Ch 6

MDP를 모를 때 최고의 정책 찾기

Ch4 review

1. 상태집합 s나 액션의 집합 a의 크기가 작은 경우
2. MDP를 알 때 = 보상함수와 전이확률을 알고 있을 경우

가정 → **정책 개선 과정(플래닝)**

Tabular method

(s, a)에 대한 테이블을 업데이트하는 방식

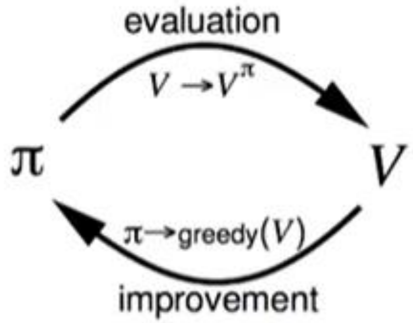
prediction

정책 π가 주어졌을 때 각 상태의 밸류를 평가하는 문제

control

최적의 정책 함수를 찾는 문제

planning



정책 이터레이션:
임의의 정책에서 시작하여 정책을 평가, 밸류를 계산하고, 계산한 밸류에 대해 그리디 정책을 만드는 과정을 반복

Introduction

MDP를 모를 때(모델-프리) 정책 이터레이션을 그대로 사용 가능한가?

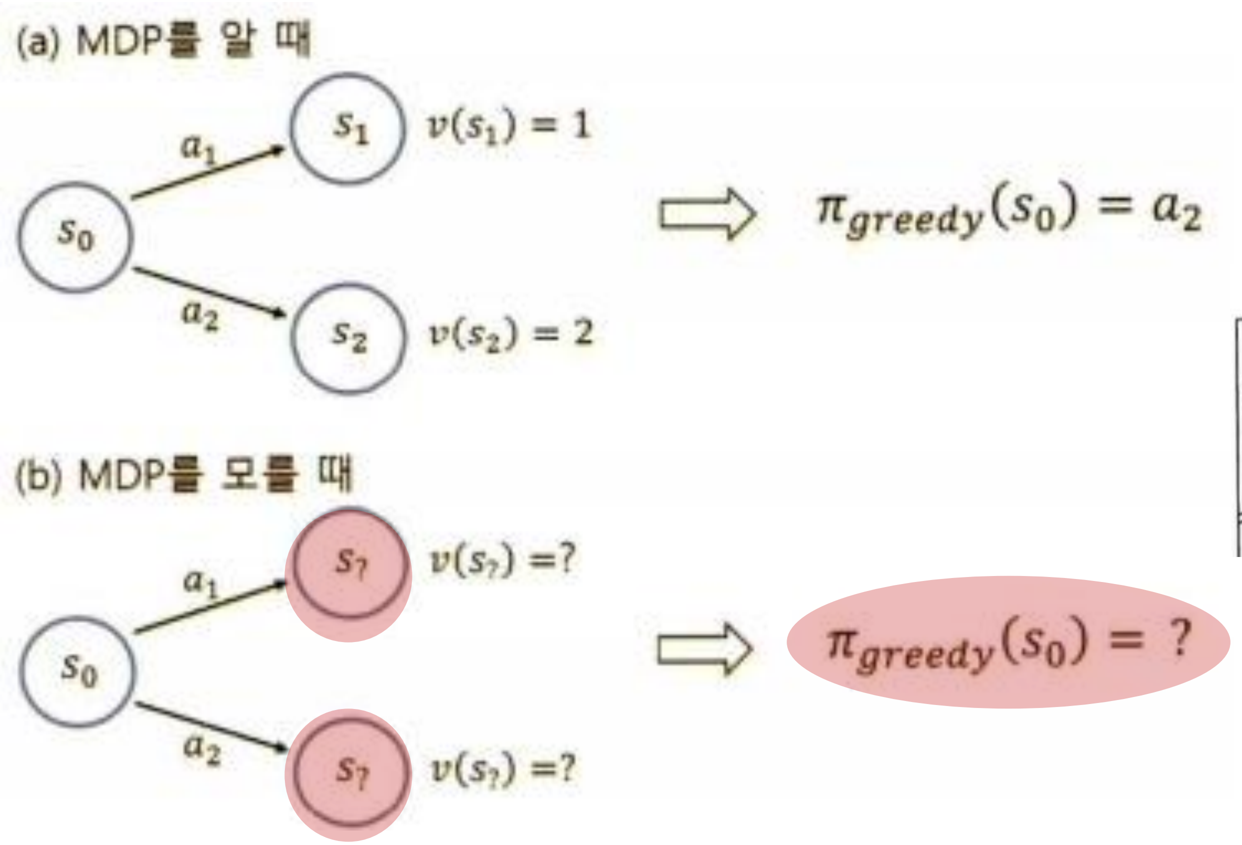
1. 정책평가

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$

Introduction

MDP를 모를 때(모델-프리) 정책 이터레이션을 그대로 사용 가능한가?

2. 정책개선



- There are two doors in front of you.
- You open the left door and get reward 0
  $V(left) = 0$
- You open the right door and get reward +1
  $V(right) = +1$
- You open the right door and get reward +3
  $V(right) = +2$

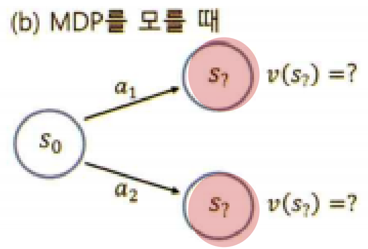
Introduction

MDP를 모를 때(모델-프리) 컨트롤 방법

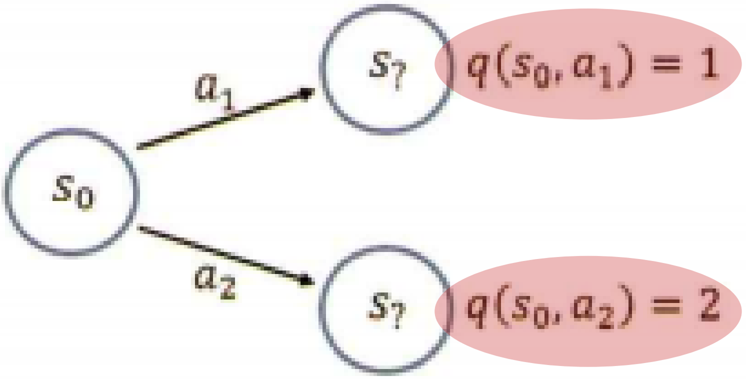
$$v_\pi(s) = \sum_{a \in A} \pi(a|s)\left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$

MC / TD

상태-액션 가치함수 q(s,a)



$\pi_{greedy}(s_0) = ?$



$\pi_{greedy}(s_0) = a_2$

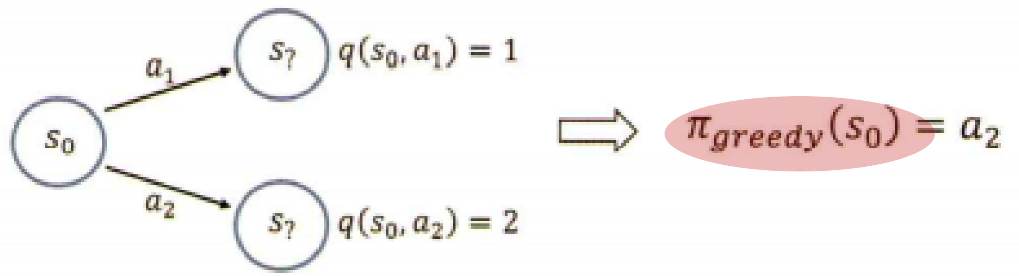
Introduction

MDP를 모를 때(모델-프리) 컨트롤 방법

$$v_\pi(s) = \sum_{a\in A} \pi(a|s)\left(r_s^a + \gamma \sum_{s'\in S} P_{ss'}^a\, v_\pi(s')\right)$$

MC / TD

상태-액션 가치함수 q(s,a)



$$\pi(a|s) = \begin{cases} 1-\varepsilon \text{ if } a^* = \underset{a}{\operatorname{argmax}}\, q(s,a) \\ \varepsilon \qquad\qquad\qquad \text{otherwise} \end{cases}$$

Decaying ε-greedy

Introduction

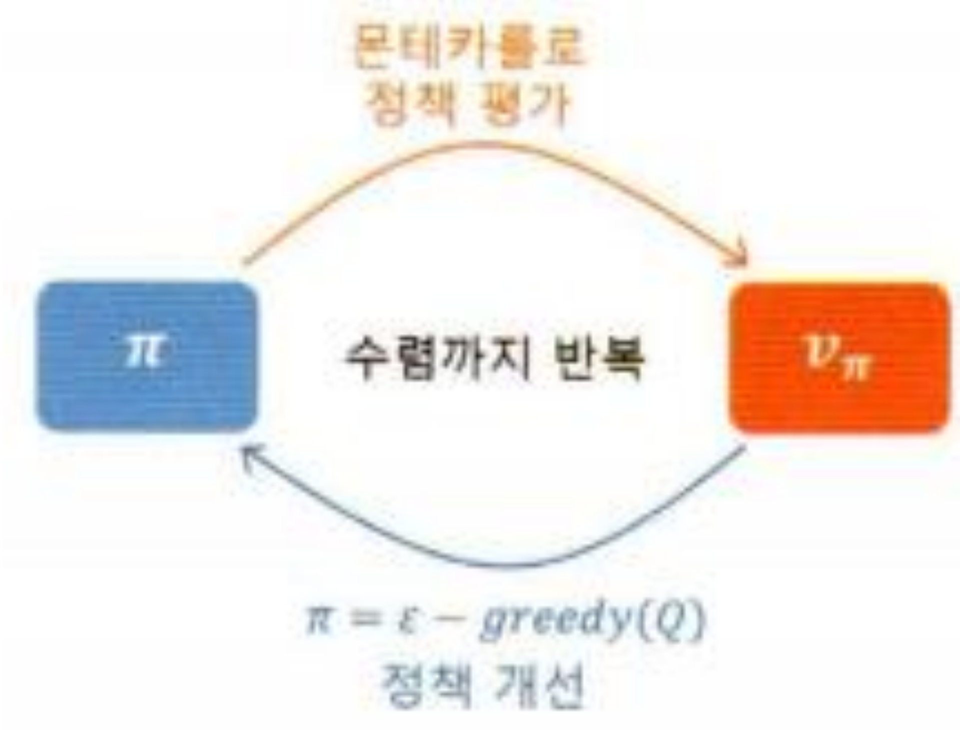
(extra) ε-greedy와 정책 개선

For any ϵ-greedy policy π, the ϵ-greedy policy π' with respect to q_π is an improvement, $v_{\pi'}(s) \geq v_\pi(s)$

$$\begin{aligned}
q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s,a) \\
&= \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s,a) + (1-\epsilon) \max_{a \in \mathcal{A}} q_\pi(s,a) \\
&\geq \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s,a) + (1-\epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \epsilon/m}{1-\epsilon} q_\pi(s,a) \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a) = v_\pi(s)
\end{aligned}$$

Introduction

MDP를 모를 때(모델-프리) 컨트롤 방법

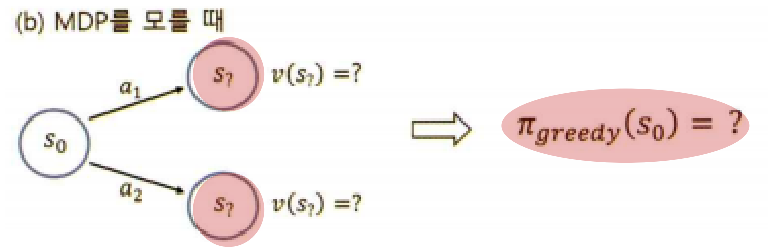


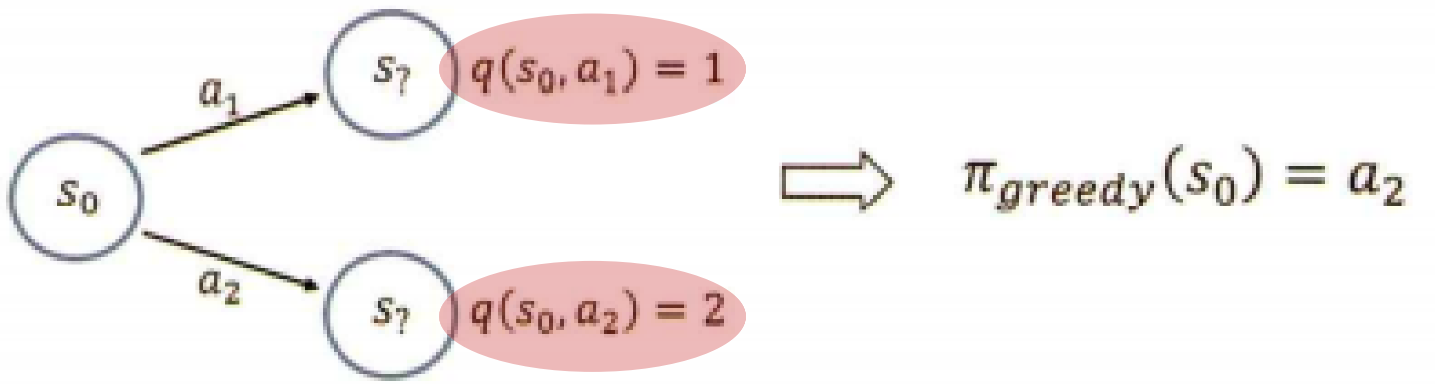
몬테카를로 컨트롤

몬테카를로 컨트롤의 프로세스

$$v_\pi(s) = \sum_{a\in A}\pi(a|s)\left(r_s^a + \gamma\sum_{s'\in S}P_{ss'}^a v_\pi(s')\right)$$

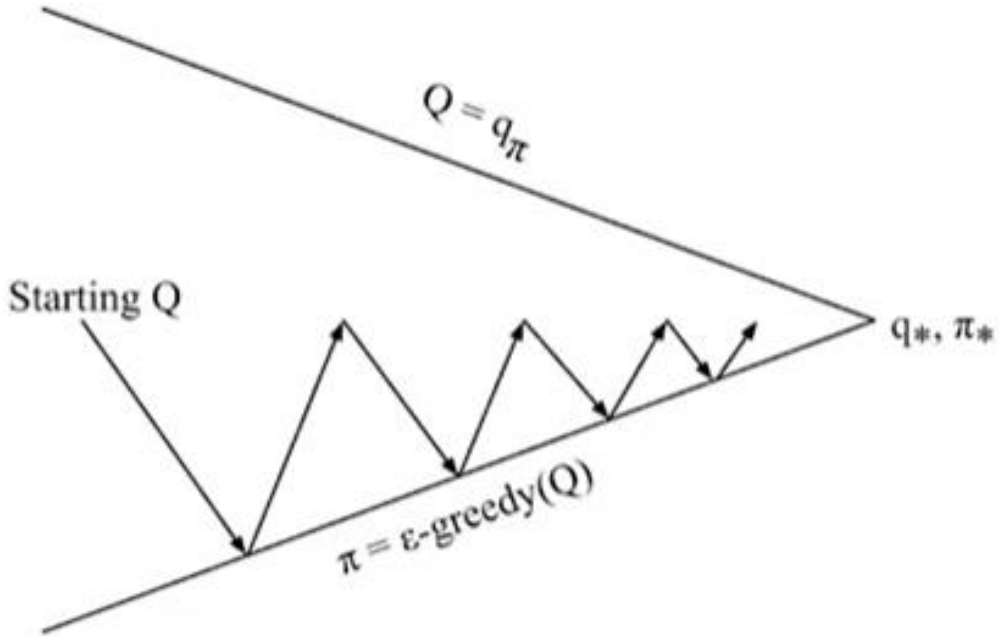
→ 몬테카를로 방법론 → 상태-액션 가치함수 q(s,a)





몬테카를로 컨트롤

업데이트 방식 및 특징



Every episode:

Policy evaluation Monte-Carlo policy evaluation, $Q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

몬테카를로 컨트롤

GLIE Monte-Carlo 컨트롤

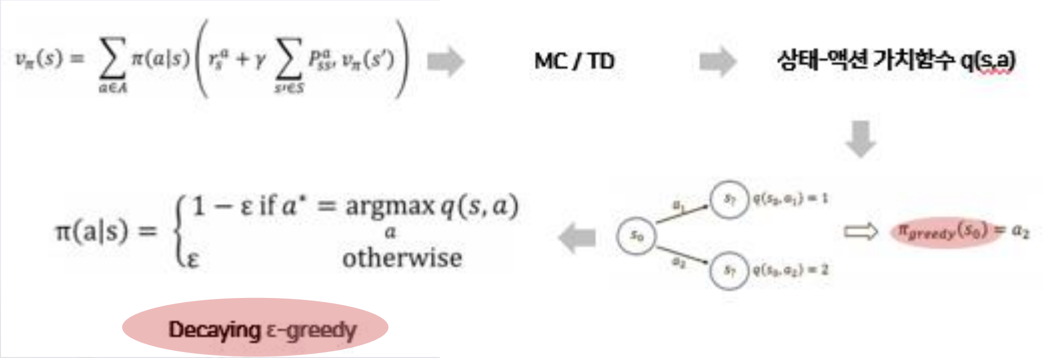
Greedy in the Limit with Infinite Exploration (GLIE)

- All state-action pairs are explored infinitely many times,

$$\lim_{k\to\infty} N_k(s,a) = \infty$$

- The policy converges on a greedy policy,

$$\lim_{k\to\infty} \pi_k(a|s) = \mathbf{1}(a = \underset{a'\in\mathcal{A}}{\operatorname{argmax}}\, Q_k(s,a'))$$

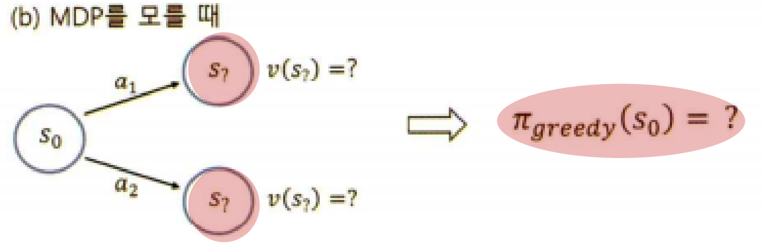


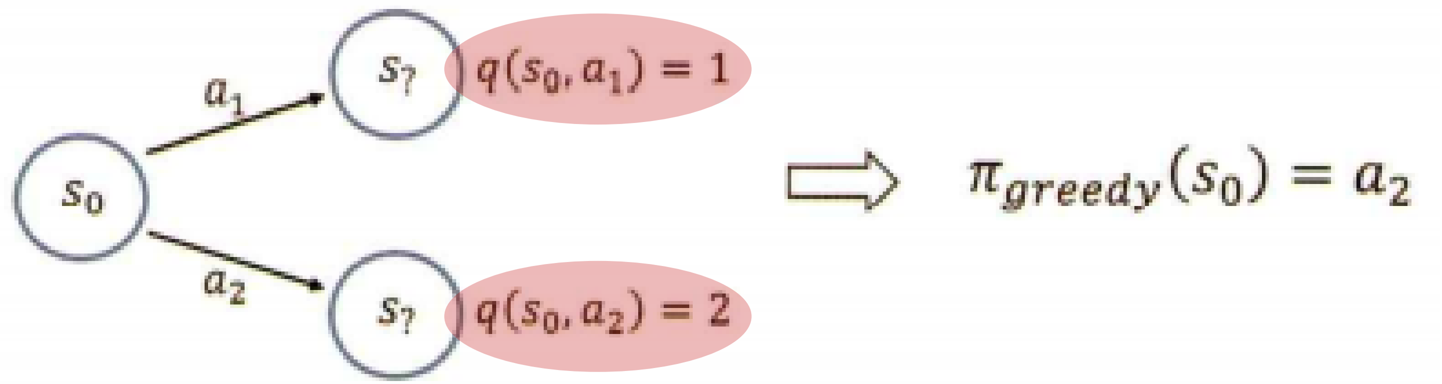
TD 컨트롤

MDP를 모를 때(모델-프리) 컨트롤 방법

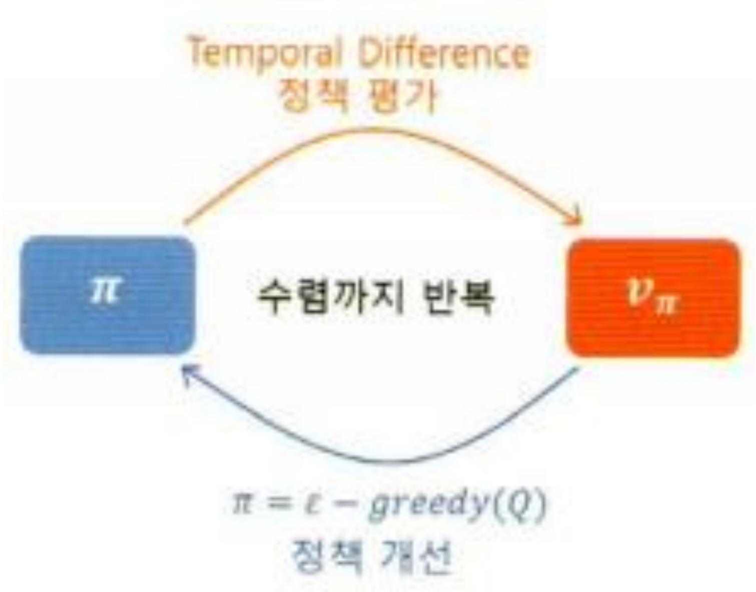
$$v_\pi(s) = \sum_{a \in A} \pi(a|s)\left(r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$

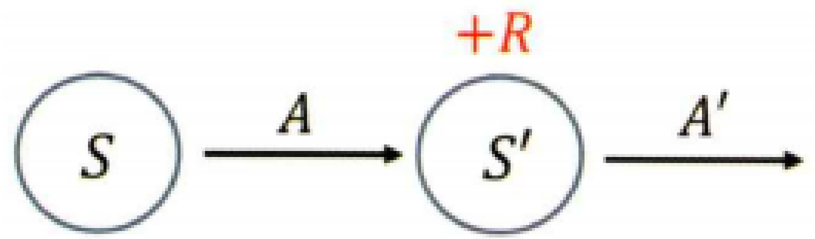
→ TD → 상태-액션 가치함수 q(s,a)





TD 컨트롤1 -SARSA

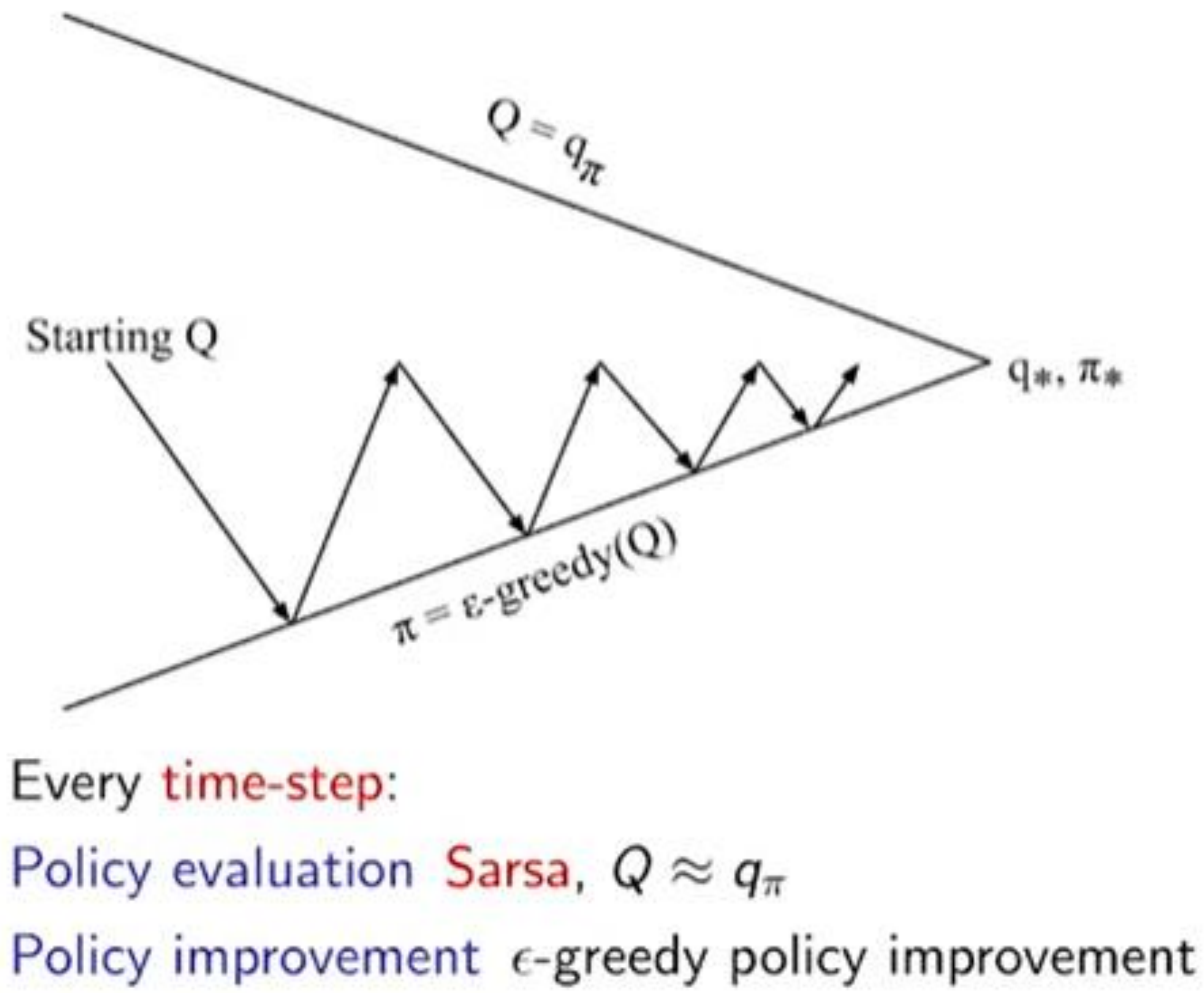




TD 에러

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma Q(S',A') - Q(S,A)\right)$$

TD 컨트롤1 -SARSA



TD 컨트롤1 -SARSA

SARSA 전제 조건

Sarsa converges to the optimal action-value function, $Q(s,a) \rightarrow q_*(s,a)$, *under the following conditions:*

- *GLIE sequence of policies* $\pi_t(a|s)$
- *Robbins-Monro sequence of step-sizes* α_t

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

TD 컨트롤2- Q러닝

On-policy / Off-policy

on-policy: 타깃 정책과 행동 정책이 같은 경우(SARSA)

off-policy: 타깃 정책과 행동 정책이 다른 경우(Q러닝)

- Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$
- While following behaviour policy $\mu(a|s)$

$$\{S_1, A_1, R_2, ..., S_T\} \sim \mu$$

Off-policy 장점

1. 과거의 경험 재사용

2. 사람의 데이터로부터 학습

3. 일대다, 다대일 학습 가능

TD 컨트롤2 - Q러닝

업데이트방식과 벨만 방정식

$$q_*(s,a) = r_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} q_*(s',a')$$

$$q_*(s,a) = \mathbb{E}_{s'}\left[\mathrm{r} + \gamma \max_{a'} q_*(s',a')\right]$$

벨만 최적방정식 0단계

Next action is chosen using behaviour policy $A_{t+1} \sim \mu(\cdot|S_t)$

But we consider alternative successor action $A' \sim \pi(\cdot|S_t)$

Q러닝 : $Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma \max_{A'} Q(S',A')\text{-}Q(S,A))$

TD 컨트롤 - SARSA와 Q러닝

on-policy vs off-policy

	SARSA	Q러닝
행동 정책 (Behavior Policy)	Q에 대해 ε-Greedy	Q에 대해 ε-Greedy
타깃 정책 (Target Policy)	Q에 대해 ε-Greedy	Q에 대해 Greedy

TD 컨트롤 - SARSA와 Q러닝

on-policy vs off-policy

SARSA : $q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$ (벨만 기대 방정식)

Q러닝 : $q_*(s, a) = \mathbb{E}_{s'}[\mathrm{r} + \gamma \max_{a'} q_*(s', a')]$ (벨만 최적 방정식)

SARSA: 샘플기반 방법

Q러닝: 임의의 정책으로부터 최적 정책

TD 컨트롤 - SARSA와 Q러닝

on-policy vs off-policy

SARSA : $q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})]$ (벨만 기대 방정식)

Q러닝 : $q_*(s, a) = \mathbb{E}_{s'}[\mathrm{r} + \gamma \max_{a'} q_*(s', a')]$ (벨만 최적 방정식)

SARSA: 샘플기반 방법

Q러닝: 임의의 정책으로부터 최적 정책

Further subjects

prediction

정책 π가 주어졌을 때 각 상태의 밸류를 평가하는 문제

Tabular method

(s, a)에 대한 테이블을 업데이트하는 방식

planning

control

최적의 정책 함수를 찾는 문제