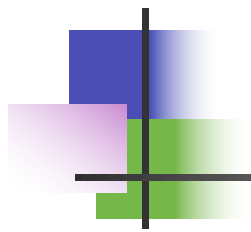


강화학습과 PPO 소개

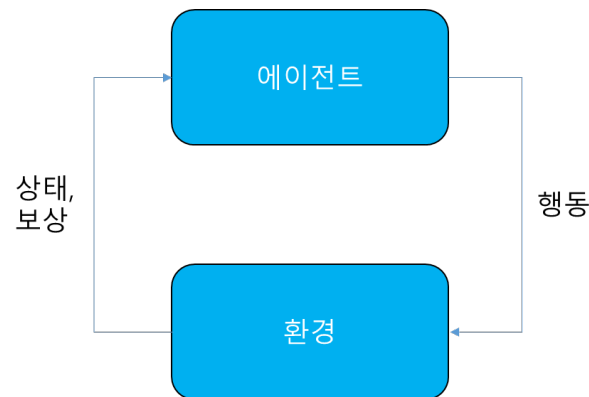
Sang Yup Lee



Reinforcement Learning

Reinforcement Learning

- 강화학습 소개
 - 강화학습은 일반적으로 순차적인 의사결정 문제에 적용
 - 강화학습 문제는 에이전트(agent)와 환경(environment)으로 구성된 시스템으로 표현
 - 에이전트는 환경과의 상호작용을 통해서 주어진 문제 해결
 - 환경은 시스템의 상태를 나타내는 정보를 생성하고 에이전트에 제공 ⇒ 이러한 정보를 상태(state)라고 표현





Reinforcement Learning

- 강화학습에서 사용되는 주요 표현
 - 타임 스텝 t 에서의 상태, 행동, 보상을 s_t, a_t, r_t 로 표현
 - $(s_t, a_t, r_t) \Rightarrow$ 하나의 경험(experience)이라고 함
 - 이러한 경험은 여러 번 반복되는데 일반적으로 정해진 타임 스텝 (예, time step= T) 까지 반복
 - 타임 스텝 0에서부터 T 까지를 하나의 에피소드(episode)라고 함
 - 궤적(trajjectory, τ /tau/라고 표현)
 - 하나의 에피소드에서 발생하는 연속된 행동들
 - $\tau = (s_0, a_0, r_0), (s_1, a_1, r_1), \dots, (s_T, a_T, r_T)$
 - 행동에 대한 보상은 보상 함수(reward function)를 이용해서 결정
 - 보상은 일반적으로 하나의 숫자



Reinforcement Learning

- 강화학습에서 사용되는 주요 표현 (cont'd)
 - 정책(policy): 주어진 상태에 따라 특정 행동을 선택할 때 사용되는 함수
 - 즉, 정책을 통해서 어떠한 행동을 수행할지를 결정
 - 좋은 정책을 학습하기 위해서는 많은 수의 에피소드 필요
 - 상태 전환 함수
 - 환경이 에이전트의 행동에 대해서 새로운 상태 정보를 제공할 때 사용되는 함수
 - $P(s_{t+1}|s_t, a_t) \Rightarrow$ 상태 전환 함수는 여러 가지 선택 가능한 상태들에 대한 확률 분포를 반환
 - 그 이전 타임 스텝에서의 상태와 행동에도 영향을 받을 수 있지만, 강화학습에서는 마르코브 특성(Markov property)을 사용

Reinforcement Learning

- 강화학습에서 사용되는 주요 표현 (cont'd)
 - 각 타임 스텝에서 에이전트는 목적을 최대화하는 행동을 선택
 - 그렇다면 목적은 어떻게 표현되는가? 이를 위해 먼저 이득(return)을 정의하는 것이 필요
 - 이득
 - $R(\tau) = r_0 + \gamma r_1 + \gamma^2 r_2 + \dots + \gamma^T r_T = \sum_{t=0}^T \gamma^t r_t$
 - τ 는 궤적을 의미, r_t 는 타임 스텝 t 에서의 보상을, γ 는 할인율 (discount factor)을 나타내며 γ 은 0과 1사이의 값을 취함
 - 할인율은 먼 미래의 보상일수록 현재 가치를 낮추기 위해서 사용
 - 목적
 - $J(\tau) = E_{\tau}[R(\tau)] = E_{\tau}[\sum_{t=0}^T \gamma^t r_t]$
 - 모든 궤적으로부터 발생하는 이득(returns)에 대한 기댓값
 - 에이전트는 이러한 목적을 최대화하는 행동 선택



Reinforcement Learning

- 강화학습에서 에이전트가 학습할 수 있는 것 세 가지: 정책, 가치함수, 환경 모형
 - 정책 (policy)
 - 정책은 주어진 상태에서 목적을 최대화하는 행동을 선택하는데 사용되는 함수, 주로 π 로 표현
 - 정책은 여러 선택 가능한 행동들에 대한 확률 분포(즉, $\pi(a|s)$)를 반환



Reinforcement Learning

- 가치 함수 (value function)

- 목적을 최대화하는데 있어서 특정 행동이나 상태가 얼마나 좋은 것인지를 평가하는 목적으로 사용되는 함수
- $V^\pi(s)$ 로 표현되는 가치 함수와, $Q^\pi(s, a)$ 로 표현되는 가치 함수 존재

$$V^\pi(s) = E_{s_0=s, \tau \sim \pi} \left[\sum_{t=0}^T \gamma^t r_t \right], \quad Q^\pi(s, a) = E_{s_0=s, a_0=a, \tau \sim \pi} \left[\sum_{t=0}^T \gamma^t r_t \right]$$

- $V^\pi(s)$ 는 목적과 관련해서 현재의 상태 s 가 얼마나 좋은지를 파악하는데 사용되는 반면, $Q^\pi(s, a)$ 는 목적과 관련해서 하나의 상태 s 와 행동 a 쌍이 얼마나 좋은지를 평가하는 목적으로 사용
- 일반적으로 $V^\pi(s)$ 보다 $Q^\pi(s, a)$ 가 더 많이 사용



Reinforcement Learning

- 환경 모형 (environment model)
 - 환경 모형은 상태 전환 함수를 의미, $P(s_{t+1}|s_t, a_t)$ 로 표현
 - 상태 전환 함수는 환경에 대한 정보를 제공
 - 만약 에이전트가 이 함수를 알 수 있다면, 에이전트는 지금의 상태 s 에 대해 행동 a 를 수행하면 (해당 행동을 수행하지 않고도) 어떠한 상태로 전환될 수 있는지를 미리 예측 가능 \Rightarrow 즉, 행동을 직접적으로 수행하지 않고서도 특정한 여러 행동들을 수행했을 때의 결과를 미리 예측할 수 있다는 것을 의미
 - 따라서 이러한 함수를 이용해서 목적을 최대로 하는 행동들을 미리 계획할 수 있다.



Reinforcement Learning

- 강화학습 알고리즘의 종류
 - 일반적으로 강화학습 알고리즘은 무엇을 학습하느냐에 따라 세 가지로 구분
 - 정책 기반 알고리즘, 가치 기반 알고리즘, 모델 기반 알고리즘
- 정책 기반 알고리즘
 - 목적을 최대로 하는 정책을 학습 \Rightarrow 그러한 정책에 의해서 에이전트의 행동이 결정
 - 대표적인 알고리즘: REINFORCE
 - PPO는 REINFORCE 알고리즘을 확장한 것으로 간주 가능 (정확하게는 PPO는 혼합 모형으로 분류)

Reinforcement Learning

■ 정책 기반 알고리즘 (cont'd)

- 딥러닝 기반의 정책 알고리즘들은 신경망 모델을 이용해서 정책을 표현, 즉, 신경망 모델이 정책을 의미
 - 신경망 모델은 파라미터(θ)를 갖는데, 이러한 신경망 모델을 파라미터 θ 를 갖는 정책 신경망이라고 하며, π_θ 라고 표현
 - 학습을 통해서 목적을 최대화하는 방향으로 파라미터를 업데이트하여 더 좋은 정책을 도출
 - 신경망 모델이 정책을 나타내는 경우, 파라미터가 구체적으로 어떠한 값을 갖느냐에 따라 정책이 달라짐
 - 즉, $\theta_1 \neq \theta_2 \Rightarrow \pi_{\theta_1}(s) \neq \pi_{\theta_2}(s)$
 - $\max_{\theta} J(\pi_\theta)$ 즉, $\max_{\theta} E_{\tau \sim \pi_\theta}[R(\tau)]$
 - 경사상승법: $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\pi_\theta)$



Reinforcement Learning

- 가치 기반 알고리즘
 - 가치 기반 알고리즘은 가치 함수를 학습
 - 학습된 가치 함수를 이용해서 특정 상태와 행동을 평가하고 더 좋은 결과를 낼 수 있는 정책을 반환
 - 대표적인 알고리즘: SARSA, Deep Q-Networks (DQN) 등
- 모형 기반 알고리즘
 - 모형 기반 알고리즘은 환경 모형, 즉 상태 전환 함수를 학습
 - 상태 전환 함수를 알게 되면, 행동을 직접적으로 수행하지 않고서도 특정한 여러 행동들을 수행했을 때 혹은 여러 궤적들에 대한 결과를 미리 예측할 수 있다는 것을 의미
 - 이러한 예측을 기반으로해서 목적을 최대로 하는 행동들을 미리 계획 가능
 - 일반적으로 컴퓨터 게임 등에서 자주 사용
 - 최근까지 몬테 카를로 트리 탐색 (Monte Carlo Tree Search, MCTS) 방법이 주로 사용
- 하이브리드 알고리즘
 - 많은 경우, 하이브리드 알고리즘은 정책과 가치 함수를 동시에 학습 \Rightarrow 행위자-평가자(Actor-Critic) 알고리즘



PPO 소개



PPO

- PPO (Proximal Policy Optimization)

- 초기의 정책 기반 알고리즘들(예, REINFORCE)이 갖는 주요한 문제:
성능 붕괴 (performance collapse)
 - 에이전트가 갑자기 결과가 좋지 못한 행동을 하는 것을 의미
 - 다르게 표현하면, 정책에 따라 선택되어진 행동의 결과가 이전 단계에서의 결과 보다 나빠지는 것을 의미
 - 강화학습의 경우, 한 번 이렇게 성능이 갑자기 나빠지게 되면 이후의 단계에서 회복하는 것이 많이 어려움
- 주요 이유
 - 파라미터 값의 차이(업데이트의 정도)가 정책을 통해 도출되는 성능의 차이와 매칭이 되지 않기 때문 \Rightarrow 예를 들어, 파라미터 값의 차이가 작을지라도 (파라미터가 조금만 업데이트될 지라도) 파라미터값의 업데이트로 달라진 정책에 의해 반환되는 결과에는 큰 차이 존재 가능
 - 이러한 차이는 경우에 따라 성능 저하로 이어짐



PPO

- 성능 붕괴 해결책
 - 이를 해결하기 위해서 직접적으로 파라미터가 업데이트 되는 정도를 제한하는 것은 효과적이지 못함
 - 대신, 파라미터 업데이트로 인한 정책 변화에 따른 성능 변화를 목적함수에 반영하는 것이 필요.
 - 그리고 이러한 목적함수를 이용해서 성능 변화가 너무 크지 않고, 성능이 좋아지는 방향으로 업데이트가 진행될 수 있도록 하는 것이 필요



PPO

- 목적함수 도출
 - 이러한 목적함수를 도출하기 위해서는 일단 두 정책 간의 성능 차이를 측정할 수 있는 지표가 존재해야 함
 - 이를 위해, 상대적 정책 성능 비교 지표(relative policy performance identity) 사용, 다음과 같이 정의
 - $J(\pi') - J(\pi) = E_{\tau \sim \pi'} [\sum_{t=0}^T \gamma^t A^\pi(s_t, a_t)]$
 - π 는 현 정책을, π' 는 업데이트된 정책을 의미
 - $A^\pi(s_t, a_t)$ 는 어드밴티지(advantage)라고 표현되며, Q 가치함수와 V 가치함수의 차로 계산, $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$
 - $J(\pi') - J(\pi)$ 를 최대화 하는 것은 $J(\pi')$ 를 최대화 하는 것과 동일하기 때문에 $J(\pi')$ 대신 $J(\pi') - J(\pi)$ 를 목적함수로 사용 가능
 - 하지만, 한 가지 문제 존재
 - 새로운 정책(π')으로부터의 궤적(τ) (즉, $\tau \sim \pi'$)에 대한 기댓값
 - 하지만, 업데이트를 하지 않은 상황에서는 업데이트된 정책(π')으로부터의 궤적을 사용 불가능 $\Rightarrow J(\pi') - J(\pi)$ 를 현재 정책(π)을 이용해서 표현해야 함

PPO

■ 목적함수 도출 (cont'd)

- $J(\pi') - J(\pi) = E_{\tau \sim \pi'} [\sum_{t=0}^T \gamma^t A^\pi(s_t, a_t)]$ 를 현재 정책을 이용해서 표현하기 위해, π 와 π' 차이가 크지 않다고 가정
 - π 가 반환하는 확률 분포와 π' 가 반환하는 확률 분포의 차이가 크지 않다는 것
 - 두 분포의 차이는 KL 발산을 이용해서 표현, 이를 이용해서 차 통제 가능
 - π 와 π' 차이가 크지 않다면, $E_{\tau \sim \pi'} [\sum_{t=0}^T \gamma^t A^\pi(s_t, a_t)]$ 을 π 으로부터의 궤적을 이용해서 근사하는 것이 가능

알려진 확률 분포를
이용해서 표집된
데이터를 이용해서
알지 못하는 분포를
추정하는 방법

→ 중요도 표집 가중치(importance sampling weights)를 사용, $\frac{\pi'(a_t|s_t)}{\pi(a_t|s_t)}$

$$\begin{aligned} \blacksquare J(\pi') - J(\pi) &= E_{\tau \sim \pi'} [\sum_{t=0}^T A^\pi(s_t, a_t)] \approx E_{\tau \sim \pi} \left[\sum_{t=0}^T A^\pi(s_t, a_t) \frac{\pi'(a_t|s_t)}{\pi(a_t|s_t)} \right] = \\ &J_\pi^{SO}(\pi') \end{aligned}$$



PPO

- 대리 목적 (surrogate objective) 함수: $J_{\pi}^{SO}(\pi')$
 - 성능 붕괴의 문제를 해결하기 위해서 원래의 목적함수를 사용하지 않고, $J_{\pi}^{SO}(\pi')$ 를 대신 사용
 - 하지만, $J_{\pi}^{SO}(\pi')$ 는 $J(\pi') - J(\pi)$ 의 근사치이기 때문에 둘 간에는 오차 존재
 - 오차가 너무 커지게 되면 학습이 제대로 되지 않기 때문에 오차를 제한하는 것이 필요 \Rightarrow 이를 위해서 KL 발산을 사용
 - $|(J(\pi') - J(\pi)) - J_{\pi}^{SO}(\pi')| \leq C \sqrt{E_t[KL(\pi'(a_t|s_t)||\pi(a_t|s_t))]}$
 - C 는 상수
 - 위 식은 만약 π 와 π' 가 반환하는 확률 분포의 차이가 크지 않다면, 두 분포 간의 KL 발산이 작다는 것을 의미하고 이는 위 식의 오른쪽 항의 값이 작다는 것 \Rightarrow 즉, $J_{\pi}^{CPI}(\pi')$ 가 $J(\pi') - J(\pi)$ 을 잘 근사한다는 것을 의미

PPO

- 대리 목적 (surrogate objective) 함수 (cont'd)

- 앞의 식을 이용하면 위 식을 이용하면 $J(\pi') - J(\pi) \geq 0$ 을 보장하는 업데이트 방식이 가능
- 이를 위해, 위 식을 아래와 같이 다시 표현

- $$J(\pi') - J(\pi) \geq J_{\pi}^{SO}(\pi') - C \sqrt{E_t[KL(\pi'(a_t|s_t) || \pi(a_t|s_t))]}$$

- 이 식을 이용해서 업데이트를 특정 조건이 만족하는 경우에만 수행

- 업데이트를 했을 때 성능이 좋아지지 않는 경우에는 $\pi' = \pi$ 로 설정하고, 업데이트를 진행하지 않음

- $J_{\pi}^{SO}(\pi') = E_{\tau \sim \pi} \left[\sum_{t=0}^T A^{\pi}(s_t, a_t) \frac{\pi'(a_t|s_t)}{\pi(a_t|s_t)} \right]$ 의 값이 0 & KL 발산의 값이 0 $\Rightarrow J(\pi') - J(\pi) \geq 0$



PPO

- 대리 목적 (surrogate objective) 함수 (cont'd)

- $J_{\pi}^{SO}(\pi') > C \sqrt{E_t[KL(\pi'(a_t|s_t)||\pi(a_t|s_t))]}$ 인 경우에만 업데이트를 수행하게 되면 $J(\pi') - J(\pi) \geq 0$ 가 보장
- 이를 위해 풀고자 하는 최적화 문제에 $C \sqrt{E_t[KL(\pi'(a_t|s_t)||\pi(a_t|s_t))]}$ 를 페널티 항으로 포함
 - $\operatorname{argmax}_{\pi'} \left\{ J_{\pi}^{SO}(\pi') - C \sqrt{E_t[KL(\pi'(a_t|s_t)||\pi(a_t|s_t))]} \right\}$
 - 위의 최적화 문제는 $J(\pi') - J(\pi) \geq 0$ 을 보장 & No performance collapse
 - 참고로, $J(\pi') - J(\pi) \geq 0$ 을 단순 성능 향상(monotonic performance improvement)이라고 표현



PPO

- 실제 구현의 경우
 - 앞의 페널티를 그대로 사용하지 않고, KL 발산의 기댓값을 직접적으로 제한하는 방법을 사용
 - $E_t[KL(\pi'(a_t|s_t)||\pi(a_t|s_t))]\leq \delta$
 - δ 는 하이퍼파라미터
 - KL 발산은 π 와 π' 의 차이를 나타내기 때문에 δ 값이 작을수록 π 와 π' 의 차이가 작아진다는 것을 의미
 - 일반적으로 δ 의 값을 작게 설정해서 업데이트된 정책(π')이 현재 정책(π)과 큰 차이가 없도록 함 \Rightarrow 즉, π' 의 후보로 π 의 근처에 존재하는 정책들만이 고려
 - 새로운 정책 고려의 대상이 되는 현 정책 주변을 신뢰 영역(trust region)이라고 표현
 - $E_t[KL(\pi'(a_t|s_t)||\pi(a_t|s_t))]\leq \delta$ 는 신뢰 영역을 한정하는 역할



PPO

- 대리 목적 함수

- $J_{\pi}^{SO}(\pi') \Rightarrow J^{SO}(\theta) = E_t \left[\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \cdot A_t^{\pi_{\theta_{old}}} \right]$
- 따라서 최종적인 최적화 문제는 아래와 같이 표현

- $$\max_{\theta} E_t \left[\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \cdot A_t^{\pi_{\theta_{old}}} \right]$$
$$\text{subject to } E_t \left[KL \left(\pi_{\theta}(a_t|s_t) || \pi_{\theta_{old}}(a_t|s_t) \right) \right] \leq \delta$$

- 신뢰 영역 최적화 문제라고 표현

- 여러 알고리즘들 제안

- 자연 정책 경사 (Natural Policy Gradient, NPG), 신뢰 영역 정책 최적화 (Trust Region Policy Optimization, TRPO) 등 \Rightarrow 이론적으로 복잡하여 구현이 어렵다는 단점
- 이러한 단점을 보완하기 위해 PPO 제안



PPO

- PPO (Proximal Policy Optimization)
 - 구현이 쉽고, 많은 컴퓨팅 파워를 필요로 하지 않고, δ 의 값을 직접적으로 설정할 필요 없다는 장점
 - 두 가지 버전
 - 적응적 KL 페널티(adaptive KL penalty) 방법
 - 목적함수에 적응적 KL 발산을 페널티항으로 추가한 방법
 - InstructGPT에서 사용
 - 한정된 목적(clipped objective) 방법
 - 목적함수가 취할 수 있는 구간을 한정(clipping)하는 방법



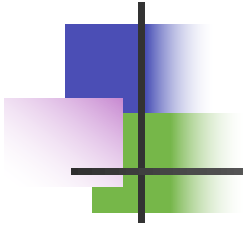
PPO

- 적응적 KL 페널티(adaptive KL penalty) 방법
 - 목적함수에 적응적 KL 발산을 페널티항(아래 식에서 $\beta KL(\pi_{\theta}(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t))$)으로 추가한 방법
 - 적응적 KL 페널티 방법에서의 최적화 문제는 아래와 같이 표현
 - $\max_{\theta} E_t \left[r_t(\theta) A_t - \beta KL(\pi_{\theta}(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t)) \right]$
 - 여기에서 $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, $A_t = A_t^{\pi_{\theta_{old}}}$ 을 의미
 - β 는 적응 계수 (adaptive coefficient)로 KL 페널티의 크기를 조절 \Rightarrow 이는 신뢰 영역을 조절하는 역할, 즉, 새로운 정책이 기존의 정책과 얼마나 달라질 수 있는지를 조절하는 역할 수행
 - β 이 클수록, 더 넓은 영역 (즉, 차이가 커질 수 있다)
 - 이러한 방법은 적절한 β 의 값을 설정하는 것이 어렵다는 단점 존재



PPO

- 적응적 KL 페널티(adaptive KL penalty) 방법 (cont'd)
 - 이러한 문제를 해소하기 위해 PPO 논문에서는 정책 파라미터 업데이트시 특정한 기준에 따라 β 의 값을 조절하는 방법을 사용
 - $KL(\pi_{\theta}(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t))$ 의 목표값(d_{target})을 설정하고 그 값과 업데이트시에 계산된 $KL(\pi_{\theta}(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t))$ 값 (d)을 비교하여 β 의 업데이트 값을 결정
 - $d < \frac{d_{target}}{1.5}$ 인 경우에는 $\beta \leftarrow \beta/2$ 로, $d > d_{target} \times 1.5$ 인 경우에는 $\beta \leftarrow 2\beta$ 로 업데이트



Q & A