CS 295:
Optimal Control and Reinforcement Learning

Winter 2020

Lecture 9: Planning

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

Today's lecture

- TRPO
- Planning
  - With a fast simulator — MCTS
  - With an arbitrary-reset simulator — VI
  - With a differentiable model — iLQR / DDP

Off-policy Policy Gradient

$$\mathcal{J}_\theta = \mathbb{E}_{\xi\sim p_{\theta'}}\left[\frac{p_\theta(\xi)}{p_{\theta'}(\xi)}R(\xi)\right]$$

$$\nabla_\theta\mathcal{J}_\theta = \mathbb{E}_{\xi\sim p_{\theta'}}\left[\frac{\nabla_\theta p_\theta(\xi)}{p_{\theta'}(\xi)}R(\xi)\right]$$

$$\nabla_\theta\mathcal{J}_\theta = \mathbb{E}_{\xi\sim p_{\theta'}}\left[\frac{\nabla_\theta p_\theta(\xi)}{p_{\theta'}(\xi)}R(\xi)\right] = \mathbb{E}_{\xi\sim p_{\theta'}}\left[\frac{p_\theta(\xi)}{p_{\theta'}(\xi)}\nabla_\theta \log p_\theta(\xi)R(\xi)\right]$$

$$= \mathbb{E}_{\xi\sim p_{\theta'}}\left[\prod_t \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}\sum_{t'}\nabla_\theta \log \pi_\theta(a_{t'}|s_{t'})\sum_{t''}\gamma^{t''}r_{t''}\right]$$

forward

$$= \mathbb{E}_{\xi\sim p_{\theta'}}\left[\sum_{t'}\prod_{t\leqslant t'} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}\nabla_\theta \log \pi_\theta(a_{t'}|s_{t'})\sum_{t''\geqslant t'}\gamma^{t''}r_{t''}\right]$$

backward

Off-policy Policy Gradient: approximation

$$\nabla_\theta \mathcal{J}_\theta = \mathbb{E}_{\xi \sim p_{\theta'}}\left[\sum_{t'} \prod_{t \leqslant t'} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \sum_{t'' \geqslant t'} \gamma^{t''} r_{t''}\right]$$

$$= \sum_{t'} \mathbb{E}_{s_{t'}, a_{t'} \sim p_{\theta'}}\left[C_{\theta,\theta',t'} \frac{\pi_\theta(a_{t'}|s_{t'})}{\pi_{\theta'}(a_t|s_t)} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \hat{A}_t\right]$$

- $C_{\theta,\theta',t'}$ is the IS coefficient of past actions, marginalized
    - Originally just ignored ¯_(ツ)_/¯

More analysis

$$\sum_t \gamma^t \hat{A}^1_{\pi_\theta}(s_t, a_t) = \sum_t \gamma^t (r(s_t, a_t) + \gamma V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t))$$

$$= \sum_t \gamma^t r(s_t, a_t) - V_{\pi_\theta}(s_0)$$

$$\mathbb{E}_{\xi \sim p_{\theta'}}\left[\sum_t \gamma^t \hat{A}^1_{\pi_\theta}(s_t, a_t)\right] = \mathbb{E}_{\xi \sim p_{\theta'}}\left[\sum_t \gamma^t r(s_t, a_t) - V_{\pi_\theta}(s_0)\right] =$$

More analysis

$$\begin{aligned}\sum_t \gamma^t \hat{A}^1_{\pi_\theta}(s_t, a_t) &= \sum_t \gamma^t (r(s_t, a_t) + \gamma V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t)) \\ &= \sum_t \gamma^t r(s_t, a_t) - V_{\pi_\theta}(s_0)\end{aligned}$$

$$\begin{aligned}\mathbb{E}_{\xi \sim p_{\theta'}}\left[\sum_t \gamma^t \hat{A}^1_{\pi_\theta}(s_t, a_t)\right] &= \mathbb{E}_{\xi \sim p_{\theta'}}\left[\sum_t \gamma^t r(s_t, a_t) - V_{\pi_\theta}(s_0)\right] = \mathcal{J}_{\theta'} - \mathcal{J}_\theta \\ &= \sum_t \gamma^t \, \mathbb{E}_{s_t, a_t \sim p_{\theta'}}[\hat{A}^1_{\pi_\theta}(s_t, a_t)] \\ &= \sum_t \gamma^t \, \mathbb{E}_{s_t \sim p_{\theta'}}\left[\mathbb{E}_{a_t | s_t \sim \pi_\theta}\left[\frac{\pi_{\theta'}(a_t | s_t)}{\pi_\theta(a_t | s_t)} \hat{A}^1_{\pi_\theta}(s_t, a_t)\right]\right]\end{aligned}$$

- Can we switch to $s_t \sim p_\theta$, so we can estimate the expectation empirically?

Change of measure

- Intuition: switching from $s_t \sim p_{\theta'}$ to $s_t \sim p_\theta$ isn't too bad if they are similar

$$\delta(q,p) = \frac{1}{2}|q-p|_1 \leqslant \sqrt{\frac{1}{2}\mathbb{D}[q\|p]}$$

- Suppose $\forall s \qquad |\pi_{\theta'}(\cdot|s) - \pi_\theta(\cdot|s)|_1 \leqslant 2\sqrt{\epsilon/2} = \epsilon'$

$$|\,\mathbb{E}_{s_t\sim p_{\theta'}}[f(s)] - \mathbb{E}_{s_t\sim p_\theta}[f(s_t)]| \leqslant |p_{\theta'} - p_\theta|_1 \max_{s_t} f(s_t) \leqslant t\epsilon' \max_{s_t} f(s_t)$$

Trust-Region Policy Optimization (TRPO)

$$\max_{\theta'} \sum_t \gamma^t \, \mathbb{E}_{s_t \sim p_\theta} \left[\mathbb{E}_{a_t | s_t \sim \pi_\theta} \left[\frac{\pi_{\theta'}(a_t | s_t)}{\pi_\theta(a_t | s_t)} \hat{A}^1_{\pi_\theta}(s_t, a_t) \right] \right]$$

$$\text{s.t.} \quad \mathbb{D}[\pi_{\theta'} \| \pi_\theta] \leqslant \epsilon$$

- For small ϵ, the objective is close to $\mathcal{J}_{\theta'} - \mathcal{J}_\theta$
    - Guarantees improvement

$$\mathcal{L}_\theta(s, a, r, s') = -\frac{\pi_\theta(a|s)}{\pi_{\bar{\theta}}(a|s)}(r + \gamma V_\phi(s') - V_\phi(s)) + \lambda(\mathbb{D}[\pi_\theta(\cdot|s) \| \pi_{\bar{\theta}}(\cdot|s)] - \epsilon)$$

- The actual algorithm is somewhat complicated; simpler variant: PPO

Planning

- Planning is finding a good policy when we "know" the MDP
    - Dynamics + reward function
- What does it mean to have a "known model"?
    - A really fast simulator
    - A simulator that can be reset to any given state
    - A differentiable model
    - An analytic model that can be manipulated symbolically

How to use a really fast simulator

- MC policy evaluation
  - Sample many trajectories using the greedy policy
  - Evaluate by optimizing the loss $\mathcal{L}_\theta(\xi) = (Q_\theta(s_0, a_0) - R)^2$
- The greedy policy doesn't explore
  - Can use near-greedy exploration policy
- How to explore optimally? Very little is known in this case.

Deterministic dynamics

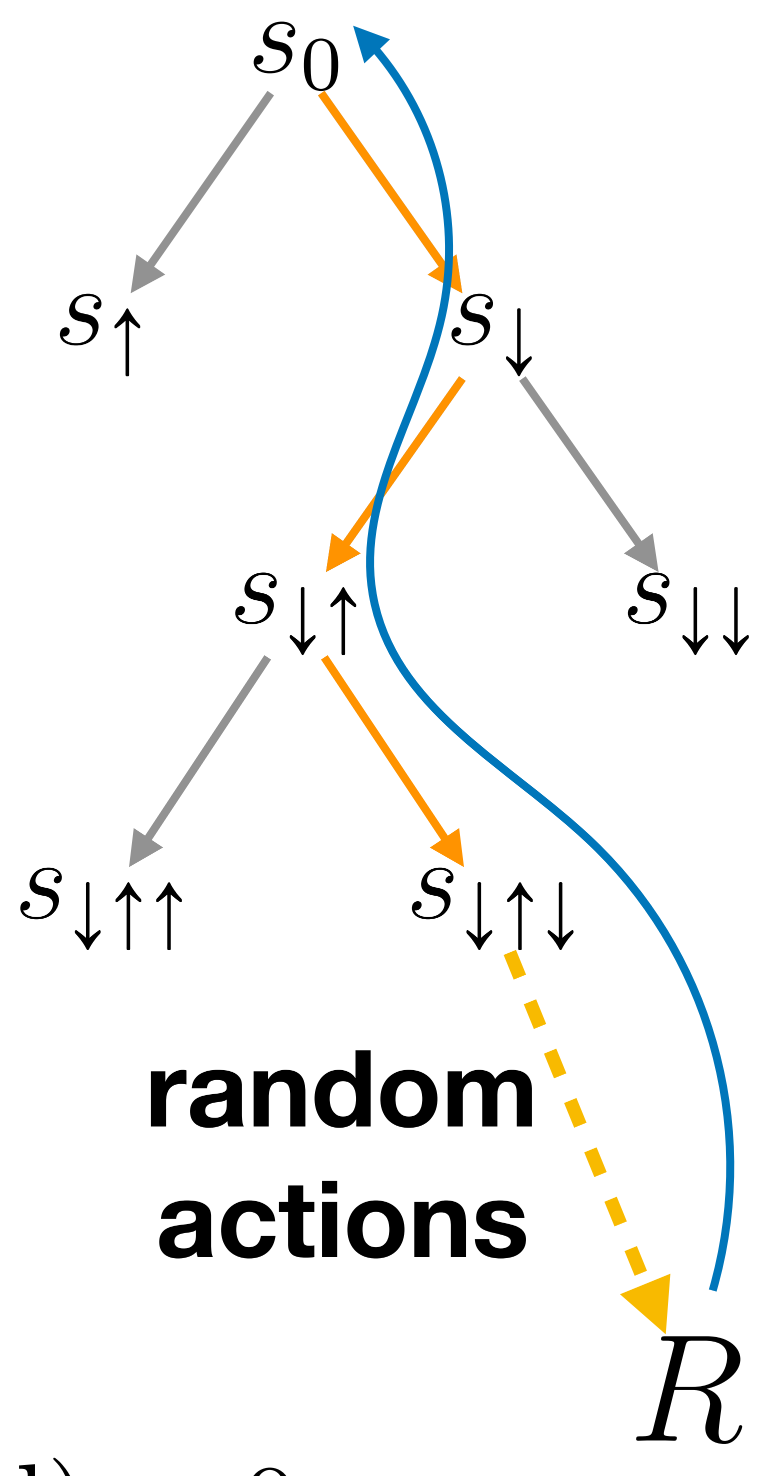
- With deterministic dynamics, policy can be just a sequence of actions

$$\max_{\vec{a}} R(\vec{a}) = \max_{\vec{a}} r(s_0, a_0) + \gamma r(f(s_0, a_0), a_1) + \gamma^2 r(f(f(s_0, a_0), a_1), a_2) + \cdots$$

- Can use Cross Entropy Method (CEM)
    - Sample $\vec{a}_1, \ldots, \vec{a}_N$ from π
    - Take top ${}^{N}\!/_{c}$ "elite" samples
    - Fit π to the elites
    - Repeat
- Scales poorly with the dimension of $\vec{a}$

Discrete action space: optimal exploration

- Action sequences have a tree structure
  - Shallow (short) prefixes are visited often → possible to learn their value
  - Deep (long) sequences are visited rarely → we can only explore
- Monte-Carlo Tree Search (MCTS):
  - Select leaf
  - Explore to end of episode
  - Update nodes along branch to leaf



- Selecting a leaf: recursively maximize $\begin{cases} \infty & N(\text{child}) = 0 \\ V(\text{child}) + C\sqrt{\frac{\log N(\text{self})}{N(\text{child})}} & \text{otherwise} \end{cases}$

How to use an arbitrary-reset simulator

- With small state space: value iteration with table parametrization

$$V(s_i) \leftarrow \max_a (r(s_i, a) + \gamma \, \mathbb{E}_{s'|s_i,a\sim p}[V(s')])$$

- But simulator is not much help under function approximation
  - Distribution should support $p_\theta(s)$ to avoid covariate shift (train–test mismatch)
  - Simulator does enable data augmentation

How to use a differentiable model

- Suppose we have differentiable $x_{t+1} = f(x_t, u_t)$ and $c(x_t, u_t)$
- Taylor expansion at a trajectory $(\hat{x}, \hat{u})$:

$$f(x_t, u_t) = f(\hat{x}_t, \hat{u}_t) + O(\epsilon)$$

How to use a differentiable model

- Suppose we have differentiable $x_{t+1} = f(x_t, u_t)$ and $c(x_t, u_t)$
- Taylor expansion at a trajectory $(\hat{x}, \hat{u})$:

$$f(x_t, u_t) = f(\hat{x}_t, \hat{u}_t) + \nabla_x \hat{f}_t \delta x_t + \nabla_u \hat{f}_t \delta u_t + O(\epsilon^2)$$

How to use a differentiable model

- Suppose we have differentiable $x_{t+1} = f(x_t, u_t)$ and $c(x_t, u_t)$
- Taylor expansion at a trajectory $(\hat{x}, \hat{u})$:

$$f(x_t, u_t) = f(\hat{x}_t, \hat{u}_t) + \nabla_x \hat{f}_t \delta x_t + \nabla_u \hat{f}_t \delta u_t + O(\epsilon^2)$$

$$c(x_t, u_t) = c(\hat{x}_t, \hat{u}_t) + O(\epsilon)$$

How to use a differentiable model

- Suppose we have differentiable $x_{t+1} = f(x_t, u_t)$ and $c(x_t, u_t)$
- Taylor expansion at a trajectory $(\hat{x}, \hat{u})$:

$$f(x_t, u_t) = f(\hat{x}_t, \hat{u}_t) + \nabla_x \hat{f}_t \delta x_t + \nabla_u \hat{f}_t \delta u_t + O(\epsilon^2)$$

$$c(x_t, u_t) = c(\hat{x}_t, \hat{u}_t) + \nabla_x \hat{c}_t \delta x_t + \nabla_u \hat{c}_t \delta u_t + O(\epsilon^2)$$

How to use a differentiable model

- Suppose we have differentiable $x_{t+1} = f(x_t, u_t)$ and $c(x_t, u_t)$
- Taylor expansion at a trajectory $(\hat{x}, \hat{u})$:

$$f(x_t, u_t) = f(\hat{x}_t, \hat{u}_t) + \nabla_x \hat{f}_t \delta x_t + \nabla_u \hat{f}_t \delta u_t + O(\epsilon^2)$$

$$\begin{aligned} c(x_t, u_t) &= c(\hat{x}_t, \hat{u}_t) + \nabla_x \hat{c}_t \delta x_t + \nabla_u \hat{c}_t \delta u_t \\ &\quad + \frac{1}{2}(\delta x_t^\intercal \nabla_x^2 \hat{c}_t \delta x_t + \delta u_t^\intercal \nabla_u^2 \hat{c}_t \delta u_t + 2\delta x^\intercal \nabla_{xu} \hat{c}_t \delta u_t) + O(\epsilon^3) \end{aligned}$$

Iterative LQR (iLQR)

Algorithm 1 iLQR

```
compute $A, B \leftarrow \nabla_x \hat{f}_t, \nabla_u \hat{f}_t$
compute $Q, R, N, q, r \leftarrow \nabla^2_x \hat{c}_t, \nabla^2_u \hat{c}_t, \nabla_{xu} \hat{c}_t, \nabla_x \hat{c}_t, \nabla_u \hat{c}_t$
$\hat{L}_t, \hat{\ell}_t \leftarrow$ LQR on $\delta x_t = x_t - \hat{x}_t$, $\delta u_t = u_t - \hat{u}_t$
$\delta x^*, \delta u^* \leftarrow$ execute policy $\delta u_t = \hat{L}_t \delta x_t + \hat{\ell}_t$ in the simulator / environment
$\hat{x} \leftarrow \hat{x} + \delta x^*$, $\hat{u} \leftarrow \hat{u} + \delta u^*$
repeat to convergence
```

Newton's method

- Compare to Newton's method for optimizing $\min_x f(x)$

```
Algorithm 1 Newton's method
  $g \leftarrow \nabla_x \hat{f}$
  $H \leftarrow \nabla_x^2 \hat{f}$
  $\hat{x} \leftarrow \operatorname{argmin}_x \frac{1}{2}\delta x^\intercal H \delta x + g^\intercal \delta x$
  repeat to convergence
```

- iLQR approximates this method for $\min_u \mathcal{J}(u)$
  - Exactly Newton's method would be expanding the dynamics to 2nd order — Differential Dynamic Programming (DDP)

Recap

- TRPO approximates Off-Policy Policy Gradient in a tractable way
    - While constraining the policy to a region where the approx can be trusted
- A fast simulator is good for any RL algorithm, particularly MC
    - MCTS explores optimally in the discrete deterministic case
- An arbitrary-reset simulator has surprisingly little use
- We can plan in a differentiable model by iterative linearization (iLQR)