4주차 PyTorch로 시작하는 강화학습 입문 Camp

출석 : 10명 (조교 2명 제외)

강의 내용

On-Policy vs Off-Policy Learning

1. On-Policy Learning

   환경을 실행(데이터를 모으는 정책 또는 모아져 있는 데이터를 만든 정책; 보통(s,a,r,s')로 주어짐)하는 정책과 학습(QL, PG, PO 등으로 업데이터하는 정책, 데이터 생성에 적용가능한 정책, 대체로 모든 정책 $\pi(a|s)$ 을 알고 있다.)하는 정책이 같은 경우

   ... **A2C, TRPO, PPO**

2. Off-Policy Learning

   환경을 실행하는 정책과 학습하는 정책이 다른 경우, 여기서

   환경을 실행하는 정책 : Behavior policy

   학습하는 정책 : Current policy 이라고 보통 표현한다.

   ... **DQN** : Behavior policy는 ϵ-greedy policy , Current policy : 현재 가치 함수에 대해 greedy policy

3. Off-Policy learning의 필요성
   1. 데이터를 효율적으로 사용 - 현재 정책으로 만든 데이터만 사용하는 On-policy에 비해 과거/전문가의 정책으로 만든 데이터를 사용가능하기 때문에 데이터를 효율적으로 사용가능하다.
   2. 탐험으로 발견한 높은 수확의 데이터를 여러번 학습 할 수 있다.
   3. 전문가, 학습을 마친 알고리즘의 정책이 만든 데이터를 기반으로 학습이 가능하다.

Retrace Lambda

GAE에서 설명 했던 TDλ 와 유사하다. 차이점은 Retrace Lambda는 Action value function를 추정하기 위해서 n-step consistency를 t-consistency로 변환한다. 행동 가치 함수의 Lambda Backup은 다음 수식과 같다.

$$Q_\lambda^\pi(s_t,a_t) = Q^\pi(s_t,a_t) + \gamma\lambda Q_\lambda^\pi(s_{t+1},a_{t+1}) + (r_t + \gamma Q^\pi(s_{t+1},a_{t+1} - Q^\pi(s_t,a_t))$$

수식을 글로 풀면, 지금까지 학습한 가치함수값에 에피소드 끝부터 계산해온 다음 시점의 가치함수값의 lambda backup을 더하고 다음 시점의 Temporal Difference를 더한 것이 행동 가치 함수의 lambda backup 이다. 위 식을 t의 끝부터 더한 것을 표현하는 Notation으로 변환하면 다음과 같다.

$$Q_\lambda^\pi(s_t,a_t) = Q^\pi(s_t,a_t) + \mathbb{E}_\pi[\sum_{n=0}^{\infty}(\gamma\lambda)^n(r_{t+n} + \gamma\mathbb{E}_\pi Q(s_{t+n+1},a_{t+n+1}) - Q(s_{t+n},a_{t+n}))]$$

위 수식에서 현재 정책이 만든 데이터에 대한 가치함수가 대신 Importance sampling을 사용하면 다음 수식과 같다. 다음 수식에서 $\frac{\pi(s_{t+s},a_{t+s})}{\mu(s_{t+s},a_{t+s})}$ 부분은 정책분포를 데이터를 만든 정책분포 π 에서 μ 로 바꾸었기에 곱해주는 것으로 이렇게 하면 결과적으로 데이터를 만든 정책 분포에 대한 기댓값을 구할 수 있다.

$$Q_\lambda^\pi(s_t,a_t) = Q^\pi(s_t,a_t) + \mathbb{E}_\mu[\sum_{n=0}^{\infty}(\prod_{s=1}^{n}\gamma\lambda\frac{\pi(s_{t+s},a_{t+s})}{\mu(s_{t+s},a_{t+s})})(r_{t+n} + \gamma\mathbb{E}_\pi Q(s_{t+n+1},a_{t+n+1}) - Q(s_{t+n},a_{t+n}))]$$

여기서 데이터를 만들 때 사용하는 정책 분포μ 는 과거의 정책, 전문가의 정책등을 사용할 수 있다.

다만, $\frac{\pi(s_{t+s},a_{t+s})}{\mu(s_{t+s},a_{t+s})}$ 은 $0 < \frac{\pi(s_{t+s},a_{t+s})}{\mu(s_{t+s},a_{t+s})} < \infty$ 로 분산이 커질 수 있다. 이 점을 보완하기 위해 다음 처럼 최댓값을 1로 clipping 해준다.
$\min(\frac{\pi(s_{t+s},a_{t+s})}{\mu(s_{t+s},a_{t+s})}, 1)$ 이를 **Truncated Importance weight**라고 부른다. 따라서 행동가치함수를 추정하는 Retrace Lambda의 최종 수식은 다음과 같다.

$$Q_\lambda^\pi(s_t,a_t) = Q^\pi(s_t,a_t) + \mathbb{E}_\mu[\sum_{n=0}^{\infty}(\prod_{s=1}^{n}\gamma\lambda\min(\frac{\pi(s_{t+s},a_{t+s})}{\mu(s_{t+s},a_{t+s})}, 1))(r_{t+n} + \gamma\mathbb{E}_\pi Q(s_{t+n+1},a_{t+n+1}) - Q(s_{t+n},a_{t+n}))]$$

Actor-Critic with Experience Replay (ACER)

: 행동가치함수를 Off-Policy Learning으로 학습하고(Retrace Lambda), PG를 Off-policy Learning으로 계산 할 수 있어야한다. 다음 수식은 Actor-Critic 방식으로 정책의 경사를 계산하는 식이다.

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

위 수식에서 $Q^\pi(s,a)$ 는 Retrace Lambda로 추정했다. 이 때 정책경사를 계산하면서도 현재 정책이 아닌 다른 정책을 사용하기위해서 Importance weight같은 방식으로 다음처럼 근사 할 수 있겠다.

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

위 식을 **Off-policy policy gradient theorem** 이라고 하였다. 그리고 다음 수식 전개에서 $\frac{\pi(a|s)}{\mu(a|s)}$ 를 $\rho(a|s)$라고 표현 할 것이다.

$\rho(a|s)$ 또한 분산이 매우 클 수 있기에 ACER에서는 이를 다음과 같은 방식으로 Cliping 한다.

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^\mu, a\sim\mu}[\min(\rho(a|s), c) \nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)] + \mathbb{E}_{s\sim d^\mu, a\sim\mu}[\max(\frac{\rho(a|s)-c}{\rho(a|s)}, 0) \nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,$$

c 는 보통 1로 설정한다. $\rho(a|s)$ 의 의미를 생각해보면, 현재 정책이 이전 정책에 비해 얼마나 크냐, 혹은 작으냐를 나타내는 것이다. $\rho(a|s)$ 가 1보다 작다는 의미는 현재 정책은 특정 상태에서 특정행동을 할 확률이 작은데 전문가/과거의 데이터를 만든 정책은 그 확률이 크다고 하는 경우이다. 이 때는 수식의 뒷부분이 max 연산 때문에 없어진다. 그 반대의 경우에는 수식의 앞부분이 min 연산으로 c(1)로 clipping 되고, 뒷부분에 1보다 작은 값이 각 정책 경사에 곱해진다.

Soft MDP

- Multi-Armed Bandit(MAB) 문제

  : state는 하나 , Action 은 여러개 인 문제

  ex> 어떤 슬롯 머신을 당길지 정하는 문제를 예로 사용

- Randomized selection

  MAB문제에서 Action 선택을 확률적으로 하는 것

  이 때 확률을 고려할 때, 지금까지 얻은 보상에 비례하게 확률분포를 구성한다. 이 때, 각 행동에 대해 지금 까지 얻은 보상의 평균이고 MAB문제 에서는 state가 하나이므로 이는 $Q^\pi(s,a)$ 라고 할 수 있겠다.

- Optimism in the face of Uncertainty

  행동에 대한 최적의 가치함수를

  $$V^*(s) = \max_\pi \mathbb{E}_{a\sim\pi}[Q^*(s,a)] + H(\pi^*(\cdot|s))$$

  로 정의한다. 이 때 엔트로피 항은 다음처럼 정의된다.

  $$\begin{aligned} H(\pi^*(\cdot|s)) &= \sum_{a\in\mathcal{A}} -\pi(a|s)\log\pi(a|s) \\ &= \mathbb{E}_\pi[-\log\pi(a|s)] \end{aligned}$$

  뒤에 엔트로피 항이 추가 되었기 때문에 행동을 선택하는 확률분포가 좀더 평평하게 되어서 더 다양한 시도를 할 수 있다.

  $V^*(s) = \max_\pi \mathbb{E}_{a\sim\pi}[Q^*(s,a)] + H(\pi^*(\cdot|s))$ 이 수식 의 값은 해당 시점에서의 action별 보상들을 softmax 연산 한 것과 거의 같다.

  $\pi^*(a|s) = \frac{\exp(Q^*(s,a))}{\sum_{a\in A}\exp(Q^*(s,a))}$ 로 하고(정책분포를 softmax로 계산하고) $V^*(s)$ 를 계산하면

  $V^*(s) = \log\sum_{a\in A}\exp(Q^*(s,a))$ 가 된다.

  이 때, 가치함수의 정의대로

  $$Q^*(s,a) = r_s^a + \gamma\sum_{s'\in S} V^*(s') T_{s,s'}^a$$

  이를 $\pi*(a|s)$ 식에 대입하면

  $$\pi^*(a|s) = \frac{\exp\left(r_s^a + \gamma\sum_{s'\in S} T_{s,s'}^a V^*(s')\right)}{\exp(V^*(s))},\ \forall s\in S, a\in A$$

  이렇게 해서 **softMDP** 의 최적정책과 최적 가치함수가 만족하는 일관성(consistency)는 다음과 같다.

  $$V^*(s) - \gamma\sum_{s'\in S} T_{s,s'}^a V^*(s') = r(s,a) - \log\pi^*(a|s)$$

  이제 일관성을 만족하도록 error를 줄이는 학습을 하면 된다. 그 방법이 다음 배울 3가지 이다.

Path consistency Learning

다음의 비용함수가 0에 수렴하도록 학습하는 것이 목적이다.

1-step consistency $C(\theta,\phi) = -V_\phi(s) + \gamma V_\phi(s') + r(s,a) - \log\pi_\theta(a|s)$

d-step consistency $C(\theta,\phi) = -V_\phi(s_t) + \gamma^d V_\phi(s_{t+d}) + \sum_{j=0}^{d-1}\gamma^j[r(s_{t+j},a_{t+j}) - \log\pi_\theta(a_{t+j}|s_{t+j})]$

$$\rightarrow \frac{1}{2}C(\theta,\phi)^2 = (-V_\phi(s_t) + \gamma^d V_\phi(s_{t+d}) + \sum_{j=0}^{d-1}\gamma^j[r(s_{t+j},a_{t+j}) - \log\pi_\theta(a_{t+j}|s_{t+j})])^2$$

Soft Actor-Critic

최적 정책 분포, 가치 함수에 대해서

$$Q^*(s,a) = r_s^a + \gamma\sum_{s'\in S}V^*(s')T_{s,s'}^a$$

$$V^*(s) = \max_\pi Q^*(s,a) + \mathbb{E}_\pi[-\log\pi^*(a|s)]$$

$$\pi^*(a|s) = \frac{\exp(Q^*(s,a))}{\sum_{a\in A}\exp(Q^*(s,a))}$$

일반적인 정책 분포, 가치 함수에 대해서는 $Q(s,a) = r_s^a + \gamma\sum_{s'\in S}V(s')T_{s,s'}^a$

$$V(s) = \max_\pi Q(s,a) + \mathbb{E}_\pi[-\log\pi(a|s)]$$

$$\pi^*(a|s) \doteq \frac{\exp(Q(s,a))}{\sum_{a\in A}\exp(Q(s,a))}$$

이를 강화학습으로 바꿔주면,

$$Q(s,a) \approx r_s^a + \gamma V(s')$$
$$V(s) \approx Q(s,a) - \log\pi(a|s)$$
$$\pi^*(a|s) \doteq \frac{exp(Q(s,a))}{\sum_{a\in A}\exp(Q(s,a))}$$

Self-imitation Learning

좋은 데이터를 확보했을 때, 해당 데이터를 잘 학습하면 그 데이터 보다 더 좋은 성적을 내면서 발전하지만 잘 학습하지 못하면 전과 비슷하고 더 좋은 성적을 내지 못한다. 데이터를 확보하는 과정에서 얻은 좋은 데이터를 잘 활용하여 학습하도록 한다.

t 시점에서 상태가치함수와 보상의 관계를 보면, t 시점에서 상태가치함수가 보상보다 크면 데이터를 모은 시점보다 더 높은 성적을 받고, 반대의 경우, 상태가치함수가 보상보다 작으면 데이터를 모은 시점보다 더 낮은 성적을 받는다. 이에 전자는 학습이 필요하지 않고, 후자는 학습이 더 필요한 데이터이다.

$$L^{sil} = \mathbb{E}_{s,a,R\in D}[L_{policy}^{sil} + \beta^{sil}L_{value}^{sil}]$$

$L_{value}^{sil} = \frac{1}{2}||(R - V_\theta(s))||^2$ ... 현재 학습한 것보다 더 좋은 상태임을 알려주고

$L_{policy}^{sil} = -\log_{\pi_\theta}(a|s)(R - V_\theta(s))$ ... 그 행동을 더 하도록 학습 한다.

최적의 행동가치함수는 다음과 같은 하한(Lower bound)를 가진다.

$$\begin{aligned} Q^*(s_t,a_t) &= \mathbb{E}_{\pi^*}[r_t + \sum_{k=t+1}^{\infty}\gamma^{k-t}(r_k + \alpha H_k^\pi)] \\ &\geq \mathbb{E}_\mu[r_t + \sum_{k=t+1}^{\infty}\gamma^{k-t}(r_k + \alpha H_k^\mu)] \end{aligned}$$

최종 비용함수의 형태는 일반정책을 이용해 하한식을 사용하고, 이를 Lower bouned soft Q-Learning 이라고 한다.

$$L = \mathbb{E}_{s,a,R\sim\mu}[\frac{1}{2}||(R - Q_\theta(s,a))_+||^2]$$

알고있는 것보다 좋으면 학습하고 아니면 학습하지 않는다.

질문