Ch.5
Monte Carlo Methods

201721153 강인호

Before we start..

Dynamic Programming -> Bellman Equation -> optimal solution

By MDP(Markov Decision Processes)

문제점

- Full-width Backup -> expensive computation
- Full knowledge about Environment

Before we start..

<u>Model-free</u> : Environment를 모르고 학습

Policy -> sampling -> value function update : Model-free prediction

-> policy update : Model-free control

Model-free

- Monte-Carlo
- Temporal Difference

Monte-Carlo vs Temporal Difference

Episode by episode

Time step by step

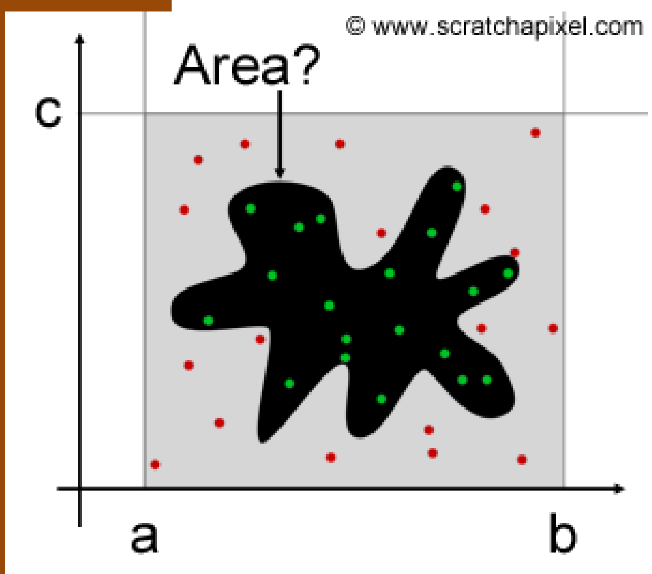
Monte-Carlo Prediction

무엇인가를 random하게 측정

Value function : 이 state에서 시작해서 미래까지 받을 기대되는 rewards의 총합

Episode를 끝까지 가본 후 rewards들로 value function을 거꾸로 계산

But, episode가 끝나지 않는다면 사용 불가능



- Goal: learn v_π from episodes of experience under policy π

$$S_1, A_1, R_2, ..., S_k \sim \pi$$

- Recall that the *return* is the total discounted reward:

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

- Recall that the value function is the expected return:

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

- Monte-Carlo policy evaluation uses *empirical mean* return instead of *expected* return

Monte-Carlo Prediction

여러 개의 episode를 진행한 후 return은 단순히 평균을 내줌
-> 쌓일수록 true value function에 가까워짐

First-Visit MC	Every-Visit MC
처음 방문한 state만 계산	방문할 때마다 계산

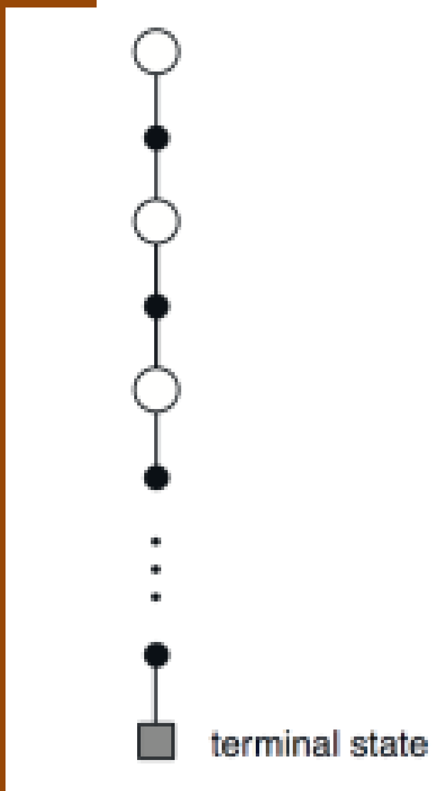
First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:
  $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$
  $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):
  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:
      Append G to $Returns(S_t)$
      $V(S_t) \leftarrow \text{average}(Returns(S_t))$

Monte-Carlo Prediction



-> DP에 비해 MC는 Variance가 높지만 bias가 낮은 편

Monte-Carlo Prediction

Blackjack Sticks and hits

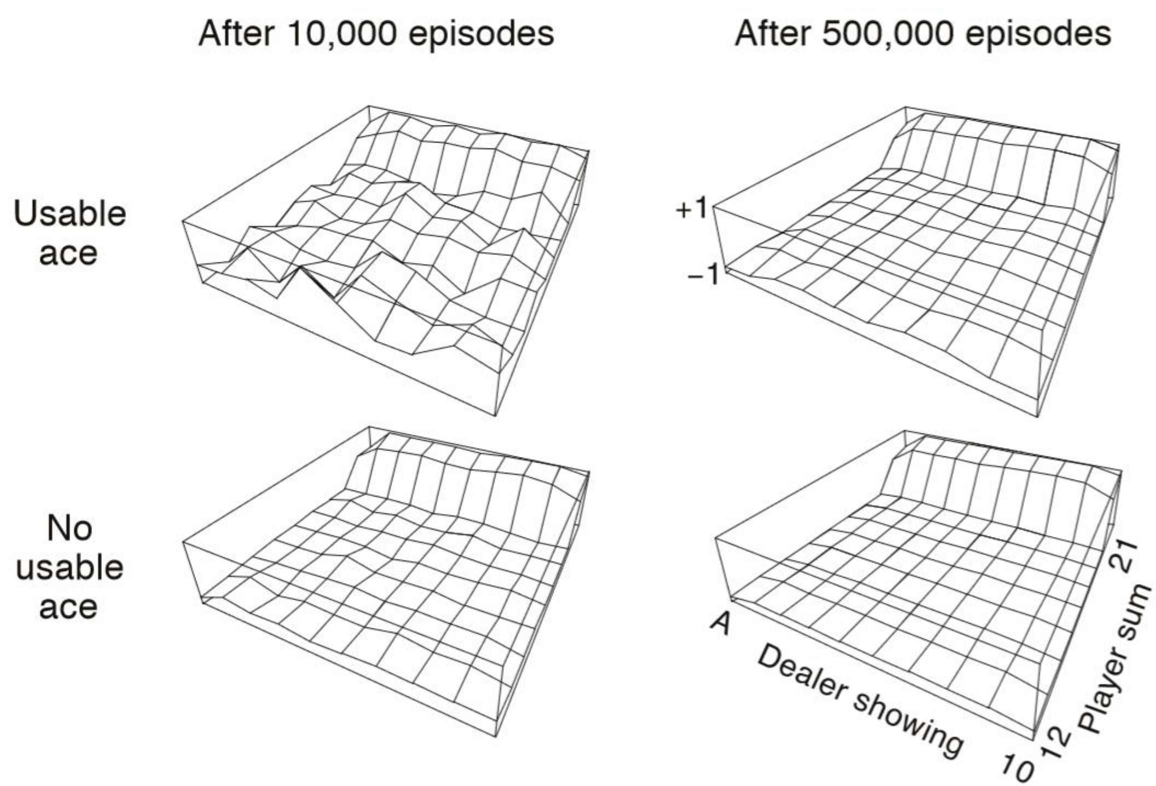
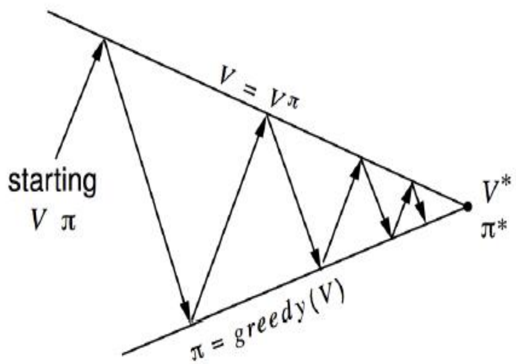


Figure 5.1: Approximate state-value functions for the blackjack policy that sticks only on 20 or 21, computed by Monte Carlo policy evaluation.

Monte-Carlo Control

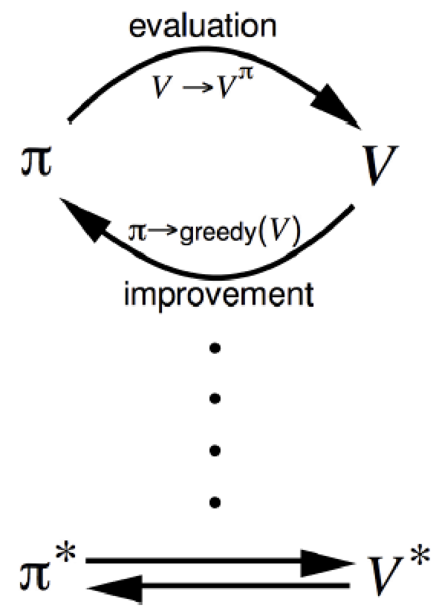
GPI(General Policy Iteration)을 이용

Generalised Policy Iteration (Refresher)



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

Policy improvement Greedy policy improvement?



*Control : RL에선 optimal policy를 찾는 것을 control이라 함

Monte-Carlo Control

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

State value function이 아닌 action value function을 사용

Monte Carlo ES (Exploring Starts)

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \leftarrow$ arbitrary
  $\pi(s) \leftarrow$ arbitrary
  $Returns(s,a) \leftarrow$ empty list

Repeat forever:
  Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
  Generate an episode starting from S_0, A_0, following π
  For each pair s, a appearing in the episode:
    $G \leftarrow$ return following the first occurrence of s, a
    Append G to $Returns(s,a)$
    $Q(s,a) \leftarrow$ average$(Returns(s,a))$
  For each s in the episode:
    $\pi(s) \leftarrow \text{argmax}_a Q(s,a)$

Exploring starts: Every state-action pair has a non-zero probability of being the starting pair

Monte-Carlo Control

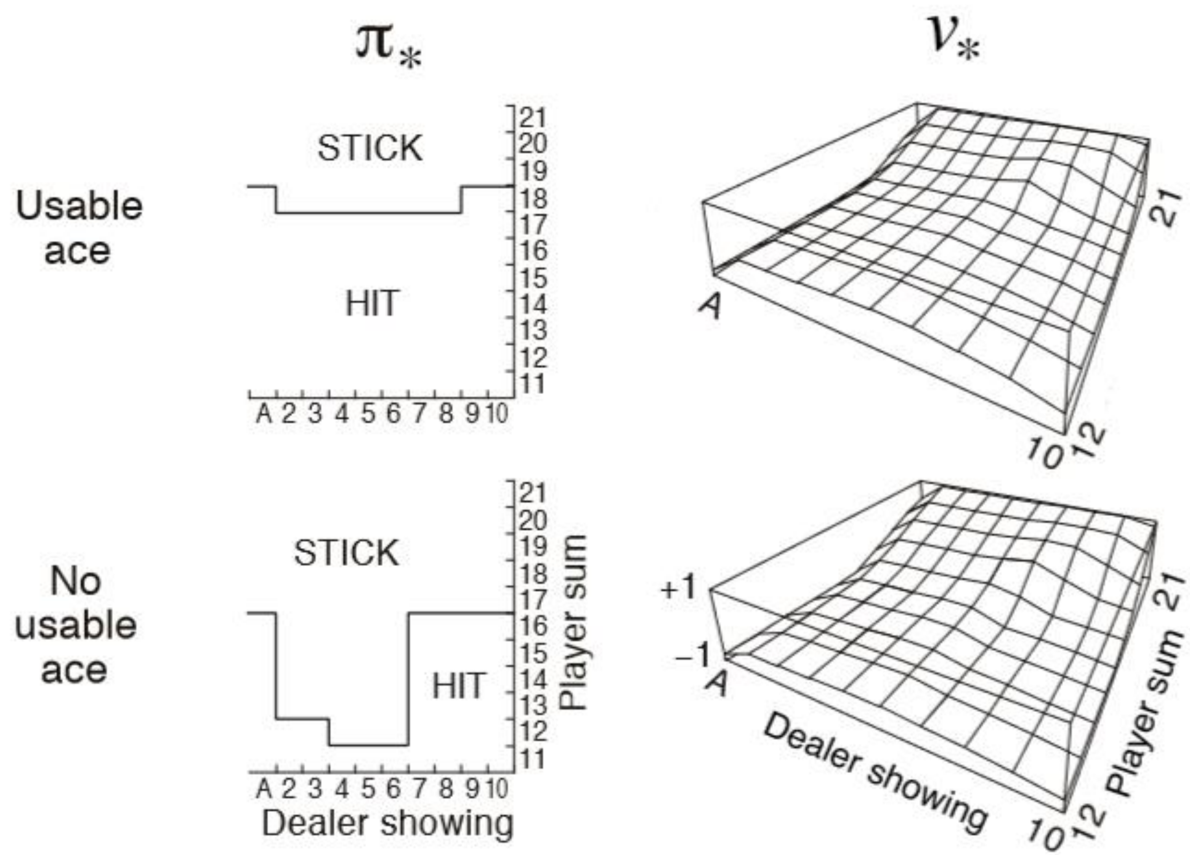


Figure 5.2: The optimal policy and state-value function for blackjack, found by Monte Carlo ES. The state-value function shown was computed from the action-value function found by Monte Carlo ES. ■

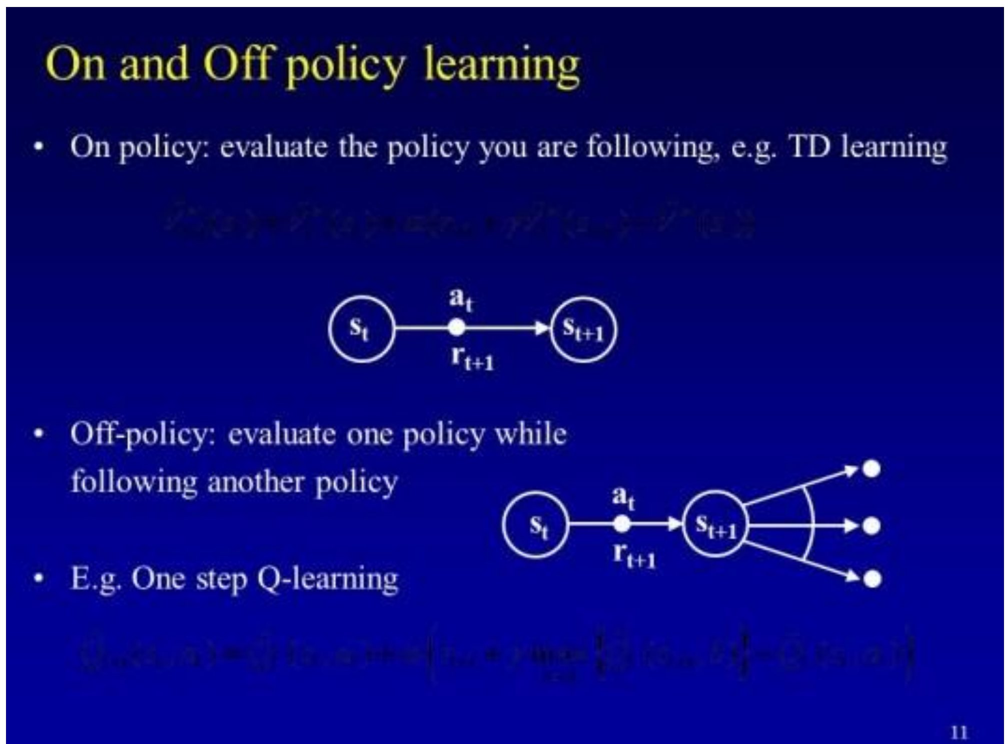
Monte-Carlo Control without Exploring Starts

On-policy methods

결정에 사용된 policy를 평가하고 개선

Off-policy methods

결정에 필요한 data를 만드는데 사용된 Policy와는 다른 policy를 평가하고 개선



Monte-Carlo Control without Exploring Starts

On-policy first-visit MC control (for ε-soft policies)

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \leftarrow$ arbitrary
  $Returns(s,a) \leftarrow$ empty list
  $\pi(a|s) \leftarrow$ an arbitrary ε-soft policy

Repeat forever:
  (a) Generate an episode using π
  (b) For each pair s, a appearing in the episode:
    $G \leftarrow$ return following the first occurrence of s, a
    Append G to $Returns(s,a)$
    $Q(s,a) \leftarrow \text{average}(Returns(s,a))$
  (c) For each s in the episode:
    $A^* \leftarrow \arg\max_a Q(s,a)$
    For all $a \in \mathcal{A}(s)$:

$$\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq A^* \end{cases}$$

Monte-Carlo Control without Exploring Starts

Off-policy

두 종류의 policy를 사용하는 방법으로,
하나는 optimal policy가 되는 방법을 학습하고,
다른 하나는 탐색을 추구하면서 behavior를 결정하도록 만드는 방법

Off-policy every-visit MC control (returns $\pi \approx \pi_*$)

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow$ a deterministic policy that is greedy with respect to Q

Repeat forever:
Generate an episode using any soft policy μ:
$S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T, S_T$
$G \leftarrow 0$
$W \leftarrow 1$
For $t = T-1, T-2, \dots$ downto 0:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$\pi(S_t) \leftarrow \mathrm{argmax}_a\, Q(S_t, a)$ (with ties broken consistently)
If $A_t \neq \pi(S_t)$ then ExitForLoop
$W \leftarrow W \frac{1}{\mu(A_t|S_t)}$

감 사 합 니 다 !