Unity 3D 기반 ML-Agents Toolkit을 이용한 강화 학습 환경 설계 및 구현

최호빈*, 김찬명**, 김주봉*, 한연희*†
*한국기술교육대학교 컴퓨터공학과
**한국기술교육대학교 첨단기술연구소
e-mail: {chb3350, cmdr, rlawnqhd, yhhan}@koreatech.ac.kr

Design and Implementation of Reinforcement Learning Environment Using Unity 3D-based ML-Agents Toolkit

Ho-Bin Choi*, Chan-Myung Kim**, Ju-Bong Kim*, Youn-Hee Han*
*Dept of Computer Science Engineering, KoreaTech University
**Advanced Technology Research Center, KoreaTech University

요 약

강화 학습은 일반적으로 제어 로봇과 관련이 있는 순차적 의사결정을 위한 학습의 한 형태이다. 이 강화 학습은 행동에 대한 보상을 최대로 하는 정책을 학습하는 것을 목표로 한다. 하지만, 강화 학습을 실제 세계에 적용하기에는 많은 제약사항이 존재하며 실제 세계의 복잡한 환경에서 좋은 정책을 학습하는 것은 매우 어렵다. Unity는 강화 학습 시뮬레이션을 위한 전용 Toolkit을 제공한다. 이러한 이유로 Unity를 시뮬레이터로서 사용하는 것이 좋은 정책을 학습하는 훈련의 근거가 된다. 따라서 본 논문에서는 강화 학습을 실제 세계에 바로 적용시키기 전에 Unity Machine Learning Agents Toolkit을 사용하여 실제 세계와 비슷한 환경을 만들고 강화 학습을 통해 에이전트를 미리 학습시켜보는 과정을 수행해봄으로써 시뮬레이터의 필요성을 부각시킨다.

1. 서론

실제 세계 데이터를 대량으로 수집하는 데 비용이 많이 든다는 사실은 강화 학습을 실제 세계에 적용시키기 어렵게 만든다. 또, 실제 세계에서는 에이전트가 마음껏 시행착오를 겪어보기엔 한계가 존재한다. 따라서 시뮬레이션을 사용하면 대량의 데이터를 쉽게 생성해 낼 수 있고 많은 시행착오를 겪어볼 수 있다. Open AI는 개발자와 연구자들이 강화 학습 알고리즘을 검증해 볼 수 있도록 게임 환경과 3차원 환경을 무료로 공개한다. 개발자와 연구자들은 동일한 환경에서 본인의 강화 학습 알고리즘을 적용해 보고 점수를 매기며 경쟁한다. 이는 보다 좋은 알고리즘이 만들어지기 위한 촉매 역할을 하지만 Open AI Gym에서는 환경을 직접 만들어 볼 수는 없다.

Unity Machine Learning Agents Toolkit (ML-Agents Toolkit) [1, 2]은 시뮬레이션으로 지능형 에이전트를 훈련시키는데 필요한 환경 역할을 할 수 있는 오픈 소스 Unity 플러그인이다. 에이전트는 사용이 비교적 간편한 Python API를 통해 강화 학습, 모방 학습, 신경 진화 또는 기타 기계 학습 방법을 사용하여 훈련할 수 있다.

훈련된 단일 에이전트는 adversarial work나 cooperative work같은 다양한 상황에서도 다중 에이전트의 일부로 동작 제어가 가능하다. 또, 게임 빌드의 자동 테스트 및 게임 설계 결정 사전 릴리즈 평가 등 다양한 용도로 사용할 수 있다. ML-Agents Toolkit은 AI의 진보를 Unity의 풍부한 환경에서 평가한 후 더 넓은 연구 및 게임 개발자 커뮤니티에 접근할 수 있는 중앙 플랫폼을 제공하기 때문에 게임 개발자와 AI 연구원 모두에게 상호 이익이 된다.

ML-Agents Toolkit은 게임 개발자와 취미 생활자들이 2D, 3D, VR/AR 게임을 위한 지능형 에이전트를 쉽게 훈련할 수 있도록 최첨단 알고리즘(Tensorflow 기반)을 제공한다. 하지만 본 논문에서는 ML-Agents Toolkit에서 제공하는 알고리즘을 사용하지 않고 직접 구현한 Asynchronous Advantage Actor-Critic (A3C) 알고리즘을 사용하여 실험을 진행한다.

본 논문에서는 Unity와 ML-Agents의 기본적인 사용법을 다루지 않는다. 2장에서는 ML-Agents가 Unity에 어떻

† 교신저자: 한연희(한국기술교육대학교)
"이 논문은 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임
(No. 2018R1A6A1A03025526, No. 2016R1D1A3B03933355)"

게 구현이 되어있고 ML-Agents가 어떠한 구조로 이루어져 있는지 설명한다. 3장에서는 ML-Agents Toolkit을 사용해 구성한 환경을 자세히 기술한다. 4장에서는 실험에 사용한 A3C 알고리즘의 핵심 기술을 설명한다. 5장에서는 설정한 하이퍼파라미터를 간략하게 기술하고 실험 결과를 그래프로 나타낸다. 마지막 6장에서는 결론을 기술한다.

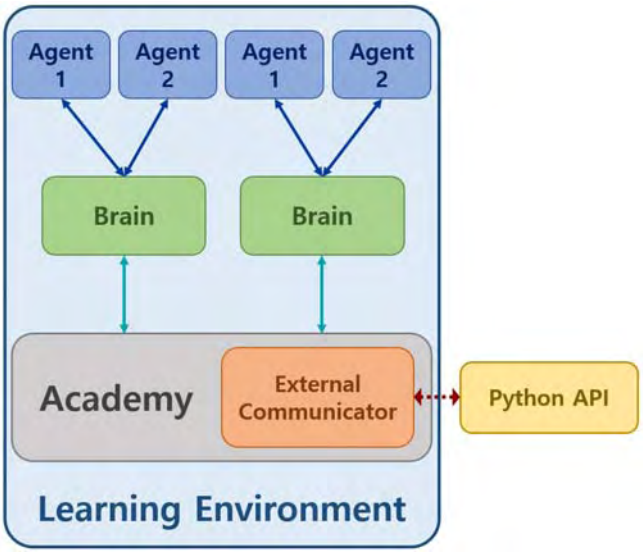
2. Unity Machine Learning Agents Toolkit

ML-Agents Toolkit은 세 가지 고급 구성요소로 이루어진 Unity 플러그인이다(그림 1). 다음은 그 세 가지 구성요소이다:

- **Learning Environment** - Unity scene과 모든 게임 에이전트를 포함한다.
- **Python API** - 훈련에 사용할 알고리즘이 위치한다. Learning Environment와 다르게 Python API는 Unity의 일부가 아니고 외부에서 External Communicator를 통해서 Unity와 통신한다.
- **External Communicator** - Learning Environment 안에 존재하며 Learning Environment와 Python API을 연결한다.

Learning Environment는 Unity scene을 조직화하는 데 도움이 되는 세 가지 구성요소로 이루어져 있다(그림 1). 다음은 그 세 가지 구성요소이다:

- **Agents** - Unity GameObject에 할당되며 observation을 얻을 수 있고 action을 수행하며 그에 따라 적절한 보상을 받는다. 각 에이전트는 정확히 하나의 Brain과 연결된다.
- **Brains** - 각 에이전트에 대한 정책을 고수하며 각 인스턴스에서 어떤 action을 취해야 하는지를 결정한다. 구체적으로는 에이전트로부터 observation과 reward를 받고 action을 반환한다.
- **Academy** - observation과 의사 결정 과정을 조율한다. 렌더링 품질과 Environment 실행 속도와 같은 여러 가지 환경적 매개변수를 설정할 수 있다. External Communicator가 바로 이곳에 위치한다.



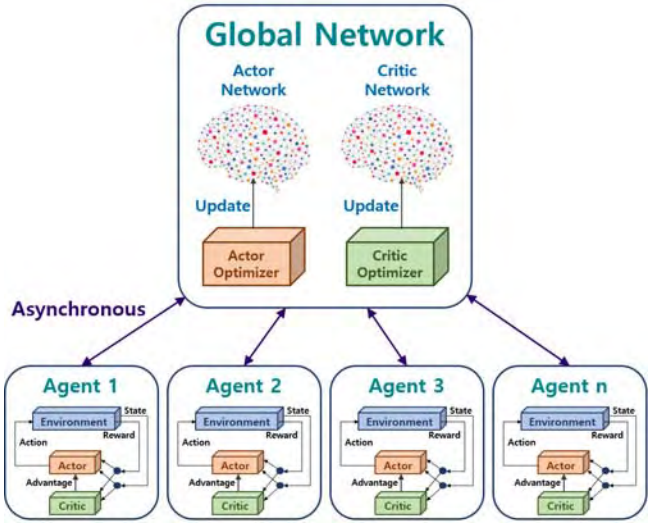
(그림 1) block diagram of ML-Agents toolkit

3. 환경 설계

강화 학습에는 속성 선택과 모델 선택이라는 두 가지 작업이 있다. 속성 선택은 목표 달성에 가장 도움이 되는 에이전트에 대한 observation 집합을 정의하는 것이다. 여기서 중요한 것은 목표 설정과 observation 집합의 정의이다. 반면, 모델 선택은 정책의 형식(observation에서 action으로의 mapping)과 그 매개변수를 정의하는 것이다. 실제로 강화 학습 훈련은 속성 선택과 모델 선택을 번갈아가며 반복하는 과정이다. 다음은 본 논문에서 사용한 환경의 속성 선택에 대한 정보를 자세하게 보여준다.

- **Set-up**: 에이전트가 스스로 굴러다닐 수 있는 플랫폼 환경
- **Goal**: 에이전트(공)가 Target(상자)에게 이동
- **Episode 종료 조건**
  - Max Step: 100
  - 에이전트가 Target에 닿았을 때
  - 에이전트가 바닥 아래로 떨어졌을 때
- **Agent**: 환경에는 brain에 연결된 하나의 에이전트가 존재
- **Agent Reward Function**:
  - 아래의 두 가지 상황이 아닌 매 step 마다 -0.01
  - 에이전트가 Target에 닿으면 +1.0
  - 에이전트가 바닥 아래로 떨어지면 -1.0
- **Brain**: 환경에는 하나의 brain이 존재하며 다음과 같은 observation/action space를 가짐
  - **Observation space (vector)**: -1에서 1사이의 연속적인 실수 6개
    ① Target의 x좌표
    ② Target의 z좌표
    ③ 에이전트의 x좌표
    ④ 에이전트의 z좌표
    ⑤ 에이전트의 x축 속도
    ⑥ 에이전트의 z축 속도
  - **Action space (vector)**: -1에서 1사이의 연속적인 실수 2개
    ① 에이전트에게 가할 x축 힘
    ② 에이전트에게 가할 z축 힘
  - **Visual observations**: 없음

모델 선택에 대해서는 5장에서 기술한다.

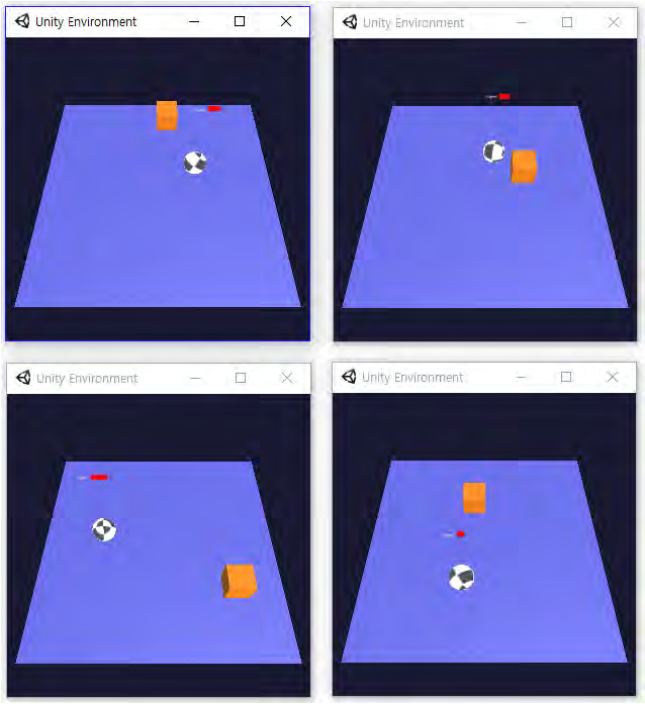


(그림 2) Asynchronous Advantage Actor-Critic (A3C)

4. A3C 알고리즘

2016년 2월 구글의 딥마인드는 DQN의 단점을 개선한 Asynchronous Advantage Actor-Critic (A3C)를 발표했다 [3]. DQN의 핵심은 샘플 사이의 강한 상관관계를 없애기 위해 리플레이 메모리를 사용하는 것이다 [4]. 하지만 리플레이 메모리에 대량의 샘플을 쌓고 학습을 시키기 때문에 메모리 공간을 많이 차지하고 오래된 샘플을 훈련에 사용한다는 단점이 있다. A3C는 샘플 사이의 강한 상관관계를 비동기 업데이트로 해결하여 리플레이 메모리를 사용하지 않는다. 리플레이 메모리를 사용하지 않아 최신의 샘플로 훈련하기 때문에 학습이 잘 된다.

A3C는 기본적으로 Actor-Critic구조를 사용한다. Actor는 말 그대로 행위자이다. 즉, 에이전트가 어떤 행동을 할지 결정한다. 마찬가지로 Critic도 말 그대로 비평가이며 Actor가 한 행동을 평가하고 그 결과를 Advantage값으로 Actor에게 전달한다. Actor와 Critic이 한 쌍이 되어 에이전트를 구성하고 이 에이전트는 환경과 상호작용하며 학습을 진행한다. 여기까지는 Advantage Actor-Critic (A2C) 알고리즘으로 볼 수 있다. A2C는 하나의 에이전트가 하나의 환경과 상호작용한다. 반면 A3C는 여러 개의 에이전트가 각각 자신의 환경과 상호작용한다(그림 2). A3C는 멀티 스레딩을 사용하여 여러 개의 A2C알고리즘이 동시에 진행된다. 각 A2C 알고리즘은 자신의 로컬 신경망을 업데이트 시키지 않고 Global Network를 업데이트 시키기 위해 Global Network의 옵티마이저로 학습에 필요한 값들을 전달한다. Global Network의 업데이트가 완료되면 해당 로컬 신경망의 가중치들을 Global Network의 가중치들로 업데이트 시킨다. 이 과정이 비동기로 진행되기 때문에 샘플 사이의 강한 상관관계를 해결할 수 있다.



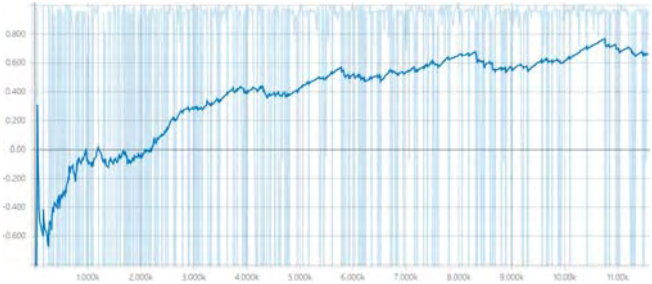
(그림 3) 학습 과정

5. 실험

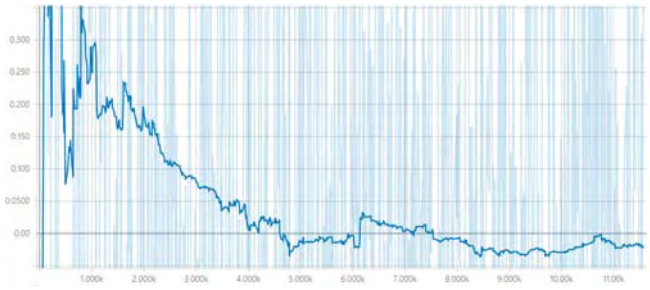
스레드는 총 5개를 사용하여 글로벌 네트워크의 업데이트는 메인 스레드가 수행하고 나머지 네 스레드는 각각 에이전트와 환경의 상호작용을 수행한다(그림 3). 즉, 4개의 A2C 알고리즘이 글로벌 네트워크를 비동기로 업데이트 시킨다. 각 A2C 알고리즘은 10스텝마다 혹은 에피소드가 끝날 때마다 글로벌 네트워크를 업데이트 시킨다.

Actor의 hidden layer는 6개, Critic의 hidden layer는 7개이고 모든 hidden layer의 뉴런 수는 128개이다. 옵티마이저는 Adam을 사용하였고 Actor의 learning rate는 0.00002, Critic의 learning rate는 0.0001로 설정하였다.

결과는 TensorFlow가 제공하는 TensorBoard를 활용하여 3개의 그래프로 나타내었다. 학습은 최근 100 Episode 동안 에이전트가 Target에 닿은 비율이 95%가 될 때까지를 종료조건으로 하여 약 11500 Episode가 진행되었다. 먼저 그림 4는 Episode에 따른 Score 그래프이다. 학습이 진행될수록 Score가 1에 가까워지는 것을 확인할 수 있다.

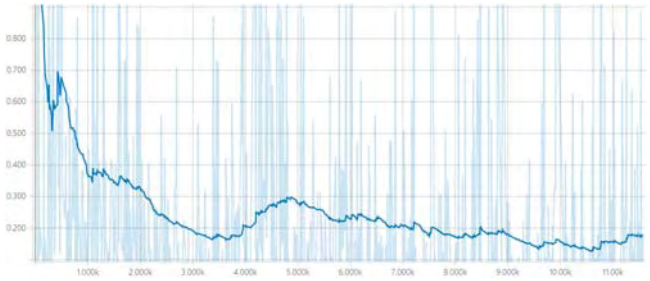


(그림 4) Score/Episode



(그림 5) Actor Loss/Episode

그림 5는 Episode에 따른 Actor Loss이다. Loss Function은 Cross Entropy를 사용하였고 최종 Loss에는 Advantage가 곱해진다. Advantage가 곱해지기 때문에 Actor Loss는 양수와 음수 모두 나올 수 있고 Actor 옵티마이저는 Loss를 감소시키는 방향으로 신경망을 업데이트 시킨다. 학습이 진행될수록 Actor Loss가 감소하는 것을 확인할 수 있다.



(그림 6) Critic Loss/Episode

그림 6은 Episode에 따른 Critic Loss이다. Loss Function은 Mean Square Error를 사용하기 때문에 Critic Loss는 양수만 나오며 Critic 옵티마이저는 Loss를 감소시키는 방향으로 신경망을 업데이트시킨다. 학습이 진행될수록 Critic Loss가 0에 가까워지는 것을 확인할 수 있다.

6. 결론

ML-Agents Toolkit을 이용해 강화 학습에 사용할 환경을 Observation부터 action, reward까지 직접 정의하여 만들 수 있다. 이는 개발자와 연구자들에게 강화 학습 알고리즘을 검증해 볼 수 있는 기회를 제공하는 것에 그치지 않고, 실제 세계와 비슷한 환경을 만들어 해당 분야에 실질적으로 강화 학습을 적용해 볼 수 있도록 하는 역할을 한다. 본 논문에서는 특정 환경을 만들고 A3C 알고리즘을 적용하여 성공적으로 학습시켜봄으로써 그것이 가능함을 입증한다.

참고문헌

[1] Juliani, A., Berges, V., Vckay, E., Gao, Y., Henry, H., Mattar, M., Lange, D., “Unity: A General Platform for Intelligent Agents,” arXiv preprint arXiv:1809.02627, 2018.

[2] https://github.com/Unity-Technologies/ml-agents.

[3] Volodymyr Mnih, et al., “Asynchronous Methods for Deep Reinforcement Learning,” arXiv:1602.01783v2, 2016.

[4] Volodymyr Mnih, et al., “Human-level control through deep reinforcement learning,” Nature, Vol. 518, pp. 529-533, 2015.