Carnegie Mellon
School of Computer Science

Monte Carlo Learning

Lecture 4, CMU 10-403

Katerina Fragkiadaki

Used Materials

- **Disclaimer**: Much of the material and slides for this lecture were borrowed from Rich Sutton's class and David Silver's class on Reinforcement Learning.

Summary so far

- So far, to estimate value functions we have been using dynamic programming with *known* rewards and dynamics functions

  Q: was our agent interacting with the world? Was our agent *learning* something?

$$v_{[k+1]}(s) = \sum_a \pi(a|s)\left(r(s,a) + \gamma \sum_{s'} p(s'|s,a) v_{[k]}(s') \right), \forall s$$

$$v_{[k+1]}(s) = \max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_{[k]}(s') \right), \forall s$$

Coming up

- So far, to estimate value functions we have been using dynamic programming with *known* rewards and dynamics functions

- Next: estimate value functions and policies from interaction experience, without known rewards or dynamics

How? With sampling all the way. Instead of probabilities distributions $p(s', r \mid s, a)$ to compute expectations, we will use empirical expectations by averaging sampled returns!

$$v_{[k+1]}(s) = \sum_a \pi(a \mid s) \left(r(s, a) + \gamma \sum_{s'} p(s' \mid s, a) v_{[k]}(s') \right), \forall s$$

$$v_{[k+1]}(s) = \max_{a \in \mathcal{A}} \left(r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) v_{[k]}(s') \right), \forall s$$

Monte Carlo (MC) Methods

- Monte Carlo methods are learning methods
  - Experience → values, policy
- Monte Carlo uses the simplest possible idea: value = mean return
- Monte Carlo methods learn from complete sampled trajectories and their returns
  - Only defined for episodic tasks
  - All episodes must terminate

Monte-Carlo Policy Evaluation

- Goal: learn $v_\pi(s)$ from episodes of experience under policy π

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

- Remember that the return is the total discounted reward:

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

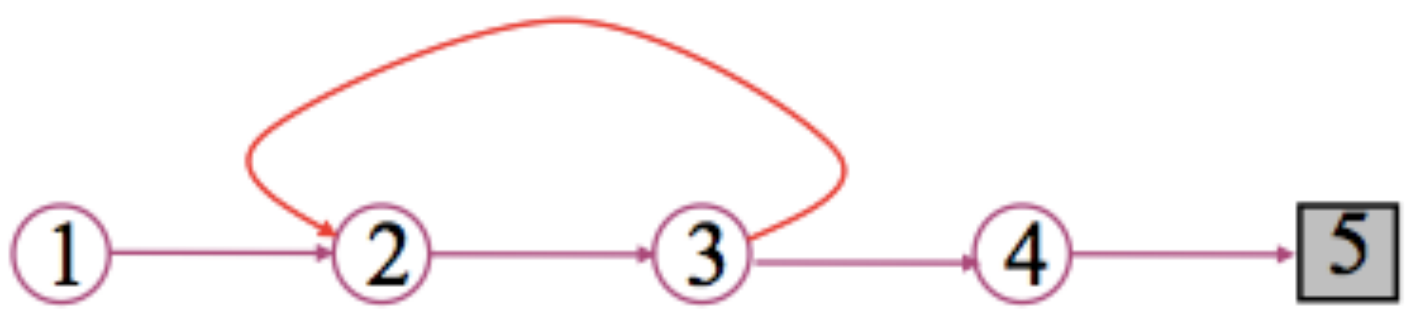
- Remember that the value function is the expected return:

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

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

Monte-Carlo Policy Evaluation

- Goal: learn $v_\pi(s)$ from episodes of experience under policy π
- Idea: Average returns observed after visits to s:



- Every-Visit MC: average returns for every time s is visited in an episode
- First-visit MC: average returns only for first time s is visited in an episode
- Both converge asymptotically

First-Visit MC Policy Evaluation

- To evaluate state s
- The first time-step t that state s is visited in an episode,
- Increment counter: $N(s) \leftarrow N(s) + 1$
- Increment total return: $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- By law of large numbers $V(s) \rightarrow v_\pi(s) \text{ as } N(s) \rightarrow \infty$

Law of large numbers

Every-Visit MC Policy Evaluation

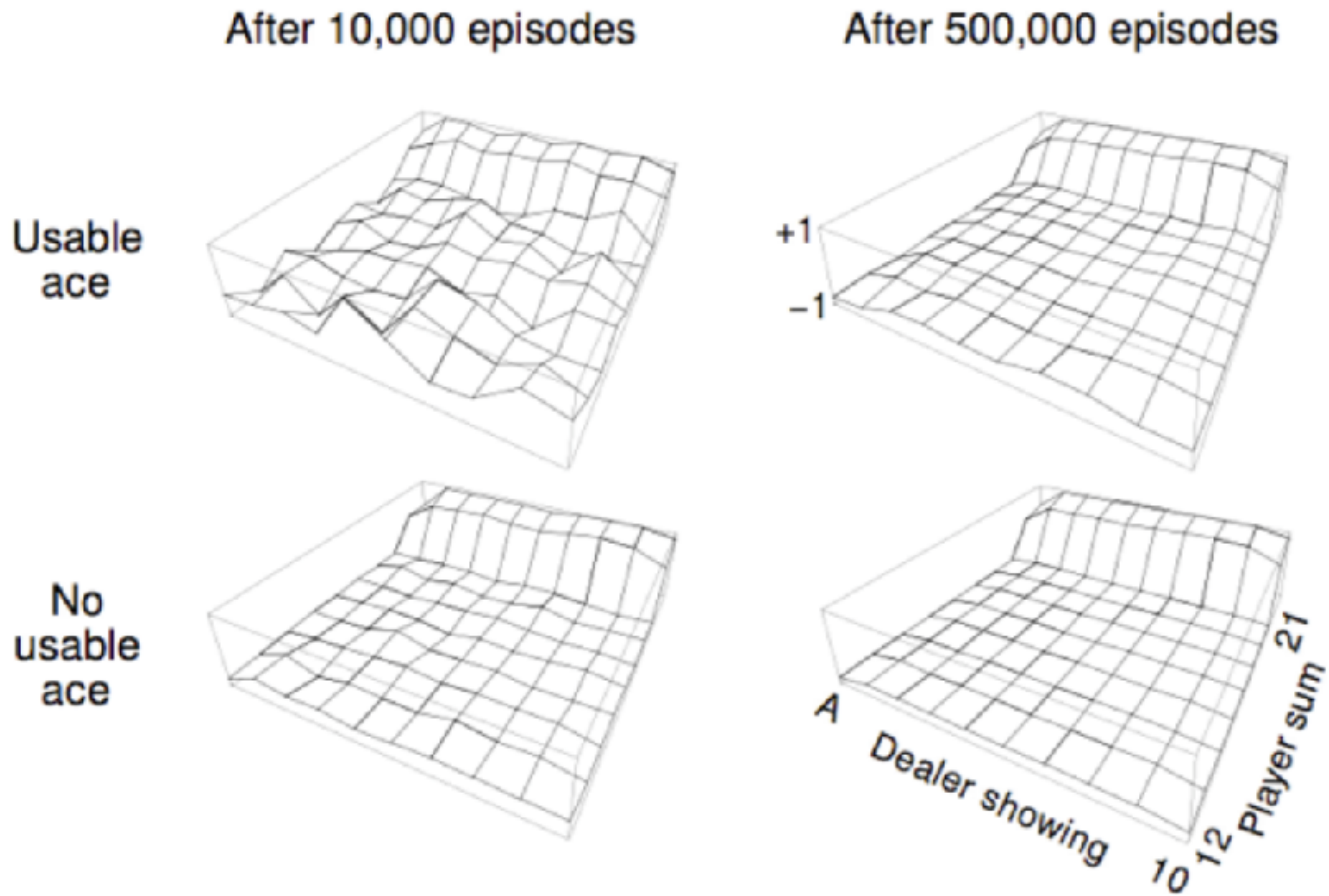
- To evaluate state s
- Every time-step t that state s is visited in an episode,
- Increment counter: $N(s) \leftarrow N(s) + 1$
- Increment total return: $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- By law of large numbers $V(s) \rightarrow v_\pi(s) \text{ as } N(s) \rightarrow \infty$

Blackjack Example

- Objective: Have your card sum be greater than the dealer's without exceeding 21.
- States (200 of them):
  - current sum (12-21)
  - dealer's showing card (ace-10)
  - do I have a useable ace?

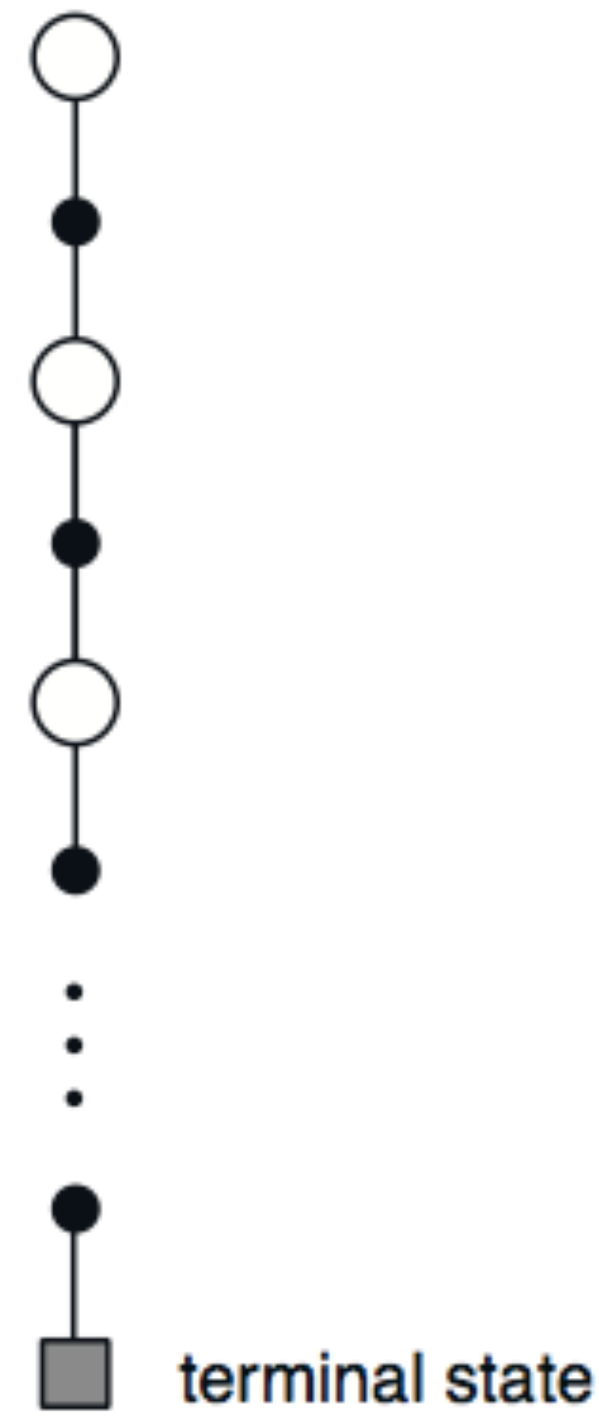
- Reward: +1 for winning, 0 for a draw, -1 for losing
- Actions: stick (stop receiving cards), hit (receive another card)
- Policy: Stick if my sum is 20 or 21, else hit
- No discounting (γ=1)

Learned Blackjack State-Value Functions



Backup Diagram for Monte Carlo

- Entire rest of episode included
- **Only one choice considered at each state (unlike DP)**
  - thus, there will be an explore/exploit dilemma
- Does not bootstrap from successor state's values (unlike DP)
- Value is estimated by mean return
- State value estimates are independent, no bootstrapping



Incremental Mean

- The mean μ_1, μ_2, ... of a sequence x_1, x_2, ... can be computed incrementally:

$$
\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}
$$

Incremental Monte Carlo Updates

- Update V(s) incrementally after episode $S_1, A_1, R_2, ..., S_T$
- For each state S_t with return G_t

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

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

- In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

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

MC Estimation of Action Values (Q)

- Monte Carlo (MC) is most useful when a model is not available
  - We want to learn q*(s,a)

- q_π(s,a) - average return starting from state s and action a following π

$$
\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big].
\end{aligned}
$$

- Converges asymptotically if every state-action pair is visited

Q:Is this possible if we are using a deterministic policy?

The Exploration problem

- If we always follow the deterministic policy we care about to collect experience, we will never have the opportunity to see and evaluate (estimate q) of alternative actions…

- Solutions:

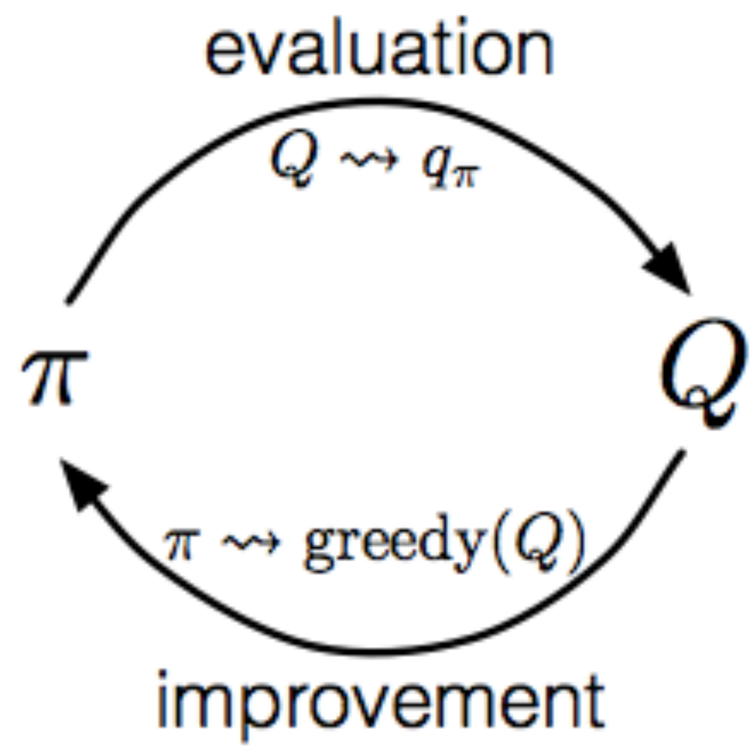
    1. exploring starts: Every state-action pair has a non-zero probability of being the starting pair

    2. Give up on deterministic policies and only search over \espilon-soft policies

    3. Off policy: use a different policy to collect experience than the one you care to evaluate

Monte-Carlo Control

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



- MC policy iteration step: Policy evaluation using MC methods followed by policy improvement

- Policy improvement step: greedify with respect to value (or action-value) function

Greedy Policy

- For any action-value function q, the corresponding greedy policy is the one that:
  - For each s, deterministically chooses an action with maximal action-value:

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

- Policy improvement then can be done by constructing each π_{k+1} as the greedy policy with respect to $q_{\pi k}$.

Convergence of MC Control

- Greedified policy meets the conditions for policy improvement:

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

- And thus must be $\geq \pi_{k.}$

- This assumes exploring starts and infinite number of episodes for MC policy evaluation

Monte Carlo Exploring Starts

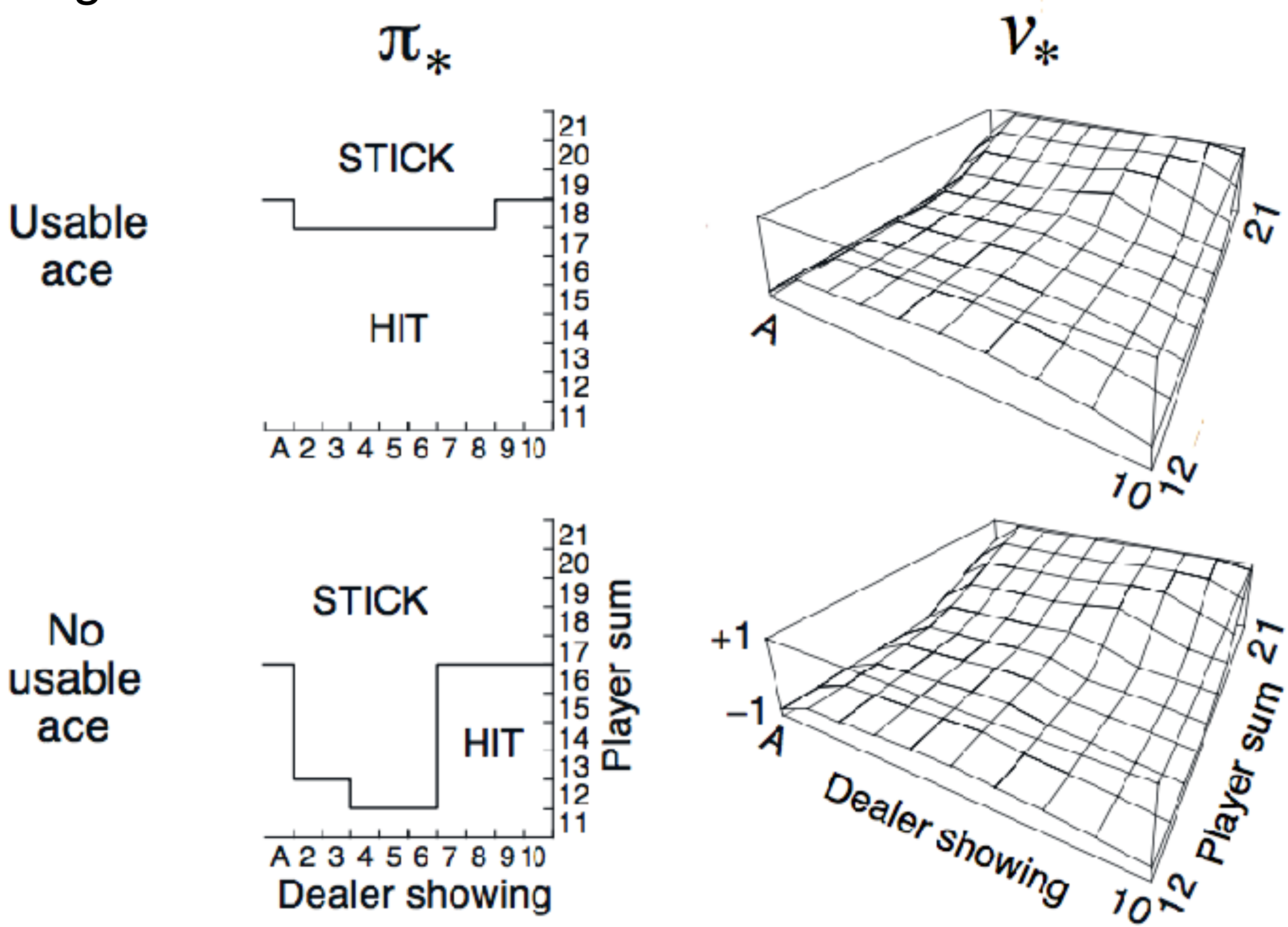
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \leftarrow$ arbitrary
  $\pi(s) \leftarrow$ arbitrary
  $Returns(s,a) \leftarrow$ empty list

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 $Returns(s,a)$
    $Q(s,a) \leftarrow \text{average}(Returns(s,a))$
  For each s in the episode:
    $\pi(s) \leftarrow \arg\max_a Q(s,a)$

Fixed point is optimal policy π^*

Blackjack example continued

- With exploring starts



On-policy Monte Carlo Control

- On-policy: learn about policy currently executing
- How do we get rid of exploring starts?
  - The policy must be eternally soft: $\pi(a|s) > 0$ for all s and a.
- For example, for ε-soft policy, probability of an action, $\pi(a|s)$,

$$= \underbrace{\frac{\epsilon}{|\mathcal{A}(s)|}}_{\text{non-max}} \quad \text{or} \quad \underbrace{1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}}_{\text{max (greedy)}}$$

- Similar to GPI: move policy towards greedy policy
- Converges to the best ε-soft policy.

On-policy Monte Carlo Control

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 $a \in \mathcal{A}(s)$:

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

Off-policy methods

- Learn the value of the target policy π from experience due to behavior policy μ.

- For example, π is the greedy policy (and ultimately the optimal policy) while μ is exploratory (e.g., ε-soft) policy

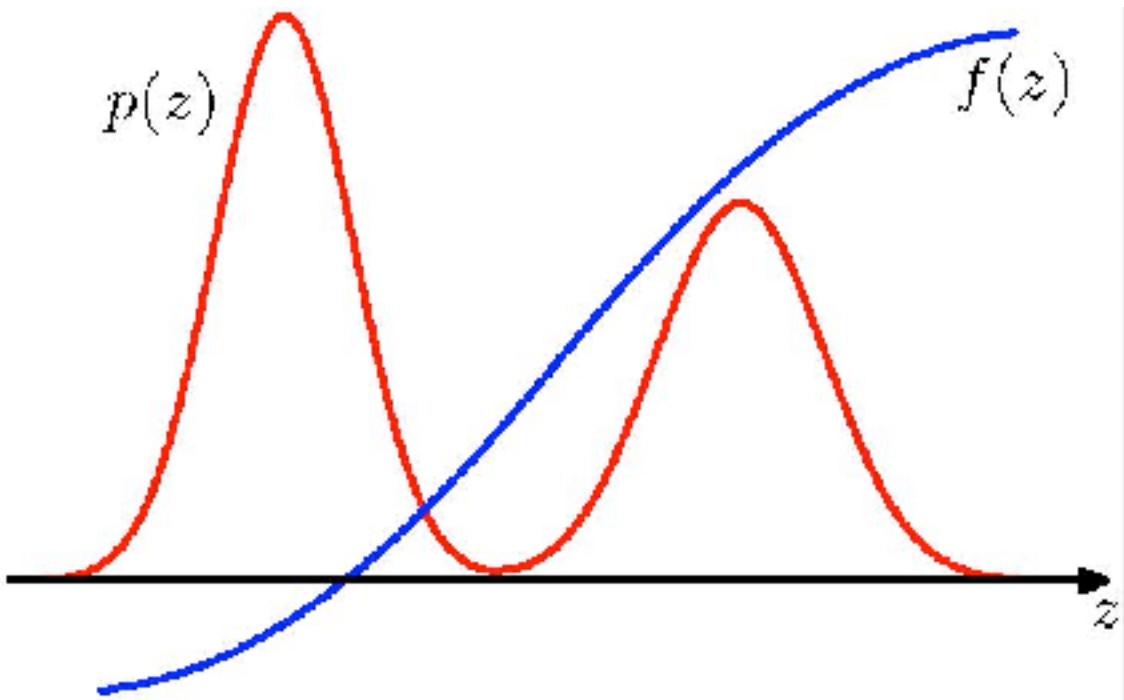
- In general, we only require coverage, i.e., that μ generates behavior that covers, or includes, π

$$\mu(a|s) > 0 \quad \text{for every } s,a \text{ at which} \quad \pi(a|s) > 0$$

- Idea: Importance Sampling:
  - Weight each return by the ratio of the probabilities of the trajectory under the two policies.

Simple Monte Carlo

• General Idea: Draw independent samples $\{z^1,..,z^n\}$ from distribution p(z) to approximate expectation:



$$\mathbb{E}[f] = \int f(z)p(z)dz \approx \frac{1}{N}\sum_{n=1}^{N} f(z^n) = \hat{f}.$$

Note that:

$$\mathbb{E}[f] = \mathbb{E}[\hat{f}].$$

so the estimator has correct mean (unbiased).

• The variance:

$$\mathrm{var}[\hat{f}] = \frac{1}{N}\mathbb{E}\left[(f - \mathbb{E}[f])^2\right].$$

• Variance decreases as 1/N.

• **Remark**: The accuracy of the estimator does not depend on dimensionality of z.

Importance Sampling

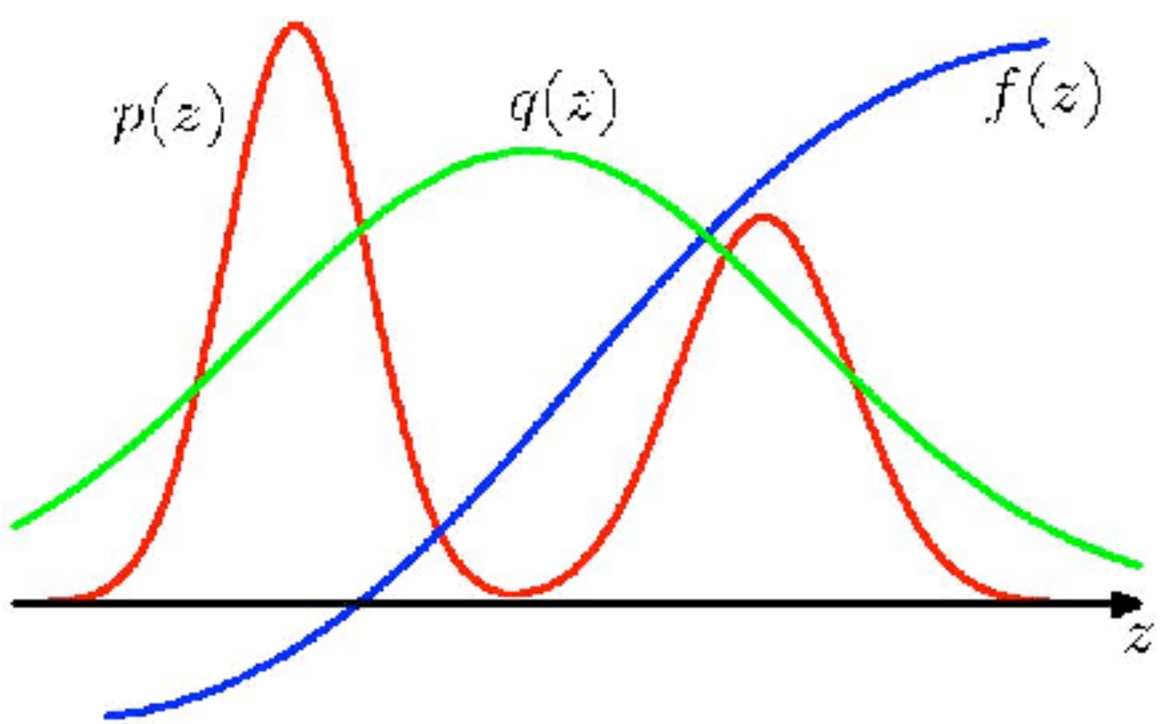
- Suppose we have an easy-to-sample proposal distribution q(z), such that

$$q(z) > 0 \quad \text{if} \quad p(z) > 0.$$

$$\mathbb{E}[f] = \int f(z)p(z)dz$$

$$= \int f(z)\frac{p(z)}{q(z)}q(z)dz$$

$$\approx \frac{1}{N}\sum_n \frac{p(z^n)}{q(z^n)}f(z^n), \quad z^n \sim q(z).$$



- The quantities

$$w^n = p(z^n)/q(z^n)$$

are known as **importance weights**.

This is useful when we can evaluate the probability p but is hard to sample from it

Importance Sampling Ratio

- Probability of the rest of the trajectory, after S_t, under policy π

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

- Importance Sampling: Each return is weighted by he relative probability of the trajectory under the target and behavior policies

$$
\rho_t^T = \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} \mu(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}
$$

- This is called the Importance Sampling Ratio

Importance Sampling

- Ordinary importance sampling forms estimate

First time of termination following time t

return after t up through T(t)

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_t^{T(t)} G_t}{|\mathcal{T}(s)|}.$$

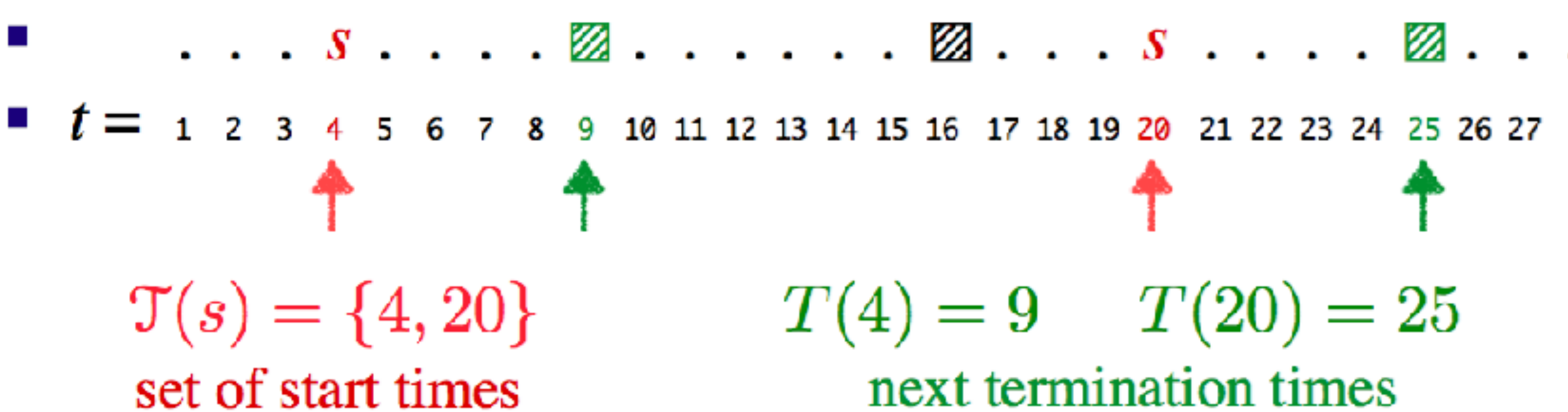
Every time: the set of all time steps in which state s is visited

Importance Sampling

- Ordinary importance sampling forms estimate

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_t^{T(t)} G_t}{|\mathcal{T}(s)|}.$$

- New notation: time steps increase across episode boundaries:



Importance Sampling

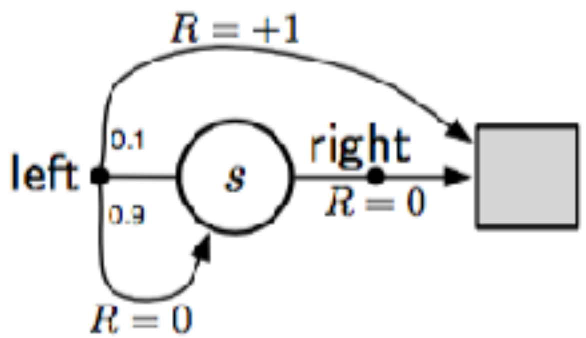
- Ordinary importance sampling forms estimate

$$V(s) \doteq \frac{\sum_{t \in \mathfrak{T}(s)} \rho_t^{T(t)} G_t}{|\mathfrak{T}(s)|}.$$

- Weighted importance sampling forms estimate:

$$V(s) \doteq \frac{\sum_{t \in \mathfrak{T}(s)} \rho_t^{T(t)} G_t}{\sum_{t \in \mathfrak{T}(s)} \rho_t^{T(t)}}$$

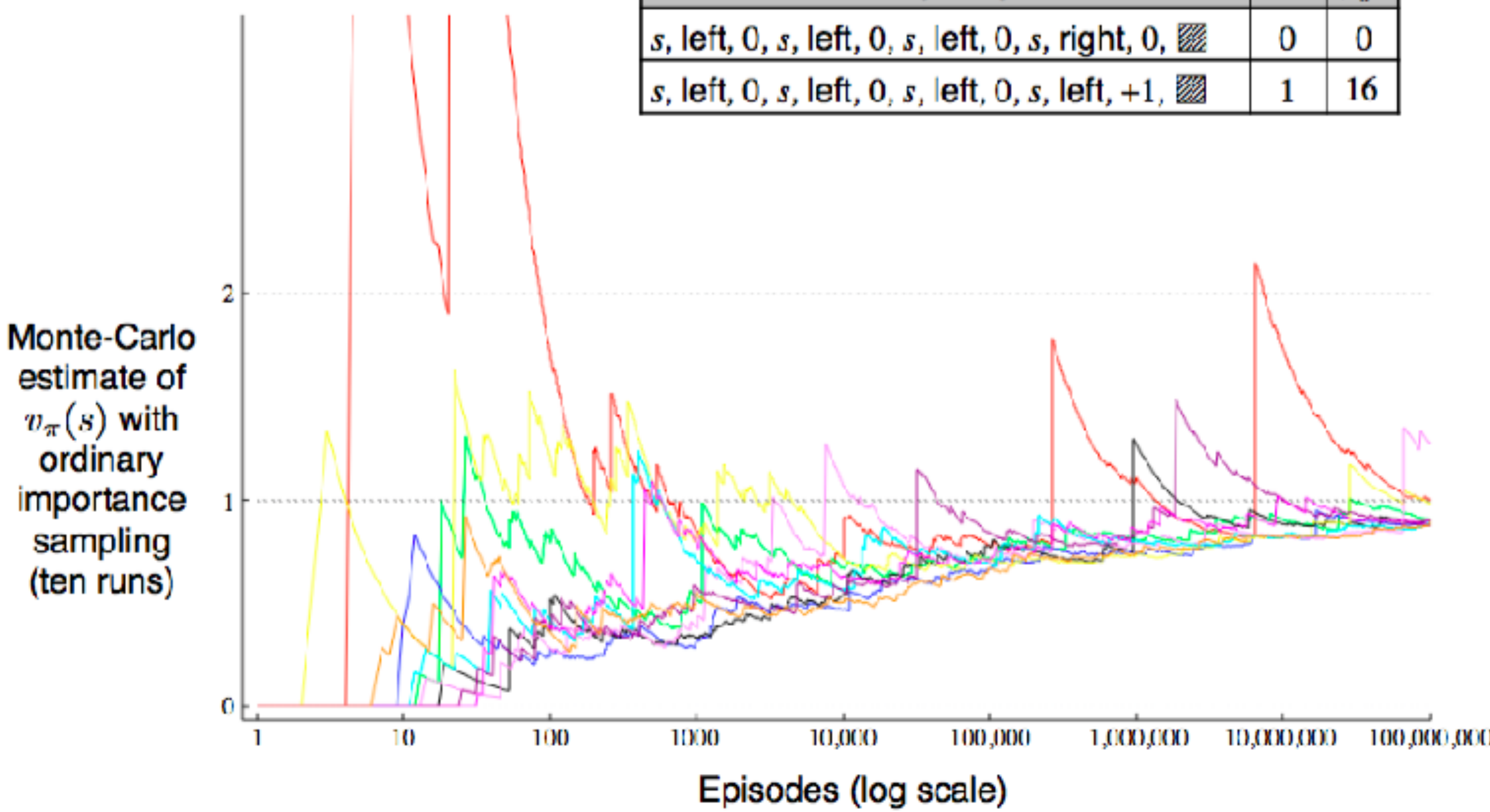
Example of Infinite Variance under Ordinary Importance Sampling



$\pi(\text{left}|s) = 1$ $\gamma = 1$ $\dfrac{\pi(\text{right}|s)}{\mu(\text{right}|s)} = 0$ $\dfrac{\pi(\text{left}|s)}{\mu(\text{left}|s)} = 2$

$\mu(\text{left}|s) = \dfrac{1}{2}$ $v_\pi(s) = 1$

Trajectory	G_0	ρ_0^T
s, left, 0, s, left, 0, s, left, 0, s, right, 0, ▨	0	0
s, left, 0, s, left, 0, s, left, 0, s, left, +1, ▨	1	16



OIS:

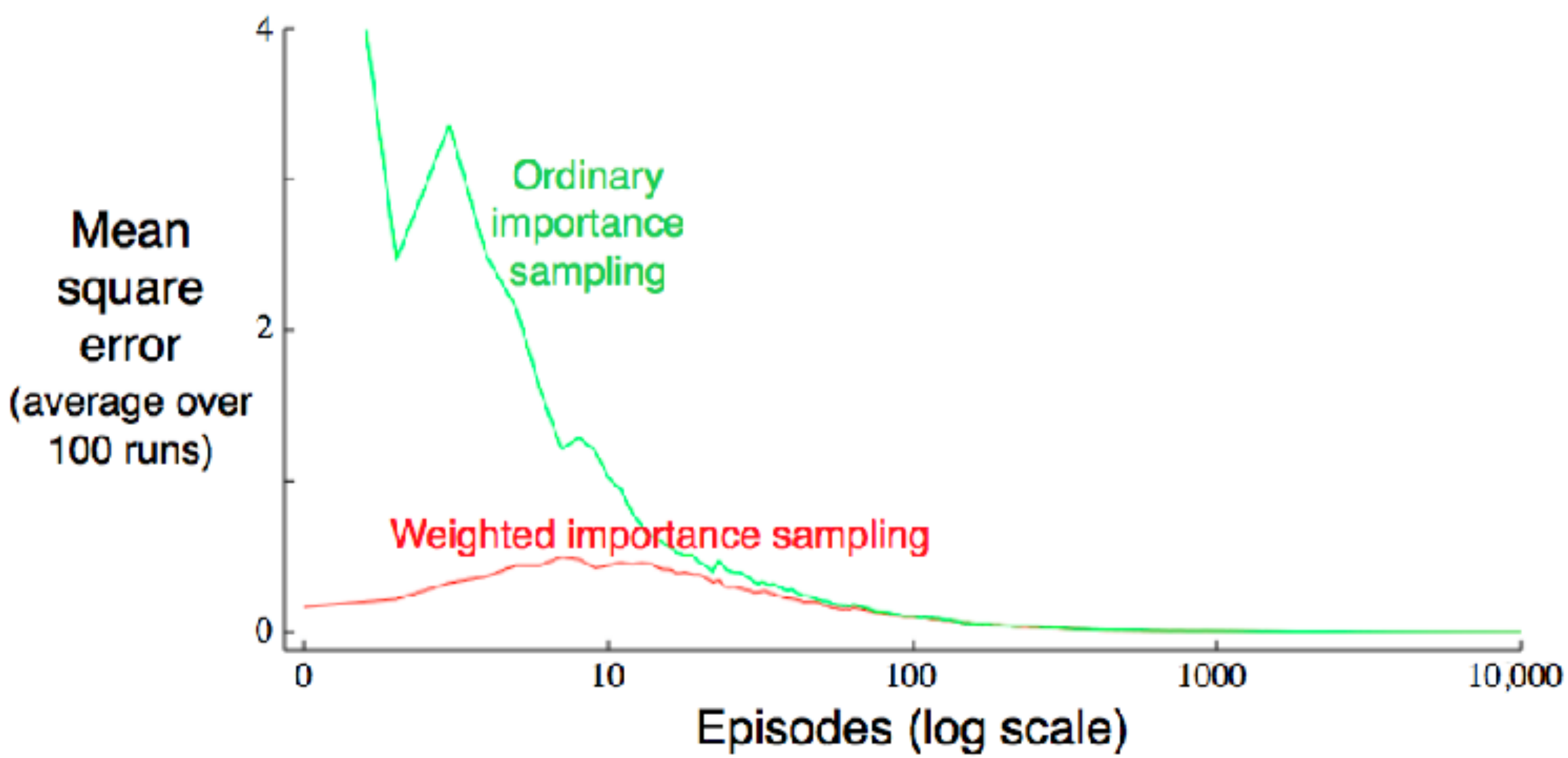
$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \rho_t^{T(t)} G_t}{|\mathcal{T}(s)|}$$

WIS:

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \rho_t^{T(t)} G_t}{\sum_{t\in\mathcal{T}(s)} \rho_t^{T(t)}}$$

Example: Off-policy Estimation of the Value of a Single Blackjack State

- State is player-sum 13, dealer-showing 2, useable ace
- Target policy is stick only on 20 or 21
- Behavior policy is equiprobable
- True value ≈ −0.27726



Incremental off-policy every-visit MC policy evaluation (returns $Q \approx q_\pi$

Input: an arbitrary target policy π

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$\quad Q(s,a) \leftarrow$ arbitrary

$\quad C(s,a) \leftarrow 0$

Repeat forever:

$\quad \mu \leftarrow$ any policy with coverage of π

$\quad$ Generate an episode using μ:

$\qquad S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$

$\quad G \leftarrow 0$

$\quad W \leftarrow 1$

$\quad$ For $t = T-1, T-2, \ldots$ downto 0:

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

$\qquad C(S_t, A_t) \leftarrow C(S_t, A_t) + W$

$\qquad Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t,A_t)} [G - Q(S_t, A_t)]$

$\qquad W \leftarrow W \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}$

$\qquad$ If $W = 0$ then ExitForLoop

$$\mu_k = \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$

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
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow \arg\max_a Q(S_t,a)$ (with ties broken consistently)

Repeat forever:

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

Target policy is greedy and deterministic

Behavior policy is soft, typically ε-greedy

Summary

- MC has several advantages over DP:
  - Can learn directly from interaction with environment
  - No need for full models
  - Less harmed by violating Markov property (later in class)
- MC methods provide an alternate policy evaluation process
- One issue to watch for: maintaining sufficient exploration
  - Can learn directly from interaction with environment
- Looked at distinction between on-policy and off-policy methods
- Looked at importance sampling for off-policy learning
- Looked at distinction between ordinary and weighted IS

Coming up next

- MC methods are different than Dynamic Programming in that they:

  1. use experience in place of known dynamics and reward functions

  2. do not bootrap

- Next lecture we will see temporal difference learning which

  3. use experience in place of known dynamics and reward functions

  4. bootrap!