5주차 PyTorch로 시작하는 강화학습 입문 Camp

강의 내용

<MAB 문제> concept은 중요한데 성능이 별로 안좋다.

Exploration & Exploitation Dilemma

Exploitation은 현재까지 모은 정보를 최대한 활용해서 움직이는 것, 현재 에피소드에서 가장 높은 보상을 얻는 것을 목표하는데 반면, **Exploration**은 새로운 정보를 얻어 이후 에피소드에서 높은 점수를 얻는 것이 목표이다. 어쨌든, 궁극적으로 우리의 목적은 모든 에피소드에 대해 높은 점수를 얻는 것이다.

Online Decision Making

모든 상황에 대해서 보상이 큰 행동 모델링 하는 것은 어렵기 때문에 Online Decision Making problem이라 정의하는 문제에서는 **하나의 행동만을 하더라도 가장 높은 보상을 얻는 행동을 찾는 것이다.**(여기서 고려하는 것은 MAB문제 이기 때문에 상태가 하나 밖에 없다.)

Regret

: 한 에피소드의 **Regret**(논문에 따라 리그렛 또는 후회로 번역된다.)은 다음 수식으로 정의된다.

$$R^t(a) - \mathbb{E}_{p^t}[R^t]$$

선택할 수 있는 행동의 보상에서 현재 에이전트가 선택한 행동에 따른 보상의 기댓값을 빼주게 된다. 이 값이 클수록 현재 에피소드가 행동 a 에 비해서 주는 보상이 적다는 의미이다.

전체 에피소드의 후회의 평균은 다음처럼 각 에피소드(행동)에서 후회의 평균 값이다. 이 후회가 가장 큰 값에 해당 하는 행동이 좋은 행동이라 할 수 있겠다.

$$\frac{1}{T}\sum_{t=1}^{T} R^t(a) - \mathbb{E}_{p^t}[R^t]$$

MAB 문제 에 대입하면 Regret은 가장 좋은 슬롯머신을 계속 당기지 않은 것에 대한 후회로 가장 좋은 슬롯 머신을 찾는 방법은 정책이 a 를 고르도록 점점 수렴하는 것이고 이는 곧 Regret이 점점 0이 되는 Online Decision Making을 하는 것이다 .

$$\max_{a \in A} \frac{1}{T}\sum_{t=1}^{T} R^t(a) - \mathbb{E}_{p^t}[R^t] \to 0$$ ... No-regret Algorithm

최적정책을 찾는 방법

1. 목적함수의 최대화 :

   DQN(Principle of Optimality with Bellman Cosistency Equation)

   Policy based Methods : Policy evaluation + Greedy Update

2. No-regret 알고리즘 :

   에피소드가 진행되면서 Regret이 0이되는 **Exploration** 전략

후회의 합이 전체 에피소드에 비해 작게 증가하기 때문에 sublinear regret 이라고도 표현한다.

- **Greedy Algorithm**

  $Q^{\pi}(s,a)$ 를 기준으로 하여 Deterministic 하게 **가장 좋은 행동**을 고르는 알고리즘

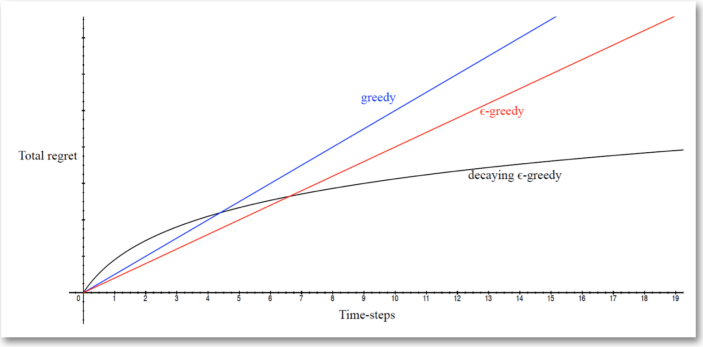
  - Linear regret - 최적 정책을 찾지 못한다.
- **ϵ-greedy Algorithm**

  거의 가장 좋은 행동을 고르는 알고리즘으로 이는 ϵ 만큼은 계속 다른 행동을 선택한다.

$$\pi(a'|s) = \begin{cases} \epsilon/|A| + 1 - \epsilon & Q^{\pi}(s,a') = \max\limits_{a \in A} Q^{\pi}(s,a) \\ \epsilon/|A| & Q^{\pi}(s,a') \neq \max\limits_{a \in A} Q^{\pi}(s,a) \end{cases}$$

- **Decaying ϵ-greedy Algorithm**

  ϵ-greedy algorithm에서 ϵ 을 Linear하게 감소시키면서 진행, 이 **줄이는 비율** 에 따라 sublinear를 달성하여 최적정책을 찾을 수 도 있다.



Upper Confidence Bounds

: 가장 좋을 **가능성**이 높은 행동을 고르는 알고리즘

선택의 기준 - $Q^{\pi}(s,a) + U_t(a)$, 행동가치함수에 불확실성(Uncertainty)을 추가해 고려한다.

- 불확실성
  - 가치가 불확실한 행동일 수록 좋은 행동일 가능성이 높다.
    - 적게 해본 행동일 수록 불확실성이 높음 ... **UCB**
    - 가치의 표준편차가 높은 행동일수록 불확실성이 높음 ... **Bayesian UCB**
  - Hoeffiding's Inequality를 이용한 불확실성 계산
    - Hoeffiding's Inequality

      $x_1, x_2, \cdots x_t$ 가 0 ~ 1 사이 값을 가진 확률변수이고 이들의 표본평균을 $\bar{X}_t$라고 할 때, 표본평균이 모평균보다 u 만큼 작을 확률은 e^{-2tu^2} 보다 작거나 같다. 즉 **모평균과 표본의 평균의 차이가 클 확률이 적다**는 의미이다.

      $P[\mathbb{E}[X] > \bar{X}_t + u \leq e^{-2tu^2}$

    이를 이용해,

    $e^{-2N_t(a)U_t(a)^2} = p$ ... p = Regret이 클 확률(시간이 지나면서 작아져야 한다.)

    $\rightarrow U_t(a) = \sqrt{\dfrac{-\log p}{2N_t(a)}}$

$p = t^{-4}$ 일 때,

$$\rightarrow U_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}}$$

UCB에서 선택은 $Q^{\pi}(s,a) + \sqrt{\frac{2 \log t}{N_t(a)}}$

이렇게하면, 후회의 합의 증가속도가 log growth이기 때문에 Sublinear Algorithm이다.

Thompson Sampling

가장 좋을 확률이 높은 행동을 고르는 Probability Matching 스타일 알고리즘으로 $Q^{\pi}(s,a)$ 의 분포를 학습 하고 각 행동에 대해 샘플링하고 그 때 가장 높은 행동가치함수를 가지는 행동을 선택하는 것이다. 이 것 또한 Sublinear Algorithm이라고 한다. 이를 구현하는 것은 Dropout을 사용하고 테스트 할 때에도 Dropout rate를 적용하여 inference를 하면된다. (행동가치함수를 Bayesian style로 추정)

$\rightarrow$ 실제로는 Dropout rate 에 따라 성능 차이가 크다.

Count-Based Exploration

이전에 언급했듯이 Optimism in the uncertainty에서는 많이 경험하지 못한 것에 대해서는 불확실성이 크고 이 불확실성을 클수록 좋은 행동일 가능성이 크다고 했다. 이 부분에서는 이 불확실성에 인센티브를 주면서 학습하는 것에 대해 다룬다.

- UCB : 행동을 선택한 횟를 기준으로 인센티브를 준다.
- Bayesian UCB : 행동가치함수 값의 표준편차가 큰 것을 기준으로 인센티브를 준다.
- Thompson sampling : 행동 가치가 높을 확률을 기준으로 인센티브를 준다.

방금 나열한 것은 행동에 인센티브를 주는 것, 행동 대신에 상태, visitation을 기준으로 인센티브를 줄 수 있다.

Curiosity-Driven Learning

다음 처럼 적은 확률로 선택하는 행동에 incentive를 준다.

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

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

Extrinsic reward : MDP에서 정의된 보상 함수로 대부분의 상태, 행동 쌍에 대해서 0이기 때문에 sparse reward라고도 한다. 이는 학습하는데 충분한 정보를 가지고 있지 않다.

Intrinsic reward : 에이전트가 스스로 주는 보상으로 외적 보상이 가지고 있지 않은 상태, 행동쌍에 대한 정보를 가지고 있을 수 있다. 이것은 sparse 하면 안된다.

Count-Based Exploration

grid world처럼 상태,행동의 갯수가 제한적이라면, 방문 분포(visitation distribution)을 직접 모델링하고 이후에 UCB를 적용해 탐험을 할 수 있다. **그럴 수 없다면,**

Stationary distribution $\rho(s)$ ($d_\pi(s)$)를 가정 하고

Recording distribution을 (정의할 수 있다고 가정하면) $\rho(s')\mathbb{P}[S_{t+2} = s | S_{1:t}, S_{t+1} = s]$라고 정의하고, Pseudo count $N_n(x)$ 를 다음을 만족하는 함수로 정의한다.

$\rho_n(x) = \frac{N_n(x)}{n}$ n 번째 까지 Pseudo Count를 따르면,

$\rho'(x) = \frac{N_n(x)+1}{n+1}$ n+1번째에서 또 일어날 확률은 Pseudo count로 결정한다.

그리고, 새롭다는 개념을 수식화한 개념을 Prediction gain은 Stationary distribution과 Recording distribution 간의 차이로 정의되고 다음 처럼 Information Gain 과 $N_n(x)$ 간의 관계를 형성한다.

$PG(s) = (-\log\rho(s)) - (-\log\rho'(s)) = \log(p'(s)) - \log\rho(s)$

$IG_n(x) \leq PG_n(x) \leq \hat{N}_n(x)^{-1}$

$PG_n(x) \leq \hat{N}_n(x)^{-1/2}$

Pseudo count는 Count와 유사한 역할을 하고, Prediction gain과 관련이 있는 intrinsic motivation 방법론들은 이론적으로 그것과 관계가 있다.

Intrinsic Motivation Algorithm

에이전트가 새로움을 표현하는 것은 에이전트가 환경의 dynamics를 얼마나 알고있는가 이다.

행동 상태 값에 따라 환경이 어떻게 바뀔 것인지를 아는 것은, $y = Env(s,a)$, 가장 직관적이고 환경에 대한 정보를 손실하지 하지 않지만, 고차원데이터 이기 때문에 학습하기 까다롭고 노이즈에 취약하다.

Autoencoder를 이용하여 $y = AE(s,a)$ 인코더로 코드된 것을 예측하면 방금과 장점을 공유하면서 저차원데이터이라는 장점이 있다. 여전히 노이즈에 취약하다고 한다.

학습되지 않은 네트워크의 output을 코드(타겟)으로 사용하는 Random Feature $y = RF(s,a)$ 를 사용하면 Autoencoder에서 학습이 필요없게 된다. 다만 노이즈에는 여전히 취약하고 정보를 충붖ㄴ히 가지고 있지 않을 가능성이 있다.

Intrinsic reward의 특성

- 내적보상은 에이전트가 상태에 대해 알게 될 수록 작아진다. 다시말해 non-stationary하다는 의미이다.
- Off-policy 알고리즘은 바뀌는 reward를 반영하기 어려워 **On-policy Learning**을 사용한다.
- Reward scaling : reward/return의 표준편차로 나눠준다.
- Advantage normalization : 작은 intrinsic reward를 기반으로 학습하기 위해 advatage를 화이트닝 한다.
- Distributed RL : 128개 이상의 환경을 동시에 학습한다.
- Feature normalization : 상태간의 차이가 큰것을 방지하기 위해 입력데이터를 화이트닝 한다.

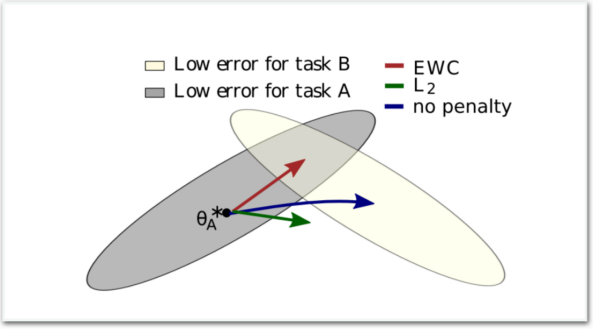
Random Network Distillation(RND)

상태를 입력으로하는 초기화가 되어 있는 네트워크(Target), 동일한 상태를 입력으로 하고 Target을 예측하는 prediction 네트워크

Prediciton네트워크가 점점 학습해 가는 것을 **Distillation**이라고 표현

Prediction 네트워크와 Target 간의 오차의 원인

- 학습의 부족 : 상태가 얼마나 새로운 지를 반영
- 환경의 무작위성(Stochasticity) : 상태의 정보만으로 예측이 어려움
- Prediction 네트워크가 target 네트워크를 표현 할 수 없는 경우가 있는데, 이를 위해 Prediction 네트워크를 더 복잡하게 구성하여 이를 방지 한다.
- Catastrophic forgetting : 이전에 학습했는데 시간이 지나서 잊어버리는 경우 거기에 다시 흥미를 느껴서 성능이 나빠지는 경우. 정의하기로는 A,B 문제를 순서대로 학습하는 경우 A를 학습하고 B를 학습하면 B의 학습이 종료되는 시점에서 A에 대한 성능이 심하게 저하되는 경우이다.



<결과 비교 차트>

	Gravitar	Montezuma's Revenge	Pitfall!	PrivateEye	Solaris	Venture
RND	**3,906**	**8,152**	-3	8,666	3,282	**1,859**
PPO	3,426	2,497	0	105	3,387	0
Dynamics	3,371	400	0	33	3,246	1,712
SOTA	2,209[1]	3,700[2]	**0**	**15,806**[2]	**12,380**[1]	**1,813**[3]
Avg. Human	3,351	4,753	6,464	69,571	12,327	1,188

Table 1: Comparison to baselines results. Final mean performance for various methods. State of the art results taken from: [1] (Fortunato et al., 2017) [2] (Bellemare et al., 2016) [3] (Horgan et al., 2018)

질문

t는 에피소드 하나? timestep ?

- timestep 은 없다. MAP에서 t는 에피소드를 의미한다.

regret을 Loss로 직접 사용 가능한가?

- 미분 불가능 고로 불가능, exploration 을 할 때 regret 을 달성하는지를 검사하면서 탐험한다.

reward는 Bound되어 있는 값만 쓸 수 있는가?

- 그렇다.

epsiolon greedy 보다 Q 네트워크를 만들때 dropout 을 쓰면 탐험이 적절하게 일어난다.

- 그렇다.

Thompson sampling을 사용하면 epsilon greedy를 사용하지 않는가?

- 현실적으로는 epsilon greedy 를 같이 쓰는게 더 좋지만, 논문에서 주장하는 바는 epsilon greedy를 쓰지 말라는 이야기 이긴하다. 추가적으로 초반에는 탐험을 많이 하는게 좋다.

Dropout rate 가 어느정도 여야 하는가?

- Dropout rate는 hyper parameter, empriclly 0.3 보다 크면 안된다.

내적 보상을 추가하는게 흔하지 않게 도달하는 상태에 대해 행동하는게 더 안좋을 수도 있는데 그런 위험을 감수하고도 사용하는 것인가?

- 그렇다.

왜 leakyrelu 를 썼는가 ?

- 명확히 밝힌 바가 없다.