Advanced Topics in Reinforcement Learning

Lecture 9: 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 8 for next week.
- Project proposals due tonight!
  - Next up: literature review (due October 30)

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.

Today

- Finishing Prediction
  - Convergence of TD / Monte Carlo.
  - TD(λ)
- On-Policy SARSA for control.
- Q-learning for control.
- Off-Policy SARSA.

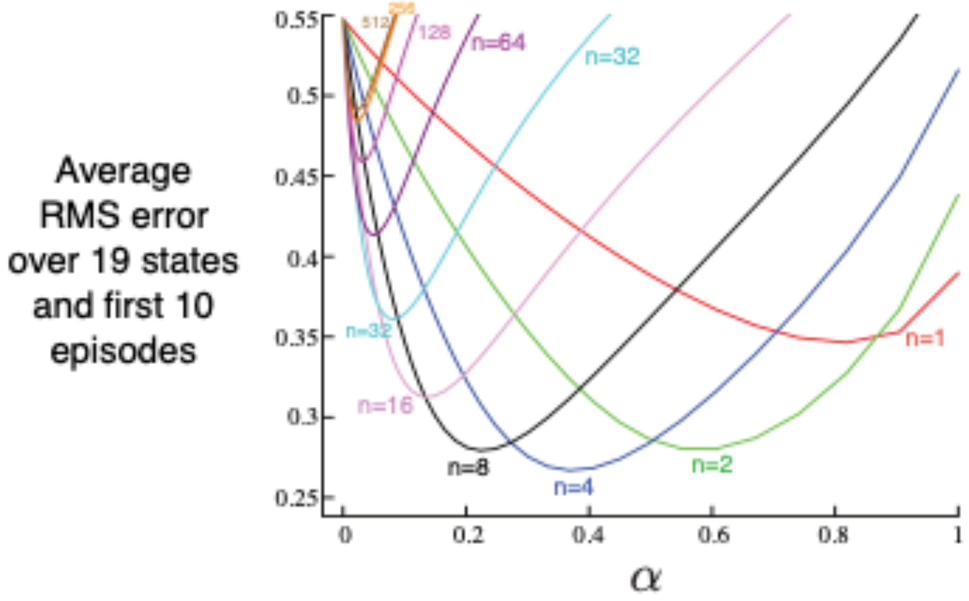
Review

- Temporal difference (TD) learning learns from experience and bootstraps
  - Allows immediate learning without a model of the environment.
- In a batch setting, TD-learning converges to the certainty-equivalence estimate.
  - Highlights the connection between TD-learning and dynamic programming.
- TD-learning and Monte Carlo methods sit at either end of a spectrum of n-step return methods.

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?



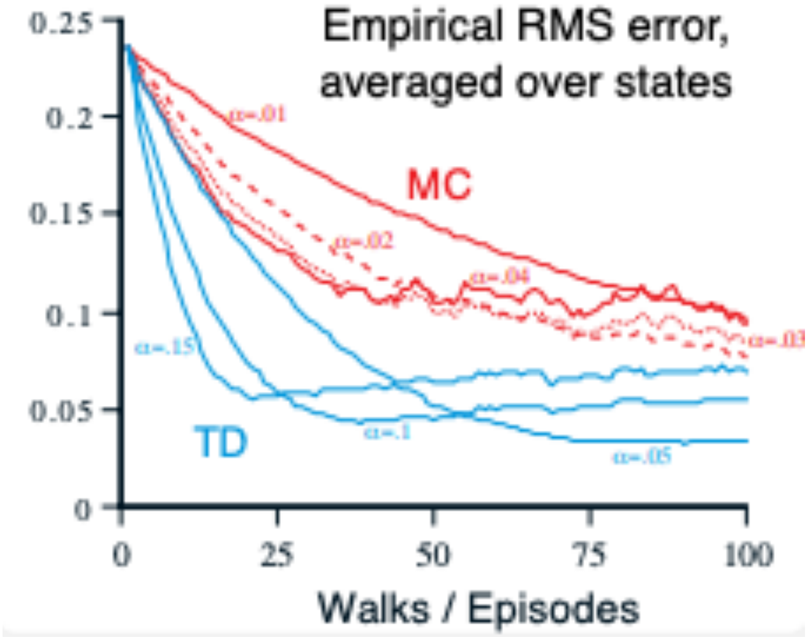


Figure 7.2: Performance of n-step TD methods as a function of α, for various values of n, on a 19-state random walk task (Example 7.1). ■

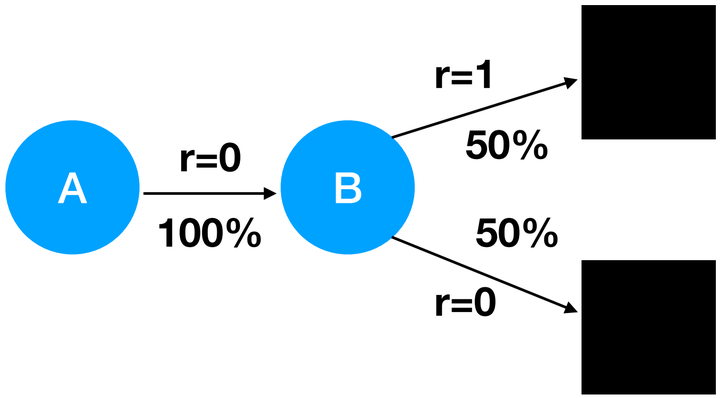
Max’s Presentation

Expected Eligibility traces
Hado van Hasselt et al. AAAI 2021.

Slides

Certainty Equivalence Updating

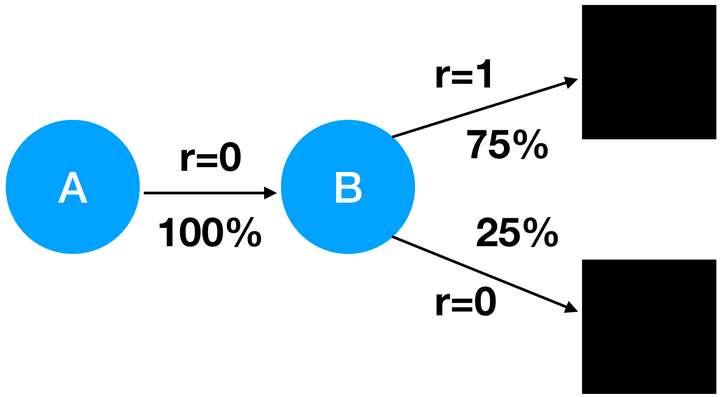
- Certainty Equivalence Learning: use data to estimate Markov process and then compute value function for the estimated process.



True Markov Process

Data:

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0



Estimated Markov Process

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 TD-errors:
- $$v_{k+1}(s) \leftarrow v_k(s) + \alpha \sum_{s',r} \#(s, r, s')[r + \gamma v_k(s') - v_k(s)]$$
- $$= v_k(s) + \alpha' \sum_{s',r} \frac{\#(s, r, s')}{\#(s)}[r + \gamma v_k(s') - v_k(s)]$$
- $$= (1 - \alpha')v_k(s) + \alpha' \sum_{s',r} \hat{p}(s', r \mid s)[r + \gamma v_k(s')]$$

Number of times we observed s, r, s'

Estimate of p

This is like dynamic programming with estimated transitions!

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

CE for n-step returns?

$$\sum_{r_{1:n},s_n} \frac{\#(s, r_{1:n}, s_n)}{\#(s)}[r_n^\gamma + \gamma^n v_k(s_n)] = \sum_{r_{1:n},s_n} \frac{\#(s, r_{1:n}, s_n)}{\#(s)} r_n^\gamma + \sum_{r_{1:n},s_n} \frac{\#(s, r_{1:n}, s_n)}{\#(s)} \gamma^n v_k(s_n)$$

$$= \sum_{r_{1:n}} \frac{\#(s, r_{1:n})}{\#(s)} r_n^\gamma + \sum_{s_n} \frac{\#(s, s_n)}{\#(s)} \gamma^n v_k(s_n)$$

$$= \hat{r}_n^\gamma(s) + \gamma^n \sum_{s_n} \hat{p}(s_n \mid s) v_k(s_n)$$

What is this if $n = \infty$? What if $n = 1$?

$$r_n^\gamma := r_1 + \gamma r_2 + \ldots + \gamma^{n-1} r_n$$

S A R S A

- Same generalized policy iteration scheme from past two weeks.
  - Evaluate π_k.
  - Make π_{k+1} greedy with respect to q_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 update on- or off-policy?
- What does generalized policy iteration with TD action-values and ϵ-greedy exploration converge to?

Q-Learning

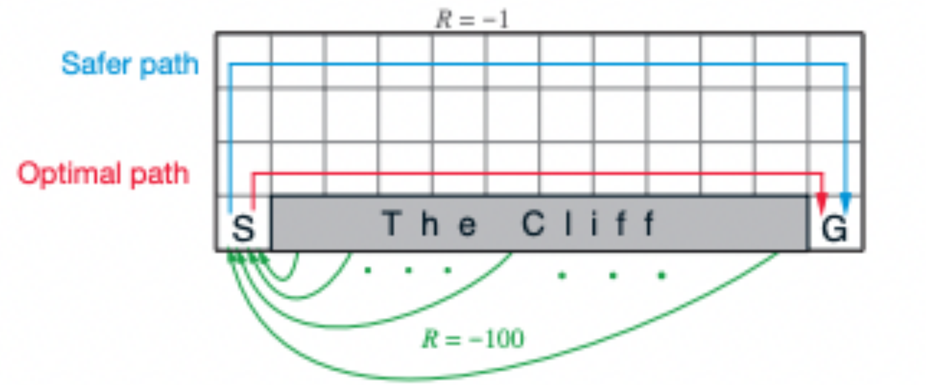
- SARSA is approximating policy iteration. What about value iteration?
- Q-learning update:

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

- 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?

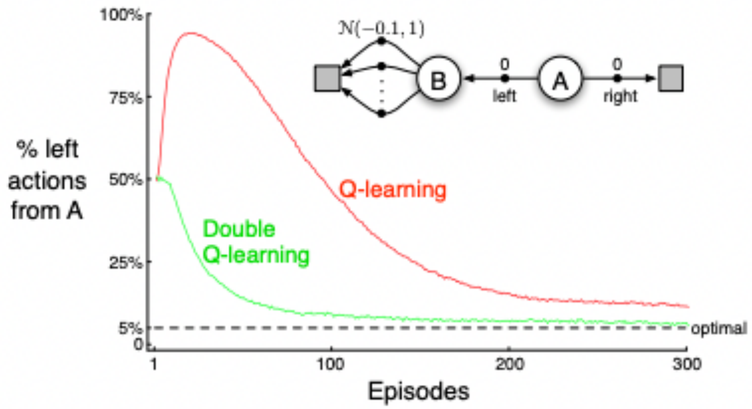


Double Q-Learning

- 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?

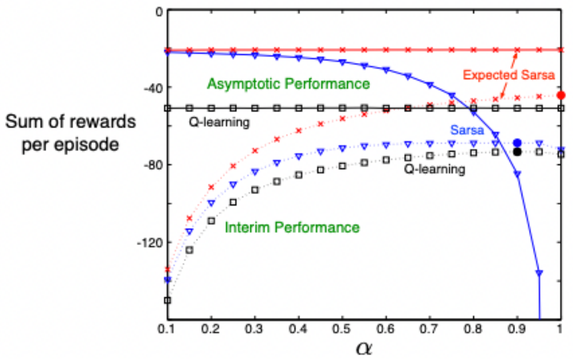


Off-Policy SARSA

- 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 $\rho_t := \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)}$.
- 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?

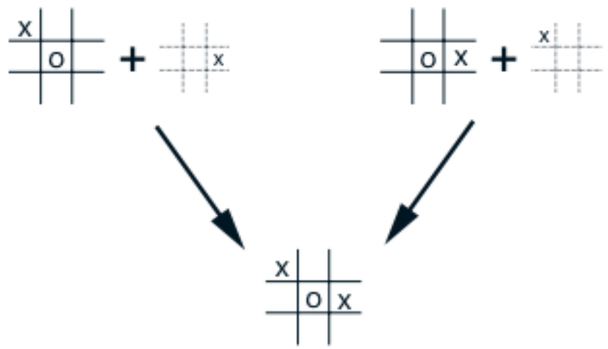
Expected SARSA

- SARSA samples the final acton A'. How could this be harmful?
  - We know π so we can compute the expected action-value exactly.
- $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \sum_{a'} \pi(a' \mid S_{t+1}) Q(S_{t+1}, a') - Q(S_t, A_t)]$$
- How is this update useful? What are its limitations?
  - (+) Lower variance —> more data efficient learning.
  - (-) Computational cost for large or continuous action sets.



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

- 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.
- These methods enable fully incremental, online, model-free learning.

Action Items

- Project proposal due midnight tonight.
- Read Chapter 8.