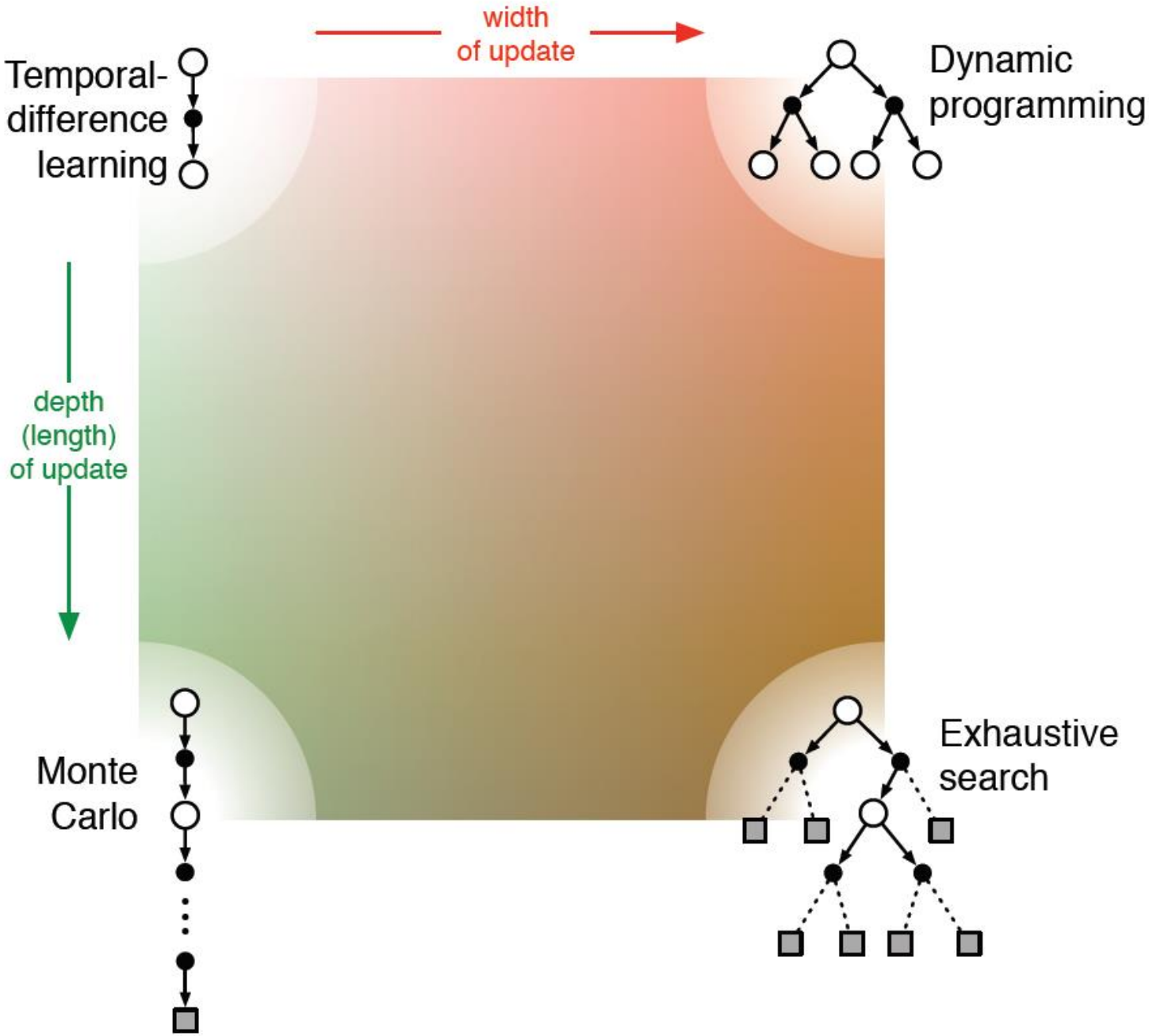
Monte Carlo Methods

Prof. Kuan-Ting Lai

2020/4/17

Monte Carlo Methods

- Learn directly from episodes of experience
- Model-free: no knowledge of MDP transitions / rewards
- Learn from complete episodes (episodic MDP): no bootstrapping
- Use the simplest idea: value = mean return



Sutton, Richard S.; Barto, Andrew G.. Reinforcement Learning (Adaptive Computation and Machine Learning series) (p. 189)

Monte Carlo Prediction

- First-visit MC vs. Every-visit MC

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

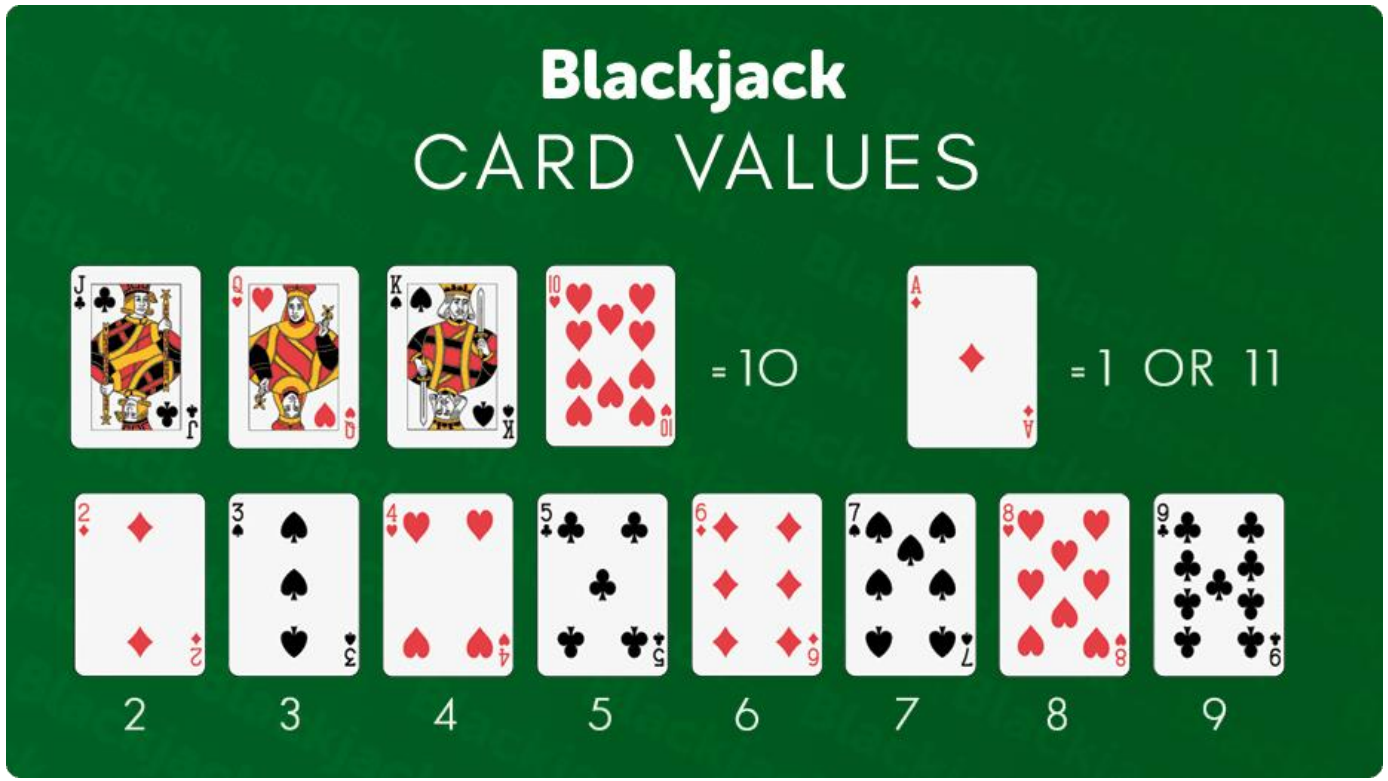
Input: a policy π to be evaluated
Initialize:
  $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in S$
  $Returns(s) \leftarrow$ an empty list, for all $s \in S$

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 $Returns(S_t)$
      $V(S_t) \leftarrow$ average($Returns(S_t)$)

Blackjack (21)

https://www.imdb.com/title/tt0478087/

Rules of Blackjack

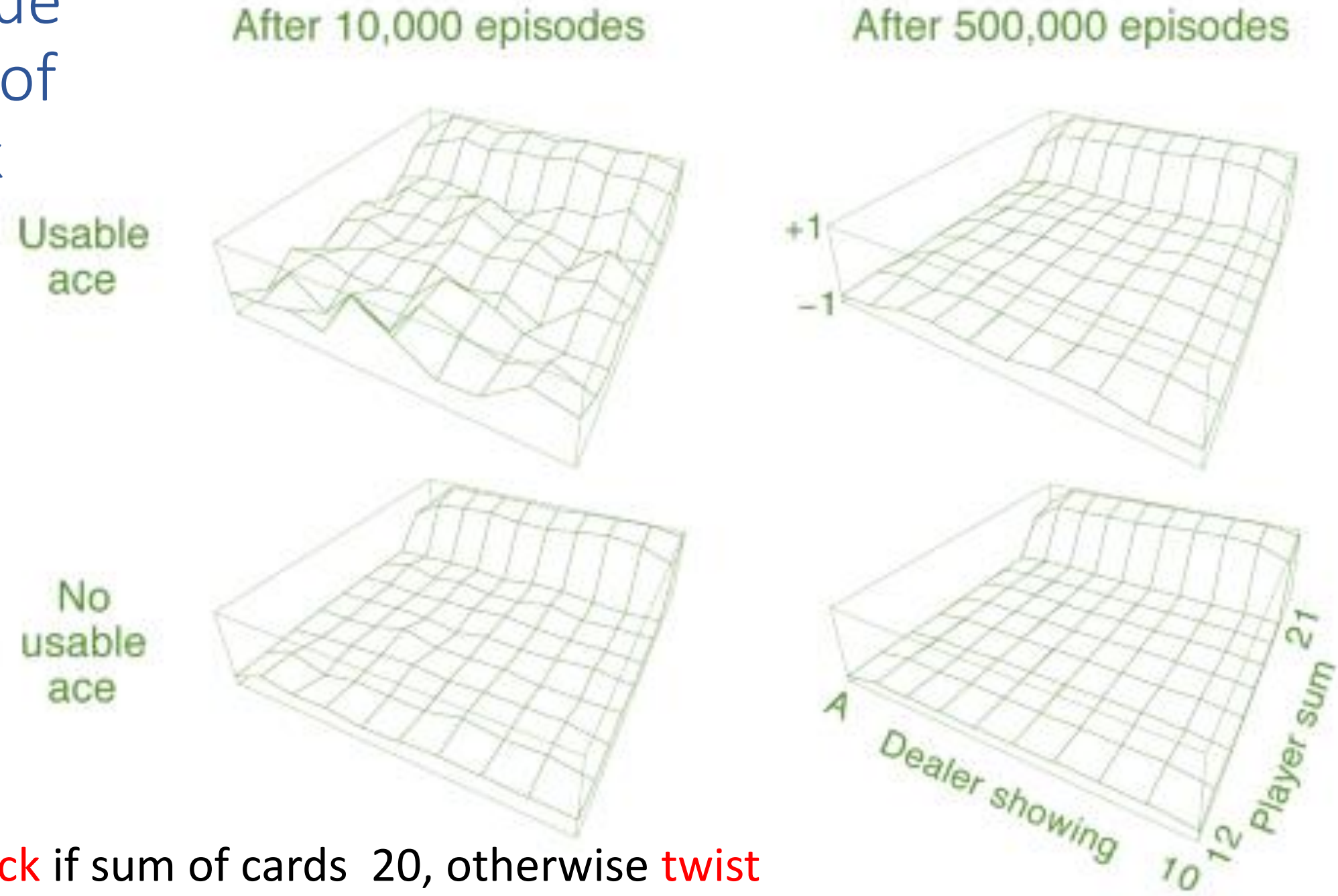


- Goal: Each player tries to beat the dealer by getting a count as close to 21 as possible
- Lose if total > 21 (bust)
- The game begins with two cards dealt to both dealer and player
- One of the dealer's cards is face up and the other is face down
- Actions
  - Hit: Requests additional card
  - Stick: stop getting cards
- Dealer sticks when his sum ≥ 17

Reinforcement Learning of Blackjack

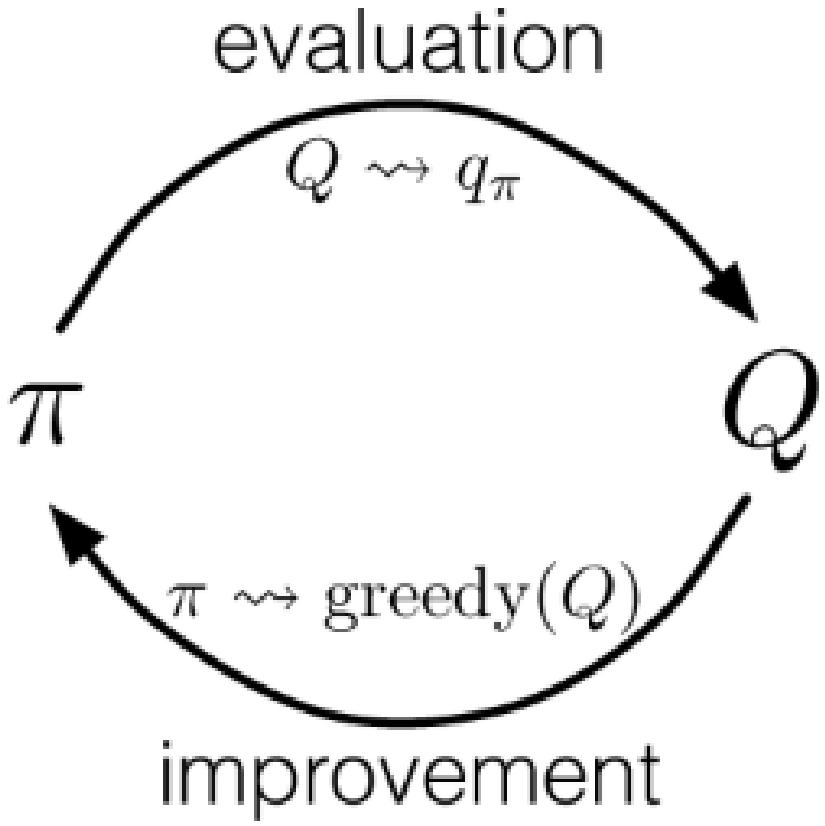
- States
  - Player's current sum (12 ~ 21)
  - Dealers' showing cards (ace, 2 ~ 10)
  - Use A as 1 or 11
  - Total states: 10*10*2 = 200
- Reward
  - 1: Winning
  - -1: losing
  - 0: drawing
- ** Automatically call if sum < 12

State-value function of Blackjack



Policy: stick if sum of cards 20, otherwise twist

Monte Carlo Control



$$\pi_0 \xrightarrow{\text{E}} q_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} q_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} q_*$$

Exploring Starts for Monte Carlo

- Many state-action may never be visited
- Randomly choose state-action pairs and run a lot of episodes

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in A(s)$ (arbitrarily), for all $s \in S$

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in S$, $a \in A(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in S$, $a \in A(s)$

Loop forever (for each episode):

Choose $S_0 \in S$, $A_0 \in A(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

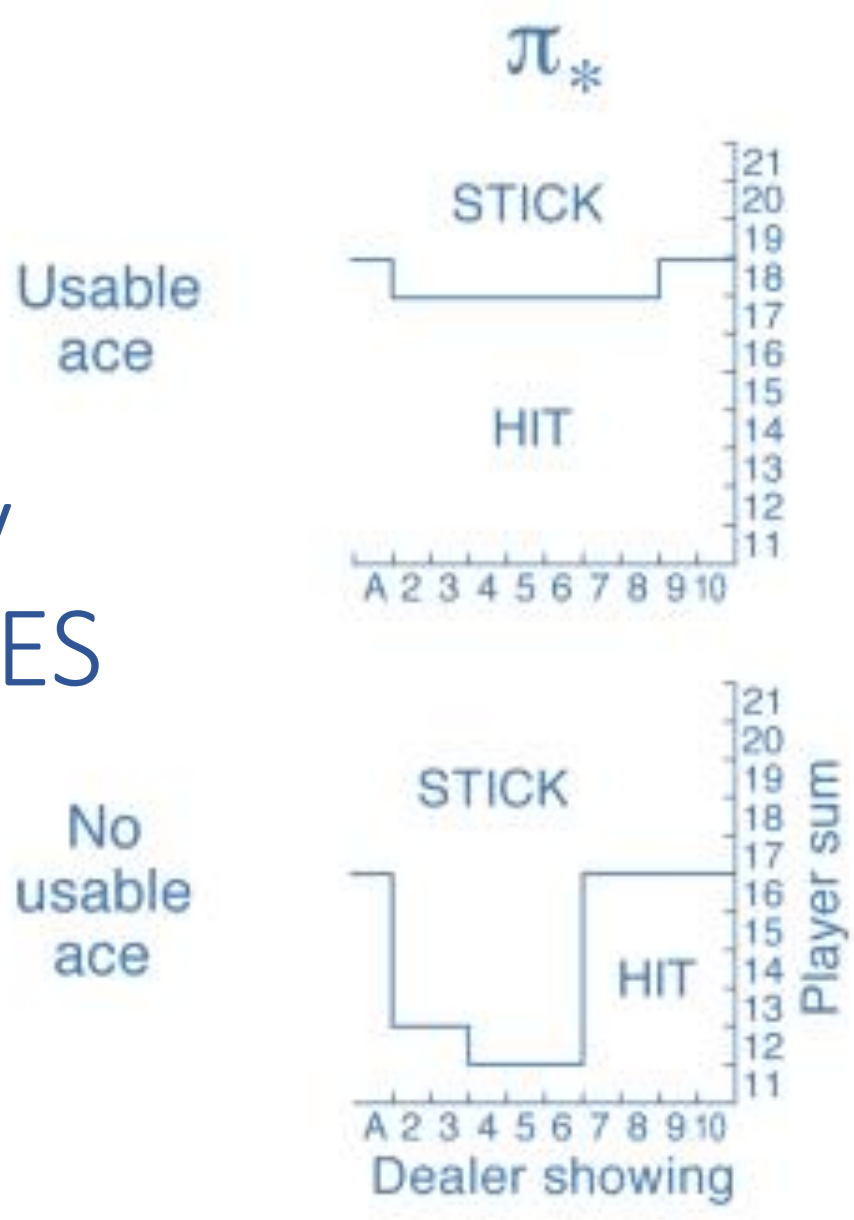
Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

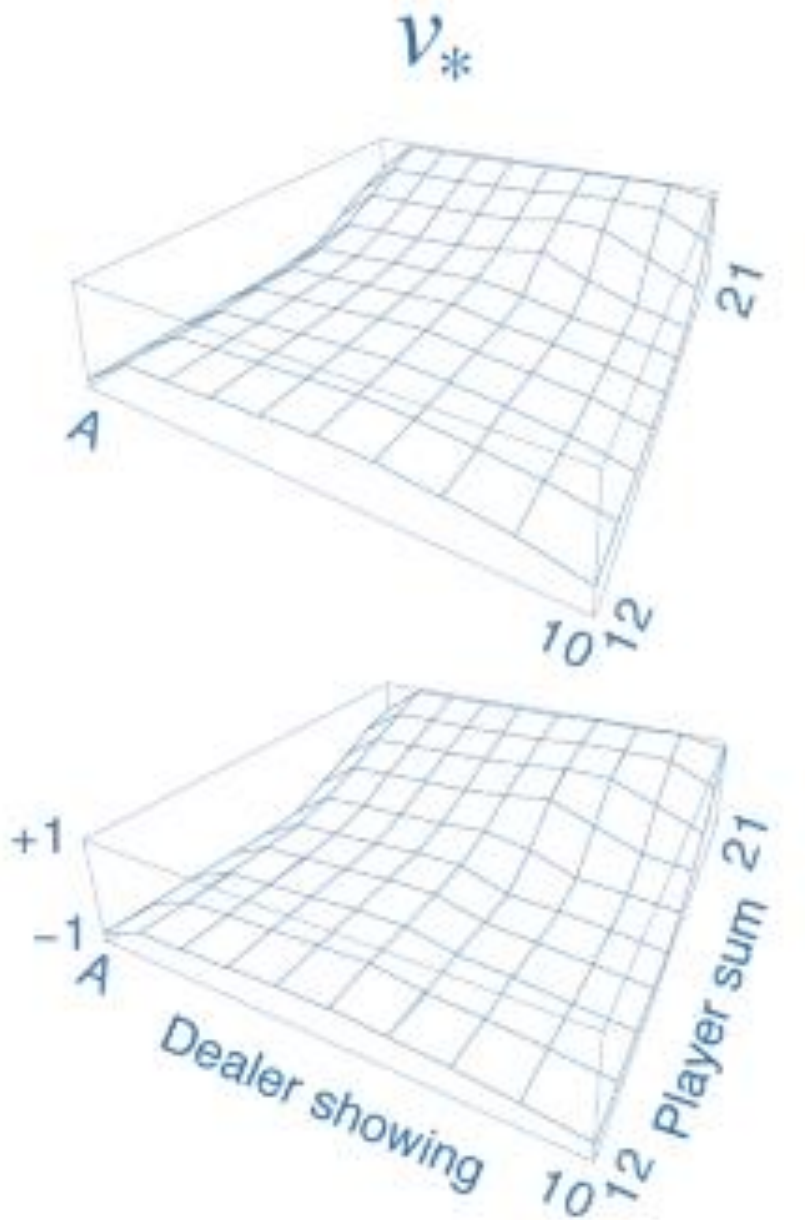
Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow$ average$(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Optimal Policy Learnt by MC ES





Monte Carlo Control without Exploring Starts

- On-policy
  - ε-greedy
- Off-policy
  - Importance sampling

On-policy first-visit MC Control (for ε-greedy)

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in S$, $a \in A(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in S$, $a \in A(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \text{argmax}_a\, Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

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

Off-policy Prediction via Importance Sampling

- Use two policies
  - Target policy: the optimal policy we want to learn
  - behavior policy: more exploratory, used to generate behaviors
- How to update target policy using behavior polic?
  - Importance sampling

Importance Sampling

- Probability of state-action trajectory

$$\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&\quad = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}$$

- Relative trajectory probability of target behavior policies

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

Update using Importance-sampling ratio

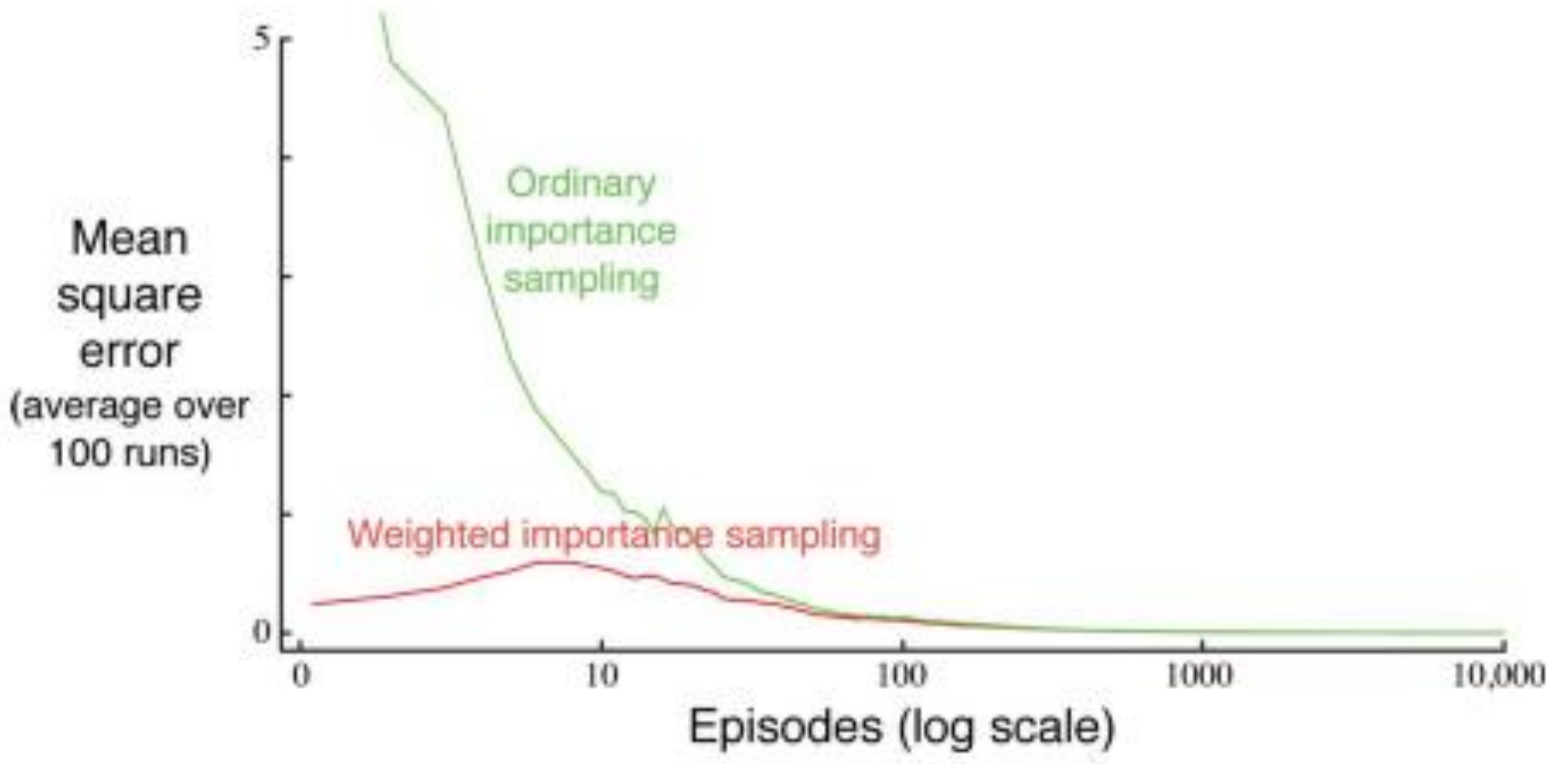
$$E[\rho_{t:T-1} G_t \mid S_t = s] = V_\pi(s)$$

Simple Average

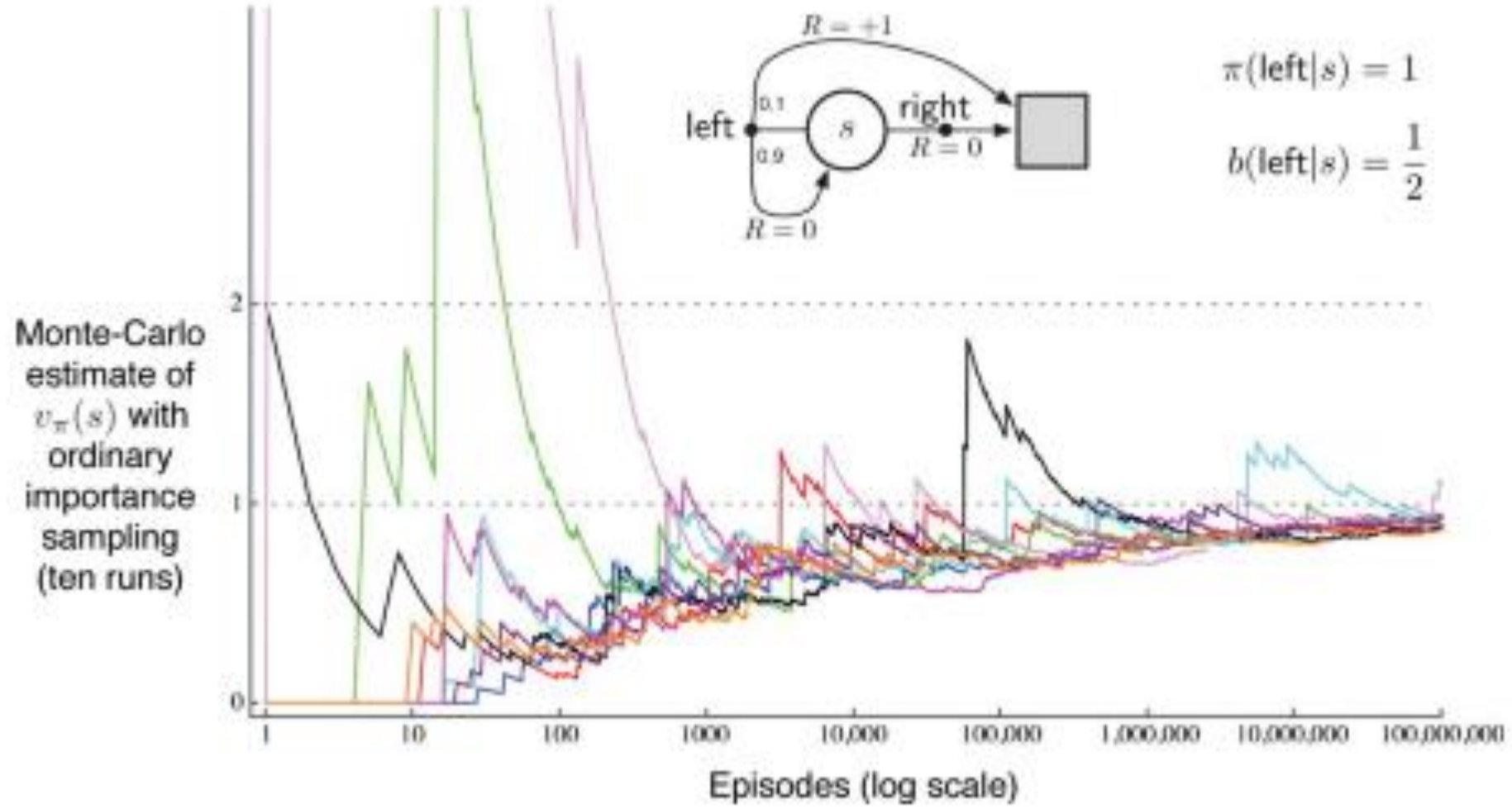
$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

Weighted Average

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$



Ordinary Importance Sampling is Unstable



Reference

- David Silver, Lecture 4: Model-Free Prediction
- Chapter 5, Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning: An Introduction,” 2nd edition, Nov. 2018