05. 모델 프리 콘트롤

김호철

Contents

---

1 소개

- 지난 장은 **모델 프리 프리딕션** : 알지 못하는 MDP의 가치 함수를 추정(Estimate)
- 이번 장은 **모델 프리 콘트롤** : 알지 못하는 MDP의 가치 함수를 최적화(Optimise)

- MDP로 모델 될 수 있는 문제들 : 엘리베이터, 로보컵 축구, 병렬 주차, 퀘이크, 배 조정, 주식 포트폴리오 관리, 생물반응기, 단백질 접합, 헬리곱터, 로봇 걷기, 항공 화물
- 이러한 문제 대부분은 다음 중 하나이다.
  - 샘플링은 할 수 있지만, MDP 모델은 알지 못한다.
  - MDP 모델은 알지만, 샘플 외에는 사용하기에 너무 크다.
- 모델 프리 콘트롤로 이러한 문제를 해결할 수 있다.
- **온-폴리시(On-policy)** 학습
  - 작업을 수행하면서 그 작업에서 학습

– π에서 샘플링된 경험으로부터 정책 π를 학습

- **오프-폴리시(Off-policy)** 학습
  – 누군가의 어깨 너머로 보고 학습
  – μ에서 샘플링된 경험으로부터 정책 π를 학습

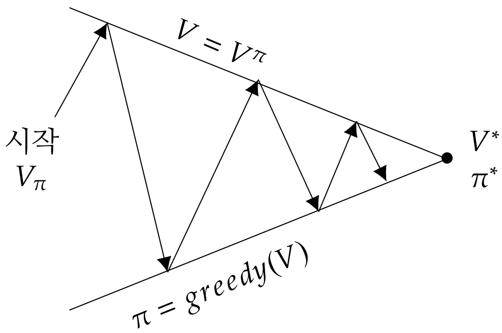
- **이터레이션**
  – 한번의 수행으로 해를 찾을 수 없고, 반복적으로 수행하면 최적의 해가 찾아지는 것이 벨만 최적 방정식인데, 이때의 반복을 **이터레이션**이라 한다.
  – **정책 이터레이션** : 명시적인 정책이 액션을 직접 결정한다.(정책 평가+정책 개선)
  – **가치 이터레이션** : 가치 함수를 보고 액션이 결정된다.

2 온-폴리시 몬테카를로 콘트롤

2.1 일반화된 정책 이터레이션(Generalised Policy Iteration)

- **정책 이터레이션**
  – 정책 평가 : 반복적인 정책 평가로 V_π를 추정(Estimate)
  – 정책 개선 : 탐욕적 정책 개선으로 $\pi' \geq \pi$를 생성

- **몬테카를로** 평가를 통한 정책 이터레이션



  – **정책 평가** : 몬테카를로 정책 평가, $V = v_\pi$?
  – **정책 개선** : 탐욕적 정책 개선?
  – DP에서는 모두 가능하였으나, MC 에서는 정책 평가는 가능하지만 정책 개선은 불가능하다.
  – 다음 상태들을 모두 알아야 그 중에 최대를 찾아 정책 개선이 가능한데, MC는 다음 상태중에 하나만 알 수 있는 구조이다.

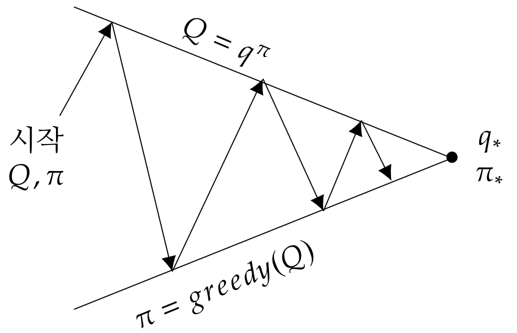
- **액션-가치 함수**를 통한 **모델 프리 정책 이터레이션**
  – $V(s)$에 대한 탐욕적인 정책 개선에는 MDP의 모델이 필요하다(MDP 모델을 알아야 한다).

$$\pi'(s) = \underset{a \in A}{\operatorname{argmax}} R_s^a + P_{ss'}^a V(s') \tag{1}$$

– $Q(s,a)$에 대한 탐욕적인 정책 개선은 모델이 필요하지 않다(모델 프리)

$$\pi'(s) = \underset{a\in A}{\operatorname{argmax}}\, Q(s,a) \tag{2}$$

- **액션-가치 함수**를 통한 **일반화된 정책 이터레이션**



  – **정책 평가** : 몬테카를로 정책 평가, $Q = q_\pi$
  – **정책 개선** : 탐욕적 정책 개선? You can stuck.

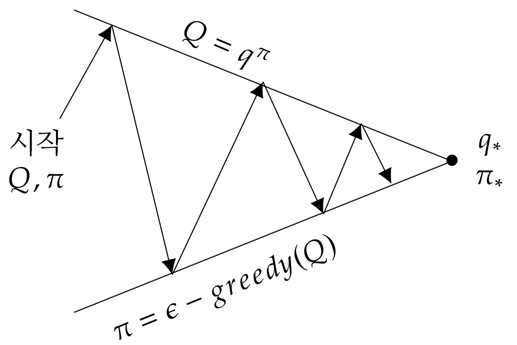
2.2 탐색(Exploration)

- **ϵ(입실론) 탐욕 탐색**
  – 지속적인 탐색을 보장하기 위한 가장 간단한 아이디어
  – 모든 m개의 액션이 0이 아닌 확률로 시도된다.
  – 확률 $1-\epsilon$ 만큼은 탐욕적 액션 선택
  – 확률 ϵ 만큼은 랜덤 액션 선택

$$\pi(a|s) = \begin{cases} \dfrac{\epsilon}{m} + 1 - \epsilon & if\ a^* = \underset{a\in A}{\operatorname{argmax}}\, Q(s,a) \\ \dfrac{\epsilon}{m} & \text{otherwise} \end{cases}$$

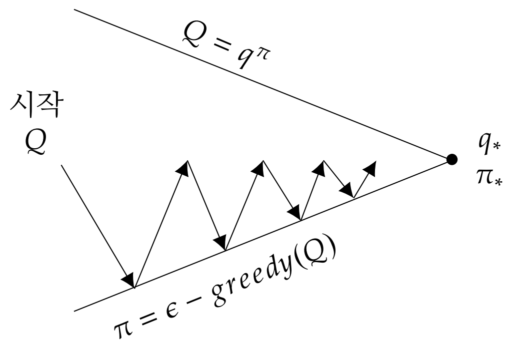
- **ϵ 탐욕 정책 개선**
  – **정리(Theorem)** : q_π에 ϵ 탐욕 정책을 수행하면 $q_{\pi'}$는 개선 된다. $v_{\pi'}(s) \geq v_\pi(s)$

- **몬테카를로 정책 이터레이션**



- **정책 평가** : 몬테카를로 정책 평가, $Q = q_\pi$
- **정책 개선** : ϵ 탐욕 정책 개선

- **몬테카를로 콘트롤**



- 에피소드 마다 :
- **정책 평가** : 몬테카를로 정책 평가, $Q \approx q_\pi$
- **정책 개선** : ϵ 탐욕 정책 개선

2.3 GLIE

정의(Definition)

Greedy in the Limit with Infinite Exploration (무한한 탐색에서 제한된 탐욕)

잘 수련되기 위한 2가지 성질

(1)모든 상태-액션 쌍들이 무한하게 여러번 반복되어야 한다.

(2)정책이 탐욕 정책으로 수렴되어야 한다

- ϵ이 $\epsilon_k = \frac{1}{k}$에서 0으로 감소하면 ϵ 탐욕은 GLIE 이다.

정리(Theorem)

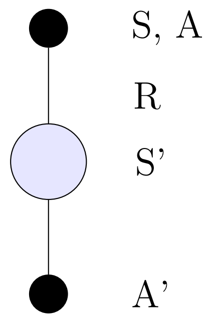
GLIE 몬테카를로 콘트롤은 최적의 액션-가치 함수 $Q(s, a) \to q_*(s, a)$로 수렴한다.

3 온-폴리시 TD 학습

- MC에 비해 TD는 몇가지 장점이 있다 : 낮은 분산, 온라인, 종료되지 않은 에피소드
- 콘트롤 루프에서 MC대신 TD를 사용하는 것은 자연스러운 아이디어
- Q(S,A)에 TD 적용, ϵ-탐욕 정책 사용, 타임 스텝마다 업데이트

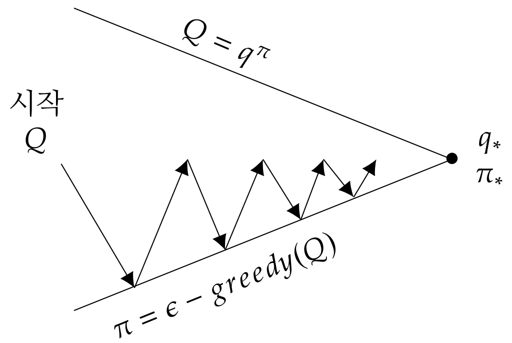
3.1 Sarsa(λ)

- **Sarsa(살사)에서 액션-가치 함수 업데이트**



$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A)) \tag{3}$$

- **Sarsa(살사)에서 온-폴리시 콘트롤**



  - 타임 스텝마다,
  - **정책 평가** : Sarsa $Q \approx q_\pi$
  - **정책 개선** : ϵ 탐욕 정책 개선

- **온-폴리시 콘트롤 살사 알고리즘**

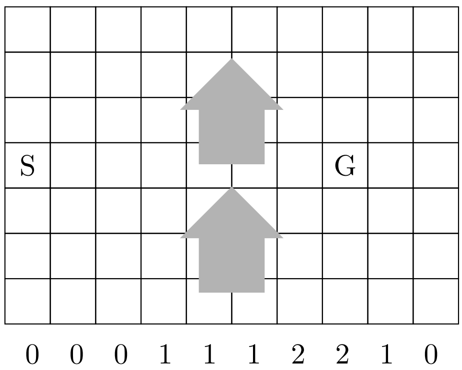
모든 $s \in S, a \in A(s)$에 대해 $Q(s,a)$를 임의의 값으로 초기화, 종료 상태 $Q = 0$
새로운 에피소드 마다 반복 :
    S 초기화
    Q에서 제공되는 정책으로 S에서 A를 선택(ϵ-탐욕)
    에피소드의 각 단계를 반복 :
        액션 A를 얻고, R과 S'를 관측(observe)
        Q에서 제공되는 정책으로 S'에서 A'를 선택(ϵ-탐욕)
        $Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$
        $S \leftarrow S', A \leftarrow A'$
    S가 종료일 때 까지

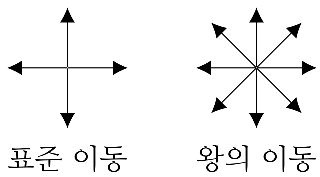
- **살사의 수렴**

> **정리(Theorem)**
>
> 살사는 다음의 조건에서 최적 액션-가치 함수 $Q(s,a) \to q_*(s,a)$로 수렴한다.
>
> - 정책 $\pi_t(a|s)$의 GLIE 시퀀스
>
> - 스텝 크기 α_t의 Robbins-Monro 시퀀스

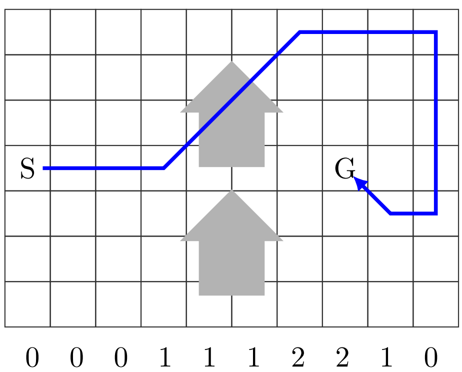
$$
\begin{aligned}
\sum_{t=1}^{\infty} \alpha_t &= \infty \\
\sum_{t=1}^{\infty} \alpha_t^2 &< \infty
\end{aligned} \tag{4}
$$

- **윈디 그리드월드 문제**





  - 보상은 Goal에 도착할 때까지 타임 스텝마다 -1
  - 할인율은 없음



- **n-스텝 살사** : 미래의 n 단계 만큼 계산하는 Sarsa
  - n이 1, 2, ∞ 에 대하여 n-스텝 리턴을 고려해보면,

$$
\begin{aligned}
&\color{red}{n = 1\ (Sarsa)} && q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}) \\
&\color{red}{n = 2} && q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}) \\
& && \vdots \\
&\color{red}{n = \infty\ (MC)} && G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + ... \gamma^{T-1} R_T
\end{aligned} \tag{5}
$$

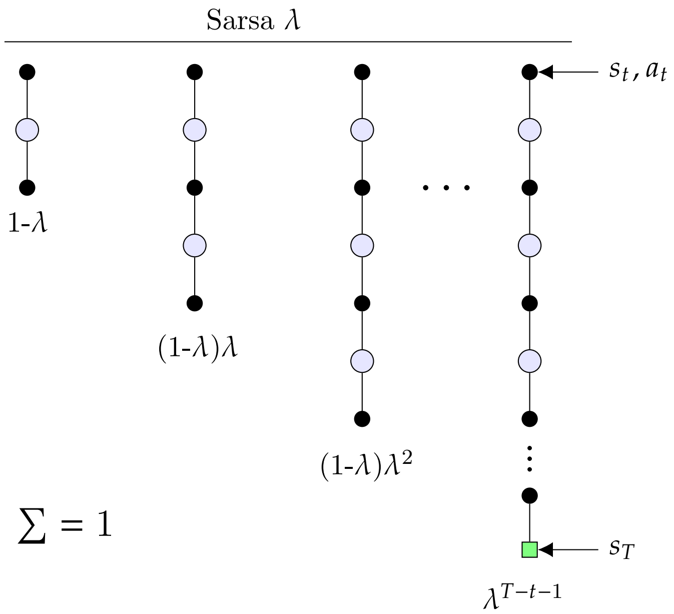
– n-스텝 Q 리턴을 정의하면,

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n}) \tag{6}$$

– n-스텝 살사는 n-스텝 Q 리턴 방향으로 $Q(s, a)$를 업데이트 한다.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(q_t^{(n)} - Q(S_t, A_t) \right) \tag{7}$$

- **전방 보기(Forward View) 살사(λ)**



– q^λ는 모든 n-스텝 Q-리턴 $q_t^{(n)}$들의 합을 리턴한다.
– 가중치 $(1 - \lambda)\lambda^{n-1}$을 사용하면,

$$q_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} q_t^{(n)} \tag{8}$$

– 전방 보기 살사(λ)

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(q_t^\lambda - Q(S_t, A_t) \right) \tag{9}$$

- **후방 보기(Backward View) 살사(λ)**

– TD(λ)와 유사하게, 온라인 알고리즘에서 eligibility traces를 사용한다.
– 그러나 살사(λ)는 각 상태-액션 쌍에서 하나의 eligibility traces를 가진다.

$$\begin{aligned} E_0(s,a) &= 0 \\ E_t(s,a) &= \gamma\lambda E_{t-1}(s,a) + 1(S_t = s, A_t = a) \end{aligned} \tag{10}$$

– Q(s,a)는 모든 상태 s와 액션 a에 대해 업데이트 된다.
– TD-에러 δ_t는 eligibility trace $E_t(s,a)$에 비례하여,

$$\begin{aligned} \delta_t &= R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \\ Q(s,a) &\leftarrow Q(s,a) + \alpha\delta_t E_t(s,a) \end{aligned} \tag{11}$$

- **살사(λ) 알고리즘**

```
모든 s ∈ S, a ∈ A(s)에 대해 Q(s,a)를 임의의 값으로 초기화
새로운 에피소드 마다 반복 :
    모든 s ∈ S, a ∈ A(s)에 대해 E(s,a) = 0
    S, A 초기화
    에피소드의 각 단계를 반복 :
        액션 A를 주고, R과 S'를 관측(observe)
        Q에서 제공되는 정책으로 S'에서 A'를 선택(ϵ-탐욕)
        δ ← R + γQ(S', A') − Q(S, A)
        E(S, A) ← (S, A) + 1
        모든 s ∈ S, a ∈ A(s)들에 대해
            Q(s, a) ← Q(s, a) + αδE(s, a)
            E(s, a) ← γλE(s, a)
        S ← S', A ← A'
    S가 종료일 때 까지
```

4 오프-폴리시 학습

- **2개의 정책이 필요** : (1)목표(target) 정책 $\pi(a|s)$, (2)행위(behaviour) 정책 $\mu(a|s)$,
- 평가(Evaluate)는 목표 정책으로 하고, 실제 액션 샘플링(상태 이동)은 행위 정책을 따른다.
- 사람이나 다른 에이전트들을 관찰하여 학습
- 오래된 정책 $\pi_1, \pi_2, ..., \pi_{t-1}$에서 축적된 경험을 재사용
- 탐색적 정책을 따르면서 최적의 정책에 대해 학습
- 하나의 정책을 따르면서 다중 정책들을 학습

4.1 중요도 샘플링(Importance Sampling)

- **중요도 샘플링**
  - 다른 분포에 대한 기댓값을 추정
  - 중요도 샘플링의 목적은 기댓값을 계산하고자 하는 확률 분포 $p(x)$의 확률 밀도 함수 (probability density function, PDF)를 알고는 있지만 p에서 샘플을 생성하기가 어려울 때, 비교적 샘플을 생성하기가 쉬운 $q(x)$에서 샘플을 생성하여 p의 기댓값을 계산하는 것
  - 결론적으로 비율을 곱해주면 된다.

$$\begin{aligned}\mathbb{E}_{X\sim P}[f(X)] &= \sum P(X)f(X)\\ &= \sum Q(X)\frac{P(X)}{Q(X)}f(X)\\ &= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)}f(X)\right]\end{aligned} \tag{12}$$

- **오프-폴리시 MC 중요도 샘플링**
  - π를 평가하기 위해 μ로부터 생성된 리턴을 사용
  - 두 정책들간의 유사성(similarity)을 따르는 가중치 리턴 G_t
  - 전체 에피소드에 걸쳐 중요도 샘플링 보정을 곱한다.

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}\frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})}\cdots\frac{\pi(A_T|S_T)}{\mu(A_T|S_T)}G_t \tag{13}$$

  - 보정된 리턴으로 $V(S_t)$를 업데이트

$$V(S_t) = V(S_t) + \alpha\left(G_t^{\pi/\mu} - V(S_t)\right) \tag{14}$$

  - μ가 0이고 π가 0이 아니면 사용할 수 없다.
  - MC방법으로 중요도 샘플링은 분산도가 극도로 커서 현실에서 사용이 어렵다.

- **오프-폴리시 TD 중요도 샘플링**
  - π를 평가하기 위해 μ로부터 생성된 TD 타겟을 사용
  - 중요도 샘플링으로 가중치 TD 타겟 $R + \gamma V(S')$
  - 오직 한번의 중요도 샘플링 보정이 필요

$$V(S_t) \leftarrow V(S_t) + \alpha\left(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}(R_{t+1} + \gamma V(S_{t+1})) - V(s_t)\right) \tag{15}$$

  - MC 중요도 샘플링 방법보다 매우 낮은 분산도
  - 두개의 정책(μ, π)들이 오직 각 단일 스텝에서만 유사하면 된다.

4.2 Q-러닝

- **Q-러닝 개요**
  - 액션-가치 Q(s,a)에 대한 오프폴리시 학습을 고민
  - 중요도 샘플링이 필요 없다.
  - 다음 액션은 행위 정책 $A_{t+1} \sim \mu(\cdot|S_t)$에서 선택되지만,
  - $Q(S_t, A_t)$의 업데이트는 타겟 정책 $A' \sim \pi(\cdot|S_t)$에 의해 업데이트 된다.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t)) \tag{16}$$

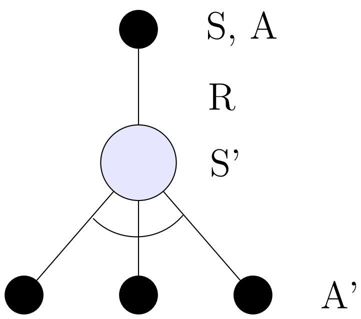
- **Q-러닝 오프-폴리시 콘트롤**
  - 행위 정책과 타겟 정책 모두 업데이트 필요
  - 타겟 정책 π는 $Q(s, a)$에 대해 탐욕적(greedy)이다.

$$\pi(S_{t+1}) = \underset{a'}{\mathrm{argmax}}\ Q(S_{t+1}, a) \tag{17}$$

  - 행위 정책 μ는 $Q(s, a)$에 대해 ϵ-탐욕을 사용
  - Q-러닝 타겟은 다음처럼 단순해진다.

$$\begin{aligned} & R_{t+1} + \gamma Q(S_{t+1}, A') \\ = & R_{t+1} + \gamma Q(S_{t+1}, \underset{a'}{\mathrm{argmax}}\, Q(S_{t+1}, a')) \\ = & R_{t+1} + max\ \gamma Q(S_{t+1}, a') \end{aligned} \tag{18}$$

- **Q-러닝 콘트롤 알고리즘**



$$Q(S, A) \leftarrow Q(S, A) + \alpha\left(R + \gamma \max_{a'} Q(S', a') - Q(S, A)\right) \tag{19}$$

정리(Theorem)

Q-러닝 콘트롤은 최적의 액션-가치(action-value) 함수로 수렴한다.

$$Q(s, a) \rightarrow q_*(s, a)$$

  - **오프-폴리시 콘트롤 Q-러닝 알고리즘**

모든 $s \in S, a \in A(s)$에 대해 $Q(s,a)$를 임의의 값으로 초기화, 종료 상태 $Q = 0$
새로운 에피소드 마다 반복 :
    S 초기화
    에피소드의 각 단계를 반복 :
        Q에서 제공되는 정책으로 S에서 A를 선택(ϵ-탐욕)
        액션 A를 얻고, R과 S'를 관측(observe)
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big(R + \gamma \max_a Q(S',a') - Q(S,A)\big)$
        $S \leftarrow S'$
    S가 종료일 때 까지

5 요약

5.1 DP와 TD 비교

	풀 백업(DP)	샘플 백업(TD)
$v_\pi(s)$ 벨만 기대 방정식	반복적(Iterative) 정책 평가 $V(s) \leftarrow \mathbb{E}[R + \gamma V(S') \mid s]$	TD 학습 $V(S) \xleftarrow{\alpha} R + \gamma V(S')$
$q_\pi(s)$ 벨만 기대 방정식	Q-정책(Policy) 이터레이션 $Q(s,a) \leftarrow \mathbb{E}[R + \gamma Q(S',A') \mid s,a]$	살사(Sarsa) $Q(S,A) \xleftarrow{\alpha} R + \gamma Q(S',A')$
$q_*(s)$ 벨만 최적 방정식	Q-가치(Value) 이터레이션 $Q(s,a) \leftarrow \mathbb{E}\big[R + \gamma \max_{a' \in A} Q(S',a') \mid s,a\big]$	Q-러닝 $Q(S,A) \xleftarrow{\alpha} R + \gamma \max_{a' \in A} Q(S',a')$