

Monte Carlo Prediction & Control

CMPUT 261: Introduction to Artificial Intelligence

S&B §5.0-5.5, 5.7

Lecture Outline

1. Recap & Logistics
2. Monte Carlo Prediction
3. Estimating Action Values
4. Monte Carlo Control
5. Importance Sampling
6. Off-Policy Monte Carlo Control

After this lecture, you should be able to:

- explain how Monte Carlo estimation for state values works
- trace an execution of first-visit Monte Carlo Prediction
- explain the difference between prediction and control
- define on-policy vs. off-policy learning
- define a behaviour policy
- define exploring starts
- explain what problem exploring starts solve
- define an epsilon-soft policy
- explain what problem epsilon-soft policies solve

Logistics

- Late submissions for **Assignment #3** accepted until Friday at 11:59pm
- **Assignment #4** was released last night!
 - Due **December 7** at 11:59pm

Recap: In-Place Iterative Policy Evaluation

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π , the policy to be evaluated

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation

Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$

- These are **expected updates**: Based on a weighted average (expectation) of **all possible next states**

Recap: Policy Improvement Theorem

Theorem:

Let π and π' be any pair of deterministic policies.

If $q_{\pi}(s, \pi'(s)) \geq v_{\pi}(s) \quad \forall s \in \mathcal{S}$,

then $v_{\pi'}(s) \geq v_{\pi}(s) \quad \forall s \in \mathcal{S}$.

If you are never worse off **at any state** by following π' for **one step** and then following π forever after, then following π' **forever** has a higher expected value **at every state**.

Recap: Policy Iteration

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

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s)) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement

$policy-stable \leftarrow true$

For each $s \in \mathcal{S}$:

$old-action \leftarrow \pi(s)$

$\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$

If $old-action \neq \pi(s)$, then $policy-stable \leftarrow false$

If $policy-stable$, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Recap: Value Iteration

Value iteration interleaves the estimation and improvement steps:

$$\begin{aligned} v_{k+1}(s) &\doteq \max_a \mathbb{E} [R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma v_k(s')] \end{aligned}$$

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop:

```
|  $\Delta \leftarrow 0$   
| Loop for each  $s \in \mathcal{S}$ :  
|    $v \leftarrow V(s)$   
|    $V(s) \leftarrow \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma V(s')]$   
|    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$   
until  $\Delta < \theta$ 
```

Output a deterministic policy, $\pi \approx \pi_*$, such that

$$\pi(s) = \operatorname{argmax}_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma V(s')]$$

Example: Blackjack

- Player gets two cards, dealer gets 1
- Player can **hit** (get a new card) as many times as they like, or **stick** (stop hitting)
- After the player is done, the dealer hits / sticks according to a fixed rule
- Whoever has the most points (sum of card values) wins
- But, if you have more than 21 points, you **lose immediately** ("bust")

Simulating Blackjack

- Given a policy for the player, it is **very easy** to simulate a game of Blackjack
 - (Treat both the cards and the dealer as part of the environment)
- **Question:** Is it easy to **compute** the full **dynamics**?
- **Question:** Is it easy to run **iterative policy evaluation**?

Experience vs. Expectation

- In order to compute **expected updates**, we need to know the exact **probability** of **every** possible transition
- Often we don't have access to the full probability distribution, but we do have access to **samples of experience**
 1. **Actual experience:** We want to learn based on interactions with a **real environment**, without knowing its dynamics
 2. **Simulated experience:** We can **simulate** the dynamics, but we don't have an **explicit representation** of transition probabilities, or there are **too many states**

Monte Carlo Estimation

- Instead of estimating expectations by a **weighted sum** over **all possibilities**, estimate expectation by **averaging** over a **sample** drawn from the distribution:

$$\mathbb{E}[X] = \sum_x f(x)x \approx \frac{1}{n} \sum_{i=1}^n x_i \quad \text{where } x_i \sim f$$

Monte Carlo Prediction

- Use a **large sample** of **episodes** generated by a policy π to estimate the state-values $v_\pi(s)$ for each state s
 - We will consider only **episodic** tasks for now
- **Question:** What is the **return** G_t for state $S_t = s$ in a given episode?
- We can estimate the expected return $v_\pi(s) = \mathbb{E}[G_t \mid S_t = s]$ by averaging the returns for that state in every episode containing a visit to s

First-visit Monte Carlo Prediction

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 : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

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

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

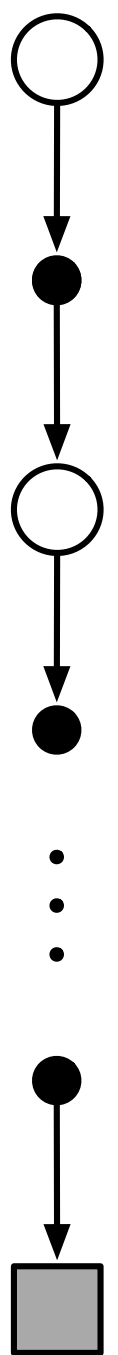
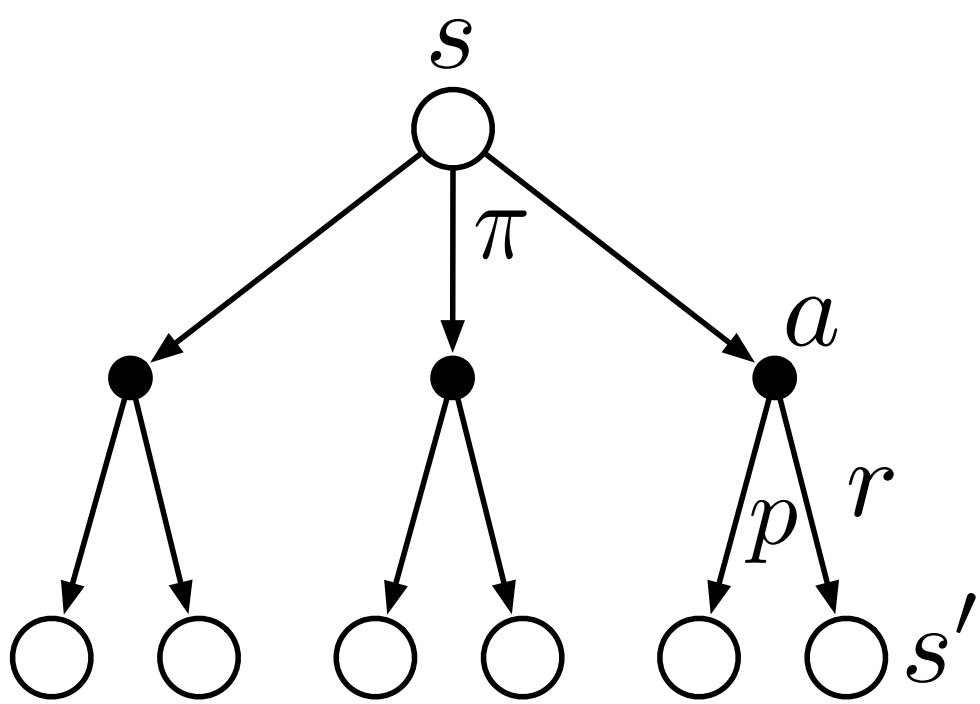
Unless S_t appears in $S_0, S_1, \dots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Monte Carlo vs. Dynamic Programming

- **Iterative policy evaluation** uses the estimates of the **next state's** value to update the value of this state
 - Only needs to compute a **single transition** to update a state's estimate
- **Monte Carlo** estimate of each state's value is **independent** from estimates of **other states'** values
 - Needs the **entire episode** to compute an update
 - Can focus on evaluating a **subset of states** if desired



Control vs. Prediction

- **Prediction:** estimate the value of states and/or actions given some **fixed policy** π
- **Control:** estimate an **optimal policy**

Estimating Action Values

- When we know the **dynamics** $p(s', r \mid s, a)$, an estimate of **state values** is sufficient to determine a good **policy**:
 - Choose the action that gives the best combination of **reward** and **next-state value**:

$$\hat{a}^* = \arg \max_{a \in \mathcal{A}} \sum_{s', r} p(s', r \mid s, a) [\textcolor{red}{r} + \gamma \hat{v}(s')]$$

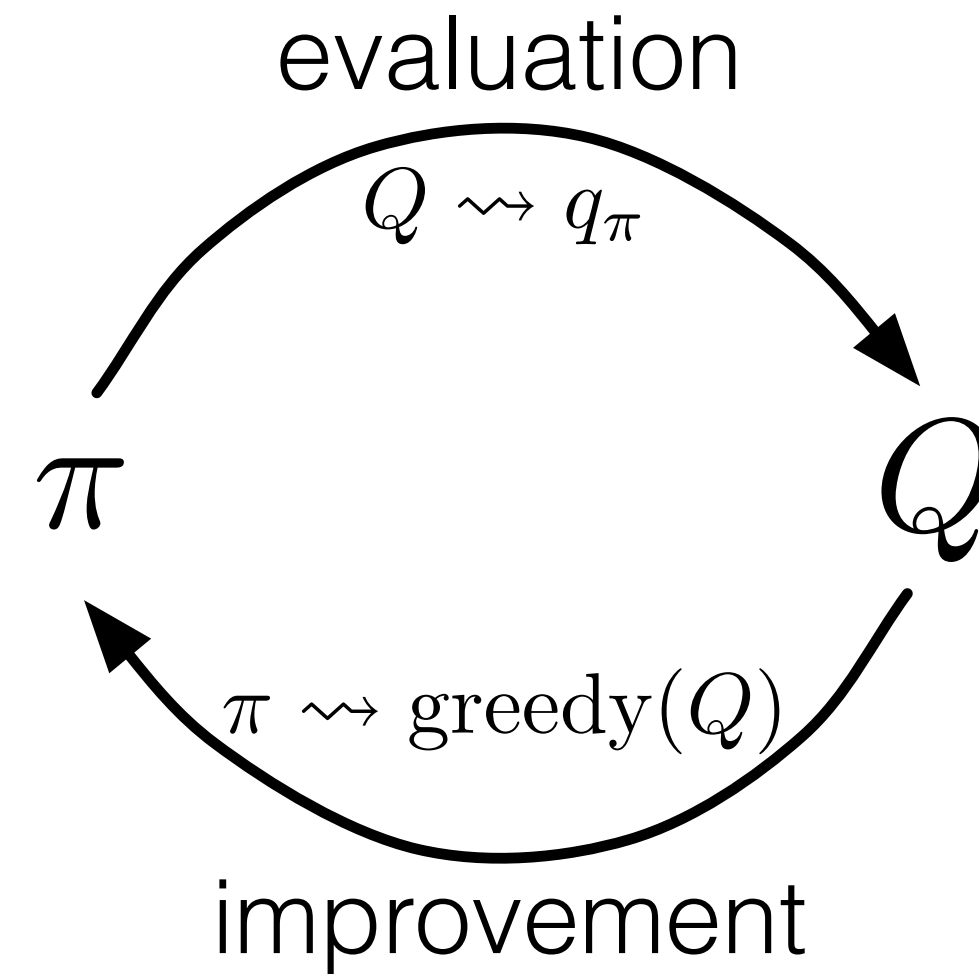
- This is why Value Iteration only explicitly estimates state values, not policy
- If we *don't* know the dynamics, state values are **not enough**
 - To estimate a good policy, we need an **explicit** estimate of **action values**

Exploring Starts for Action-Value Estimation

- We can just run first-visit Monte Carlo and approximate the returns to each **state-action pair**
- **Question:** What do we do about state-action pairs that are **never visited**?
 - If the current policy π never selects an action a from a state s , then Monte Carlo can't estimate its value
- **Exploring starts assumption:**
 - Every episode **starts** at a random state-action pair S_0, A_0
 - **Every pair** has a positive probability of being selected for a start

Monte Carlo Control

Monte Carlo control can be used for **policy iteration**:



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

Monte Carlo Control with Exploring Starts

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

Initialize:

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

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

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

Loop forever (for each episode):

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

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

$G \leftarrow 0$

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

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

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

Append G to $Returns(S_t, A_t)$

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

$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$

Question: What **unlikely assumptions** does this rely upon?

ϵ -Soft Policies

- The **exploring starts** assumption requires that we see **every** state-action pair with positive probability
 - Even if π **never** chooses a from state s
- Another approach: Simply **force** π to (sometimes) choose a !
- An **ϵ -soft policy** is one for which $\pi(a \mid s) \geq \frac{\epsilon}{|\mathcal{A}(s)|} \quad \forall s, a$
- **Example: ϵ -greedy policy**

$$\pi(a \mid s) = \begin{cases} \frac{\epsilon}{|\mathcal{A}(s)|} & \text{if } a \notin \arg \max_a Q(s, a), \\ (1 - \epsilon) + \frac{\epsilon}{|\mathcal{A}(s)|} & \text{otherwise.} \end{cases}$$

Monte Carlo Control w/out Exploring Starts

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 $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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

Repeat forever (for each episode):

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

$G \leftarrow 0$

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

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

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

Append G to $Returns(S_t, A_t)$

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

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

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

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

Monte Carlo Control w/out Exploring Starts

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 $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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

Repeat forever (for each episode):

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

$G \leftarrow 0$

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

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

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

Append G to $Returns(S_t, A_t)$

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

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

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

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

Question:

Will this procedure converge to the **optimal** policy π^* ?

Why or why not?

Importance Sampling

- **Monte Carlo sampling:** use samples from the **target** distribution to estimate expectations
- **Importance sampling:** Use samples from **proposal** distribution to estimate expectations of **target** distribution by **reweighting** samples

$$\mathbb{E}[X] = \sum_x f(x)x = \sum_x \frac{g(x)}{g(x)} f(x)x = \sum_x g(x) \frac{f(x)}{g(x)} x \approx \frac{1}{n} \sum_{x_i \sim g} \boxed{\frac{f(x_i)}{g(x_i)}} x_i$$

↑
Importance sampling
ratio

Off-Policy Prediction via Importance Sampling

Definition:

Off-policy learning means using data generated by a **behaviour policy** to learn about a distinct **target policy**.

← Target
distribution

Proposal
distribution ↗

Off-Policy Monte Carlo Prediction

- Generate episodes using **behaviour policy** b
- Take **weighted** average of returns to state s over all the episodes containing a visit to s to estimate $v_\pi(s)$
 - Weighed by **importance sampling ratio** of trajectory starting from $S_t = s$ until the end of the episode:

$$\rho_{t:T-1} \doteq \frac{\Pr[A_t, S_{t+1}, \dots, S_T \mid S_t, A_{t:T-1} \sim \pi]}{\Pr[A_t, S_{t+1}, \dots, S_T \mid S_t, A_{t:T-1} \sim b]}$$

Importance Sampling Ratios for Trajectories

- **Probability of a trajectory** $A_t, S_{t+1}, A_{t+1}, \dots, S_T$ from S_t :

$$\Pr[A_t, S_{t+1}, \dots, S_T | S_t, A_{t:T-1} \sim \pi] = \\ \pi(A_t | S_t) p(S_{t+1} | S_t, A_t) \pi(A_{t+1} | S_{t+1}) \dots p(S_T | S_{T-1}, A_{T-1})$$

- **Importance sampling ratio** for a trajectory $A_t, S_{t+1}, A_{t+1}, \dots, S_T$ from S_t :

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

$= 1$

Ordinary vs. Weighted Importance Sampling

- **Ordinary importance sampling:**

$$V(s) \doteq \frac{1}{n} \sum_{i=1}^n \rho_{t(s,i):T(i)-1} G_{i,t}$$

- **Weighted importance sampling:**

$$V(s) \doteq \frac{\sum_{i=1}^n \rho_{t(s,i):T(i)-1} G_{i,t}}{\sum_{i=1}^n \rho_{t(s,i):T(i)-1}}$$

Example: Ordinary vs. Weighted Importance Sampling for Blackjack

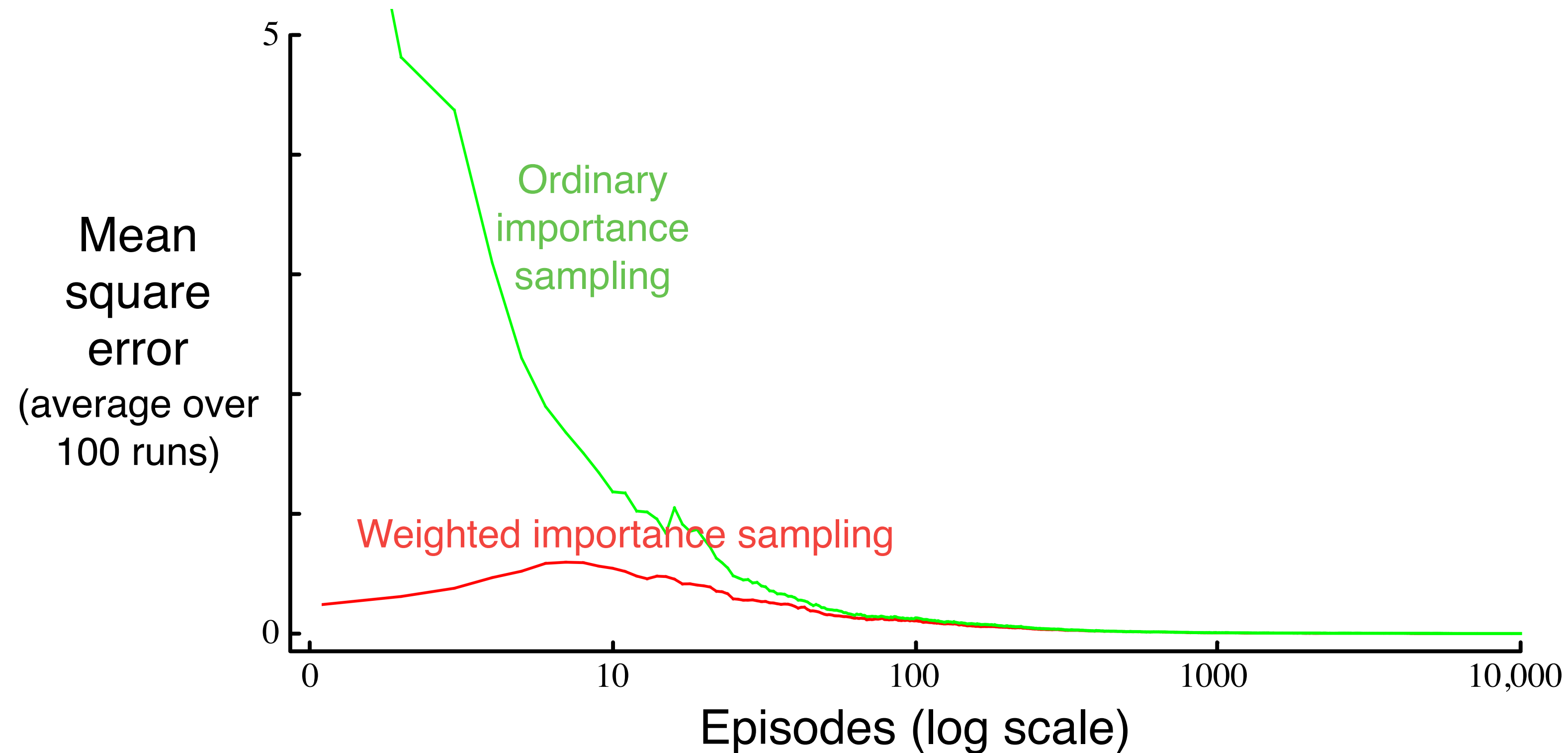


Figure 5.3: Weighted importance sampling produces lower error estimates of the value of a single blackjack state from off-policy episodes. ■

Off-Policy Monte Carlo **Prediction**

Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π

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

$Q(s, a) \in \mathbb{R}$ (arbitrarily)

$C(s, a) \leftarrow 0$

Loop forever (for each episode):

$b \leftarrow$ any policy with coverage of π

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

$G \leftarrow 0$

$W \leftarrow 1$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$, while $W \neq 0$:

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

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

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

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

Off-Policy Monte Carlo **Control**

Off-policy MC control, for estimating $\pi \approx \pi_*$

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

$Q(s, a) \in \mathbb{R}$ (arbitrarily)

$C(s, a) \leftarrow 0$

$\pi(s) \leftarrow \operatorname{argmax}_a Q(s, a)$ (with ties broken consistently)

Loop forever (for each episode):

$b \leftarrow$ any soft policy

Generate an episode using b : $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

$W \leftarrow 1$

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

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

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

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

$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$ (with ties broken consistently)

If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)

$W \leftarrow W \frac{1}{b(A_t|S_t)}$

Off-Policy Monte Carlo **Control**

Off-policy MC control, for estimating $\pi \approx \pi_*$

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

$Q(s, a) \in \mathbb{R}$ (arbitrarily)

$C(s, a) \leftarrow 0$

$\pi(s) \leftarrow \operatorname{argmax}_a Q(s, a)$ (w

Loop forever (for each episode):

$b \leftarrow$ any soft policy

Generate an episode using b :

$G \leftarrow 0$

$W \leftarrow 1$

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

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

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

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

$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$ (with ties broken consistently)

If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)

$W \leftarrow W \frac{1}{b(A_t | S_t)}$

$$\begin{aligned} Q_n &= \frac{\sum_{i=1}^n W_i G_i}{\sum_{i=1}^n W_i} = \frac{\sum_{i=1}^n W_i G_i}{C - W} \\ Q_{n+1} &= \frac{\sum_{i=1}^{n+1} W_i G_i}{\sum_{i=1}^{n+1} W_i} = \frac{(C - W)Q_n + WG}{C} \\ &= \frac{C}{C}Q_n - \frac{W}{C}Q_n + \frac{W}{C}G = Q_n + \frac{W}{C} [G - Q_n] \end{aligned}$$

Questions:

1. Will this procedure converge to the **optimal** policy π^* ?
2. Why do we break when $A_t \neq \pi(S_t)$?
3. Why do the weights W not involve $\pi(A_t | S_t)$?

Summary

- **Monte Carlo estimation** estimates values by averaging returns over **sample episodes**
 - Does not require access to full model of **dynamics**
 - Does require access to an entire **episode** for each sample
- Estimating **action values** requires either **exploring starts** or a **soft policy** (e.g., ϵ -greedy)
- **Off-policy learning** is the estimation of value functions for a **target policy** based on episodes generated by a different **behaviour policy**
 - **Importance sampling** is one way to perform off-policy learning
 - **Weighted** importance sampling has lower **variance** than **ordinary** importance sampling
- **Off-policy control** is learning the **optimal policy** (target policy) using episodes from a **behaviour policy**