Basic RL Settings

Monte-Carlo policy evaluation

- Given π, estimate $J(\pi) := \mathbb{E}_{s\sim d_0}[V^\pi(s)]$
- Alg outputs v; evaluated by $|v - J(\pi)|$
- Data: trajectories starting from $s_1 \sim d_0$ using π (i.e., $a_t = \pi(s_t)$)

$$\{(s_1^{(i)}, a_1^{(i)}, r_1^{(i)}, s_2^{(i)}, \ldots, s_H^{(i)}, a_H^{(i)}, r_H^{(i)})\}_{i=1}^n$$

(for simplicity, assume process terminates in H time steps)

- Algorithm: output $\frac{1}{n}\sum_{i=1}^n \sum_{t=1}^H \gamma^{t-1} r_t^{(i)}$
- Guarantee: w.p. at least $1 - \delta$, $|v - J(\pi)| \leq \frac{R_{\max}}{1-\gamma}\sqrt{\frac{1}{2n}\ln\frac{2}{\delta}}$
  - Depends on value range & sample size
  - No dependence on anything else, e.g., state/action spaces

Monte-Carlo optimization

- Want to optimize $J(\pi) := \mathbb{E}_{s\sim d_0}[V^\pi(s)]$; have parameterized π_θ.
- For each θ, roll-out trajectories using π_θ to estimate $J(\pi_\theta)$
- Pick a bunch of θ, estimate return, pick the best
- In general, reduce to 0-th order optimization
  - e.g., CMA-ES for RL
- Guarantee: depends on optimization

Model-based RL w/ sampling oracle

- Want to optimize $J(\pi) := \mathbb{E}_{s\sim d_0}[V^\pi(s)]$, tabular state space
- Sampling oracle: can generate $s'\sim P(s, a)$ for any (s, a)
- Use a total of $n|S\times A|$ samples: n samples per (s, a)
- Estimate transition probabilities
- Plan in the estimated model
- Guarantee (will analyze later) depends on:
  - Sample size, value range
  - Horizon (error compounding)
  - $|S\times A|$ ("curse of dimensionality")

Categorization of RL settings

What's the goal?

- Policy evaluation: given π, estimate its value.
  - Estimate $J(\pi) := \mathbb{E}_{s\sim d_0}[V^\pi(s)]$
    e.g., alg outputs v; evaluated by $|v - J(\pi)|$
  - Estimate the entire value function
    e.g., alg outputs $V \in \mathbb{R}^{\mathcal{S}}$; evaluated by $\|V - V^\pi\|_\infty$
- Policy optimization
  - Optimize $J(\pi) := \mathbb{E}_{s\sim d_0}[V^\pi(s)]$
    e.g., alg outputs π; evaluated by $J(\pi^*) - J(\pi)$
  - No particular initial state
    e.g., alg outputs π; evaluated by $\|V^* - V^\pi\|_\infty$

Where do data come from?

- Passively given
  - For policy optimization: usually needs to be exploratory e.g., fixed number of samples from each (s,a)
  - For policy evaluation of π
    - Data generated w/ π: on-policy (somewhat easy)
    - Data generated w/ π': off-policy (requires counter-factual reasoning)
- Collect own data
  - Can query any (s,a): planning w/ “generative model”
  - Can only roll-out trajectories: exploration!

What's the basic principle?

Monte-Carlo

- e.g., roll-out trajectories to estimate return
- e.g., policy gradient, evolutionary strategies
- e.g., Monte-Carlo tree search

✓ Often no dependence on state space

✓ No compounding error

X Learning signals can be sparse

X Run into exponential variance when off-policy

What's the basic principle?

Dynamic programming

- use data to approximately solve Bellman Equation
- often in a "bottom-up" fashion (e.g., value iteration)

✗ "Curse of dimensionality" (dependence on state space)

✗ Error compounds over time

✓ Leverage immediate signals to learn

✓ Handles off-policy data w/o exponential variance

Manageable state spaces?

Yes: tabular RL

No (sample size << state space): function approximation.

Approximate what?

- Model?
- Value function?
- Policy?