CS 287 Advanced Robotics (Fall 2019)

Lecture 8: Optimization for (Locally) Optimal Control

Pieter Abbeel

UC Berkeley EECS

Optimal Control -- Approaches

	Return open-loop controls $u_0, u_1, \ldots, u_H$	Return feedback policy $\pi_\theta(\cdot)$ (e.g. linear or neural net)
shooting	$\min_{u_0,u_1,\ldots,u_H} c(x_0,u_0) + c(f(x_0,u_0),u_1) + c(f(f(x_0,u_0),u_1),u_2) + \ldots$	$\min_\theta c(x_0,\pi_\theta(x_0)) + c(f(x_0,\pi_\theta(x_0)),\pi_\theta(f(x_0,\pi_\theta(x_0)))) + \ldots$
collocation	$\min_{x_0,u_0,x_1,u_1,\ldots,x_H,u_H} \sum_{t=0}^{H} c(x_t,u_t)$ s.t. $x_{t+1} = f(x_t,u_t)\ \forall t$	$\min_{x_0,x_1,\ldots,x_H,\theta} \sum_{t=0}^{H} c(x_t,\pi_\theta(x_t))$ s.t. $x_{t+1} = f(x_t,\pi_\theta(x_t))\ \forall t$ $\min_{x_0,u_0,x_1,u_1,\ldots,x_H,u_H,\theta} \sum_{t=0}^{H} c(x_t,u_t)$ s.t. $x_{t+1} = f(x_t,u_t)\ \forall t$, $u_t = \pi_\theta(x_t)\ \forall t$

3rd Axis of Choice

- Roll-out $u_0, u_1, \ldots, u_H$ or $\pi_\theta(\cdot)$

OR:

- Model-Predictive Control (MPC)
  - Just take the first action u_0 or $\pi_\theta(x_0)$ then resolve the optimization problem from time t=1..H
  - Repeat for time t=1, 2, … H

E.g.: Collocation + Open-loop + MPC

Given: $\bar{x}_0$

`for` t=0, 1, 2, …, T

- Solve
$$\begin{aligned} \min_{x,u} \quad & \sum_{k=t}^{T} c_k(x_k, u_k) \\ \text{s.t.} \quad & x_{k+1} = f(x_k, u_k), \qquad \forall k \in \{t, t+1, \ldots, T-1\} \\ & x_t = \bar{x}_t \end{aligned}$$
- Execute u_t
- Observe resulting state, $\bar{x}_{t+1}$

Computational trick (often critical for low-latency control):

Initialize with solution from t - 1 to solve fast at time t

(In)stability of Open-loop Shooting

- Let's reconsider:

$$\min_{u_0,u_1,\ldots,u_H} c(x_0,u_0) + c(f(x_0,u_0),u_1) + c(f(f(x_0,u_0),u_1),u_2) + \ldots$$

  - Rolling out u_0, u_1, …, u_H can often be unstable because small numerical errors (or noise) can amplify in case of unstable dynamics
  - In turn, this can make this formulation unstable to optimize

- Solutions:
  - During roll-out, use MPC
  - Use the feedback version $\min_{\theta} c(x_0,\pi_\theta(x_0)) + c(f(x_0,\pi_\theta(x_0)),\pi_\theta(f(x_0,\pi_\theta(x_0)))) + \ldots$ (or collocation)

BackPropagation Through Time (BPTT)

- Let's reconsider shooting formulations:

$$\min_{u_0, u_1, \ldots, u_H} c(x_0, u_0) + c(f(x_0, u_0), u_1) + c(f(f(x_0, u_0), u_1), u_2) + \ldots$$

$$\min_{\theta} c(x_0, \pi_\theta(x_0)) + c(f(x_0, \pi_\theta(x_0)), \pi_\theta(f(x_0, \pi_\theta(x_0)))) + \ldots$$

- when computing derivatives w.r.t. u (or theta), the different terms share the same subcomputations

→ BackPropagation Through Time avoids duplicate computation

BackPropagation Through Time (BPTT)

- Let’s reconsider shooting formulations:

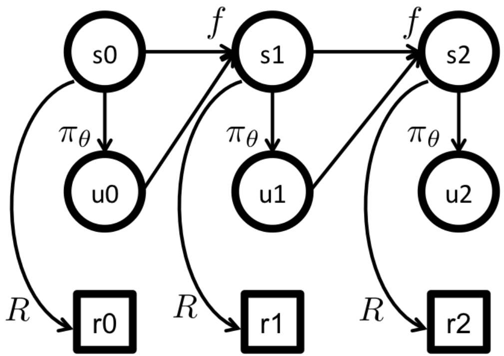
$$\min_{u_0, u_1, \ldots, u_H} c(x_0, u_0) + c(f(x_0, u_0), u_1) + c(f(f(x_0, u_0), u_1), u_2) + \ldots$$

$$\min_{\theta} c(x_0, \pi_\theta(x_0)) + c(f(x_0, \pi_\theta(x_0)), \pi_\theta(f(x_0, \pi_\theta(x_0)))) + \ldots$$

- when computing derivatives w.r.t. u (or theta), the different terms share the same subcomputations

→ BackPropagation Through Time avoids duplicate computation

BackPropagation Through Time (BPTT)



- Reminder of optimization objective:

$$\max_\theta \; U(\theta) = \max_\theta \; \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

- Can compute gradient estimate along current roll-out:

$$\frac{\partial U}{\partial \theta_i} = \sum_{t=0}^{H} \frac{\partial R}{\partial s}(s_t)\frac{\partial s_t}{\partial \theta_i}$$

$$\frac{\partial s_t}{\partial \theta_i} = \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1})\frac{\partial s_{t-1}}{\partial \theta_i} + \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1})\frac{\partial u_{t-1}}{\partial \theta_i}$$

$$\frac{\partial u_t}{\partial \theta_i} = \frac{\partial \pi_\theta}{\partial \theta_i}(s_t, \theta) + \frac{\partial \pi_\theta}{\partial s}(s_t, \theta)\frac{\partial s_t}{\partial \theta_i}$$

Collocation versus Shooting

- Shooting:
  - Improve sequence of controls over time, at all times u (or pi) are meaningful
  - Often poorly conditioned (effect of early u so much higher than later u)
  - Not clear how to initialize in a way that nudges towards a goal state
- Collocation
  - Might converge to a local optimum that's infeasible, and until converged often not feasible
  - x provides decoupling between time-steps, making computation stable
  - Can initialize with simple linear interpolation or guess of good trajectory
- Iterative LQR?
  - Specific example of a shooting method, with linear controllers, and second order optimization

Collision-free Path for Dubin's Car