

Policy Evaluation

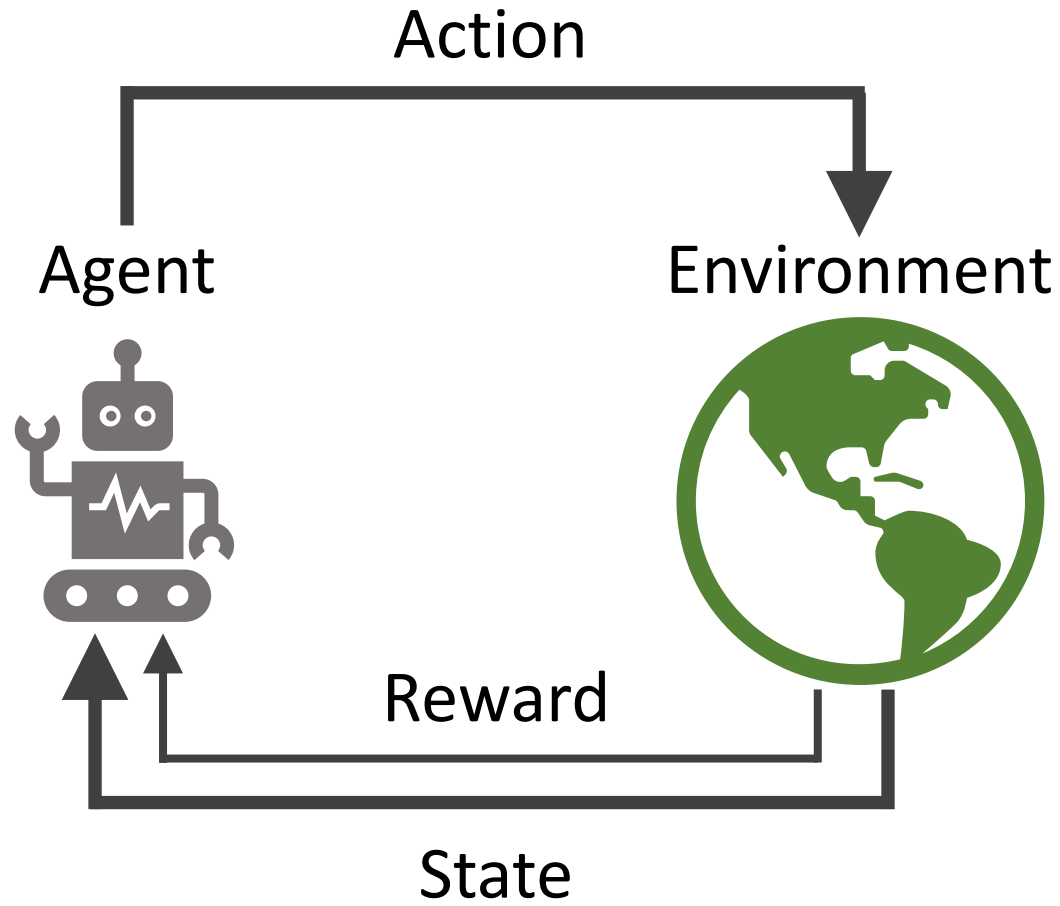
CMPT 729 G100

Jason Peng

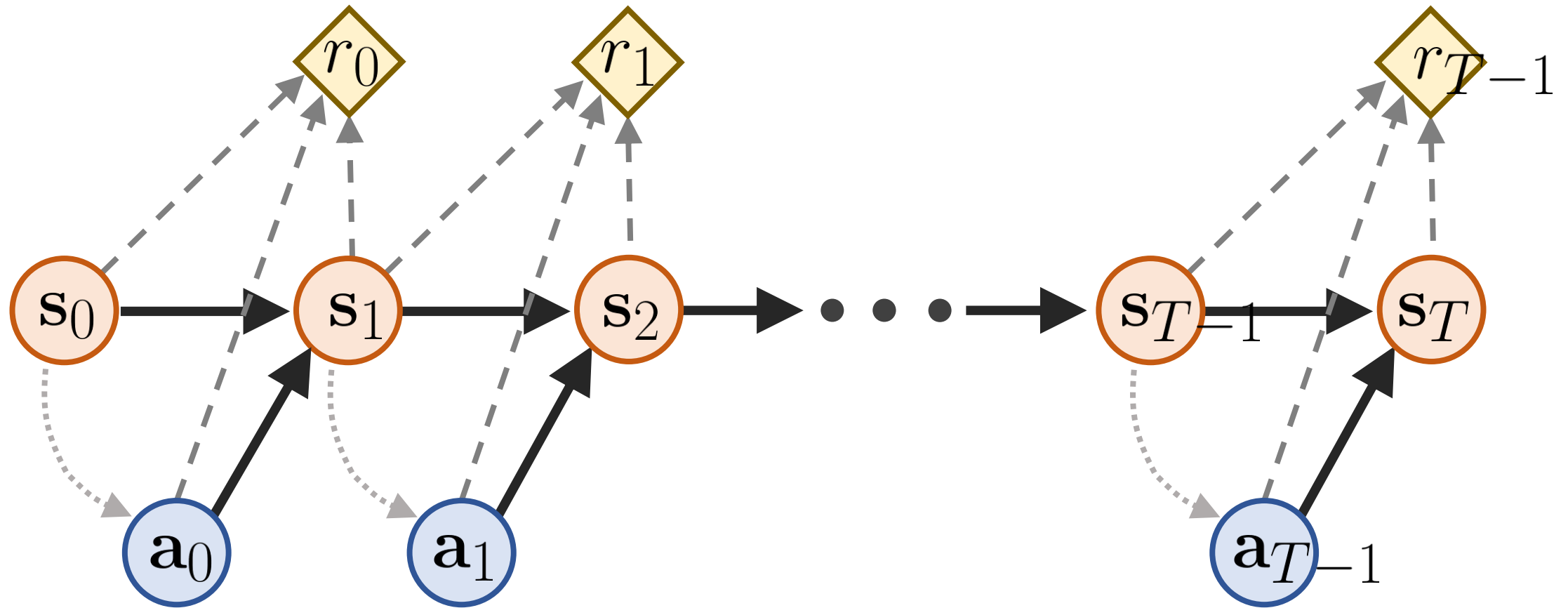
Overview

- Policy Evaluation
- Value Functions
- Monte-Carlo Methods
- Dynamic Programming Methods
- Optimal Policies

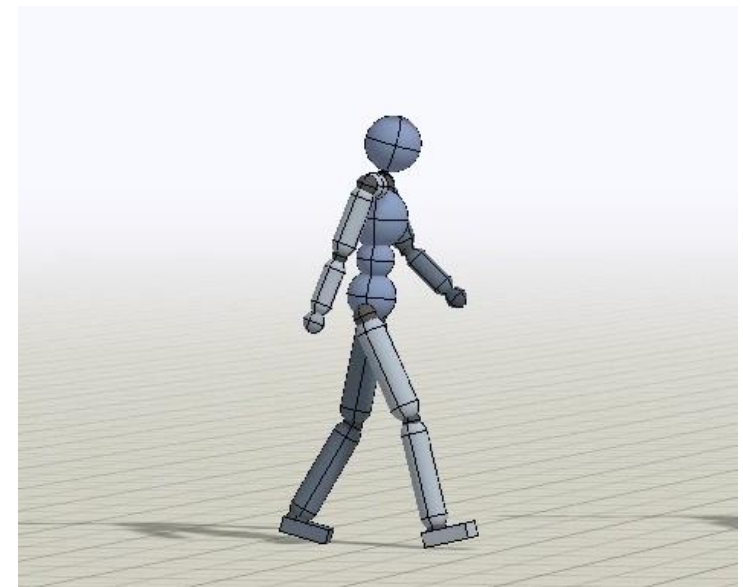
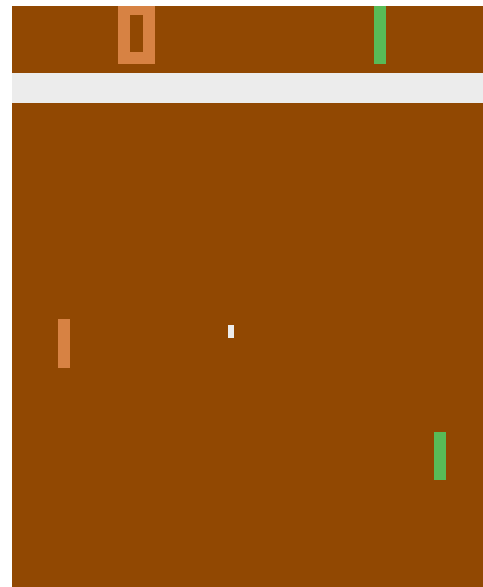
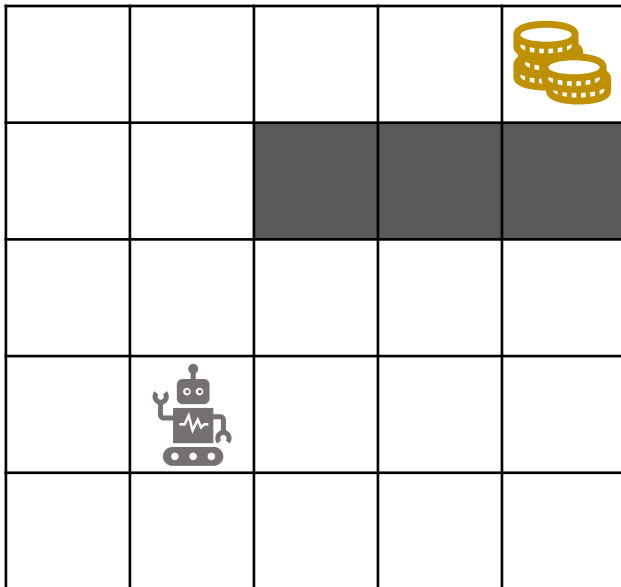
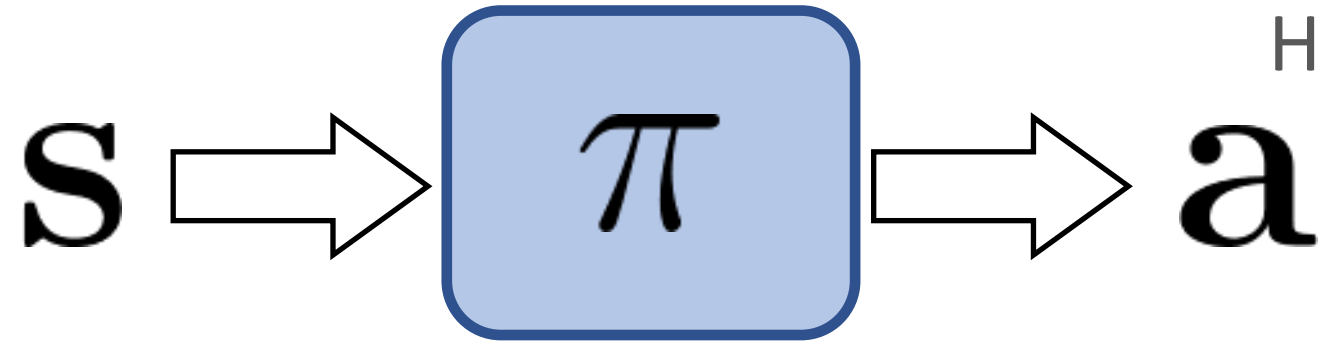
Agent-Environment Interface



Markov Decision Process



Policy Evaluation



Performance

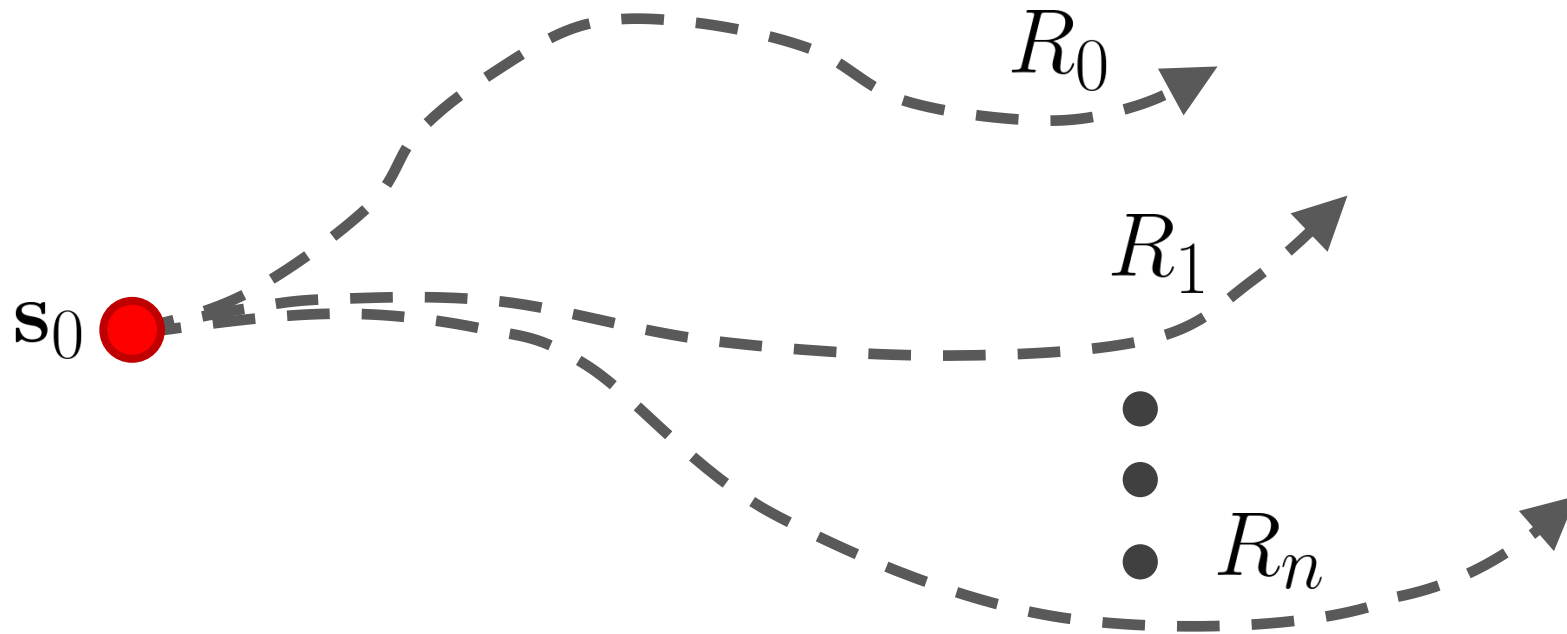
$$\underline{J(\pi)} = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\underline{\sum_{t=0}^{T-1} \gamma^t r_t} \right]$$

Expected Return

Episodic Return

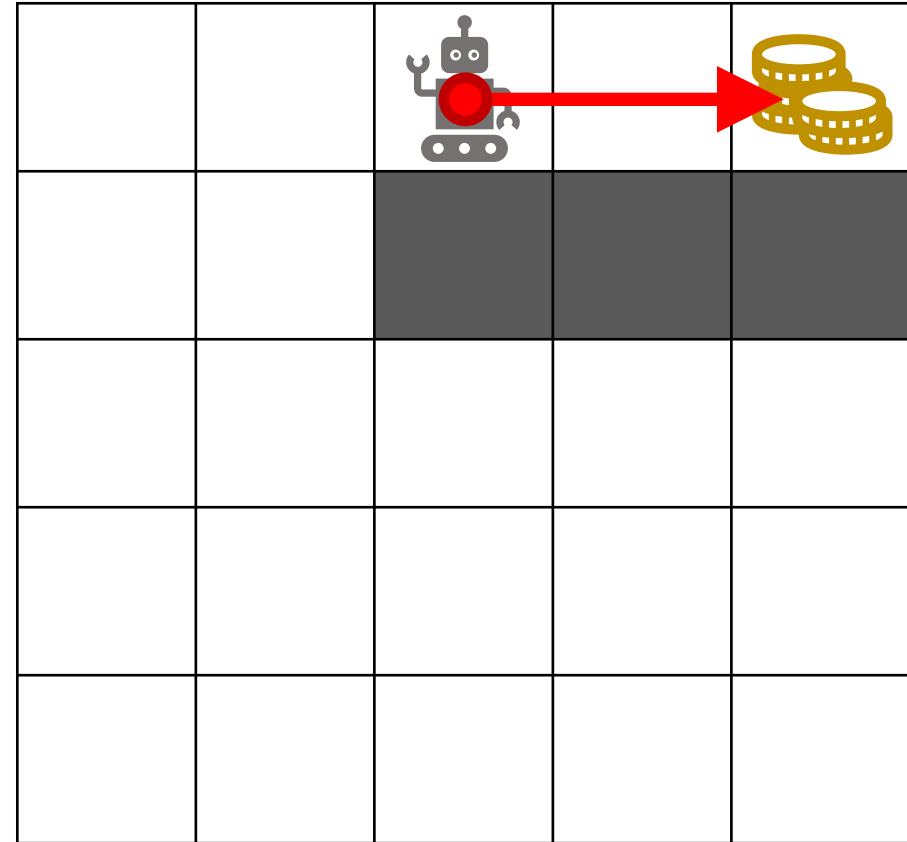
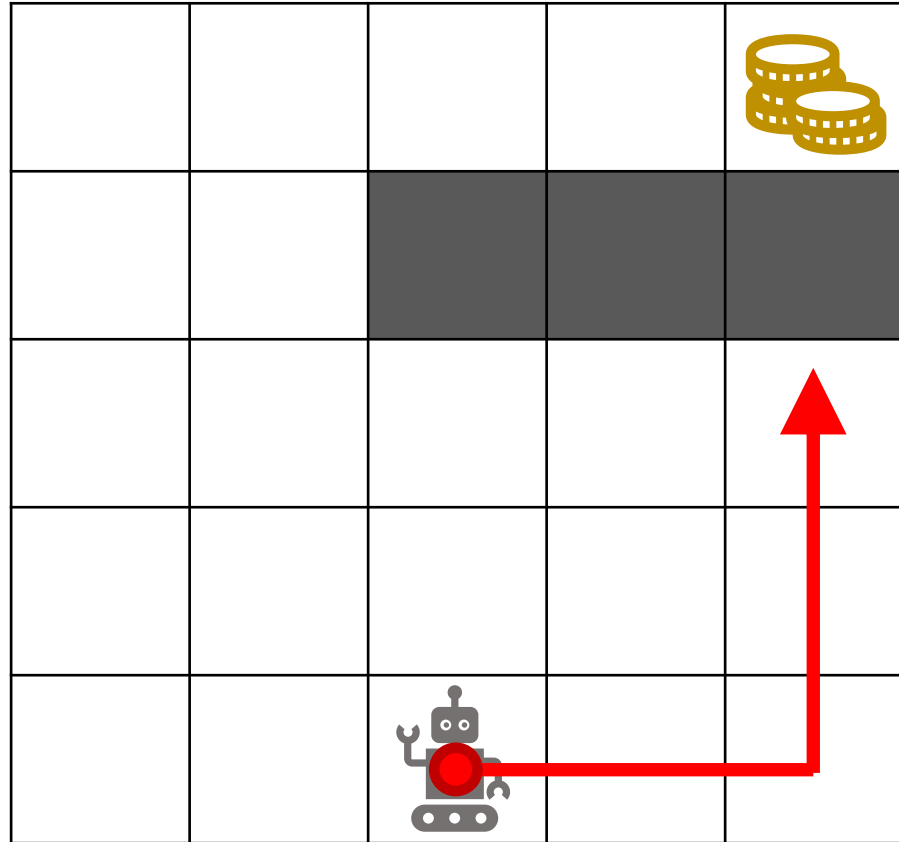
Monte-Carlo Estimate

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

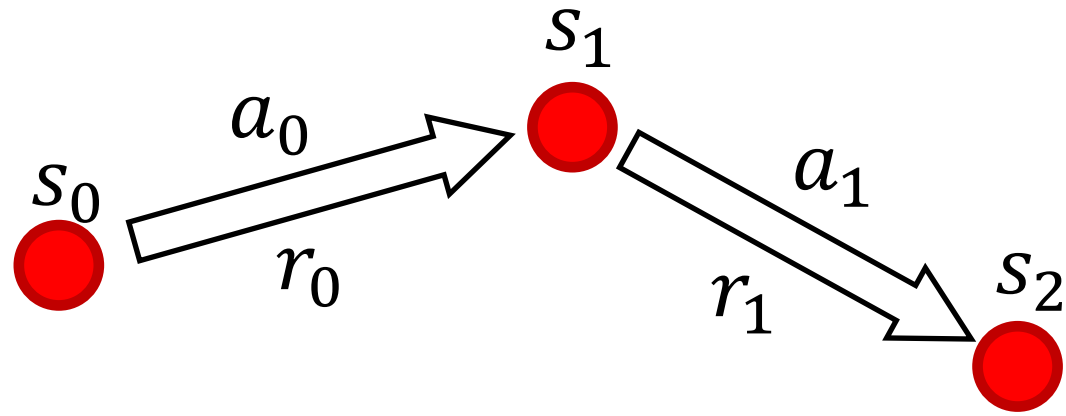


$$J(\pi) \approx \frac{1}{n} \sum_{i=0}^n R_i$$

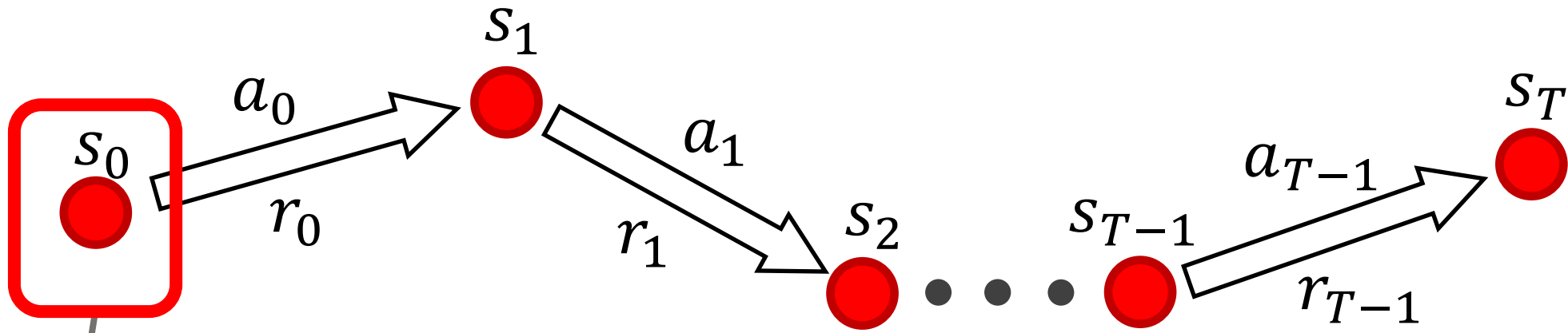
Expected Return



Reward-To-Go

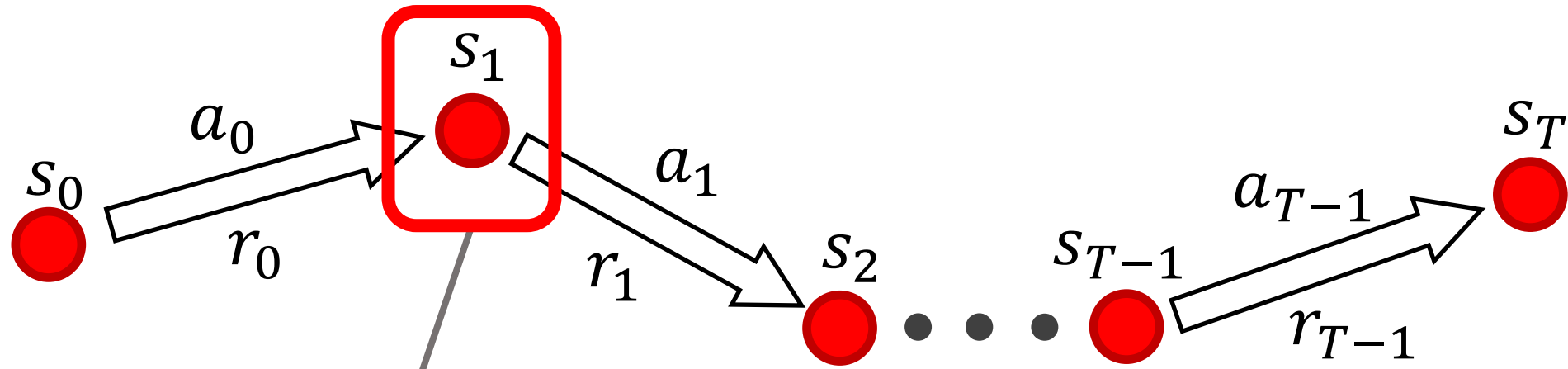


Reward-To-Go



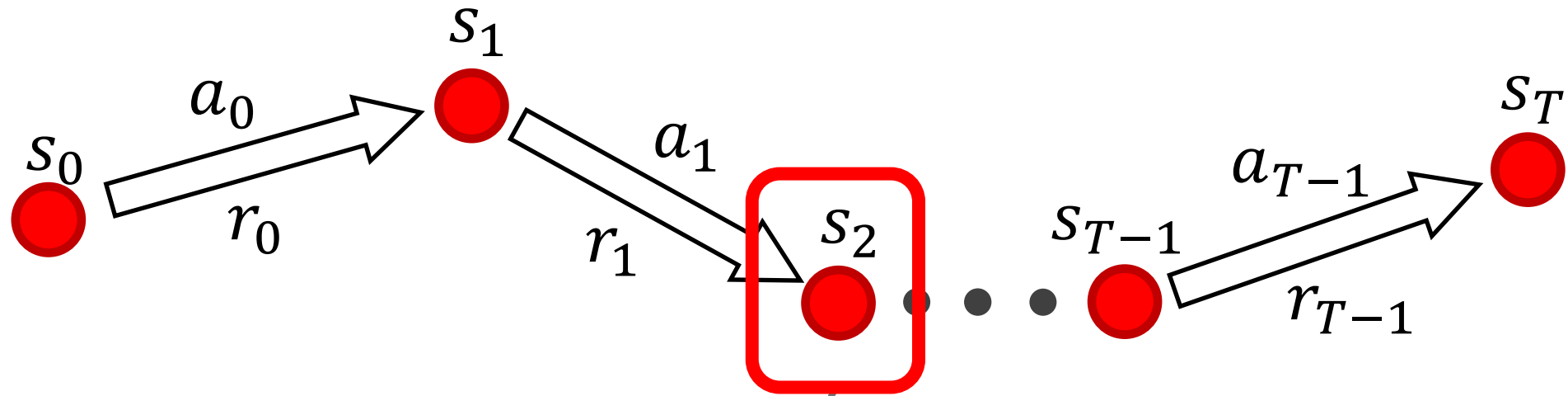
$$R_0 = r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$$

Reward-To-Go



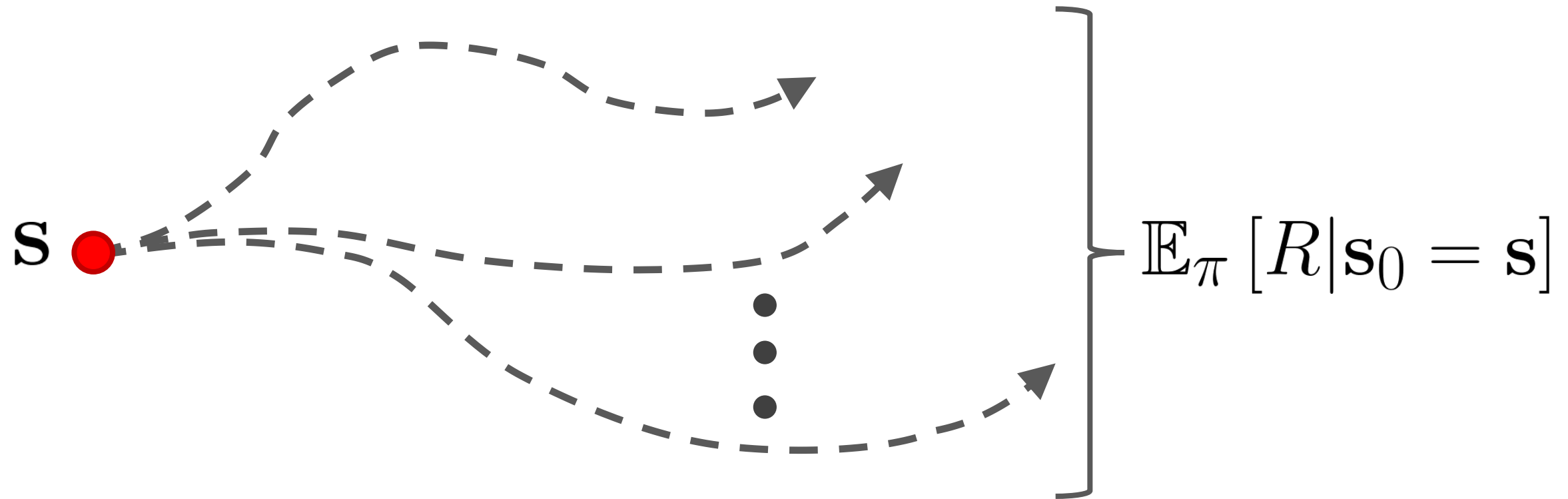
$$R_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots$$

Reward-To-Go



$$R_2 = r_2 + \gamma r_3 + \gamma^2 r_4 + \dots$$

Reward-To-Go



Value Function

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s})} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

Value Function

- Input: state \mathbf{s}
- Output: expected return of following a policy π start at a state \mathbf{s}

Value Function

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\tau \sim \underline{p(\tau|\pi, \mathbf{s}_0 = \mathbf{s})}} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

Likelihood of a trajectory
under π starting at \mathbf{S}

Value Function

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\tau \sim p(\tau | \pi, \underline{\mathbf{s}_0 = \mathbf{s}})}$$
$$\left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau | \underline{\pi})}$$
$$\left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

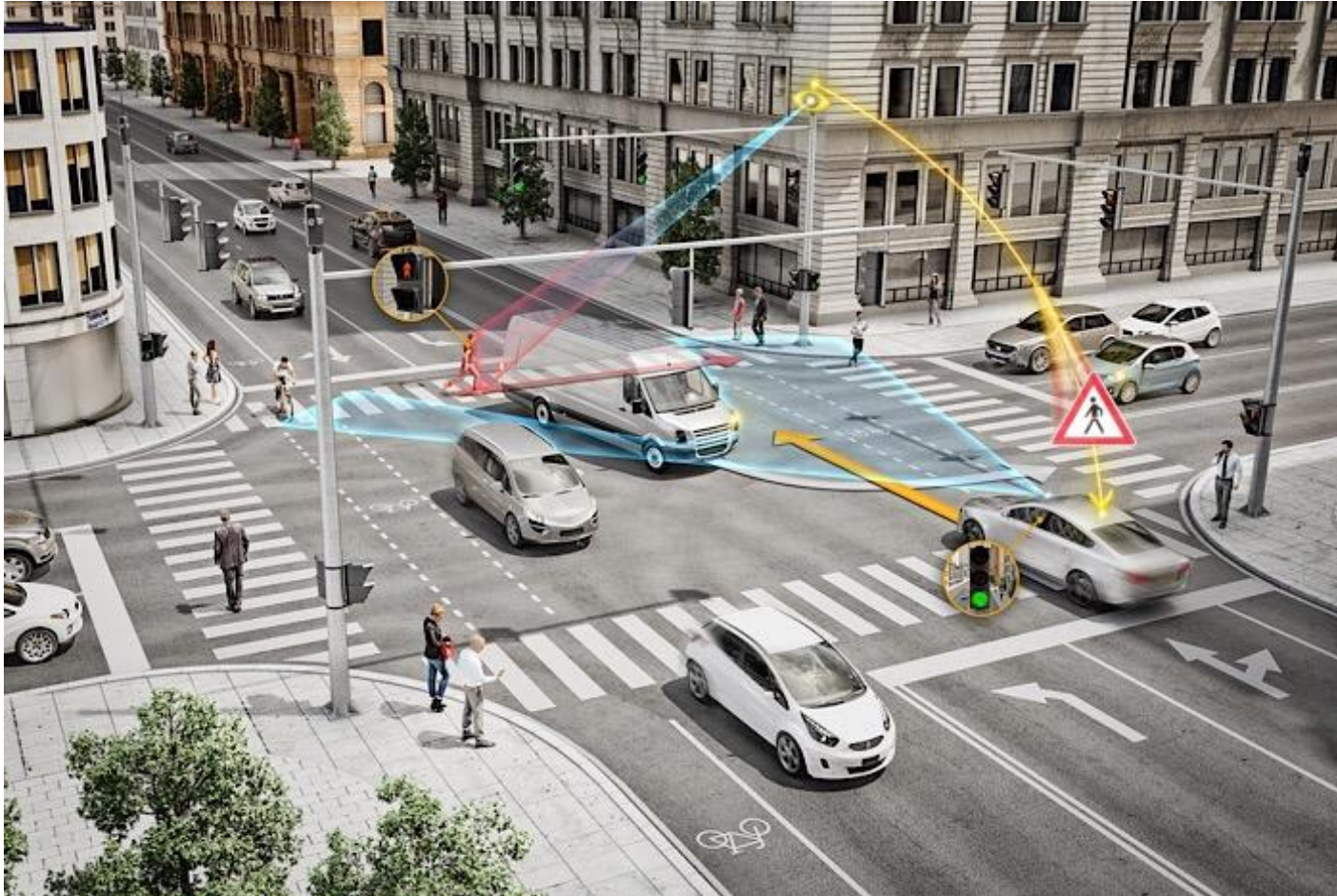
Value Function

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s})} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

$$J(\pi) = V^\pi(\mathbf{s}_0)$$

Why a Value Function?

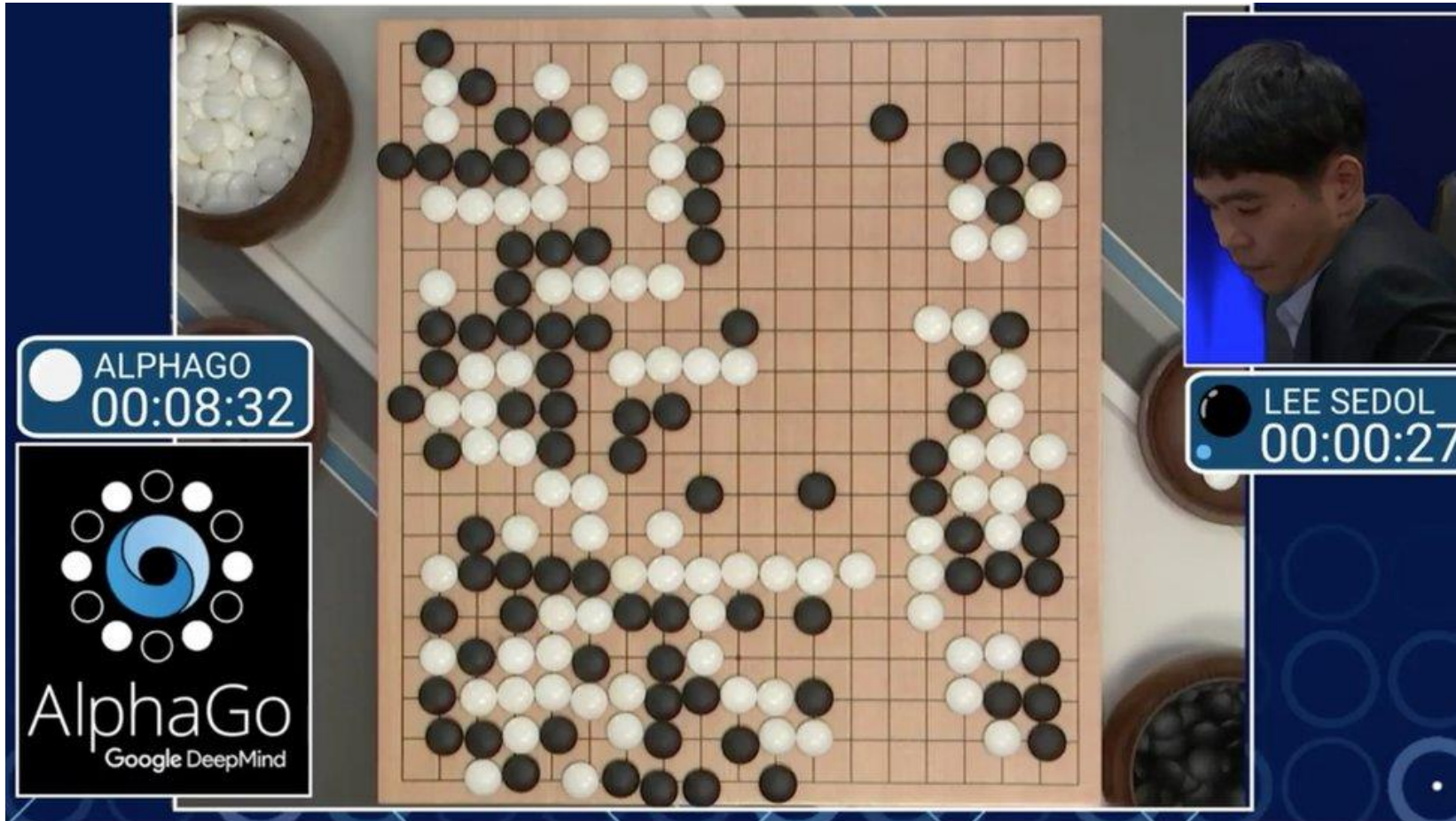


[Impact Lab]

$$V^{\pi}(s)?$$

Can the policy do well here?

Why a Value Function?

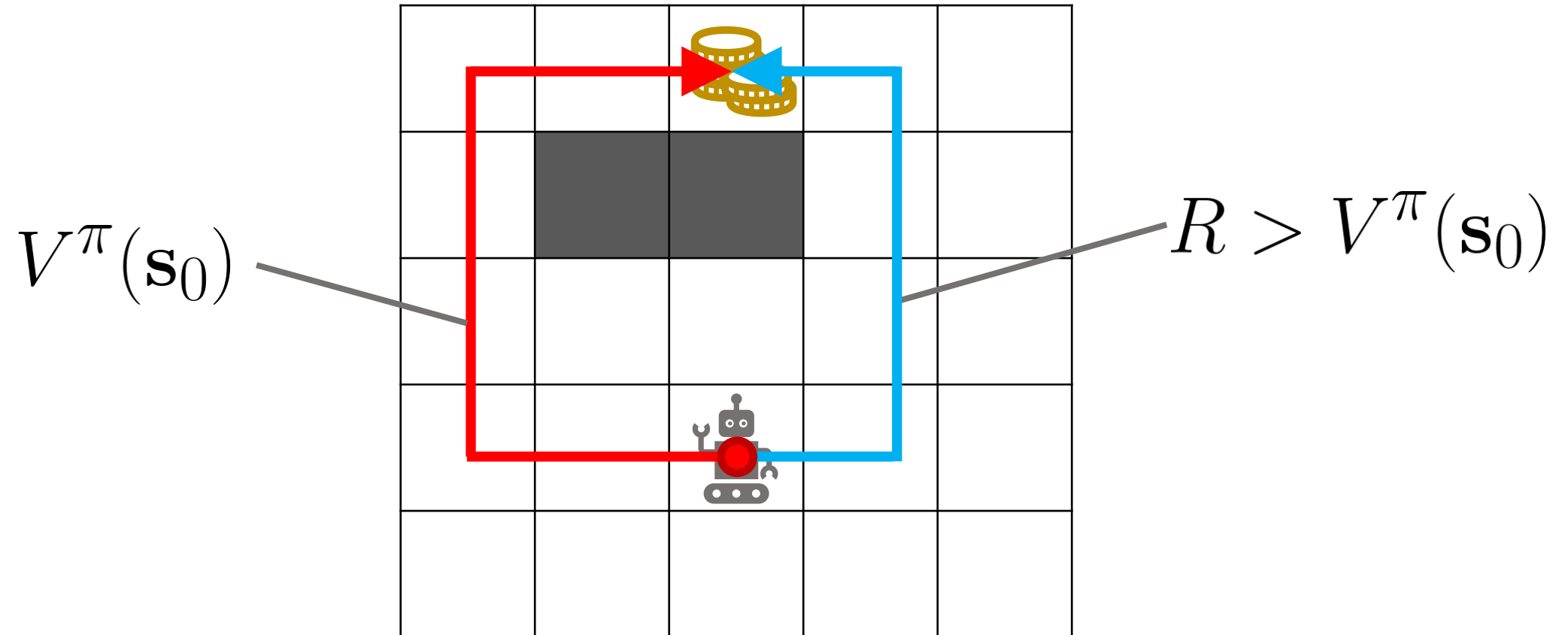


AlphaGo [DeepMind]

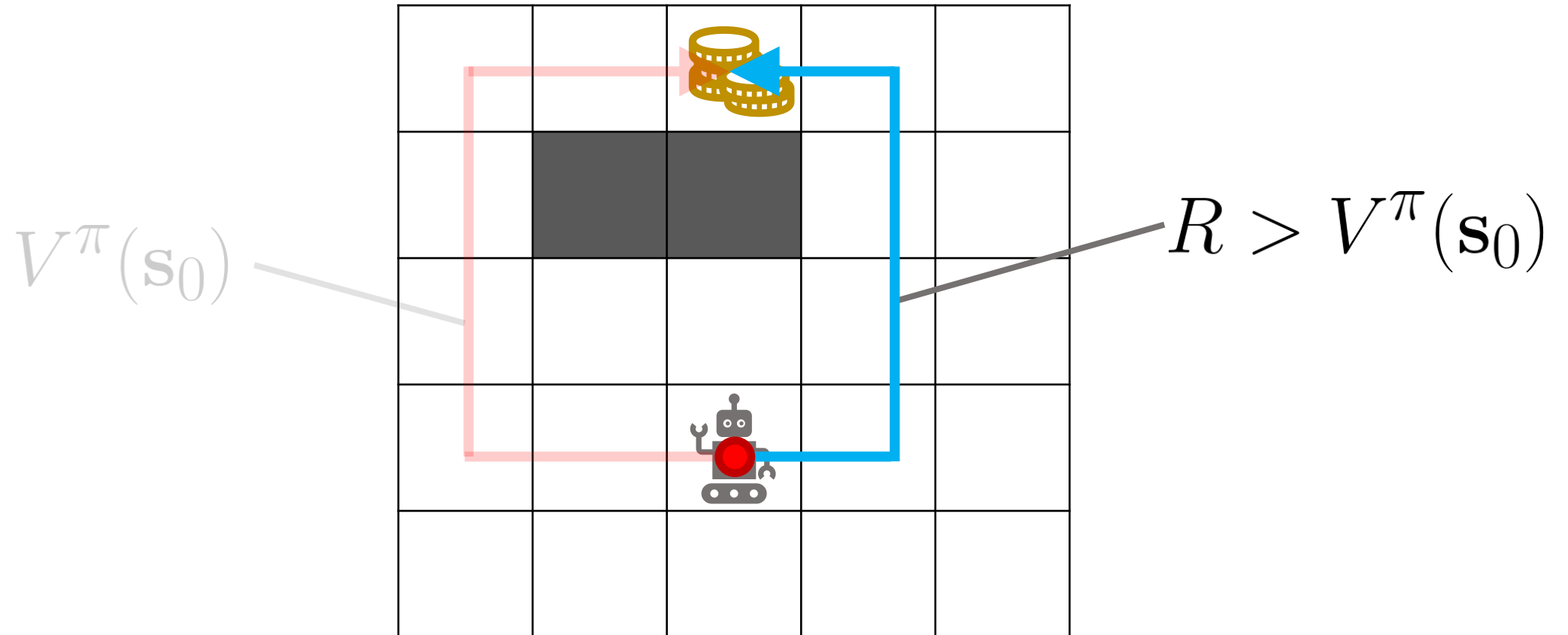
$$V^{\pi}(\mathbf{s})?$$

Can the policy win?

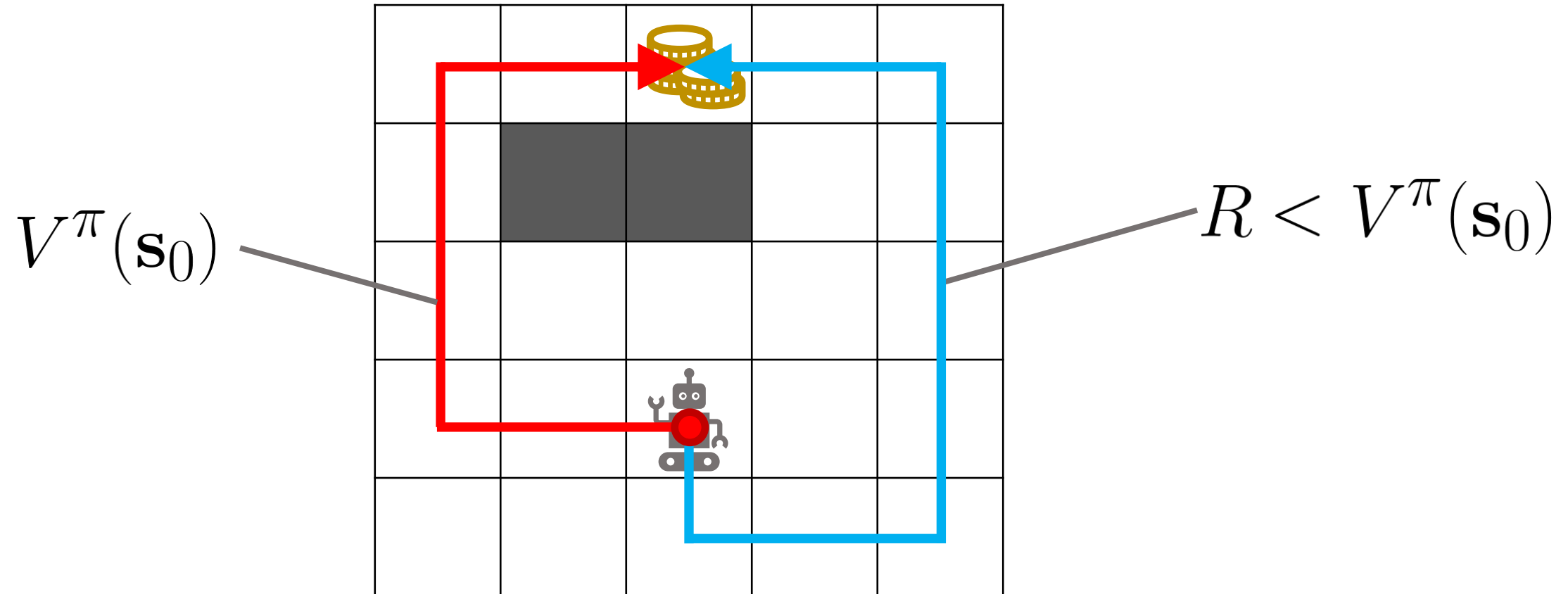
Policy Improvement



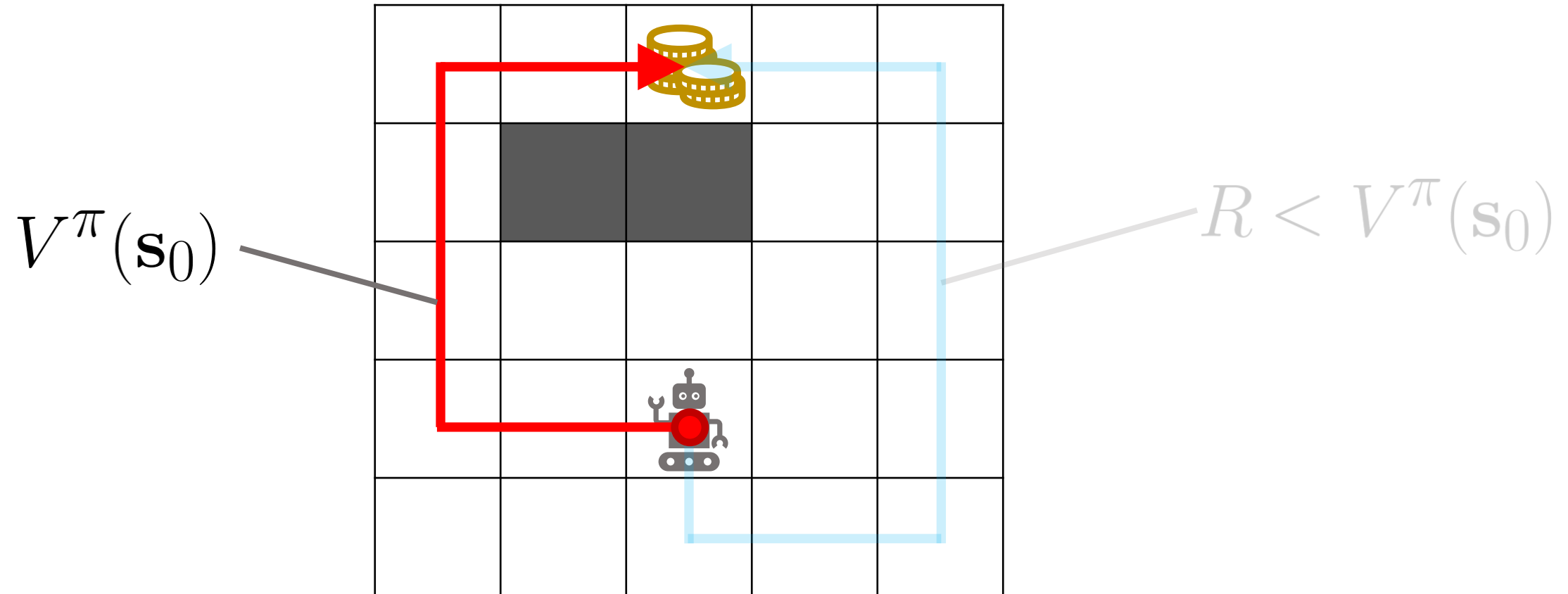
Policy Improvement



Policy Improvement



Policy Improvement



Value Function Approximation

$$V^\pi ?$$

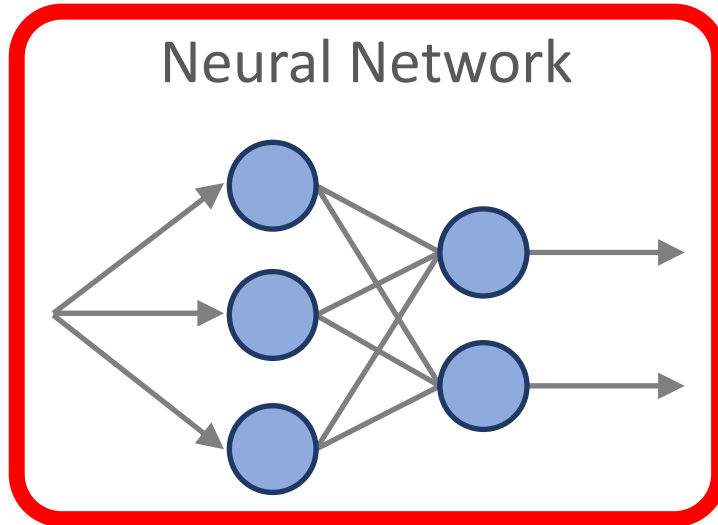
Value Function Approximation

$$\hat{V}^{\pi} \approx V^{\pi}$$

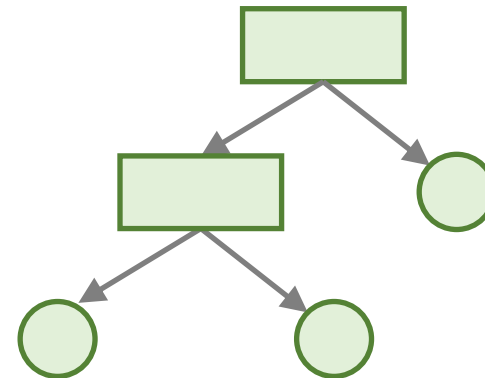
Function Approximator

Look-Up Table

Neural Network

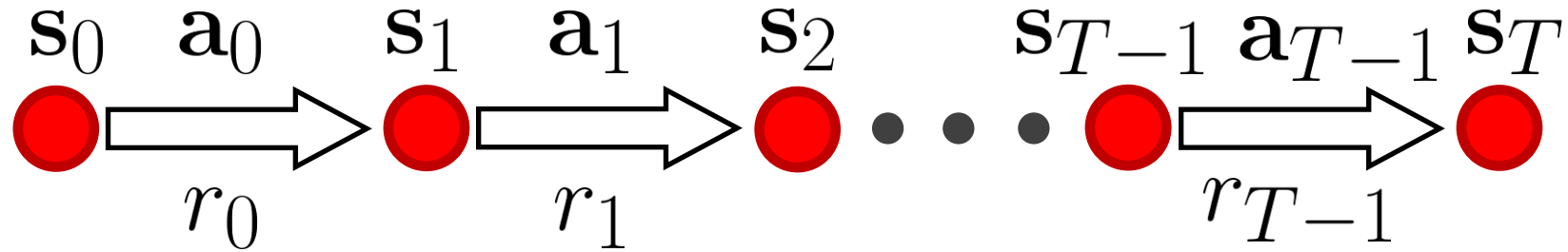


Decision Tree



Etc...

Learning



$$R_0 = r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \Rightarrow y_0$$

$$R_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots \Rightarrow y_1$$

$$R_2 = r_2 + \gamma r_3 + \gamma^2 r_4 + \dots \Rightarrow y_2$$

⋮

Dataset
 $\{s_i, y_i\}$

Supervised Learning

$$\hat{V}^\pi = \arg \min_V \mathbb{E}_{(\mathbf{s}_i, y_i) \sim p(\mathbf{s}, y | \pi)} \left[\|y_i - V(\mathbf{s}_i)\|^2 \right]$$

Supervised Learning

$$\hat{V}^{\pi} = \arg \min_V \mathbb{E}_{(\mathbf{s}_i, y_i) \sim p(\mathbf{s}, y | \pi)} \left[\|y_i - V(\mathbf{s}_i)\|^2 \right]$$

Supervised Learning

$$\hat{V}^\pi = \arg \min_{\underline{V}} \mathbb{E}_{(\mathbf{s}_i, y_i) \sim p(\mathbf{s}, y | \pi)} \left[\|y_i - V(\mathbf{s}_i)\|^2 \right]$$

Supervised Learning

$$\hat{V}^\pi = \arg \min_V \mathbb{E}_{(\mathbf{s}_i, y_i) \sim p(\mathbf{s}, y | \pi)} \left[\|y_i - V(\mathbf{s}_i)\|^2 \right]$$

Mean Prediction Error

Supervised Learning

$$\hat{V}^\pi = \arg \min_V \mathbb{E}_{(\mathbf{s}_i, y_i) \sim p(\mathbf{s}, y | \pi)} \left[\|y_i - V(\mathbf{s}_i)\|^2 \right]$$

Collect data from policy

Supervised Learning

$$\hat{V}^\pi = \arg \min_V \mathbb{E}_{(\mathbf{s}_i, y_i) \sim p(\mathbf{s}, y | \pi)} \left[\underbrace{\|y_i - V(\mathbf{s}_i)\|^2}_{\text{Prediction Error}} \right]$$

Supervised Learning

$$\hat{V}^\pi = \arg \min_V \mathbb{E}_{(\mathbf{s}_i, y_i) \sim p(\mathbf{s}, y | \pi)} \left[\|\underline{y_i} - V(\mathbf{s}_i)\|^2 \right]$$

“Target Value”
Monte-Carlo Estimate

$$y = \sum_{t=0}^{T-1} \gamma^t r_t$$

Monte-Carlo Method

$$y = \sum_{t=0}^{T-1} \gamma^t \underline{r_t}$$

Random Variable



Unbiased



High variance → Slow convergence

Dynamic Programming

Recursive Property of Value Function

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s})} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

Likelihood of a trajectory
under π starting at \mathbf{s}

Recursive Property of Value Function

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s})} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

Recursive Property of Value Function

$$\begin{aligned} V^\pi(\mathbf{s}) &= \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s})} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right] \\ &= \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s})} \left[\underline{r_0} + \sum_{\underline{t=1}}^{T-1} \gamma^t r_t \right] \end{aligned}$$

Recursive Property of Value Function

$$\begin{aligned} V^\pi(\mathbf{s}) &= \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s})} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right] \\ &= \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s})} \left[r_0 + \sum_{t=1}^{T-1} \gamma^t r_t \right] \\ &= \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s})} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \end{aligned}$$

Recursive Property of Value Function

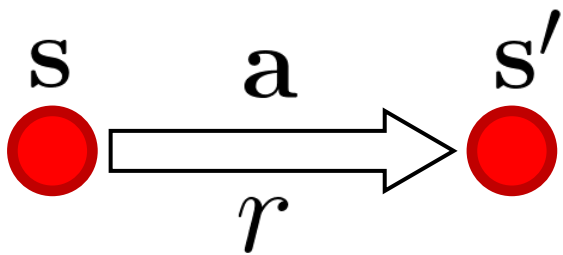
$$V^\pi(\mathbf{s}) = \mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s})} \left[\underbrace{r_0}_{\text{Immediate reward at } \mathbf{s}_0} + \gamma \underbrace{\sum_{t=1}^{T-1} \gamma^{t-1} r_t}_{\text{Return starting at } \mathbf{s}_1} \right]$$

Recursive Property of Value Function

$$\begin{aligned} V^\pi(\mathbf{s}) &= \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s})} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \\ &= \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s}')} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \end{aligned}$$

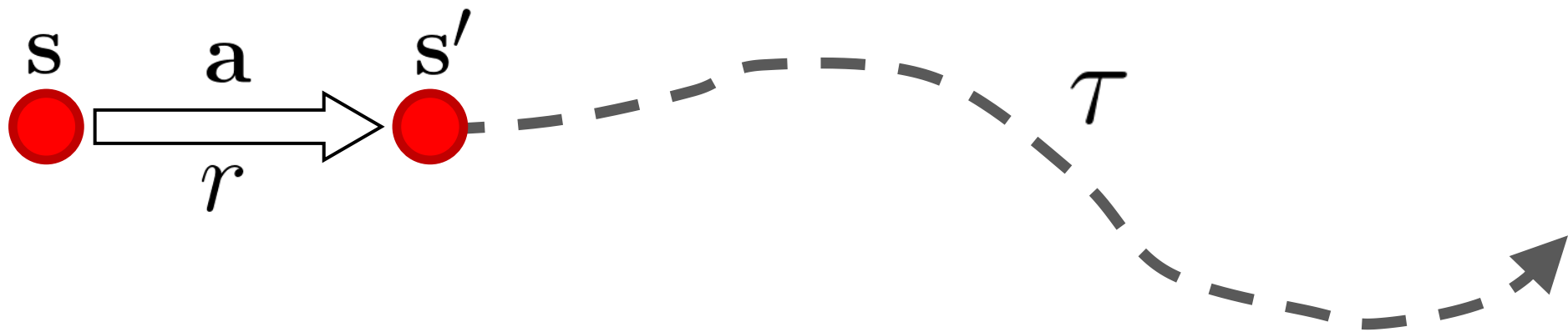
Recursive Property of Value Function

$$\begin{aligned} V^\pi(\mathbf{s}) &= \mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s})} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \\ &= \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a} | \mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' | \mathbf{s}, \mathbf{a})} \mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s}')} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \end{aligned}$$



Recursive Property of Value Function

$$\begin{aligned} V^\pi(\mathbf{s}) &= \mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s})} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \\ &= \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a} | \mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' | \mathbf{s}, \mathbf{a})} \underline{\mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s}')}} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \end{aligned}$$



Recursive Property of Value Function

$$\begin{aligned} V^\pi(\mathbf{s}) &= \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s})} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \\ &= \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} \mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s}')} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \\ &= \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} \left[r_0 + \gamma \underbrace{\mathbb{E}_{\tau \sim p(\tau|\pi, \mathbf{s}_0=\mathbf{s}')} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]} \right] \end{aligned}$$

Recursive Property of Value Function

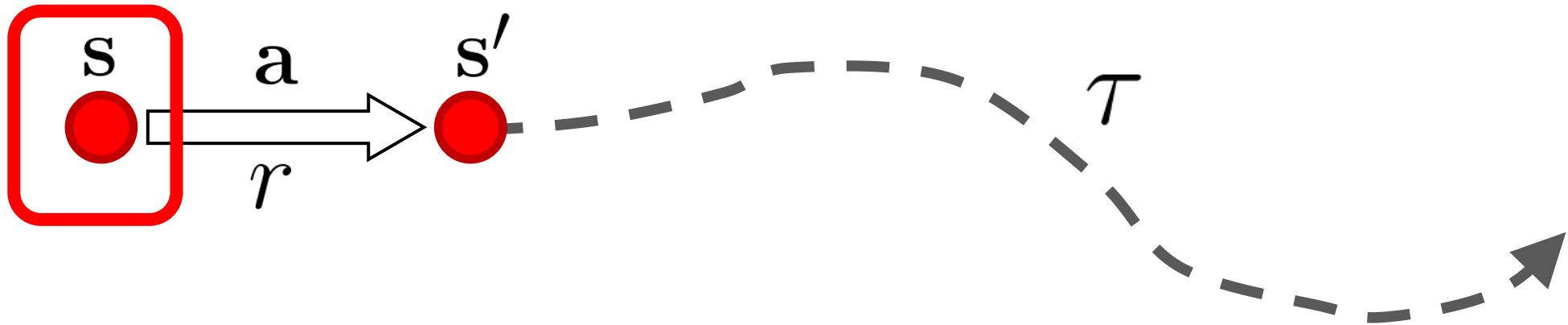
$$\begin{aligned} V^\pi(\mathbf{s}) &= \mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s})} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \\ &= \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a} | \mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' | \mathbf{s}, \mathbf{a})} \mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s}')} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \\ &= \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a} | \mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' | \mathbf{s}, \mathbf{a})} \left[r + \gamma \underbrace{\mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s}')} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]}_{= V^\pi(\mathbf{s}')} \right] \\ &= V^\pi(\mathbf{s}') \end{aligned}$$

Recursive Property of Value Function

$$\begin{aligned} V^\pi(\mathbf{s}) &= \mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s})} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \\ &= \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a} | \mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' | \mathbf{s}, \mathbf{a})} \mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s}')} \left[r_0 + \gamma \sum_{t=1}^{T-1} \gamma^{t-1} r_t \right] \\ &= \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a} | \mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' | \mathbf{s}, \mathbf{a})} \left[r + \gamma \mathbb{E}_{\tau \sim p(\tau | \pi, \mathbf{s}_0 = \mathbf{s}')} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right] \right] \\ &= \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a} | \mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}' | \mathbf{s}, \mathbf{a})} [r + \gamma V^\pi(\mathbf{s}')] \end{aligned}$$

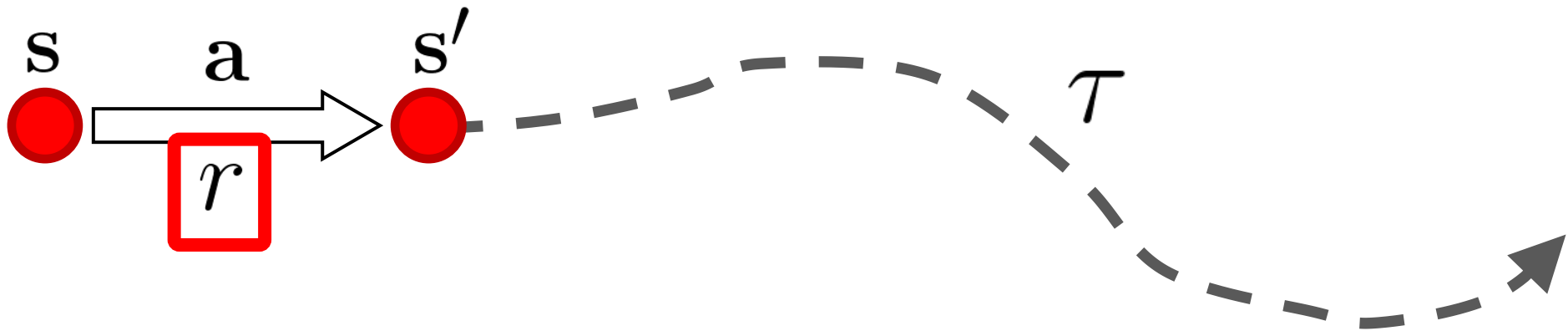
Recursive Property of Value Function

$$\underline{V^\pi(\mathbf{s})} = \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [r + \gamma V^\pi(\mathbf{s}')]]$$



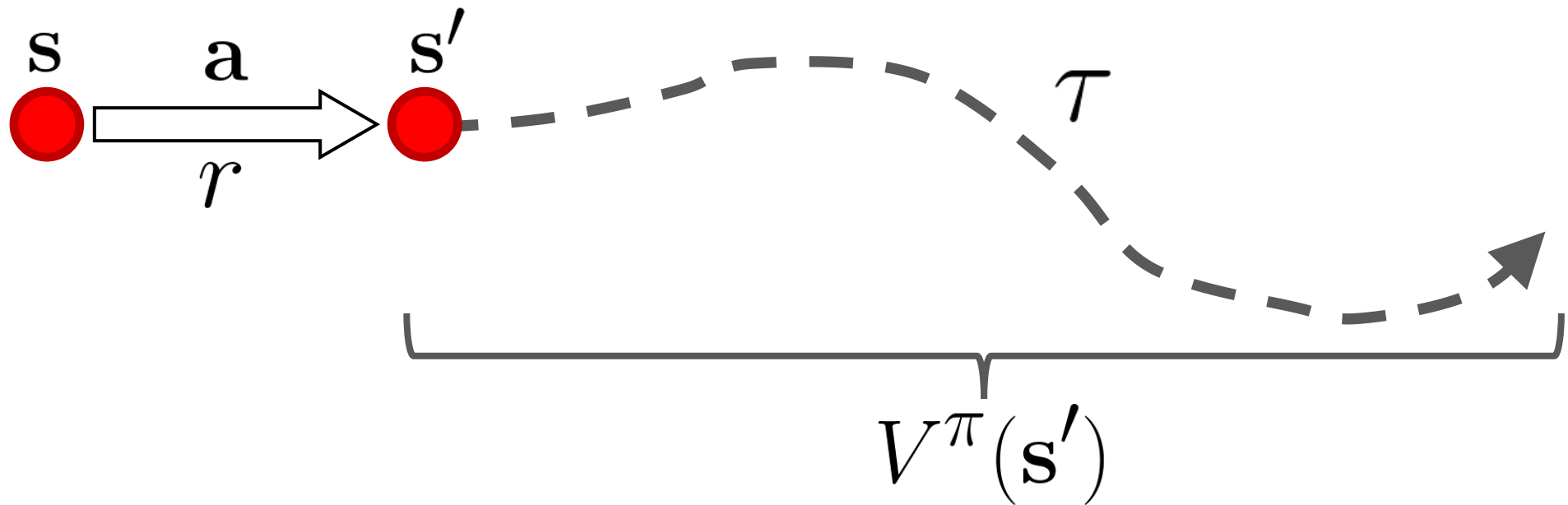
Recursive Property of Value Function

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [\underline{r} + \gamma V^\pi(\mathbf{s}')]]$$



Recursive Property of Value Function

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [r + \gamma \underline{V^\pi(\mathbf{s}')}]$$




Recursive Property of Value Function

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [r + \gamma V^\pi(\mathbf{s}')]]$$

Bellman equation for V^π

Recursive Property of Value Function

$$y = r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$$

$$y = r_0 + \gamma V^\pi(\mathbf{s}_1)$$


Recursive Property of Value Function

$$y = r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$$

$$y = \underline{r_0} + \gamma \underline{V^\pi(\mathbf{s}_1)}$$

random variable
high-variance

deterministic
low-variance



Unbiased



Low-variance

Recursive Property of Value Function

$$y = r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$$

$$y = r_0 + \gamma \boxed{V^\pi}(\mathbf{s}_1)$$



Unbiased



Low-variance

How do we get this?

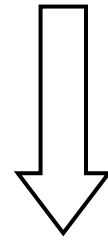
Supervised Learning

$$\hat{V}^\pi = \arg \min_V \mathbb{E}_{(\mathbf{s}_i, y_i) \sim p(\mathbf{s}, y | \pi)} \left[\|y_i - V(\mathbf{s}_i)\|^2 \right]$$

$$\hat{V}^\pi \approx V^\pi$$

Bootstrapping

$$y = r + \gamma V^\pi(\mathbf{s}')$$



$$y = r + \gamma \hat{V}^\pi(\mathbf{s}')$$



Biased



Low-variance

Temporal Difference

$$V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim p(\mathbf{s}, y | \pi)} \left[\|\underline{y_j} - V(\mathbf{s}_j)\|^2 \right]$$

$$y_j = r_j + \gamma V^i(\mathbf{s}'_j)$$

Temporal Difference

$$V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim p(\mathbf{s}, y | \pi)} \left[\underbrace{\|y_j - V(\mathbf{s}_j)\|^2}_{\text{“Temporal-Difference”}} \right]$$

“Temporal-Difference”

$$\underbrace{r_j + \gamma V^i(\mathbf{s}'_j)}_{\text{new prediction}} - \underbrace{V(\mathbf{s}_j)}_{\text{old prediction}}$$

new prediction

old prediction

Dynamic Programming

ALGORITHM: DP Policy Evaluation

- 1: input π : policy
 - 2: $\mathcal{B} \leftarrow$ collect trajectories τ from π
 - 3: $V^0 \leftarrow$ initialize value function

 - 4: **for** iteration $i = 0, \dots, k - 1$ **do**
 - 5: $\mathcal{D} \leftarrow \emptyset$ initialize dataset
 - 6: **for** $(\mathbf{s}_j, r_j, \mathbf{s}'_j)$ in \mathcal{B} **do**
 - 7: $y_j = r_j + \gamma V^i(\mathbf{s}'_j)$
 - 8: Store (\mathbf{s}_j, y_j) in dataset \mathcal{D}
 - 9: **end for**

 - 10: $V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim \mathcal{D}} [\|y_j - V(\mathbf{s}_j)\|^2]$
 - 11: **end for**

 - 12: return V^k
-

Dynamic Programming

ALGORITHM: DP Policy Evaluation

- 1: **input** π : policy
 - 2: $\mathcal{B} \leftarrow$ collect trajectories τ from π
 - 3: $V^0 \leftarrow$ initialize value function

 - 4: **for** iteration $i = 0, \dots, k - 1$ **do**
 - 5: $\mathcal{D} \leftarrow \emptyset$ initialize dataset
 - 6: **for** $(\mathbf{s}_j, r_j, \mathbf{s}'_j)$ in \mathcal{B} **do**
 - 7: $y_j = r_j + \gamma V^i(\mathbf{s}'_j)$
 - 8: Store (\mathbf{s}_j, y_j) in dataset \mathcal{D}
 - 9: **end for**

 - 10: $V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim \mathcal{D}} [\|y_j - V(\mathbf{s}_j)\|^2]$
 - 11: **end for**

 - 12: **return** V^k
-

Dynamic Programming

ALGORITHM: DP Policy Evaluation

- 1: **input** π : policy
 - 2: $\mathcal{B} \leftarrow$ collect trajectories τ from π
 - 3: $V^0 \leftarrow$ initialize value function

 - 4: **for** iteration $i = 0, \dots, k - 1$ **do**
 - 5: $\mathcal{D} \leftarrow \emptyset$ initialize dataset
 - 6: **for** $(\mathbf{s}_j, r_j, \mathbf{s}'_j)$ in \mathcal{B} **do**
 - 7: $y_j = r_j + \gamma V^i(\mathbf{s}'_j)$
 - 8: Store (\mathbf{s}_j, y_j) in dataset \mathcal{D}
 - 9: **end for**

 - 10: $V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim \mathcal{D}} [\|y_j - V(\mathbf{s}_j)\|^2]$
 - 11: **end for**

 - 12: **return** V^k
-

Dynamic Programming

ALGORITHM: DP Policy Evaluation

- 1: **input** π : policy
 - 2: $\mathcal{B} \leftarrow$ collect trajectories τ from π
 - 3: $V^0 \leftarrow$ initialize value function

 - 4: **for** iteration $i = 0, \dots, k - 1$ **do**
 - 5: $\mathcal{D} \leftarrow \emptyset$ initialize dataset
 - 6: **for** $(\mathbf{s}_j, r_j, \mathbf{s}'_j)$ in \mathcal{B} **do**
 - 7: $y_j = r_j + \gamma V^i(\mathbf{s}'_j)$
 - 8: Store (\mathbf{s}_j, y_j) in dataset \mathcal{D}
 - 9: **end for**

 - 10: $V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim \mathcal{D}} [\|y_j - V(\mathbf{s}_j)\|^2]$
 - 11: **end for**

 - 12: **return** V^k
-

Dynamic Programming

ALGORITHM: DP Policy Evaluation

- 1: **input** π : policy
 - 2: $\mathcal{B} \leftarrow$ collect trajectories τ from π
 - 3: $V^0 \leftarrow$ initialize value function

 - 4: **for** iteration $i = 0, \dots, k - 1$ **do**
 - 5: $\mathcal{D} \leftarrow \emptyset$ initialize dataset
 - 6: **for** $(\mathbf{s}_j, r_j, \mathbf{s}'_j)$ in \mathcal{B} **do**
 - 7: $y_j = r_j + \gamma V^i(\mathbf{s}'_j)$
 - 8: Store (\mathbf{s}_j, y_j) in dataset \mathcal{D}
 - 9: **end for**

 - 10: $V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim \mathcal{D}} [\|y_j - V(\mathbf{s}_j)\|^2]$
 - 11: **end for**

 - 12: **return** V^k
-

Dynamic Programming

ALGORITHM: DP Policy Evaluation

- 1: **input** π : policy
 - 2: $\mathcal{B} \leftarrow$ collect trajectories τ from π
 - 3: $V^0 \leftarrow$ initialize value function

 - 4: **for** iteration $i = 0, \dots, k - 1$ **do**
 - 5: $\mathcal{D} \leftarrow \emptyset$ initialize dataset
 - 6: **for** $(\mathbf{s}_j, r_j, \mathbf{s}'_j)$ in \mathcal{B} **do**
 - 7: $y_j = r_j + \gamma V^i(\mathbf{s}'_j)$
 - 8: Store (\mathbf{s}_j, y_j) in dataset \mathcal{D}
 - 9: **end for**

 - 10: $V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim \mathcal{D}} [\|y_j - V(\mathbf{s}_j)\|^2]$
 - 11: **end for**

 - 12: **return** V^k
-

Dynamic Programming

ALGORITHM: DP Policy Evaluation

- 1: **input** π : policy
 - 2: $\mathcal{B} \leftarrow$ collect trajectories τ from π
 - 3: $V^0 \leftarrow$ initialize value function

 - 4: **for** iteration $i = 0, \dots, k - 1$ **do**
 - 5: $\mathcal{D} \leftarrow \emptyset$ initialize dataset
 - 6: **for** $(\mathbf{s}_j, r_j, \mathbf{s}'_j)$ in \mathcal{B} **do**
 - 7: $y_j = r_j + \gamma V^i(\mathbf{s}'_j)$
 - 8: Store (\mathbf{s}_j, y_j) in dataset \mathcal{D}
 - 9: **end for**

 - 10: $V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim \mathcal{D}} [\|y_j - V(\mathbf{s}_j)\|^2]$
 - 11: **end for**

 - 12: **return** V^k
-

Dynamic Programming

ALGORITHM: DP Policy Evaluation

- 1: **input** π : policy
 - 2: $\mathcal{B} \leftarrow$ collect trajectories τ from π
 - 3: $V^0 \leftarrow$ initialize value function

 - 4: **for** iteration $i = 0, \dots, k - 1$ **do**
 - 5: $\mathcal{D} \leftarrow \emptyset$ initialize dataset
 - 6: **for** $(\mathbf{s}_j, r_j, \mathbf{s}'_j)$ in \mathcal{B} **do**
 - 7: $y_j = r_j + \gamma V^i(\mathbf{s}'_j)$
 - 8: Store (\mathbf{s}_j, y_j) in dataset \mathcal{D}
 - 9: **end for**

 - 10: $V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim \mathcal{D}} [\|y_j - V(\mathbf{s}_j)\|^2]$
 - 11: **end for**

 - 12: return V^k
-

Dynamic Programming

ALGORITHM: DP Policy Evaluation

- 1: **input** π : policy
 - 2: $\mathcal{B} \leftarrow$ collect trajectories τ from π
 - 3: $V^0 \leftarrow$ initialize value function

 - 4: **for** iteration $i = 0, \dots, k - 1$ **do**
 - 5: $\mathcal{D} \leftarrow \emptyset$ initialize dataset
 - 6: **for** $(\mathbf{s}_j, r_j, \mathbf{s}'_j)$ in \mathcal{B} **do**
 - 7: $y_j = r_j + \gamma V^i(\mathbf{s}'_j)$
 - 8: Store (\mathbf{s}_j, y_j) in dataset \mathcal{D}
 - 9: **end for**

 - 10: $V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim \mathcal{D}} [\|y_j - V(\mathbf{s}_j)\|^2]$
 - 11: **end for**

 - 12: **return** V^k
-

Dynamic Programming

ALGORITHM: DP Policy Evaluation

- 1: **input** π : policy
 - 2: $\mathcal{B} \leftarrow$ collect trajectories τ from π
 - 3: $V^0 \leftarrow$ initialize value function

 - 4: **for** iteration $i = 0, \dots, k - 1$ **do**
 - 5: $\mathcal{D} \leftarrow \emptyset$ initialize dataset
 - 6: **for** $(\mathbf{s}_j, r_j, \mathbf{s}'_j)$ in \mathcal{B} **do**
 - 7: $y_j = r_j + \gamma V^i(\mathbf{s}'_j)$
 - 8: Store (\mathbf{s}_j, y_j) in dataset \mathcal{D}
 - 9: **end for**

 - 10: $V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim \mathcal{D}} [\|y_j - V(\mathbf{s}_j)\|^2]$
 - 11: **end for**

 - 12: return V^k
-

Dynamic Programming

ALGORITHM: DP Policy Evaluation

- 1: **input** π : policy
 - 2: $\mathcal{B} \leftarrow$ collect trajectories τ from π
 - 3: $V^0 \leftarrow$ initialize value function

 - 4: **for** iteration $i = 0, \dots, k - 1$ **do**
 - 5: $\mathcal{D} \leftarrow \emptyset$ initialize dataset
 - 6: **for** $(\mathbf{s}_j, r_j, \mathbf{s}'_j)$ in \mathcal{B} **do**
 - 7: $y_j = r_j + \gamma V^i(\mathbf{s}'_j)$
 - 8: Store (\mathbf{s}_j, y_j) in dataset \mathcal{D}
 - 9: **end for**

 - 10: $V^{i+1} = \arg \min_V \mathbb{E}_{(\mathbf{s}_j, y_j) \sim \mathcal{D}} [\|y_j - V(\mathbf{s}_j)\|^2]$
 - 11: **end for**

 - 12: **return** V^k
-

Bias-Variance Tradeoff

Monte-Carlo

$$y = \sum_{t=0}^{T-1} \gamma^t r_t$$



Unbiased



High-variance

Bootstrap

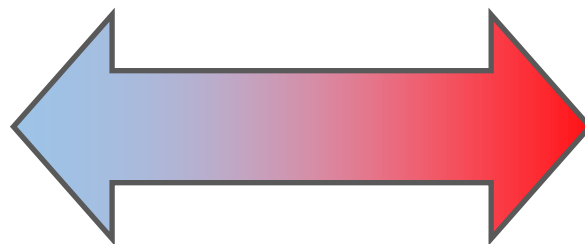
$$y = r_0 + \gamma \hat{V}^\pi(\mathbf{s}_1)$$



Biased



Low-variance



N-Step Bootstrapping

1-step bootstrap: $y = r_0 + \gamma \hat{V}^\pi(\mathbf{s}_1)$

2-step bootstrap: $y = r_0 + \gamma r_1 + \gamma^2 \hat{V}^\pi(\mathbf{s}_2)$

3-step bootstrap: $y = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 \hat{V}^\pi(\mathbf{s}_3)$



n-step bootstrap: $y = \sum_{t=0}^{n-1} \gamma^t r_t + \gamma^n \hat{V}^\pi(\mathbf{s}_n)$

Bias

N-Step Bootstrapping

1-step bootstrap: $y = r_0 + \gamma \hat{V}^\pi(\mathbf{s}_1)$

2-step bootstrap: $y = r_0 + \gamma r_1 + \gamma^2 \hat{V}^\pi(\mathbf{s}_2)$

3-step bootstrap: $y = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 \hat{V}^\pi(\mathbf{s}_3)$



n-step bootstrap: $y = \sum_{t=0}^{n-1} \gamma^t r_t + \gamma^n \hat{V}^\pi(\mathbf{s}_n)$

decays exponentially

N-Step Bootstrapping

1-step bootstrap: $y = r_0 + \gamma \hat{V}^\pi(\mathbf{s}_1)$

2-step bootstrap: $y = r_0 + \gamma r_1 + \gamma^2 \hat{V}^\pi(\mathbf{s}_2)$

3-step bootstrap: $y = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 \hat{V}^\pi(\mathbf{s}_3)$



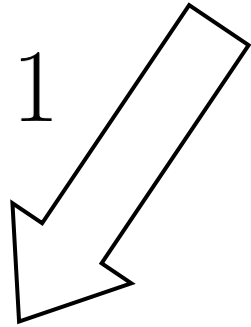
n-step bootstrap: $y = \sum_{t=0}^{n-1} \gamma^t r_t + \gamma^n \hat{V}^\pi(\mathbf{s}_n)$

$n \rightarrow \infty, \quad \gamma^n \rightarrow 0$

N-Step Bootstrapping

$$y = \sum_{t=0}^{n-1} \gamma^t r_t + \gamma^n \hat{V}^\pi(\mathbf{s}_n)$$

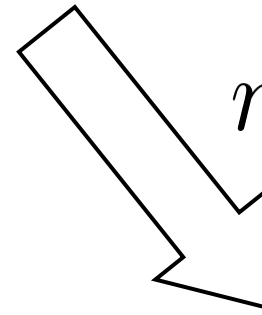
$$n = 1$$



Bootstrap

$$y = r_0 + \gamma \hat{V}^\pi(\mathbf{s}_1)$$

$$n \rightarrow \infty$$



Monte-Carlo

$$y = \sum_{t=0}^{T-1} \gamma^t r_t$$

N-Step Bootstrapping

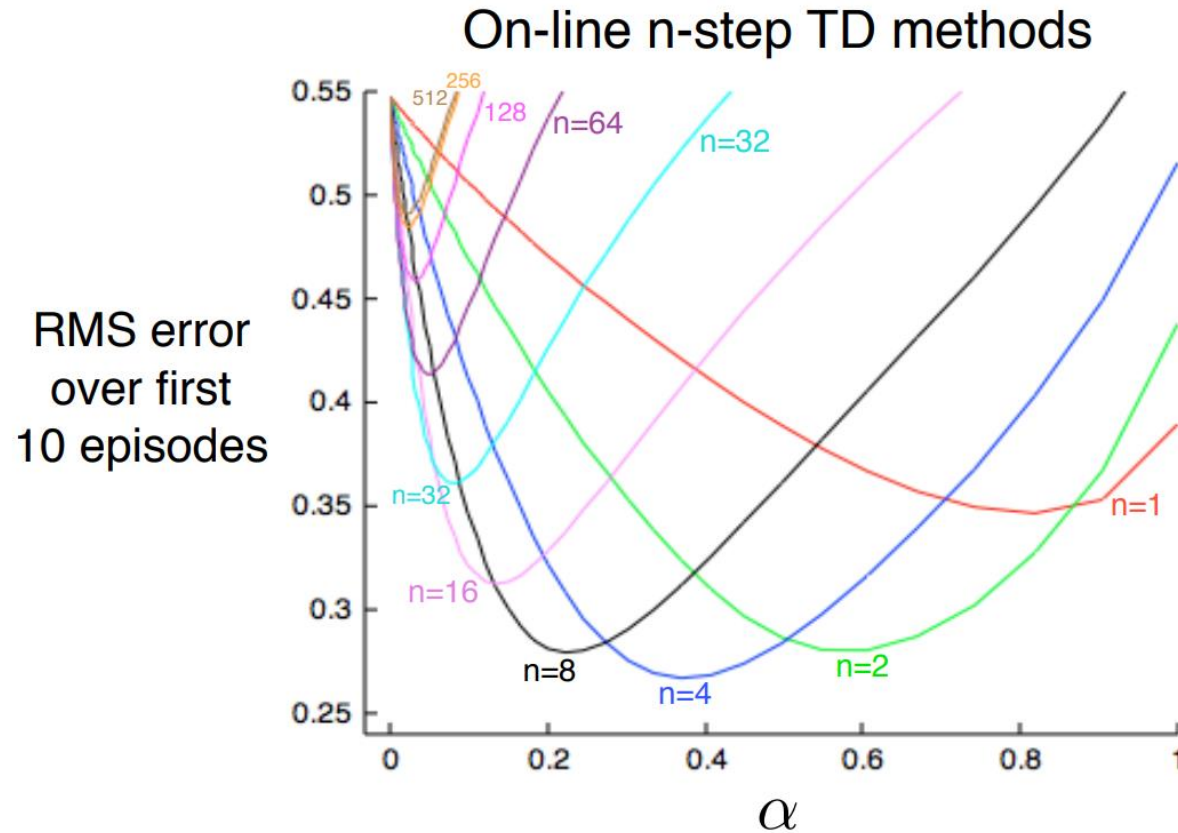
Small n :

- High bias
- Low variance

Large n :

- Low bias
- High variance

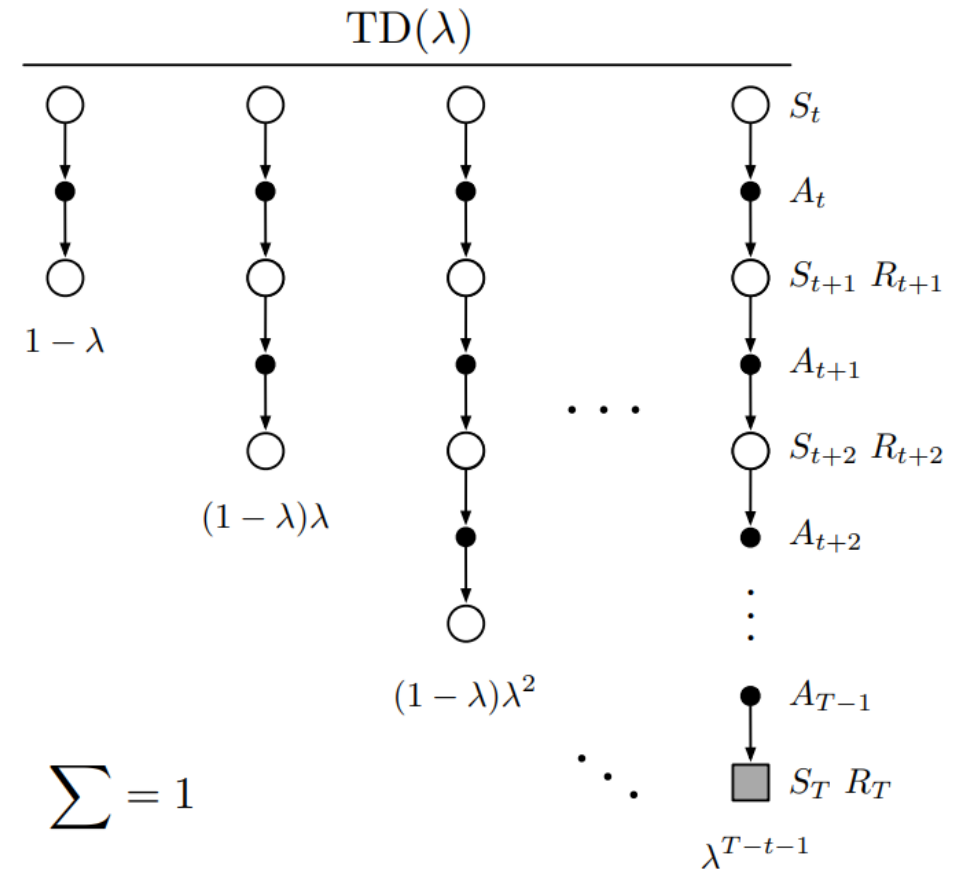
N-Step Bootstrapping



Reinforcement Learning: An Introduction 2nd Ed
[Sutton and Barto 2018]

TD(λ)

- How do we pick n ?
- TD(λ):
 - Average multi-step returns across **all** lengths n !

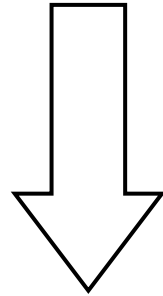


Reinforcement Learning: An Introduction
[Sutton and Barto 2018]

Optimal Policies

Optimal Policy

$$\pi^* = \arg \max_{\pi} J(\pi)$$



$$J(\pi^*) \geq J(\pi) \quad \text{for all } \pi$$

Optimal Value Function

$$V^*(\mathbf{s})$$

Optimal Value Function

$$V^*(\mathbf{s}) \geq V^\pi(\mathbf{s}) \quad \text{for all } \pi \text{ and } \mathbf{s}$$

$$V^*(\mathbf{s}) = \mathbb{E}_{\mathbf{a} \sim \pi^*(\mathbf{a}|\mathbf{s})} \mathbb{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} [r + \gamma V^*(\mathbf{s}')]]$$

Bellman equation of the
optimal policy

Optimal Value Function

- For a given MDP
 - The optimal value function is unique
 - Can be many optimal policies
 - Given an optimal policy, can recover the optimal value function
 - Given the optimal value function, can recover an optimal policy

Testing

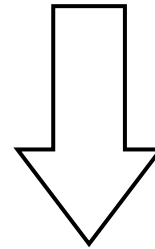
$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

Biased towards earlier steps

Testing

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

No discount during testing



$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} r_t \right]$$

Overview

- Policy Evaluation
- Value Functions
- Monte-Carlo Methods
- Dynamic Programming Methods
- Optimal Policies