3주차 PyTorch로 시작하는 강화학습 입문 Camp

강의 내용

- DQN으로 풀 수 **있는** 문제 들

  : 주어진 상태에서 가장 큰 가치함수 값을 가지는 행동을 빠르게 고를 수 있는 문제. 행동과 상태 집합의 크기가 그리 크지 않다. TD target 값을 빠르게 구할 수 있다.

  TD target - $r_t^{*(n)} + \gamma \max_{a \in A} Q_\theta(s_{t+1}, a)$

- DQN으로 풀 수 **없는** 문제 들

  : 고를 수 있는 행동이 연속적인 경우 (행동 집합이 연속적인 경우)

  : 고를 수 있는 행동이 많은 경우 (또는 무한한 경우) ... Max 값을 구하기가 힘들기 때문이다.

$\rightarrow$ 오늘 배울 Policy gradient 방법은 상대적으로 행동이 많거나 연속적인 MDP 문제에 대해서 더 적합하다.(다른 말로 High dimension, continuous action 에 대해서 nice하다 표현)

- 정책 분포를 표현하는 방법
  - Action 유한 한 경우 - Softmax로 정책을 표현
  - 행동집합이 연속적인 경우 Gaussian으로 정책 표현
  - Implicit distribution - 4주차에 다룰 예정

Policy Optimization

: MDP 문제에서 RL의 목적은 최적의 정책을 찾는 것이고 최적의 정책은 다음 목적함수를 만족하는 것이다.

$$\max_\theta J(\pi_\theta) \doteq \max_\theta \mathbb{E}_{\pi_\theta}[R_0] = \mathbb{E}_{\pi_\theta^*}[R_0]$$

$\longrightarrow$ **현재 정책을 기준으로 조금씩 개선(정책에 대한 경사) 하다보면 최적정책을 근사할 수 있지 않을까 ?** Policy Gradient

- **정책 경사**

  : MDP의 목적함수(방금의 수식; 초기 수확의 기댓값이 가장 큰 정책)를 증가시키는 모수의 방향

  $$\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbb{E}_{\pi_\theta}[R_0]$$
  $$= \frac{1}{1-\gamma} \mathbb{E}_{\pi_\theta}[Q^\pi(s,a) \nabla_\theta \log \pi_\theta(a|s)]$$

- 정책경사 정리
  - Stationary distribution

    고정된 정책에 따라 많은 샘플링을 하면 이전 시점과 현 시점의 확률분포가 같아지도록 수렴하는 마코프체인에서 그 확률분포를 stationary distribution이라고 한다. 전체 시간동안 에이전트가 특정 상태에 얼마나 머물렀는지 알려준다.

    $$d^\pi(s) = (1-\gamma)\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t 1_{s_t=s}] = (1-\gamma) \sum_{t=0}^{\infty} P_{\pi_\theta}(s_t = s)$$

$\sum d^{\pi}(s) = 1$

- paper link : 목적함수 그레디언트 정리

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta V^\pi(s_0)$$
$$\vdots$$
$$= \frac{1}{1-\gamma}\mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)]$$

- 정책을 증가시키는 모수의 방향(θ)은 가치가 높은 행동이 선택될 확률을 높이는 방향이다.

  이를 계산하는 방법으로 REINFORCEMENT가 가장 첫번째이다.

Policy Gradient 계산

- REINFORCE

  1. 현재 정책으로 데이터를 만들고, 시뮬레이션으로 얻은 수확 값으로 그레디언트를 계산한다.

  $$R_t^{(n)} = \sum_{k=0}^{\infty} \gamma^k r_{t+k}^{(n)}$$

  $$\hat{\nabla}_\theta J(\pi_\theta) \approx \frac{1}{1-\gamma}\frac{1}{|M|}[\nabla_\theta \log \pi_\theta^{(n)}(a|s) R_t^{(n)}]$$

  2. 계산된 Policy gradient로 θ 를 갱신 한다.

  $$\theta^{(n+1)} \rightarrow \theta^{(n)} + \eta \cdot \hat{\nabla}_\theta J(\pi_\theta)$$

  :: 데이터를 만드는 정책과 가치함수(여기서는 수확으로)를 계산하는 정책이 같으므로 On-policy learning

- Baseline

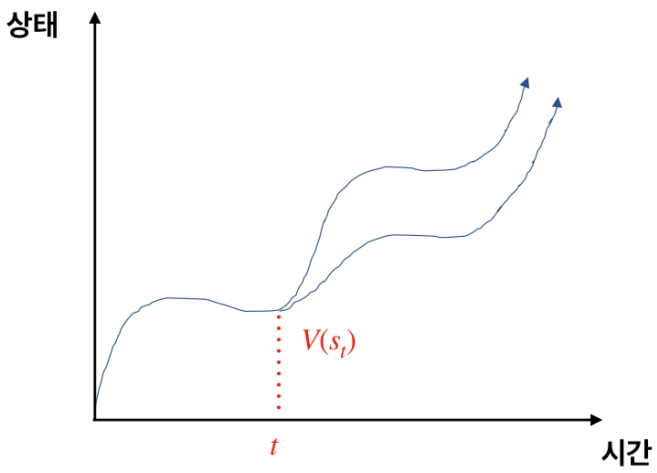
  - 정책경사의 문제점

    On-policy learning 이기 때문에 현재 정책이 만든 데이터는 다시 사용하지 못한다. 그래서 현재 정책이 만든 데이터 하나로 한번 가중치를 갱신하게되는데 한 번의 시뮬레이션이 수확의 평균을 잘 측정한다고 하기 어렵고 무엇보다 시점이 늘어날 수록 분산이 커진다는 문제가 있다.

  $\rightarrow$ 분산을 줄이기 위한 방법으로 Baseline 방법은 가치함수 값에서 b 라는 바이어스를 빼주는 방법을 사용한다.

  $$\nabla_\theta J(\pi_\theta) = \frac{1}{1-\gamma}\frac{1}{|M|}[\nabla_\theta \log \pi_\theta(a|s)(Q^\pi(s,a) - b)]$$

  <시간이 지나면서 분산이 커지는 예>



$$\nabla_\theta J(\pi_\theta) = \frac{1}{1-\gamma}\mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s)(Q^\pi(s,a) - b(s))]$$
$$= \frac{1}{1-\gamma}\mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s)(Q^\pi(s,a))] - \frac{1}{1-\gamma}\mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s)(b(s))]$$
$$= \frac{1}{1-\gamma}\mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s)(Q^\pi(s,a))] - \frac{1}{1-\gamma}\sum_{s\in S}\sum_{a\in A} d^\pi(s)\pi(a|s)\nabla_\theta \log \pi_\theta(a|s)b(s)$$
$$= \frac{1}{1-\gamma}\mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s)(Q^\pi(s,a))] - \frac{1}{1-\gamma}\mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s)(b(s))]$$
$$= \frac{1}{1-\gamma}\mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s)(Q^\pi(s,a))] - \frac{1}{1-\gamma}\sum_{s\in S} d^\pi(s)\sum_{a\in A}\nabla_\theta\pi(a|s)b(s)$$
$$= \frac{1}{1-\gamma}\mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s)(Q^\pi(s,a))] - \frac{1}{1-\gamma}\sum_{s\in S} d^\pi(s)\nabla_\theta\sum_{a\in A}\pi(a|s)b(s)$$

여기서 Bias는 θ와 무관하기 때문에 Gradient의 Appoximation이 동일하다.

$$= \frac{1}{1-\gamma}\mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s)(Q^\pi(s,a))]$$

그리고 실제로 분산도 작아진다. paper link

Actor Critic

시뮬레이션의 길이가 길어지면 분산은 자연스레 증가한다. Actor Critic에서는 가치함수 따로 학습하여 실제 수확 대신 학습한 가치함수 값으로 그레디언트를 계산한다. 중간에서 그레디언트를 계산한다.

$$\nabla_\theta J(\pi_\theta) = \frac{1}{1-\gamma}\mathbb{E}_\pi[\nabla_\theta \log \pi_\theta(a|s)Q_\phi(s,a)]$$

- **Advantage Actor Critic**

  Baseline 방법을 더하고 Bias 값을 학습하는 상태가치 함수값을 사용하면 수식은 다음과 같고 행동가치함수에서 상태가치함수를 빼면 행동에 대한 이익함수(Advatage) 함수가 된다.

  $$\nabla_\theta J(\pi_\theta) \approx \frac{1}{1-\gamma}\frac{1}{|M|}\sum_{i\in M}[\nabla_\theta \log \pi_\theta(a|s)(Q_\phi(s,a) - V_\psi(s_i))]$$

이렇게 되면 두 개의 가치함수에 대해 학습이 필요할텐데 가치함수 간의 관계를 이용하면 다음처럼 나타내고 상태가치함수 하나만 학습하면 된다.

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{1-\gamma}\frac{1}{|M|}\sum_{i\in M}[\nabla_\theta \log \pi_\theta(a|s)(r_i + V_\psi(s_{i+1}) - V_\psi(s_i))]$$

여기서 계산된 PG를 이용해 파라미터를 갱신한다.

Generalized Advantage Estimator (GAE)

MC장점 : 모든 s,a 쌍에 대해 영향을 주고 이점이 학습 수렴에 도움을 줄 수 있다.

MC단점 : 에피소드의 끝까지 가야하기 때문에 분산이 커진다. 그리고 중간에 학습을 할 수 없다.

TD장점 : 중간에 학습을 할 수 있고, 분산이 작아진다.

TD단점 : 딱 하나의 수확으로 그 때 s,a에 영향을 주고 그게 첫번째 s에 대해 영향을 못준다. long-term한 시그널을 줄 수 없다.

... 각각의 장점을 아우룰 수 있는 것 - GAE

- TD n
  - n-step consistency

    $A^\pi(s_t, a_t) \simeq r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t) = \delta_t^1$

    $A^\pi(s_t, a_t) \simeq r_t + \gamma r_{t+1} + \gamma^2 V^\pi(s_{t+2}) - V^\pi(s_t) = \delta_t^2$

    $A^\pi(s_t, a_t) \simeq r_t + \cdots + \gamma r_{t+n-1} + \gamma^n V^\pi(s_{t+n}) - V^\pi(s_t) = \delta_t^n$

    δ_t^n 는 t+n 시점의 데이터가 주는 정보를 뜻한다.
- TD λ 위의 δ_t^n 를 각 $\lambda^{(n-1)}$ 곱하여 더하면 (기하평균)

  $\rightarrow (\delta_t^1 \times \lambda^0 + \delta_t^2 \times \lambda^1 + \cdots + \delta_t^n \times \lambda^{n-1}) \simeq \frac{1-\lambda^n}{1-\lambda} A^\pi(s_t, a_t)$

  이를 재배열하면

  $$\frac{1-\lambda^n}{1-\lambda} A^\pi(s_t, a_t) = \gamma^0\lambda^0(\lambda^0 + \lambda^1 + \cdots + \lambda^{n-1})\delta_t^1 + \gamma\lambda\frac{1-\lambda^{n-1}}{1-\lambda} A^\pi(s_{t+1}, a_{t+1})$$

  $$A^\pi(s_t, a_t) = \gamma_t^1 + \gamma\lambda\frac{1-\lambda^{n-1}}{1-\lambda} A^\pi(s_{t+1}, a_{t+1})$$

  마지막 수식이 Generalized Advantage Extimator이다. paper link
- Advantage actor critic(A2C) with GAE

  현재 정책으로 만든 에피소드로 GAE 가치함수의 target을 다음 처럼 만든다.

  $A^\pi(s_i, a_i) = \delta_i^1 + \gamma\lambda A^\pi(s_{i+1}, a_{i+1})$

  그리고 상태가치 함수를 학습한다.

  $$\min_\psi \sum_{i \in M} (r_i + \gamma A^\pi(s_{i+1}, a_{i+1}) + \gamma V_\psi(s_{i+1}) - V_\psi(s_i))^2$$

  위에서 구한 수확을 기반으로 PG를 계산한다.

  $$\nabla_\theta J(\pi_\theta) \approx \frac{1}{1-\gamma}\frac{1}{|M|}\sum_{i \in M}[\nabla_\theta \log \pi_\theta(a|s)(A^\pi(s_{i+1}, a_{i+1}))]$$

  이 그레디언트로 파라미터를 업데이트한다.

Proximal Policy Optimization (PPO)

Policy Gradient에서 Policy evaluation 와 Policy Improvement

-TD target을 만들고 오차를 줄이며 가치함수를 갱신

-정책경사정리에 따라서 더 좋은 행동의 확률을 높이는 방향을 학습

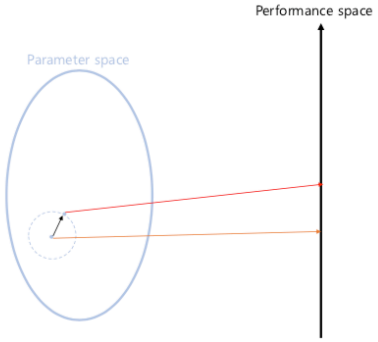
이 때, 정책경사정리 가 실제로 더 좋은 정책을 만들지 않을 수 있다. 파라미터가 업데이트하면서 리턴도 바꾸지만 정책도 바꾸고 이에 따라 의도치 않은 변경을 불러올 수 있다. .... 변화량의 제약을 두는 법의 필요성(Regularization)

- **Natural Policy Gradient**

파라미터의 구를 설정하고 그 구안에서 가장 빠르게 목적함수가 증가하는 방향으로 정책을 갱신한다.

$$\nabla_\theta J(\pi_\theta) = \arg \max_{\Delta:||\Delta||\leq\delta} J(\pi_{\theta+\Delta}) \approx \arg\max J(\pi_\theta) + \Delta^\intercal \nabla_\theta J(\pi_\theta) + \frac{1}{2}\Delta^\intercal\Delta$$

<예시>



$$\nabla_\theta J(\pi_\theta) = \arg \max_{\Delta:D_{KL}(\pi_\theta||\pi_{\theta+\delta})\leq\delta} J(\pi_{\theta+\Delta})$$

이전 분포와 갱신된 분포 사이의 거리가 커지지 않게 해준다. (Numerical Stability)

$$\nabla_\theta J(\pi_\theta) = \arg \max_{\Delta:D_{KL}(\pi_\theta||\pi_{\theta+\delta})\leq\delta} J(\pi_{\theta+\Delta})$$

$$\approx \arg\max J(\pi_\theta) + \Delta^\intercal \nabla_\theta J(\pi_\theta) + \frac{1}{2}\Delta^\intercal F_\theta \Delta$$

$$= F_\theta^{-1}\nabla_\theta J(\pi_\theta)$$ ... **Natural Gradient** 이다.

paper link

$$F_\theta^{-1}\nabla_\theta J(\pi_\theta) = F_\theta^{-1}\nabla_{\theta'}\mathbb{E}_{s\sim d^\pi, a\sim\pi_{\theta'}}[A^\pi(s,a)]$$

$$F_\theta^{-1}\nabla_\theta J(\pi_\theta) = F_\theta^{-1}\nabla_{\theta'}\mathbb{E}_\pi\left[\frac{\pi_{\theta'}(s,a)}{\pi_\theta(s,a)}A^\pi(s,a)\right]$$

현재 Policy에서 Critic 이 알려주는 좋은 정도 만큼 증가한다.

- Natual Policy Gradient 의 계산

  Fisher Information Matrix를 계산하는 것

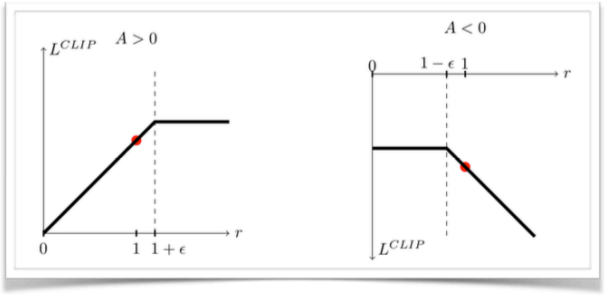
  - 매우 큰 행렬의 평균(네트워크 size x 네트워크 사이즈)이기에 표본이 많이 필요
  - 역행렬 계산이 필요-- 다른 알고리즘이 더 필요하다.
    - K-FAC으로 근사하게 되면 (Natual gradient를 구하는 알고리즘) → ACKTR
    - Conjugate Gradient Descent(선형식의 의사역행렬을 구하는 알고리즘) → TRPO
- Proximal Policy Optimization

  Natural Gradient를 사용하는 것은 계산이 복잡하고 시간도 많이 소요된다.

  원래 TRPO의 목적함수 에서 실제로 Regulization 역할을 하는 것은 $r_t(\theta) = \frac{\pi_{\theta'}(s,a)}{\pi_\theta(s,a)}$ 이다. 이 비율을 Clip 하게 되면 A_t가 0보다 클 때는 갱신 후를 많이 키워도 효과가 적을 것이고 A_t가 0보다 작을 때는 갱신 후를 작게 해도 효과가 적을 것이다.

  <Clip 효과>

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$



PPO를 GAE와 함께 사용하는 과정은 현재 정책에 따라 데이터를 얻고 GAE를 통해 만든 가치함수의 target

$$A^{\pi}(s_i, a_i) = \delta_i^1 + \gamma\lambda A^{\pi}(s_{i+1}, a_{i+1})$$

과 상태 가치 함수로 이루워진 TD-error를 최소화 하도록 학습하고,

$$\min_{psi} \sum_{i \in M} (r_i + \gamma A^{\pi}(s_{i+1}, a_{i+1} + \gamma V_{\psi}(s_{i+1}) - V_{\psi}(s_i))^2$$

PPO 목적함수를 최대화 한다.

$$\max_{\theta} \mathbb{E}_{\pi}[\min(r_t(\theta) A^{\pi_o ld}(s,a), clip(r_t(\theta), 1-\epsilon, 1+\epsilon) A^{\pi_o ld}(s,a))]$$

<MuJoCo 환경에서 각 알고리즘의 퍼포먼스 비교>

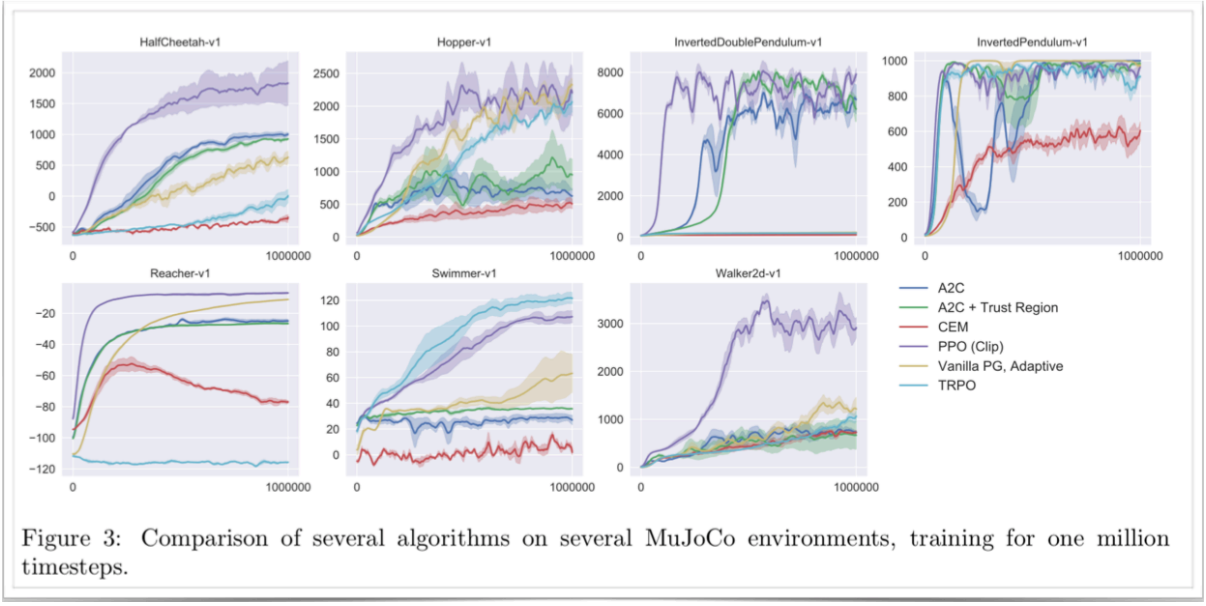


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

질문

- 행동집합이 더 넓다면 Gaussian을 써도 되는 건가 ?

  답 :표준편차를 조절하면 넓은 영역도 커버가 가능하다.

- (정책 경사 정리 (PG)부분에서) 현재 정책에 대한 가치함수를 기준으로 학습을 하는 것인가

  답 : 현재 정책으로 실행한 데이터에 대해서 현재 정책의 가치함수가 높은 방향으로 정책을 갱신한다.

- PG의 계보를 알려주세요

  답 : PG - NPG, AOARL - TRPO(2015),ACKTR - PPO - DDPG, D4PG

- 실제 데이터가 변화가 급격한, 분산이 큰 데이터의 경우에 Actor Critic을 적용해도 괜찮은가?

답 : 우리가 풀고 있는 것은 MDP문제, 즉 각 행동에 따른 전이 확률이 알든 모르든 고정되어 있는 문제를 풀고 있는 것이기 때문에 그런 상황에서는 가정 자체가 다르다.