

Advanced Topics in Reinforcement Learning

Lecture 6: Temporal Difference Learning

Josiah Hanna

University of Wisconsin — Madison

Announcements

- Homework due October 8 at 9:30AM (minute class starts)
- Read chapter 8 for next week.
- Project proposals due October 1st.
- Let me know ASAP if trouble finding a group or want to work in group of three.

Learning Outcomes

After this week, you will be able to:

1. Explain how TD-learning combines ideas from dynamic programming and Monte Carlo value function learning.
2. Implement TD-learning algorithms such as TD(0), Q-learning, and SARSA.
3. Compare and contrast on- and off-policy TD-learning methods for control.
4. Compare and contrast TD and Monte Carlo value function learning.

Review

- Dynamic Programming Methods
 - Require a model of the environment (know p).
 - Bootstrap, i.e., use the current value function estimate, v_k , to compute v_{k+1} .
- Monte Carlo Methods
 - No need for an environment model.
 - No bootstrapping and wait until termination to update v_k to v_{k+1} .

TD(0) Prediction

- Basic learning rule: $V(S_t) \leftarrow V(S_t) + \alpha[Y_t - V(S_t)]$.

- Y_t is a learning target.

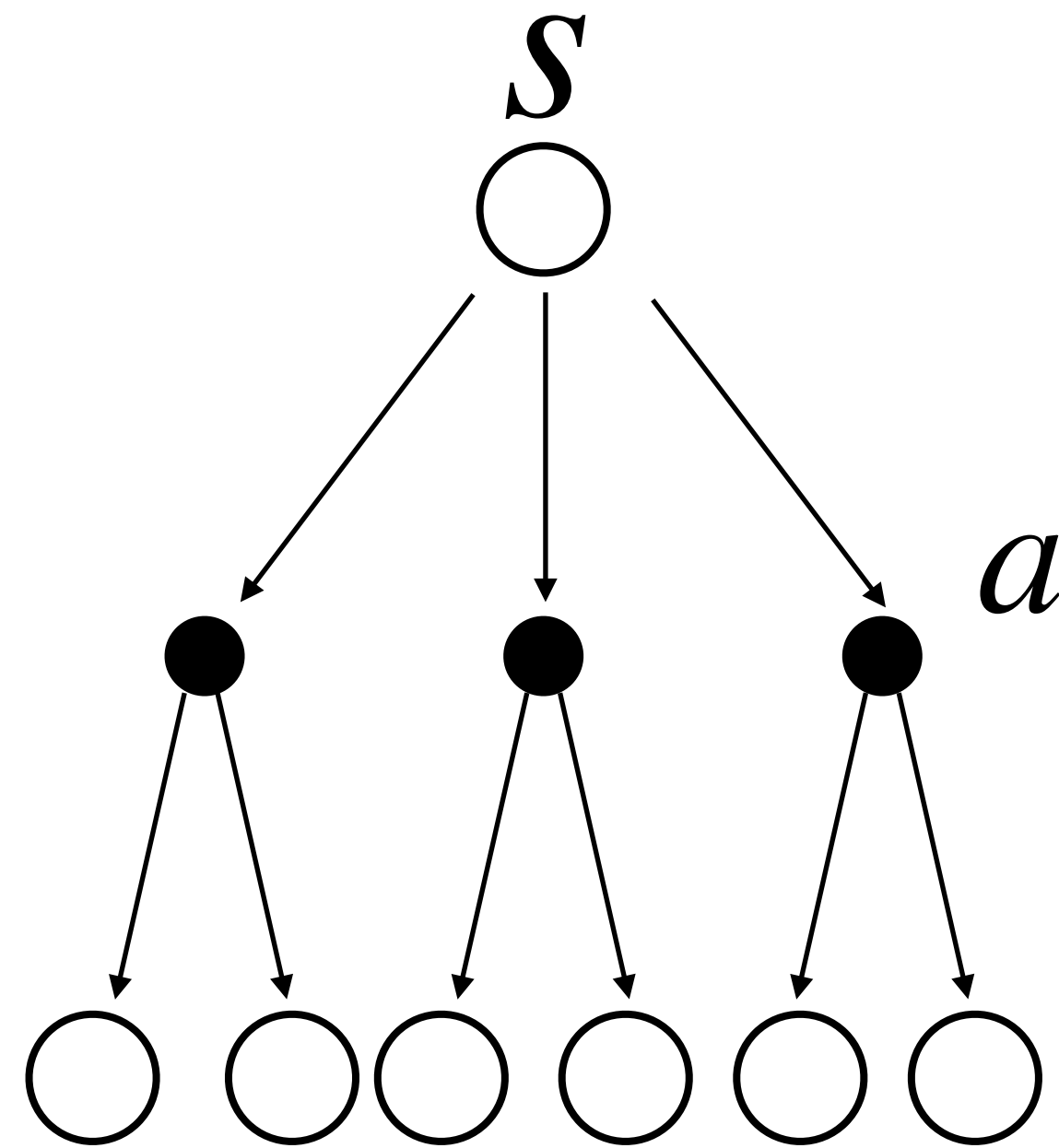
$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

TD-error

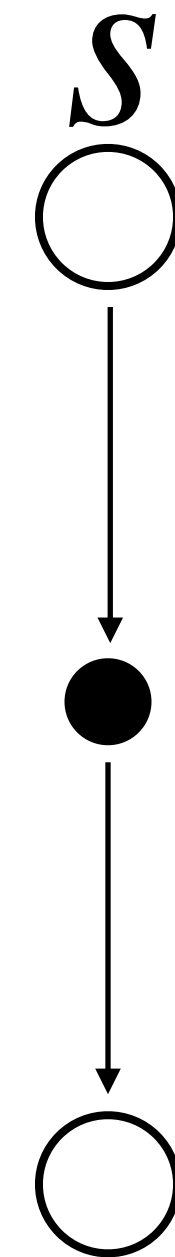
- Monte Carlo update: $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$
- TD update: $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$
- Compare to dynamic programming:

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

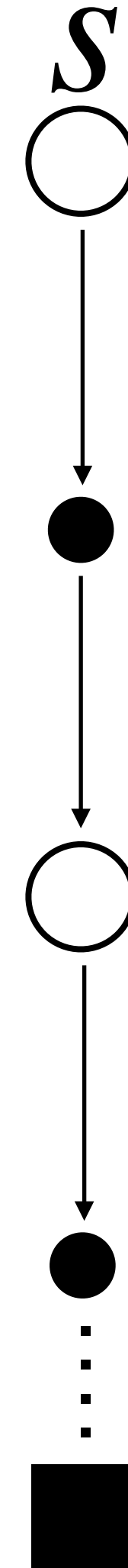
Back-up Diagrams



Dynamic Programming



TD(0)



Monte Carlo

Driving Home Example

$$\text{Cumulative reward } V(s) = \sum_{t=0}^{t'} R_t + V(S_{t'})$$

<i>State</i>	<i>Elapsed Time (minutes)</i>	<i>Predicted Time to Go</i>	<i>Predicted Total Time</i>
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

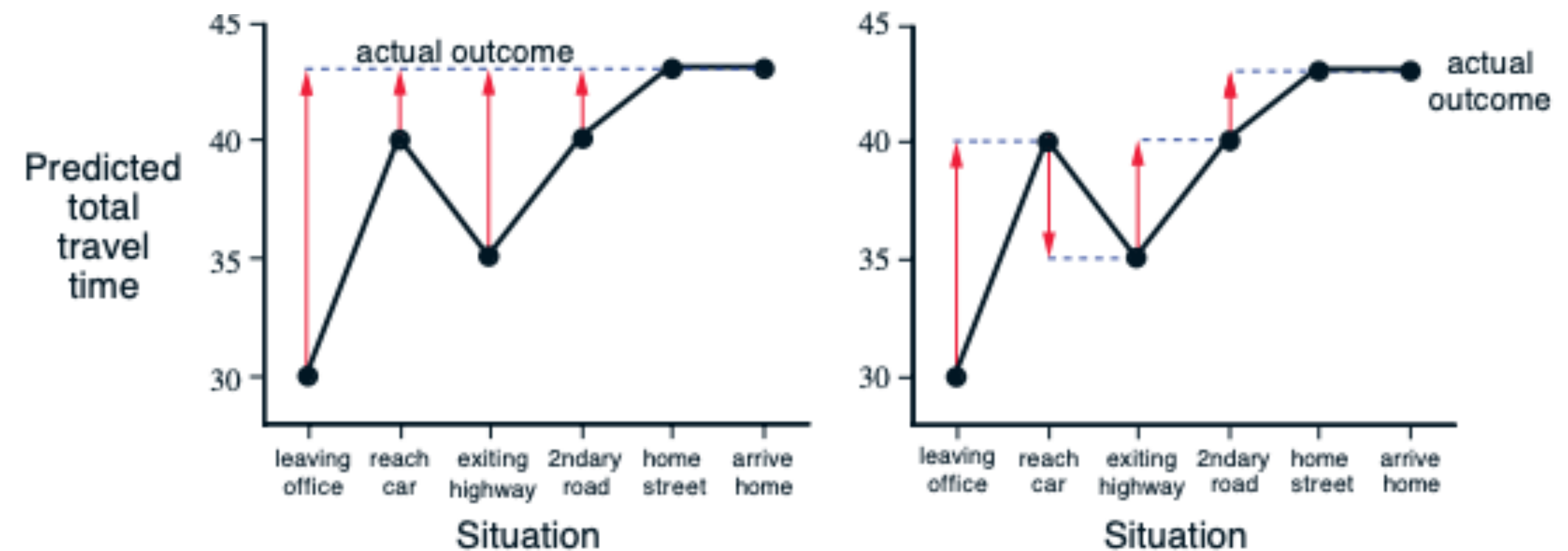
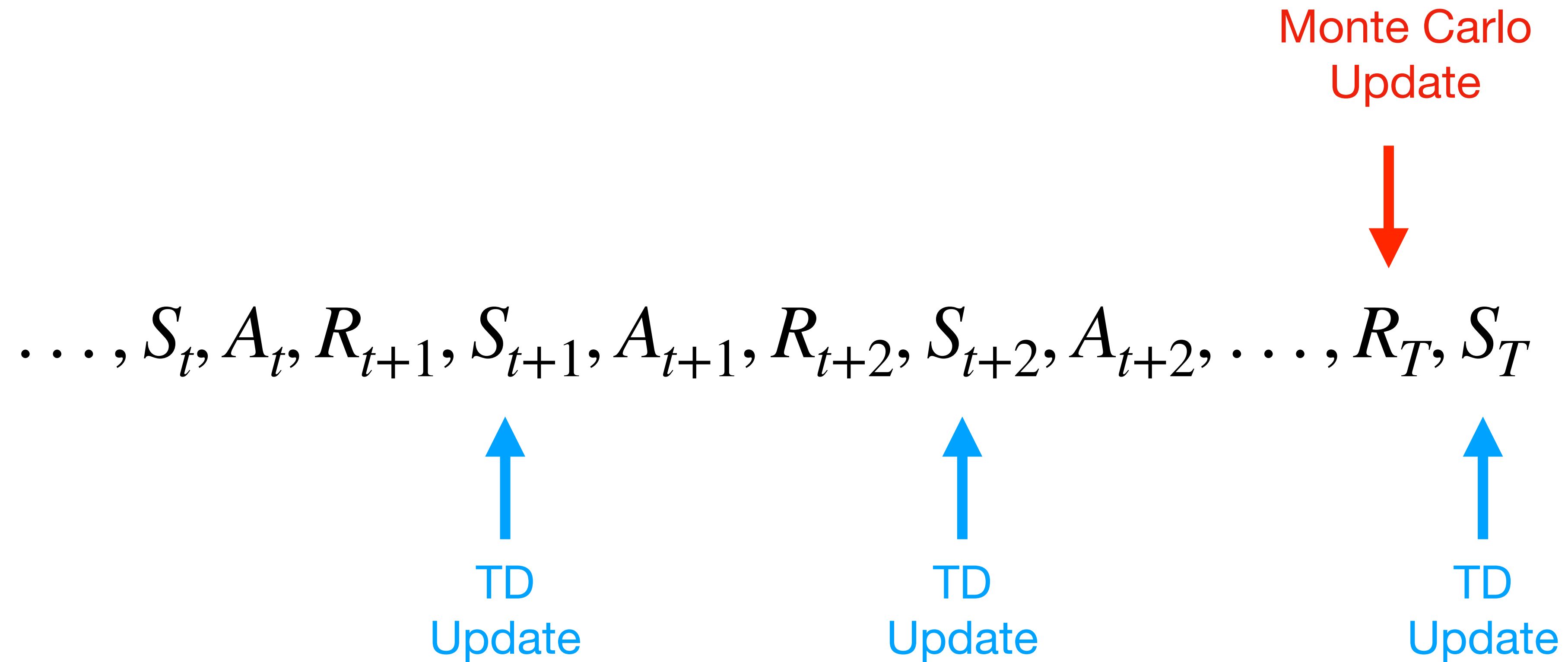


Figure 6.1: Changes recommended in the driving home example by Monte Carlo methods (left) and TD methods (right).

Update Frequency



TD(0) / Monte Carlo

- Neither require an environment model.
- TD methods are online and incremental.
 - Learning happens at every time-step and only requires constant storage.
 - Can be used for continuing tasks, i.e., no termination.
- (To be shown) More robust to off-policy exploratory actions.
- Monte Carlo methods rely less on the Markov property.
- Monte Carlo methods may propagate values faster.

N-Step Returns

- Possible to combine Monte Carlo and TD-Learning.
- General Update: $V(S_t) \leftarrow V(S_t) + \alpha[Y_t - V(S_t)]$
- Consider $Y_t := R_{t+1} + \gamma R_{t+2} + \dots + \gamma^n V(S_{t+n})$
- TD(0) is $n = 1$ and Monte Carlo is $n = \infty$; TD(λ) blends between extremes.

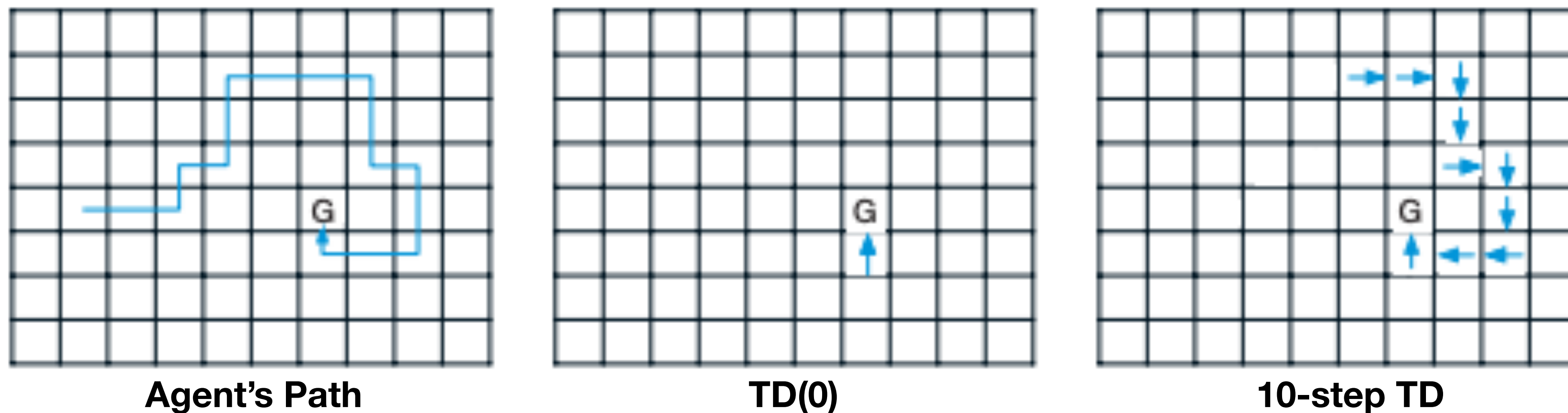


Figure 7.4

Convergence

- TD(0) and Monte Carlo both converge but TD methods are usually faster when using a constant step-size.

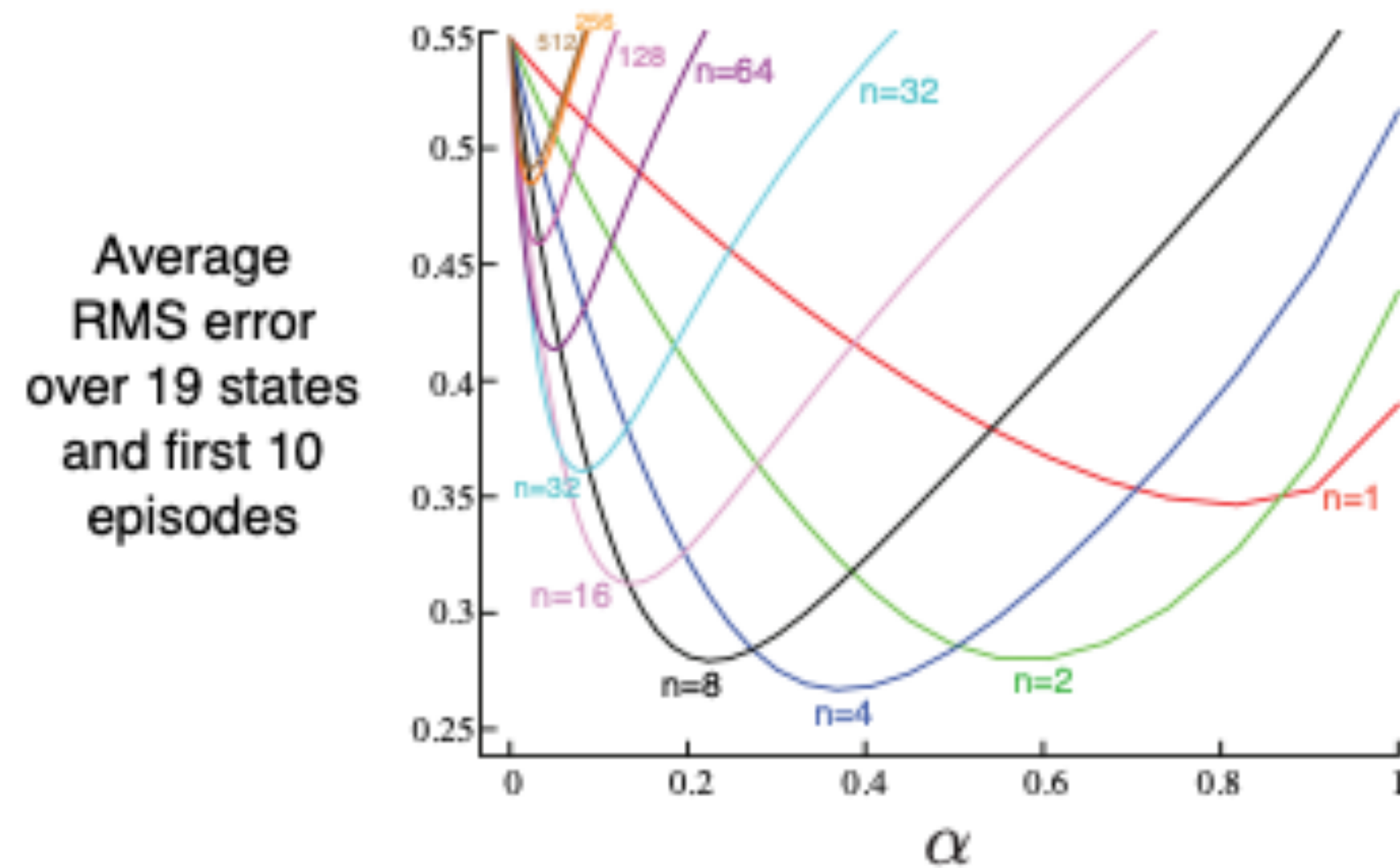
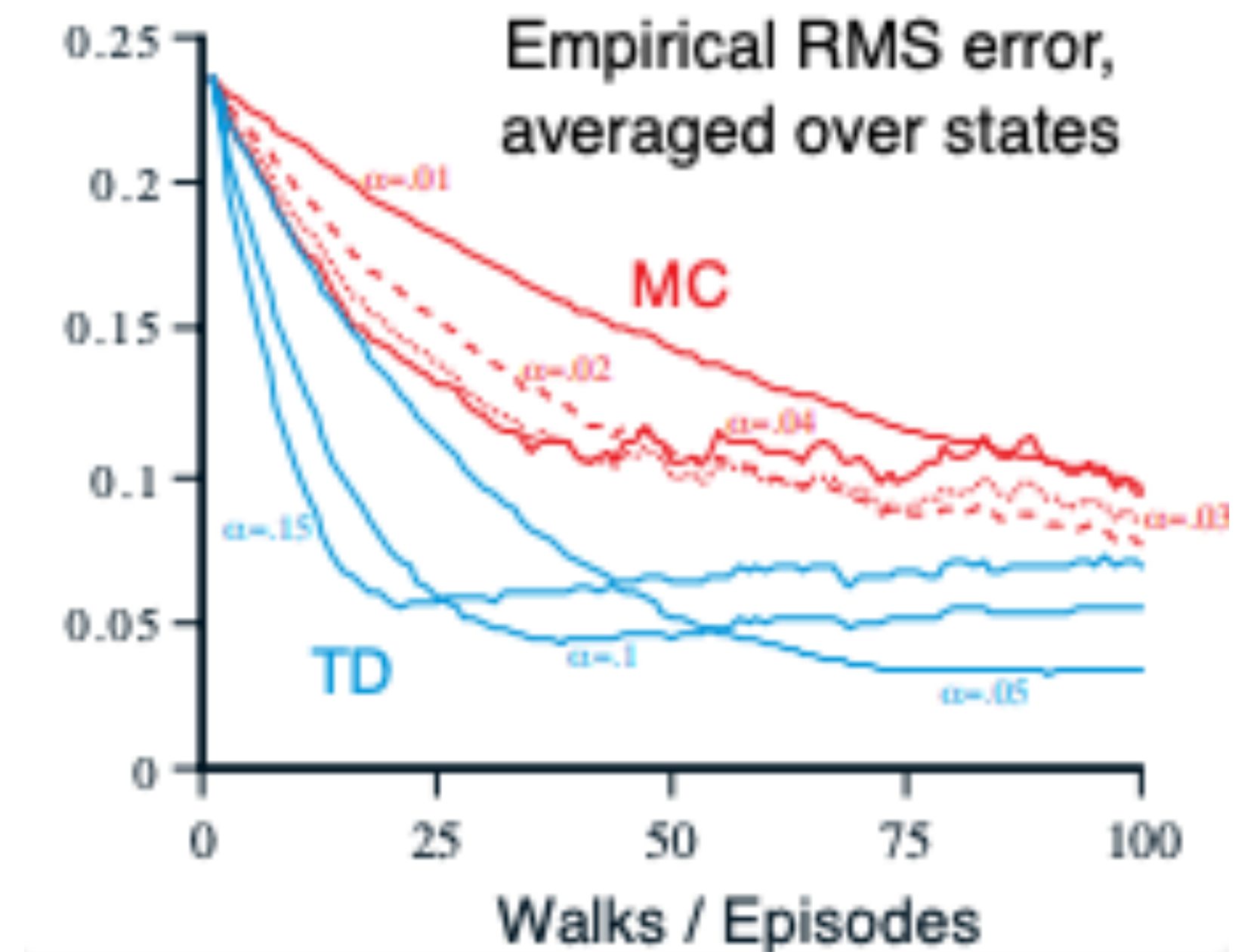


Figure 7.2: Performance of n -step TD methods as a function of α , for various values of n , on a 19-state random walk task (Example 7.1).



Certainty Equivalence Updating

Data:

A, 0, B, 0

B, 1

B, 1

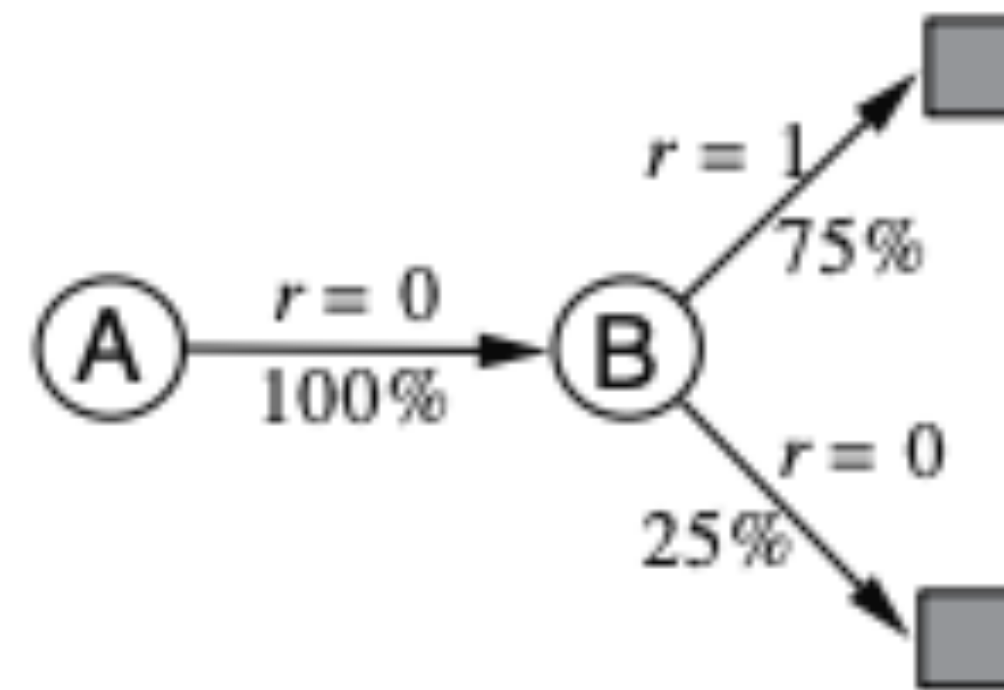
B, 1

B, 1

B, 1

B, 1

B, 0



Certainty Equivalence Updating

- Consider a **Markov reward process** — not an MDP!
 - If policy is fixed (as in prediction) then have a Markov chain on states.
- Given a batch of data $D = \{(s_i, r_i, s'_i)\}$, compute the value function.

- For TD(0), update value function with the sum of all increments:

Number of times we observed s, r, s'

- $$v_{k+1}(s) \leftarrow v_k(s) + \alpha \sum_{s',r} \#(s, r, s') [r + \gamma v_k(s') - v_k(s)]$$

Estimate of p

- $$= v_k(s) + \alpha' \sum_{s',r} \frac{\#(s, r, s')}{\#(s)} [r + \gamma v_k(s') - v_k(s)]$$

- $$= \alpha' \sum_{s',r} \hat{p}(s', r | s) [r + \gamma v_k(s')]$$

This is dynamic programming with estimated transitions!

Note: For MDPs, see Reducing Sampling Error in Batch Temporal Difference Learning [Pavse et al. 2020]

SARSA

- Same generalized policy iteration scheme from last week.

- Evaluate π_k .

- Make π_{k+1} greedy with respect to π_k .

Note: if S_{t+1} is terminal then
 $Q(S_{t+1}, A_{t+1}) = 0$.

- Now, use TD(0) to learn action-values:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

- Is this on- or off-policy?
- What does generalized policy iteration with TD-learned action-values (SARSA) and ϵ -greedy exploration converge to?

Q-Learning

- SARSA is approximating policy iteration. What about value iteration?

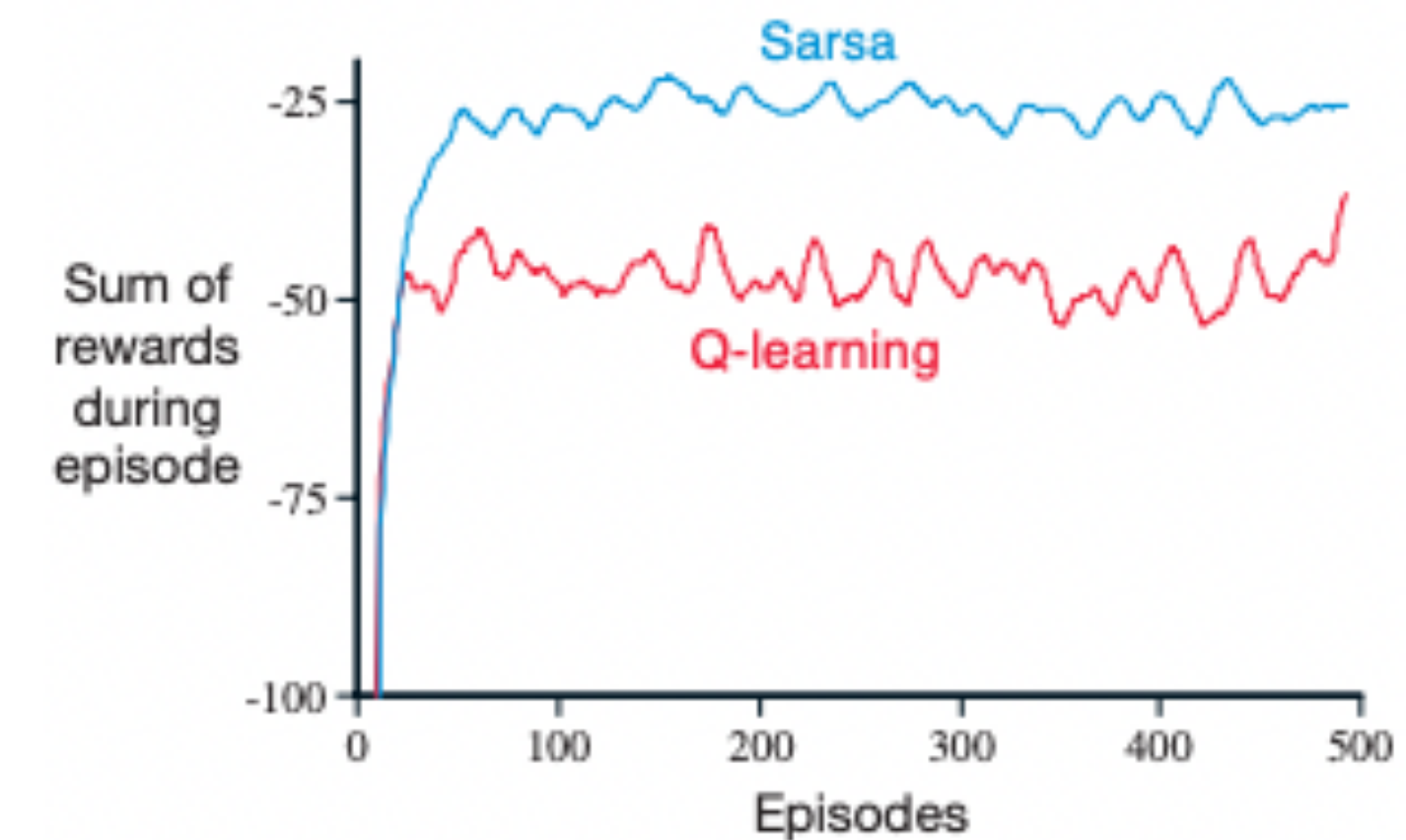
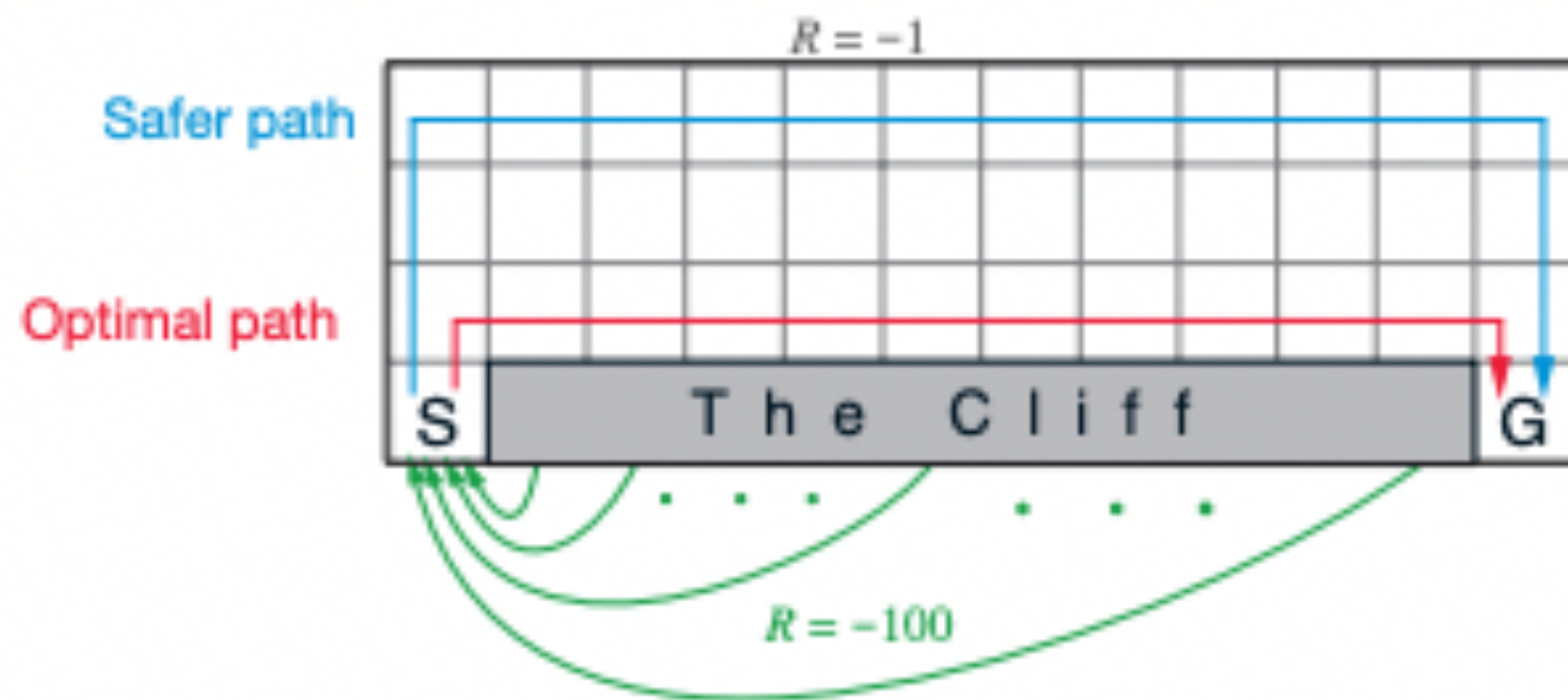
- Q-learning update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)]$$

- Is this update on- or off-policy?
 - Off-policy: can follow any policy (e.g., ϵ -greedy) while learning $q_\star$.
 - “Follow a policy derived from Q” — still off-policy!
- What does the Q-learning update converge to?
 - $q_\star$

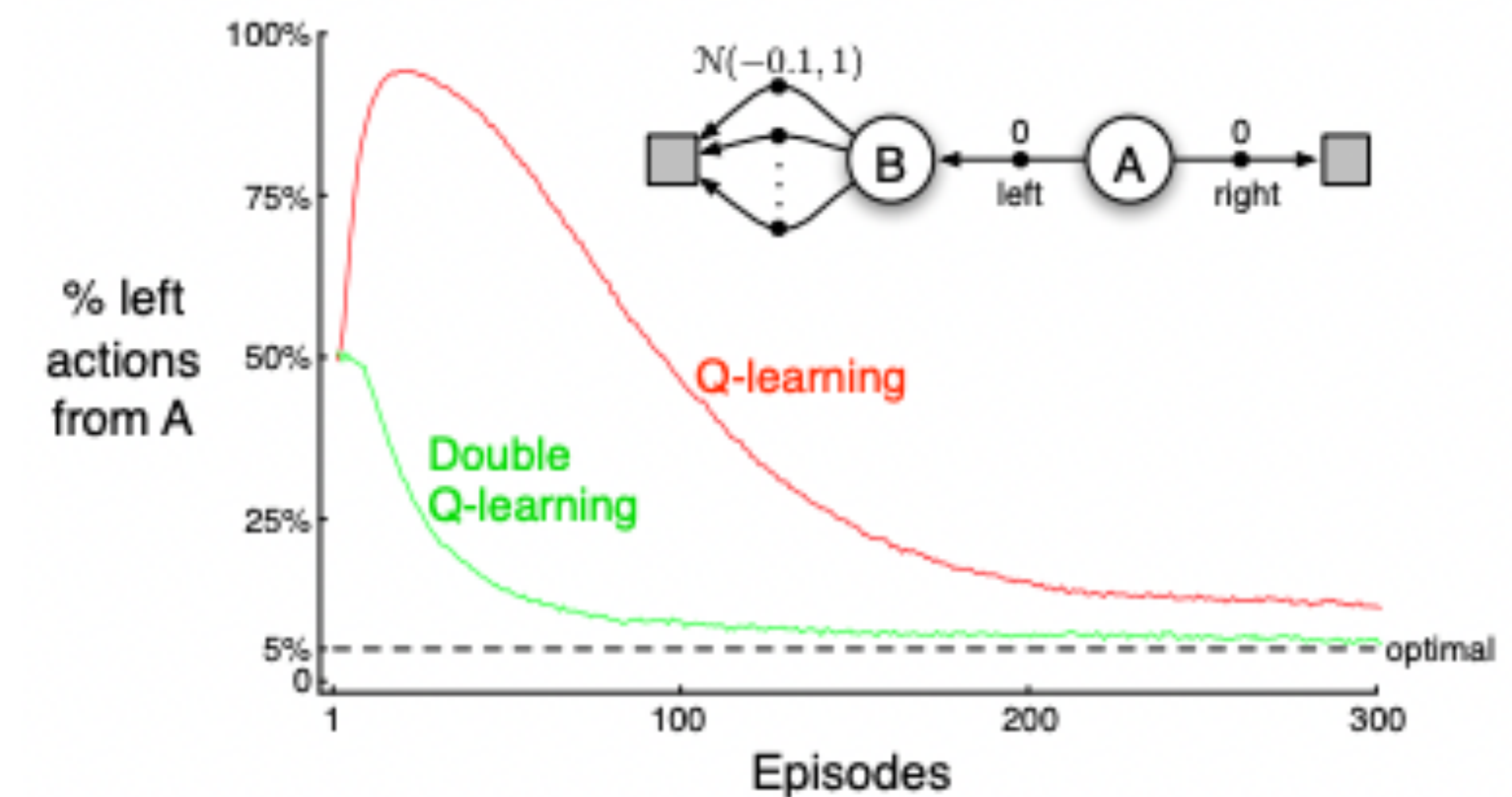
Q-Learning or SARSA?

- Q-learning is off-policy; SARSA is on-policy.
- Q-learning follows an exploration policy and learns $q_{\star}$.
- SARSA follows an exploration policy, π , and learns q_{π} .
- What if exploration policy is greedy?



Double Q-Learning

- Q-learning may suffer from maximization bias.
 - What is it?
- Double Q-learning **mitigates this bias** by learning two action-value functions:
$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha[R_{t+1} + \gamma Q_2(S_{t+1}, \arg \max_{a'} Q_1(S_{t+1}, a')) - Q_1(S_t, A_t)]$$
- Is this on- or off-policy?
- What does double Q-learning converge to?



Off-Policy SARSA

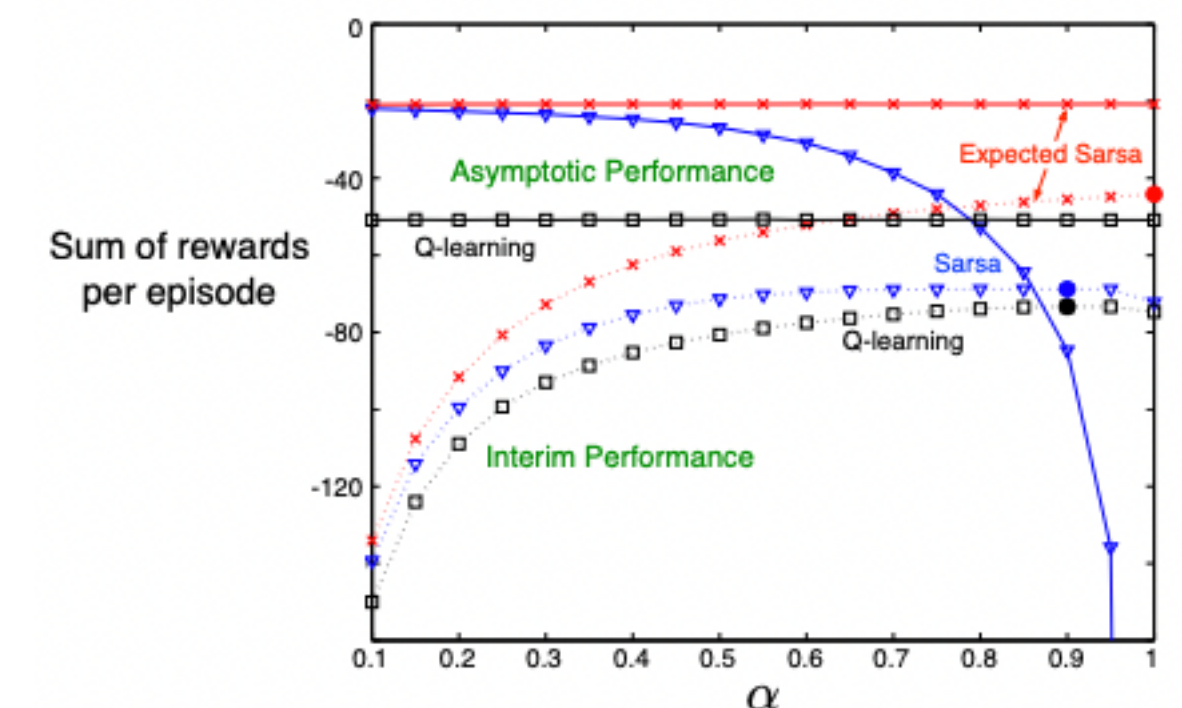
- Can SARSA learn off-policy?
- Yes, with importance sampling!
- $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \rho_t [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$
- Where $\rho_t := \frac{\pi(A_t | S_t)}{b(A_t | S_t)}$.
- Note that we only have a single factor in the importance weight.
 - What advantage would this have compared to off-policy Monte Carlo?
 - What is the off-policy variant of n-step returns?

Expected SARSA

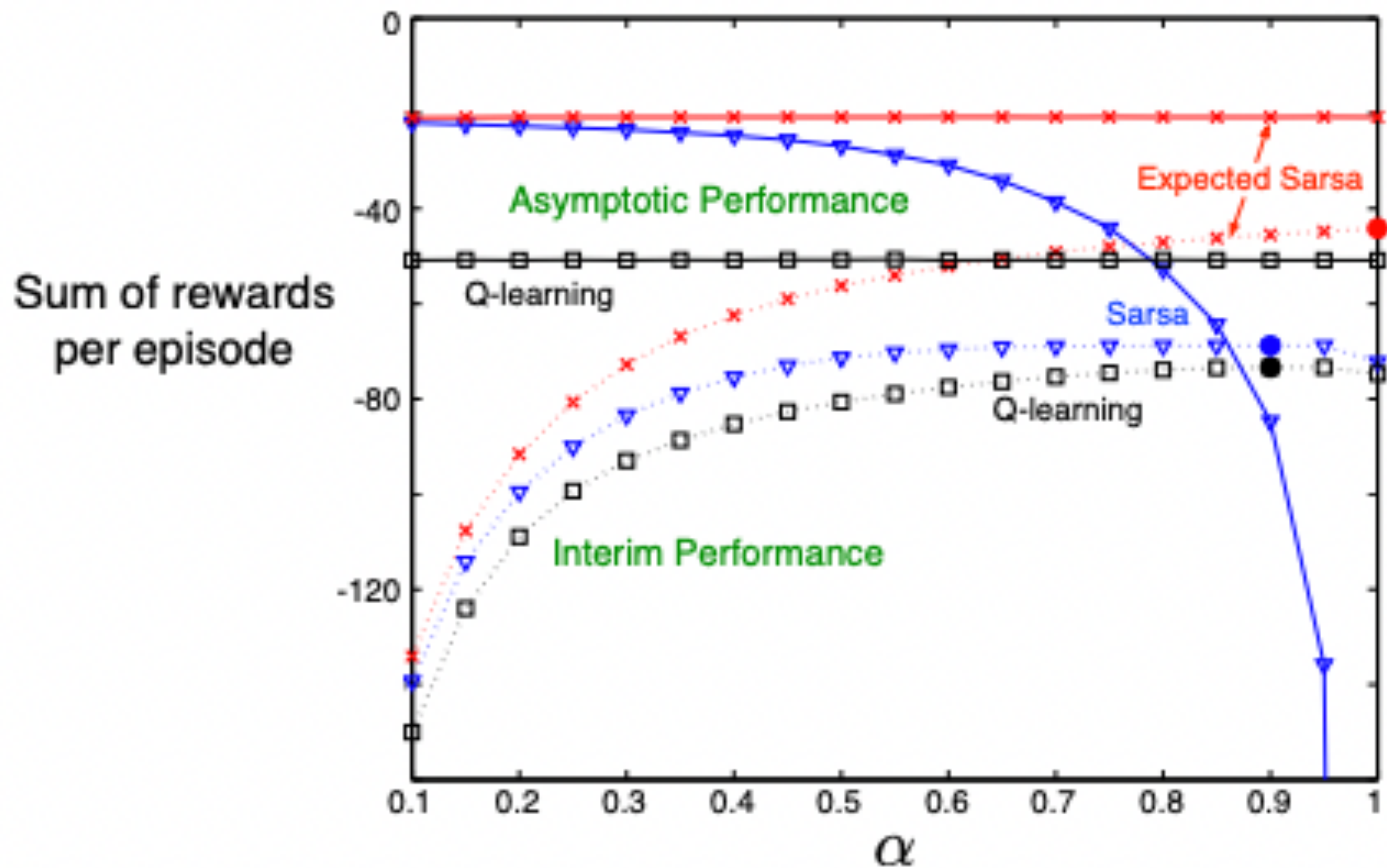
- SARSA samples the final action A' . How could this be harmful?
- We know π so we can compute the expected action-value exactly.

- $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \sum_{a'} \pi(a' | S_{t+1}) Q(S_{t+1}, a') - Q(S_t, A_t)]$$

- How is this update useful? What are its limitations?
- (+) Lower variance $\rightarrow$ more data efficient learning.
- (-) Computational cost for large or continuous action sets.



Expected SARSA



Discussion

At your table, identify two potential applications:

1. Identify an application where you expect Monte Carlo learning to be more effective.
2. Identify an application where you expect TD-learning to be more effective.

Compare and contrast Monte Carlo and TD-learning on each task and justify why you would prefer one approach over the other.

Summary

- Temporal Difference learning enables fully incremental, online, model-free learning.
- TD-learning often learns faster than Monte Carlo methods in MDPs but can combine the two approaches through n-step returns.
- TD-learning can be integrated into generalized policy iteration in several ways.
 - SARSA uses on-policy TD-learning.
 - Q-learning learns $q_{\star}$ while acting off-policy.
 - Expected SARSA generalizes Q-learning and usually improves upon SARSA.

Action Items

- Start on homework
- Start reading chapter 8 for next week.
- Be thinking about final project — proposal due October 1st.
 - The more concrete your proposal is, the better guidance you will receive!