강화학습의 로봇 응용 소개

김태우 · 이주행
과학기술연합대학원대학교(UST/ETRI) · 한국전자통신연구원(ETRI/UST)

1. 서 론

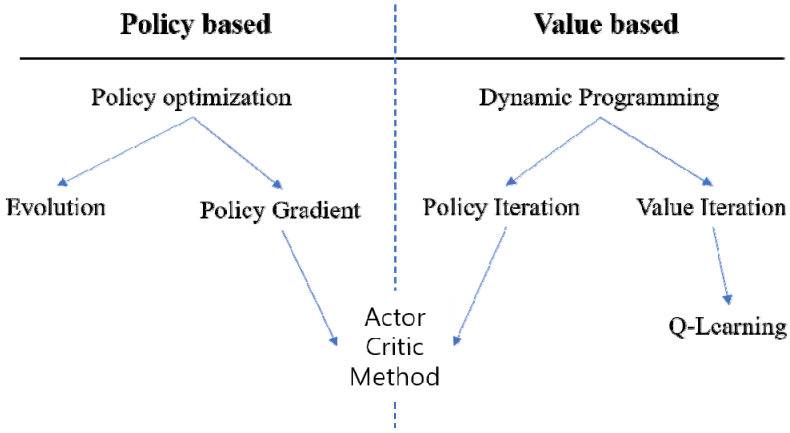
로봇을 제어하기 위한 전통적인 방법은 로봇의 기구학(Kinematics) 및 동역학(Dynamics)의 모델링에 기반한다. 그러나 로봇의 구조 및 작업 환경이 복잡해질수록 정확한 모델링이 어려워진다. 특히 로봇의 동역학과 환경을 정확하게 모델링하는 것은 높은 수준의 지식과 경험, 그리고 정밀한 하드웨어를 요구하며, 이는 많은 제반 비용이 요구되는 큰 장벽으로 존재해왔다.

알파고[1]의 등장과 함께 큰 주목을 받게 된 강화학습은 게임 및 로봇 분야에 널리 활용되어 연구되기 시작했고 많은 성공사례가 등장했다. 특히 복잡한 환경에 대한 모델링 없이 체험적(heuristic)으로 학습할 수 있다는 점은 로봇과 같이 정확한 모델링이 필요한 분야에서 큰 이점으로 작용한다. 더불어 하드웨어 없이도 로봇을 학습시키기 위한 다양한 시뮬레이터들이 무료로 제공됨에 따라 더욱 진입 장벽이 낮아지고 있다. 이로 인해 최근 로봇 분야에서 강화학습을 이용한 연구가 활발히 진행되고 있다.

본 기고문에서는 주요 강화학습 모델들의 기본 개념과 연구 동향을 살펴보고 로봇을 학습시키기 위한 시뮬레이션 환경 구성에 대해 알아보고자 한다.

2. 강화학습

강화학습(Reinforcement Learning)은 두 갈래의 큰 흐름으로 발전해왔다. 한 가지는 가치 함수(value function)를 이용하여 최적 해를 찾는 가치기반(value based)방법이고 다른 한 가지는 trial and error를 통해 정책을 직접 근사하는 정책기반(policy based)방법이다. 가치기반 방법은 어떤 상태에서 취한 행동의 가치를 계산하여 학습학고, 행동을 선택할 땐 최고의 가치를 갖는 액션을 선택하는 것으로 정책이 묵시적(implicit)으로 결정된다. 반면 정책기반 방법은 정책이 명시적(explicit)으로 정의되며, 특정 상태에서 최적의 행동을 선택하는 것이 목표이다. [그림 1]은 강화학습을 가치기반



[그림 1] 강화학습의 학습 기반에 따른 분류[2]. (좌)정책기반 강화학습. (우) 가치기반 강화학습.

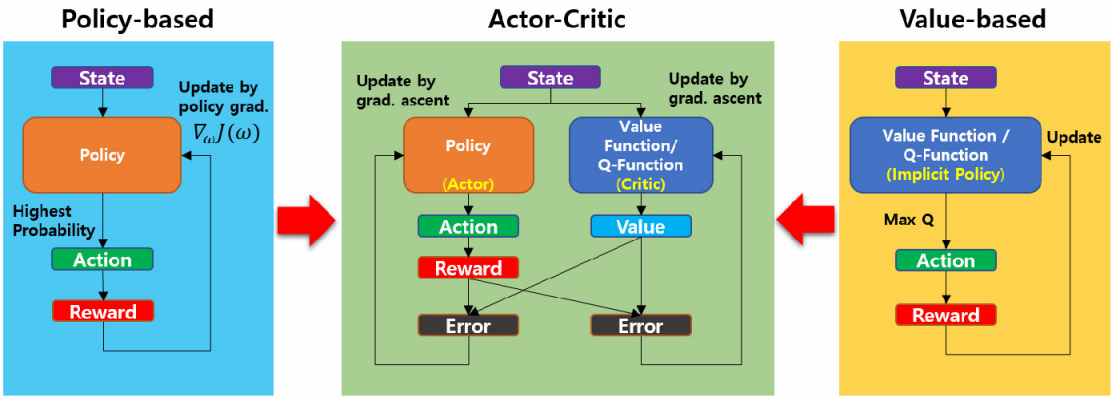
과 정책기반 방법으로 분류한 모습이다. 두 방식의 기본적인 개념과 발전과정, 그리고 로봇에 널리 이용되는 최신 알고리즘에 대해 간략히 살펴보자.

2.1 가치기반 강화학습

[그림 1]의 우측에 표현된 가치기반(value based) 강화학습은 벨만 방정식을 기반으로 고안된 다이내믹 프로그래밍을 기저에 두며, 이로 부터 파생된 알고리즘이다[3]. 다이내믹 프로그래밍은 학습 알고리즘 이라기보다는 어떤 문제를 작은 단위의 문제들로 나누어 반복적 계산을 통해 풀어내는 최적화 방법이다. 이는 다시 정책 이터레이션(Policy Iteration) 방법과 가치 이터레이션(Value Iteration) 방법으로 나눌 수 있다[4]. 정책 이터레이션은 정책과 정책을 평가하기 위한 가치함수를 따로 두고 정책 평가와 정책 발전을 순차적으로 반복하여 최적 정책을 찾는 방법이다. 반면 가치 이터레이션은 정책을 따로 두지 않고 오직 가치함수만 고려하는 방법이다. 정책이 겉으로 드러나 있진 않지만, 가치함수에 내재적(implicit)으로 정의되어 있으므로 최적 가치함수를 찾게 되면 자동적으로 최적 정책이 구해진다. 이 방법은 에이전트가 특정 상태에서 행동을 선택할 때 따로 정책이 없으므로 가치 함수를 통해 최대 가치를 갖는 행동을 선택한다. 즉 현재의 가치함수를 최적 가치함수라 가정하고 보상을 최대화 하는 방향으로 계속 업데이트해서 최적 가치함수를 구하는 것이다.

그러나 이러한 다이내믹 프로그래밍은 환경의 모델을 알아야 하고 상태의 크기가 증가함에 따라 계산복잡도가 급격히 증가하는 문제, 차원의 늘어날수록 계산량이 기하급수적으로 증가하는 차원의 저주(Curse of Dimensionality) 등의 한계가 존재한다. 이러한 한계를 극복하기 위해 나온 알고리즘이 가치 이터레이션에서 발전된 큐러닝(Q-Learning)이다[5]. 큐러닝은 어떤 상태에서 할 수 있는 모든 행동에 대한 가치(value)를 고려한 가치함수대신 에이전트가 상태 s에서 특정 행동 a를 했을 때의 큐값을 반환해주는 큐함수(Q function)를 학습한다. 가치 이터레이션과 마찬가지로 큐함수에 정책이 내재되어 있기 때문에 정책을 따로 두지 않고 큐함수만을 학습시키고, 에이전트는 큐함수를 통해 가장 큰 큐값을 보이는 행동을 수행한다. 이후 에이전트는 행동을 통해 환경으로부터 얻은 보상을 기반으로 현재의 상태 및 행동에 대한 큐함수를 업데이트하게 된다. 가치기반 방법 및 큐러닝에 대한 기본 개념이 [그림 2]의 우측에 표현되어있다.

큐러닝은 다이내믹 프로그래밍과는 달리 환경의 모델을 몰라도 학습이 가능하다. 또한 계산복잡도 문제를 해결하기 위해 일반적으로 큐함수는 근사(approximation)



[그림 2] 강화학습의 학습기반에 따른 기본 개념도. (좌)정책기반 방법. (중)혼합방법. (우)가치기반 방법

시키는데, 가장 널리 사용되는 근사방법은 인공신경망이다. 대표적인 가치기반 강화학습 알고리즘에는 인공신경망을 이용하여 큐함수를 근사시킨 DQN이 있다[6][7].

2.2 정책 기반 강화학습

가치 기반 강화학습에선 정책이 내재된 가치함수 혹은 큐함수를 토대로 가장 큰 가치를 지니는 행동을 선택했다면, 정책 기반 강화학습에선 정책 자체를 보유한다. 즉 에이전트는 주어진 정책에서 현재의 상태에 따른 행동을 한 뒤 보상을 받고 정책을 업데이트 하는 과정을 따르게 된다. 예를 들어 에이전트는 뉴럴넷으로 직접 근사되는 정책을 갖는다고 생각해보자. 뉴럴넷에 상태를 입력하여 나온 출력은 현재 상태에서 현재 정책을 따랐을 때의 각 행동을 할 확률값이 되고, 가장 높은 확률을 갖는 행동을 취할 것이다. 에이전트는 가장 높은 확률의 행동을 지속하여 그에 따른 보상을 얻고, 시간에 따라 누적되는 보상의 합을 최대화 시키는 방향으로 정책을 최적화 시킨다.

정책 최적화 과정은 먼저 목표함수(Objective Function)에 대한 미분을 정의하고, 누적 보상을 최대화 시키는 방향으로 파라미터를 업데이트한다. 여기서 파라미터란 뉴럴넷으로 근사했을 경우 각 층의 입력값에 곱해지는 가중치(weight)와 바이어스(bias)를 의미한다. 목표함수의 값을 최대로 만들어주는 파라미터를 찾는 것이 목표이므로 경사상승법(gradient ascent)을 이용하여 에러를 역전파하여 파라미터를 업데이트 한다. [그림 2]의 왼쪽에 정책기반 방법에 대한 개념도가 있다. 결국 누적 보상을 최대화 시키는 최적 정책에 대한 파라미터를 찾는 것이 정책 기반 강화학습의 목표이며, 대표적인 알고리즘은 REINFORCE[8] 알고리즘이 있다. 이 외에도 생물의 진화과정을 모방하여 만든 진화 알고리즘(Evolution)이 정책 최적화 방법의 한 가지로 존재한다[9].

2.3 혼합 방법

정책기반 알고리즘은 가치 기반 방법에 비해 여러 장점을 가지고 있다. 가치 기반 방법에 비해 수렴 과정이 비교적 안정적이고 고차원 혹은 연속적인 행동 공간에서의 학습에 훨씬 효과적이다. 또한 확률론적(stochastic) 정책을 학습할 수 있다. 그러나 지역 최적점(local optima)에 빠질 확률이 높고 정책 평가에서의 분산이 크다는 단점이 있다.

이러한 정책기반 방법의 단점을 보완하고 가치기반 방법의 장점을 취하기 위해 나온 것이 [그림 1]의 하단에 표현된 액터-크리틱(Actor-Critic) 방법이다. 액터-크리틱 방법은 가치기반 방법과 정책기반 방법을 혼합한 것

[그림 3] 강화학습의 로봇 응용 사례[13-14]. (a) 헬리콥터 호버링. (b) 병뚜껑 닫기

으로써 정책과 가치함수를 모두 가지고 있다. 가치기반 방법의 정책 이터레이션과 마찬가지로 정책 평가-정책 발전의 사이클이 존재한다.

에이전트는 현재 자신의 상태에서 주어진 정책을 따르는 행동을 선택하고 환경으로부터 보상을 얻는다. 에이전트가 현재 상태에서 적절한 행동을 하도록 만드는 것이 정책이고, 이러한 일련의 행동을 결정해주는 주체가 액터-크리틱 방법에서 액터(Actor)에 해당한다. 현재의 상태는 정책뿐만 아니라 가치함수(큐함수)에도 입력되는데, 여기서 가치함수는 현재의 상태에 따른 행동의 가치를 출력하고, 이 출력된 가치는 에이전트가 현재 정책을 따랐을 때 얼마나 좋은 행동을 산출해내는지에 대한 평가 척도가 된다. 결국 가치함수는 현재 상태 및 행동에 대한 가치값(큐값)을 기반으로 현재의 정책을 평가하게 된다. 따라서 액터-크리틱에서 가치함수는 현재 정책을 비판하고 평가하는 역할을 하기 때문에 크리틱(Critic)에 해당한다.

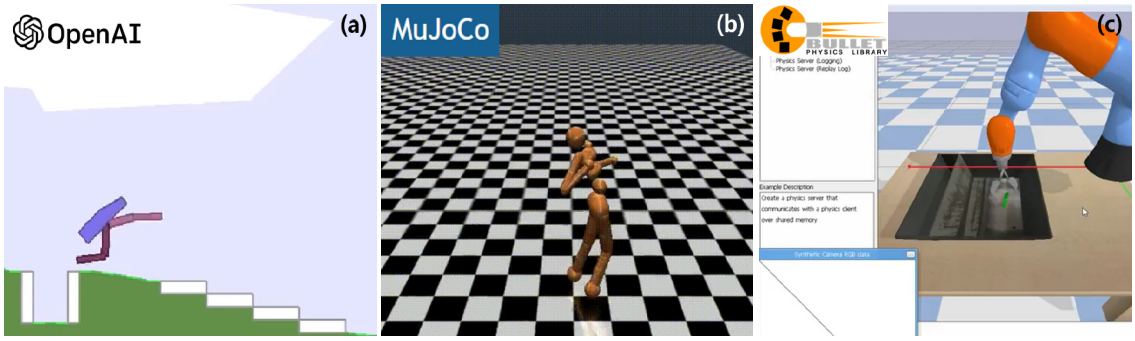
일반적으로 정책은 근사되기 때문에 Policy Gradient 방법을 기반으로 하고, 보상 및 크리틱의 평가를 통해 계산된 에러 만큼 업데이트된다. 동시에 가치함수역시 에러를 계산하여 업데이트 되며, 이때의 에러는 환경으로부터 얻은 보상과 다음 상태의 값을 더한 정답과, 현재 가치함수를 통해 예측한 예측값 사이의 시간차 에러(Temporal Difference Error)를 통해 업데이트를 수행한다. 결국 액터를 통해 에이전트의 행동을 결정하고, 액터에 대한 크리틱의 평가를 통해 보다 나은 정책으로 계속 발전하는 것이 액터-크리틱 방법이다. [그림 2]의 가운데에 액터-크리틱 방법의 구조가 표현되어 있다.

정책 기반 방법과 가치 기반 방법의 장점들을 취하여 만든 액터-크리틱은 최근 연구에서 쏟아져 나오는 수많은 알고리즘들의 기반이 되었다. 대표적인 알고리즘에는 2016년에 발표된 비동기 업데이트 방식의 A3C가 있다[10]. 특히, 로봇과 같이 행동 공간이 연속적인 경우에 최근 널리 사용되는 알고리즘은 TPRO[11], PPO[12]등이 있다.

3. 강화학습의 로봇 응용 사례

최근 개최된 국제 로봇학회 ICRA 2018의 홈페이지에 게재된 정보에 따르면 제출된 논문 중 딥러닝을 이용한 로봇 응용이 가장 높은 비중을 차지할 정도로 로봇 분야에서도 머신러닝의 인기는 그 어느 때보다 뜨겁다. 그 중 강화학습을 이용한 로봇 응용은 어떠한 것들이 있는지 살펴보자.

Andrew Y. Ng 연구팀은 강화학습을 이용하여헬리콥터의 자동 비행을 구현했다[13]. 헬리콥터의 비행은 잡음 및 비선형성으로 인해 모델링하기 매우 까다로운 문제이나, [그림 3]의 왼쪽과 같이 강화학습을 이용하여 곡예비행을 성공시켰다. Sergey Levin은 PR2로봇을 이용하여 옷걸이 걸기, 큐브 맞추기, 망치 작업, 병뚜껑 닫기



[**그림 4**] 강화학습을 위한 시뮬레이션 환경[19-21]. (a) OpenAI. (b) Mujoco. (c) pybullet

등 일상생활의 다양한 행위를 강화학습으로 풀었다[14]. [그림 3]의 오른쪽과 같이 로봇에 달린 카메라로 인식한 환경을 통해 미션을 수행하는 복잡한 문제를 성공시킨 사례이다. Aleksandra Faust는 샘플링 기반 경로계획방법에 강화학습을 조합하여 모바일 로봇과 항공화물운송에 대한 로봇 네비게이션 문제를 다루었다[15]. 비카리우스(Vicarious)는 로봇의 인공일반지능(Artificial general intelligence)을 연구하고 있는 회사다. 게임을 위한 강화학습, 컴퓨터 비전, 로봇 등 다양한 분야의 연구를 진행하고 있다[16-18].

4. 시뮬레이션 환경

시뮬레이션은 하드웨어에 알고리즘을 적용하기 전에 거쳐야 할 필수적인 과정이다. 강화학습이 주목을 받기 시작하면서 강화학습을 위한 무료 시뮬레이션 환경이 제공되기 시작하였고 이를 통해 많은 연구가 이루어지고 있다. 본 장에서는 강화학습 시뮬레이션을 위한 몇 가지 환경을 소개한다.

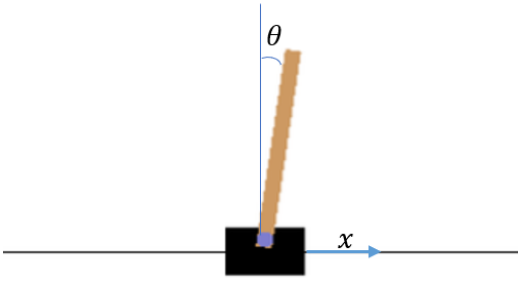
[그림 4]의 (a)는 2015년에 설립된 비영리 연구회사인 OpenAI에서 제공하는 Gym의 모습이다[19]. OpenAI Gym에선 강화학습 시뮬레이션을 위한 각종 게임 및 간단한 형태의 로봇 등 다양한 환경을 제공한다. 모든 기능은 무료이다.

(b) 는 Mujoco라고 하는 물리 시뮬레이터이다[20]. 휴머노이드 로봇, 치타 로봇, 매니퓰레이터 등 다관절 로봇의 역학 시뮬레이션 환경을 제공해주며 OpenAI Gym과 연동하여 강화학습을 할 수 있다. 처음 한 달간은 무료로 이용할 수 있으며, 학생인 경우 1년 무료 라이센스를 얻을 수 있다.

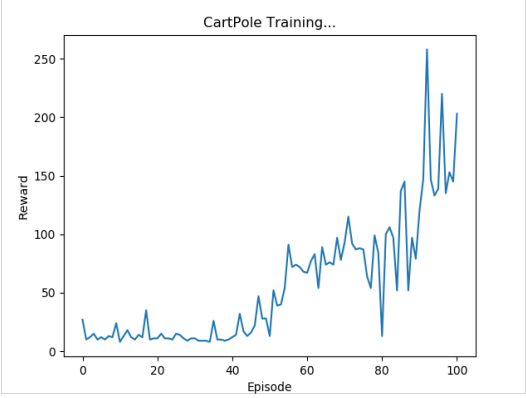
마지막으로 (c)는 pybullet이라는 시뮬레이터이다[21]. bullet 물리 엔진을 python모듈로 만든 것이며 사용자가 시뮬레이션 환경을 구성함에 있어 자유도가 높다. ROS의 URDF, SDF 포맷을 이용하여 자유롭게 로봇 및 환경을 구성할 수 있다[22]. Kuka 로봇, racecar 등의 기본적인 로봇 모델을 제공해주며 정/역기구학 및 동역학, 그리고 충돌 검출 등을 위한 라이브러리도 함께 제공된다. 사용자의 문제에 맞게 시뮬레이션 환경을 구성하기 편리하다. 모든 기능은 무료로 제공된다.

5. 강화학습 실습

앞서 설명한 시뮬레이션 환경에서 강화학습을 이용한 간단한 형태의 로봇 제어 문제를 실습해보자[23]. 구현에 사용한 언어는 python이며 강화학습은 pytorch 0.4.0 버전을 사용하였다. 버전에 따라 함수 사용법이 달라지는 부분이 있으므로 가급적 0.4.0 버전으로 맞추는 것을 추천한다.



[그림 5] OpenAI Gym의 CartPole



[그림 6] CartPole의 에피소드별 보상

[표 1] CartPole의 MDP 정의.

MDP	정의	차원
상태(state)	$[x, \theta, \dot{x}, \dot{\theta}]$	4
행동(action)	왼쪽/오른쪽 방향의 임펄스	2
보상(reward)	Pole이 중심으로부터 일정 각도 내에 있으면 1, 아니면 0	1

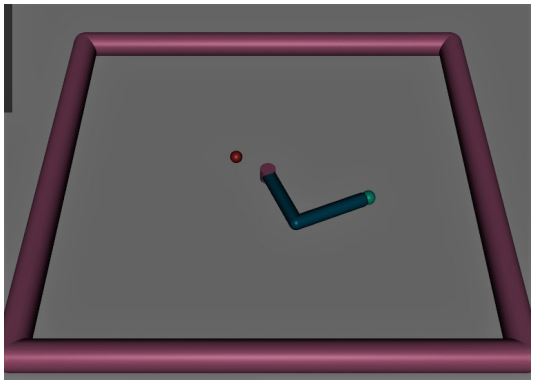
5.1 CartPole

CartPole은 좌우로 움직일 수 있는 cart위에 회전이 가능한 pole이 붙어있고, cart를 좌우로 적절히 움직여서 pole을 쓰러트리지 않는 것이 목표이다. [그림 5]는 CartPole의 실행모습을 나타낸다.

강화학습으로 이 문제를 풀기 위해선 먼저 MDP (Markov Decision Process)를 정의해야 한다. MDP를 정의한다는 것은 풀고자 하는 문제에서 상태(state), 행동(action), 보상(reward) 그리고 정책(policy)이 각각 무엇인지 정의하는 것이다. 정책은 우리가 강화학습을 이용하여 앞으로 풀어야 할 문제이므로 당장은 정의할 수 없다. 따라서 나머지 상태, 행동, 보상을 CartPole 문제에 대해 정의하면 [표 1]과 같다. CartPole의 상태는 cart의 위치와 중심축을 기준으로 pole이 회전된 각도, 그리고 각각의 속도이다. 행동은 왼쪽 혹은 오른쪽 방향으로 카트를 밀어서 움직이는 것이다. 보상은 한 step을 수행한 뒤 pole이 일정 각도 범위 내에 있으면 1, 해당 각도를 벗어나면 0을 얻게 된다. [표 1]에 설정된 MDP는 OpenAI Gym에 설정된 기본값과 동일하다. DQN을 이용하여 학습시킨 결과 [그림 6]과 같은 에피소드별 보상값을 얻었다. 보상이 높을수록 pole을 더 오랜 시간동안 세운 것을 의미한다. 학습이 진행될수록 점차 높은 보상을 얻게 된다. 실험 결과는 설정된 파라미터에 따라 상이할 수 있으며, 특히 보상 설정 방법에 따라 성능이 많이 달라질 수 있다. 자세한 파라미터 설정 및 구현 코드는 [23]에서 확인할 수 있다.

5.2 Reacher (Mujoco)

Reacher는 Mujoco에서 제공해주는 평면 2자유도의 로봇 팔이다. [그림 7]의 Reacher는 중앙에 고정되어 있는 2자유도 로봇 팔이 있고 팔 끝을 목표지점(빨간색 점)까지 움직여야 한다. [표 2]의 Reacher의 MDP는 다음과 같다. 상태는 각 조인트의 cos과 sin값, 목표점의 위치, 각속도, 팔 끝점과 목표점 사이의 거리로 구성되어 있다. 이와 같은 상태정의는 우리가 눈앞의 어떤 물건을 집으려 할 때 물체의 상대적인 위치와 자신의 팔의 현재 위치, 자세, 속도 등을 인지한 상태에서 팔을 움직여 물건을 집는 과정을 모사한 것이다. 행동은 각 관절의 속도이고, 보상은 로봇의 팔 끝과 목표 지점과의 거리, 그리고 현재 각속도의 제곱의 합을 음의 보상으로 준다. 이와 같이 보상을 설정한다면 로봇은 목표지점에 최대한 가까이, 그리고 최대한 빠르고 적은 움직임을 통해 도달하는 방향



[그림 7] Mujoco의 Reacher 로봇

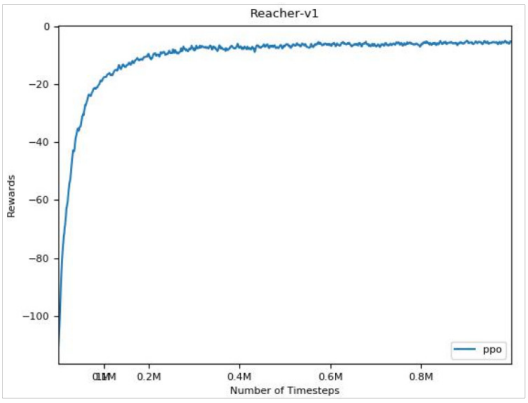
[표 2] Reacher의 MDP 정의.

MDP	정의	차원
상태(state)	$[c(\theta_1), c(\theta_2), s(\theta_1), s(\theta_2)]$ $[t_x, t_y, \dot{\theta}_1, \dot{\theta}_2, P_f - P_t]$	11
행동(action)	각 조인트의 각속도	2
보상(reward)	로봇 팔 끝과 목표 지점과의 거리, 각속도의 제곱의 합	1

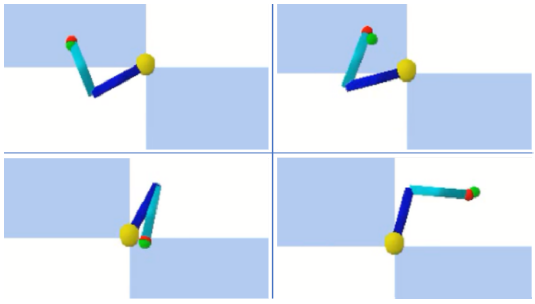
으로 학습이 될 것이다.

Reacher와 같은 로봇은 행동을 하는 공간이 연속적(continuous)이다. 따라서 DQN과 같은 가치 기반 방법은 연속적인 문제에 적용하는 것이 불가능한 것은 아니지만 그 한계가 존재한다. 예를 들어 Reacher의 행동 공간을 각 관절별로 [-0.05, 0.01, 0, 0.01, 0.05] rad/step으로 정의했다고 가정하자. Reacher는 두 개의 관절을 가지고 있으므로 고려해야 할 행동의 경우의 수는 5x5=25가지이다. 조인트가 3개인 경우 5^3 =125, 4개인 경우엔 5^4=625로 조인트가 늘어날 때마다 계산량이 급격히 증가하는 것을 볼 수 있다. 이처럼 DQN과 같은 가치 기반 방법은 로봇제어와 같은 연속행동 공간 문제에 그 한계가 명확하다는 것을 알 수 있다. 따라서 정책 기반 방법 혹은 혼합 방법을 사용하는 것이 유리하다.

앞서 언급했던 PPO 알고리즘은 Importance Sampling과 KL-divergence 등을 이용하여 신뢰 영역(Trust Region) 내에서 정책의 확률 분포를 추정하는 TPRO 알고리즘에 clipping과 엔트로피 방법 등을 더해 발전시킨 알고리즘



[그림 8] PPO알고리즘을 이용하여 학습시킨 Reacher의 에피소드 별 보상



[그림 9] pybullet으로 구현한 Reacher의 모습

이다. 연속적인 문제에 비교적 성능이 우수한 것으로 알려져 있다. [그림 8]의 Reacher학습 결과를 보면 초반부터 비교적 안정적으로 수렴하는 것을 볼 수 있다. pytorch 구현 코드는 [24]을 참고하기 바란다.

5.3 Reacher (pybullet)

[그림 9]는 Reacher를 pybullet에서 구현하여 실행한 모습이다. 로봇은 URDF 포맷을 이용하여 구, 실린더 등을 조합하여 생성하였다. MDP는 Mujoco의 Reacher와 동일하게 설정하였으며 로봇 링크의 길이는 임의로 설정하였다. 알고리즘은 PPO를 이용하였으며 [24]의 구현 코드를 활용하여 재구성하였다. 자세한 파라미터 및 구현 코드는 [23]에서 확인할 수 있다.

6. 결 론

이상에서 강화학습에 대한 기본적인 개념과 주요 연구동향 및 로봇 분야에서의 간단한 적용 예를 제시하였다. 알파고로 유명세를 탔던 강화학습은 게임 분야를 넘어 최근 로봇 분야에서 하나의 트렌드로 자리 잡았다. 특히 무료로 제공되는 강화학습 특화 시뮬레이션 환경은 강화학습을 공부하고자 하는 사람들에게 진입장벽을 낮춰줌으로써 보다 많은 사람들이 참여할 수 있도록 하였다. 그러나 여전히 실제 문제에 강화학습을 적용하여 성공시키기란 쉬운 일은 아니다. 문제에 대한 정확한 MDP 설정과 적합한 근사 함수 및 파라미터를 찾는 일, 정교한 테스트 환경 구축 등 어느 한 가지 라도 부족하다면 문제는 제대로 풀리지 않는다. 이는 탄탄한 이론적 배경 지식뿐만 아니라 많은 경험도 요구되는 일이다. 본 기고문이 강화학습에 입문하는 사람들에게 미약하나마 도움이 되었으면 하는 바람이다.

참고문헌

[1] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *Nature* 529.7587: 484, 2016.

[2] P. Abbeel, "Lecture 1 Intro to MDPs and Exact Solution Methods," *Deep RL Bootcamp*, 2017.

[3] R. Bellman, "The theory of dynamic programming," *Bull. Am. Math. Soc.*, vol. 60, no. 6, pp. 503-515, 1954.

[4] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. *MIT Press*, 1998.

[5] C. J. and P. D. Watkins, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279-292, 1992.

[6] V. Mnih et al., "Human-level control th rough deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, 2015.

[7] V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," *arXiv preprint arXiv:1312.5602*. 2013.

[8] R. S. Sutton, D. Mcallester, S. Singh, Y. Ma nsour, P. Avenue, and F. Park, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," *Adv. Neural Inf. Process. Syst.*, pp. 1057-1063, 2000.

[9] M. Kim-Fung, T. Kit-Sang, and K. Sam, "Genetic algorithms: concepts and applications [in engineering design]," *IEEE Trans. Ind. Electron.*, vol. 43, no. 5, pp. 519-534, 1996.

[10] V. Mnih et al., "Asynchronous Methods for Deep Reinforcement Learning," *Int. Conf. Mach. Learn.*, pp. 1928-1937, 2016.

[11] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization," *Int. Conf. Mach. Learn.*, pp. 1889-1897, Feb. 2015.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*. 2017.

[13] H. Kim, M. Jordan, S. Sastry, and A. Ng, "Autonomous helicopter flight via reinforcement learning," *Adv. Neural Inf. Process. Syst.*, pp. 799-806, 2004.

[14] S. Levine, C. Finn, D. T, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334-1373, 2016.

[15] A. Faust et al., "PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling -based Planning," *arXiv preprint arXiv: 1710.03937.* 2017.

[16] D. George et al., "A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs," *Science* (80-.)., vol. 358, no. 6368, p. eaag2612, 2017.

[17] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, X. Lázaro-Gredilla, M., Lou, and D. George, "Schema networks: Zero-shot transfer with a generative causal model of intuitive physics," *arXiv preprint arXiv: 1706.04317.* 2017.

[18] A. Stone, H. Wang, M. Stark, Y. Liu, and D. Phoenix, D. S. George, "Teaching Compositionality to CNNs," *arXiv:1706. 04313.* 2017.

[19] G. Brockman et al., "OpenAI Gym," arXiv p reprint *arXiv:1606.01540.* 05-Jun-201 6.

[20] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model -based control," in 2012 *IEEE/RSJ Inter national Conference on Intelligent Robots and Systems*, 2012, pp. 5026-5033.

[21] E. Coumans and Y. Bai, "pybullet, a python module for physics simulation for games, robotics and machine learning," *GitHub repository.* 2016.

[22] M. Quigley et al., "ROS: an open-source Robot Operating System," *ICRA*, vol. 3, no. 3.2, p. 5, 2009.

[23] T. Kim, "PyTorch Implementations for Reinforcement Learning Tutorial," *github. com/gd-goblin/RL_robotics_tutorial.* GitHub, 2018.

[24] I. Kostrikov, "PyTorch Implementations of Reinforcement Learning Algorithms," *GitHub repository.* GitHub, 2018.

김태우

2011 충남대학교 메카트로닉스(공학사)
2012~현재 과학기술연합대학원대학교 컴퓨터 소프트웨어 통합과정
관심분야 : 로보틱스, 컴퓨터 비전, 기계학습
E-mail : twkim0812@gmail.com

이주행

1994 POSTECH 전자계산학과 (학사)
1996 POSTECH 전자계산학과 (석사)
1999 POSTECH 전자계산학과 (박사)
1999~현재 한국전자통신연구원 지능로보틱스 연구본부 책임연구원
2008~현재 과학기술연합대학원대학교 컴퓨터소프트웨어전공 교수
관심분야 : 기하모델링, 컴퓨터그래픽스, AR/VR, 컴퓨터 비전, 로보틱스, 기계학습, 미디어 아트
E-mail : joohaeng@etri.re.kr