The Learning Setting

RL: Planning or Learning?

- So far we have considered planning
  - i.e., given MDP model, how to compute optimal policy
  - More broadly, whenever the MDP model (i.e., reward & transition functions) is known, it is the planning setting
- Learning: MDP model is unknown, but we are given/can collect data from the MDP (often in the form of (s, a, r, s'))
- Defining a concrete learning **problem** involves many factors…
  - Is data passively given (batch/offline/off-policy), or we can collect ourselves and decide how to act (online)?
  - Is data a bag of 4-tuples, or are they in the form of trajectories?
  - Are we interested in policy evaluation or optimization?
  - …

RL: Planning or Learning?

- Learning can be useful even if the final goal is planning
  - esp. when |S| is large and/or only blackbox simulator
  - e.g., AlphaGo, video game playing, simulated robotics
  - "Sampling-based planning"—what RL has been mostly about historically (despite the word "learning" in its name!)
  - Can run simulator to generate data indefinitely
  - Major concern: computational complexity
- Learning as a problem
  - e.g., adaptive medical treatment, dialog systems
  - Data is limited. Sample complexity (data efficiency) is as important as computational complexity
  - Additional concerns about e.g., safety

Simplest Setting: Monte-Carlo policy evaluation

- Given π, estimate $J(\pi) := \mathbb{E}_{s\sim d_0}[V^\pi(s)]$ (d_0 is initial state distribution)
- Alg outputs some scalar v; accuracy measured 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)

- Estimator: $\frac{1}{n}\sum_{i=1}^n \sum_{t=1}^H \gamma^{t-1} r_t^{(i)}$
- Guarantee: w.p. at least $1 - \delta$, $|v - J(\pi)| \le \frac{R_{\max}}{1-\gamma}\sqrt{\frac{1}{2n}\ln\frac{2}{\delta}}$
  - Direct consequence of Hoeffding's inequality (not required)
  - Depends on value range & sample size
  - No dependence on anything else, e.g., state/action spaces

What does "Monte-Carlo" mean?

- Suppose we want to know the value of $\mathbb{E}_{x \sim p}[f(x)]$
- Monte-Carlo estimate: draw $x_1, x_2, \ldots, x_n$ i.i.d. from p; estimator: $\frac{1}{n}\sum_{i=1}^{n} f(x_i)$
- Beauty of MC: if the value f takes has bounded range, the approximation guarantee of MC has **no dependence** on the cardinality of the X space
- Mapping things to policy evaluation: x is a trajectory, f maps the trajectory to the discounted return, p is the distribution of the trajectory determined by the MDP, the initial state distribution, and the policy
- In RL, Monte-Carlo generally means forming estimates by rolling out trajectories, typically without using concepts from Bellman equations

Turning Monte-Carlo policy evaluation into a policy optimization algorithm

- Want to optimize $J(\pi) := \mathbb{E}_{s \sim d_0}[V^\pi(s)]$
- have a set of candidate policies
- Estimate the expected return of each candidate, pick the best
- Limitation: can only evaluate a small number of policies
  - 0-th order optimization heuristics can be applied (e.g., CMA-ES for RL; look up the term and do some readings if you are interested); typically no guarantees
- Even if the MDP has finite & small state/action spaces ("tabular RL"), finding optimal policy using this strategy takes exponential sample/computational complexity

Model-based RL with a sampling oracle

- Assume we can sample $r \sim R(s, a)$ and $s' \sim P(s, a)$ for any (s, a)
- Collect n samples per (s, a): $\{(r_i, s_i')\}_{i=1}^{n}$. Total sample size $n|S \times A|$
- Estimate an empirical MDP $\hat{M}$ from data
  - $\hat{R}(s, a) := \frac{1}{n} \sum_{i=1}^{n} r_i$, $\hat{P}(s' \,|\, s, a) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[s_i' = s']$
  - i.e., treat the empirical frequencies of states appearing in $\{s_i'\}_{i=1}^{n}$ as the true distribution
- Plan in the estimated model and return the optimal policy
- Guarantee (not required): to make sure that the optimal policy of $\hat{M}$ is ε-optimal in the true MDP with probability at least 1-δ, we need a total sample size of $poly(\,|S|, |A|, 1/(1-\gamma), 1/\epsilon, 1/\delta)$
- Can be applied on an arbitrarily generated dataset; works as long as each (s,a) has enough samples.

Model-based RL with a sampling oracle

- Useful as an efficient approximate planner for tabular MDPs with moderately large state spaces
- Exact value iteration: O(|S|^2 |A|) computation per iteration
- With sampled data: O(|S||A|n) per iteration
  - Note: you don't even need to explicitly build $\hat{M}$!
  - Bellman update at (s,a): $\hat{R}(s,a) + \gamma \mathbb{E}_{s' \sim \hat{P}(s,a)}[\max_{a'} f(s',a')]$
    - equal to $\frac{1}{n}\sum_{i=1}^{n}\left(\underline{r_i + \gamma \max_{a'} f(s_i',a')}\right)$ → "empirical Bellman update"
- In practice, n = 20 is usually sufficient
  - Even if |S| is much larger!
  - which means the transition distributions are estimated very poorly… but we can still find optimal policy, and this is backed up by theory (won't be covered in this course)

Model-based RL with a sampling oracle

- Can also be applied to policy evaluation—in fact you can do almost everything given that you have a generative model of the world (though it's approximate)
- Also known under the name "certainty-equivalence"
- Will switch gears to other methods, and mention connection later