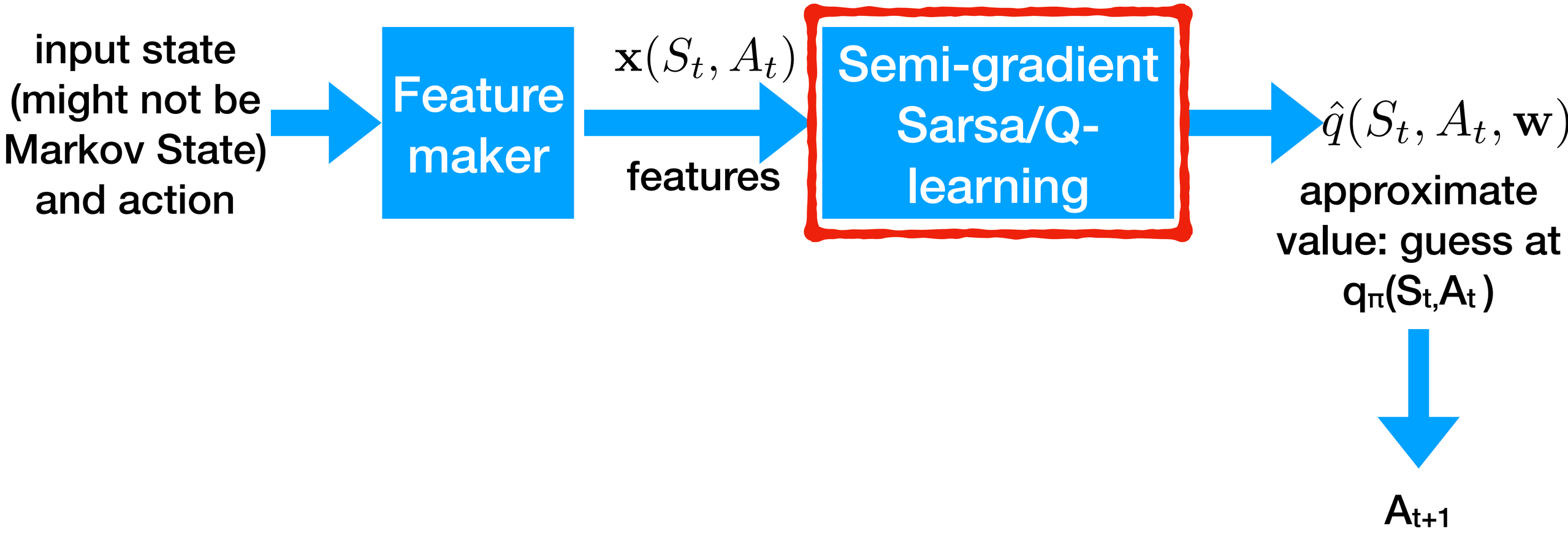
Course 3, Module 3
Control with Approximation

CMPUT 397
Fall 2020

Announcements

- Capstone project is due Monday at Midnight (deadline on last day of classes)
- No class on Monday! Instead, we will review the Practice Final on Wednesday or Thursday
  - I'll release a practice final this week
- Some course review this Friday, please come prepared to ask questions

Review of Course 3, Module 3
How to do control (learn a good policy) with function approximation


input state (might not be Markov State) and action
Feature maker
$\mathbf{x}(S_t, A_t)$
features
Semi-gradient Sarsa/Q-learning
$\hat{q}(S_t, A_t, \mathbf{w})$
approximate value: guess at $q_\pi(S_t, A_t)$
A_{t+1}

Video 1: Episodic Sarsa with Function Approximation

- We know how to do function approximation with TD; how about using that to learn action-values and a policy. **On-policy TD control** with approximation

- Goals:

  - Understand how to construct **action-dependent features** for approximate action-values >> stacking

  - and explain how to use Sarsa in episodic tasks with function approximation

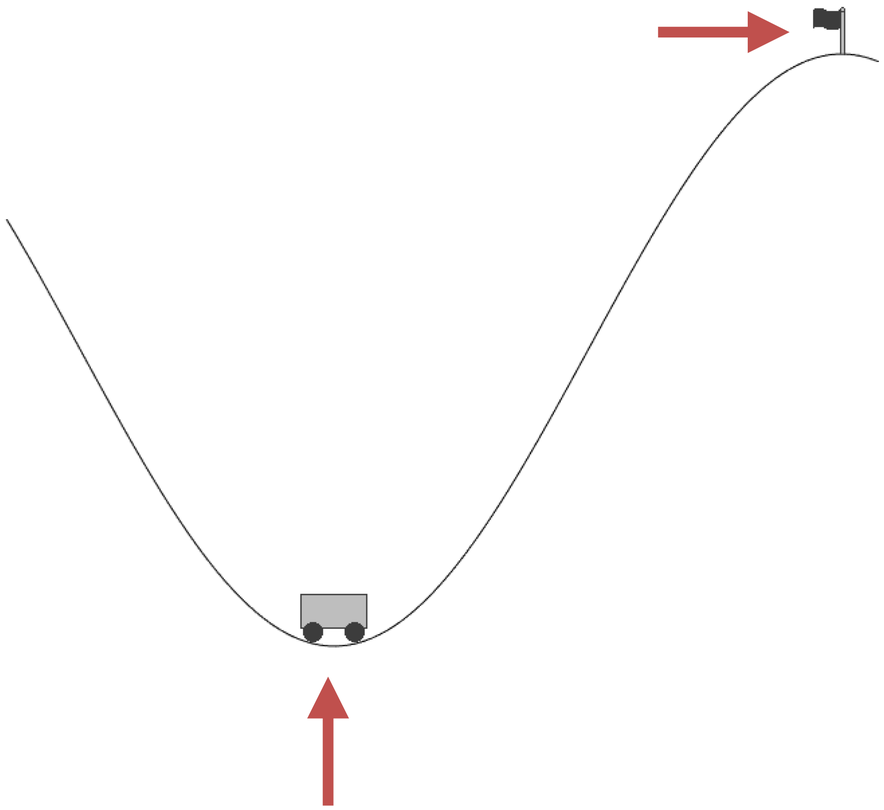
Video 2: Episodic Sarsa in Mountain Car

- **Can we do a large number of states with Semi-gradient Sarsa?** How about an infinite number of state? Yep. We do a classic control task: **Mountain Car**

- Goals:

  - gain experience analyzing the performance of an approximate TD control method

The Mountain Car environment



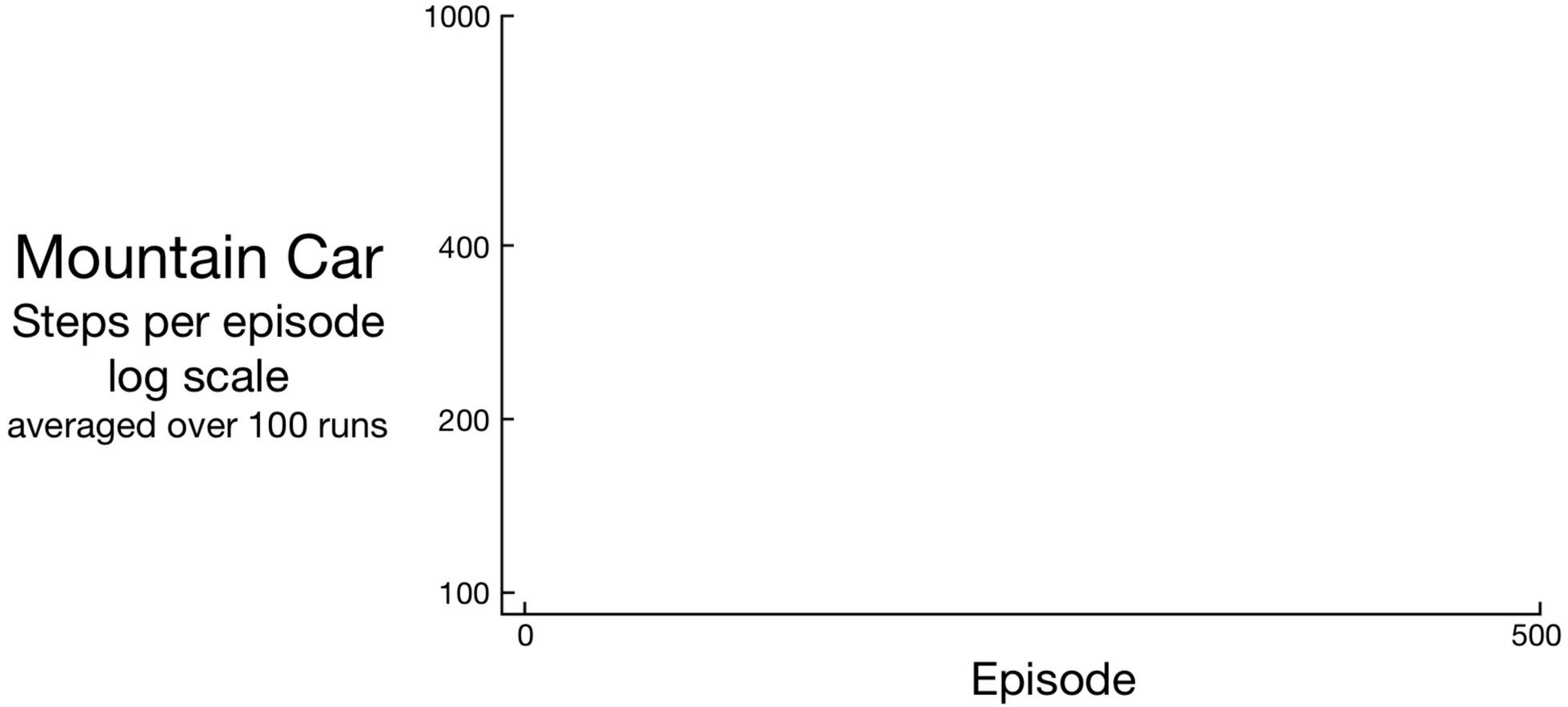
$$R_{step} = -1$$

$$\gamma = 1.0$$

State: Car position
Car velocity

Actions: Accelerate right
Accelerate left
Coast (no acceleration)

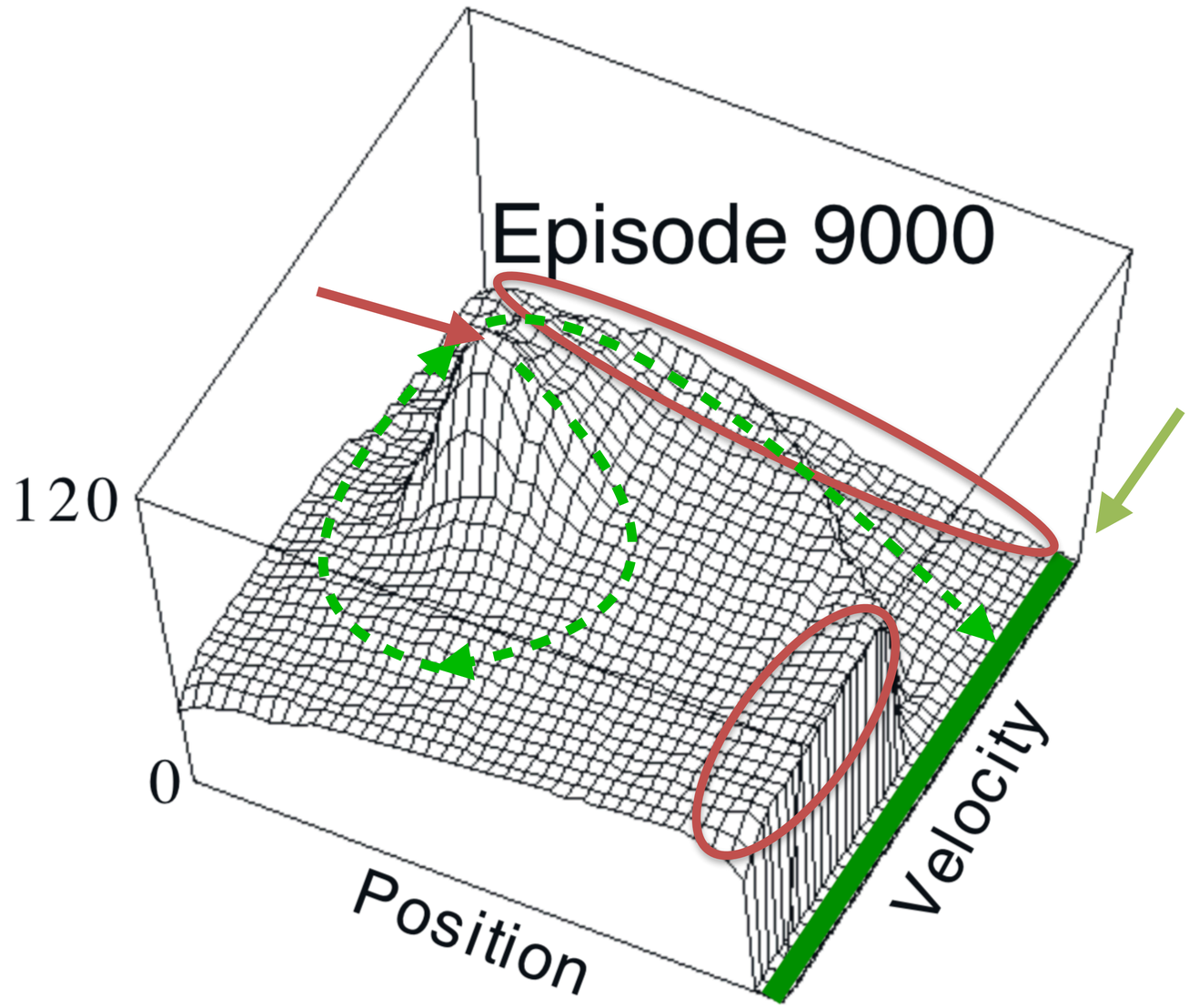
Learning curves



From Slido: “In Episodic Sarsa in the Mountain car environment, is step size with 0.5/8 the best step size?”

Learned values

– $\max_a Q(s, a, \mathbf{w})$



Be a good RL Scientist!

- Notice, even for this tiny problem we tried different alpha. We did **many runs**. We studied learning **speed**; **final performance**; even the value function
    - we have a good idea of how Sarsa works on this problem. It's robust and stable and pretty easy to tune it's parameters
- We want to do such careful analysis every time! Especially when comparing algorithms!
- ML and AI are growing! Lots of people want jobs
- One way to stand out, is to become a really careful empiricist! A master of **good experiments.** It's a rare skill

Video 3: Expected Sarsa with Function Approximation

- If we can do Semi-gradient Sarsa, then its just **small changes** to make Semi-gradient **Expected Sarsa** and Semi-gradient **Q-learning**!
- Goals:
  - Explain the update for Expected Sarsa with function approximation
  - And explain the update for Q-learning with function approximation

Self-test

- Slido: “How do you turn the expected sarsa algorithm to an update of Q-Learning?”

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
    $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
    Loop for each step of episode:
        Take action A, observe R, S'
        If S' is terminal:
            $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
            Go to next episode
        Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
        $S \leftarrow S'$
        $A \leftarrow A'$

Video 4: Exploration under Function Approximation

- Balancing exploration and exploitation in RL is **hard, even in the tabular case**. Recall that some of the ideas from the Bandit problem could not be easily translated into the tabular RL problem. It is even harder in function approximation. Counting state visits? How do we do optimistic initial values with a tile coder or a NN?

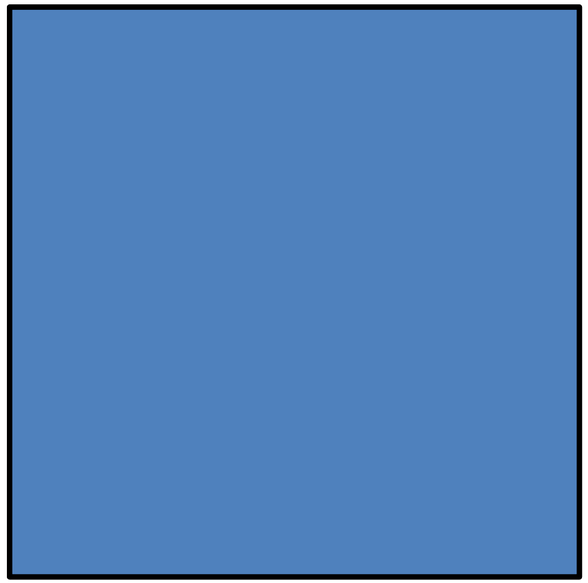
- Goals:

  - Describe how **optimistic initial values** and **epsilon-greedy** can be used with function approximation.

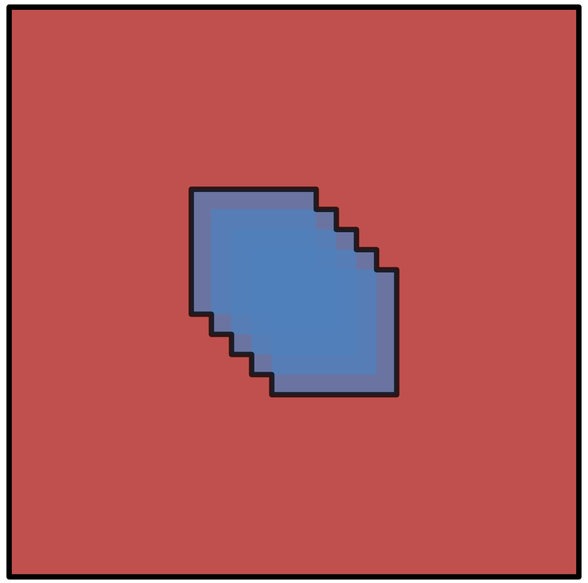
Self-test

- How do you use epsilon-greedy with semi-gradient Q-learning? How is the pseudocode different from tabular Q-learning?
- For tabular Q-learning, we said using epsilon-greedy ensures you visit every state. Is that true for semi-gradient Q-learning?
- For tabular Q-learning, do optimistic initial values ensure we visit every state at least once? How about for semi-gradient Q-learning?

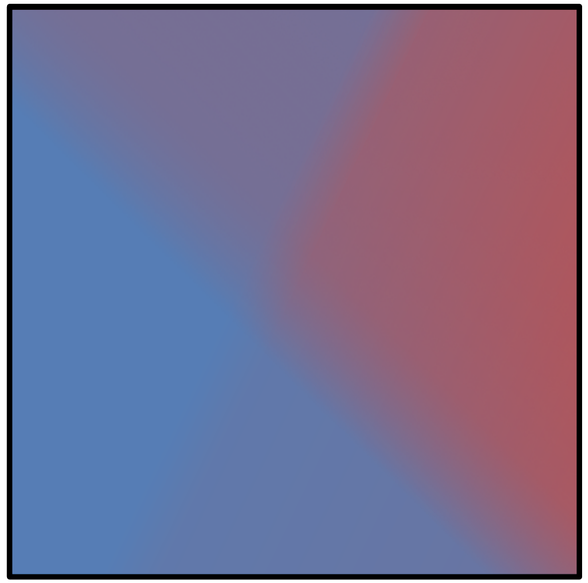
How Optimism Interacts with Generalization



Single feature



Tile coding



Neural network

Video 5: Average Reward: A New Way of Formulating Control Problems

- In some situations **discounting might not be the best choice**. For example, in **continuing tasks with function approximation**. Let's consider another way to formulate the RL task: average reward!

- Goals:

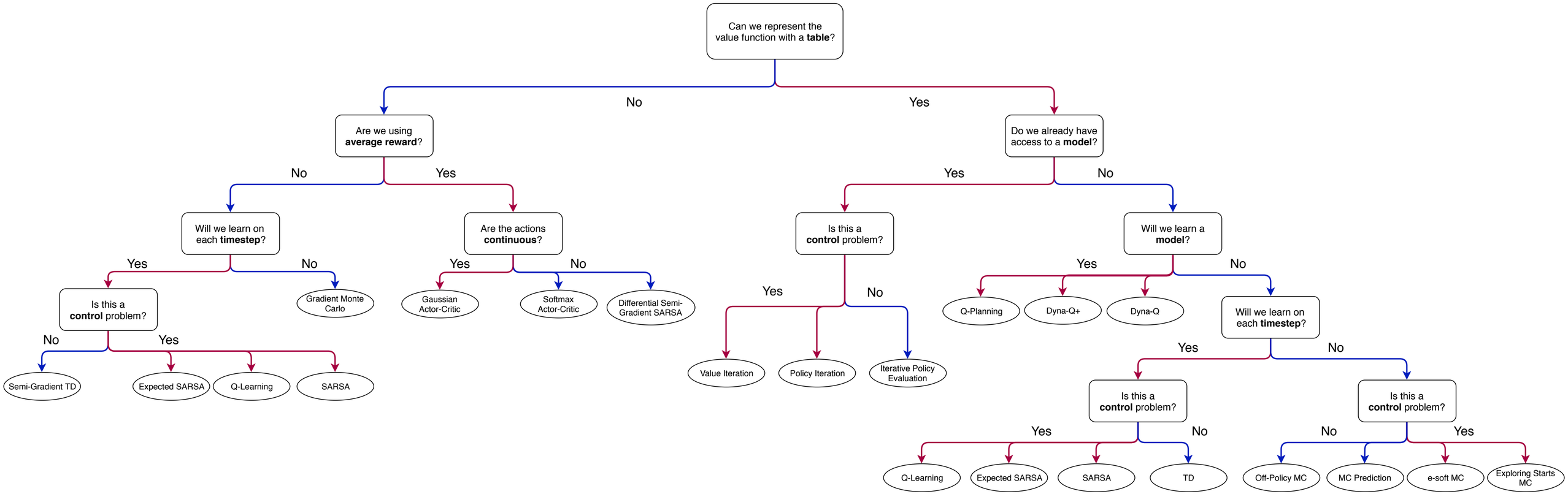
  - Describe the average reward setting

  - Explain when average reward optimal policies are different from policies obtained under discounting

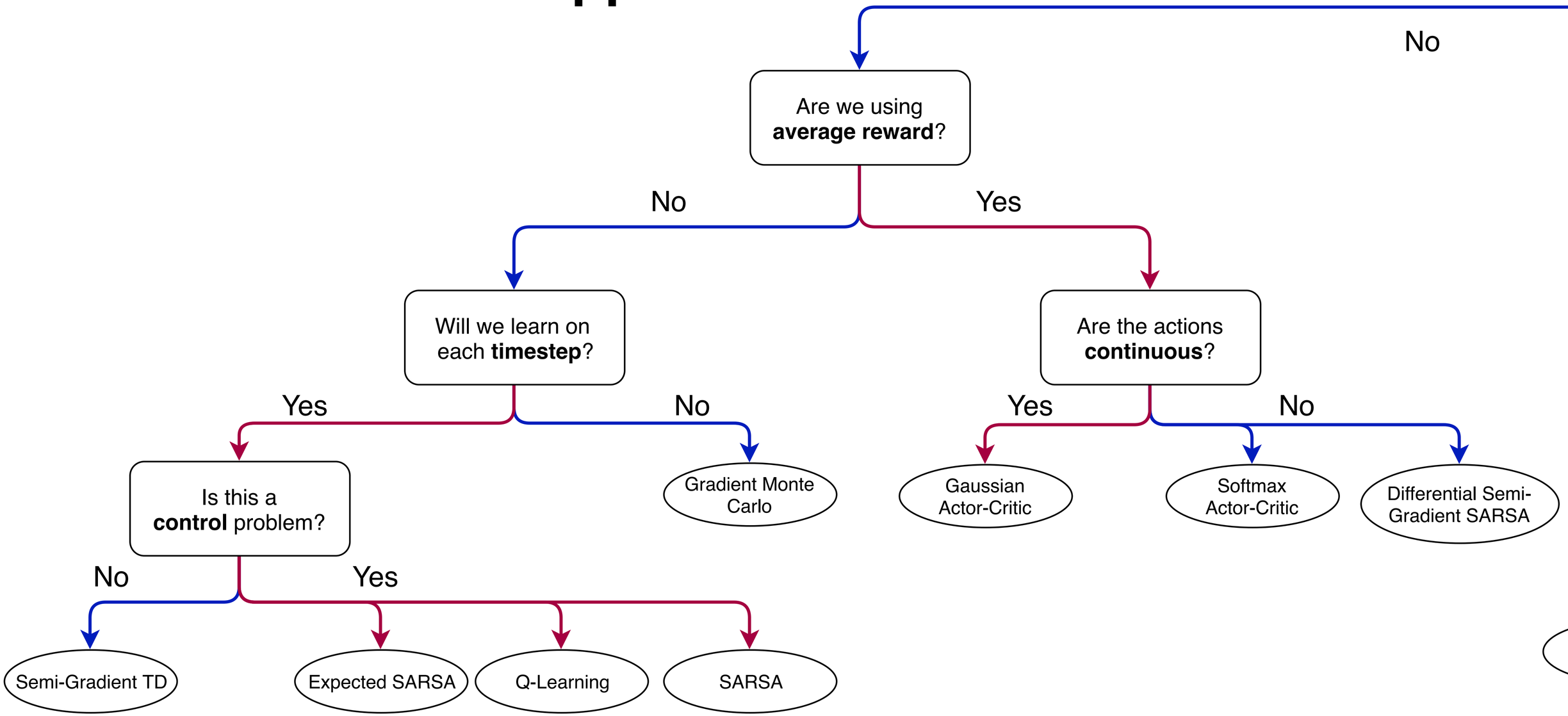
  - And understand differential value functions.

Where are we?

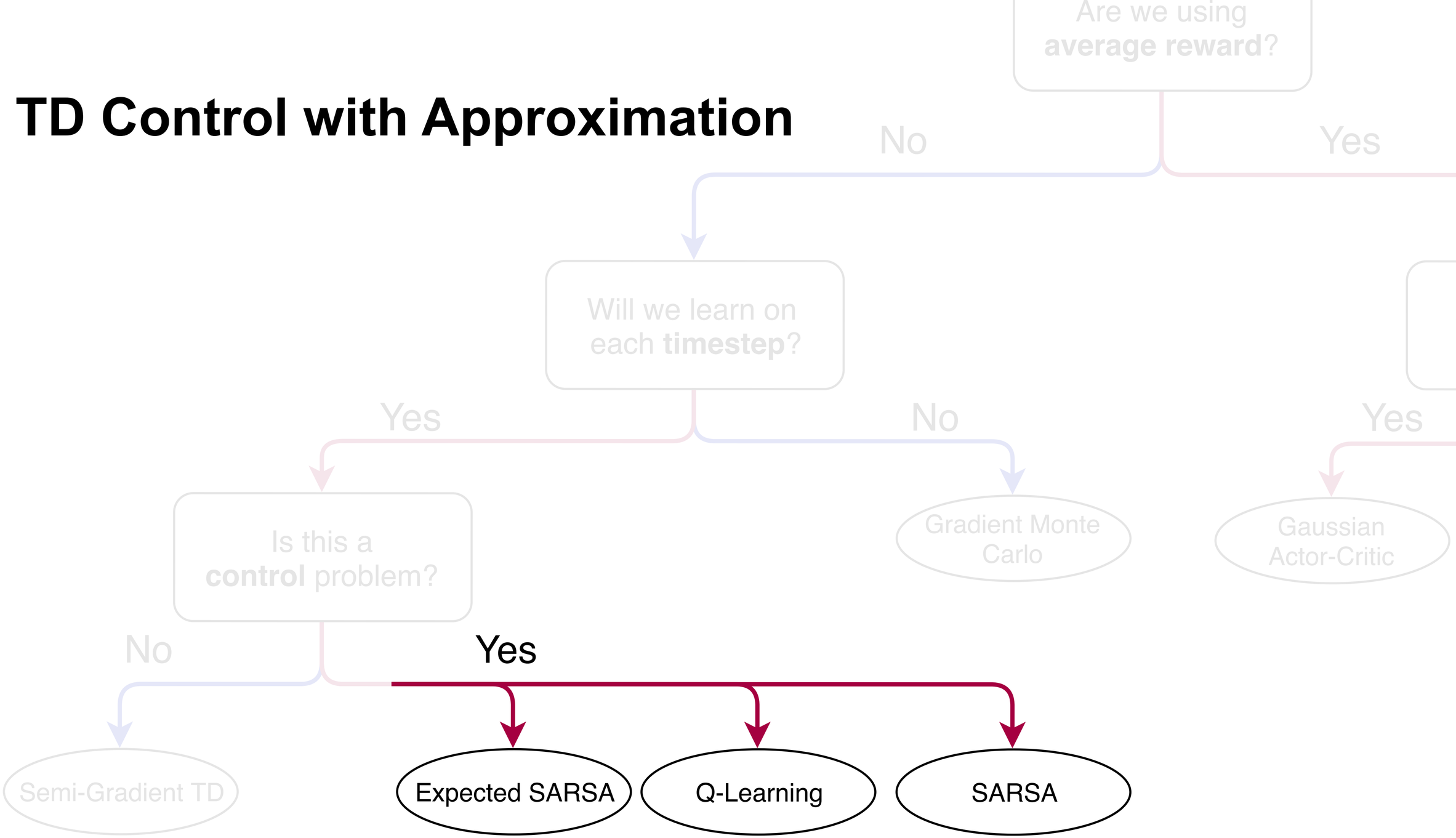
TD Control with Approximation



TD Control with Approximation



TD Control with Approximation



Slido Questions: FA vs tabular

- Why does updating the state of one not affect the state of others in the tabular case?
- For the tabular, since it updates each value function for each state, when over many steps and episodes, does it has a high variance due to accumulation.

Slido: Implementations

- When implementing function approximation for action values in reality, do we care about the space inefficiency due to stacking or is the performance gain minimal after fixing this?
- For a given time t, is the estimate of the average reward, R_t bar, always updated the same way (R_t+1 bar = R_t bar + beta*delta)? Or are there other methods of calculating R_t bar that converge to r(pi) differently? (worksheet question)
- do we always optimistically initialize values?
  - Or maybe this means: Should we always optimistically initialize values?

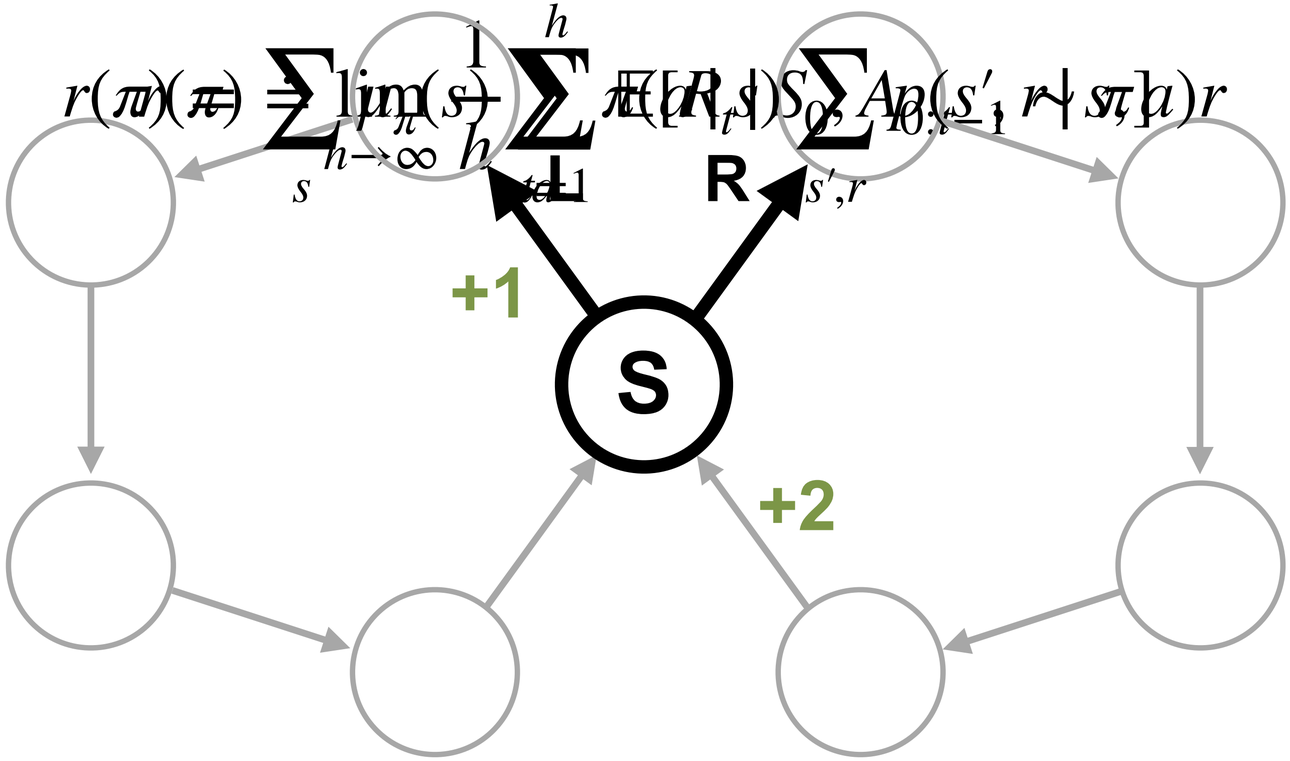
Slido: Differential Return

- How is differential return calculated? In the example, the average reward of going left is 0.2, then shouldn't the differential return be 0 when going left? Since applying average reward to all time steps equals the sum of reward in all time steps.

- When calculating returns for average reward, how does subtracting the average reward from every return not give us a return of 0? I am struggling to see how we get a different value when H goes to infinity.

The Average Reward objective

$$r(\pi) = \lim_{h \to \infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t \mid S_0, A_{0:t-1} \sim \pi] = \sum_s \mu_\pi(s) \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) r$$



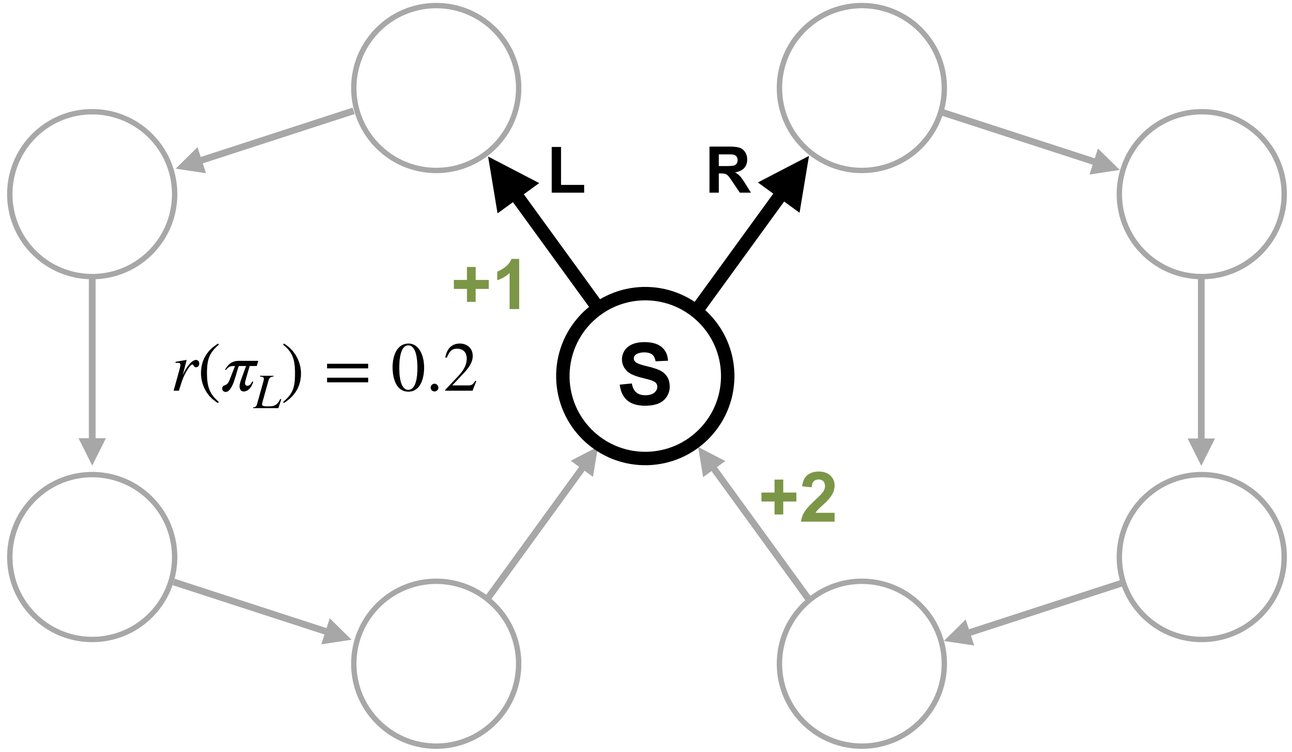
$r(\pi_L) = 1/5 = 0.2$

$r(\pi_R) = 2/5 = 0.4$

Returns for Average Reward

$$G_t = R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \dots$$

$$G_t = 1 - 0.2 + 0 - 0.2 + 0 - 0.2 + 0 - 0.2 + 0 - 0.2 + \vdots \; H \to \infty = 0.4$$

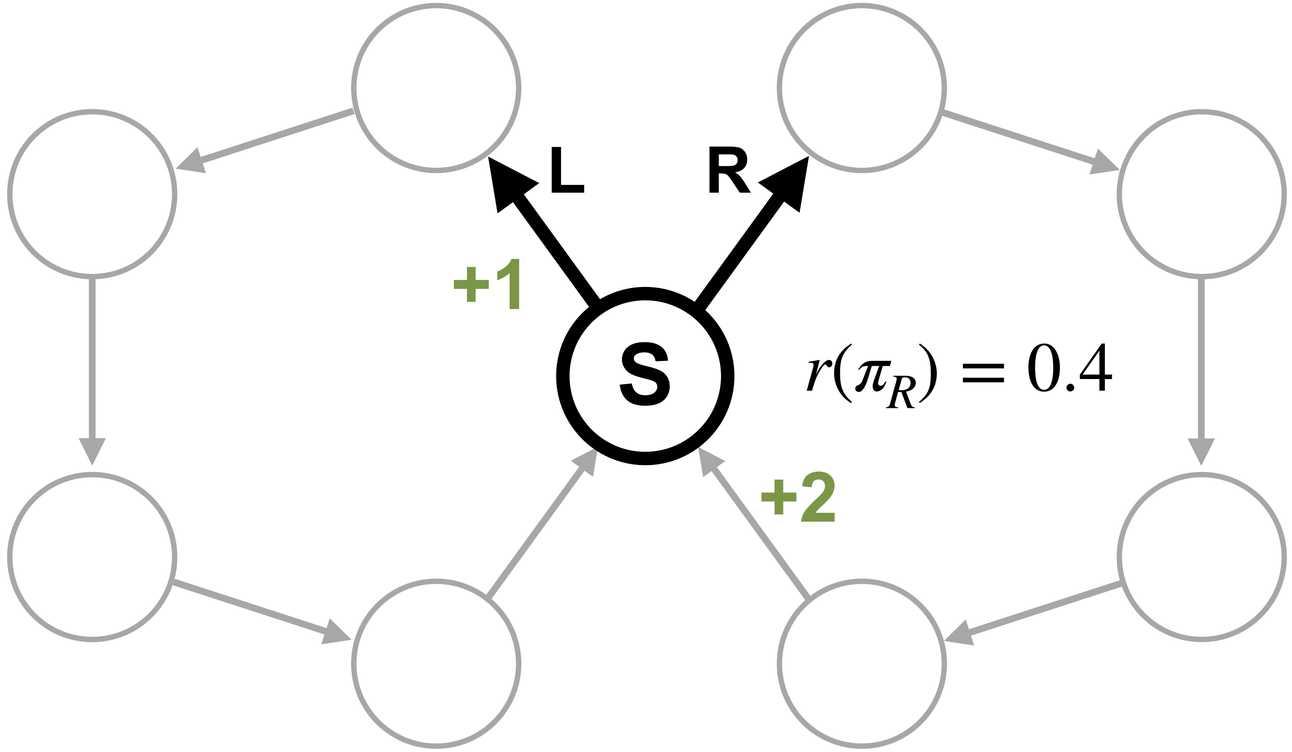


$$G_t = 0 - 0.2 + 0 - 0.2 + 0 - 0.2 + 0 - 0.2 + 2 - 0.2 + 0.4 = 1.4$$

Returns for Average Reward

$$G_t = \boxed{R_{t+1} - r(\pi)} + \boxed{R_{t+2} - r(\pi)} + \boxed{R_{t+3} - r(\pi)} + \dots$$

$$\begin{aligned} G_t = {} & 1 - 0.4 + \\ & 0 - 0.4 + \\ & 0 - 0.4 + \\ & 0 - 0.4 + \\ & 0 - 0.4 + \\ & (-0.8) \\ = {} & -1.8 \end{aligned}$$



$$\begin{aligned} G_t = {} & 0 - 0.4 + \\ & 0 - 0.4 + \\ & 0 - 0.4 + \\ & 0 - 0.4 + \\ & 2 - 0.4 + \\ & \vdots \quad H \to \infty \\ = {} & -0.8 \end{aligned}$$

Slido: Problem Specification

- Is there any reference about how to design a reward function in some virtue environment setting? For example, we know in Go the reward could be 1 for win, -1 for loss, 0 for tie. How about a universal or general environment?

- With the discounted reward, we can adjust the gamma value to determine how much we care about the future. Does it mean that with the implication of average reward, we will be losing this ability?

- Since the average reward and differential return method avoids one of the key issues with discounting, is there any reason that we would want to use discounted rewards over differential rewards?

- Can the average reward method be applied to episodic tasks?

A paragraph from the book

- “This example and the more general argument in the box show that if we optimized discounted value over the on-policy distribution, then the effect would be identical to optimizing undiscounted average reward; the actual value of would have no effect. This strongly suggests that discounting has no role to play in the definition of the control problem with function approximation. One can nevertheless go ahead and use discounting in solution methods. The discounting parameter changes from a problem parameter to a solution method parameter! Unfortunately, discounting algorithms with function approximation do not optimize discounted value over the on-policy distribution, and thus are not guaranteed to optimize average reward. ”

Slido: Issues Estimating Rbar

- Differential semi-gradient Sarsa for estimating $\hat{q} \approx q_*$ estimates the average reward. Since the estimate is not necessarily equal to the true average reward, can't the differential return diverge to positive or negative infinity? If so, how do we avoid this?

- Related: What will happened if average reward term significantly larger or smaller than the True value?

Slido: Advanced/Misc

- How is discounted return suffer from exponentially large variance when using a large discount factor?
- Couldn't we do some "dummy" training to initialize an NN with optimistic values? For instance, run through a sufficient number of episodes with high step sizes and an artificially high reward?
- Average reward method comes from the idea of using Cesàro sum to calculate divergent series. Can we develop other methods from different summations of divergent series?