Advanced Topics in Reinforcement Learning

Lecture 8: On-Policy Temporal Difference Learning

Josiah Hanna
University of Wisconsin — Madison

Announcements

- Homework released. Due: October 21 at 9:30AM (minute class starts)
- Read chapter 7 for next week.
- Project proposals due: Thursday!
    - 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.

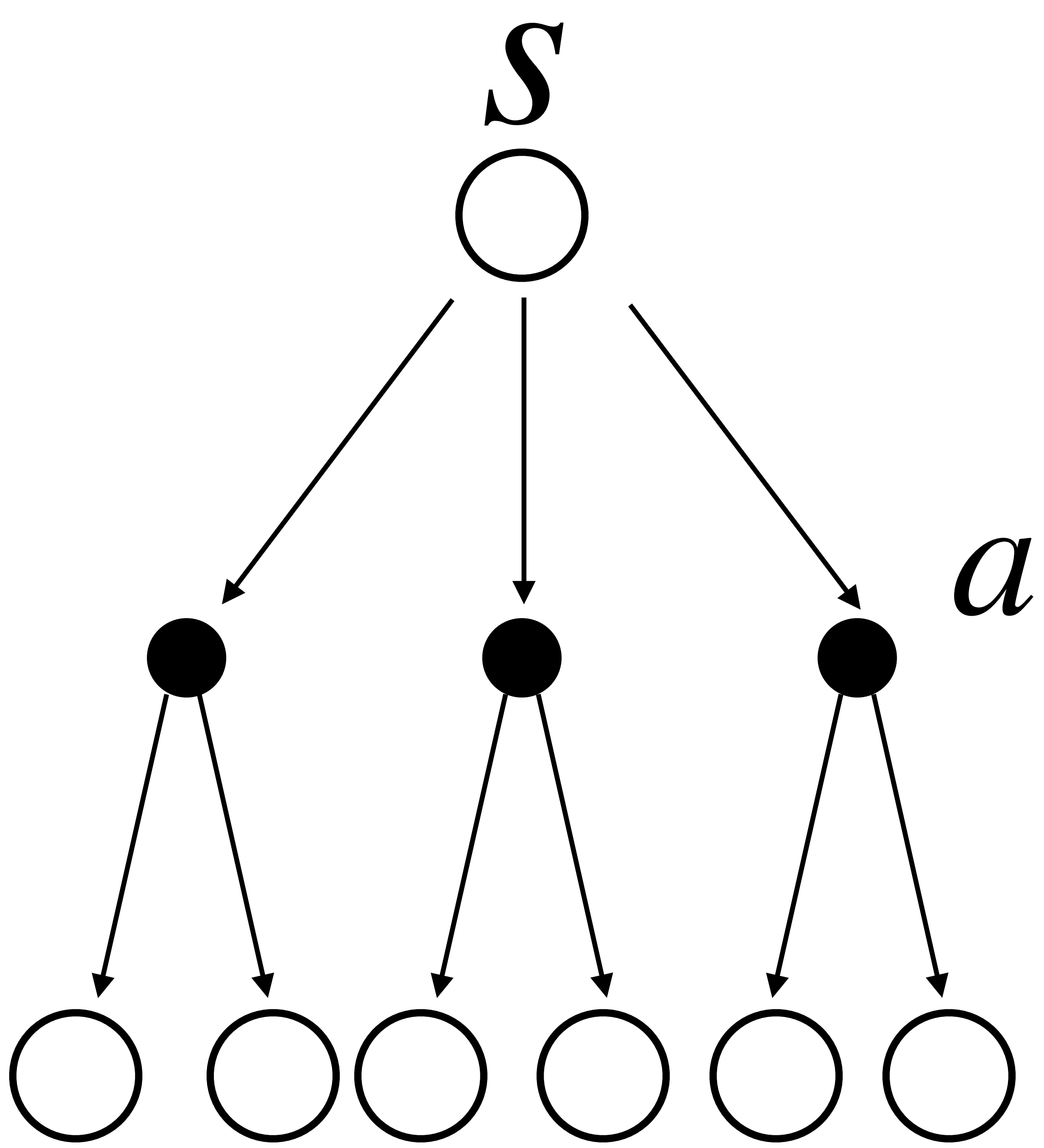
$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

TD-error

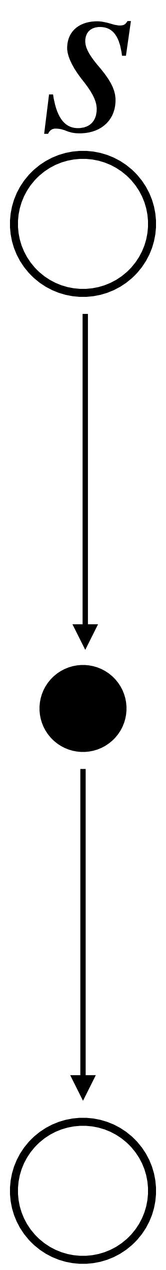
- Monte Carlo update: $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$
- TD update: $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$
- Compare to dynamic programming:

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

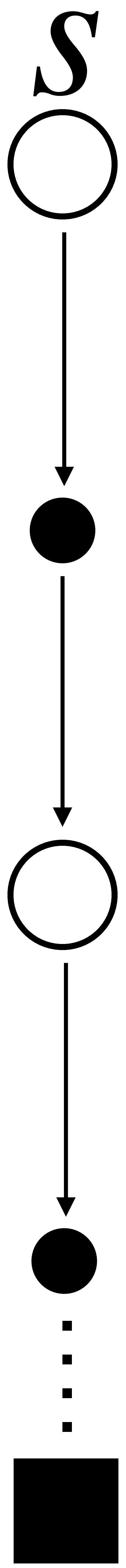
Back-up Diagrams



Dynamic Programming



TD(0)



Monte Carlo

Driving Home Example

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

State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
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

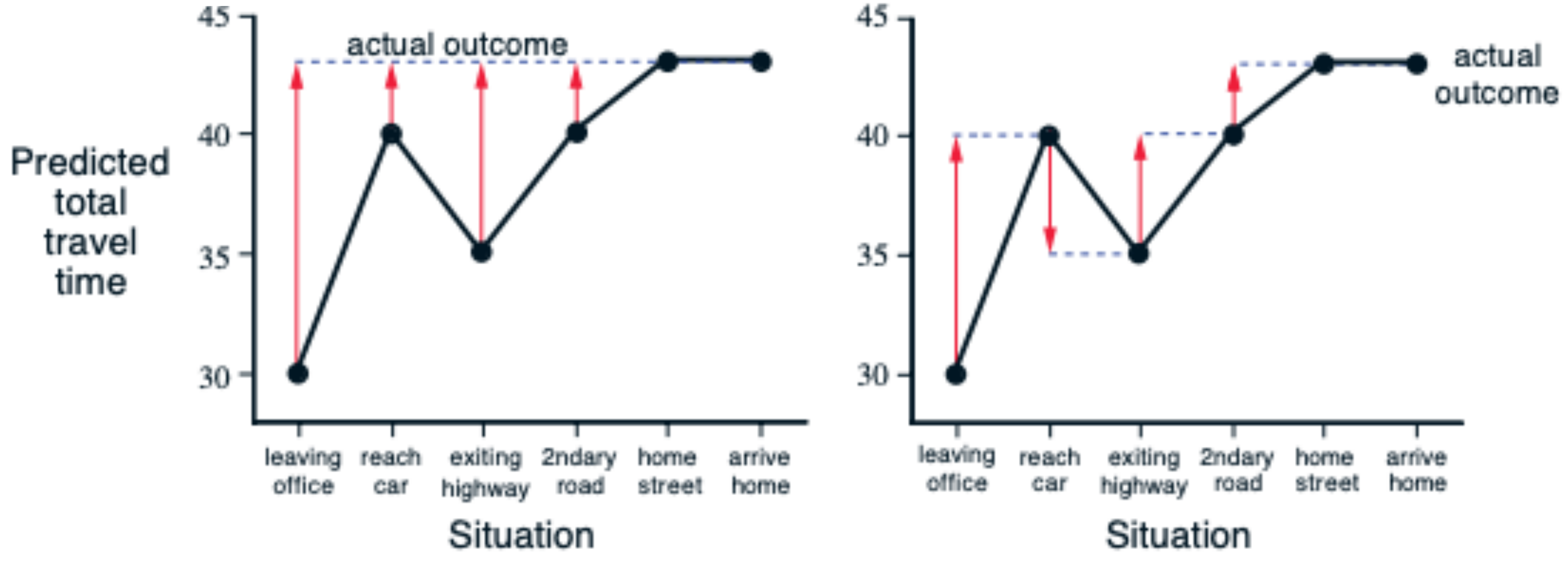


Figure 6.1: Changes recommended in the driving home example by Monte Carlo methods (left) and TD methods (right).

Update Frequency

Monte Carlo
Update

$$\ldots, S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}, A_{t+2}, R_{t+3}, S_{t+3}$$

TD
Update

TD
Update

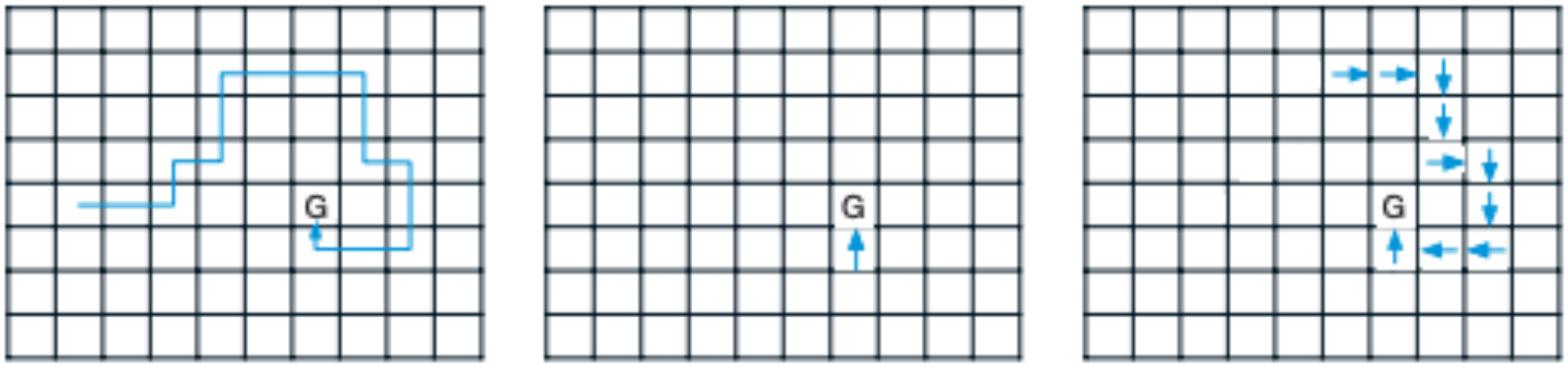
TD
Update

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 $Y_t := R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^n V(S_{t+n})$
- TD(0) is $n = 1$ and Monte Carlo is $n = \infty$; TD(λ) blends between extremes.



Agent's Path **TD(0)** **10-step TD**

Figure 7.4

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.

Convergence

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

- Where do these methods fall on the bias-variance trade-off?

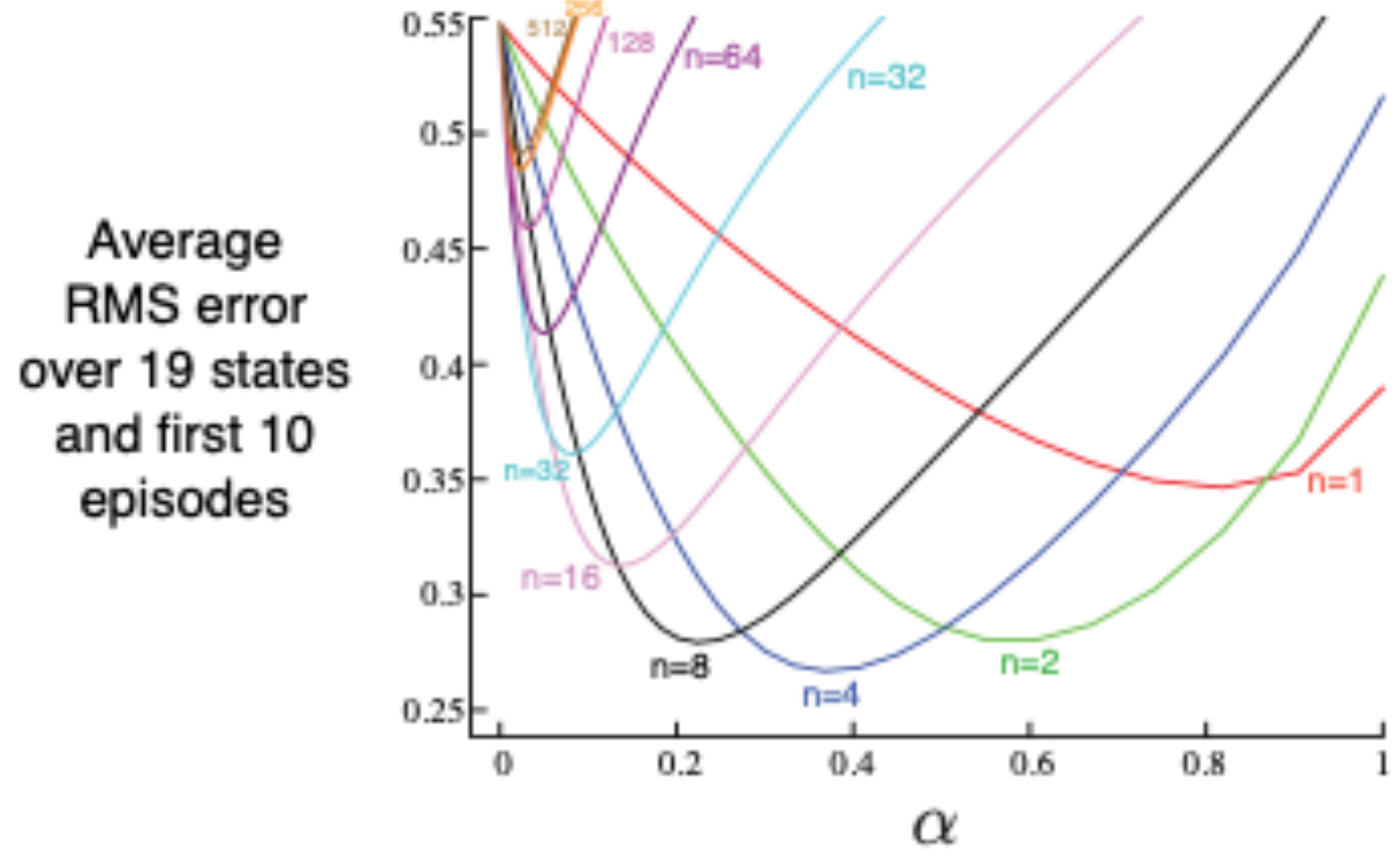
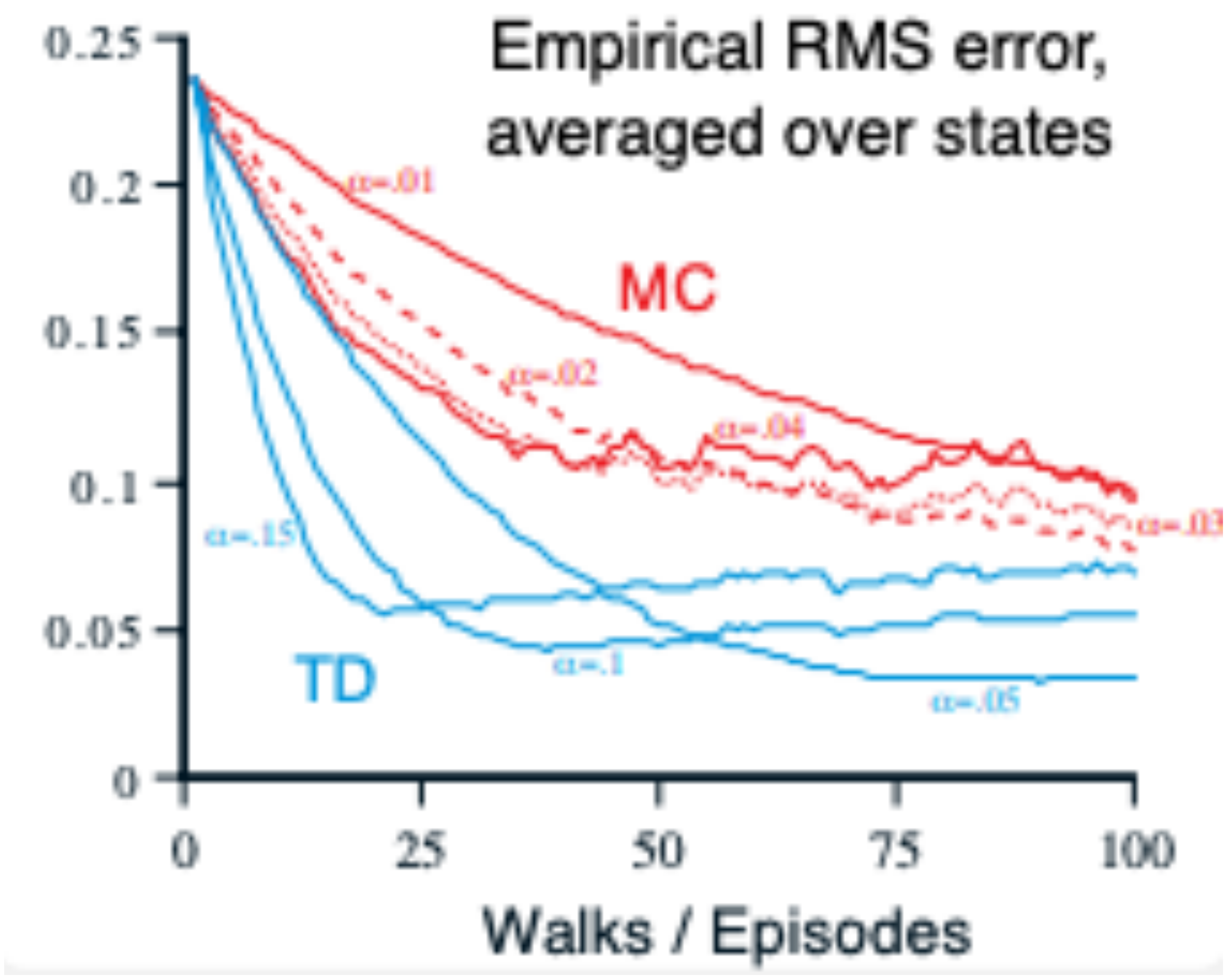
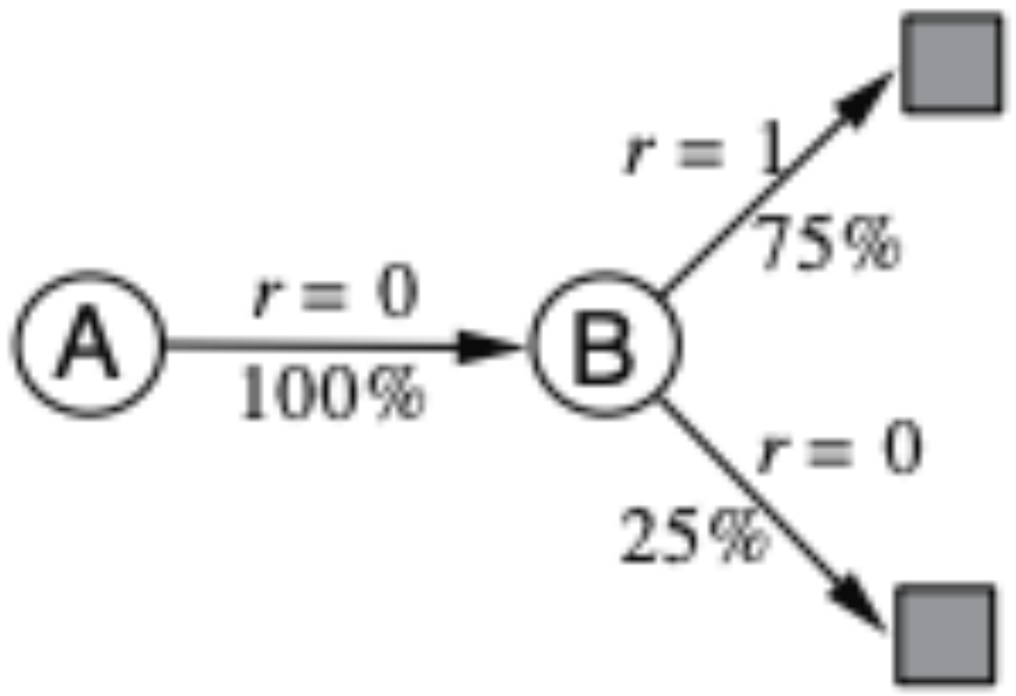


Figure 7.2: Performance of n-step TD methods as a function of α, for various values of n, on a 19-state random walk task (Example 7.1). ■



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

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

This is dynamic programming with estimated transitions!

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

Aaron’s Presentation

Slides

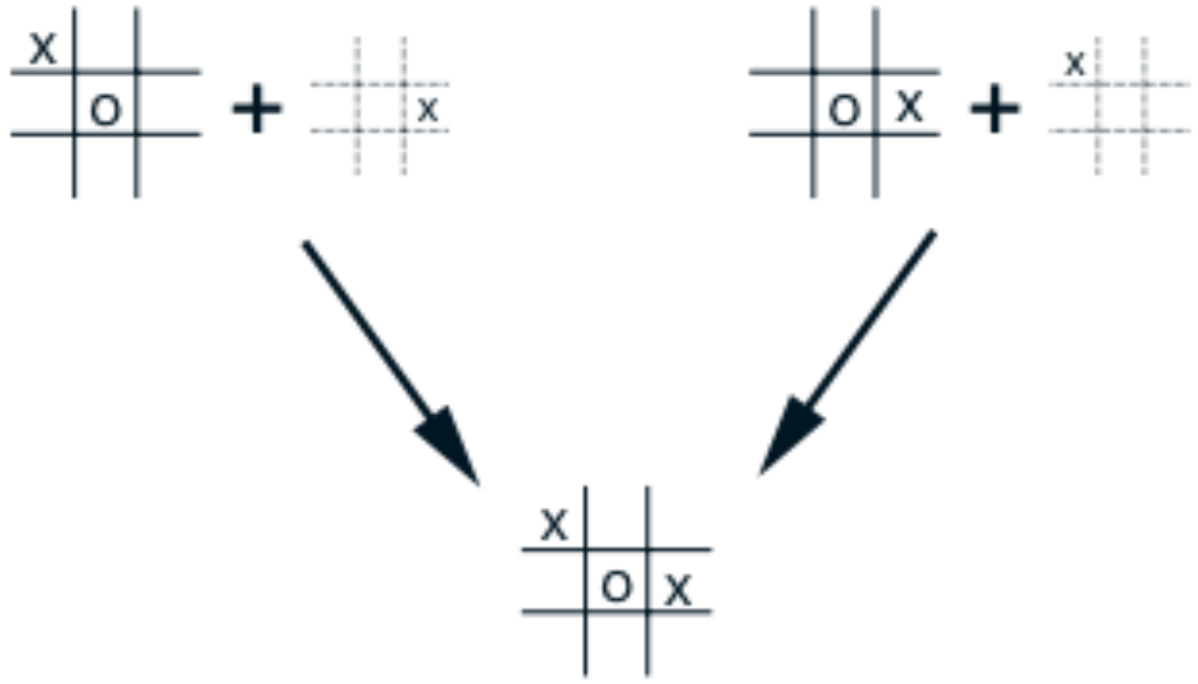
SARSA

- Same generalized policy iteration scheme from past two weeks.
  - Evaluate π_k.
  - Make π_{k+1} greedy with respect to π_k.
- 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 action-values and ϵ-greedy exploration converge to?

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

After-States

- In RL, the environment is usually a blackbox.
- But sometimes we have intermediate state changes that are available immediately after an action is taken.
- Such knowledge can be built into RL algorithms to help generalize learning.



Summary

- Temporal Difference learning enables online learning without a model of the environment.
- TD-learning often learns faster than Monte Carlo methods in MDPs but can combine the two approaches through n-step returns.
- SARSA uses TD-learning of action-values for policy evaluation in policy iteration — enables incremental, model-free policy improvement.

Action Items

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