Lecture 5: Model-Free Control

최적의 Policy를 찾아냄.

David Silver

Control : Prediction + Improvement.

Outline

on policy : 학습 Policy = 행동 Policy

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

Uses of Model-Free Control

아직은 Table lookup.
State가 많을 땐 → Function Approximator 사용해야한다.

Some example problems that can be modelled as MDPs

- Elevator
- Parallel Parking
- Ship Steering
- Bioreactor
- Helicopter
- Aeroplane Logistics
- Robocup Soccer
- Quake
- Portfolio management
- Protein Folding
- Robot walking
- Game of Go

For most of these problems, either:

- MDP model is unknown, but experience can be sampled
- MDP model is known, but is too big to use, except by samples

Model-free control can solve these problems

On and Off-Policy Learning

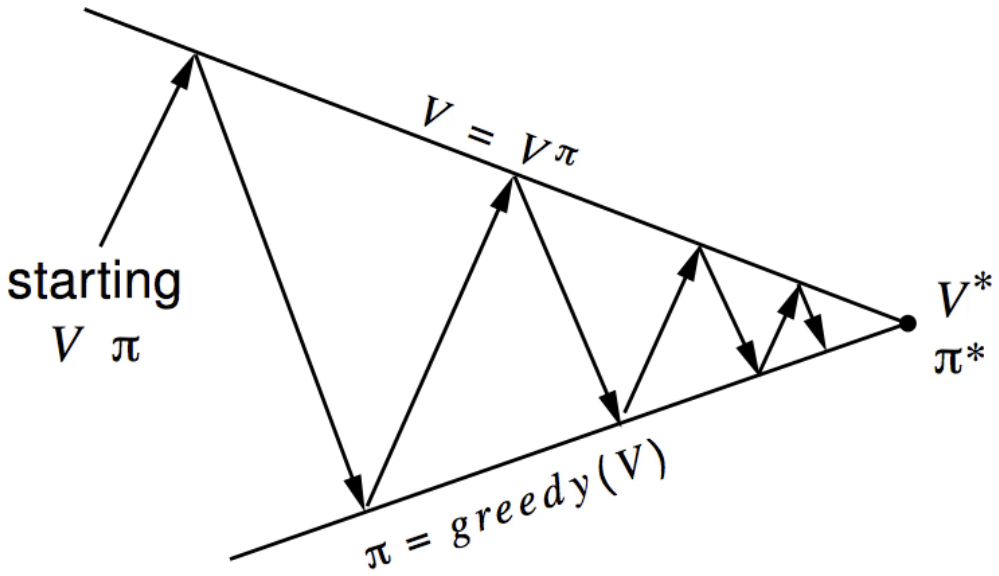
- On-policy learning
  - "Learn on the job"
  - Learn about policy π from experience sampled from π
- Off-policy learning
  - "Look over someone's shoulder"
  - Learn about policy π from experience sampled from μ

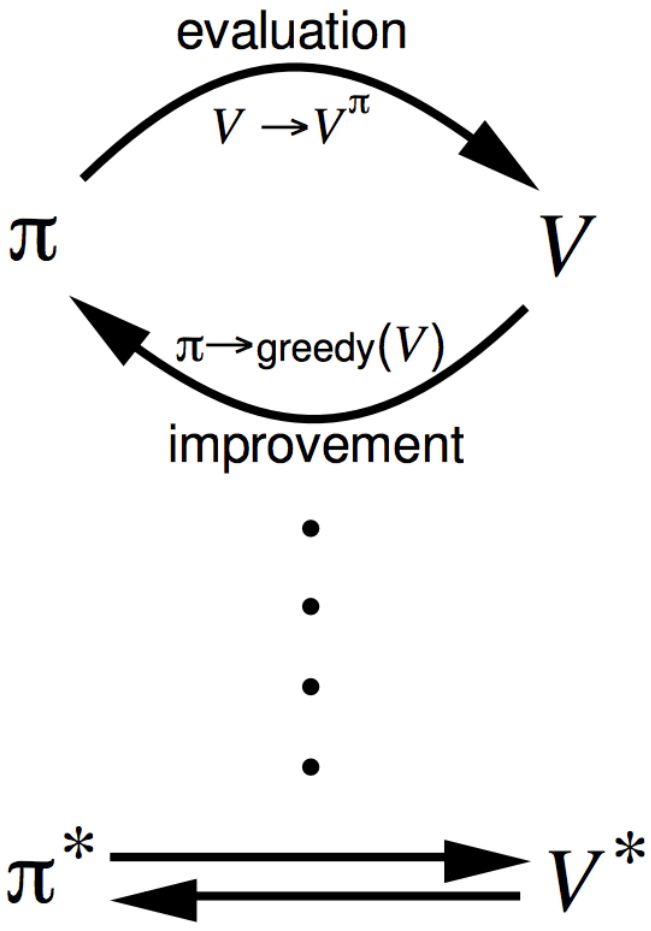
학습 Policy = 행동 Policy

학습 Policy ≠ 행동 Policy

Generalised Policy Iteration (Refresher)

Evaluation과 greedy Improvement를 한번씩 번갈아가면서 한다.





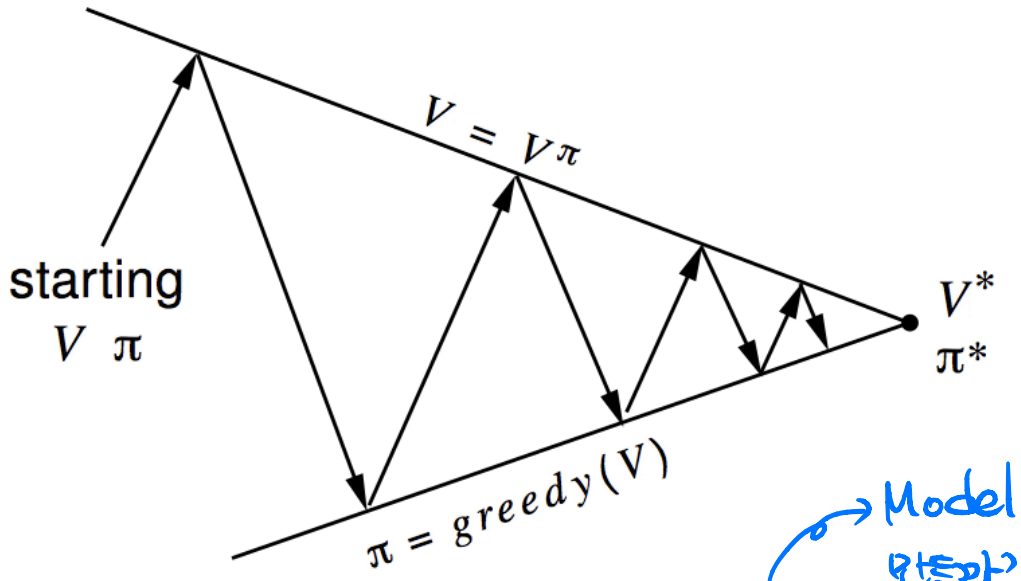
Policy evaluation Estimate v_π
e.g. Iterative policy evaluation

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

Generalised Policy Iteration With Monte-Carlo Evaluation

↳한계점

1. episode가 다끝나야만함
2. 충분한 경험이 있으면 variance가 큼.



↳ Model-free인데 어떻게 greedy policy를 만들까?

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

Policy improvement Greedy policy improvement?

Model-Free Policy Iteration Using Action-Value Function

Greedy Policy Improvement는 MDP를 알아야만 한다.

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

→ state마다 값이 있는가.

↓ 불가능

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

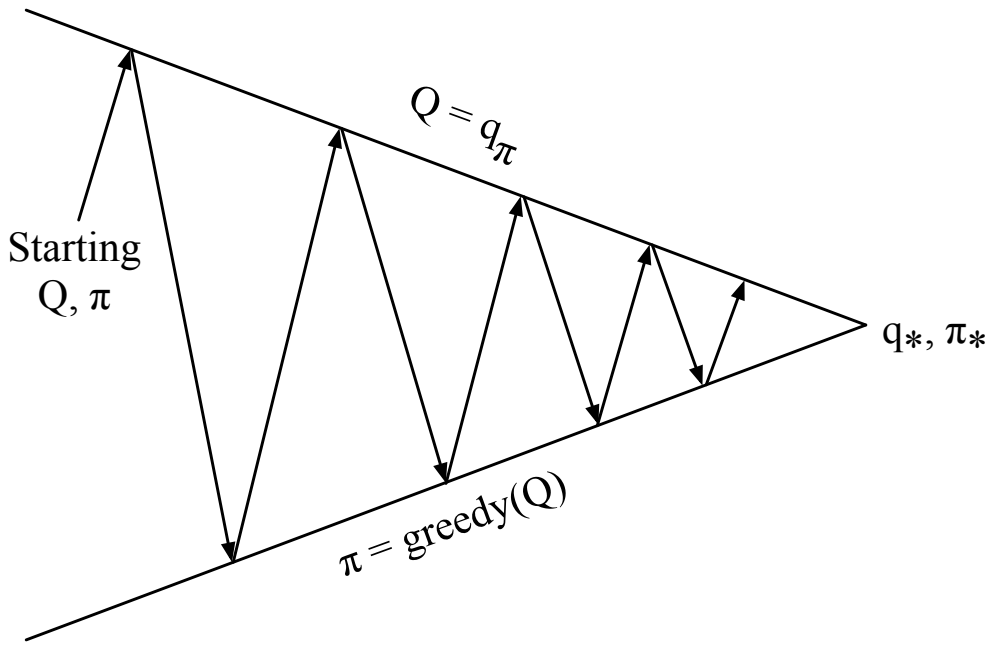
- Greedy policy improvement over $Q(s, a)$ is model-free

↓ 가능함.

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

↳ state × action 만큼 있다.

Generalised Policy Iteration with Action-Value Function



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

Policy improvement Greedy policy improvement?

↳ exploration 문제.

Example of Greedy Action Selection

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

- 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 $+3$
  $V(right) = +2$
- You open the right door and get reward $+2$
  $V(right) = +2$

$\vdots$

- Are you sure you've chosen the best door?

ϵ-Greedy Exploration

ϵ 만큼의 확률로 random action을 선택.
나머지 $1-\epsilon$로 greedy하게 action 선택.

- 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

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

1. 모든 액션이 explore 함을 보장할 수 있다.
2. Policy가 계속 Improve함을 보장할 수 있다.

ϵ-Greedy Policy Improvement

식이해안감..

Theorem

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

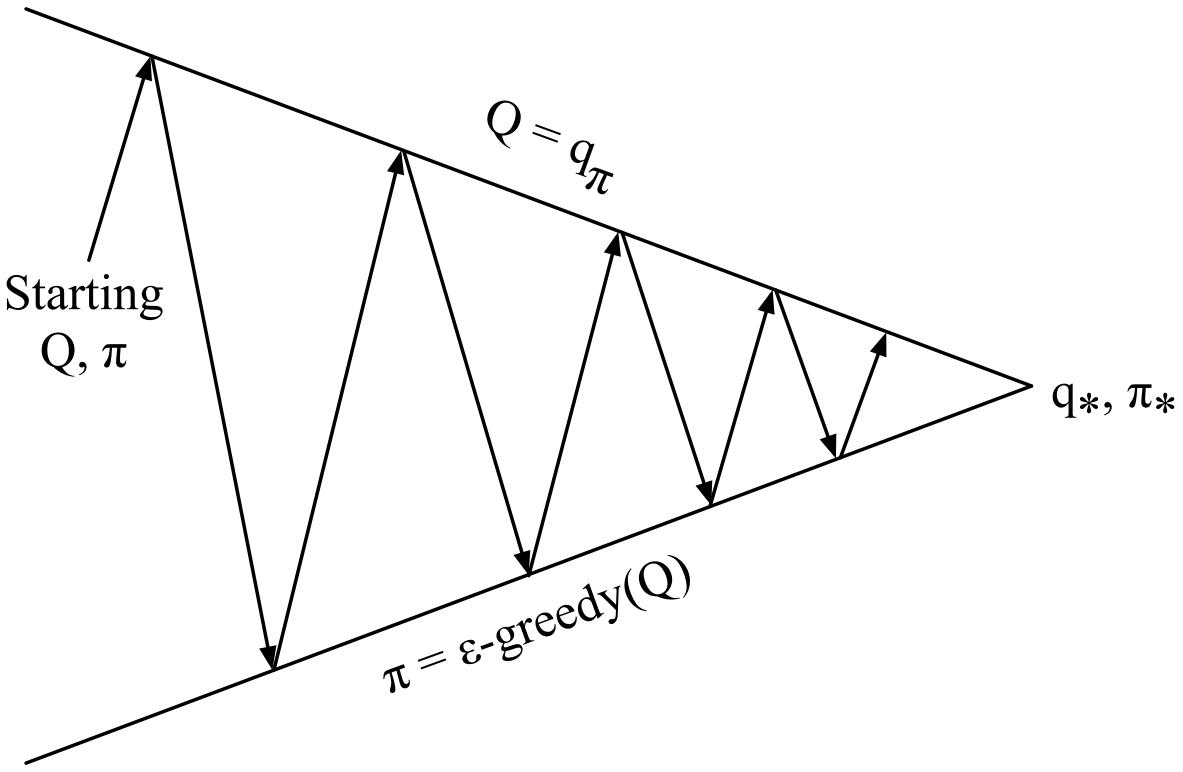
m: action 수

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

현재 Policy 따라서 State s일때, action은 $\pi(s)$에서 선택한다.

Therefore from policy improvement theorem, $v_{\pi'}(s) \geq v_\pi(s)$

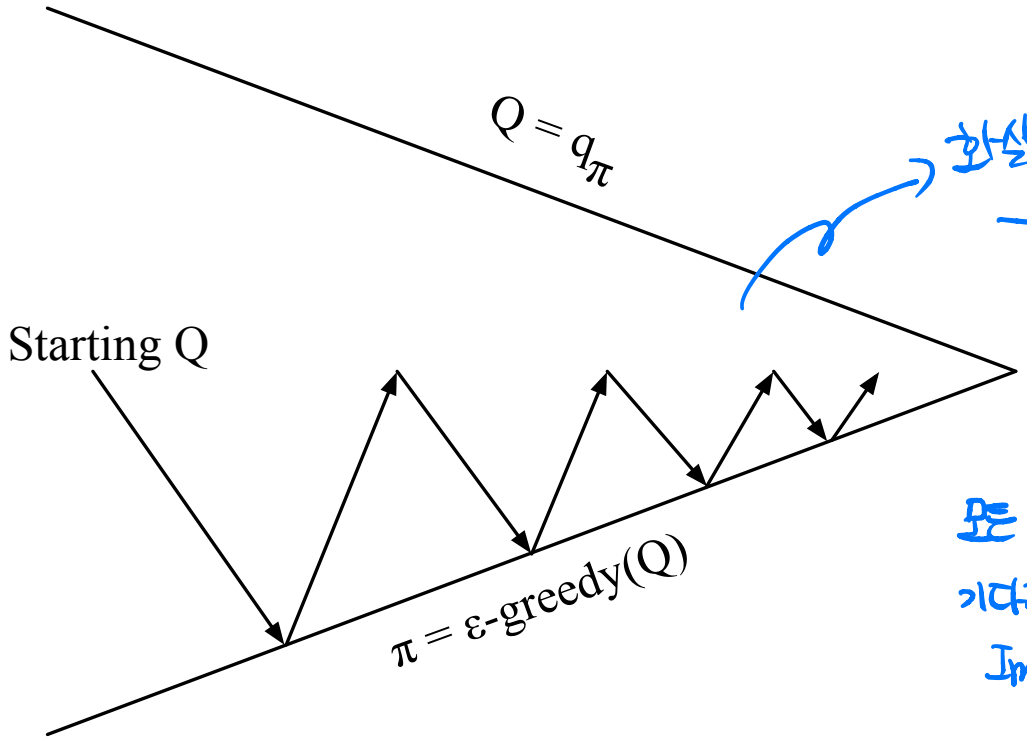
Monte-Carlo Policy Iteration



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

Policy improvement ϵ-greedy policy improvement

Monte-Carlo Control



실선까지 안가는 이유
→ MC는 episode 단위로 update 한다.

또 episode가 다끝날 때까지 기다리는 것이 아니라 한 epi 끝나면 바로 Improvement 한다.

Every episode:

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

Policy improvement ϵ-greedy policy improvement

GLIE

Definition

Greedy in the Limit with Infinite Exploration (GLIE)

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

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

- The policy converges on a greedy policy, exploitation

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

- For example, ϵ-greedy is GLIE if ϵ reduces to zero at $\epsilon_k = \frac{1}{k}$

GLIE 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,

evaluate

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

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

- Improve policy based on new action-value function

improve

$$\epsilon \leftarrow 1/k$$

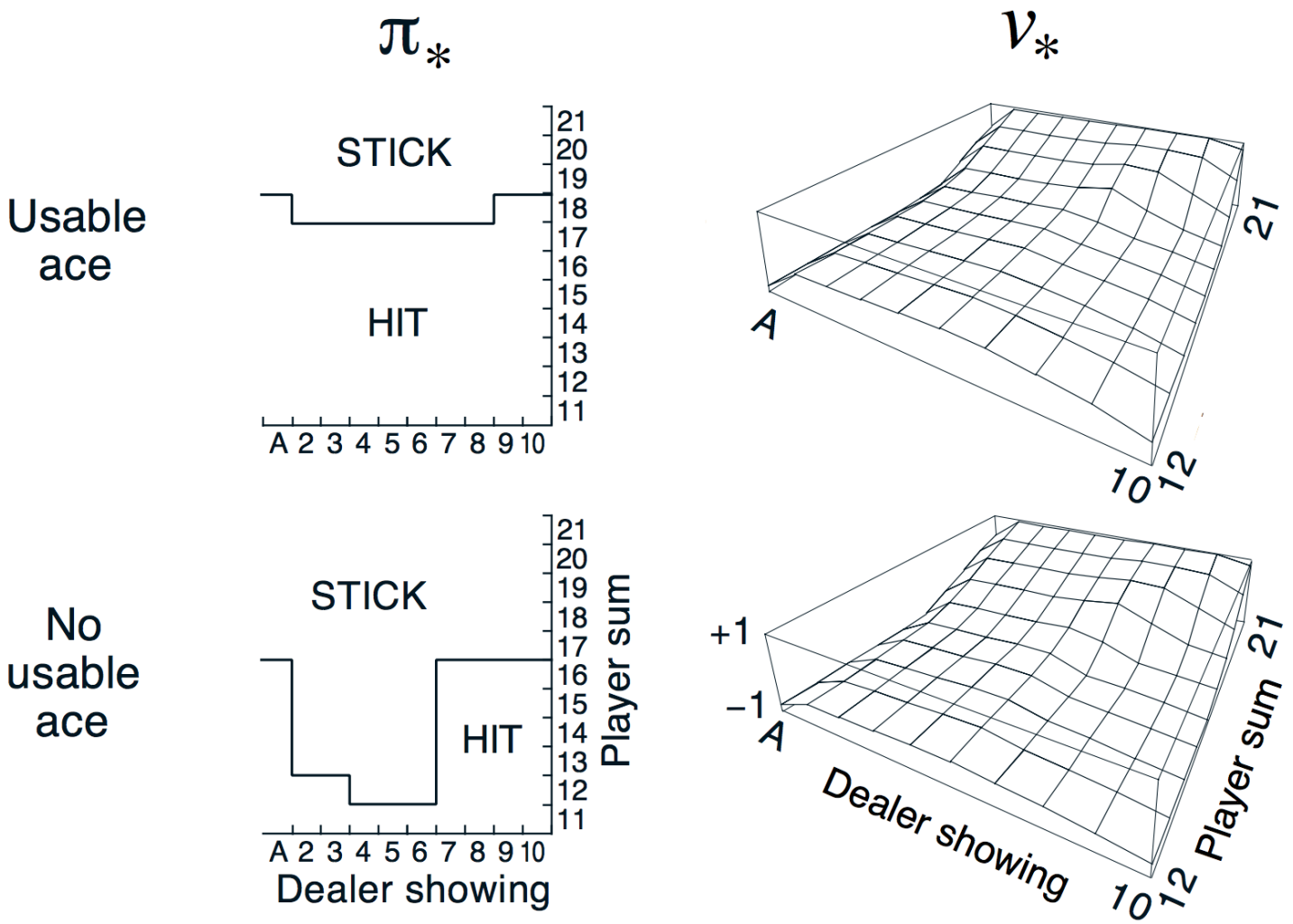
$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

Theorem

GLIE Monte-Carlo control converges to the optimal action-value function, $Q(s, a) \rightarrow q_*(s, a)$

Back to the Blackjack Example

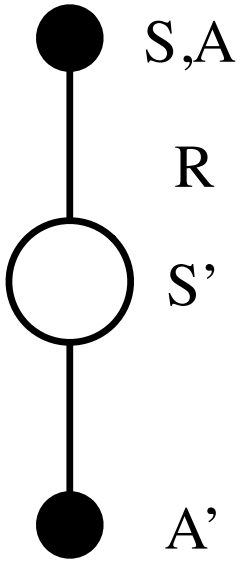
Monte-Carlo Control in Blackjack



MC vs. 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

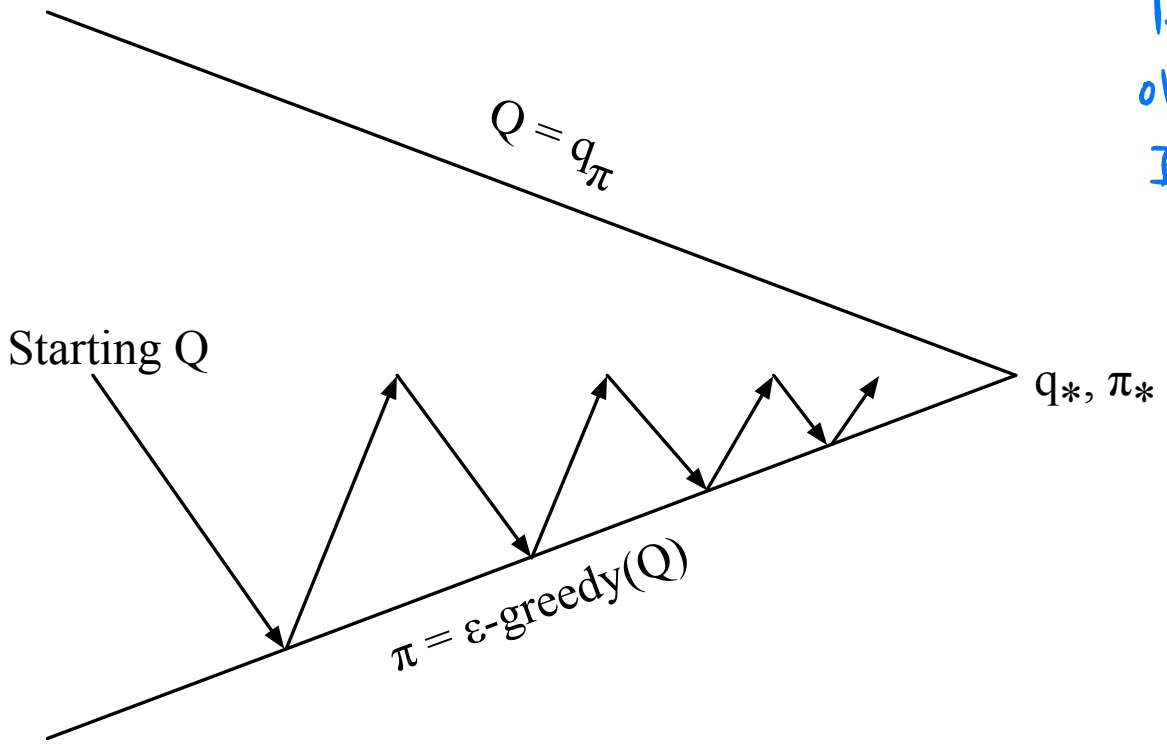
Updating Action-Value Functions with Sarsa



State S에서 action A를 하면
Reward R을 받고 next state S'
로 가서 A' action을 한다.

$$Q(S,A) \leftarrow Q(S,A) + \alpha \underbrace{\left(\overbrace{R + \gamma Q(S',A')}^{\text{TD target}} - Q(S,A)\right)}_{\text{TD error}}$$

On-Policy Control With Sarsa



1스텝 가서 Q update하고
이걸 바탕으로 ε-greedy policy
Improve한다.

Every time-step:

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

Policy improvement ϵ-greedy policy improvement

Sarsa Algorithm for On-Policy Control

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Convergence of Sarsa

Theorem

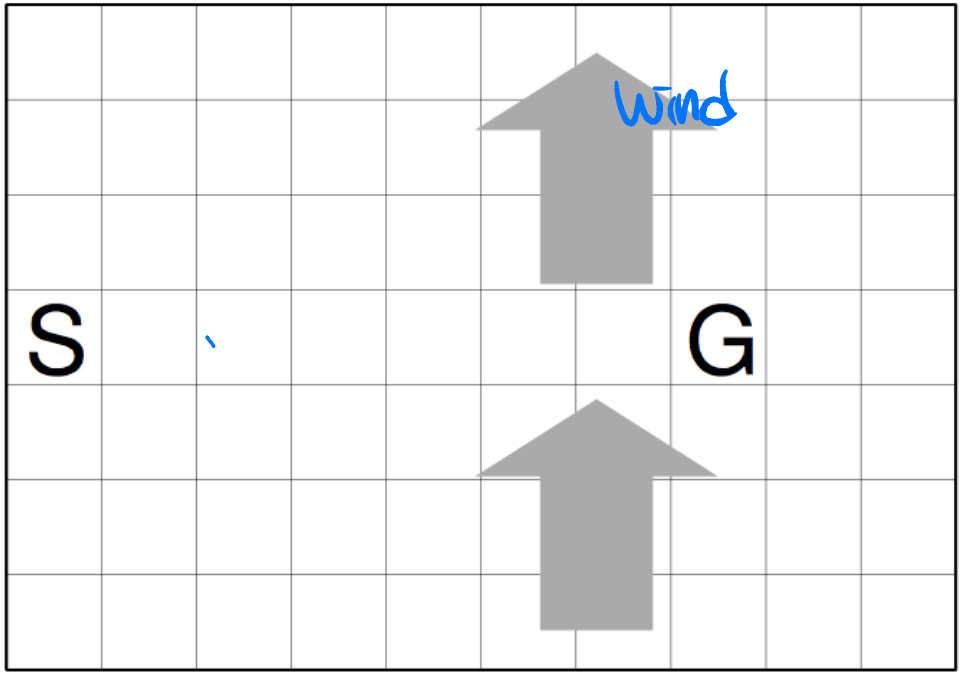
Sarsa converges to the optimal action-value function, $Q(s,a) \rightarrow q_*(s,a)$*, under the following conditions:*

- *GLIE sequence of policies* $\pi_t(a|s)$
- *Robbins-Monro sequence of step-sizes* α_t → 실제로는 크게 신경안써도 잘된다.

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

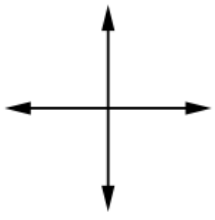
$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

Windy Gridworld Example

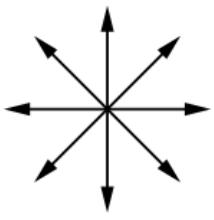


→ 바람이 불리는 칸수.

움직이는 방법



standard moves

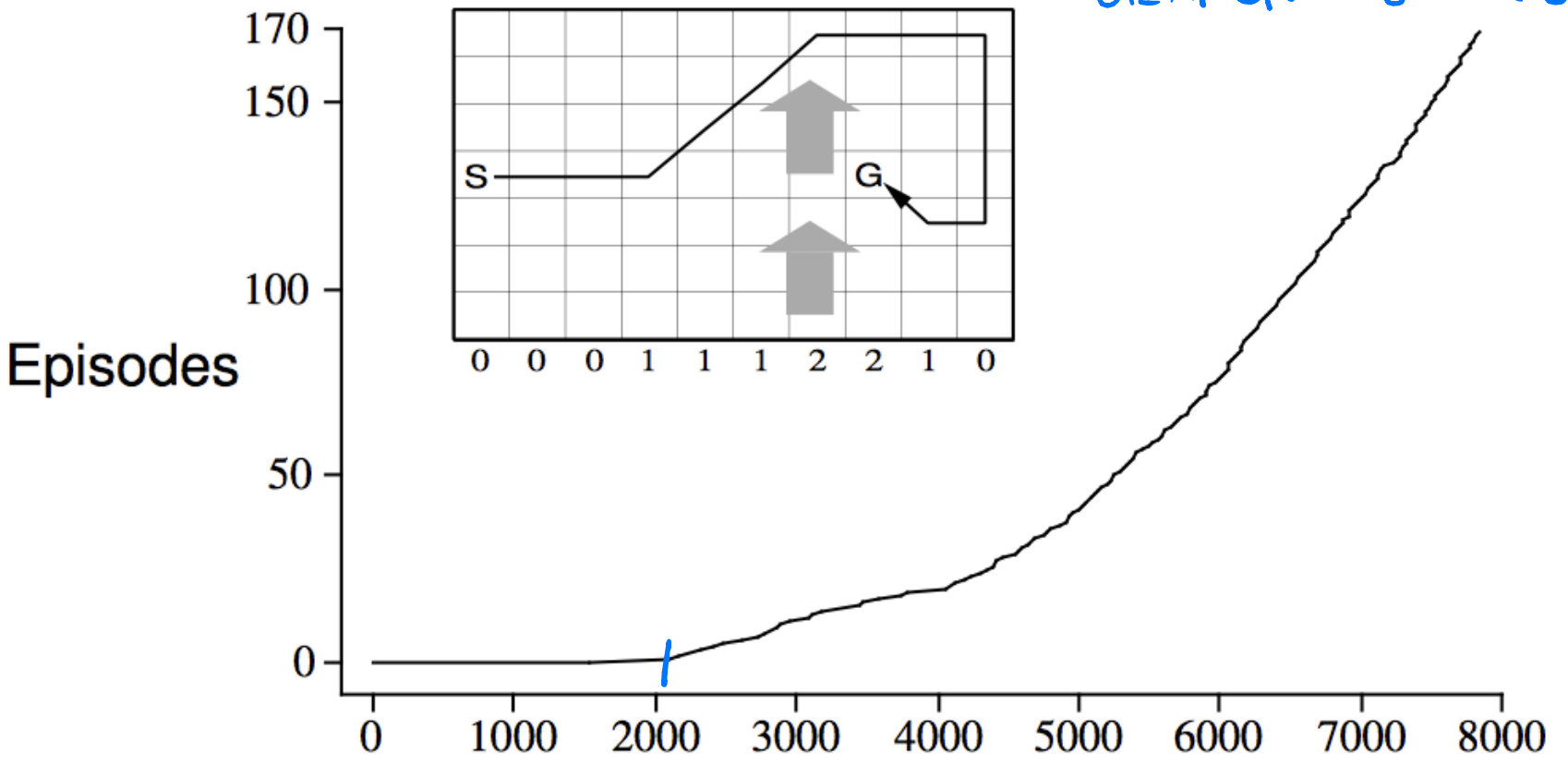


king's moves

- Reward = -1 per time-step until reaching goal
- Undiscounted

Sarsa on the Windy Gridworld

한번 도달하면
그정보가 전파되기때문에
빠르게 epi가 증가하게된다.



2000번 움직여야 1번 epi끝남. → 그래서 한번 도달하더라도 상당히 빨리 epi끝남.

n-Step Sarsa

- Consider the following n-step returns for $n = 1, 2, \infty$:

$$\begin{array}{lll} n = 1 & (Sarsa) & q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}) \\ n = 2 & & q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}) \\ \vdots & & \vdots \\ n = \infty & (MC) & q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T \end{array}$$

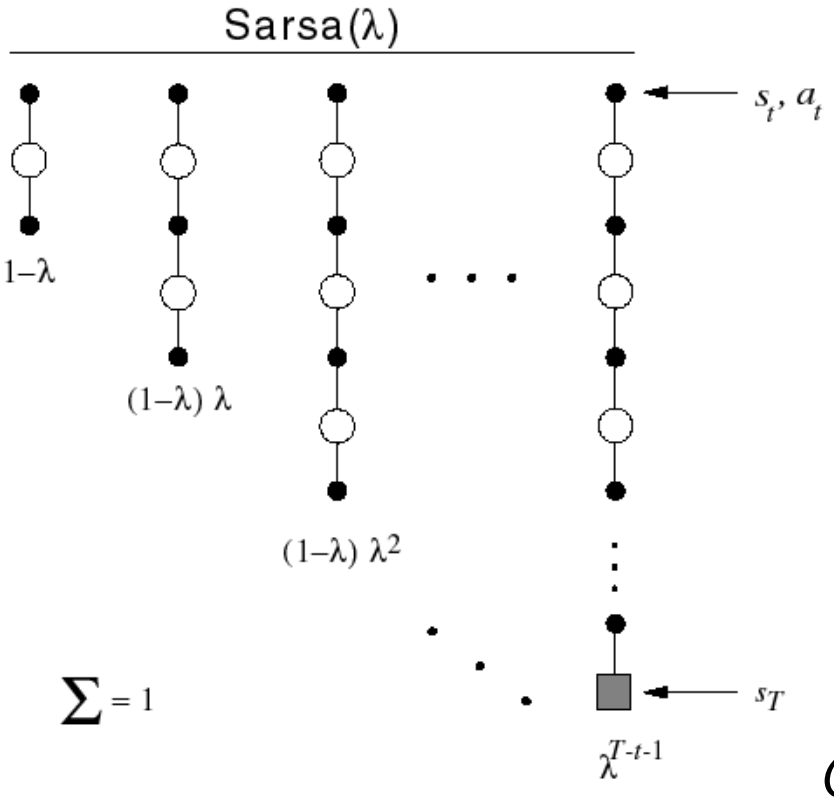
- Define the n-step Q-return

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

- n-step Sarsa updates $Q(s, a)$ towards the n-step Q-return

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

Forward View Sarsa(λ)



- The q^λ *return* combines all n-step Q-returns $q_t^{(n)}$
- Using weight $(1-\lambda)\lambda^{n-1}$

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

- Forward-view Sarsa(λ)

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

Backward View Sarsa(λ)

- Just like TD(λ), we use eligibility traces in an online algorithm
- But Sarsa(λ) has one eligibility trace for each state-action pair

$$E_0(s, a) = 0$$
$$E_t(s, a) = \gamma\lambda E_{t-1}(s, a) + \mathbf{1}(S_t = s, A_t = a)$$

- $Q(s, a)$ is updated for every state s and action a
- In proportion to TD-error δ_t and eligibility trace $E_t(s, a)$

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

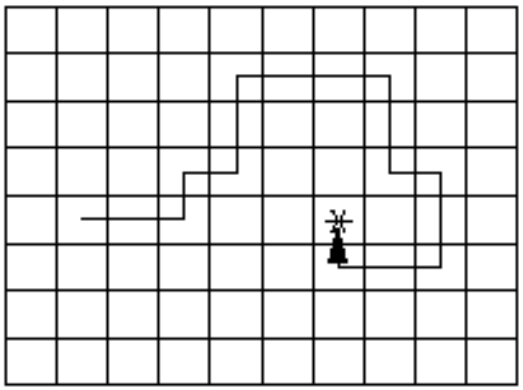
Sarsa(λ) Algorithm

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Repeat (for each episode):
  $E(s,a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
  Initialize S, A
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $\delta \leftarrow R + \gamma Q(S',A') - Q(S,A)$
    $E(S,A) \leftarrow E(S,A) + 1$
    For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
      $Q(s,a) \leftarrow Q(s,a) + \alpha \delta E(s,a)$
      $E(s,a) \leftarrow \gamma \lambda E(s,a)$
    $S \leftarrow S'$; $A \leftarrow A'$
  until S is terminal

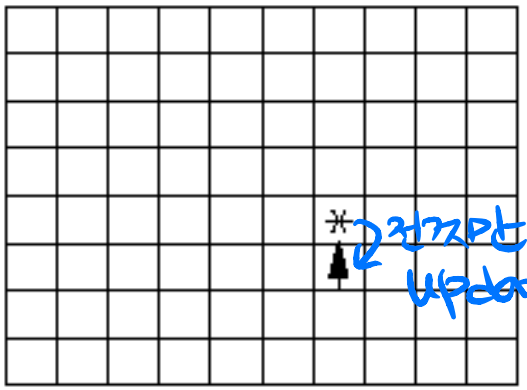
모든 s, a에 대해 update함
과거에 지난 칸들도 E 값이 있기때문이다.
계산량 ↑ 정보전파 ↑

Sarsa(λ) Gridworld Example

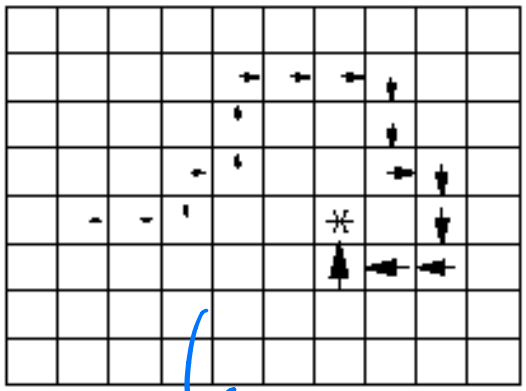
Path taken



Action values increased by one-step Sarsa



Action values increased by Sarsa(λ) with λ=0.9



Off-Policy Learning

target policy π
behaviour policy μ

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

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

- Why is this important? 사람이 한걸음을 보고 좋으면 따라하고 아니면 안하는 agent를 설한다.
- Learn from observing humans or other agents
- Re-use experience generated from old policies $\pi_1, \pi_2, ..., \pi_{t-1}$ ← 예전에 한 경험을 재사용할수있다.
- Learn about *optimal* policy while following *exploratory* policy
- Learn about *multiple* policies while following *one* policy

Importance Sampling → rejection sampling 보완하려고나온 것.

↳ 기대값, 확률을 계산하기 위함.

- Estimate the expectation of a different distribution

$$\mathbb{E}_{X\sim P}[f(X)] = \sum P(X)f(X)$$
$$= \sum Q(X)\frac{P(X)}{Q(X)}f(X)$$
$$= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)}f(X)\right]$$

?

X: 확률분포 P에서 샘플링되는 것들.

함수

Variance가 너무 커서 실제로 적용할 수 없는 알고리즘이 아니다. (등장을 안함)

Importance Sampling for Off-Policy Monte-Carlo

- Use returns generated from μ to evaluate π
- Weight return G_t according to similarity between policies
- Multiply importance sampling corrections along whole episode

μ와 π의 비율을 → 매우 고려한다.

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

- Update value towards *corrected* return

교정한다고 생각하면 된다.

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

- Cannot use if μ is zero when π is non-zero

μ가 0이면 동작 X.

- Importance sampling can dramatically increase variance

Importance Sampling for Off-Policy TD

↳ 1step DP update.

- Use TD targets generated from μ to evaluate π
- Weight TD target $R + \gamma V(S')$ by importance sampling
- Only need a single importance sampling correction

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

→ variance ↑ 가능.

- Much lower variance than Monte-Carlo importance sampling
- Policies only need to be similar over a single step

Q-Learning

off-Policy Algorithm.

기본 TD는 Policy가 하나.

- 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

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

Bootstrap

상상의 일임.

기본 TD는 $R + \gamma Q(\,)$

사용함.

여기에 μ 대신 π 를 사용함.

Next action을 하나 뽑아서 진행한다. S_{t+1}에 도착함.

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. $Q(s, a)$

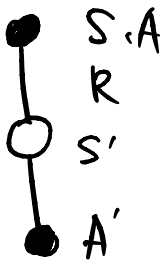
$$\pi(S_{t+1}) = \underset{a'}{\text{argmax}}\ Q(S_{t+1}, a')$$

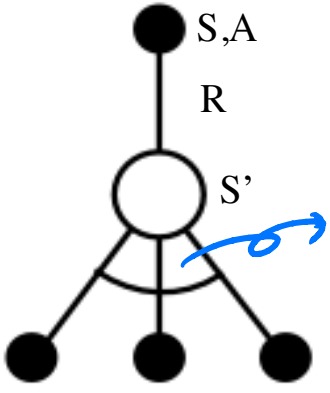
- The behaviour policy μ is e.g. ϵ-greedy w.r.t. $Q(s, a)$
- The Q-learning target then simplifies:

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

Q-Learning Control Algorithm = SARSA Max

SARSA





max

→내가할수있는 action들.

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

Theorem

Q-learning control converges to the optimal action-value function, $Q(s,a) \rightarrow q_*(s,a)$

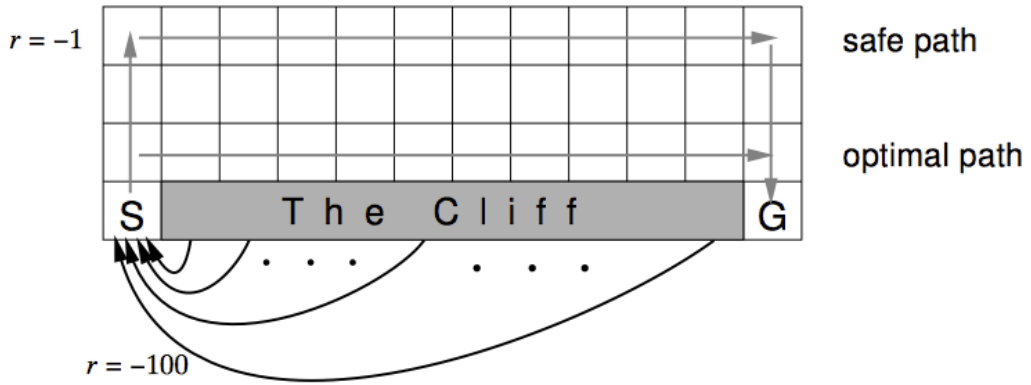
Q-Learning Algorithm for Off-Policy Control

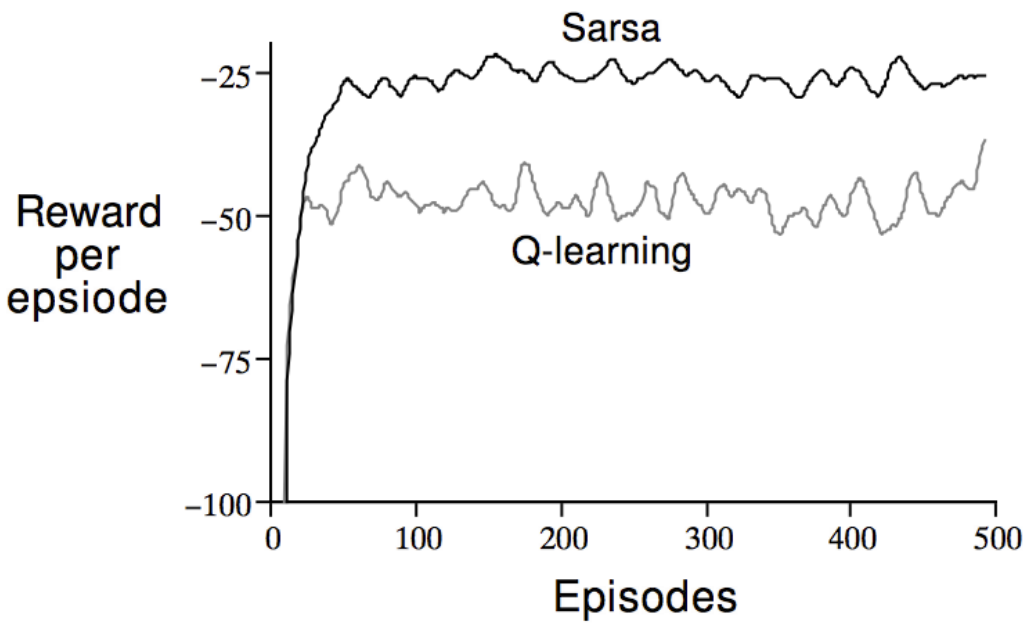
Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$;
  until S is terminal

Q-Learning Demo

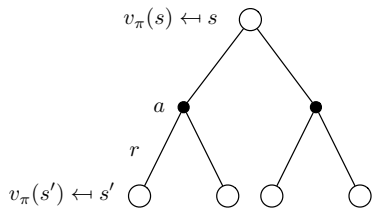
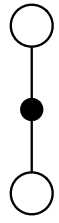
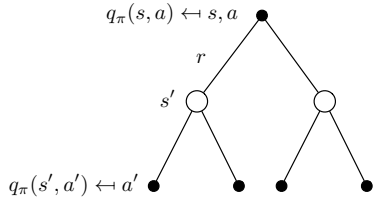
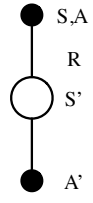
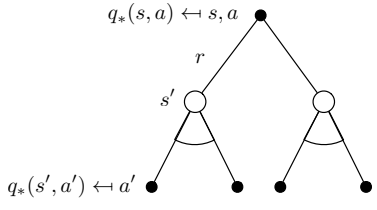
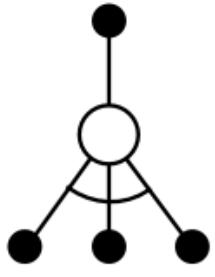
Q-Learning Demo

Cliff Walking Example





Relationship Between DP and TD

	Full Backup (DP)	*Sample Backup (TD)*
Bellman Expectation Equation for $v_\pi(s)$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $q_\pi(s,a)$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $q_*(s,a)$	 Q-Value Iteration	 Q-Learning

Relationship Between DP and TD (2)

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

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

Questions?

그동안 내용에 대한 피드백.

MDP 알 때

Prediction

Q-learning

Model free
Control
off policy

MDP 모를 때

Control
- on policy
- off policy