게임 인공지능에 사용되는 강화학습 알고리즘 비교

김덕형 · 정현준*

군산대학교

Comparison of Reinforcement Learning Algorithms used in Game AI

Deokhyung Kim · Hyunjun Jung*

Kunsan National University

E-mail : {kdh9477, junghj85}@kunsan.ac.kr

요 약

강화학습에는 다양한 알고리즘이 있으며 분야에 따라 사용되는 알고리즘이 다르다. 게임 분야에서도 강화학습을 사용하여 인공지능을 개발할 때 특정 알고리즘이 사용된다. 알고리즘에 따라 학습 방식이 다르고 그로 인해 만들어지는 인공지능도 달라진다. 그러므로 개발자는 목적에 맞는 인공지능을 구현하기 위해 적절한 알고리즘을 선택해야 한다. 그러기 위해서 개발자는 알고리즘의 학습 방식과 어떤 종류의 인공지능 구현에 적용되는 것이 효율적인지 알고 있어야 한다. 따라서 이 논문에서는 게임 인공지능 구현에 사용되는 알고리즘인 SAC, PPO, POCA 세 가지 알고리즘의 학습 방식과 어떤 종류의 인공지능 구현에 적용되는 것이 효율적인지 비교한다.

ABSTRACT

There are various algorithms in reinforcement learning, and the algorithm used differs depending on the field. Even in games, specific algorithms are used when developing AI (artificial intelligence) using reinforcement learning. Different algorithms have different learning methods, so artificial intelligence is created differently. Therefore, the developer has to choose the appropriate algorithm to implement the AI for the purpose. To do that, the developer needs to know the algorithm's learning method and which algorithms are effective for which AI. Therefore, this paper compares the learning methods of three algorithms, SAC, PPO, and POCA, which are algorithms used to implement game AI. These algorithms are practical to apply to which types of AI implementations.

키워드

Reinforcement learning, Algorithms, SAC, PPO, POCA

Ⅰ. 서 론

최근 다양한 분야에서 인공지능을 구현하기 위해 강화학습을 사용하고 있으며 이는 게임 분야에서도 마찬가지이다. 기존의 게임 인공지능 구현 방법인 FSM(Finite State Machine)[1]은 인공지능이 적용될 객체가 취할 행동과 행동이 전이될 조건을 코드로 구현해야하기 때문에 개발자는 객체의 행동을 구현할 때 행동 조건을 명확히 인지하고 코드로 구현할 수 있어야 한다. 즉, 인공지능이 적용될 객체의 행동 조건을 명확히 알고 있어야 구현이 가능했다. 반면, 강화학습은 객체가 취할 수 있는 행동과 행동에 대한 보상을 설정해주면 상태에 적합한 행동을 스스로 학습한다. 행동이 전이될 조건도 스스로 학습하기 때문에 개발자가 행동 전이 조건을 코드로 구현할 필요가 없다. 그림 1은 강화학습의 아키텍처이다. 에이전트(Agent)는 학습의 주체이며 환경(Environment)과 상호작용을 한다. 환경은 에이전트에게 상태(State) 정보를 주고 에이전트는 해당 상태에서 취한 행동(Action) 정보를 환경에게 전달한다. 환경은 에이전트가 취한 행동에 맞는 보상(Reward)을 에이전트에게 준다. 에이전트는 학습을 진행하며 보상을 최대화 하는 행동 패턴을 찾아간다.

* corresponding author

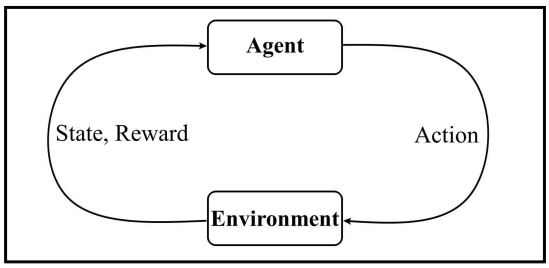

그림 1. 강화학습 아키텍처

강화학습에는 아키텍처를 기반으로 한 여러 알고리즘이 있으며 적용되는 분야에 따라 사용되는 알고리즘이 달라진다. 게임 인공지능 구현에 사용되는 알고리즘은 종류에 따라 인공지능이 학습하는 방식이 달라지므로 게임의 장르, 객체의 역할, 객체의 개수 등에 따라 알고리즘의 효율이 달라진다. 따라서 개발자는 인공지능이 적용될 게임의 장르와 객체의 특성에 따라 알고리즘을 다르게 선택할 필요가 있다. 효율적인 알고리즘을 선택하려면 사용하려는 알고리즘의 학습 방식과 특성을 인지하고 있어야 한다. 따라서 이 논문에서는 강화학습을 사용한 게임 인공지능 구현에서 사용할 수 있는 SAC(Soft Actor-Critic)[2], PPO(Proximal Policy Optimization)[3], POCA(Posthumous Credit Assignment)[4] 알고리즘의 학습 방식을 소개하려 한다. 또한, PPO, SAC, POCA 알고리즘이 어떤 목적을 가진 인공지능을 구현할 때 사용되는 것이 효율적인지 제안하려 한다.

II. 관련 연구

Unity의 예제 게임인 3DBall에서 연속적인 행동 패턴을 찾을 때 선택적으로 PPO 알고리즘을 사용할 수 있다[5]. PPO 알고리즘이 연속적인 환경에서 제대로 작동하는 것을 보여준다.

강화학습을 이용한 카오틱 시스템 제어에선 SAC 알고리즘을 사용하여 연속적인 행동 패턴을 학습했다[6]. 연속적인 행동 패턴을 찾는 환경에서 SAC 알고리즘을 사용하여 빠르고 안정적으로 최적 행동 패턴을 찾았다.

III. ML-Agents 알고리즘 소개

표 1은 강화학습 알고리즘의 속성을 보여준다. SAC, PPO, POCA 알고리즘의 기반이 되는 알고리즘과 환경 유형, 그리고 인공지능을 구현하기에 특화된 게임을 보여준다.

표 1. 강화학습 알고리즘의 속성

알고리즘 / 속성	SAC	PPO	POCA
기반 알고리즘	off-policy	on-policy	주의망
환경 유형	연속 행동 환경	이산/연속 행동 환경	이산/연속 행동 환경
특화 게임	레이싱, 슈팅게임	RPG, 퍼즐게임	스포츠, 보드게임

SAC 알고리즘의 학습 방식은 Off-Policy[7] 기반이다. Off-Policy는 위험구역을 사람이 직접 탐색하지 않고 드론을 이용하는 방식과 비슷하다. 사람이 직접 위험 구역을 탐사하면 최대한 위험 요소를 배척하며 이동경로를 결정하겠지만, 드론을 이용하면 안전한 길보단 빠른 길, 즉 효율적인 경로를 탐색할 것이다. 따라서 SAC 알고리즘은 후에 서술할 PPO 알고리즘보다 최적 행동 패턴을 찾는 시간이 빠르다. 하지만 SAC 알고리즘은 연속적인 행동(Continuous Action) 결정이 필요한 환경에서만 제대로 작동한다. 연속적인 행동이란 행동에 대한 상세값을 설정할 수 있는 행동을 뜻한다. 전진, 후진, 회전의 세 가지 행동이 있다고 한다면, 연속적인 행동 결정에서는 [전진0.3, 후진0, 회전0.5]와 같이 상세값을 설정할 수 있어 좀 더 세밀하고 다양한 행동 결정이 가능하다. 그림 2는 연속적인 행동의 예시이다. 자동차를 운전하는 게임에서 조이스틱을 사용하면 조이스틱의 기울기를 통해 행동의 상세값을 설정할 수 있다. 즉, 연속적인 행동을 취할 수 있기 때문에 반원으로 표시된 범위를 세밀하게 움직일 수 있다.

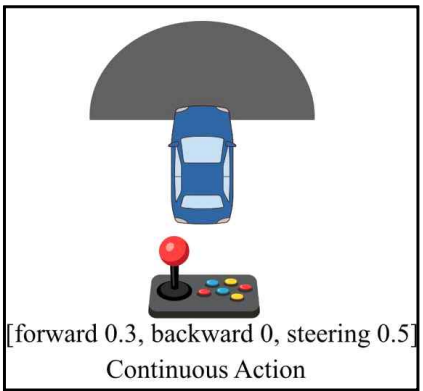

그림 2. 연속적인 행동의 예시

연속적인 행동 결정은 레이싱 게임이나 슈팅 게임과 같이 세밀한 행동 결정이 필요한 게임에서 사용된다. 따라서 레이싱 게임과 슈팅 게임과 같이 연속적인 행동 결정이 필요한 게임 장르에서는 SAC 알고리즘을 사용해 인공지능을 구현하는 것이 유리하다.

PPO 알고리즘의 학습 방식은 On-Policy[8] 기반이다. On-Policy란, 사람이 직접 위험구역을 탐사하는 방식과 같다. 만약 사람이 위험구역을 탐사한다면 최대한 위험 요소를 배척하며 이동경로를 결정할 것이다. 즉, 빠른 길보단 안전한 길을 선택한다는 것이다. 따라서 On-Policy 기반의 PPO 알고리즘은 위험 요소와 가까운 최적 행동 패턴을 찾는 문제에선 최적 행동 패턴을 찾는 시간이 SAC 알고리즘보다 길어진다. 그림 3은 위험 요소가 최적 경로 가까이에 있는 환경이다. 'S'로 표시된 출발점에서 'G'로 표시된 목적지로 이동하는 것이 목표이며, 한 칸씩 이동할 때마다 보상(R)을 1점 잃으며 'The Cliff'로 표시된 영역에 빠지게 되면 보상(R)을 100점 잃는다. 그림 2의 환경에서는 최대한 적게 이동하면서 위험 영역을 피해 G의 목적지로 이동하는 경로가 최적 경로(Optimal Path)가 된다. 하지만 PPO 알고리즘은 위험 요소를 최대한 배척하기 때문에 위험 요소와 가장 멀리 떨어진 경로인 비교적 안전한 경로(Safer Path)를 최적 경로로 인식한다. 최적 경로를 찾을 때까지 학습을 지속하면 결국 최적 경로를 찾긴 하지만 그만큼 학습 시간이 늘어나게 된다.

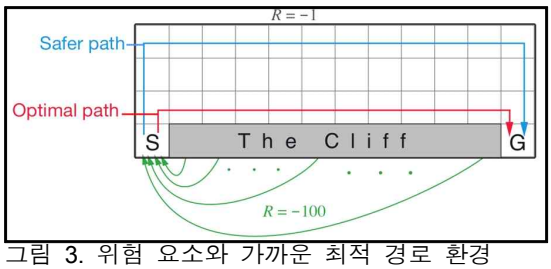

그림 3. 위험 요소와 가까운 최적 경로 환경

SAC 알고리즘과 달리 PPO 알고리즘은 연속적인 행동(Continuous Action) 결정이 필요한 환경뿐만 아니라 이산적인 행동(Discrete Action) 결정이 필요한 환경에서도 사용할 수 있다. 이산적인 행동이란 행동에 대한 상세값을 설정할 수 없는 행동을 뜻한다. 그림 4는 이산적인 행동의 예시이다. 이산적인 행동 결정에서는 행동을 [전진O, 후진X, 회전O]와 같이 결정할 수 있어 선택할 수 있는 행동의 수가 적다. 자동차를 운전하는 게임에서 키보드를 사용하면 버튼의 입력으로만 행동을 결정하기 때문에 행동의 상세값을 설정할 수 없다. 즉, 이산적인 행동을 취하기 때문에 자동차는 화살표 방향대로만 움직일 수 있다.

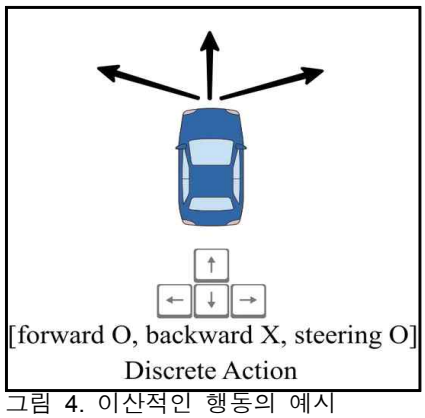

그림 4. 이산적인 행동의 예시

이산적인 행동 결정은 주로 RPG(Role Playing Game)나 퍼즐 게임과 같이 단순한 행동으로만 이루어진 게임에서 사용된다. 따라서 RPG나 퍼즐 게임과 같이 이산적인 행동 결정이 필요한 게임 장르에서는 PPO 알고리즘을 사용하여 인공지능을 구현하는 것이 유리하다.

POCA 알고리즘은 2021년에 Unity ML-Agents에서 개발한 알고리즘이며 에이전트 간의 협력이 필요한 환경에서 원활한 학습을 지원한다. POCA 알고리즘은 기존의 알고리즘에 주의망(Attention network)[9] 개념을 접목시켜 에이전트가 각자의 상황에 맞게 학습할 수 있도록 설계된 알고리즘이다. 주의망은 에이전트가 환경 전체가 아닌 자신의 기준에서 가장 효율적으로 보상을 획득할 수 있는 행동에 집중할 수 있게 해준다. 그림 5는 총알 피하기 게임의 화면이다. 총알은 랜덤한 개수가 생성되며 에이전트(Agent)는 총알을 피하며 최대한 오래 버텨야한다. 주의망을 적용하지 않으면 에이전트는 생성된 총알 전체에 집중하기 때문에 안전한 경로를 찾기 힘들다. 하지만 주의망을 적용하면 충돌 궤적에 있지 않은 총알은 무시하고 충돌 궤적에 있는 총알에 집중(Attention)하여 현재 에이전트 자신의 상황에서 안전한 경로를 쉽게 찾을 수 있다.

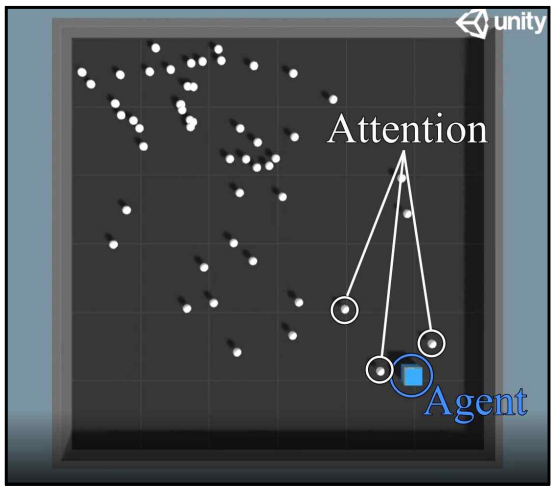

그림 5. 총알 피하기 게임의 화면

PPO, SAC 알고리즘은 다수의 에이전트 간의 협력이 필요한 환경에서 사용하기에 어려움이 있다. 그 이유는 PPO, SAC 알고리즘은 하나의 게임 환경에 다수의 에이전트가 있을 때, 보상을 에이전트끼리 서로 공유한다. 이렇게 되면 다수의 에이전트 중에서 하나의 에이전트만 올바른 행동을 취하면 다른 에이전트들은 아무런 행동을 취하지 않아도 보상을 얻게 된다. 아무런 행동을 취하지 않는 행동 패턴을 최적 행동 패턴으로 인식할 위험이 있는 것이다. 또한, 환경에 존재하는 모든 에이전트들이 항상 보상을 가장 많이 얻을 수 있는 행동을 취하기 때문에 역할의 분담이 어렵다. 하나의 환경에 획득하면 보상을 얻는 오브젝트 A, B, C가 존재하고 보상은 각각 10점, 5점, 1점이라고 한다면 게임에서 보상을 획득하는 가장 효율적인 방법은 세 개의 에이전트가 각각의 오브젝트를 하나씩 획득하는 것이다. 하지만, PPO, SAC 알고리즘을 사용하면 보상을 가장 많이 얻을 수 있는 오브젝트에 집중하기 때문에 세 개의 에이전트가 모두 A, B, C 순서로 오브젝트를 획득하려 한다. 즉, 비효율적으로 학습된다. POCA 알고리즘을 사용하면 위에 서술한 주의망을 통해 각 에이전트가 각자의 상태에서 가장 효율적으로 보상을 얻을 수 있는 행동을 찾는다. 그로 인해 각 에이전트가 자신과 가까운 오브젝트를 획득하게 된다. 이처럼 POCA 알고리즘은 다수의 에이전트 간의 협력이 필요하거나 각자 다른 역할을 맡아야하는 환경에서 유리하다. 협력이 필요한 게임은 대표적으로 스포츠 게임이 있고, 객체간의 협력이 필요한 게임은 보드게임이 있다. 따라서 스포츠 게임이나 보드 게임과 같이 협력이 필요하거나 객체마다 다른 역할을 수행해야 하는 게임에서는 POCA 알고리즘을 사용하여 인공지능을 구현하는 것이 유리하다.

Ⅳ. 결론 및 향후 과제

이 논문에서는 강화학습을 이용한 게임 인공지능 구현에 사용되는 SAC, PPO, POCA 알고리즘의 학습 방식과 특징을 소개하고 어떤 인공지능 구현에 사용되는 것이 효율적일지 이론적으로 제안했다. 강화학습을 통해 인공지능을 구현할 때 좀 더 효율적인 인공지능 구현이 가능할 것이다.

향후에는 실제 게임 환경에서 SAC, PPO, POCA 알고리즘을 사용해 강화학습 인공지능을 구현하며 학습 과정과 구현된 인공지능의 효율을 분석하고자 한다.

References

[1] 송규진, “컴퓨터 게임에서의 인공지능 기술에 관한 연구”, 한국컴퓨터정보학회지, Vol. 25, No. 1, pp. 11-19, Jun. 2017.

[2] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. H. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, “Soft Actor-Critic Algorithms and Applications”, arXiv:1812.05905, Jan. 2019.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms”, arXiv:1707.06347, Aug. 2017.

[4] Unity Blog. ML-Agents v2.0 릴리스: 복잡한 협동형 동작 훈련 지원[Internet]. Available : https://blog.unity.com/kr/technology/ml-agents-v20-release-now-supports-training-complex-cooperative-behaviors.

[5] 박해찬, 백낙훈, “딥러닝 지원 게임엔진을 통한 강화학습”, 한국정보과학회 2020 한국소프트웨어종합학술대회 논문집, pp. 731-732, Dec. 2020.

[6] 김윤희, 백종찬, 최진석, 박종혁, 김해연, 한수희, “강화학습을 이용한 카오틱 시스템 제어”, 제어로봇시스템학회 국내학술대회 논문집, Vol. 35, pp. 389-390, Jul. 2020.

[7] T. Degris, M. White, and R. S. Sutton, “Off-Policy Actor-Critic”, arXiv:1205.4839, Jun 2013.

[8] M. Andrychowicz, A. Raichuk, P. Stanczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, and O. Bachem, “What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study”, arXiv:2006.05990, Jun. 2020.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need”, arXiv:1706.03762, Dec. 2017.