Off Policy Prediction

Reinforcement Learning

- ✦ Target policy π
- ✦ Behaviour policy μ
- ✦ Assumption of coverage: We require that wherever $\pi(a \mid s) > 0$ implies $\mu(a \mid s) > 0$
  - The behaviour policy μ must be stochastic in states where it is not identical to π
  - The target policy π may be deterministic

- Given a starting state S_t, the probability of the subsequent state-action trajectory, $A_t, S_{t+1}, A_{t+1}, \ldots, S_T$, occurring under any policy π is

$$\sum_{k=t}^{T-1} \pi(A_k \mid S_k)\, p(S_{k+1} \mid S_k, A_k)$$

✦ The importance sampling ratio is defined as:

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

Monte Carlo Algorithm

- The algorithm uses a batch of observed episodes following policy μ to estimate $v_\pi(s)$.
- We define the set of all time steps in which state s is visited, denoted $\mathcal{T}(s)$
- Let $T(t)$ denote the first time of termination of episode following time t.
- Let G_t denote the return after t up through $T(t)$.
- Then $\{G_t\}_{t \in \mathcal{T}(s)}$ are the returns that pertain to state s, and $\{\rho_t^{T(t)}\}_{t \in \mathcal{T}(s)}$ are the corresponding sampling ratios

Ordinary Importance Sampling

- To estimate $v_\pi(s)$, we simply scale the returns by the ratios and average the results

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

- This is the averaging of scaled returns

Weighted Importance Sampling

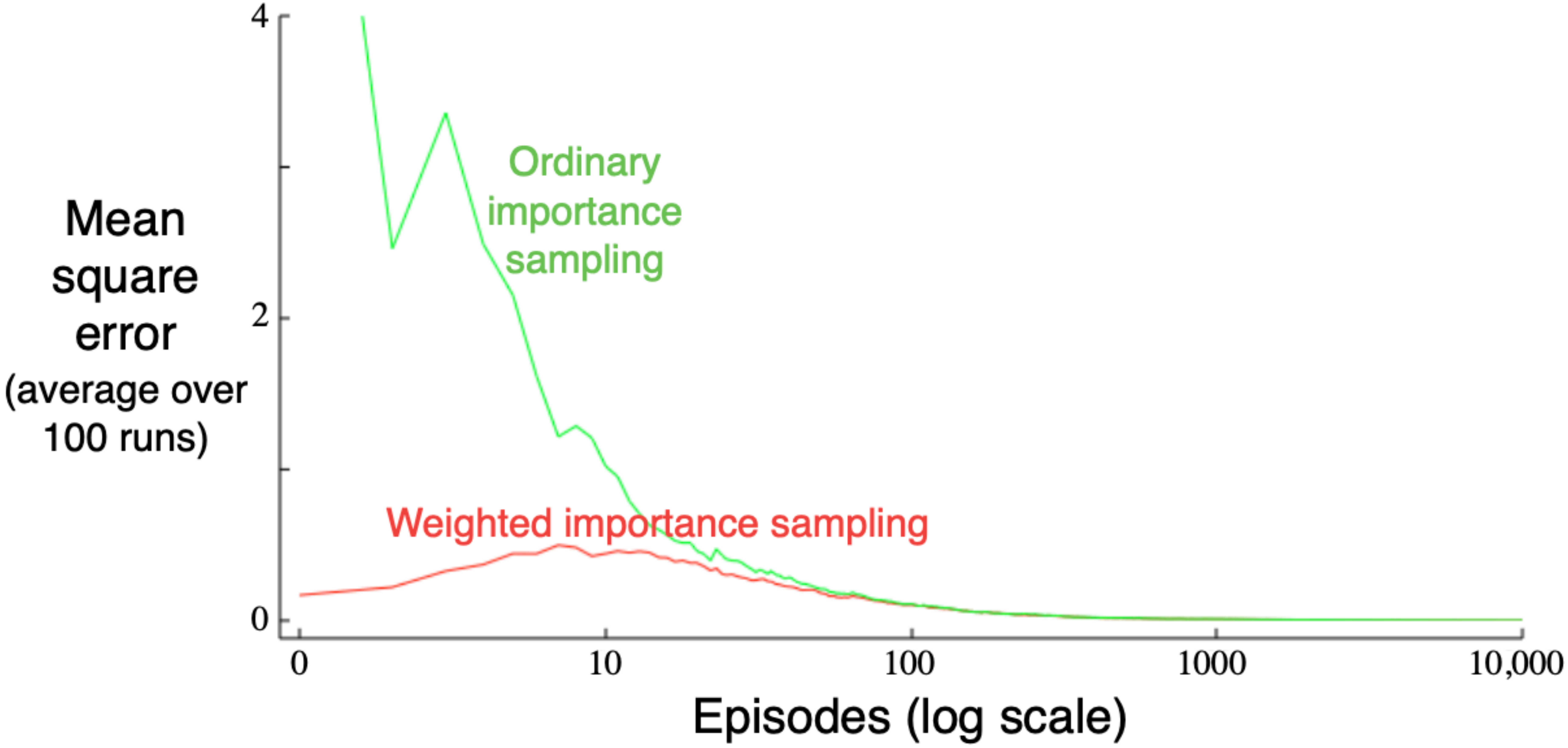
- This uses a weighted average of returns

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

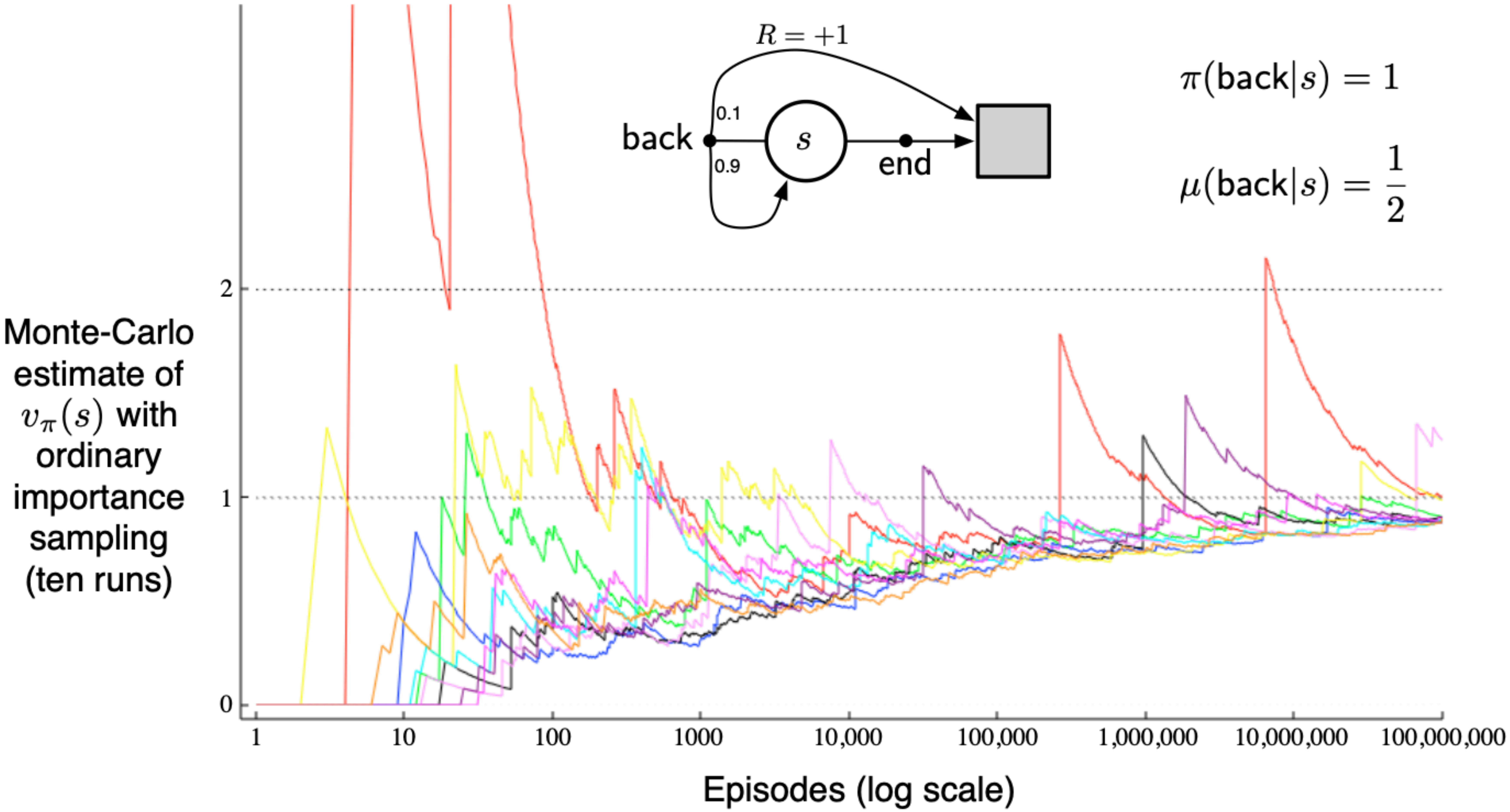
- The simple average is not biased by can have high variance
- The weighted average has high bias, but has low variance.

- The variance of ordinary importance sampling is in general unbounded because the variance of the ratios is unbounded
- The variance of weighted importance-sampling estimator converges to zero even if the variance of the ratios themselves is infinite.

Off-Policy Estimation of a Blackjack State Value



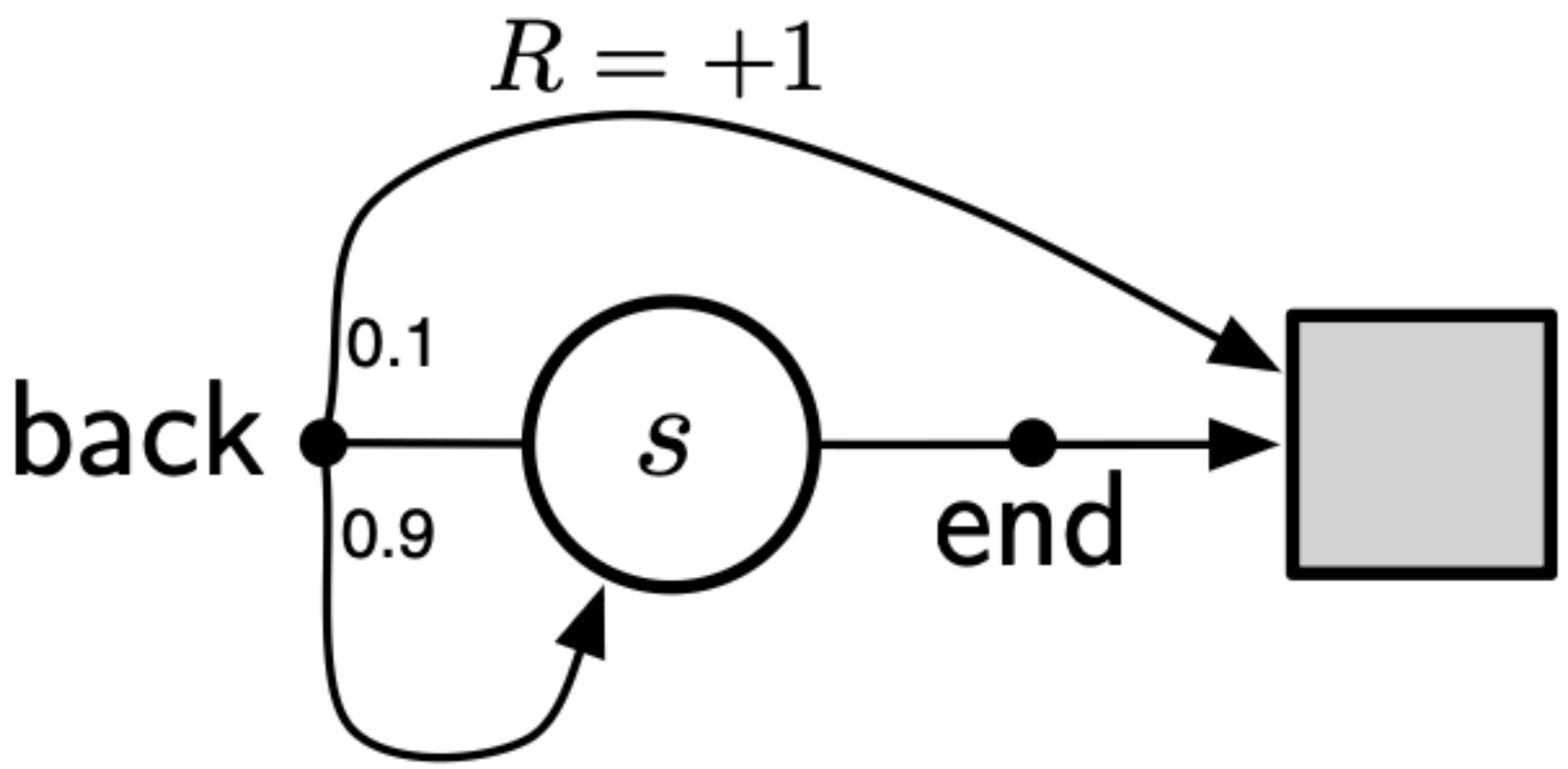
Demonstration of Infinite Variance in Ordinary Importance Sampling



- ✦ The weighted importance-sampling algorithm would give an estimate of exactly 1 everafter the first episode that was consistent with the target policy (i.e., ended with the back action).
    - The algorithm produces a weighted average of the returns consistent with the target policy, all of which would be exactly 1.

Ordinary Importance Sampling

- We can verify that the variance of the importance-sampling-scaled returns is infinite in this example.



$$\pi(\textsf{back}|s) = 1$$

$$\mu(\textsf{back}|s) = \frac{1}{2}$$

- Variance of a random variable X

$$\mathrm{Var}[X] = \mathbb{E}\left[\left(X - \bar{X}\right)^2\right] = \mathbb{E}\left[X^2 - 2X\bar{X} + \bar{X}^2\right] = \mathbb{E}\left[X^2\right] - \bar{X}^2.$$

- The expectation of the square of the scaled returns:

$$\mathbb{E}\left[\left(\prod_{t=0}^{T-1} \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} G_0\right)^2\right]$$

$\mathbb{E}[X^2]$

$$
\begin{aligned}
&= \frac{1}{2} \cdot 0.1 \left(\frac{1}{0.5}\right)^2 && \text{(the length 1 episode)} \\
&\quad + \frac{1}{2} \cdot 0.9 \cdot \frac{1}{2} \cdot 0.1 \left(\frac{1}{0.5}\frac{1}{0.5}\right)^2 && \text{(the length 2 episode)} \\
&\quad + \frac{1}{2} \cdot 0.9 \cdot \frac{1}{2} \cdot 0.9 \cdot \frac{1}{2} \cdot 0.1 \left(\frac{1}{0.5}\frac{1}{0.5}\frac{1}{0.5}\right)^2 && \text{(the length 3 episode)} \\
&\quad + \cdots \\
&= 0.1 \sum_{k=0}^{\infty} 0.9^k \cdot 2^k \cdot 2 \\
&= 0.2 \sum_{k=0}^{\infty} 1.8^k \\
&= \infty.
\end{aligned}
$$

Deep Reinforcement Learning

- How do we get feedback on our estimate $Q(s, a)$?

- If we look one move ahead by taking action, we get an estimate of $Q(s, a)$ as

$$R(s, a) + \gamma \max_{a'} Q(s', a')$$

- The squared difference between the old estimate and the new estimate can serve as the loss

$$\left[Q(s,a) - \left(R(s,a) - \gamma \max_{a'} Q(s',a')\right)\right]^2$$

- This difference is also called as the temporal difference error or TD(0).
- If we looked two steps into the future it would be TD(1).

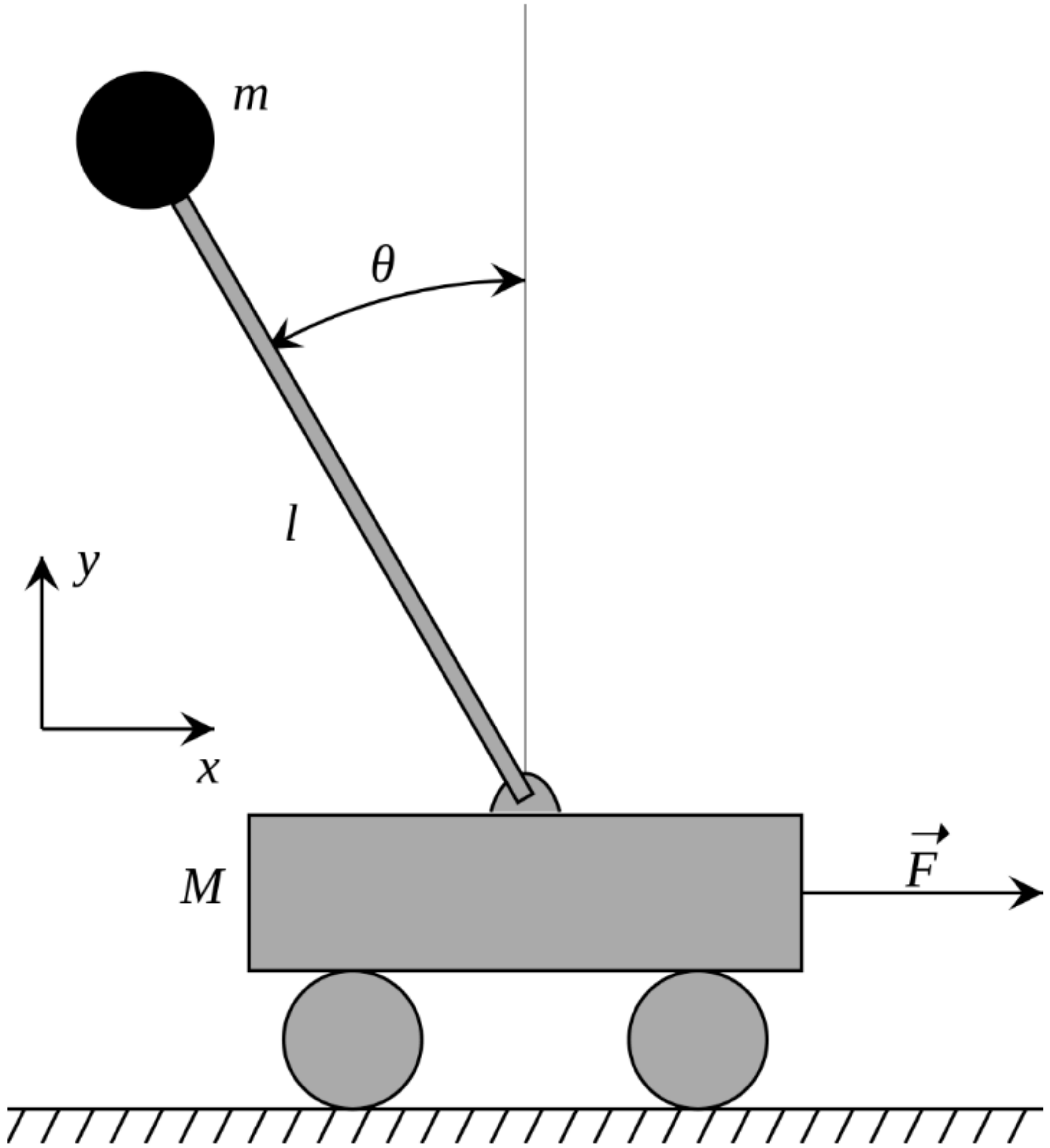
Policy Gradient Methods

✦ Cart Pole

✦ State: (4 vector)

- Position of the cart
- Angle of the pole

(after the previous and the current move)



Policy Gradient Methods

- ✦ The objective is to learn the policy, i.e., the recommended action for a given state.
- ✦ We play an entire game
  - We make 20 moves before the pole tips over
- ✦ We compute the discounted reward for the first state

$$D_0(\mathbf{s}, \mathbf{a}) = \sum_{t=0}^{n-1} \gamma^t R(s_t, a_t, s_{t+1})$$

REINFORCE

- If we took n steps we compute the future discounted reward for any of the state-action combinations s_i, a_i from the recurrence relation

$$D_n(\mathbf{s}, \mathbf{a}) = 0$$
$$D_i(\mathbf{s}, \mathbf{a}) = R(s_i, a_i, s_{i+1}) + \gamma D_{i+1}(\mathbf{s}, \mathbf{a})$$

- Loss function

$$L(\mathbf{s}, \mathbf{a}) = \sum_{t=0}^{n-1} D_t(\mathbf{s}, \mathbf{a}) \ \left(-\log \Pr(a_t \,|\, s_t)\right)$$

Actor-Critic Methods

- These work better than the actor programs
- They do not require playing the full episode

Advantage Actor-Critic

- The advantage of a state-action pair is the difference between the state-action Q value and the state's value:

$$A(s, a) = Q(s, a) - V(s)$$

- Advantage is a negative number
- The a2c loss:

$$L_A(\mathbf{s}, \mathbf{a}) = \sum_{t=0}^{n-1} A(s_t, a_t)\ \left(\log \mathsf{Pr}(a_t \,|\, s_t)\right)$$

- ✦ The variance of advantage $A(s_t, a_t)$ is smaller than the variance of $D_t(\mathbf{s}, \mathbf{a})$
- ✦ It is easier to approximate a function with low variance than one with high.

A2C

- ✦ How do we estimate $A(s_t, a_t)$?

- ✦ We estimate $Q(s, a)$ in the same way as we estimate $D(\mathbf{s}, \mathbf{a})$

- ✦ We estimate $V(s)$ by using another function called *Critic*.

  - It is another neural network trained to produce good estimates of $V(s)$

  - It is trained by using the loss on the disparity between the actual rewards found and the output of the NN approximation.

A2C

- Need not play an entire episode
- Can make use of multiple environments (games) as a batch of training examples.

Experience Replay

- ✦ One big problem with the application of RL in real world is the acquisition of training data.
- ✦ In experience replay we use the same training data multiple times.
- ✦ For each time t we save $\langle s_t, a_t, s_{t+1}, r_t \rangle$.
  - We can do a forward and backward pass on our data.
  - We can play and replay, each time step in a random order.
  - Taking random actions from several different game plays reduces the correlation in data.

- ✦ But this too has a problem — the old data can go stale !
- ✦ We can instead keep a buffer
    - We then replace the oldest game in the buffer with a new game played using the new policy based upon the up-to-date parameters.