

# COMS W4701: Artificial Intelligence

## Lecture 5a: RL Prediction

Tony Dear, Ph.D.

Department of Computer Science  
School of Engineering and Applied Sciences

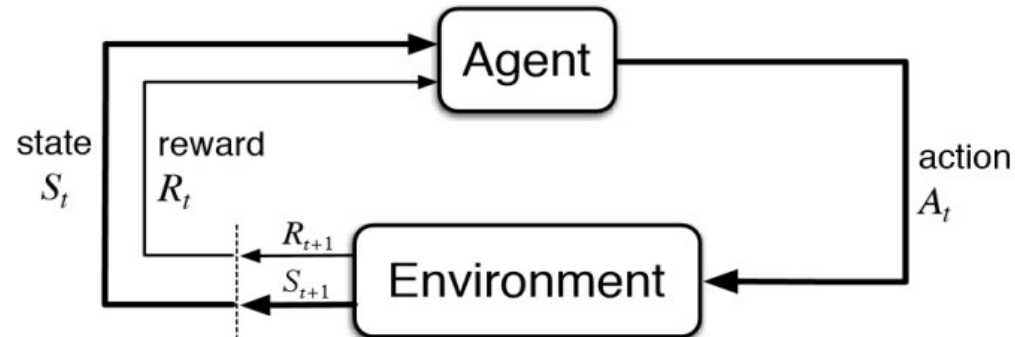
# Today

---

- Reinforcement learning
- Monte Carlo (MC) prediction
- Temporal difference (TD) prediction
- Comparing DP, MC, and TD

# Learning from Experience

- Dynamic programming requires knowledge of environment *model* (transition and reward functions), but often inaccessible or intractable
- **Reinforcement learning:** Find optimal policies through experience
- Interact with environment, receive rewards, and formulate policies



# Dimensions of RL

---

- *Model-based* methods learn an approximation of the underlying model
- *Model-free* methods directly learn policies or value functions
- Can be useful even when model is known but DP is intractable
  
- *Prediction*: Given a fixed policy  $\pi$ , learn  $V^\pi$
- *Control*: Learn an optimal policy  $\pi^*$ , *state-action* value function  $Q^*$
  
- Model-free methods for both prediction and control include *Monte Carlo* and *temporal difference* algorithms

# Monte Carlo Methods

---

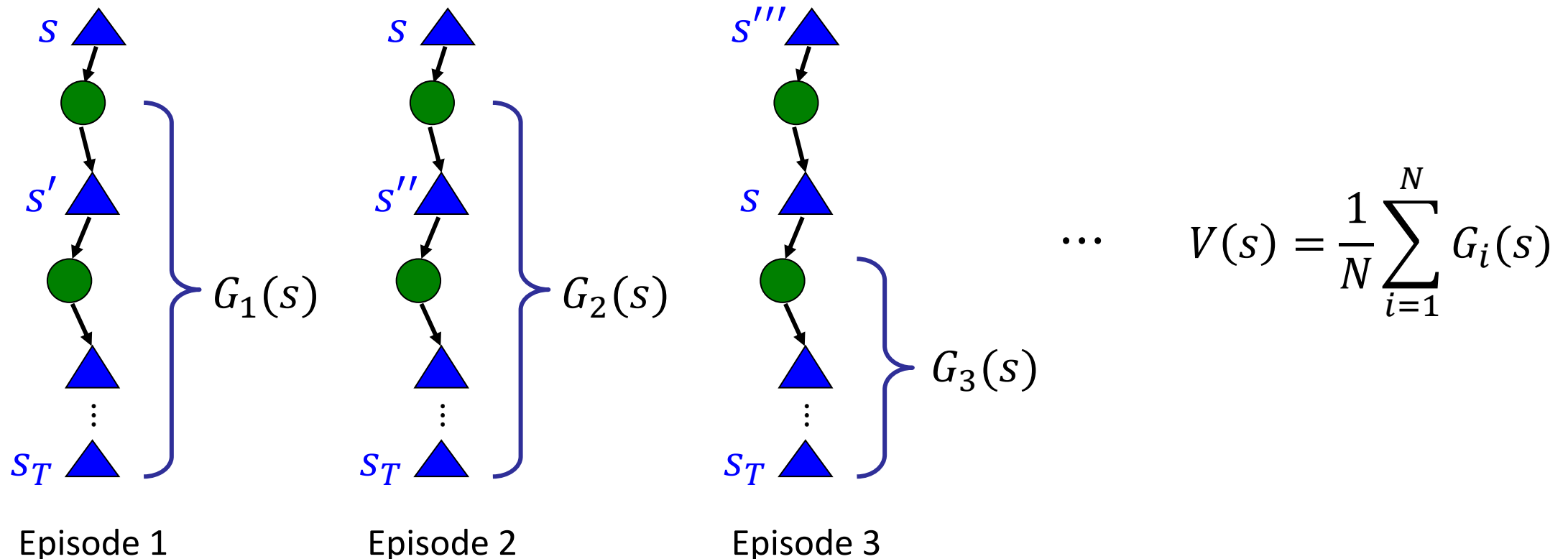
- **Monte Carlo** methods: Generate sampled experience and average them for different states and actions
- Recall the definition of value function for a fixed policy  $\pi$ :

$$V^{\pi}(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1}) \right]$$

- Idea: Approximate the expectation by taking *averages* of sample reward sequences over multiple *episodes*

# State Value Estimation

- Idea:  $V(s)$  can be estimated by averaging utilities observed *after* visiting  $s$
- Think of each sample as a path from a given node to a leaf node in search tree

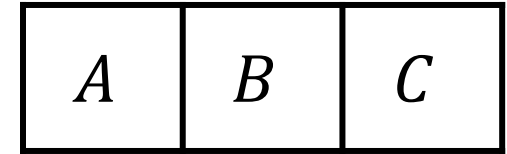


# First-Visit MC Prediction

- **MC prediction:** Estimate state values by averaging utilities over multiple episodes
- *First-visit* MC: A value is estimated after first visit to state within episode
- **Initialize**  $V^\pi(s) \leftarrow 0$ ,  $N(s) \leftarrow 0$  for each state  $s \in S$
- **Loop:**
  - **Generate** episode  $E$  following  $\pi$ :  $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$
  - **For** each state  $s$ :
    - $G \leftarrow \sum_{j=t+1}^T \gamma^{j-(t+1)} r_j$ , where  $s_t$  is first occurrence of  $s$  in  $E$
    - $V^\pi(s) \leftarrow \frac{1}{N(s)+1} (N(s) \times V^\pi(s) + G)$
    - $N(s) \leftarrow N(s) + 1$

# Example: Mini-Gridworld

- States:  $A, B, C$ ; actions:  $L, R$ ; rewards received upon entering each state
- Policy:  $\pi(s) = L$  for all states  $s$
- Suppose we use episodes with 5 transitions



- Episode 1:  $(A, +3, A, -2, B, +1, C, -2, B, +3)$

$$\gamma = 0.5$$

$$V^\pi(A) \leftarrow G(A) = 3 + \gamma(-2) + \gamma^2(1) + \gamma^3(-2) + \gamma^4(3) = 2.1875$$

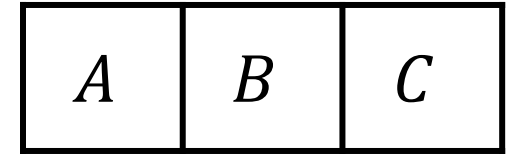
$$V^\pi(B) \leftarrow G(B) = 1 + \gamma(-2) + \gamma^2(3) = 0.75$$

$$V^\pi(C) \leftarrow G(C) = -2 + \gamma(3) = -0.5$$



# Example: Mini-Gridworld

- States:  $A, B, C$ ; actions:  $L, R$ ; rewards received upon entering each state
- Policy:  $\pi(s) = L$  for all states  $s$
- Suppose we use episodes with 5 transitions



- Episode 2:  $(A, -2, B, +3, A, -2, B, +1, C, -2)$

$\gamma = 0.5$

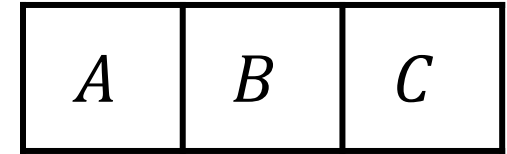
$$V^\pi(A) \leftarrow \frac{1}{2} (V^\pi(A) + G(A)) = \frac{1}{2} (2.1875 - 1) = 0.59375$$

$$V^\pi(B) \leftarrow \frac{1}{2} (V^\pi(B) + G(B)) = \frac{1}{2} (0.75 + 2) = 1.375$$

$$V^\pi(C) \leftarrow \frac{1}{2} (V^\pi(C) + G(C)) = \frac{1}{2} (-0.5 - 2) = -1.25$$

# Example: Mini-Gridworld

- States:  $A, B, C$ ; actions:  $L, R$ ; rewards received upon entering each state
- Policy:  $\pi(s) = L$  for all states  $s$
- Suppose we use episodes with 5 transitions



- Episode 3:  $(C, +1, C, -2, B, +3, A, -2, B, +3)$

$$\gamma = 0.5$$

$$V^\pi(A) \leftarrow \frac{1}{3} (2V^\pi(A) + G(A)) = \frac{1}{3} (2(0.59375) - 0.5) = 0.229$$

$$V^\pi(B) \leftarrow \frac{1}{3} (2V^\pi(B) + G(B)) = \frac{1}{3} (2(1.375) + 2.75) = 1.833$$

$$V^\pi(C) \leftarrow \frac{1}{3} (2V^\pi(C) + G(C)) = \frac{1}{3} (2(-1.25) + 0.6875) = -0.604$$

# Finer Points

---

- Different states will have different visited frequencies, but all states will be visited infinitely often in the limit—values will converge to true  $V^\pi$
- Estimates of different state values are *independent* (in contrast to DP)
- Accuracy of  $V^\pi(s)$  does *not* depend on accuracy of  $V^\pi(s')$
- Result: Computational complexity of estimating specific state values is independent of state space size!
- Can choose to focus on certain states and ignore others

# Temporal-Difference Learning

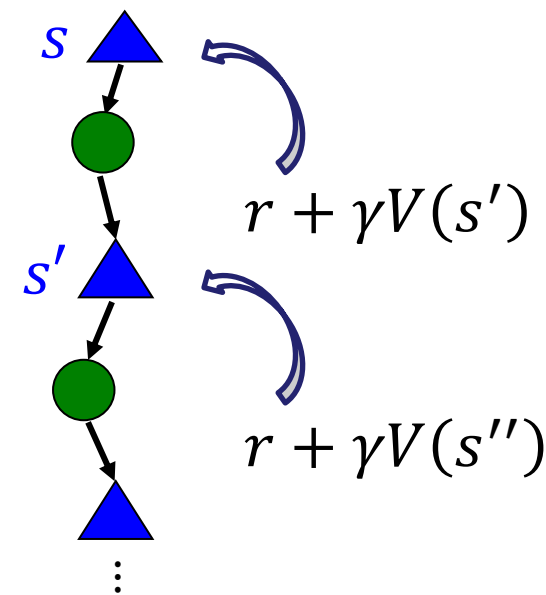
- MC requires episodic structure—what about infinite horizon problems?
- State values in MC are estimated entirely independently of each other
- Maybe we can borrow the idea of the *Bellman update* from dynamic programming
- **One-step TD ( $TD(0)$ ):** We can replace the *target* term with the sum of immediate reward with discounted successor state value
- We can update  $V^\pi(s)$  *immediately* before the episode ends

$$V^\pi(s) \leftarrow V^\pi(s) + \alpha \left( \overbrace{r_{t+1} + \gamma V^\pi(s')}^{\text{TD error } \delta_t} - V^\pi(s) \right)$$

Target

# $TD(0)$ for Prediction

- **Given:** Policy  $\pi$ , *learning rate*  $\alpha$  between 0 and 1
- **Initialize**  $V^\pi(s) \leftarrow 0$
- **Loop:**
  - **Initialize** starting state  $s$  if needed
  - **Generate** sequence  $(s, \pi(s), r, s')$
  - $V^\pi(s) \leftarrow V^\pi(s) + \alpha(r + \gamma V^\pi(s') - V^\pi(s))$
  - $s \leftarrow s'$



# Example: Mini-Gridworld

- All values initialized to 0;  $\gamma = 0.8$ ,  $\alpha = 0.5$
- Policy to evaluate:  $\pi(s) = L$  for all states

$A$	$B$	$C$
-----	-----	-----

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha(r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t))$$

- Observed state and reward sequence:  $(A, +3, A)$

$$\begin{pmatrix} V^\pi(A) \\ V^\pi(B) \\ V^\pi(C) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$V^\pi(A) \leftarrow V^\pi(A) + \alpha(r + \gamma V^\pi(A) - V^\pi(A))$$

$$V^\pi(A) \leftarrow 0 + 0.5(3 + 0.8(0) - 0) = 1.5$$

# Example: Mini-Gridworld

- All values initialized to 0;  $\gamma = 0.8$ ,  $\alpha = 0.5$
- Policy to evaluate:  $\pi(s) = L$  for all states

$A$	$B$	$C$
-----	-----	-----

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha(r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t))$$

- Observed state and reward sequence:  $(A, -2, B)$

$$\begin{pmatrix} V^\pi(A) \\ V^\pi(B) \\ V^\pi(C) \end{pmatrix} = \begin{pmatrix} 1.5 \\ 0 \\ 0 \end{pmatrix} \quad \begin{aligned} V^\pi(A) &\leftarrow V^\pi(A) + \alpha(r + \gamma V^\pi(B) - V^\pi(A)) \\ V^\pi(A) &\leftarrow 1.5 + 0.5(-2 + 0.8(0) - 1.5) = -0.25 \end{aligned}$$

# Example: Mini-Gridworld

- All values initialized to 0;  $\gamma = 0.8$ ,  $\alpha = 0.5$
- Policy to evaluate:  $\pi(s) = L$  for all states

$A$	$B$	$C$
-----	-----	-----

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha(r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t))$$

- Observed state and reward sequence:  $(B, +1, C)$

$$\begin{pmatrix} V^\pi(A) \\ V^\pi(B) \\ V^\pi(C) \end{pmatrix} = \begin{pmatrix} -0.25 \\ 0 \\ 0 \end{pmatrix} \quad \begin{aligned} V^\pi(B) &\leftarrow V^\pi(B) + \alpha(r + \gamma V^\pi(C) - V^\pi(B)) \\ V^\pi(B) &\leftarrow 0 + 0.5(1 + 0.8(0) - 0) = 0.5 \end{aligned}$$



# Example: Mini-Gridworld

- All values initialized to 0;  $\gamma = 0.8$ ,  $\alpha = 0.5$
- Policy to evaluate:  $\pi(s) = L$  for all states

$A$	$B$	$C$
-----	-----	-----

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha(r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t))$$

- Observed state and reward sequence:  $(C, -2, B)$

$$\begin{pmatrix} V^\pi(A) \\ V^\pi(B) \\ V^\pi(C) \end{pmatrix} = \begin{pmatrix} -0.25 \\ 0.5 \\ 0 \end{pmatrix} \quad \begin{aligned} V^\pi(C) &\leftarrow V^\pi(C) + \alpha(r + \gamma V^\pi(B) - V^\pi(C)) \\ V^\pi(C) &\leftarrow 0 + 0.5(-2 + 0.8(0.5) - 0) = -0.8 \end{aligned}$$

# Finer Points

---

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha \delta_t$$

- TD methods perform updates immediately with no episodic structure (MC)
- Useful if problems have long episodes or are continuing tasks
- If  $\alpha$  is constant,  $V^{\pi}$  will continue jumping around with each new target
- May be desirable if problem is nonstationary
- We can also shrink  $\alpha$  over time if we want  $V^{\pi}$  to converge

# Optimality of $TD(0)$

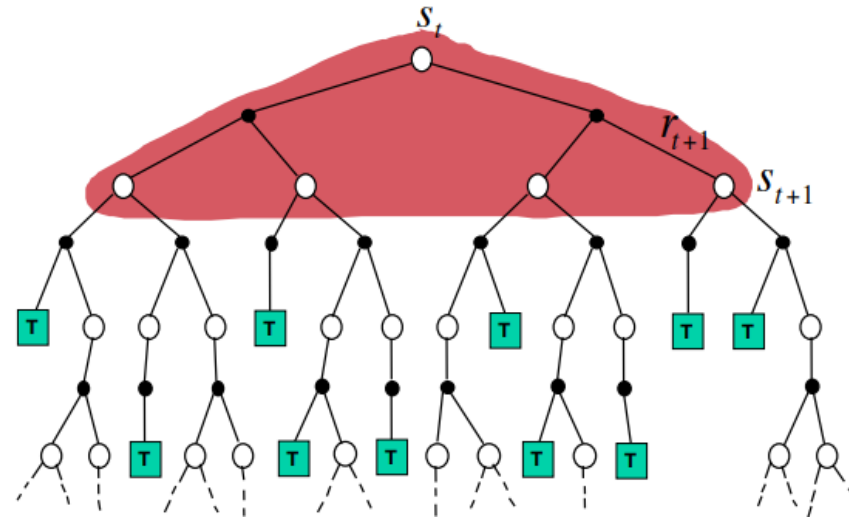
- Suppose we use a *sequence* of  $\alpha_n$  step size values over time
- Stochastic approximation theory assures us that  $TD(0)$  if  $\alpha_n$  meets the following conditions:

$$\sum_{n=1}^{\infty} \alpha_n = \infty \quad \sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

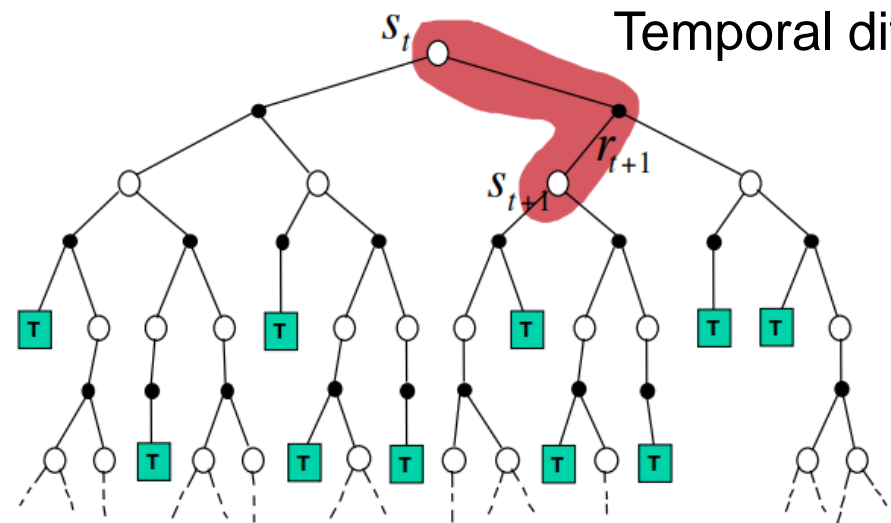
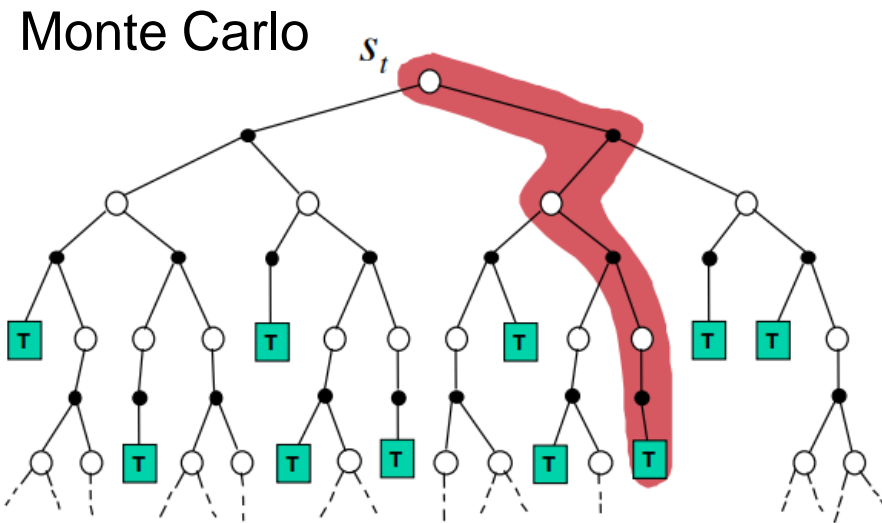
- First condition ensures that initial steps are large enough to overcome initial conditions or random fluctuations
- Second condition ensures that updates do eventually shrink to 0
- These conditions are satisfied by the sample averaging method  $\alpha_n = \frac{1}{n}$

# MDP Method Comparison

<https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf>



Dynamic programming



# Summary

---

- Reinforcement learning is used to estimate values/policies from data
- Instead of using underlying models, agent observes state-action-rewards
- Prediction problem: Evaluate a given policy
- Monte Carlo methods estimate by averaging samples of episodic returns
- Temporal difference methods bootstrap by using estimates to inform other estimates