 5] State in the instance of catched between opponents

4.2 실험결과

[Table 1] Experiment outcomes in advanced reinforcement learning

Play / Learn	Win 1	Win 2	Lose 1	Lose 2	Tie
1000	162	5	0	0	27
2000	338	9	0	0	50
3000	511	17	0	0	73
4000	653	20	0	0	105
5000	815	26	0	0	133

[Table 1] 에서의 Win1과 Win2는 미니맥스 알고리즘으로 학습된 말이 이긴 경우이다. Lose1과 Lose2는 상대방말이 이긴 경우이다. Win1과 Win2가 있는 것은 이겼을 경우와, 실수로 상대방말이 직선상에 들어와서 이긴 경우이다. Win1과

Lose1은 정상적인 경우이고, Win2와 Lose2는 실수인 경우를 말한다. 학습한말과 상대방말의 실험결과는 차이가 있다.

학습속도에서 본다면, 개선된말은 학습의 초기에서부터 승리횟수에서 앞선다. 패배의 경우에는 정상적이거나 실수로 진 경우가 한 번도 없었다. 계속 실험을 해도, 개선된말의 패배횟수는 0이며 개선된말의 승리가 증가할 것이다.

미니맥스를 적용하여 학습속도가 개선되었다. 결과적으로 승률이 향상되었다. 그러나 일반적인 미니맥스알고리즘이 일정한 깊이(depth) 까지 탐색하기 때문에 무승부가 발생하게 되었다. 무승부를 해결하기 위해서는 점진적인 심화 알고리즘이 추가적으로 필요하다.

[Table 2] Experiment outcomes of previously reinforcement learning

Play / Learn	Win 1	Win 2	Lose 1	Lose 2	Tie
1000	24	8	80	1	0
2000	89	27	132	1	1
3000	266	80	138	1	3
4000	462	135	138	1	6
5000	675	186	138	1	7

[Table 1]은 미니맥스알고리즘이 적용된 실험결과이고 [Table 2]는 기존의 Q러닝으로 학습한 결과이다. [Table 1]은 학습속도의 개선으로 항상 승률이 높지만, [Table 2]는 초기에 학습데이터의 부족으로 승률이 낮다. 3000회 이후부터 승리횟수가 높아진다.

개선된말은 [Fig. 3]의 알고리즘으로 이동하도록 설계되었으며, 상대방말은 개선된말의 위치를 파악하여, 최단거리로 이동하여 공격한다. [Table 2]의 결과는 개선된말 또는 상대방말이 1회씩 수를 둔 것이 1000회가 되었을 때의 승부를 표로 작성한 것이다. 총 1000 에서 113회의 승부가 결정되었다. 그중에서 개선된말이 승리한 대국이 32회, 상대방말이 승리한 대국이 81회이다. 1회의 승부가 결정되면, 초기위치로 돌아가서 다시 대국이 이어진다.

[Table 3] Experiment outcomes in advanced reinforcement learning without tie

Play / Learn	Win 1	Win 2	Lose 1	Lose 2	Tie
1000	209	6	0	14	0
2000	424	10	0	28	0
3000	629	18	0	42	0
4000	829	24	0	60	0
5000	1043	28	0	75	0

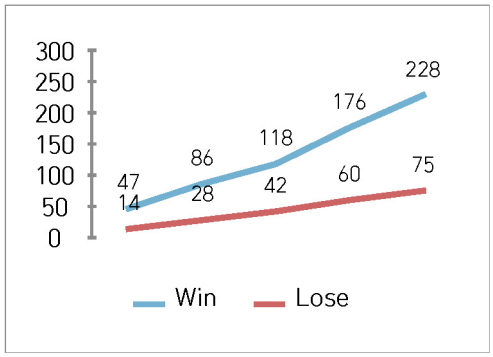
[Table 1]에서 동점을 허용하였을 때의 경우보다 [Table 3]에서의 경우를 보면 승리횟수가 증가하였다. 그러므로 동점일 때, 점진적인 심화로 탐색을 더 진행한 경우는 많은 승리를 하게 된다.

개선된말은 승리횟수는 1000회부터 상대방말을 앞선다. 이후로도 승리횟수는 커다란 차이를 나타낸다. 패배의 경우에는 작은 범위에서 소량으로 증가한다. 그러므로 Q러닝의 학습초기에, 보상값이 누적되기 전에는 학습되지 않는다는 단점을 많은 승리횟수로 보완하는 것이다.

[Table 4] Win percentage in advanced reinforcement learning

Play / Learn	Win	Lose	Rate
1000	47	14	77 %
2000	86	28	75 %
3000	118	42	74 %
4000	176	60	75 %
5000	228	75	75 %

[Table 4]는 동점이 발생했을 때, 그 동점을 점진적인 심화로 탐색을 더 진행시켜서 나온 승패결과를 의미한다. 1000회에서부터 5000회까지 승리횟수가 앞선다. 평균적으로 75% 의 승률이 전체적인 승률에 추가되었다.



[Fig. 6] In case of a Tie, Win Rate

미니맥스 알고리즘이 정해진 깊이만큼 탐색하여 경기할 때, 동점인 상황이 만들어졌었다. 이때 다음 수를 탐색하여 경기하여 승리한 것이다. 이 실험이 4방향이 아니라 8방향, 또는 그보다 다양하게 움직인다면, 더 좋은 승률을 나타내지 않았을까 한다.

5. 결 론

일반적인 강화학습에서는 보상값이 누적되어 저장되기 전에는 정확한 결과값을 알 수가 없다. 그러므로 학습 초기에는 동일한 결과값이 산출되었을 때 무작위로 선택한다.

그러나 단점을 개선하기 위해 같은 값들이 나올 때, 미니맥스 알고리즘을 활용하였고, 학습속도가 향상되었다. 또한 패배가 없었다.

기존의 학습으로 움직이는 방법과 개선된 방법을 적용한 실험결과를 비교하였다. 실험결과, 미니맥스를 활용한 방법의 패배횟수는 0 이다. 그러나 패배횟수는 없었지만 무승부의 횟수가 증가하였다.

무승부의 횟수를 줄이기 위하여 미니맥스 알고리즘에 점진적인 심화를 적용하였다. 그 결과 무승부는 없어지고, 승률이 약 75% 향상되었다.

REFERENCES

[1] Korea Creative Content Agency, “White Paper On Korean Games 2019”, 2019
[2] Richard Sutton, Andrew G. Barto, "Reinforcement Learning :An Introduction", MIT Press, Cambridge, MA, 1998.
[3] Imran Ghory, "Reinforcement learning in board games.", available at http://www.cs.bris.ac.uk /Publications/Papers/2000100.pdf, 2004.
[4] Nee Jan van Eck, Michiel van Wezel., "Reinforcement Learning and its Application to Othello", available at http://www.few.eur.nl /few/people/mvanwezel/rl.othello.ejor.pdf, 2004
[5] Yongwoo Shin, “Artificial Engine Development through Reinforcement Learning on Jul-Gonu Game ”, Journal of Internet Computing and Services, Vol 10, No 1, pp93-99, 2009
[6] Patrick Henry Winston, "Artificial Intelligence", Addison Wesley, 1993
[7] Sukin You, “Artificial Intelligence Fundamentals”, Kyohaksa, 1988
[8] Woosung Sim, “50 traditional games korean folk play”, Nonghyup, 1996
[9] Woosung Sim, “Korean folk play”, Dongmoonsun, 1996
[10] Steve Woodcock, "Game AI : The State of the Industry", Game Developer Magazine, 2000.
[11] Steve Rabin, AI Game Programming Wisdom 2, Charles River Media, 2003
[12] Steve Rabin, AI Game Programming Wisdom, Charles River Media, 2002

신 용 우 (YongWoo Shin)

약 력 : 2014 경희대학교 컴퓨터공학과 박사
1990-2000 프리랜서 게임프로그래머, LG U+
2000-현재 동아방송예술대학교 교수

관심분야 : 게임인공지능, VR/AR/MR게임