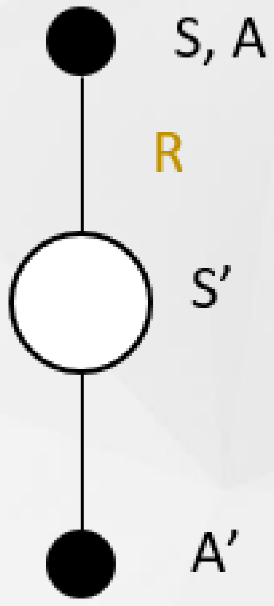
Chapter 06. 스스로 전략을 짜는 강화학습 (Reinforcement Learning)

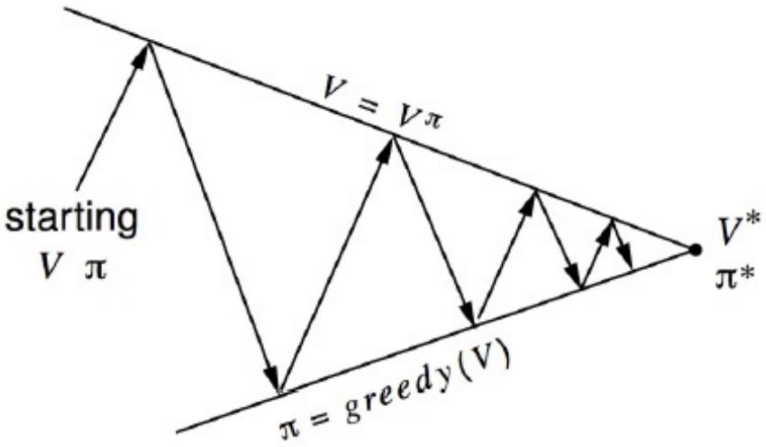
Sarsa , Q- learning



Monte Carlo Method

Monte-Carlo Control

Policy Iteration *= (policy evaluation + policy improvement)*
Monte-Carlo policy Iteration *= (MC policy evaluation + policy improvement)*



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

Problem 1 . Value Function -> MDP

$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Problem 2 . Greedy policy improvement
-> Local Optimum

Monte Carlo Method

Problem 1 . Value Function -> MDP

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

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

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

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

Problem 2 . Greedy policy improvement -> Local Optimum

$$\pi(s) \doteq \arg\max_a q(s, a).$$

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s, a)) \\ &= \max_a q_{\pi_k}(s, a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &\geq v_{\pi_k}(s). \end{aligned}$$

- 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

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

Temporal Difference

https://sumniya.tistory.com/14?category=781573

에피소드 마다 가 아니라 매 타임스텝 마다 가치함수를 업데이트

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

TD(0) : $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$

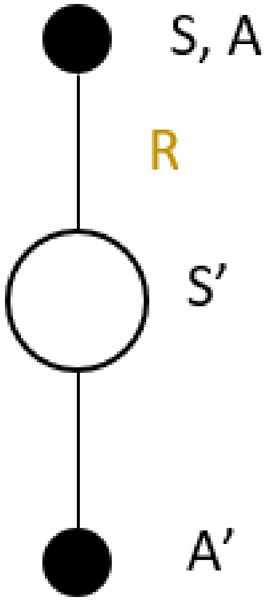
Input : the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0, \forall s \in S^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A; observe reward, R, and next state, S'
    $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
    $S \leftarrow S'$
  until S is terminal

Temporal Difference Control

https://www.slideshare.net/DongMinLee32/part-2-91522217

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

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

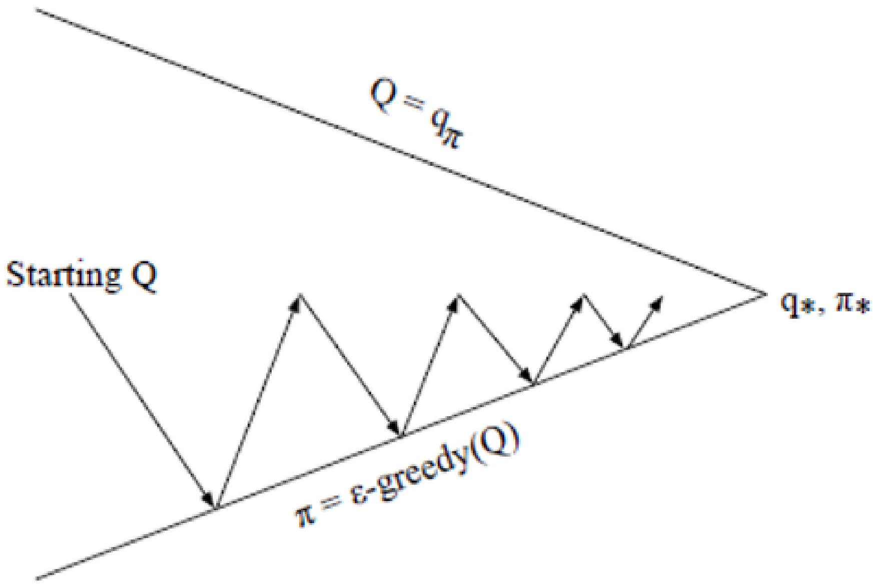


$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

$[S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}]$을 하나의 샘플로 사용하기 때문에 SARSA라고 합니다.
앞으로는 시간차 제어가 아닌 살사라고 부르겠습니다.

SARSA

On-Policy Control With Sarsa



Every time-step:

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

Policy improvement ϵ-greedy policy improvement

SARSA

Initialize $Q(s, a)$, $\forall s \in S$, $a \in A(s)$, arbitrarily and $Q(\text{terminal-state}, \cdot) = 0$
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)
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

N- step SARSA

n-Step Sarsa

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

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

- Define the n-step Q-return

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

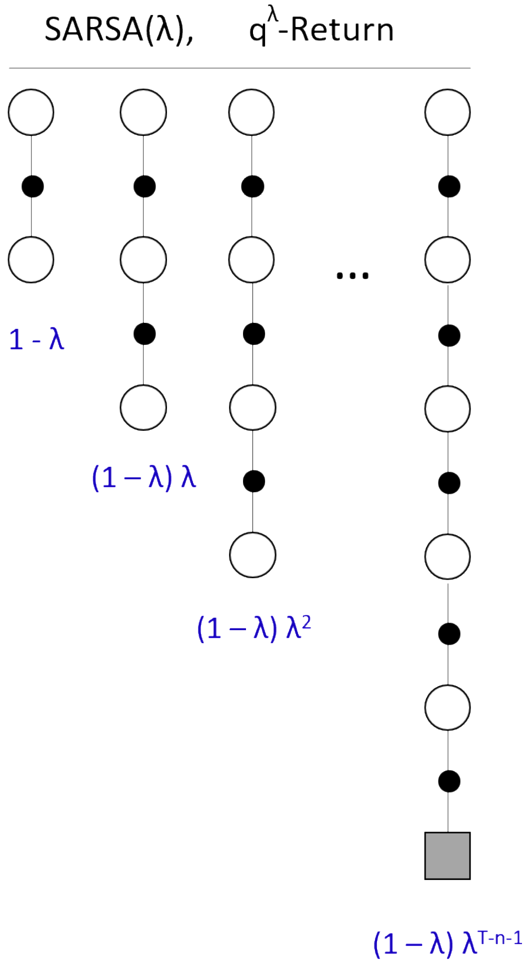
- n-step Sarsa updates $Q(s, a)$ towards the n-step Q-return

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

SARSA(λ)

https://sumniya.tistory.com/

Forward Sarsa



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

$$\text{when } q^\lambda\text{-return, } q_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}q_t^{(n)}$$

SARSA(λ)

$$E_t(s, a) = \begin{cases} \gamma\lambda E_{t-1}(s, a) + 1 & if\ s = s_t,\ a = a_t \\ \gamma\lambda E_{t-1}(s, a) & otherwise \end{cases}$$

Backward Sarsa

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\delta_t E_t(s, a)$$

$$when\ \delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\ (TD\ error)$$

Initialize $Q(s, a)$ arbitrarily, $\forall s \in S$, $a \in A(s)$
Repeat (for each episode):
    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 ∈ S, , a ∈ 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

On / Off Policy

On-policy :

학습하는 policy와 행동하는 policy가 반드시 같아야만 학습이 가능한 강화학습 알고리즘.
ex) Sarsa

on-policy의 경우 1번이라도 학습을 해서 policy improvement를 시킨 순간, 그 policy가 했던 과거의 experience들은 모두 사용이 불가능하다. 즉 매우 데이터 효율성이 떨어진다. 바로바로 exploration해서 학습하고 재사용이 불가능하다.

Off-policy :

학습하는 policy와 행동하는 policy가 반드시 같지 않아도 학습이 가능한 알고리즘.
ex) Q-learning

off-policy는 현재 학습하는 policy가 과거에 했던 experience도 학습에 사용이 가능하고, 심지어는 해당 policy가 아니라 예를 들어 사람이 한 데이터로부터도 학습을 시킬 수가 있다.

Off Policy

- Learn from observing humans or other agents
- Re-use experience generated from old policies $\pi_1, \pi_2, \dots, \pi_{t-1}$
- Learn about ***Optimal Policy*** while <u>following exploratory policy</u>
- Learn about ***multiple policies*** while <u>following one policy</u>

Q- Learning

https://www.slideshare.net/DongMinLee32/part-2-91522217

행동하는 정책

ε-탐욕 정책

학습하는 정책

다음 상태에서 어떤 행동을 할 때
다음 상태의 **최대 큐함수**를
현재 상태의 큐함수로 업데이트

Q- Learning

https://www.slideshare.net/DongMinLee32/part-2-91522217

살사의 큐함수 업데이트

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

$$\downarrow$$

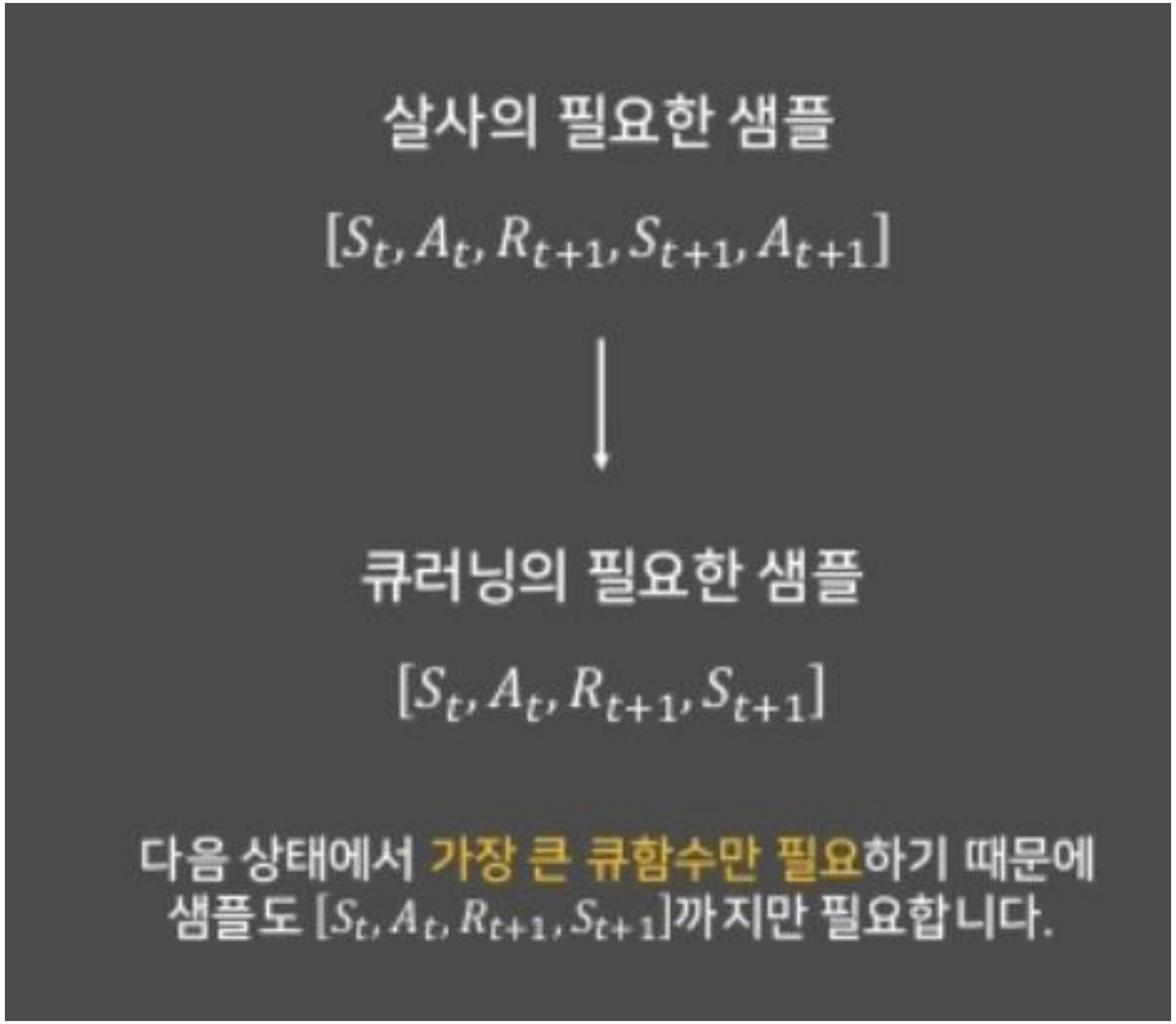
큐러닝의 큐함수 업데이트

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + a(R_{t+1} + \gamma \max_{a`} Q(S_{t+1}, a`) - Q(S_t, A_t))$$

다음 상태에서 다음 행동을 해보는 것이 아니라
다음 상태에서 가장 큰 큐함수를 가지고 업데이트

Q- Learning

https://www.slideshare.net/DongMinLee32/part-2-91522217



Q- Learning

https://sumniya.tistory.com/15?category=781573

1) 현재 state S 에서 behavior policy, μ(e.g. ε-greedy)에 따라 action A을 선택.
2) q-func.을 이용하여 다음 state S'에서의 action A'는 π(e.g. greedy)에 따라 선택.

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

3) Q-learning의 target은 아래 식으로 도출.

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

4) 아래 식에 따라 q-func.을 update.

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \underset{a'}{max} Q(S', A') - Q(S, A))$$

fast campus

Q- Learning

https://www.slideshare.net/CurtPark1/dqn-reinforcement-learning-from-basics-to-dqn

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big].$$

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

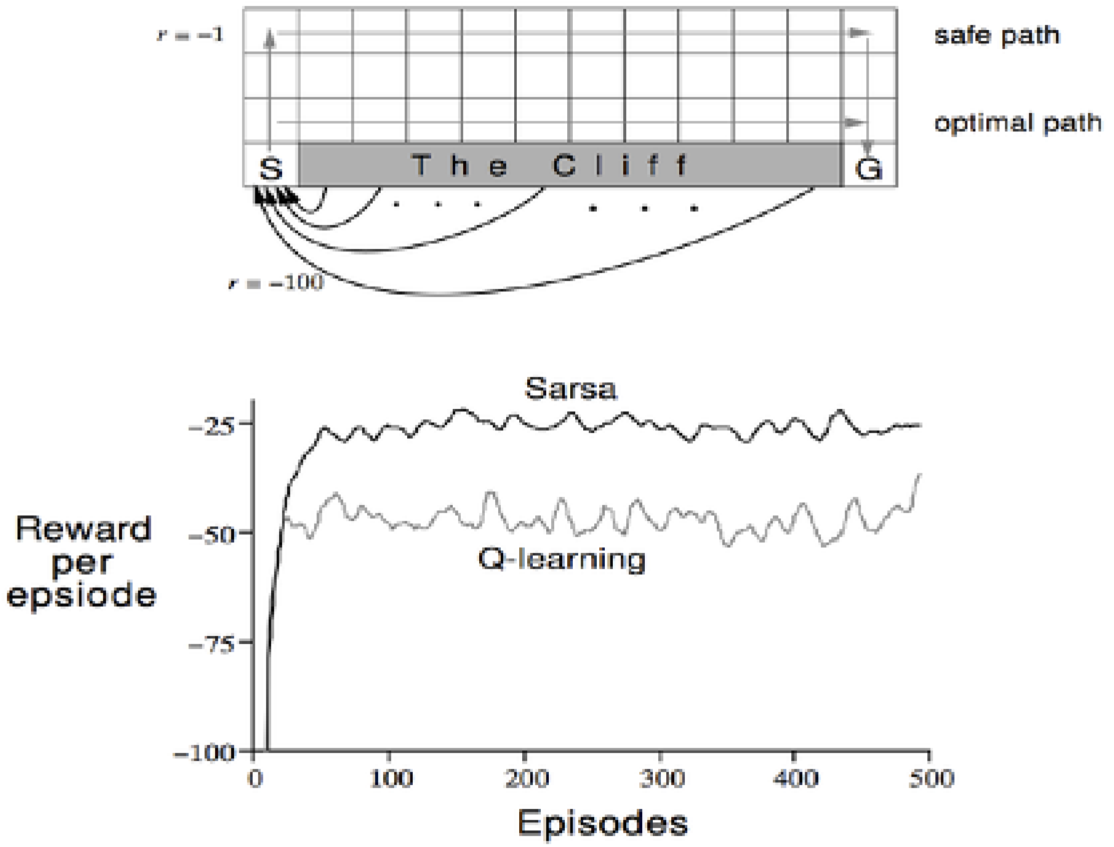
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode: Behavior policy로 동작
    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

Target policy로 동작

Sarsa & Q- Learning

Cliff Walking Example



- *Thank you*