2주차.PyTorch로 시작하는 강화학습 입문 Camp

강의 내용

복습

- MDP
  - 상태 집합 : 환경이 제공할 수 있는 모든 상태의 집합
  - 행동 집합 : 에이전트가 고를 수 이쓴 모든 행동의 집합
  - 전이 확률 : 상태와 행동이 정해졌을 때, **환경이 다음 상태를 고르는 규칙**
  - 보상 - 두가지 표현법을 혼용하여 사용하는데, 행동과 상태가 정해졌을 때 환경이 Agent에게 주는 효용은 r_s^a 라고 표현하고 시점에 초점을 맞추어 특정시점에 환경으로부터 에이전트가 얻는 효용은 r_t라고 표현한다.
  - 할인률 : 시점간의 보상의 차이를 할인하는 비율
- Policy distribution : $\pi(a|s) 특정한 상태 조건에서 에이전트의 행동 집합의 확률분포
- 수확(Return) = $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$
- 목적함수

  : $\max_\theta J(\pi_\theta) \doteq \max_\theta \mathbb{E}_{\pi_\theta}[R_0] = \mathbb{E}_{\pi_\theta *}[R_0]$
  - State Value Function : t시점에서 상태가 정해지고 행동은 기존의 정책을 따를 때 기대되는 수확으로 이 상태가 얼마나 좋은지를 나타내는 함수 - $V^\pi(s) = \mathbb{E}_\pi[R_t|s_t = s]$
  - Action Value Function : t시점에서 상태와 행동이 정해지고 나머지 행동은 기존의 정책을 따를 때 기대되는 수확 - $Q^\pi(s,a) = \mathbb{E}_\pi[R_t|s_t = s, a_t = a]$
- 일관성

  행동가치함수에서 a_t 또한 기존의 정책을 따르면 $\mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)] = V^\pi(s)$

  t시점에서의 행동을 특정한 행동으로 고정하면 $Q^\pi(s,a) = r_s^a + \mathbb{E}_{\pi, T_{s,s'}^a}[V^\pi(s')]$

  그래서, 최적의 $Q^{*\pi}, V^{*\pi}$ 는 각각 다음과 같은 수학적 일관성을 유지 해야한다.

  $Q^\pi(s,a) = r_s^a + \gamma\mathbb{E}_{T_{s,s'}^a}[\mathbb{E}_{a'\sim\pi(\cdot|s')}[Q^\pi(s',a')]]$

  $V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[r_s^a + \gamma\mathbb{E}_{\pi, T_{s,s'}^a}[V^\pi(s')]]$

  $\rightarrow$ 고정된 정책에 대해서 일관성을 유지하는 함수를 찾는다.
- 최적의 정책

  최적의 원리(principle of Optimality) : 도달 할 수 있는 상태들에 대해서 $V^{\pi^*}(s) = \max_{a\in A} Q^{\pi^*}(s,a)$

  현재의 가치함수를 사용해서 현재의 최적 행동들로 정책을 갱신한다.
  - $Q^\pi(s,a) = r_s^a + \gamma\mathbb{E}_{T_{s,s'}^a}[\max[Q^\pi(s',a')]]$
  - $V^\pi(s) = \max_{a\in A}[r_s^a + \gamma\mathbb{E}_{\pi, T_{s,s'}^a}[V^\pi(s')]]$

이를 이용하여 *그리드월드 문제에 대해 가치반복 알고리즘을 사용할 수 있다.

    - 그리드월드 : 상태집합과 행동집합이 유한하고, 가치 함수들을 행렬로 표현 할 수 있는 MDP; Frozen Lake

- Policy iteration

  : 정책과 행동가치함수를 행렬로 저장하고, Policy evaluation 단계에서 가치함수를 찾고, Policy improvement 단계에서는 현재 가치 함수에서 최적의 정책으로 개선하는 것을 반복하는 알고리즘

- Value iteration
    - 행동가치 함수 만 행렬로 저장하고, 최적의 정책에 대한 일관성을 이용해 최적 정책의 가치 함수를 찾는 것. 최적 정책에서는 가치함수값이 높은 행동만을 선택하기 때문에
    - 정책을 행렬로 저장 할 필요가 없고, Policy evaluation 과 Policy Improvement 단계를 나눌 필요가 없으며, 이렇게 해도 최적정책을 찾을 수 있다고 *Richard Bellman*에 의해 증명되어 있다.

동적계획법의 한계

: 전이 확률을 모른다면 가치함수의 계산이 불가능

: 정책과 가치함수를 표현하는 데 있어, 그리드 월드가 아니거나 경우의 수가 많이 크다면 물리적으로 불가능

- 용어 정리 - 예측 / 제어 문제
    - Prediction 문제 : 전이 확률을 모르는 MDP에서 주어진 정책의 가치함수를 찾는 것
    - Control 문제 : 전이확률을 모르는 MDP에서 최적정책/최적정책의 가치함수를 찾는 것

Prediction 문제

Monte-Carlo Learning

: 수많은 샘플링을 한 뒤 얻어지는 표본의 평균은 대수의 법칙에 의해 실제 모평균에 수렴한다. 우리가 알고 싶어하는 가치함수는 각 시점의 수확이라는 확률변수(R_t)의 기댓값(평균)이다. 이를 Monte-Carlo 방식으로 여러 번 샘플링 하여 실제 가치 함수와 가까운 가치함수를 찾는 것이다.

이 때 일반적으로 평균을 계산할 수 있지만, 샘플링과 동시에 가치함수를 업데이트하는 Moving Average를 사용한다.

$$
\begin{aligned}
\mu_t^{(n)} &= \frac{1}{n}\sum_{j=1}^{n} R_t^{(j)} \\
&= \frac{1}{n}(R_t^{(n)} + \sum_{j=1}^{n-1} R_t^{(j)}) \\
&= \frac{1}{n}(R_t^{(n)} + (n-1)\mu_t^{(n-1)}) \\
&= \mu_t^{(n-1)} + \frac{1}{n}(R_t^{(n)} - \mu_t^{(n-1)})
\end{aligned}
$$

이 때, n이 커지는 경우 업데이터가 거의 되지 않기 때문에 α 를 설정하고 동일한 비율로 업데이트 한다.

$$\mu_t^{(n)} = \mu_t^{(n-1)} + \alpha(R_t^{(n)} - \mu_t^{(n-1)})$$

순서대로, n 번째 에피소드를 주어진 정책에 따라 실행 $\rightarrow$ 에피소드에서의 각시점의 수확을 계산한다.

$R_t^{(n)} = \sum_{k=0}^{\infty} \gamma^k r_{t+k}^{(n)} \rightarrow \mu_t^{(n)} = Q_{(n)}^{\pi}(s_t, a_t)$ 를 갱신한다.

Temporal Difference Learning

Temporal Difference Learning은 에피소드의 중간부터 계산해놓은 가치 함수를 사용하는 방식을 사용하여, 에피소드가 끝나지 않은 경우, 불완전한 에피소드가 데이터인 경우에 사용할 수 있는 학습 방법이다.

방법은 Monte-Carlo와 비슷하다. 먼저 n 번째 에피소드를 주어진 정책에 따라 실행한다. 그리고 에피소드에서 각 상태 행동 쌍에 대해 평균을 계산한다. 이 때 MA방식을 사용하여 평균을 계산 한다.

다만 $\mu_t^{(n)} = \mu_t^{(n-1)} + \alpha(R_t^{(n)} - \mu_t^{(n-1)})$ 이 수식에서 $R_t^{(n)}$ 을 에피소드가 끝난 후 모든 r_s^a 를 평균 하는 것이 아니라 이전의 가치함수로 만든 수확($R_t^{(n)}$)으로 갱신한다.
$\mu_t^{(n)} = \mu_t^{(n-1)} + \alpha(r_t^{(n)} + \gamma Q^{\pi}_{(n-1)}(s_{t+1}, a_{t+1}) - \mu_t^{(n-1)})$

TDL 방식은 하나의 샘플로 여러개의 샘플을 추출할 수 있어 데이터를 더 효율적으로 사용할 수 있다. 계산하는 과정이 재귀함수에서 메모마이제이션을 한 것을 떠올리게 한다.

<갱신의 깊이와 넓이에 따른 RL 방법론 선택>

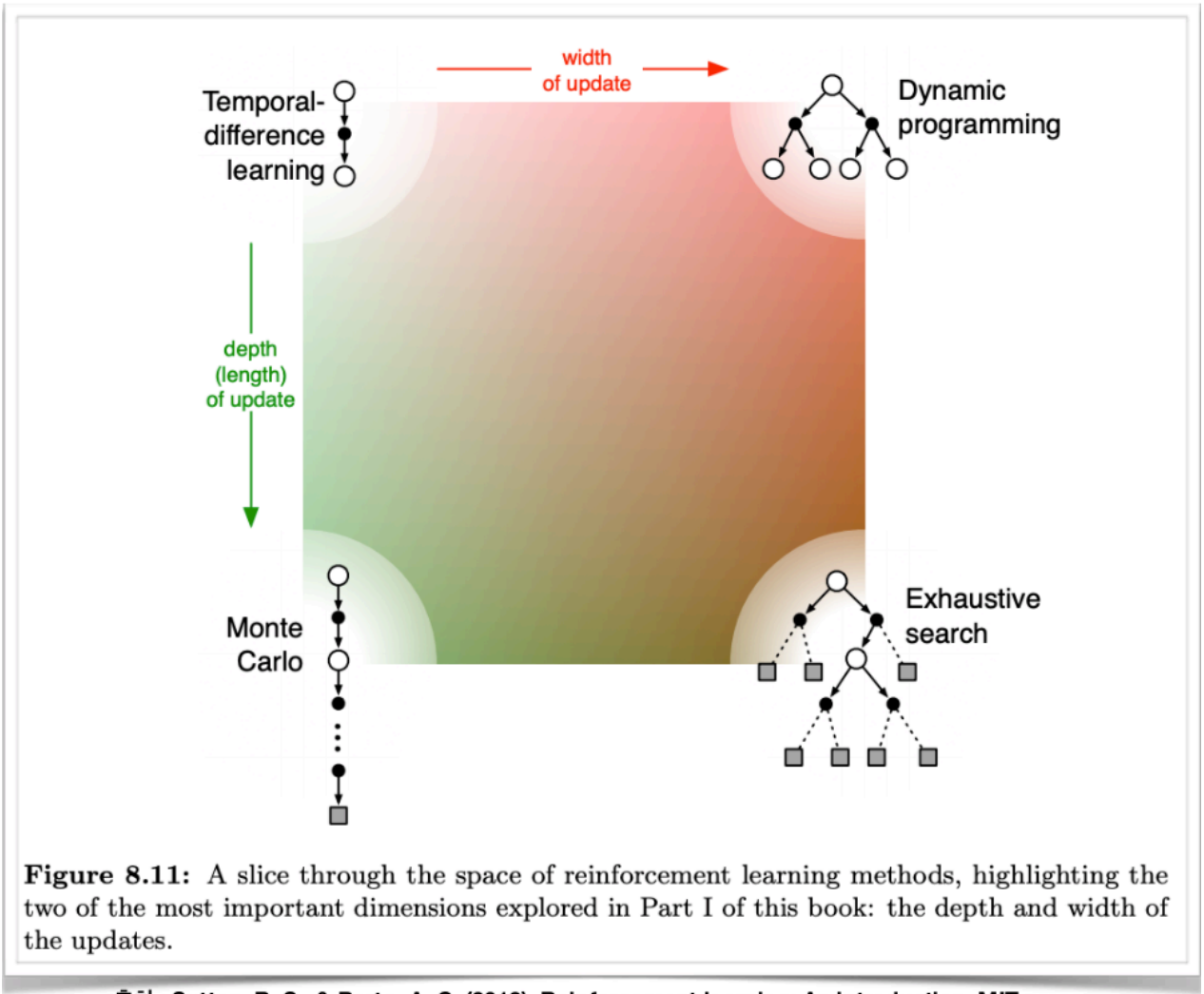


Figure 8.11: A slice through the space of reinforcement learning methods, highlighting the two of the most important dimensions explored in Part I of this book: the depth and width of the updates.

출처 : Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Control 문제

기존에 전이확률을 모두 알고 있을 때는 가치함수가 모든 경우의 수와 확률을 전부 고려한 것이지만, MC,TD 방법으로 근사한 가치함수 값은 시뮬레이션에서 한번도 가지 않았던 행동 상태 쌍에 대해서는 0이기 때문에 Greedy Update 방식으로 정책을 갱신하면 Local Minima에 빠질 우려가 있다. 따라서 현재 정책이 선택 하지 않은 행동들에 대한 가치 또한 측정이 필요한데 이를 좀 유식한 말로 **탐험(Exploration)** 이라고 한다.

- ϵ - Greedy Exploration
  - 가장 간단한 탐험 방법으로 ϵ 를 0.01 ~ 0.05(이 비율을 고정해두는 것은 추천하지 않는다.) 사이의 값으로 그 확률로는 균등 분포로 행동을 선택하게끔하는 것이다. $\epsilon - 1$ 의 확률로는 현재 가치 함수가 알려주는 가장 좋은 행동을 선택한다.

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

  - 다만 행동집합이 유한한 경우에 가능한 방법이다.
  - 이 방법을 통해서 최적 정책을 찾을 수 있으려면 $\epsilon^{(n)} = \frac{1}{n}$ 또는 $\frac{1}{n \times k}$ 를 만족해야 한다.(중요한 것은 ϵ이 선형적으로 변화한다는 점이다.)

그래서 MC, TD 와 Greedy Exploration의 조합에 따라서 Control problem을 푸는 방법을 정리 할 수 있다.

- GLIE MC Control

  : 고정된 정책에 대해 Monte-Carlo learning 으로 가치함수를 학습 $\rightarrow$ 학습된 가치함수를 기반으로 정책을 업데이트(GLIE improvement)

- GLLE TD Control (**Sarsa 알고리즘**)

  : 고정된 정책에 대해 Temporal Difference learning 으로 가치함수를 학습 $\rightarrow$ 학습된 가치함수를 기반으로 정책을 업데이트(GLIE improvement)

$\rightarrow$ 이 두 가지 방법은 정책과 가치함수를 모두 저장 해야한다. 따라서 저장공간에 대해 제약이 크다. 그리하여 Policy Iteration에서 Value Iteration으로 갈 때 처럼 **가치함수만을 저장**하고 학습하는 알고리즘이 필요했는데 그것이 바로 **Q-Learning**이다.

Q-Learning

1. n 번째 에피소드를 현재 가치함수의 ϵ-greedy 정책에 따라 실행한다.
   $\tau^{(n)} = (s_1^{(n)}, a_1^{(n)}, r_1^{(n)}, s_2^{(n)}, a_2^{(n)}, r_2^{(n)}, \cdots)$
2. 에피소드의 각 상태, 행동 쌍에 대해서 지금까지 계산된 MA와 **이전 가치함수로 만든 수확** 을 기반으로 MA을 갱신한다.

$$\mu_t^{(n)} = \mu_t^{(n-1)} + \alpha(r_t^{(n)} + \gamma \max_{a \in A} Q_{(n-1)}^{\pi^*}(s_{t+1}, a) - \mu_t^{(n-1)})$$

   여기서 Sarsa 와 다른 점은 Sarsa에서는 $\mu_t^{(n)}$ 을 갱신하기 위해서 n번째 에피소드에서 t+1시점의 행동과 상태에 대한 가치함수 값을 사용하는데, Q-Learning에서는 현재 상태와 행동과 그다음 상태까지만 주어지면 그 상태에 따른 행동은 기존의 가치함수를 보아 최댓값인 행동을 선택하고 그 에 따른 보상을 사용하여 이동 평균(가치 함수)값을 갱신한다.

그리고 중요한 점은, Q-Learning 에서 에피소드를 실행할 때 사용하는 정책과 가치함수를 만들 때 사용하는 정책이 서로 다르다는 것이다. 이를 두고 Off-Policy Learning이라고 분류한다. MC/TD Learning의 경우는 On-Policy Learning으로 데이터를 만드는(에피소드를 진행하는) 정책과 가치함수를 찾을 때 사용하는 정책이 같다. 이 때 정책을 갱신(Policy Improvement)할 때 epsilon greedy 하게 갱신 하게되는데 이론상 이 방법으로 정책을 갱신하여 최적의 정책을 찾기 위해서는 ϵ이 0에 수렴해야 하기 때문에(위 조건에 나와있음), exploration이 보장되지 않는다. 그리고 On-Policy Learning에서는 현재 Policy 로 만드는 에피소드만을 데이터로 사용할 수 있다. 이에 Off-Policy Learning이 생겨났다. 방금 말한 두가지 한계를 극복하기 위함이다. Q-Learning에서는 에피소드를 진행할 때는 ϵ - greedy 하게 진행하고 가치함수를 갱신할 때는 다음 상태에서 이전 가치함수 기준 가장 높은 행동을 선택하고 그 가치함수 값을 이용한다.

DQN

Q-Learning의 가치함수 근사에 뉴럴넷 구조를 사용한 것이다.

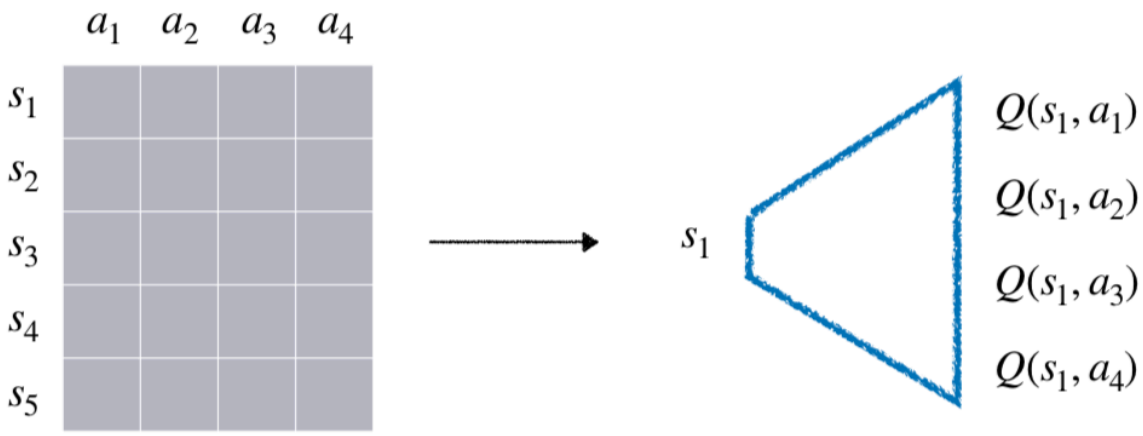
MA방식으로 평균값을 갱신하는 수식 $\mu_t^{(n)} = \mu^{(n-1)} + \alpha(R_s^a - \mu_t^{(n-1)})$ 에서 $\left(R_s^a - \mu_t^{(n-1)}\right)$ 부분을 **TD error**라고 한다. 이를 Q-learning 방식으로 풀어 쓰면 다음과 같다.

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

DQN은 이를 최소화하는 Q를 뉴럴넷으로 근사하는 것이다. (여기서 $Q_\theta(s_t, a_t)$ 와 $r_t^{(n)}$ 의 부호가 이해가 잘 가지 않는다.)

$$\min_\theta (Q_\theta(s_t, a_t) - r_t^{(n)} + \gamma \max_{a \in A} Q_\theta(s_{t+1}, a))^2$$

이 때 다음 그림처럼 모든 상태 행동 쌍에 대한 가치함수값을 출력하도록 출력단을 구성한다.



$$\min_\theta \left(Q_\theta(s_t, a_t) - r_t^{(n)} + \gamma \max_{a \in A} Q_\theta(s_{t+1}, a)\right)^2$$

- 실제 네트워크 학습에 관해서

  $$\min_\theta (Q_\theta(s_t, a_t) - r_t^{(n)} + \gamma \max_{a \in A} Q_\theta(s_{t+1}, a))^2$$

  푸른 부분은 정답이고, 붉은 부분은 예측하는 값이다 매 스텝에서 이를 같이 학습 하면 학습이 불안정하기 때문에 정답과 예측을 분리 한다.

  1. 1000 ~ 10000 마다 가치함수를 서로 동기화 한다.
  2. 파라미터를 Moving Average

실습

- <전처리>
  - 16장의 이미지 중 4개의 step으로 4개중 하나를 state로 정의 (16장에서 4장 추출)
  - 모든 양의 보상을 1, 음의 보상을 -1로 제한(Gradient가 폭발하지 않도록)
  - Image size 보정, 84x84의 그레이스케일 이미지로 변환

    - 몫이 여러 개라면, 하나의 몫만을 기준으로 가치를 측정 (여러개의 몫을 모두 고려하면 episode) 가 매우 길어짐

- Double Deep Q Network

  Q-Learning의 Target에서 가치함수의 최댓값을 이용해 가중치를 갱신하는 것은 Over-estimation을 발생시킨다. 그래서 최댓값을 찾는 함수또한 Prediction Network를 사용한다.

- **Prioritized experience Replay**

  모든 state의 transition이 똑같이 중요하지 않다는 문제제기를 한다. 실제로 보상이 발생하는 시점과 가장 가까운 데이터에서 부터 학습이 시작되기 때문에 그렇다. 학습을 하면서 TD-error가 큰 데이터일 수록 네트워크가 알지못하는 정보를 가지고 있으므로 이를 기준으로 중요도를 반영하여 IW(importance weight)만들어 학습에 이용한다는 것이다.

    - Importance Sampling
        - 원하는 분포와 뽑고자하는 분포가 서로 다를 때 사용하는 것으로 우리는 균등분포를 원하지만 뽑고자하는 분포는 데이터의 우선순위를 고려한 분포이다. 이 때 기댓값을 계산하는

- Dueling Architecture

  행동가치함수는 어떤 상태와 행동이 주어졌을 때 나머지는 기존의 정책을 따른다면 기대되는 보상을 의미하는데 이를 특정상태에 대한 값과 특정 행동에 대한 값으로 분리하겠다는 것이다.

    - Advantage function
        - 어떤 상태에서 특정 행동이 기존의 정책을 따르는 것보다 얼마나 좋은지

          $A^{\pi}(s,a) \doteq Q^{\pi}(s,a) - V^{\pi}(s)$

            - 기댓값은 0 $\mathbb{E}[A^{\pi}(s,a)]$
            - $\arg\max_{a \in A} A^{\pi}(s,a) = \arg\max_{a \in A} Q^{\pi}(s,a)$
            - $\max_{a \in A} A^{\pi^*}(s,a) = 0$

  행동가치함수 대신에 상태 가치함수와 이익 함수를 학습한다.

질문

- TD Learning에서 t 시점 그리고 시점에 대한 표현이 헷갈린다. 다시 설명해주길 바란다.
- Behavior 정책 과 target policy 를 나눈 이유 ?

  on policy learning- 메모리가 커도 얻는 이점이 없고 지금 당장의 Policy로 만드는 데이터만 쓸 수 있으니까. 학습이 안정적일 수 있다. 다만 Off Policy 를 사용하면 남의 데이터, 옛날 데이터를 사용할 수 있다는 이점이 있다.