CS292F StatRL Lecture 4 Finite-Horizon MDPs / Temporal Difference Learning

Instructor: Yu-Xiang Wang

Spring 2021

UC Santa Barbara

Homework 1 released; project ideas shared

- You will learn the various elements of MDPs by solving problems. Also you will practice using Hoeffding's inequality and Bernstein's inequality.

- Mostly similar to what I covered in the lectures / sometimes the solutions are readily available by reading the AJKS book.

- I shared a document with recent RL theory papers by categories.
  - You do NOT have to pick one from there
  - Application projects are just as welcome --- e.g., applying RL to your problem / formulate your problem as an MDP.
  - I am happy to discuss with you if you have some fresh ideas.

Recap: MDP planning with access to generative models

- Motivation:
  1. Solving MDP faster / approximately with randomized algs that sample
  2. Study sample complexity of RL with unknown transitions (without worrying about exploration)

- Algorithm of interest: Model-based plug-in estimator.
  - Sample all state-action pairs uniformly. Estimate the transition kernel.
  - Do VI / PI on the approximate MDP.

Recap: on a brief digression, we learned concentration inequalities.

- Hoeffding's inequality

$$\left|\bar{X} - \mathbb{E}[\bar{X}]\right| \leq \sqrt{\frac{B^2}{2n}\log(2/\delta)}$$

- Bernstein inequality

$$\left|\bar{X} - \mathbb{E}[X_1]\right| \leq \sqrt{\frac{2\mathbf{Var}[X_1]}{n}\log(2/\delta)} + \frac{2M\log(2/\delta)}{3n}$$

- McDiarmid's inequality
  - Concentration of $f(X_1,\ldots,X_n)$ when f is stable / coordinate-wise Lipschitz.
  - Concentration is now enough, usually we need to also compute expectation.

- Union bound: merging failure probabilities.

Recap: Sample complexity bound Attempt 1

- Simulation Lemma

$$Q^\pi - \widehat{Q}^\pi \quad = \quad \gamma(I - \gamma \widehat{P}^\pi)^{-1}(P - \widehat{P})V^\pi$$

- Uniform convergence bound for all policies
  - By Holder's inequality, McDiarmid inequality.

- Sample complexity bound it suffices that we call this many times.

$$O\left(\frac{S^2 A + SA\log(2SA/\delta))}{(1-\gamma)^4 \epsilon^2}\right)$$

Recap: Sample complexity bound Attempt 2

- Show that the V* of the estimated MDP is close to the the V* function of the true MDP.

$$\|Q^\star - \widehat{Q}^\star\|_\infty \leq \frac{\gamma}{1-\gamma}\|(P - \widehat{P})V^\star\|_\infty$$

- Use Q-value amplification lemma:

$$V^{\pi_Q} \geq V^\star - \frac{2\|Q - Q^\star\|_\infty}{1-\gamma}\mathbb{1}.$$

- Overall sample complexity bound:

$$O\left(\frac{SA\log(2SA/\delta))}{(1-\gamma)^6\epsilon^2}\right)$$

Recap: optimal sample complexity

- Optimal sample complexity:

$$\Theta\left(\frac{SA\log(2SA/\delta))}{(1-\gamma)^3\epsilon^2}\right)$$

- Ideas to achieve it:
  - Bernstein inequality. (HW1)
  - Strong variance bound. (HW1)
  - Advanced Q-value error to policy value (not covered in the class)

This lecture

1. Wrap up MDPs
    - Performance difference lemma and advantage decomposition (Readings: AJKS Section 1.6)
    - Remarks about **finite horizon / episodic MDPs.** (Readings: AJKS Section 1.2)

2. RL algorithms
    - Model-based vs Model-free RL algorithms
    - Temporal difference learning. (Sutton and Barto Ch 5-6)
    - TD learning with linear function approximation.

Advantage function and Performance Difference Lemma

- Advantage function: $A^\pi(s,a) := Q^\pi(s,a) - V^\pi(s)\,.$
  - The advantage of taking given action over following the policy.
  - Simple fact: $A^*(s,a) := A^{\pi^*}(s,a) \leq 0$

- Performance Difference Lemma

Lemma 1.16. *(The performance difference lemma) For all policies π, π' and distributions μ over $\mathcal{S}$,*

$$V^\pi(\mu) - V^{\pi'}(\mu) = \frac{1}{1-\gamma}\mathbb{E}_{s'\sim d_\mu^\pi}\mathbb{E}_{a'\sim\pi(\cdot|s')}\left[A^{\pi'}(s',a')\right].$$

where $d_\mu^\pi(s) = (1-\gamma)\sum_{t=1}^{\infty}\gamma^{t-1}\mathbb{P}^\pi[S_t = s] = (1-\gamma)\nu_\mu^\pi(s)$

Proof of Performance Difference Lemma

$$
\begin{aligned}
V^\pi(s) - V^{\pi'}(s) &= \mathbb{E}_{\tau\sim\Pr^\pi(\tau|s_0=s)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] - V^{\pi'}(s) \\
&= \mathbb{E}_{\tau\sim\Pr^\pi(\tau|s_0=s)}\left[\sum_{t=0}^{\infty}\gamma^t \left(r(s_t,a_t) + V^{\pi'}(s_t) - V^{\pi'}(s_t)\right)\right] - V^{\pi'}(s) \\
&\overset{(a)}{=} \mathbb{E}_{\tau\sim\Pr^\pi(\tau|s_0=s)}\left[\sum_{t=0}^{\infty}\gamma^t \left(r(s_t,a_t) + \gamma V^{\pi'}(s_{t+1}) - V^{\pi'}(s_t)\right)\right] \\
&\overset{(b)}{=} \mathbb{E}_{\tau\sim\Pr^\pi(\tau|s_0=s)}\left[\sum_{t=0}^{\infty}\gamma^t \left(r(s_t,a_t) + \gamma \mathbb{E}[V^{\pi'}(s_{t+1})|s_t,a_t] - V^{\pi'}(s_t)\right)\right] \\
&\overset{(c)}{=} \mathbb{E}_{\tau\sim\Pr^\pi(\tau|s_0=s)}\left[\sum_{t=0}^{\infty}\gamma^t \left(Q^{\pi'}(s_t,a_t) - V^{\pi'}(s_t)\right)\right] \\
&= \mathbb{E}_{\tau\sim\Pr^\pi(\tau|s_0=s)}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi'}(s_t,a_t)\right] = \frac{1}{1-\gamma}\mathbb{E}_{s'\sim d_s^\pi}\,\mathbb{E}_{a\sim\pi(\cdot|s)}\gamma^t A^{\pi'}(s',a),
\end{aligned}
$$

Finite horizon MDPs

- Parameterization / Setup

$$M = (\mathcal{S}, \mathcal{A}, \{P\}_h, \{r\}_h, H, \mu)$$

- Finite horizon MDPs with stationary transitions / non-stationary transitions

Bellman equations and optimal policies for the finite horizon MDPs

- Even if P and r are stationary
  - the V functions are Q functions are not.

- By the Markovian property, it suffices to consider "nonstationary" but "memoryless" policies.
  - There exists a deterministic / memoryless optimal policy.

Other aspects of finite-horizon MDPs

- Advantage function and Performance Difference Lemma

$$V^\pi - V^{\widetilde{\pi}} = \sum_{h=0}^{H-1} \mathbb{E}_{s,a\sim\mathbb{P}_h^\pi}\left[A_h^{\widetilde{\pi}}(s,a)\right]$$

$$A_h^\pi(s,a) = Q_h^\pi(s,a) - V_h^\pi(s)$$

- Simulation lemma (HW1, last question)
- LP-formulation and occupancy measures
- Sample complexities under a generative model setting

Two-way reductions between finite horizon MDPs and infinite horizon / discounted MDPs

- Infinite horizon ➔ finite horizon
  - Clip at O(1/(1-γ)).
  - Define time-varying rewards.

- Finite horizon ➔ infinite horizon
  - The last step transitions into an absorbing state with self-loops and zero rewards.
  - Discounting factor γ set to be 1.

Two-way reductions between finite-H MDPs with stationary and non-stationary transitions.

- Stationary ➔ Non-stationary

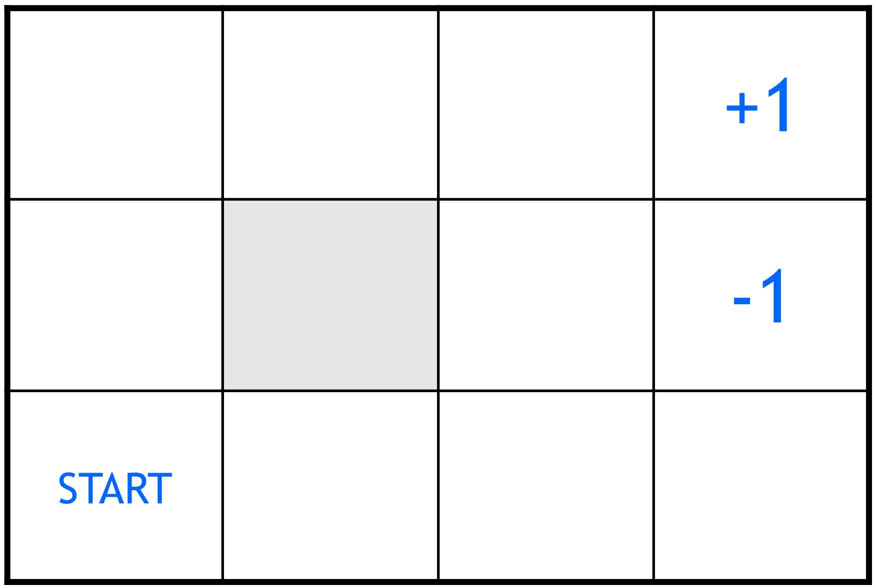
- Non-stationary ➔ Stationary

Other MDP settings that we will not consider in this course

- Infinite-horizon average reward MDPs
  - Usually require additional conditions for this to be well-defined.
- Indefinite-horizon setting
  - H is a random variable
  - e.g. Frozen-lake / Mountain car / other navigation tasks
  - Tricky issue: not invariant to scaling / translation of the rewards.

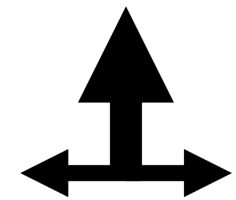
***We are not going to cover these settings in this course.**

Example: Frozen lake.



actions: UP, DOWN, LEFT, RIGHT

UP e.g.,



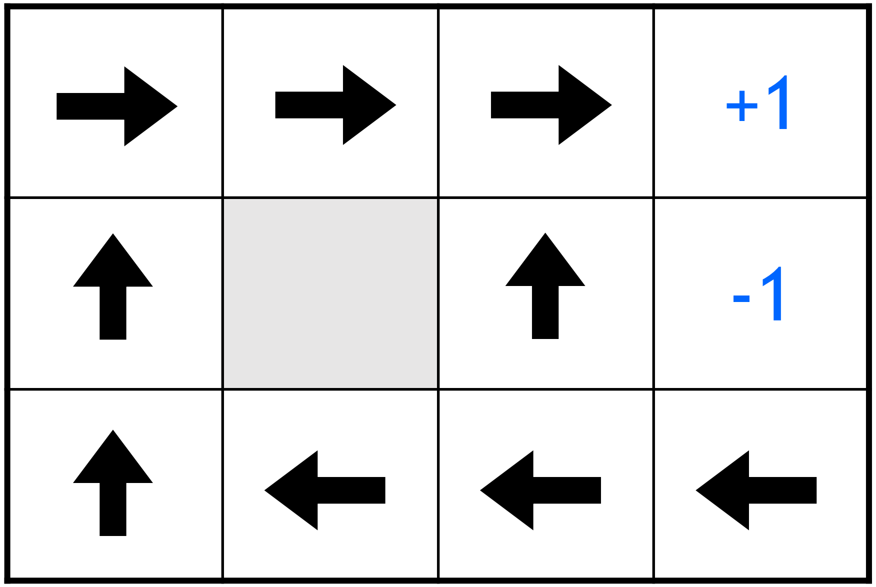
State-transitions with action **UP**:

80% move up
10% move left
10% move right

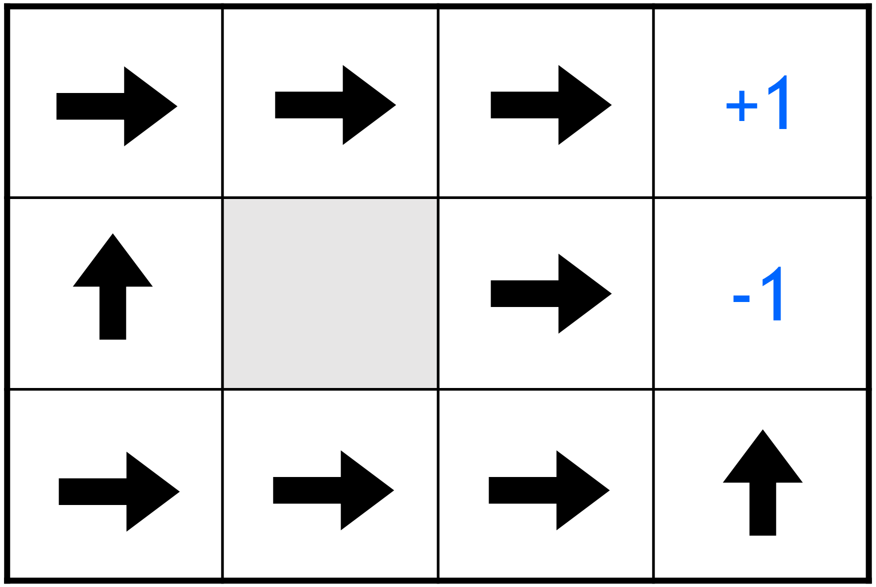
*If you bump into a wall, you stay where you are.

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step
- Finite horizon or infinite horizon?
- What is a good policy?

Optimal policies in the different reward settings



reward **-0.04** for each step



reward **-2** for each step

What if there is a positive reward for each step?

Partially Observed MDPs

- POMDP:
  - Estimate belief states (posterior distribution of state given history, i.e., Kalman filter)
  - Take actions according to the belief state.
- Computational considerations
  - MDP-planning: P-complete
  - POMDP-planning: PSPACE-complete (harder than NP-complete)
  - MDP-learning: polynomial sample complexity
  - POMDP-learning: often not identifiable.

***We are not going to cover POMDP in this course, but good references are available.**

This lecture

1. Wrap up MDPs
   - Performance difference lemma and advantage decomposition (Readings: AJKS Section 1.6)
   - Remarks about **finite horizon / episodic MDPs.** (Readings: AJKS Section 1.2)

2. RL algorithms
   - Model-based vs Model-free RL algorithms
   - Temporal difference learning. (Sutton and Barto Ch 5-6)
   - TD learning with linear function approximation.

Recap: Policy Iterations and Value Iterations

- What are these algorithms for?
  - Algorithms of computing the V* and Q* functions from MDP parameters
- Policy Iterations

$$\pi_0 \rightarrow^{E} V^{\pi_0} \rightarrow^{I} \pi_1 \rightarrow^{E} V^{\pi_1} \rightarrow^{I} \ldots \rightarrow^{I} \pi^* \rightarrow^{E} V^*$$

- Value iterations

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k(s')]$$

- How do we make sense of them?
  - Recursively applying the Bellman equations until convergence.

*These methods are called "Dynamic Programming" approaches in Chap 4 of Sutton and Barto.

They are no longer valid in RL

- Policy Evaluation

$$V_{k+1}^{\pi}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \cancel{P(s'|s,a)}[\cancel{r(s,a,s')} + \gamma V_k^{\pi}(s')]$$

- Policy improvement

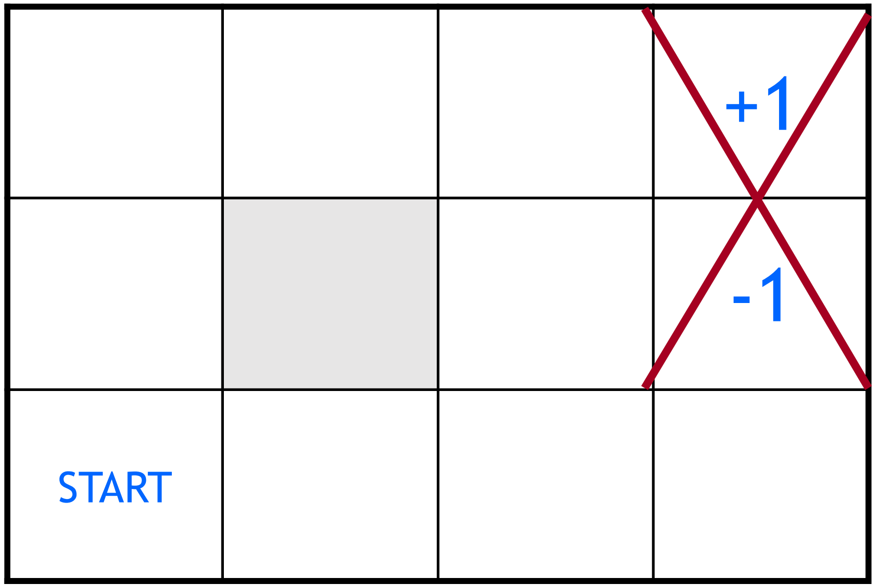
$$\pi'(s) = \arg\max_a Q^{\pi}(s,a)$$
$$= \arg\max_a \sum_{s'} \cancel{P(s'|s,a)}[\cancel{r(s,a,s')} + \gamma V_k^{\pi}(s')]$$

- Value iterations

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} \cancel{P(s'|s,a)}[\cancel{r(s,a,s')} + \gamma V_k(s')]$$

***We do not have the MDP parameters in RL!**

Example: Frozen lake

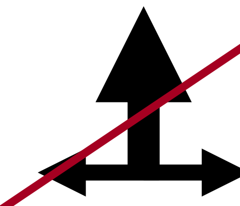


Action 1, Action 2, Action 3, Action 4

actions: ~~UP, DOWN, LEFT, RIGHT~~

UP

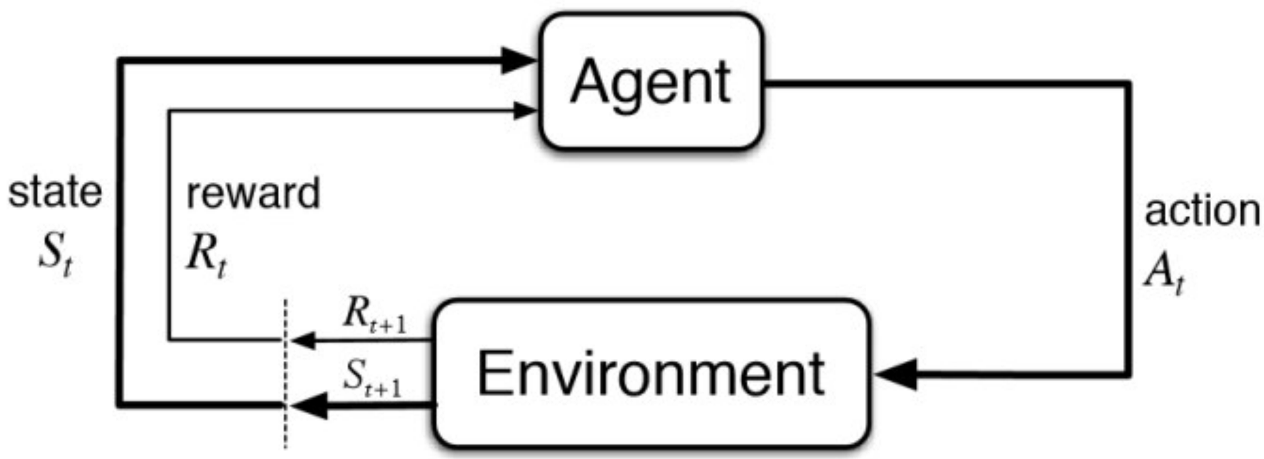
80% move UP
10% move LEFT
10% move RIGHT



- ~~reward +1 at [4,3], -1 at [4,2]~~
- ~~reward -0.04 for each step~~
- what’s the strategy to achieve max reward?

Instead, reinforcement learning agents have “online” access to an environment

- State, Action, Reward
- Unknown reward function, unknown state-transitions.
- Agents can “act” and “experiment”, rather than only doing offline planning.



Idea 1: **Model-based** Reinforcement Learning

- Model-based idea
  - Let's approximate the model based on experiences
  - Then solve for the values as if the learned model were correct

- Step 1: Get data by running the agent to explore
  - Many data points of the form: $\{(s_1, a_1, s_2, r_1), \dots, (s_N, a_N, s_{N+1}, r_N)\}$

- Step 2: Estimate the model parameters
  - $\widehat{P}(s'|s, a)$ --- plug-in / MLE. We need to observe the transition many times for each s, a
  - $\hat{r}(s', a, s)$ --- this is an estimate of the empirical rewards.

Then we can plug in these estimates and then use dynamic programming for policy evaluation / improvements.

$$V_{k+1}^{\pi}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \boxed{\hat{P}(s'|s,a)[\hat{r}(s,a,s')} + \gamma V_k^{\pi}(s')]$$

$$\pi' \leftarrow \arg\max_a \sum_{s'} \boxed{\hat{P}(s'|s,a)[\hat{r}(s,a,s')} + \gamma V_k^{\pi}(s')]$$

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} \boxed{\hat{P}(s'|s,a)[\hat{r}(s,a,s')} + \gamma V_k(s')]$$

* As usual, **"hat"** indicates empirical estimates.

* These iterations will produce $\hat{V}^*$ and $\hat{Q}^*$ functions, and then $\hat{\pi}^*$

This is OK if we have a generative model! But there are complications.

- For MDPs
  - Often we need to take a carefully chosen sequence of actions to reach a state
  - The chance of randomly running into a state can be **exponentially small,** if we decide to take random actions.
  - **Question: What is an example of this?**

*Need to somehow update the “exploration policy” on the fly!

More generally, model-based method is a algorithm design principle.

- We use function approximation on P

- Function classes:

- Simulation lemma still applies

$$Q^\pi - \widehat{Q}^\pi \quad = \quad \gamma(I - \gamma \widehat{P}^\pi)^{-1}(P - \widehat{P})V^\pi$$

  - If:
  - But:

Idea 2: **Model-free** Reinforcement Learning

- Do we need the model? Can we learn the Q function directly?
  - **How many free parameters are there to represent the Q-function?**

- Recall: Policy iterations

$$\pi_0 \rightarrow^E V^{\pi_0} \rightarrow^I \pi_1 \rightarrow^E V^{\pi_1} \rightarrow^I \ldots \rightarrow^I \pi^* \rightarrow^E V^*$$

  - Maybe we can do policy evaluation / value iterations without estimating the model?

Model-free method is yet another algorithm design principle

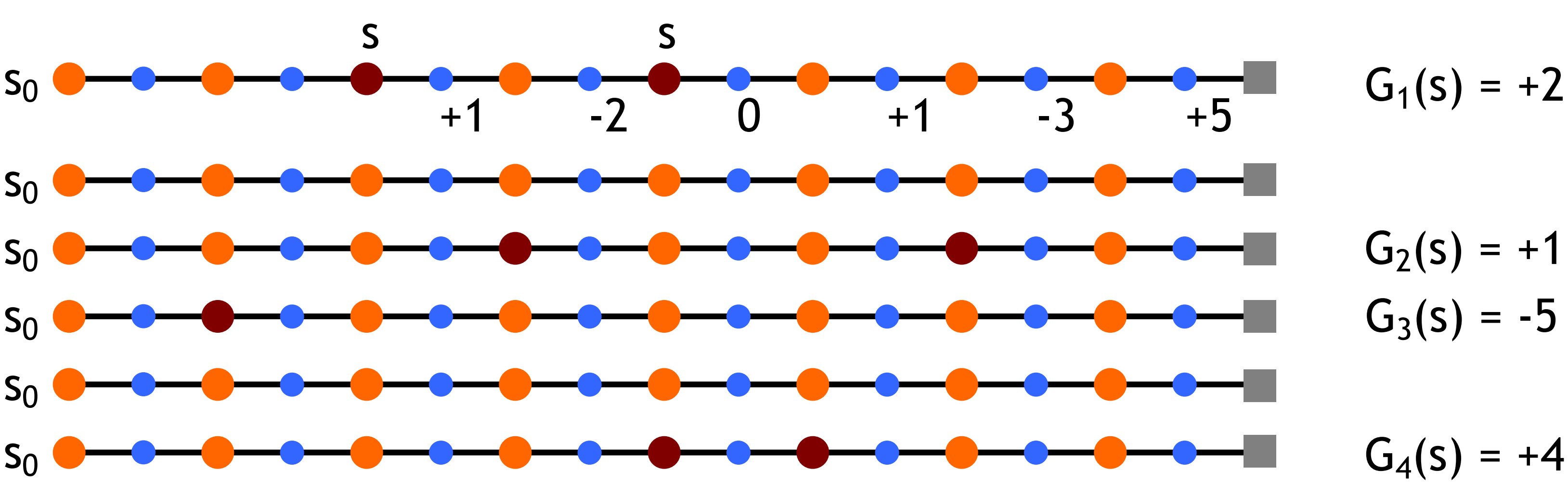
- We use function approximation on Q directly

- Function classes

- Induced policy class

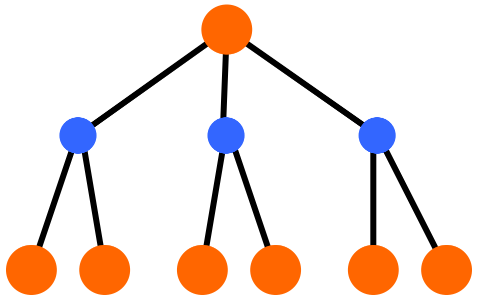
Monte Carlo Policy Evaluation (Prediction)

- want to estimate $V^{\pi}(s)$
  = expected return starting from s and following π
  - estimate as average of observed returns in state s
- We can execute the policy π
- first-visit MC
  - average returns following the first visit to state s



$V^{\pi}(s) \approx (2 + 1 - 5 + 4)/4 = 0.5$

Monte Carlo Policy Optimization (Control)



- V^π not enough for policy improvement
  - need exact model of environment
- estimate $Q^\pi(s,a)$

$$\pi'(s) = \arg\max_a Q^\pi(s,a)$$

- MC control

$$\pi_0 \xrightarrow{E} Q^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} Q^*$$

  - update after each episode
- Two problems
  - greedy policy won't explore all actions eps-greedy, or bonus design.
  - Requires many independent episodes for the estimated value function to be accurate.

Improved Monte-Carlo Q-function estimate using Bellman equations

- Recall:

$$Q^\pi(s,a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s',a')]$$

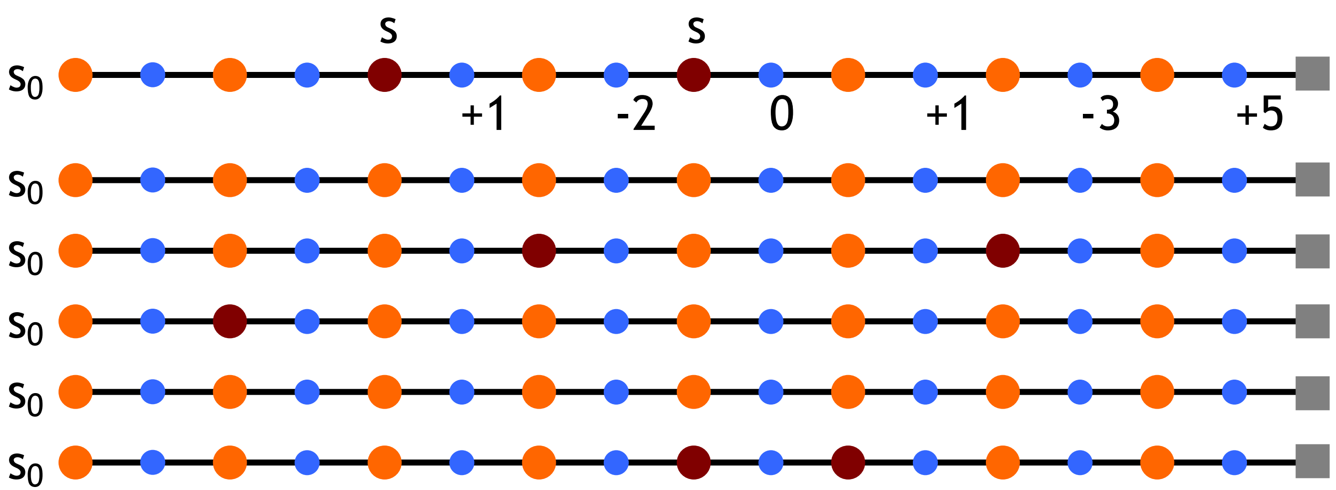
$$Q^\pi(s,a) = r^\pi(s,a) + \gamma \mathbb{E}_{s' \sim P(s'|s,a)}[V^\pi(s')]$$

- We can use the empirical (Monte Carlo) estimate.

$$\widehat{Q^\pi}(s,a) = \widehat{r^\pi}(s,a) + \gamma \widehat{\mathbb{E}}_{s' \sim P(s'|s,a)}[\widehat{V}^\pi(s')]$$

*No need to estimate P(s' | s,a) or r(s,a,s') as intermediate steps.
*Require only O(SA) space, rather than O(S^2A)

Online averaging representation of MC



$G_1(s) = +2$

$G_2(s) = +1$

$G_3(s) = -5$

$G_4(s) = +4$

$V^\pi(s) \approx (2 + 1 - 5 + 4)/4 = 0.5$

- Alternative, *online averaging* update

$$V(S_t) \leftarrow V(S_t) + \alpha\left[G_t - V(S_t)\right], \quad \text{where } \alpha = 1/N_{S_t}$$

DP + MC = Temporal Difference Learning

- Monte Carlo $V(S_t) \leftarrow V(S_t) + \alpha\Big[G_t - V(S_t)\Big],$

Issue: G_t can only be obtained after the entire episode!

- The idea of TD learning:

$$\mathbb{E}_\pi[G_t] = \mathbb{E}_\pi[R_t|S_t] + \gamma V^\pi(S_{t+1})$$

We only need one step before we can plug-in and estimate the RHS!

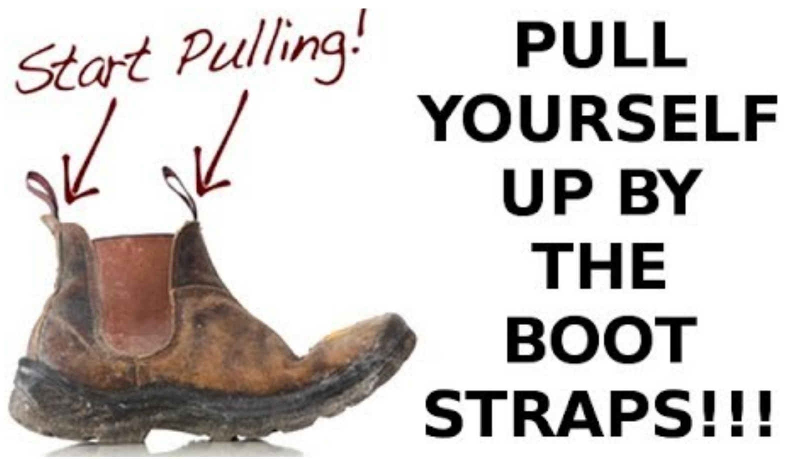
- TD-Policy evaluation

Bootstrapping!

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big]$$

Bootstrap’s origin

- “The Surprising Adventures of Baron Münchausen”
  - Rudolf Erich Raspe, 1785



- In statistics: Brad Efron’s resampling methods
- In computing: Booting…
- In RL: It simply means TD learning

TD policy optimization (TD-control)

- SARSA (On-Policy TD-control)
  - Update the Q function by bootstrapping Bellman Equation

$$Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$$

  - Choose the next A’ using Q, e.g., eps-greedy.

- Q-Learning (Off-policy TD-control)
  - Update the Q function by bootstrapping Bellman Optimality Eq.

$$Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$$

  - Choose the next A’ using Q, e.g., eps-greedy, or any other policy.

Remarks:

- These are **proven to converge** asymptotically.
- Much more data-efficient in practice, than MC.
- Regret analysis is still active area of research.

Advantage of TD over Monte Carlo

- Given a trajectory, a roll-out, of T steps.
  - MC updates the Q function only once
  - TD updates the Q function (and the policy) T times!

Remark: This is the same kind of improvement from Gradient Descent to Stochastic Gradient Descent (SGD).

Model-free vs Model-based RL algorithms

- Different function approximations

- Different space efficiency

- Which one is more statistically efficient?
  - More or less equivalent in the tabular case.
  - Different challenges in their analysis.