

Advanced Topics in Reinforcement Learning

Lecture 5: Monte Carlo methods
(And finish dynamic programming)

Josiah Hanna
University of Wisconsin — Madison

Announcements

- Homework released. Due: October 8 at midnight.
- Start reading chapter 6 for next week.
- Project proposals due: Thursday, October 1st.

Course Overview

- So far we've seen:
 - Learning in a simplified setting (k-armed bandits).
 - Formalized reinforcement learning problems (MDPs).
 - Exact solution methods for MDPs (dynamic programming methods).
- Today: first learning methods for MDPs.
- Next week: learning methods that bootstrap like dynamic programming methods.

Learning Outcomes

After this week, you will be able to:

1. Differentiate between value function computation and learning.
2. Describe and implement approaches to estimating value functions from sampled experience in an MDP.
3. Learn optimal policies from sampled experience.
4. Differentiate between on- and off-policy learning.

Finishing Dynamic Programming in RL

- Goal: use value functions to find improved policies.
- Method: turn Bellman equations into value function updates.
- Bellman equation for policy value becomes policy evaluation:

$$v_{k+1}(s) \leftarrow \sum_a \pi(a | s) \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_k(s')]$$

- Bellman optimality equation becomes value iteration:

$$v_{k+1}(s) \leftarrow \max_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_k(s')]$$

Policy Iteration

- We have $v_\pi(s)$ for the current policy π . How can we improve π ?
- Alternate:
 - Run policy evaluation updates to find v_π .

$$\text{Set } \pi'(s) \leftarrow \arg \max_a \underbrace{\sum_{s',r} p(s', r | s, a) [r + \gamma v_\pi(s')]}_{q_\pi(s,a)}$$

- Why does this work?

Policy Improvement Theorem

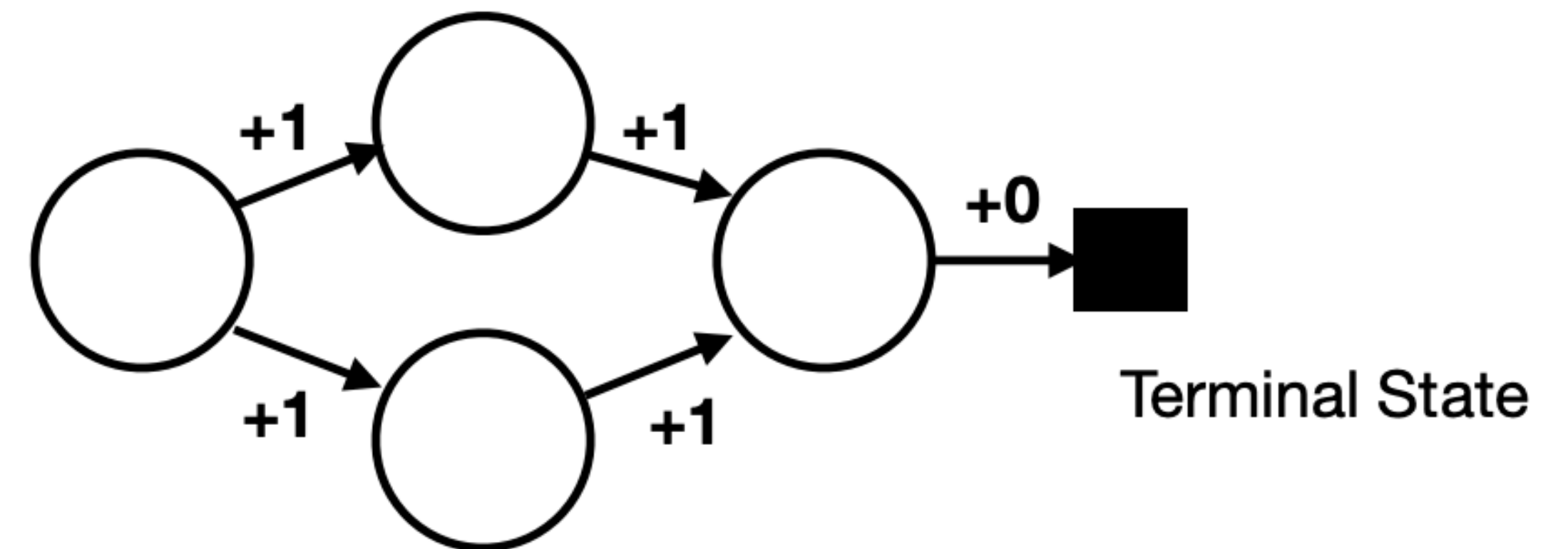
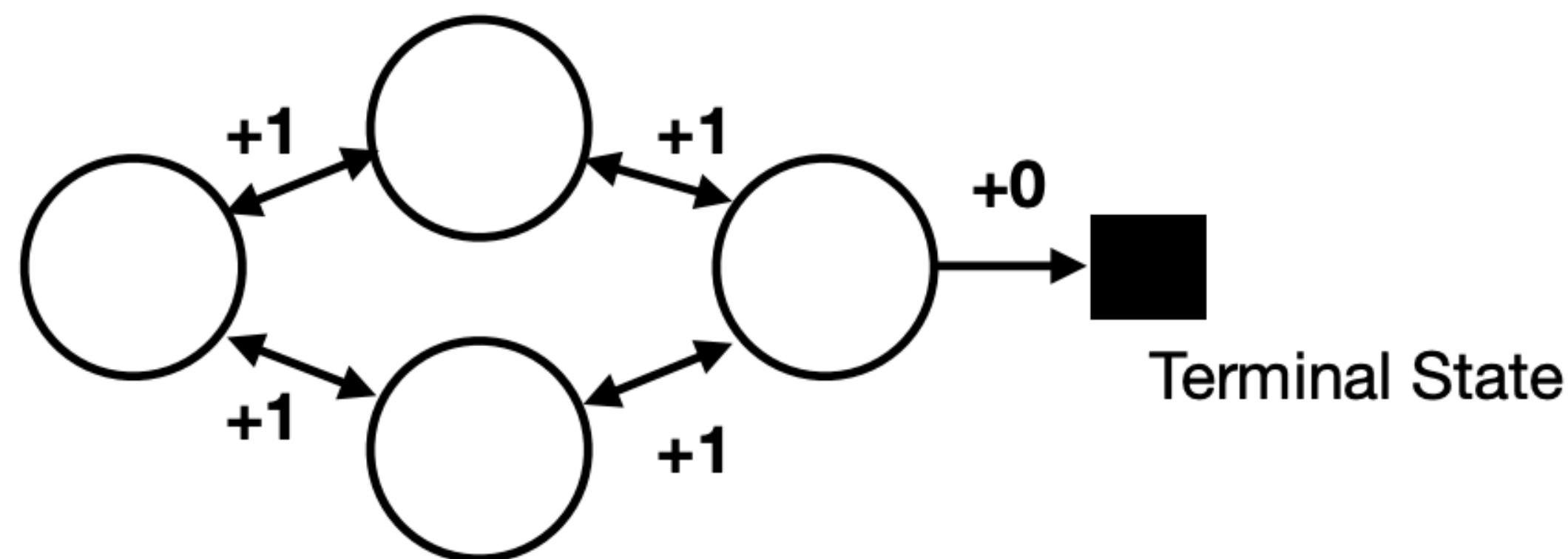
- Suppose for π that $\exists s, a$ such that $q_\pi(s, a) \geq v_\pi(s)$.
- Let $\pi'(s) = a$ and $\pi'(\tilde{s}) = \pi(\tilde{s})$ for all other states $\tilde{s}$.
- What is true about π' ? Why?
 - As good as or better than π , i.e., $v_{\pi'}(s) \geq v_\pi(s), \forall s$
- If π is sub-optimal, does there exist s, a such that $q_\pi(s, a) \geq v_\pi(s)$?
 - Yes, this follows from Bellman Optimality. Must be at least one state where π is not greedy w.r.t. its action-value function.
 - Optimal value function: $v_\star(s) = \max_a q_\star(s, a) \forall s$

Policy Iteration Demo

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

Asynchronous DP

- Basic DP methods require a sweep over the entire state space *per iteration*.
- Infeasible with a huge state space.
- Actually unnecessary — can update states in any order and still converge as long as all states are updated infinitely often in the limit.
- Why does this help?



Generalized Policy Iteration

- What is it?
 - We can be quite permissive in how we mix evaluation and improvement.
 - As long as q becomes closer to q_π and π becomes greedy w.r.t. q we will converge to $q_\star, \pi^\star$.
- A general framework for all algorithms we will introduce in this class.
- Do you think this holds when q_π must generalize across states? I.e., increasing the value of $q_\pi(s, a)$ will also increase the value of $q_\pi(s', a')$ for s', a' close to s .

Markov Reward Process

- Like MDPs but no actions; Markov chains with rewards
 - $p(s', r | s)$ is the state transition and reward distribution function.
- A fixed policy induces a Markov reward process on the state space:

$$p(s', r | s) = \sum_a \pi(a | s) p(s', r | s, a).$$

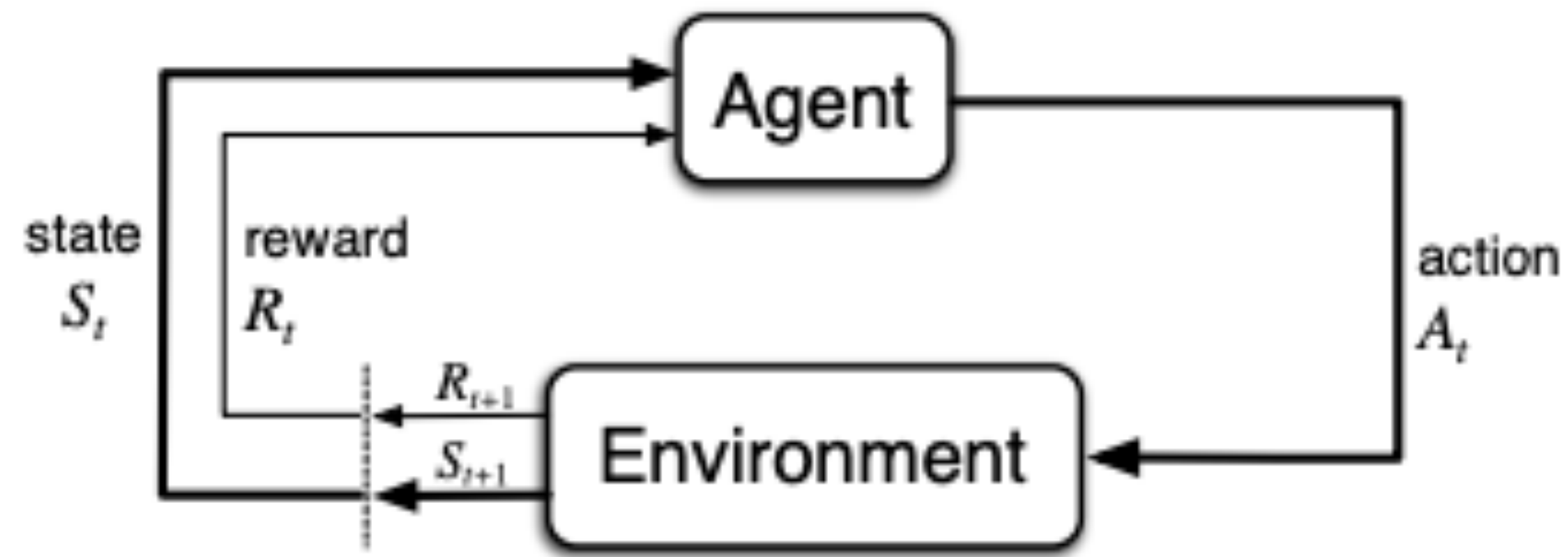
State values for this MRP are same
as $v_\pi(s)$ in MDP

- A fixed policy induces a Markov reward process on the state-action space:

$$p((s', a'), r | (s, a)) = \pi(a' | s') p(s', r | s, a)$$

State values for this MRP are same
as $q_\pi(s, a)$ in MDP

Markov Decision Processes



$$\dots S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, \dots$$

$$S_{t+1}, R_{t+1} \sim p(\cdot | S_t, A_t)$$

$$A_{t+1} \leftarrow \pi(S_{t+1})$$

We observe the trajectory but don't know p .

Learn / estimate value functions vs. compute value functions

Statistics Review

- We have random variable $X \sim d$ and use X as an estimate of unknown value μ . The expected value of X is $E_d[X]$.

- **Variance** of X :

$$\text{Var}_d[X] = \mathbf{E}_d[(X - \mathbf{E}_d[X])^2]$$

- **Bias** of X :

$$\text{Bias}_d[X] = \mu - \mathbf{E}_d[X]$$

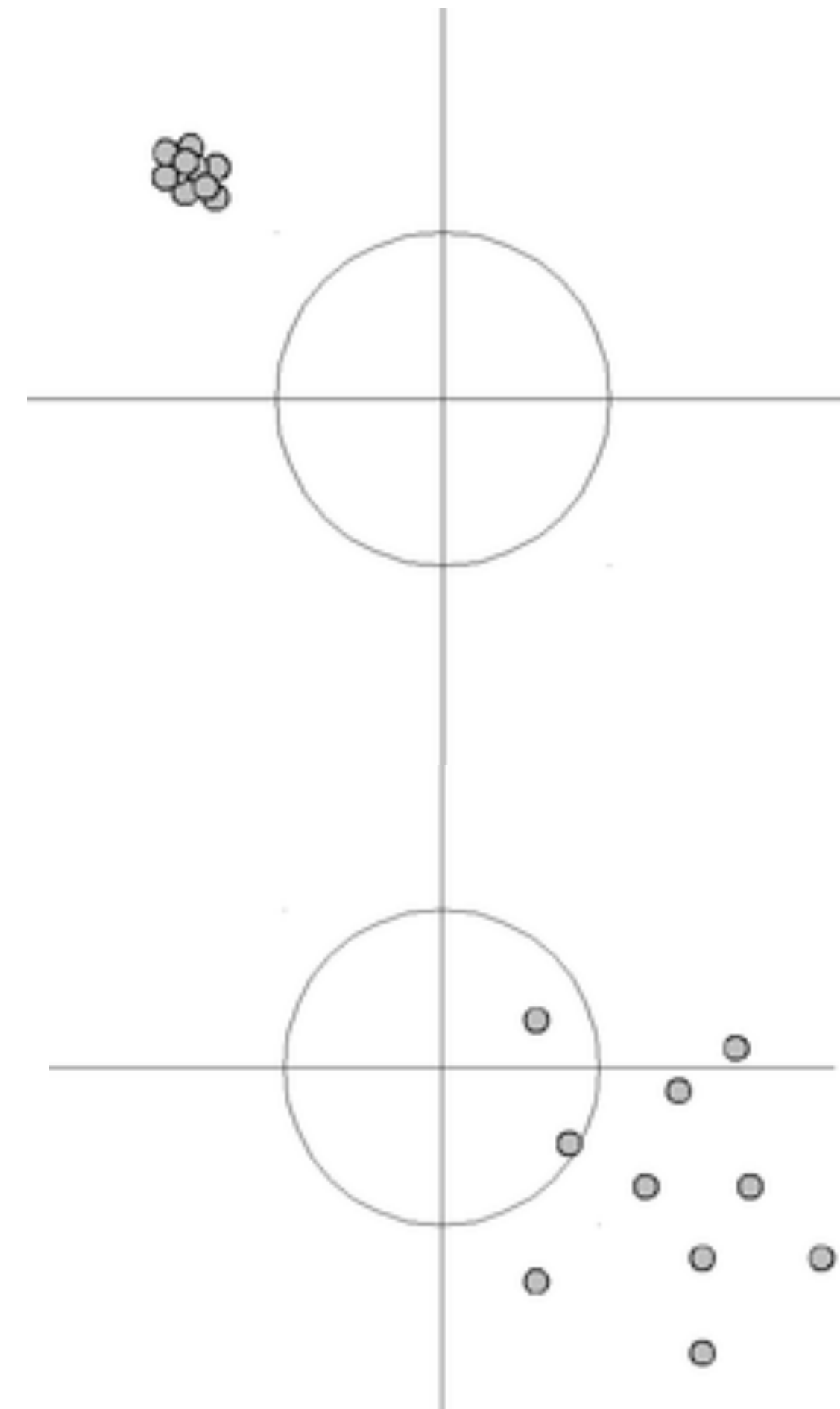
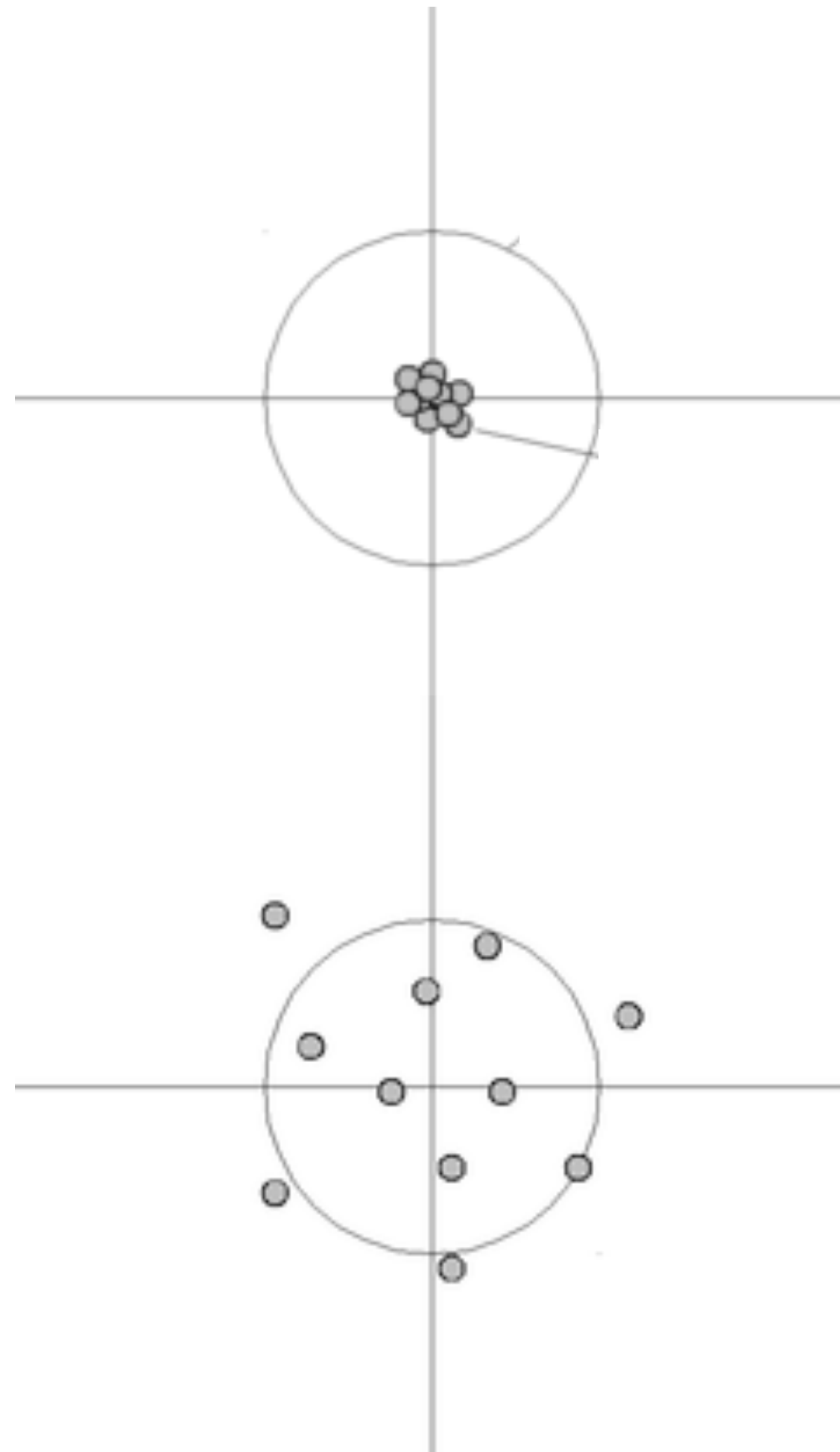
- An estimate is a **consistent** estimator of an unknown value if it converges (probabilistically) to the value being estimated.

Bias / Variance

Low Bias

High Bias

Low Variance



High Variance

Monte Carlo Methods

- Random variable $X \sim d$ and real-valued function $f(X)$, estimate:

$$\mathbf{E}_d[f(X)] = \sum_x d(x)f(x)$$

- The distribution d is unknown but we can sample $X \sim d$.
- Monte Carlo approximation:

$$\sum_x d(x)f(x) \approx \frac{1}{n} \sum_{i=1}^n f(X_i) \qquad X_i \sim d$$

- The law of large numbers tells us that as $n \rightarrow \infty$ that error in the approximation goes to zero.
- Error is order $1/\sqrt{n}$.

Monte Carlo in RL

- Given a policy, compute its state- or action-value function.

$$q_{\pi}(s, a) = \mathbf{E}_{\pi} \left[\sum_{t=0}^T \gamma^t R_{t+1} \mid S_t = s, A_t = a \right]$$

- X is a trajectory $S_0, A_0, R_1, S_1, A_1, \dots, R_T, S_T$ generated by following π starting with (s, a) .
- d is a probability distribution over trajectories that is induced from MDP and π .

$$\Pr(s_0, a_0, r_1, s_1, \dots, r_T, s_T) = \prod_{t=0}^{T-1} \pi(a_t \mid s_t) p(s_{t+1}, r_{t+1} \mid s_t, a_t)$$

- $f(X)$ is the sum of discounted rewards along trajectory X : $\sum_{t=0}^T \gamma^t R_{t+1}$

Single State First-Visit Monte Carlo

How would you change
for state-values?

- Estimate $q_\pi(s_0, a_0)$ for a fixed state, s_0, a_0 .
- Assume we always start in state s_0 and all episodes eventually terminate.
- To evaluate policy π , set `total` $\leftarrow 0$, and repeat n times:

- Start at s_0 , take action a_0 .

- Until termination: $S_t, R_t \sim p(S', R | S_{t-1}, A_{t-1}), A_t \sim \pi(A | S_t)$.

- $\text{total} \leftarrow \text{total} + \sum_{t=0} \gamma^t R_{t+1}$.

- Return $Q_n(s_0, a_0) \leftarrow \text{total}/n$
- As $n \rightarrow \infty, Q_n(s_0, a_0) \rightarrow q_\pi(s_0, a_0)$.

What is storage requirement for first-
visit Monte Carlo?

On-Policy First-Visit Monte Carlo

- To evaluate policy π , set $\text{returns}(s, a) \leftarrow \{\}$, and repeat n times:

- Sample $S_0 \sim d_{\text{initial}}$

- Until termination: $S_t, R_t \sim p(S', R | S_{t-1}, A_{t-1}), A_t \sim \pi(A | S_t)$.

- For first visit to (s, a) in episode (at timestep i):

- $\text{returns}(s, a) \leftarrow \text{returns}(s, a) \cup \left\{ \sum_{t=i}^T \gamma^t R_{t+1} \right\}.$

- Return $Q_n(s_0, a_0) \leftarrow \text{mean}(\text{returns})$

- As $n \rightarrow \infty, Q_n(s_0, a_0) \rightarrow q_\pi(s_0, a_0)$.

Every-Visit Monte Carlo

- In general, we may see the same state multiple times per-episode.
- How does every-visit Monte Carlo differ from first-visit Monte Carlo?
 - Uses return following each occurrence of a state-action pair.
 - May converge faster depending on number of extra occurrences.
- Does every-visit Monte Carlo give unbiased estimates of values?
 - No, but will converge in the limit (statistically consistent).

Monte Carlo or Dynamic Programming?

- When would you prefer Monte Carlo methods?
 - No model of the environment or simulation-only model.
 - No Markov state.
- When would you prefer dynamic programming methods?
 - No episode termination.
 - Model known, small number of Markov states and actions.

Monte Carlo for Control

- Either first-visit or every-visit Monte Carlo can estimate v_π or q_π from experience generated by following policy π . What else is needed for control?
- Must estimate action-values (not state-values). Why?
 - With state-values, the best action is: $a^\star = \arg \max \sum_{s', r} p(s', r | s, a) [r + \gamma v_\star(s')]$
 - One-step search requires model to be known.
- Must see all states and actions but π may only select a single action in any given state.
 - Need exploration!

Exploring Starts

- Simple idea to provide exploration.
- How does it work?
 - Non-zero probability of starting in any state and then taking a random action.
- Is it practical?
 - Depends.
 - Inapplicable to continuing problems or problems where we do not control the initial state distribution.
 - Is applicable and potentially beneficial when we DO control the initial state distribution.

Monte Carlo Policy Iteration

- To find $\pi^\star$, start with arbitrary π_0 , and alternate:
 - Run Monte Carlo policy evaluation of π_k for n episodes.
 - Make π_{k+1} the greedy policy w.r.t. q_k .
- How large must n be?
- Exploring starts ensures convergence only if all returns averaged come from same policy. Why?
 - Conjectured that there is no need to discard returns as policy changes but no formal proof.

Ensuring Exploration

- Exploring starts are restrictive. What else to do?
 - ϵ -greedy policies: select $a^\star = \arg \max_a q(s, a)$ with probability $1 - \epsilon$; else random action.
 - Hard policy $\equiv$ Deterministic policy, Soft policy $\equiv$ Stochastic policy.
- Do ϵ -greedy methods converge? If so, to what?
- Can we still reach $\pi^\star$?
 - What if we decay epsilon?

Off-Policy Motivation

- What is the difference between off-policy and on-policy learning?
 - Trajectories generated by behavior policy, used to evaluate target policy.
 - If behavior = target ($\forall s, a, \pi(a | s) = b(a | s)$), then on-policy. Otherwise, off-policy.
- Why do we need off-policy learning?
 - Behavior policy explores, target policy exploits.
 - Learn for many reward functions at the same time.
 - Behavior policy is a known and safe policy.
- What is the main challenge in off-policy learning?
 - Distribution shift! Behavior policy and target policy induce different trajectory distributions. Thus, $q_b(s, a) \neq q_\pi(s, a)$ in general.

Importance Sampling Methods

- Given distribution $d(X)$ and real-valued function $f(X)$, estimate:

$$\mathbf{E}_d[f(X)] = \sum_x d(x)f(x)$$

- The distribution d is unknown but we can sample $X \sim b$.

- Monte Carlo approximation:

$$\sum_x d(x)f(x) = \sum_x b(x) \frac{d(x)}{b(x)} f(x) \approx \frac{1}{n} \sum_{i=1}^n \frac{d(X_i)}{b(X_i)} f(X_i) \quad X_i \sim b$$

If we set $b \leftarrow d$ then reduces to standard Monte Carlo.

- Law of large numbers tells us that as $n \rightarrow \infty$ that error in the approximation goes to zero.

- Error is order $1/\sqrt{n}$ (assuming $\frac{d(x)}{b(x)}$ is bounded).

Off-Policy Monte Carlo in RL

- Key idea: correct return distribution with importance sampling.
- Trajectory distribution that is induced from MDP and behavior policy.

$$\Pr(s_0, a_0, r_1, s_1, \dots, r_T, s_T) = \prod_{t=0}^{T-1} b(a_t | s_t) p(s_{t+1}, r_{t+1} | s_t, a_t)$$

- Desired trajectory distribution induced from MDP and target policy.

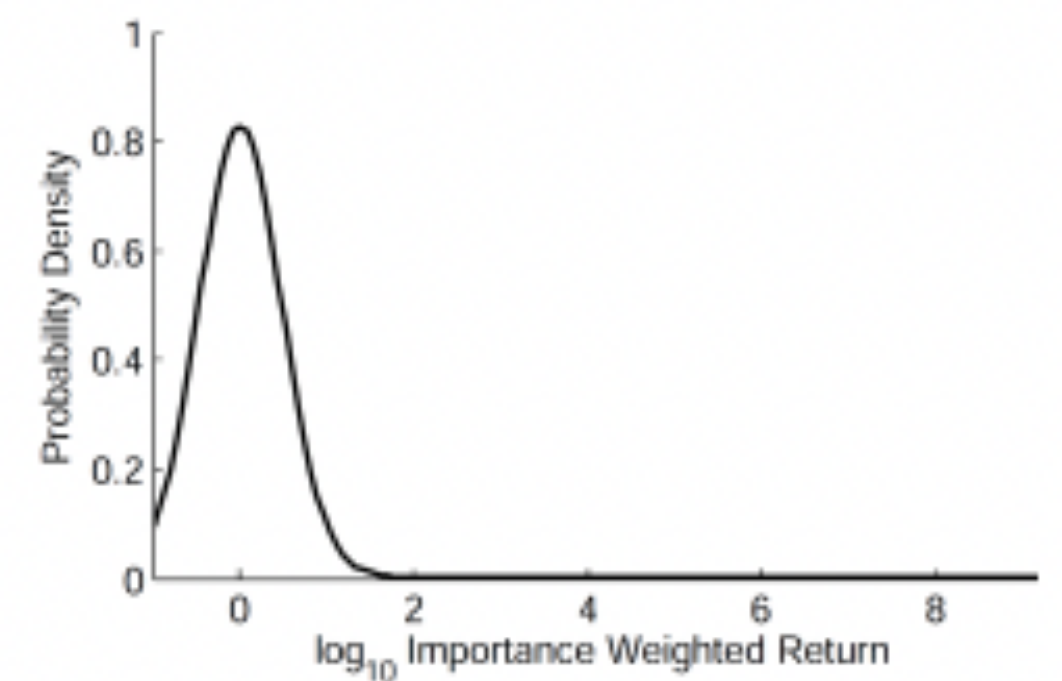
$$\Pr(s_0, a_0, r_1, s_1, \dots, r_T, s_T) = \prod_{t=0}^{T-1} \pi(a_t | s_t) p(s_{t+1}, r_{t+1} | s_t, a_t)$$

- Importance weighted returns:

$$\rho_{t:T} := \prod_{i=t}^{T-1} \frac{\pi(a_i | s_i) p(s_{i+1}, r_{i+1} | s_i, a_i)}{b(a_i | s_i) p(s_{i+1}, r_{i+1} | s_i, a_i)} = \prod_{i=t}^{T-1} \frac{\pi(a_i | s_i)}{b(a_i | s_i)} \quad v_{\pi}(s_t) \approx \rho_{t:T} G_t$$

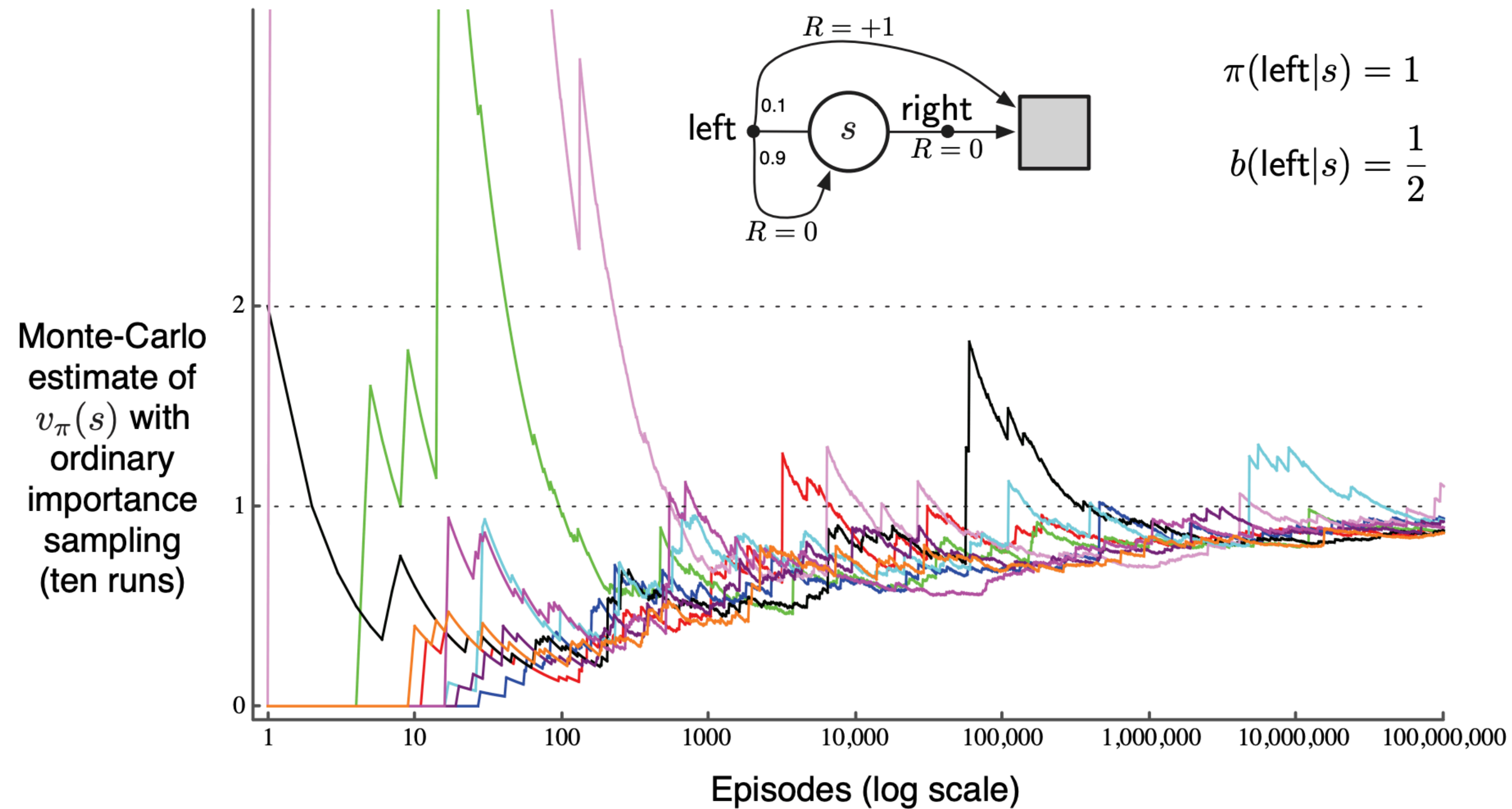
Importance Sampling Variance

- Importance sampling provides unbiased estimates of $v_\pi(s)$ using returns sampled by running the behavior policy.
 - Assuming that, if $\pi(a | s) > 0$, then $b(a | s) > 0$.
- In practice:
 - Can have infinite variance.
 - Most of the time, importance sampling severely under-estimates and then rarely, massively over-estimates.
 - Can return implausible estimates.
 - Ex: Suppose you *know* G_t is bounded and hence $v_\pi(s)$ is bounded. Importance sampling may estimate a value much greater than the bound.



Thomas et al. 2015

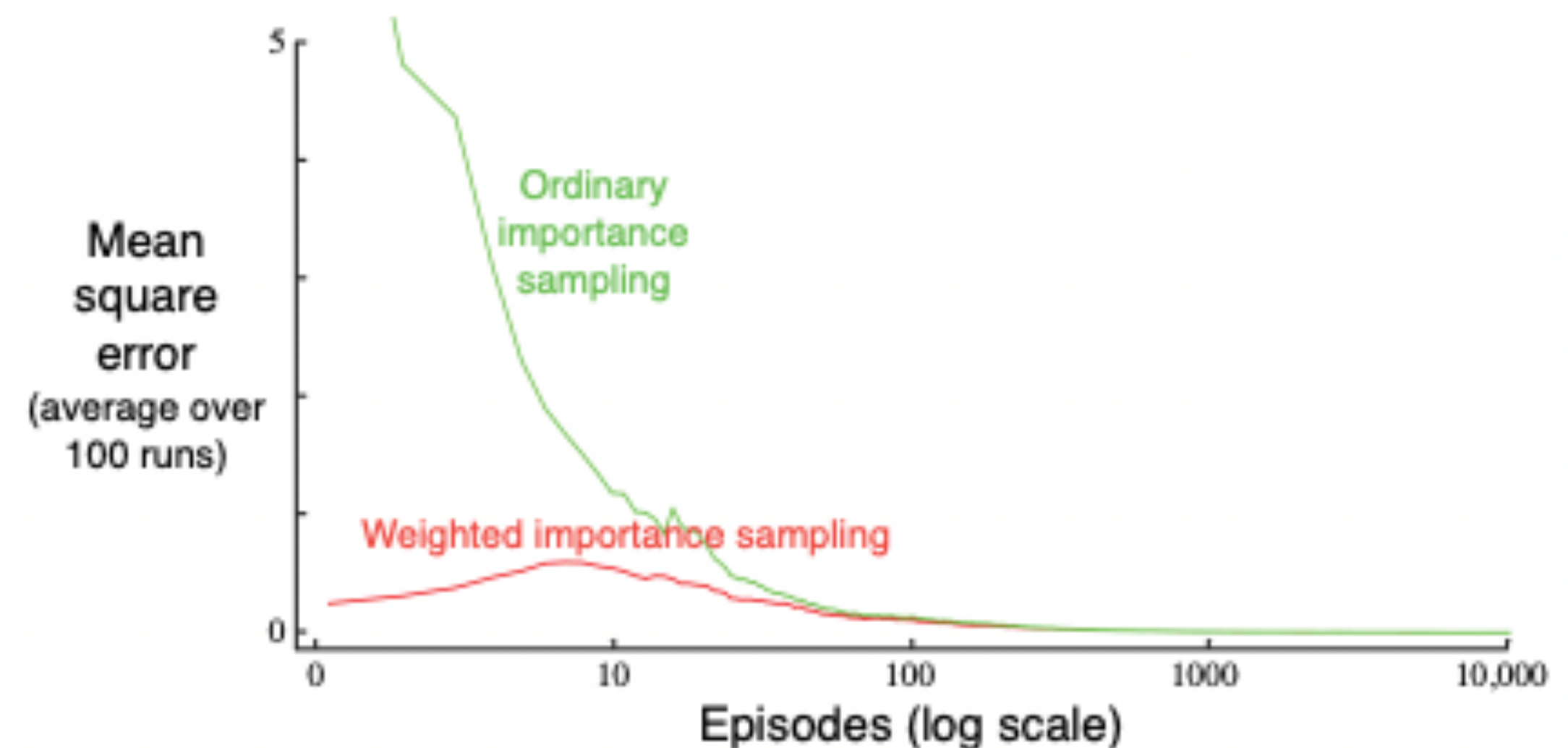
Importance Sampling Variance



Weighted Importance Sampling

- Estimation error = Variance + Bias². Often a trade-off: can reduce variance by introducing bias.
- Weighted Importance Sampling introduces bias but can drastically lower variance.

$$V(s) := \frac{\sum_{t \in T(s)} \rho_{t:T} G_t}{\sum_{t \in T(s)} \rho_{t:T}}$$



Per-Decision Importance Sampling

- Ordinary importance sampling re-weights all rewards the same:

$$\rho_{t:T}G_t = \rho_{t:T}(R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-t-1}R_T)$$

- Actions that follow a reward do not affect the likelihood of that reward.

$$\rho_{t:T}\gamma^k R_{t+k+1} = \rho_{1:k} \cdot \rho_{k+1:T}\gamma^k R_{t+k+1}$$

- Per-decision importance sampling takes advantage of this by dropping factors in the importance ratios.

$$\mathbf{E}_b[\rho_{t:T}\gamma^k R_{t+k+1}] = \mathbf{E}_b[\rho_{1:k}\gamma^k R_{t+k+1}]$$

Off-Policy Control

- With off-policy prediction, we can run a *soft* behavior policy to provide exploration while improving the target policy greedily.
 - Behavior policy must ensure state-action coverage.
 - Ex: Behavior policy is ϵ -greedy and target policy is greedy.
- Still follow general policy iteration scheme:
 - Evaluate target policy (i.e., estimate q_π) with off-policy Monte Carlo.
 - Make target policy greedy w.r.t. q_π .
 - Converges to $\pi^\star$.

- Is this efficient?

$$\rho_{t:T} := \prod_{i=t}^{T-1} \frac{\pi(a_i | s_i) p(s_{i+1}, r_{i+1} | s_i, a_i)}{b(a_i | s_i) p(s_{i+1}, r_{i+1} | s_i, a_i)} = \prod_{i=t}^{T-1} \frac{\pi(a_i | s_i)}{b(a_i | s_i)}$$

Off-Policy First-Visit Monte Carlo

- To evaluate policy π , set $\text{returns}(s, a) \leftarrow \{\}$, and repeat n times:
 - Sample $S_0 \sim d_{\text{initial}}$
 - Until termination: $S_t, R_t \sim p(S', R | S_{t-1}, A_{t-1})$ $A_t \sim b(A | S_t)$.
 - For first visit to (s, a) in episode (at timestep i):

$$\bullet \text{ returns}(s, a) \leftarrow \text{returns}(s, a) \cup \left\{ \rho_{i:T} \sum_{t=i}^T \gamma^t R_{t+1} \right\}.$$

- Return $Q_n(s_0, a_0) \leftarrow \text{mean}(\text{returns})$
- As $n \rightarrow \infty$, $Q_n(s_0, a_0) \rightarrow q_\pi(s_0, a_0)$.

$$\rho_{t:T} := \prod_{i=t}^{T-1} \frac{\pi(a_i | s_i) p(s_{i+1}, r_{i+1} | s_i, a_i)}{b(a_i | s_i) p(s_{i+1}, r_{i+1} | s_i, a_i)} = \prod_{i=t}^{T-1} \frac{\pi(a_i | s_i)}{b(a_i | s_i)}$$

How to use IS in practice

- Clip or bound weights, i.e., $\rho \leftarrow \min(\frac{\pi(a | s)}{b(a | s)}, 1)$.
- Restrict policy difference.
- Baselines and doubly robust estimators.
- Bootstrap (next week) — truncate the return after k steps and use $\gamma^{t+k-1}v_{\pi}(S_{t+k})$ in place of the sum of the remaining rewards.

Rohan's Presentation

- Data-Efficient Policy Evaluation Through Behavior Policy Search. Hanna, Thomas, Stone, and Niekum. 2017.
- [Link to slides.](#)

Summary

- Monte Carlo methods learn value functions for the observed return without model knowledge.
- Must learn action-values for control and require an exploration mechanism to ensure coverage of all state-action pairs.
- Basic idea of policy iteration still applies even though we only have an approximate policy evaluation step.

Action Items

- Start on homework
- Start reading chapter 6 for next week.
- Be thinking about final project — proposal due October 1st.
 - The more concrete your proposal is, the better guidance you will receive!