Model-free Reinforcement Learning

Slides from

1. 이웅원 외, 파이썬과 케라스로 배우는 강화학습, 주교재
2. 이웅원, 가깝고도 먼 DeepRL, PPT
3. David Silver, Reinforcement Learning, PPT

References

1. Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, MIT Press
2. 유튜브, 전민영, 노승은, 강화학습의 기초 이론, 팡요랩

Contents

Model-Free Reinforcement Learning

- Last lecture:
  - Planning by dynamic programming
  - Solve a *known* MDP
- This lecture:
  - Model-free prediction
  - Estimate the value function of an *unknown* MDP
- Next lecture:
  - Model-free control
  - Optimise the value function of an *unknown* MDP

DP vs RL

- DP
  - Model-based Planning
- RL
  - Model-free Learning
  - Prediction (예측) – 주어진 정책에 대한 가치함수를 학습
    -> Monte Carlo (MC), Temporal Difference (TD)
  - Control (제어) – 가치함수를 토대로 정책을 발전시켜 최적 정책을 학습
    -> MC control, SARSA, Q-Learning
- DP vs RL
  - Policy evaluation --> Prediction
  - Policy improvement --> Control

DP vs RL

- DP
  - 환경에 대한 정확한 지식을 가지고 모든 상태에 대해 가치함수를 동시에 계산
    -> 차원의 저주 (Curse of Dimensionality)

- RL
  - 에이전트가 겪은 경험으로부터 학습
    1) 일단 해보고
    2) 자신을 평가하며
    3) 평가한 대로 자신을 업데이트
    4) 위의 과정을 반복

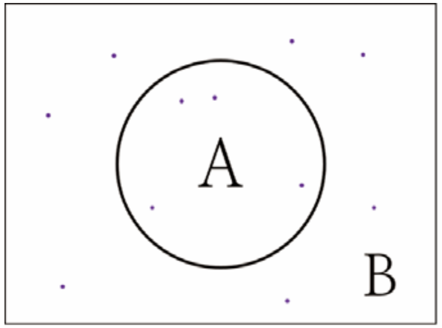
Monte Carlo Approximation

- Monte Carlo Approximation
  - 무작위로 무엇인가를 해본다
  - 샘플을 통해 "원래의 값에 대해 이럴 것이다"라고 추정
  - 샘플의 개수가 많아지면 점점 원래의 값과 비슷해 질 것이다

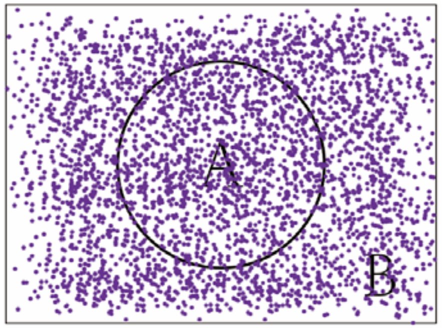
Monte Carlo Approximation

- 원의 넓이 계산 예

$$\frac{S(A)}{S(B)} \sim \frac{\#\ \text{of dots} \in A}{\#\ \text{of dots} \in B}$$



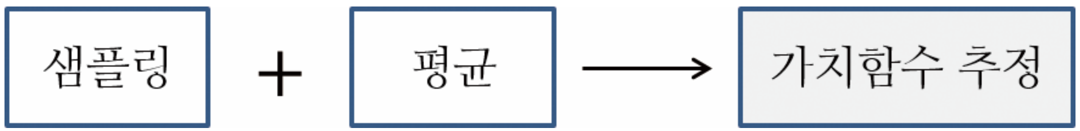
→



-> 점이 하나의 샘플이고, 점을 찍는 것이 샘플링

정책평가 1: Monte Carlo 예측

- 샘플링 – 에이전트가 한 번 환경에서 에피소드를 진행하는 것
  -> 샘플링을 통해 얻은 샘플의 평균으로 참 가치함수의 값을 추정



정책평가 1: Monte Carlo 예측

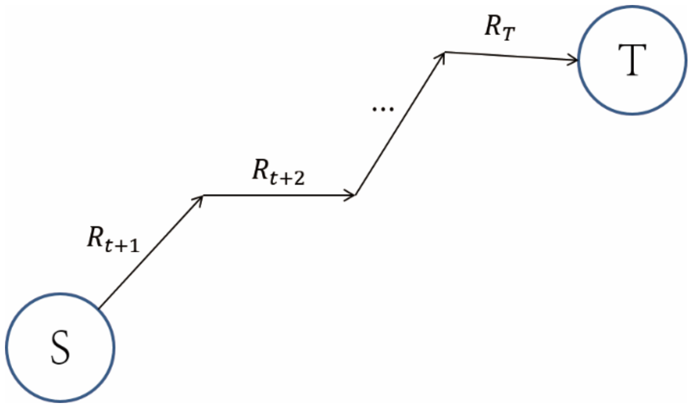
- 가치함수

$$v_\pi(s) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s]$$

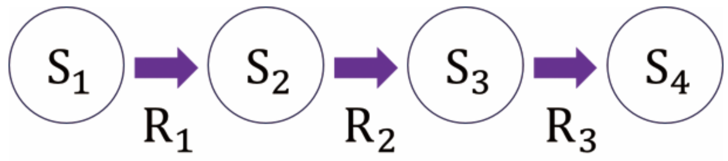
- 정책평가에서 기대값을 계산하지 않고 샘플링을 통해 샘플들의 평균으로 참가치함수를 예측하려면?
  -> 현재 정책에 따라서 계속 행동해 보면 됨
  -> 행동을 하면 그에 따른 보상을 받고,
  -> 에피소드의 끝까지 가면 반환값을 알 수 있음

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$$

정책평가 1: Monte Carlo 예측



샘플링을 통해 하나의 에피소드를 진행



$$G(s_1) = r_1 + \gamma r_2 + \gamma^2 r_3$$

$$G(s_2) = r_2 + \gamma r_3$$

$$G(s_3) = r_3$$

지나온 상태들의 반환값

- 여러 에피소드를 통해 구한 반환값의 평균을 통해 $v_\pi(s)$를 추정

$$v_\pi(s) \sim \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_i(s)$$

정책평가 1: Monte Carlo 예측

- 상태 s에서의 반환값의 평균

$$v_\pi(s) \sim \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_i(s)$$

$$V_{n+1} = \frac{1}{n} \sum_{i=1}^{n} G_i = \frac{1}{n} \left(G_n + \sum_{i=1}^{n-1} G_i \right)$$

$$= \frac{1}{n} \left(G_n + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} G_i \right)$$

$$= \frac{1}{n} (G_n + (n-1) V_n)$$

$$= \frac{1}{n} (G_n + nV_n - V_n)$$

$$= V_n + \frac{1}{n} (G_n - V_n)$$

$$\therefore V(s) \leftarrow V(s) + \frac{1}{n} (G(s) - V(s))$$

정책평가 1: Monte Carlo 예측

- 몬테카를로 예측에서 가치함수의 업데이트 식

$$V(s) \leftarrow V(s) + \frac{1}{n}(G(s) - V(s))$$

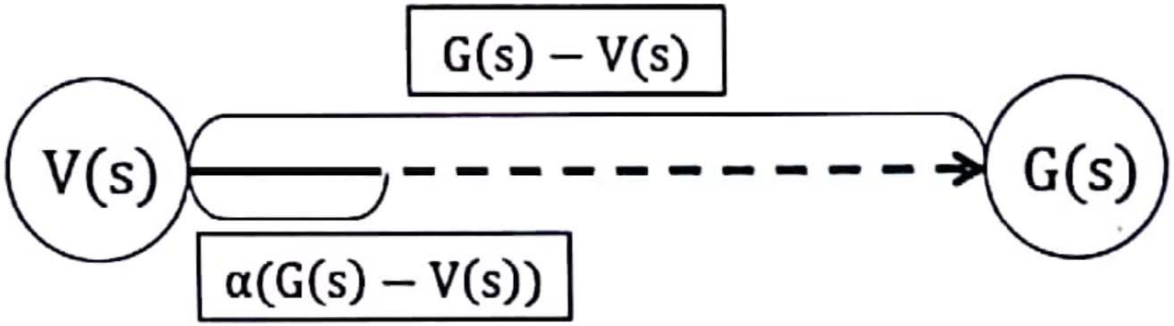
$$where\ error = G(s) - V(s),\ step\ size = 1/n$$

- 몬테카를로 예측에서 가치함수의 업데이트 일반식

$$V(s) \leftarrow V(s) + \alpha(G(s) - V(s))$$

정책평가 1: Monte Carlo 예측

$$V(s) \leftarrow V(s) + \alpha(G(s) - V(s)\ )$$



1. $G(s)$ = 업데이트의 목표
2. $\alpha(G(s) - V(s)\ )$ = 업데이트의 크기

Monte Carlo Reinforcement Learning

- MC methods learn directly from episodes of experience
- MC is *model-free*: no knowledge of MDP transitions / rewards
- MC learns from *complete* episodes: no bootstrapping
- MC uses the simplest possible idea: value = mean return
- Caveat: can only apply MC to *episodic* MDPs
  - All episodes must terminate

David Silver, Reinforcement Learning, PPT

정책평가 2: 시간차 예측

- 몬테카를로 예측의 단점
  - 가치함수를 업데이트하기 위해서는 에피소드가 끝날 때까지 기다려야 함
  - 실시간이 아님
  - 에피소드의 끝이 없거나 에피소드의 길이가 긴 경우에는 적합하지 않음
- 시간차 예측 (Temporal Difference Prediction)
  - 에피소드마다가 아니라 타임스텝마다 가치함수를 업데이트

정책평가 2: 시간차 예측

- 몬테카를로 예측에서의 가치함수 업데이트 식

$$v_\pi(s) = \boldsymbol{E}_\pi[G_t | S_t = s]$$

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

- 시간차 예측에서의 가치함수 업데이트 식

$$v_\pi(s) = \boldsymbol{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

$$V(S_t) \leftarrow V(S_t) + \alpha(R + \gamma V(S_{t+1}) - V(S_t))$$

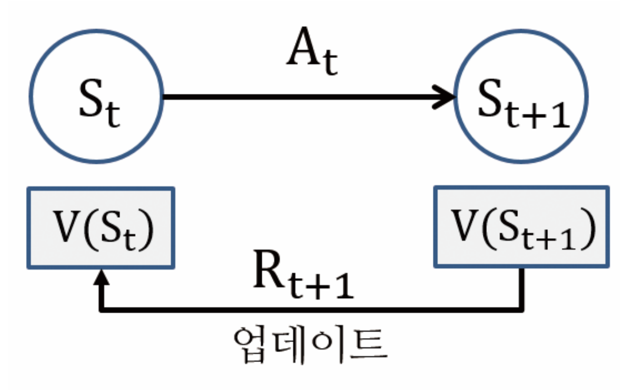
정책평가 2: 시간차 예측

$$V(S_t) \leftarrow V(S_t) + \alpha(R + \gamma V(S_{t+1}) - V(S_t))$$
$$where\ TD\ error = R + \gamma V(S_{t+1}) - V(S_t)$$

1. $R + \gamma V(S_{t+1})$ = 업데이트의 목표
2. $\alpha(R + \gamma V(S_{t+1}) - V(S_t))$ = 업데이트의 크기

- 업데이트의 목표는 실제값이 아닌 에이전트가 가지고 있는 예측치
- Bootstrap : 다른 상태의 가치함수 예측값을 통해 지금 상태의 가치함수를 예측하는 방식

정책평가 2: 시간차 예측



- 어떤 상태에서 행동을 하면 보상을 받고 다음 상태를 알게 되고 다음 상태의 가치함수와 알게 된 보상을 더해 그 값을 업데이트의 목표로 삼음
- 에피소드가 끝날 때까지 기다릴 필요 없이 바로 가치함수를 업데이트 할 수 있음

Temporal-Difference Learning

- TD methods learn directly from episodes of experience
- TD is *model-free*: no knowledge of MDP transitions / rewards
- TD learns from *incomplete* episodes, by *bootstrapping*
- TD updates a guess towards a guess

David Silver, Reinforcement Learning, PPT

MC and TD

- Goal: learn v_π online from experience under policy π
- Incremental every-visit Monte-Carlo
  - Update value $V(S_t)$ toward *actual* return G_t

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $V(S_t)$ toward *estimated* return $R_{t+1} + \gamma V(S_{t+1})$

$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

  - $R_{t+1} + \gamma V(S_{t+1})$ is called the *TD target*
  - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the *TD error*

David Silver, Reinforcement Learning, PPT

Advantages and Disadvantages of MC vs. TD

- TD can learn *before* knowing the final outcome
  - TD can learn online after every step
  - MC must wait until end of episode before return is known
- TD can learn *without* the final outcome
  - TD can learn from incomplete sequences
  - MC can only learn from complete sequences
  - TD works in continuing (non-terminating) environments
  - MC only works for episodic (terminating) environments

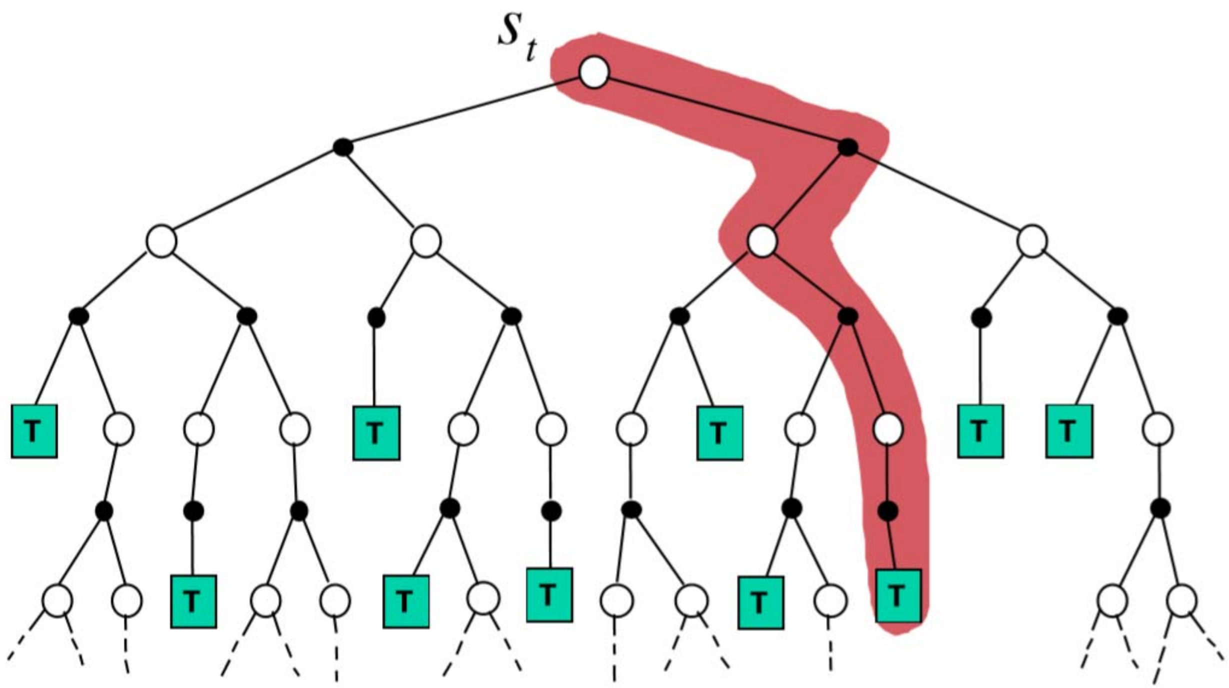
David Silver, Reinforcement Learning, PPT

Bias/Variance Trade-Off

- Return $G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$ is *unbiased* estimate of $v_\pi(S_t)$
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is *unbiased* estimate of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma V(S_{t+1})$ is *biased* estimate of $v_\pi(S_t)$
- TD target is much lower variance than the return:
  - Return depends on *many* random actions, transitions, rewards
  - TD target depends on *one* random action, transition, reward

Monte-Carlo Backup

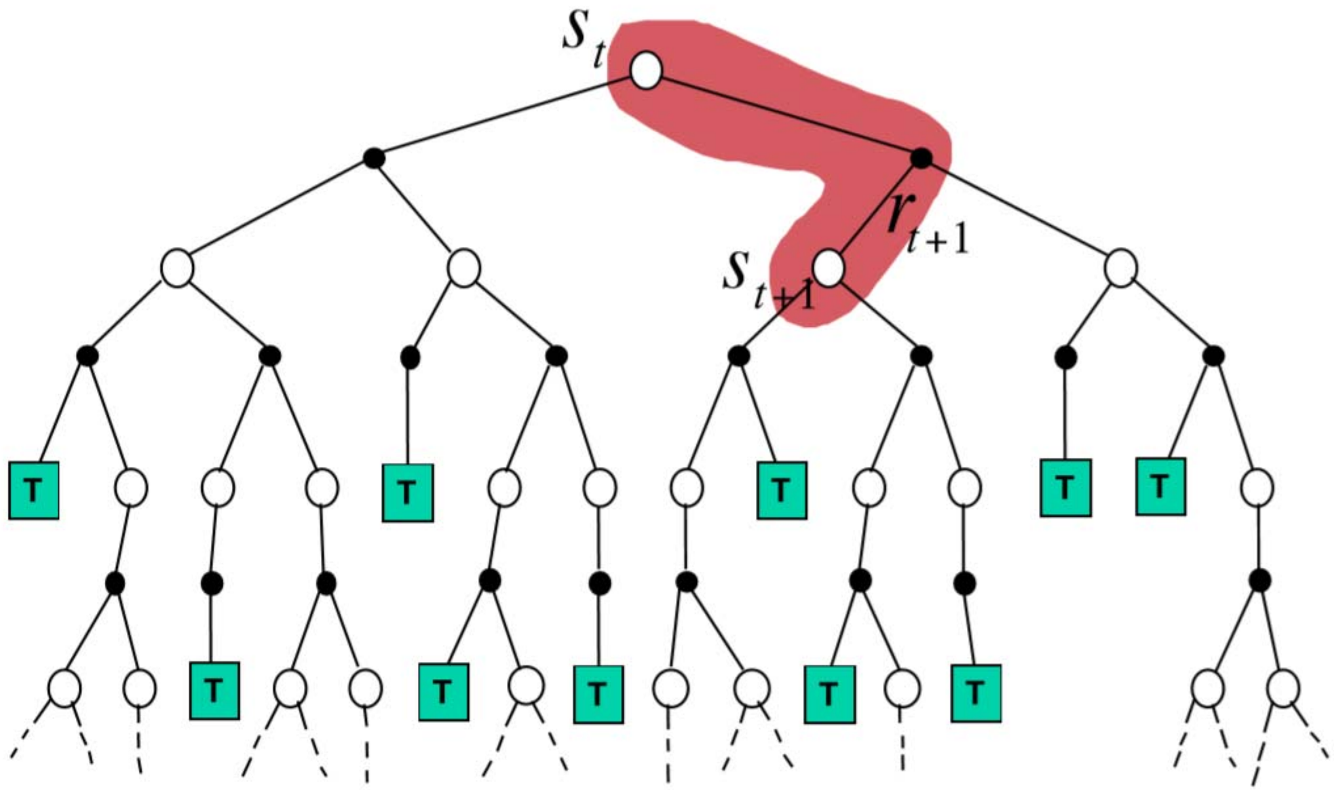
$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$$



David Silver, Reinforcement Learning, PPT

Temporal-Difference Backup

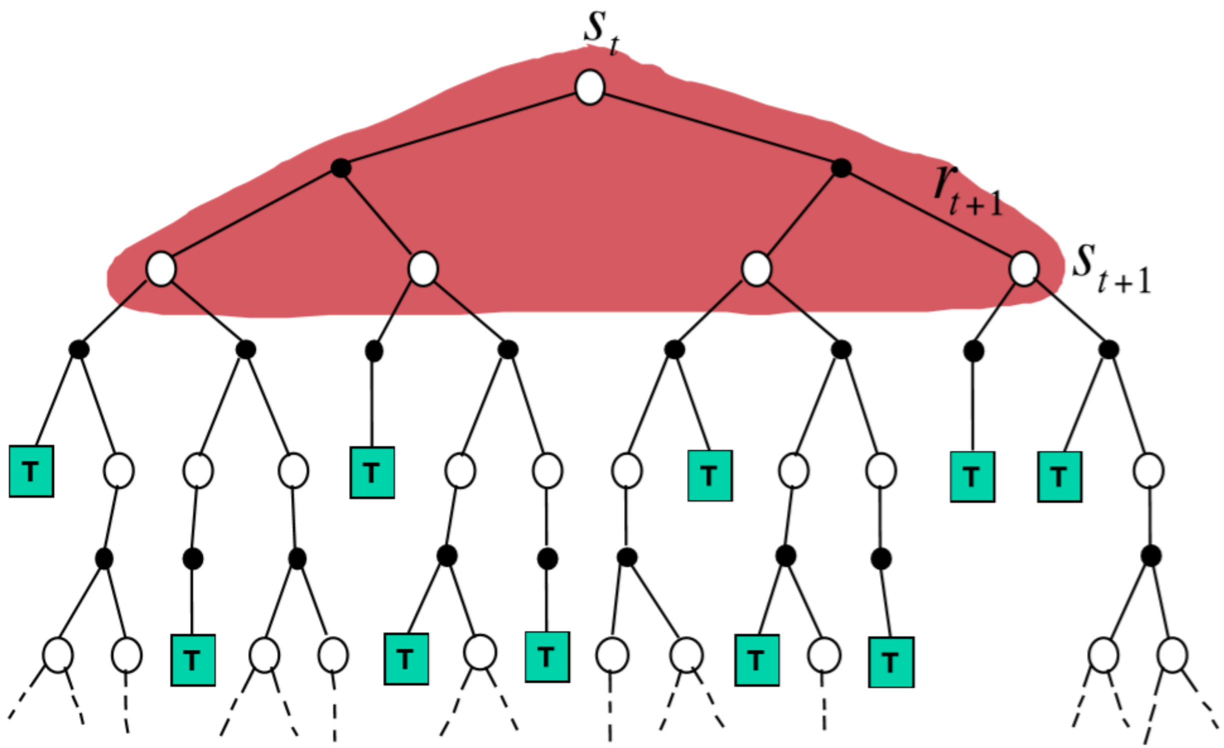
$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$



David Silver, Reinforcement Learning, PPT

Dynamic Programming Backup

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1}) \right]$$



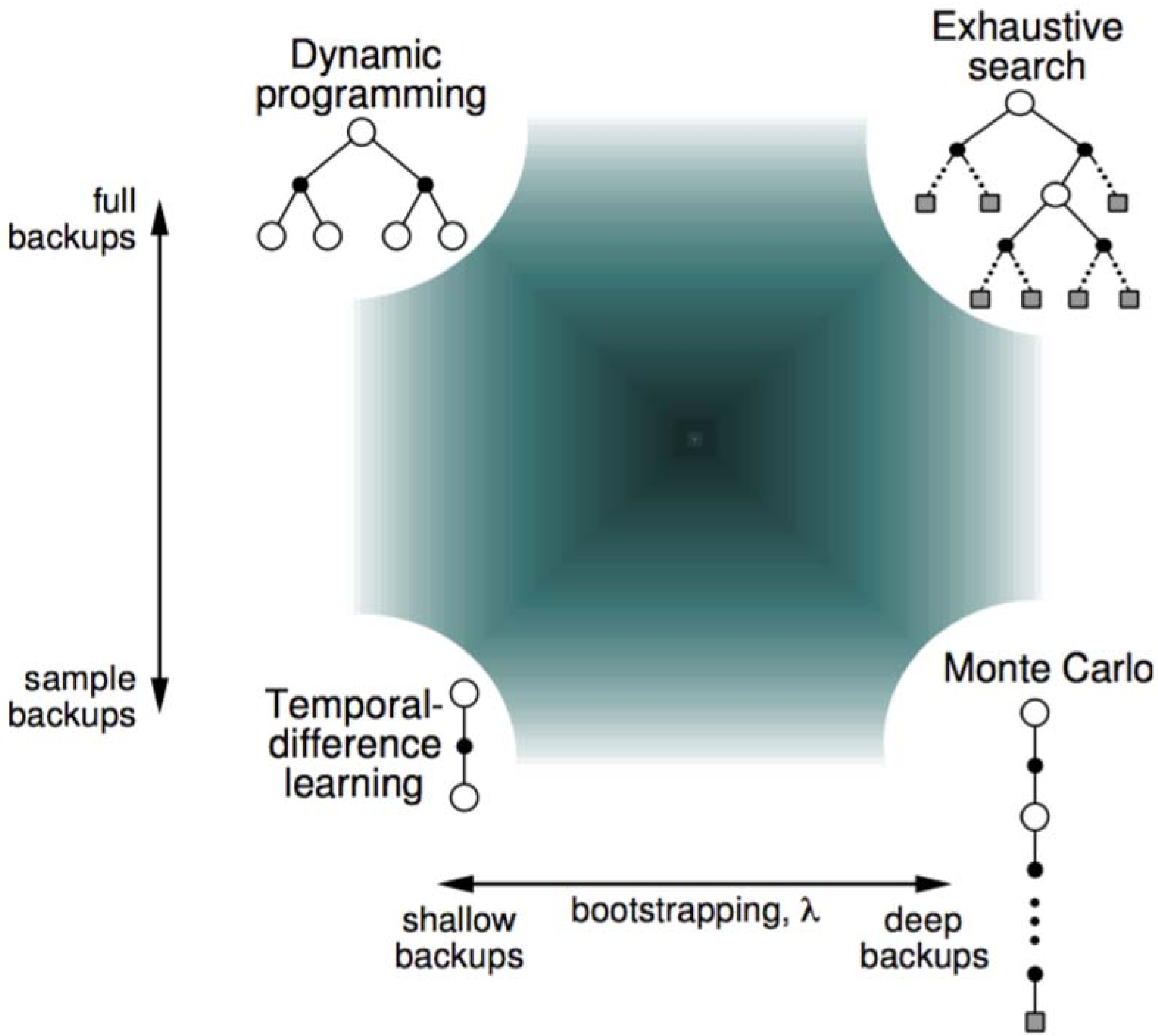
David Silver, Reinforcement Learning, PPT

Bootstrapping and Sampling

- Bootstrapping: update involves an estimate
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- Sampling: update samples an expectation
  - MC samples
  - DP does not sample
  - TD samples

David Silver, Reinforcement Learning, PPT

Unified View of Reinforcement Learning



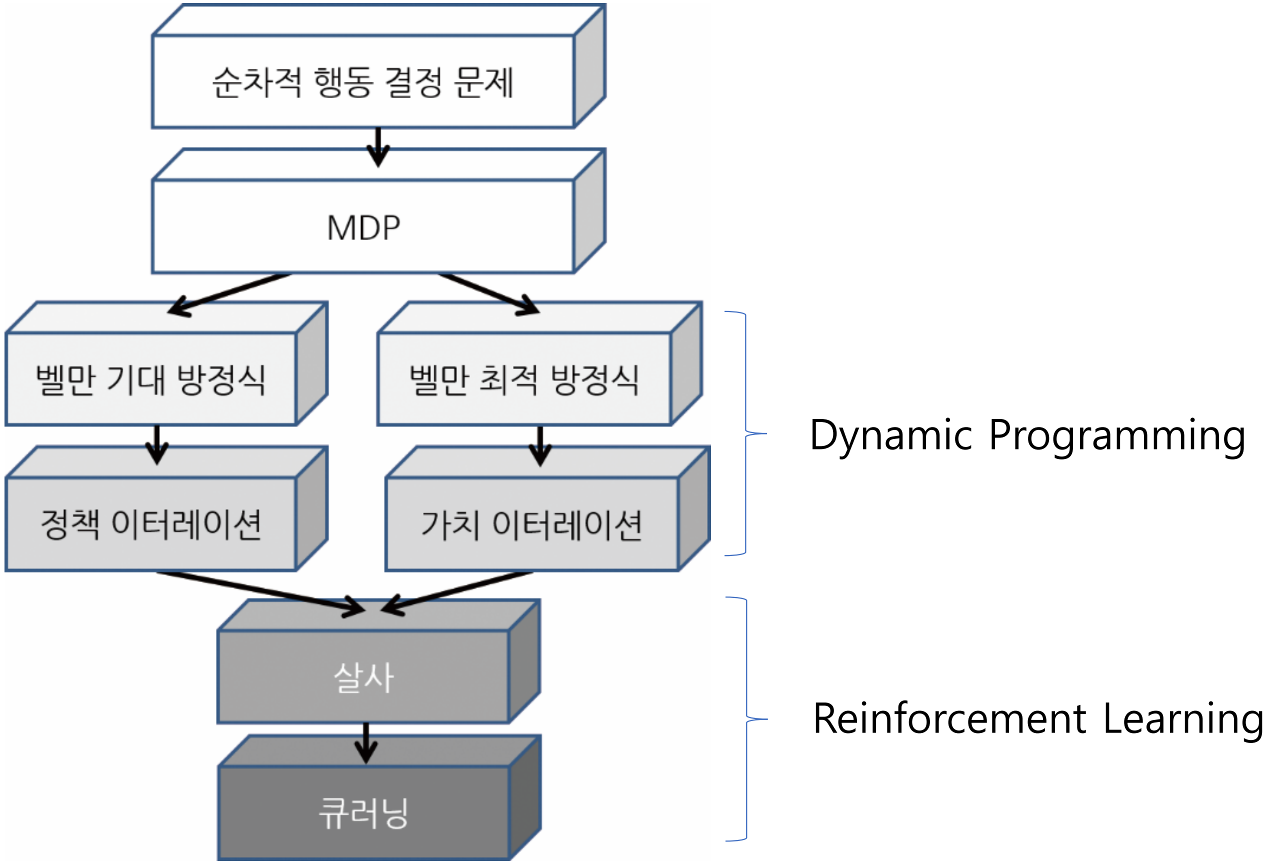
David Silver, Reinforcement Learning, PPT

Model-Free Reinforcement Learning

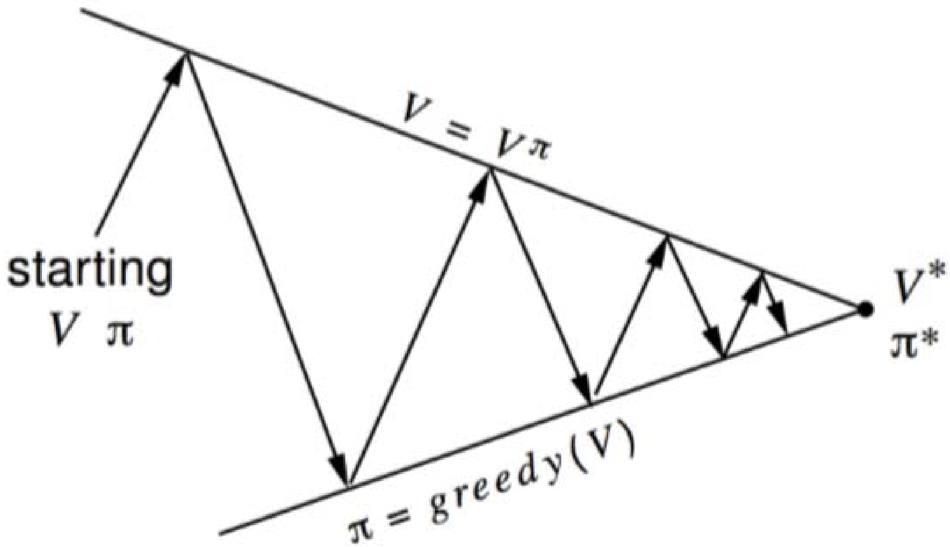
- Last lecture:
  - Model-free prediction
  - *Estimate* the value function of an *unknown* MDP
- This lecture:
  - Model-free control
  - *Optimise* the value function of an *unknown* MDP

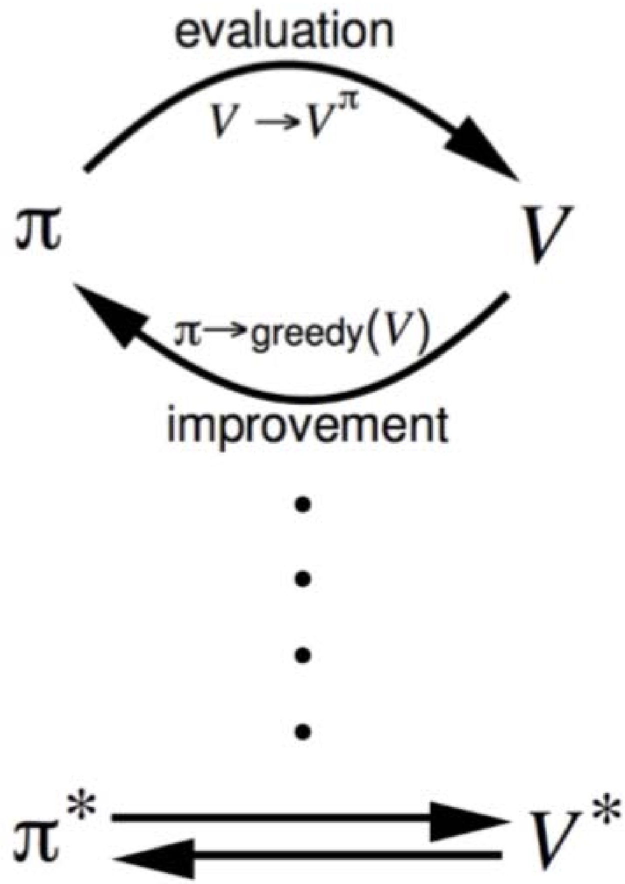
David Silver, Reinforcement Learning, PPT

강화학습 알고리즘의 흐름



Generalised Policy Iteration



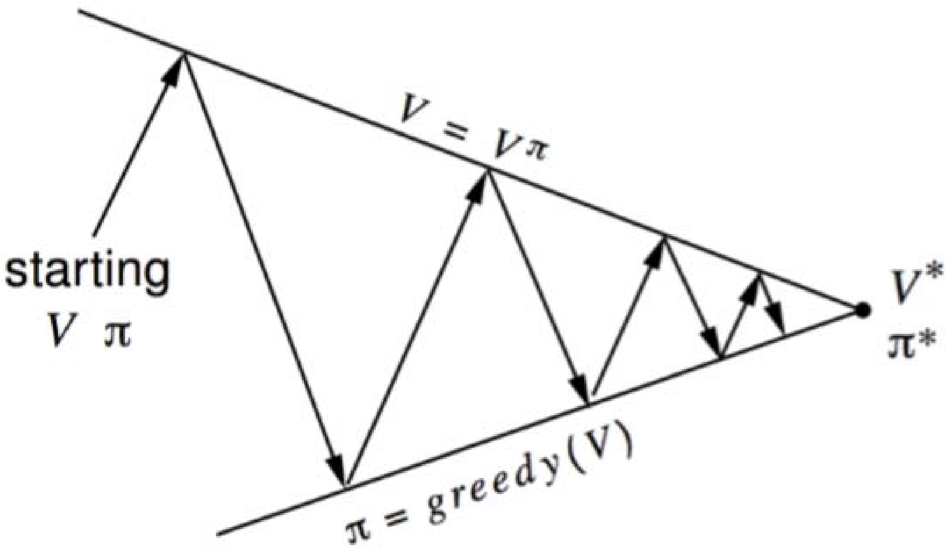


Policy evaluation Estimate v_π
e.g. Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
e.g. Greedy policy improvement

David Silver, Reinforcement Learning, PPT

Generalised Policy Iteration With Monte-Carlo Evaluation



Policy evaluation Monte-Carlo policy evaluation, $V = v_\pi$?

Policy improvement Greedy policy improvement?

David Silver, Reinforcement Learning, PPT

Model-Free Policy Iteration Using Action-Value Function

- Greedy policy improvement over $V(s)$ requires model of MDP

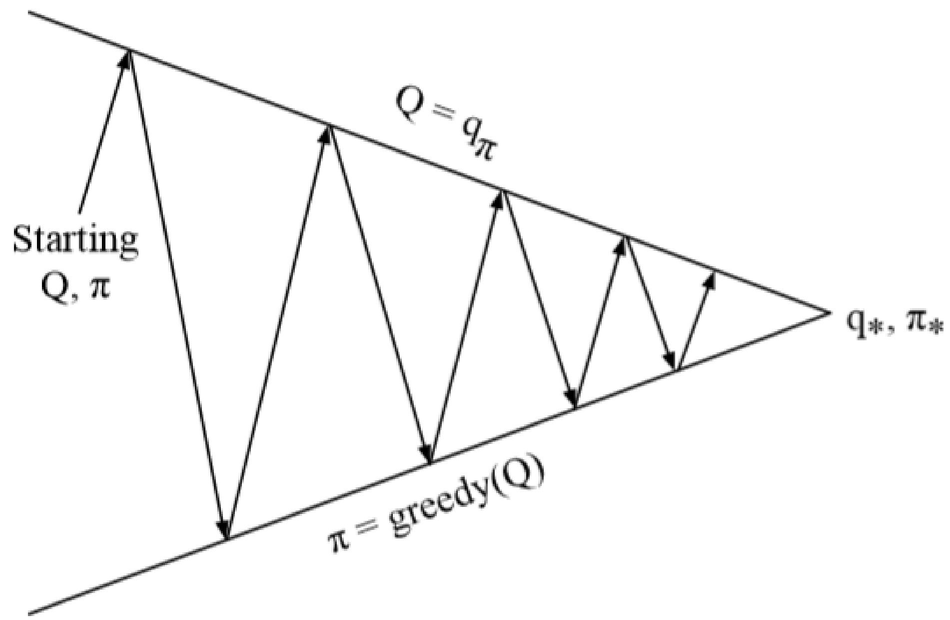
$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ \mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')$$

- Greedy policy improvement over $Q(s, a)$ is model-free

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

David Silver, Reinforcement Learning, PPT

Generalised Policy Iteration with Action-Value Function



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

Policy improvement Greedy policy improvement?

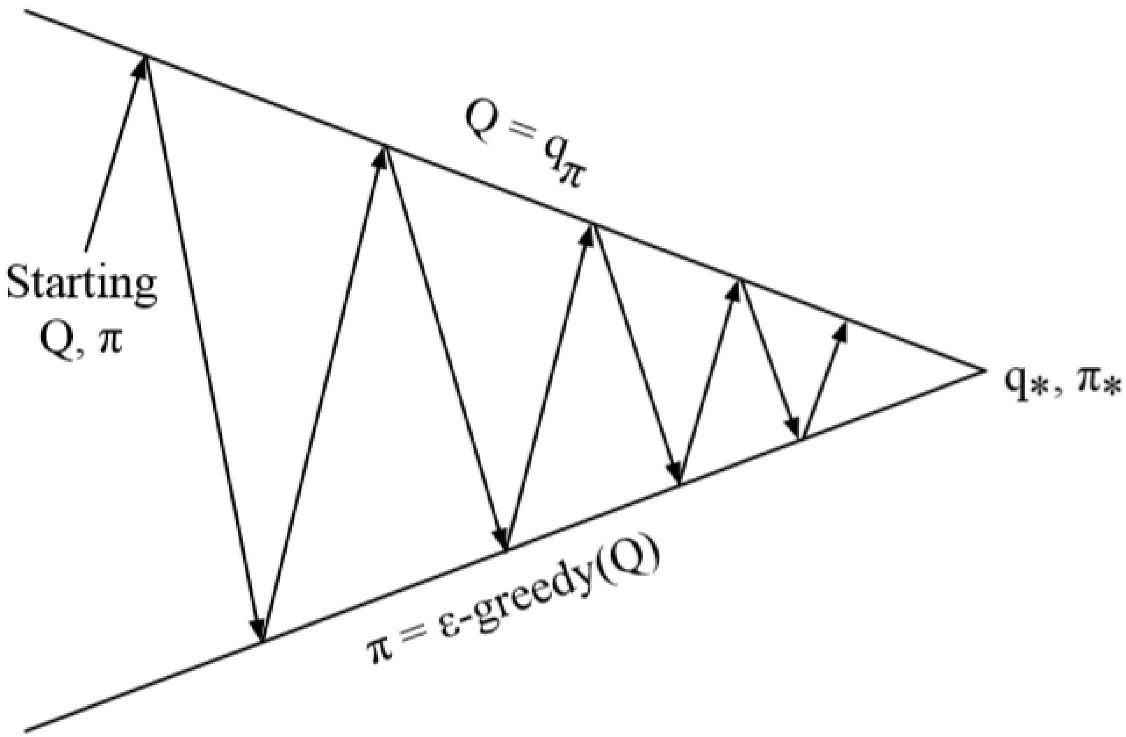
David Silver, Reinforcement Learning, PPT

ε-Greedy Exploration

- Simplest idea for ensuring continual exploration
- All m actions are tried with non-zero probability
- With probability $1 - \epsilon$ choose the greedy action
- With probability ϵ choose an action at random

David Silver, Reinforcement Learning, PPT

Monte-Carlo Policy Iteration

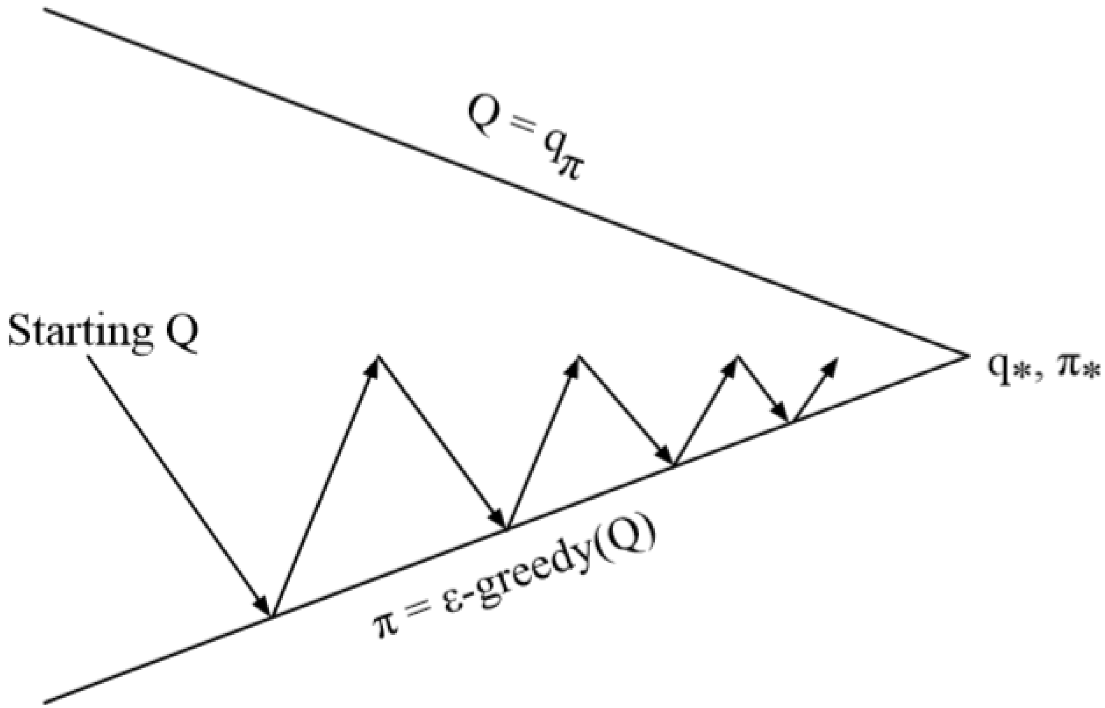


Policy evaluation Monte-Carlo policy evaluation, $Q = q_\pi$

Policy improvement ϵ-greedy policy improvement

David Silver, Reinforcement Learning, PPT

Monte-Carlo Control



Every episode:

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

Policy improvement ϵ-greedy policy improvement

David Silver, Reinforcement Learning, PPT

Monte-Carlo Control

- Sample kth episode using π: $\{S_1, A_1, R_2, ..., S_T\} \sim \pi$
- For each state S_t and action A_t in the episode,

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} \left(G_t - Q(S_t, A_t)\right)$$

- Improve policy based on new action-value function

$$\epsilon \leftarrow 1/k$$

$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

David Silver, Reinforcement Learning, PPT

TD Control

- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Incomplete sequences
- Natural idea: use TD instead of MC in our control loop
  - Apply TD to $Q(S, A)$
  - Use ϵ-greedy policy improvement
  - Update every time-step

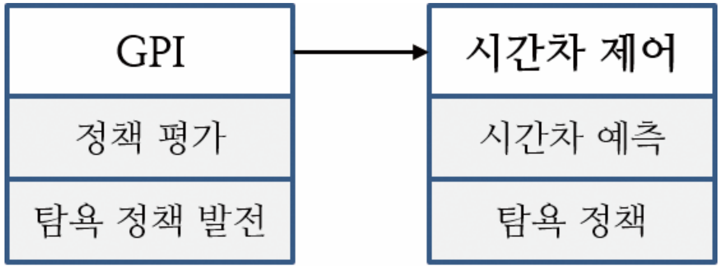
David Silver, Reinforcement Learning, PPT

GPI vs. TD

- GPI(Generalized Policy Iteration)
  - 정책 평가와 정책 발전을 한번씩 번갈아 가면서 진행

- 시간차 예측
  - 타임스텝마다 가치함수를 현재 상태에 대해서만 업데이트

- 시간차 제어



시간차 제어에서의 탐욕 정책

- GPI에서의 정책 발전 식

$$\pi'(s) = argmax_{a \in A}[R_s^a + \gamma P_{ss'}^a V(s')]$$

-> 환경의 모델인 R_s^a와 $P_{ss'}^a$ 를 알아야 함

- 큐함수를 사용한 탐욕 정책 식

$$\pi(s) = argmax_{a \in A} Q(s, a)$$

-> 다음 상태의 가치함수를 보고 판단하는 것이 아니라
현재 상태의 큐함수를 보고 판단

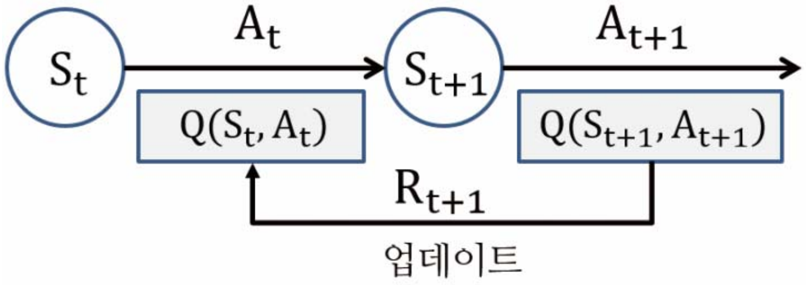
-> 환경의 모델을 몰라도 됨

강화학습 알고리즘 1: SARSA

- 시간차 제어에서는 큐함수를 업데이트 함

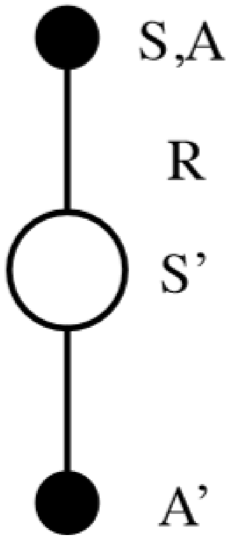
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

➢다음 상태의 큐함수인 $Q(S_{t+1}, A_{t+1})$를 알기 위해서는 다음 상태 S_{t+1} 에서 다음 행동 A_{t+1}까지 선택해야 함



➢시간차 제어에서 큐함수를 업데이트하기 위해 [S_t,A_t,R_{t+1},S_{t+1},A_{t+1}]를 샘플로 사용 -> SARSA

Updating Action-Value Functions with Sarsa



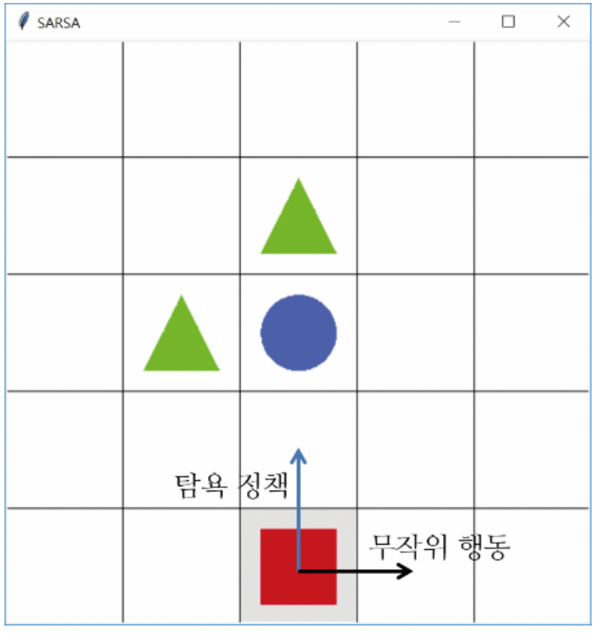
$$Q(S, A) \leftarrow Q(S, A) + \alpha \left(R + \gamma Q(S', A') - Q(S, A) \right)$$

David Silver, Reinforcement Learning, PPT

강화학습 알고리즘 1: SARSA

- Exploitation vs Exploration
  - ➢ ε-greedy 정책

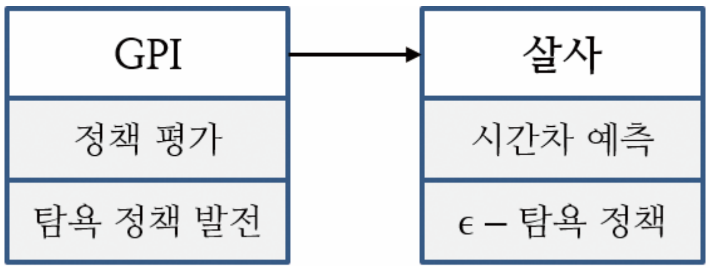
$$\pi(s) = \begin{cases} a^* = argmax_{a \in A} Q(s,a) & 1-\varepsilon \\ a \neq a^* & \varepsilon \end{cases}$$

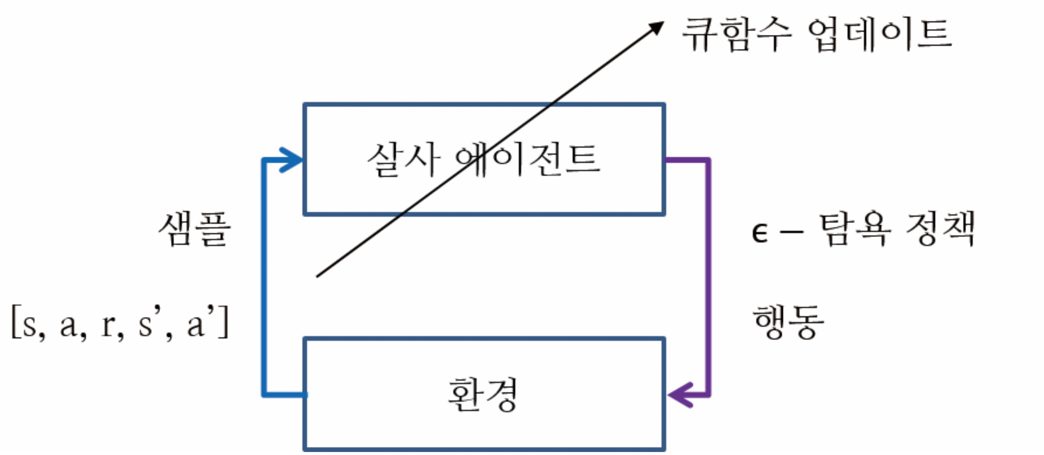


강화학습 알고리즘 1: SARSA

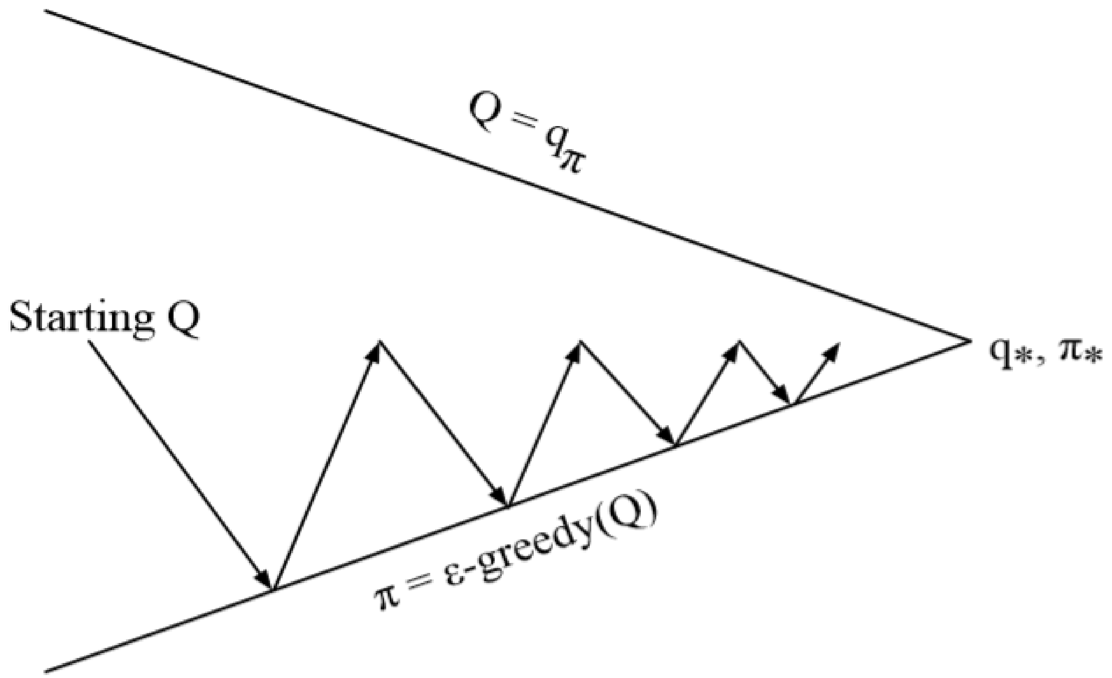
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

1. ε-greedy 정책을 통해 정책 $[S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}]$을 획득
2. 획득한 샘플로 다음 식을 통해 큐함수 $Q(S_t, A_t)$를 업데이트





On-Policy Control With Sarsa



Every time-step:

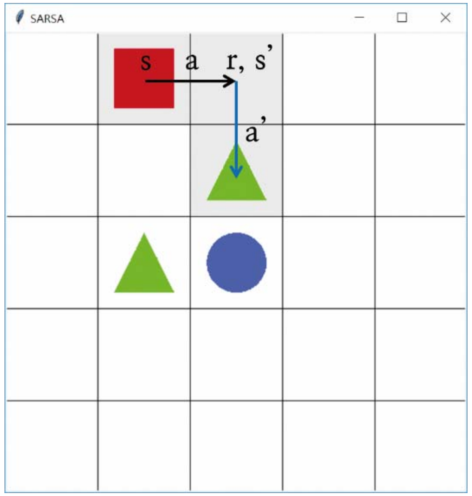
Policy evaluation Sarsa, $Q \approx q_\pi$

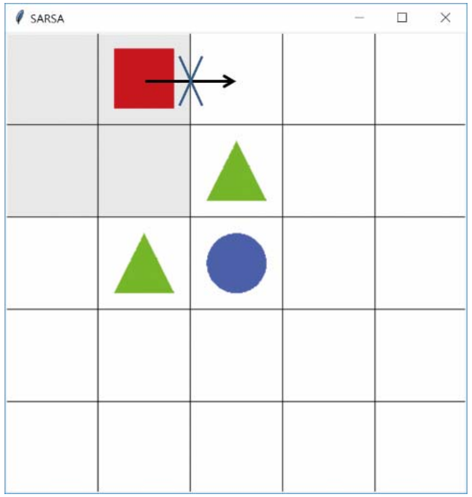
Policy improvement ϵ-greedy policy improvement

David Silver, Reinforcement Learning, PPT

강화학습 알고리즘 2: Q Learning

- SARSA의 한계





- 에이전트가 s′에서 탐험에 따라 아래로 가는 a′을 함 -> Q(s′,a′)는 -1의 보상을 받음
- 따라서 아래의 큐함수 업데이트 식에 따라 Q(s,a)의 값도 낮아지게 됨

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

- 이후에 에이전트가 다시 s에 오게 되면 오른쪽으로 가는 행동 a′이 안 좋다고 판단함
- 따라서 에이전트가 s에서 오른쪽으로 가지 못하고 갇혀버림

강화학습 알고리즘 2: Q Learning

- SARSA
  - On-Policy : 자신이 행동하는 대로 학습
  - 탐험을 위해 선택한 ε-greedy 정책 때문에 에이전트는 최적 정책을 학습하지 못하고 오히려 잘못된 정책을 학습

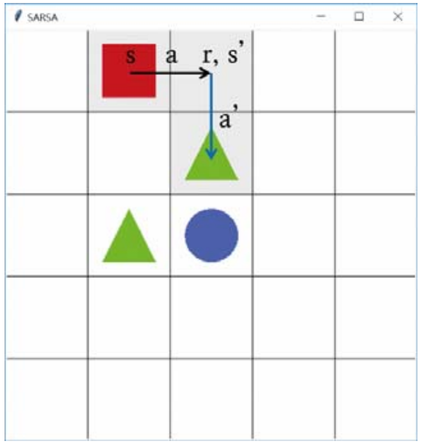
- Q-Learning
  - Off-Policy : 행동하는 정책과 학습하는 정책을 분리
  - 행동은 ε-greedy 정책에 따라 선택
  - 큐함수 업데이트는 greedy 정책(가장 큰 큐함수로 업데이트)으로 함

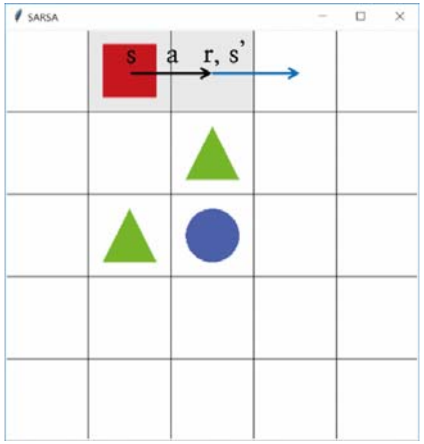
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t) \right)$$

  - 큐함수를 업데이트하기 위해 필요한 샘플은 [s,a,r,s']

강화학습 알고리즘 2: Q Learning

- Q-Learning





- 에이전트가 s′에서 탐험에 따라 아래로 가는 a′을 함
  - ➢ 하지만 큐함수 업데이트에는 a′을 반영하지 않고, 그 상태(s′)에서 가장 큰 큐함수로 현재 큐함수를 업데이트

- 이후에 에이전트가 다시 s에 오게 되면 아래의 큐함수 업데이트 식에 따라 오른쪽으로 진행할 수 있음

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\right)$$

Q-Learning

- We now consider off-policy learning of action-values $Q(s, a)$
- No importance sampling is required
- Next action is chosen using behaviour policy $A_{t+1} \sim \mu(\cdot|S_t)$
- But we consider alternative successor action $A' \sim \pi(\cdot|S_t)$
- And update $Q(S_t, A_t)$ towards value of alternative action

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

David Silver, Reinforcement Learning, PPT

Off-Policy Control with Q-Learning

- We now allow both behaviour and target policies to improve
- The target policy π is greedy w.r.t. $Q(s,a)$

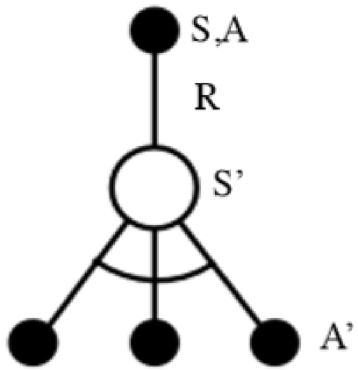
$$\pi(S_{t+1}) = \underset{a'}{\text{argmax}}\ Q(S_{t+1}, a')$$

- The behaviour policy μ is e.g. ϵ-greedy w.r.t. $Q(s,a)$
- The Q-learning target then simplifies:

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

David Silver, Reinforcement Learning, PPT

Q-Learning Control Algorithm



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

David Silver, Reinforcement Learning, PPT

강화학습 알고리즘 2: Q Learning

- SARSA vs Q-Learning

$$q_\pi(s,a) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma q_\pi(S_{t+1},A_{t+1}) | S_t = s, A_t = a]$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

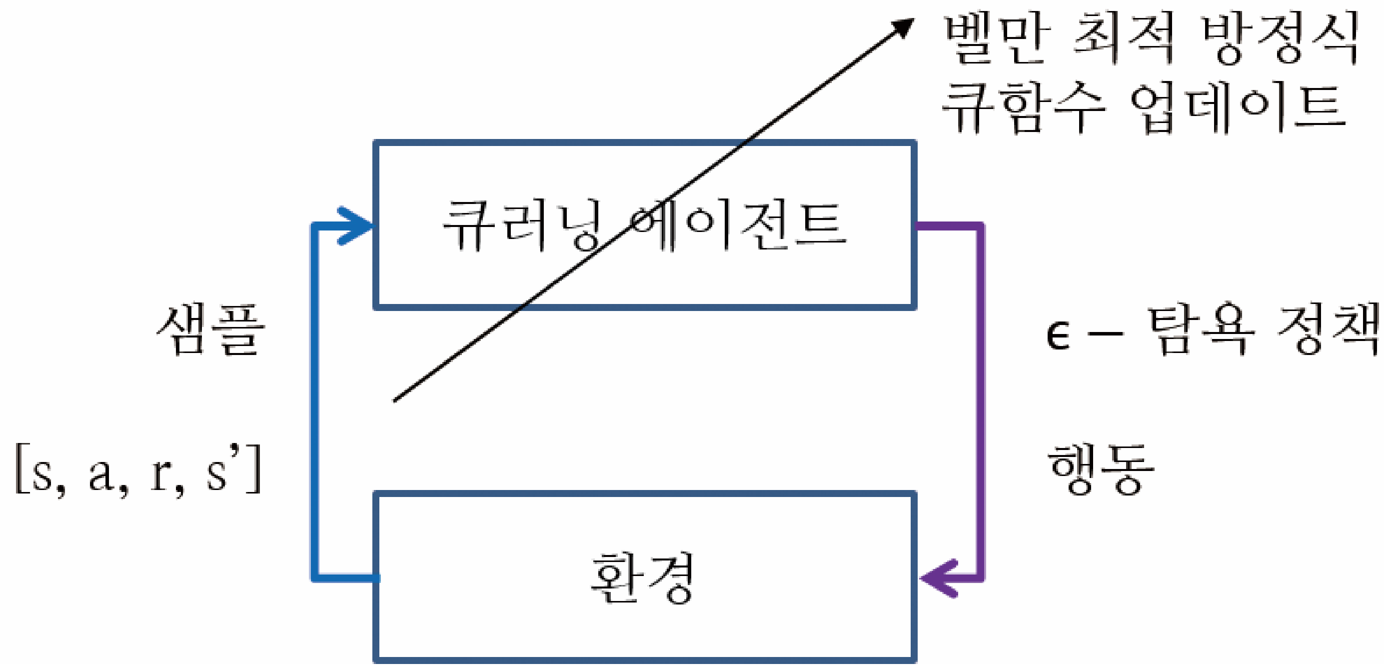
벨만 기대 방정식 → 정책 이터레이션 → 살사

벨만 최적 방정식 → 가치 이터레이션 → 큐러닝

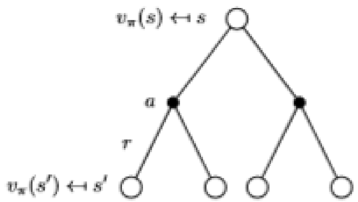
$$q^*(s,a) = \boldsymbol{E}[R_{t+1} + \gamma \max_{a'} q^*(S_{t+1},a') \,| S_t = s, A_t = a]$$

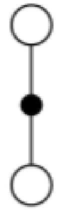
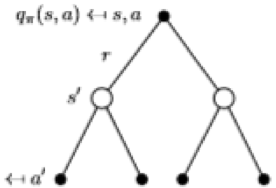
$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha\left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1},a') - Q(S_t,A_t)\right)$$

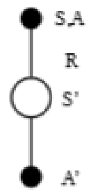
강화학습 알고리즘 2: Q Learning

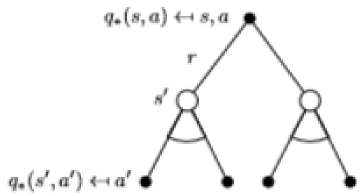


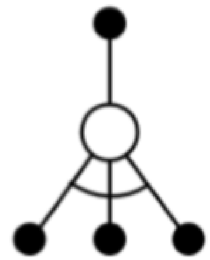
Relationship Between DP and TD

<table>
<tr><th></th><th>Full Backup (DP)</th><th>Sample Backup (TD)</th></tr>
<tr><td>Bellman Expectation Equation for $v_\pi(s)$</td><td>

Iterative Policy Evaluation</td><td>
TD Learning</td></tr>
<tr><td>Bellman Expectation Equation for $q_\pi(s, a)$</td><td>

Q-Policy Iteration</td><td>

Sarsa</td></tr>
<tr><td>Bellman Optimality Equation for $q_*(s, a)$</td><td>

Q-Value Iteration</td><td>
Q-Learning</td></tr>
</table>

David Silver, Reinforcement Learning, PPT

Relationship Between DP and TD (2)

Full Backup (DP)	*Sample Backup (TD)*
Iterative Policy Evaluation	TD Learning
$V(s) \leftarrow \mathbb{E}\left[R + \gamma V(S') \mid s\right]$	$V(S) \xleftarrow{\alpha} R + \gamma V(S')$
Q-Policy Iteration	Sarsa
$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma Q(S',A') \mid s,a\right]$	$Q(S,A) \xleftarrow{\alpha} R + \gamma Q(S',A')$
Q-Value Iteration	Q-Learning
$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max\limits_{a' \in \mathcal{A}} Q(S',a') \mid s,a\right]$	$Q(S,A) \xleftarrow{\alpha} R + \gamma \max\limits_{a' \in \mathcal{A}} Q(S',a')$

where $x \xleftarrow{\alpha} y \equiv x \leftarrow x + \alpha(y - x)$

David Silver, Reinforcement Learning, PPT

Summary

- 몬테카를로 예측

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t)\ )$$

- 시간차 예측

$$V(S_t) \leftarrow V(S_t) + \alpha(R + \gamma V(S_{t+1}) - V(S_t))$$

- MC Control

$$Q(s,a) \leftarrow Q(s,a) + \alpha(G_t - Q(s,a))$$

- SARSA

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

- Q-Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\right)$$