7. 폴리시 그레디언트(Policy Gradient)

김호철

Contents

1 소개

- **정책(Policy) 기반 강화학습**
  - 지금까지는 파라메터 θ(가중치)를 사용하여 상태가치함수(State-Value Function) 또는 액션가치함수(Action-Value Function)를 근사(approximate)하였다.

$$\begin{aligned} V_\theta(s) &\approx V_\pi(s) \\ Q_\theta(s,a) &\approx Q_\pi(s,a) \end{aligned} \tag{1}$$

  - 정책은 가치함수로부터 직접 생성되었다.(예를 들어 ϵ-탐욕을 사용하여)
  - 이제 부터는 정책(Policy)에 직접 파라메터를 적용한다.

$$\pi_\theta(s,a) = P\left[a|s,\theta\right] \tag{2}$$

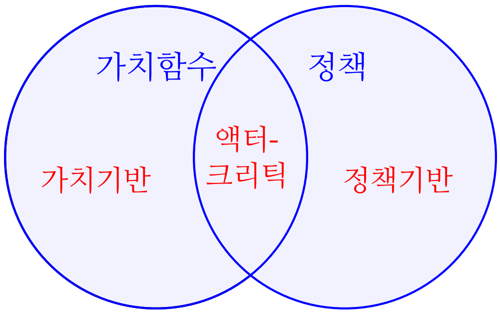
  - VFA에서는 입력이 상태이고, 출력은 가치함수였는데,
  - 폴리시 그레디언트에서는 입력은 상태이고, 출력은 각 액션을 할 확률이 된다.
  - 강화학습에서 가장 Hot한 분야

- **가치(Value)기반과 정책(Policy)기반 강화학습**

  가치기반: 가치함수를 학습하고, 암묵적으로 ϵ-탐욕 정책 사용

  정책기반: 가치함수 없이 바로 정책을 학습

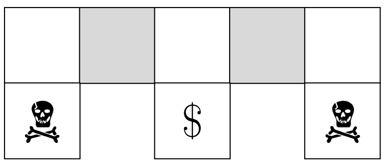
  액터-크리틱: 가치함수, 정책 모두 학습



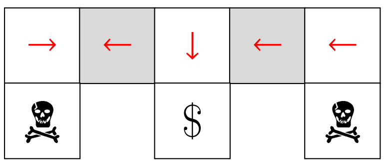
- **정책 기반 강화학습의 장단점**
  - **장점** :
    * 속성들의 수렴이 더 좋다.
    * 고차원이나 연속적인 액션 공간(예:0~1사이의 실수)에 효율적
    * 가치 기반은 결정적(deterministic) 정책들만 학습했으나, 정책 기반은 확률적(stochastic)정책의 학습이 가능
  - **단점** :
    * 일반적으로 글로벌 최적화보다 로컬 최적화에 수렴하기 쉽다.
    * 정책을 평가하는 것은 비효율적이거나 분산이 클 수 있다.

1.1 예제

- **예제 1 : 가위바위보**
  - 한 가지만 내는 결정적(deterministic) 정책은 곧 상대에게 수가 읽혀 패하게 된다.
  - $\frac{1}{3}$의 확률(stochastic)로 랜덤의 정책이 최적의 정책이다.
  - 내시 균형(Nash equilibrium)
- **예제 2 : 그리드월드**



  - 에이전트는 위쪽 5가지 상태 어디에서나 시작할 수 있으며, 해골에 가면 죽고 달러로 가는 것이 목표다.
  - 회색 상태는 구분할 수 없어서 하나의 정책으로 학습된다.



– **결정적 정책**으로 학습한 경우에 흰색 상태는 각자 최적의 정책으로 학습되지만, 회색 상태들의 경우 한쪽으로만 학습되어 갇힐 수 있다.

→	↔	↓	↔	←
☠		$		☠

– **확률적 정책**의 최적 학습은 회색 상태들은 50%확률로 양방향으로 가는 것이다.
– 이 정책은 갇히지 않고 목표점에 도달할 수 있게 학습된다.

1.2 정책(Policy)에 대한 고찰

- **정책 목적 함수들**
  – 정책 $\pi_\theta(s,a)$는 액션이 취해질 확률을 출력하는 함수이다.
  – 목표는 주어진 파라메터 θ들이 있는 정책 $\pi_\theta(s,a)$에서 최고의 θ를 찾는 것이다.
  – 이 파라메터 θ들을 업데이트하려면 기준이 필요한데, TD(0)에서 TD 에러를 사용한 것처럼, 폴리시 그레디언트에서는 **목적 함수(Objective Function)**라는 것을 정의
  – 어떤 정책을 따랐을 때 보상(Reward)의 합이 가장 많은 것이 좋은 정책이다.

  – **에피소딕 환경**(시작 상태가 같고 종료가 있는)에서는 **시작 상태의 가치함수(start value)**를 최대로 하고자 하는 것이 목표가 된다. 에피소딕에서 시작 상태는, 시작부터 종료될 때까지의 모든 상태의 보상을 가지고 있다.

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1] \tag{3}$$

  – **연속적 환경**에서는 **평균 가치(average value)**를 사용하기도 하고, (평균 가치 = 어떤 상태가 발생한 확률 × 그 상태의 가치함수)

$$J_{av}v(\theta) = \sum_s d^{\pi_\theta}(s)V^{\pi_\theta}(s) \tag{4}$$

  – **연속적 환경**에서는 **시간 단계별 평균 보상(average reward per time-step)**을 사용할 수도 있다.(시간 단계별 평균 보상=각 타임 스텝마다 받는 보상의 기대값)

$$J_{av}R(\theta) = \sum_s d^{\pi_\theta}(s)\sum_a \pi_\theta(s,a)R_s^a \tag{5}$$

  – $d^{\pi_\theta}(s)$ 고정 분포(stationary distribution) : 마르코프 체인에서 정책 π_θ를 따랐을 때 에이전트가 그 상태에 머무를 확률

- **정책 최적화**
  – 정책 기반 강화학습은 최적화(Optimisation) 문제이다.
  – 목적함수 $J(\theta)$를 최대화 하는 θ를 찾는 문제 (θ가 정책을 결정)
  – 많은 최적화 방법이 있지만, 기울기 하강법(gradient descent)에 기반을 두고, 다양한 확장이 가능하다.

2 유한 차분 폴리시 그레디언트(Finite Difference Policy Gradient)

- **폴리시 그레디언트(Policy Gradient)**
  - $J(\theta)$를 폴리시 목적 함수(policy objective function)로 정의
  - 폴리시 그레디언트 알고리즘은, 파라메터 θ에 대하여 $J(\theta)$의 값을 가장 급격하게 변하는 방향(오름차순)으로 α (학습률)만큼 업데이트 해주는 방법

$$\Delta\theta = \alpha \nabla_\theta J(\theta) \tag{6}$$

  - 여기서 $\nabla_\theta J(\theta)$는 **폴리시 그레디언트**이다. 즉, 각 θ로 편미분 한 기울기 벡터이다.

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix} \tag{7}$$

  - 그리고 α는 스텝-크기(학습률)이다.

- **유한 차분으로 기울기 계산하기**
  - $\pi_\theta(s, a)$의 정책 기울기를 평가하기 위해
  - 각 차원(dimension) $k \in [1, n]$ 마다,
    * k번째 차원에서 작은 ϵ만큼 θ를 조정하여, θ에 대한 목적함수의 k번째 편도함수를 추정

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon} \tag{8}$$

    * 여기서 u_k는 k번째는 1이고, 다른 곳에는 0이 되는 단위 벡터
  - n차원의 정책 기울기를 계산하기 위해 n번의 평가가 필요
  - 간단하고, 잡음이 있고, 비효율적이지만 가끔 효과적이다.
  - 정책이 미분 가능하지 않아도 가능하고, 임의(arbitrary)의 정책에도 작동한다.
  - 최근에는 잘 사용되지 않는다.

3 몬테카를로 폴리시 그레디언트

3.1 Likelihood ratios(우도비)

- **스코어 함수(Score Function)**
  - 해석적(analytically)으로 정책 기울기를 계산
  - 정책 π_θ는, 0이 아니고 미분 가능(differentiable)하다면, 기울기는 $\nabla_\theta \pi_\theta(s, a)$ 이다.

– 미분 정리 리마인드

$$\begin{aligned}\frac{d\log x}{dx} &= \frac{1}{x} \\ \frac{dx}{x} &= d\log x\end{aligned} \tag{9}$$

– Likelihood ratios(우도비)는 일종의 트릭으로 다음을 따른다.

$$\begin{aligned}\nabla_\theta \pi_\theta(s,a) &= \pi_\theta(s,a)\frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} \\ &= \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)\end{aligned} \tag{10}$$

– 스코어 함수는 $\nabla_\theta \log \pi_\theta(s,a)$이다.

3.2 폴리시 그레디언트 정리

- **원-스텝 MDP들**
  - 원-스텝 MDP : 상태($s \sim d(s)$)에서 시작해서, 한 스텝 후에 보상($r = \mathcal{R}_{s,a}$)을 받고 종료하는 MDP
  - 폴리시 그레디언트를 구하기 위해 우도비(Likelihood ratios)를 사용. $J(\theta)$는 폴리시 목적함수

$$\begin{aligned}J(\theta) &= \mathbb{E}_{\pi_\theta}[r] \\ &= \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(s,a)\mathcal{R}_{s,a} \\ \nabla_\theta J(\theta) &= \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)\mathcal{R}_{s,a} \\ &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log\ \pi_\theta(s,a) r]\end{aligned} \tag{11}$$

  - 기대값의 정의는 (확률 × 받을값)
  - $\sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(s,a)$ 는 에이전트가 어떤 상태 s에서 행동 a를 선택할 확률을 의미
  - Likelihood ratios를 한 이유는, 그렇게 하지 않고 미분을 하면 π_θ가 사라져 expectation($\mathbb{E}$)을 취할 수가 없다
  - expectation으로 묶어서 그 안을 샘플링하게 되어야 강화학습이 된다.
  - 따라서 expectiation을 취하기 위해서 Likelihood ratios 트릭을 사용한 것이다.

- **폴리시 그레디언트 정리**
  - 폴리시 그레디언트 정리는 다중 스텝 MDP에 대해서도 likelihood ratio 접근을 일반화 시켜준다.
  - 순간 보상 r을 장기(long-term) value $Q^\pi(s,a)$ (오라클이 준 함수)로 대체 한다.
  - 폴리시 그레디언트 정리는 시작 상태 목표, 평균 보상 및 평균 값(value) 목표에 적용된다.

정리(Theorem)

어떤 미분 가능한 정책이 $\pi_\theta(s,a)$ 이고,

폴리시 목적 함수가 $J = J_1, J_{avR}$, 또는 $\frac{1}{1-\gamma}J_{avV}$ 이면,

폴리시 그레디언트는,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)\right]$$

- **몬테카를로 폴리시 그레디언트(REINFORCE)**
  - 통계적(stochastic) 기울기 상승(ascent)으로 파라메터를 업데이트
  - 폴리시 그레디언트 정리를 이용
  - $Q^{\pi_\theta}(s_t,a_t)$에 편향되지 않게 샘플링된 리턴 v_t를 사용

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(s_t,a_t) v_t \tag{12}$$

```
function REINFORCE
    θ를 임의의 값으로 초기화
    각 에피소드 ({s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T} ~ π_θ)에서 수행
        t가 1에서 T-1까지 수행
            θ ← θ + α∇_θ log π_θ(s_t, a_t) v_t
    return θ
end function
```

4 액터-크리틱 그레디언트

- **크리틱으로 분산 줄이기**
  - 몬테카를로 폴리시 그레디언트는 여전히 분산이 높다.
  - 액션-가치함수를 추정하기 위해 **크리틱**을 사용한다.

$$Q_w(s,a) \approx Q^{\pi_\theta}(s,a) \tag{13}$$

  - 액터-크리틱 알고리즘은 2개의 파라메터를 가진다.
    - **크리틱**: 액션-가치 함수 파라메터 $\mathbf{w}$를 업데이트 한다.
    - **액터**: 크리틱이 제안한 방향으로 정책(policy) 파라메터 θ를 업데이트 한다.
  - 액터-크리틱 알고리즘은 폴리시 그레디언트 근사를 따른다.

$$\begin{aligned} \nabla_\theta J(\theta) &\approx \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)] \\ \Delta\theta &= \alpha \nabla_\theta \log \pi_\theta(s,a) Q_w(s,a) \end{aligned} \tag{14}$$

- 크리틱을 사용하는 **액션-가치함수의 추정**은 MC 정책평가, TD 학습, TD(λ)나 최소 제곱 에러도 사용 가능하다.
- **액션-가치 액터-크리틱(Action-Value Actor-Critic)**
  - 액션-가치를 사용하는 크리틱 기반의 단순한 액터-크리틱 알고리즘
  - 선형(linear) 가치 함수 근사를 사용 : $Q_w(s,a) = \phi(s,a)^\top w$
    - **크리틱**: 선형 TD(0)로 w 업데이트
    - **액터**: 폴리시 그레디언트로 θ 업데이트

```
function QAC
    s, θ 초기화
    정책에서 액션을 샘플링 a ~ π_θ
    다음 스텝 반복 수행
        보상 r (r = R_s^a)과 다음 상태 s'(s' ~ P_s^a)를 샘플링
        정책에서 다음 액션을 샘플링 : a' ~ π_θ(s', a'))
        δ = r + γQ_w(s', a') − Q_w(s, a)
        θ = θ + α∇_θ log π_θ(s, a)Q_w(s, a))
        w ← w + βδφ(s, a)
        a ← a', s ← s'
end function
```

4.1 어드밴티지 펑션 크리틱(Advantage Function Critic)

- **베이스라인(Baseline)을 사용하여 분산 줄이기**
  - 폴리시 그레디언트에서 베이스라인 함수 $B(s)$를 빼 준다.
  - 이는 기대값의 변화없이 분산을 줄일 수 있다.(Likelihood ratios(우도비)의 역이용)

$$\begin{aligned}\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)B(s)] &= \sum_{s\in S} d^{\pi_\theta}(s) \sum_{a\in A} \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)B(s) \\ &= \sum_{s\in S} d^{\pi_\theta}(s) \sum_{a\in A} \nabla_\theta \pi_\theta(s,a)B(s) \\ &= \sum_{s\in S} d^{\pi_\theta} B(s)\nabla_\theta \sum_{a\in A} \pi_\theta(s,a) \\ &= 0\end{aligned} \tag{15}$$

  - 상태 가치 함수는 좋은 베이스라인이 된다. : $B(s) = V^{\pi_\theta}(s)$
  - 그래서 어드밴티지 펑션 $A^{\pi_\theta}(s,a)$을 사용하여 폴리시 그레디언트를 다시 구성하면,

$$\begin{aligned}A^{\pi_\theta}(s,a) &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \\ \nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A^{\pi_\theta}(s,a)]\end{aligned} \tag{16}$$

- **어드밴티지 펑션으로 추정하기**
  - 어드밴티지 펑션은 폴리시 그레디언트의 분산을 **급격히** 줄인다.

- 그래서 크리틱은 어드밴티지 펑션으로 추정하는 것이 좋다.
- $V^{\pi_\theta}(s)$ 와 $Q^{\pi_\theta}(s,a)$ 의 추정을 위해, 2개의 펑션 근사와 2개의 파라메터 벡터를 사용한다.

$$\begin{aligned} V_v(s) &\approx V^{\pi_\theta}(s) \\ Q_w(s,a) &\approx Q^{\pi_\theta}(s,a) \\ A(s,a) &\approx Q_w(s,a) - V_v(s) \end{aligned} \tag{17}$$

- 그리고 두 가치 함수를 TD-러닝 같은 방법으로 모두 업데이트 한다.
- 액터-크리틱은 폴리시 학습에 사용되는 가중치(θ)와, q를 학습하기 위한 가중치(w)가 필요했는데,
- 어드밴티지 펑션을 위해서는 v(상태-가치)를 학습하기 위한 가중치(v)가 추가적으로 필요하다.

- **어드밴티지 펑션을 개선**
  - 트루 가치 함수(true value function, 오라클이 만들어준 완전한 펑션) $V^{\pi_\theta}(s)$ 로 가정하면, TD 에러 δ^{π_θ}는,

$$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s) \tag{18}$$

  - 이 TD 에러는 어드밴티지 펑션 ($A^{\pi_\theta}(s,a)$)의 편향되지 않은 추정이 된다.

$$\begin{aligned} E_{\pi_\theta}[\delta^{\pi_\theta}|s,a] &= \mathbb{E}_{\pi_\theta}[r + \gamma V^{\pi_\theta}(s')|s,a] - V^{\pi_\theta}(s) \\ &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \\ &= A^{\pi_\theta}(s,a) \end{aligned} \tag{19}$$

  - δ^{π_θ} 의 기대값이 $A^{\pi_\theta}(s,a)$ 라는 것은 δ^{π_θ}를 많이 수행하면 결국 $A^{\pi_\theta}(s,a)$에 수렴한다는 것이고,
  - 이것은 TD 에러가 어드밴티지 펑션의 샘플(어떤 분포의 샘플이라는 것)이 된다는 것이다. 그러므로, 폴리시 그레디언트를 계산하기 위해 TD 에러를 사용하면 된다.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\delta^{\pi_\theta}] \tag{20}$$

  - 실제에서는 트루 가치 함수 $V^{\pi_\theta}(s)$ 대신, TD 에러 근사를 그냥 사용하면 된다.

$$\delta_v = r + \gamma V_v(s') - V_v(s) \tag{21}$$

  - 결론은 크리틱을 근사하기 위해 두개(A=Q-V)를 학습하였으나, Q학습이 필요 없어지고, V만 학습하면 된다.
  - 그래서 어드밴티지 펑션 자리에 TD 에러를 사용하면 된다.

4.2 폴리시 그레디언트 알고리즘 정리

$$
\begin{aligned}
\nabla_\theta J(\theta) = \quad & \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\ v_t] && \text{REINFORCE} \\
& \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\ Q^w(s,a)] && \text{Q 액터-크리틱} \\
& \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\ A^w(s,a)] && \text{어드밴티지 액터-크리틱} \\
& \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\ \delta] && \text{TD 액터-크리틱}
\end{aligned}
$$

- 모두 확률적 경사 상승(stochastic gradient ascent) 알고리즘을 사용한다.
- 크리틱 학습은 $Q^\pi(s,a), A^\pi(s,a), V^\pi(s)$ 중에 하나를 추정하기 위해, 정책 평가(MC나 TD 학습)를 사용한다.