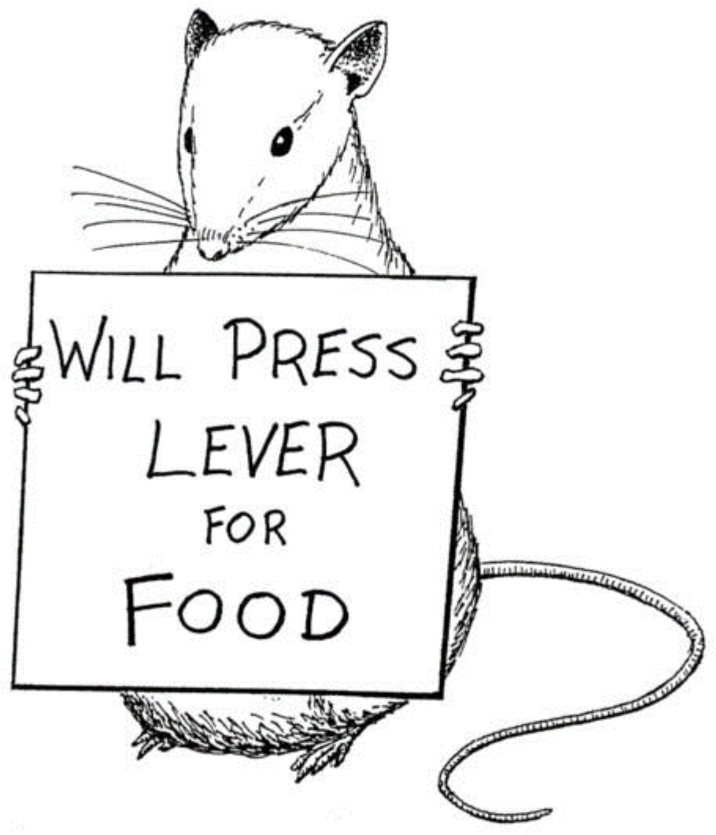
CS 277: Control and Reinforcement Learning

Winter 2021

Lecture 9: Planning

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



Logistics

assignments

- Assignment 3 to be published this week
  - Due next Friday

Today's lecture

Linear–Quadratic–Gaussian control

Planning

iLQR, DDP

Optimal control: properties

- Linear control policy: $u_t = L_t x_t \qquad L_t \in \mathbb{R}^{m \times n}$
  - Feedback gain: $L_t = -(R + B^\intercal S_{t+1} B)^{-1} B^\intercal S_{t+1} A$
- Quadratic value (cost-to-go) function $\mathcal{J}_t(x_t)^* = \frac{1}{2} x_t^\intercal S_t x_t$
  - Cost Hessian $S_t = \nabla^2_{x_t} \mathcal{J}^*_t$ is the same for all x_t
- Ricatti equation for S_t can be solved recursively backward

$$S_t = Q + A^\intercal (S_{t+1} - S_{t+1} B (R + B^\intercal S_{t+1} B)^{-1} B^\intercal S_{t+1}) A$$

  - Without knowing any actual states or controls (!) = at system design time
- Woodbury matrix identity shows $S_t = Q + A^\intercal (S^\dagger_{t+1} + B R^{-1} B^\intercal)^\dagger A \geq 0$

Kalman filter

$e_t = y_t - C\hat{x}'_t$

- Linear belief update: $\hat{x}_t = A\hat{x}_{t-1} + K_t e_t = (I - K_t C)A\hat{x}_{t-1} + K_t y_t$
- Kalman gain: $K_t = \Sigma'_t C^\intercal (C\Sigma'_t C^\intercal + \Sigma_\psi)^{-1}$
- Covariance update — Ricatti equation:

$$\Sigma'_{t+1} = A(\Sigma'_t - \Sigma'_t C^\intercal (C\Sigma'_t C^\intercal + \Sigma_\psi)^{-1} C\Sigma'_t)A^\intercal + \Sigma_\omega$$

  - Compare to prior (no observations): $\Sigma_{x_{t+1}} = A\Sigma_{x_t}A^\intercal + \Sigma_\omega$
  - Observations help, but actual observation not needed to say by how much

Control as inference

- View Bayesian inference as optimization: minimizes MSE $\mathbb{E}[\|x_t - \hat{x}_t\|^2]$

- Control and inference are deeply connected:

$$\Sigma'_{t+1} = A(\Sigma'_t - \Sigma'_t C^\intercal (C \Sigma'_t C^\intercal + \Sigma_\psi)^{-1} C \Sigma'_t) A^\intercal + \Sigma_\omega$$

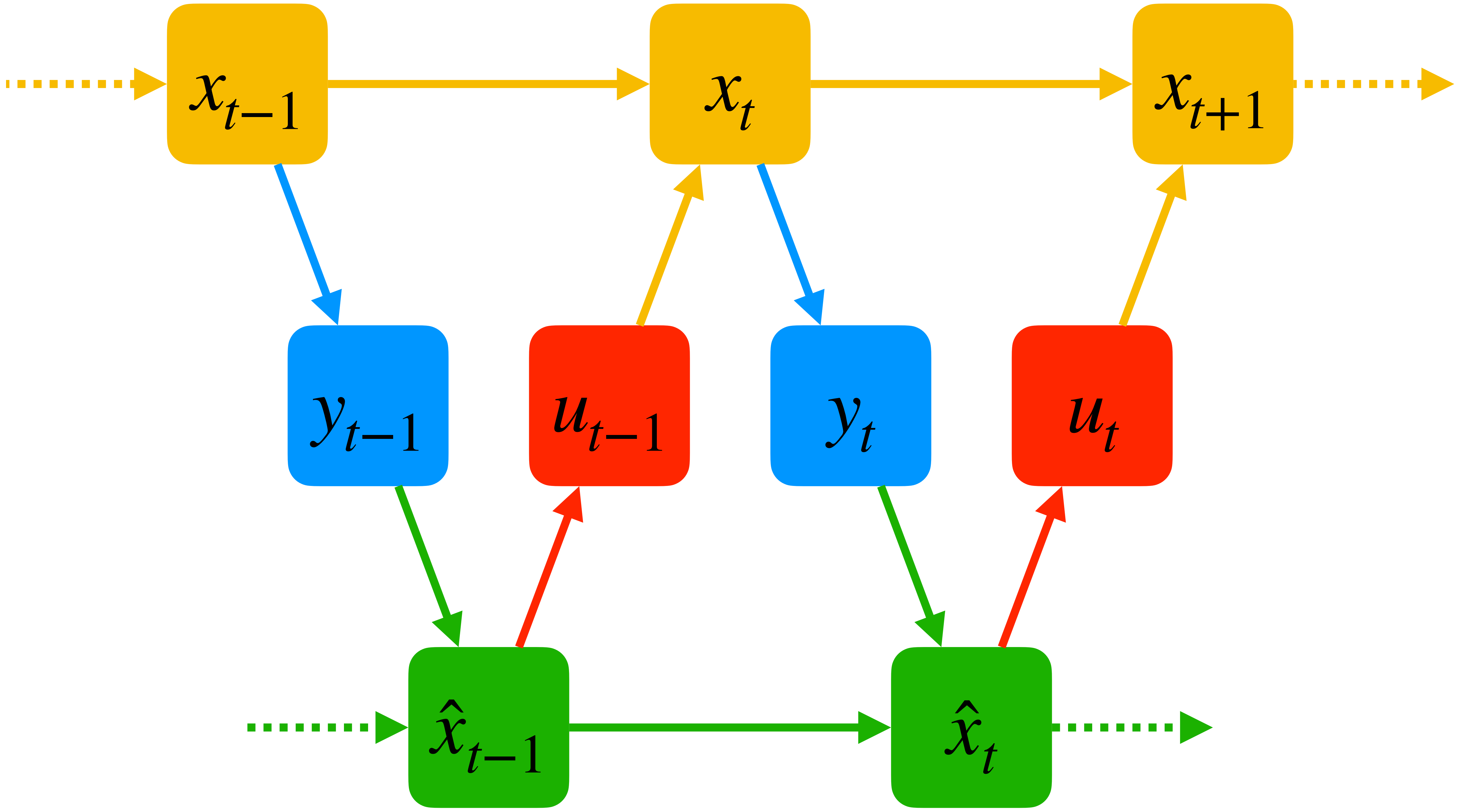
$$S_t = Q + A^\intercal (S_{t+1} - S_{t+1} B (R + B^\intercal S_{t+1} B)^{-1} B^\intercal S_{t+1}) A$$

- The shared form (Ricatti) suggests duality:

LQR	LQE
backward	forward
S_{T-t}	Σ'_t
A	$A^\intercal$
B	$C^\intercal$
Q	Σ_ω
R	Σ_ψ

- Information filter: recursion on $(\Sigma'_t)^{-1}$, presents better principled duality

Linear–Quadratic–Gaussian (LQG) control



- Putting it all together:

$$x_{t+1} = Ax_t + Bu_t + \omega_t \qquad \omega_t \sim \mathcal{N}(0, \Sigma_\omega) \qquad \Sigma_\omega \in \mathbb{R}^{n \times n}$$

$$y_t = Cx_t + \psi_t \qquad \psi_t \sim \mathcal{N}(0, \Sigma_\psi) \qquad C \in \mathbb{R}^{k \times n}, \Sigma_\psi \in \mathbb{R}^{k \times k}$$

LQE with control

- How does control affect estimation?
  - Shifts predicted next state $\hat{x}'_{t+1} = A\hat{x}_t + Bu_t$
  - Bu_t known $\Longrightarrow$ no change in covariances, Ricatti equation still holds
  - Same Kalman gain K_t

$$\hat{x}_t = A\hat{x}_{t-1} + K_t e_t = (I - K_t C)(A\hat{x}_{t-1} + Bu_{t-1}) + K_t y_t$$

- And... that's it, everything else the same

LQR with partial observability

- Bellman recursion must be expressed in terms of what u_t can depend on: $\hat{x}_t$
  - Problem: but value depends on the true state x_t
- Value recursion for full state:

$$\mathcal{J}_t(x_t, \hat{x}_t, u) = \mathbb{E}[c(x_t, u_t) + \mathcal{J}_{t+1}(x_{t+1}, \hat{x}_{t+1}, u) \mid x_t, \hat{x}_t]$$

- In terms of only $\hat{x}_t$:

works because $\hat{x}_{t+1}$ is sufficient for x_{t+1}, separating it from $\hat{x}_t$

$$\mathcal{J}_t(\hat{x}_t, u) = \mathbb{E}[\mathcal{J}_t(x_t, \hat{x}_t, u) \mid \hat{x}_t] = \mathbb{E}[c(x_t, u_t) + \mathcal{J}_{t+1}(x_{t+1}, \hat{x}_{t+1}, u) \mid \hat{x}_t] = \mathbb{E}[c(x_t, u_t) + \mathcal{J}_{t+1}(\hat{x}_{t+1}, u) \mid \hat{x}_t]$$

- Certainty equivalent control: $u_t = L_t \hat{x}_t$ with the same feedback gain L_t
- And... that's it, everything else the same

LQG separability

Given $(A, B, C, \Sigma_\omega, \Sigma_\psi, Q, R)$, solve LQG = LQR + LQE separately

- LQR:
  - Compute value Hessian recursively backwards

$$S_t = Q + A^\intercal(S_{t+1} - S_{t+1}B(R + B^\intercal S_{t+1}B)^{-1}B^\intercal S_{t+1})A$$

  - Compute feedback gain:

$$L_t = -(R + B^\intercal S_{t+1}B)^{-1}B^\intercal S_{t+1}A$$

  - Control policy: $u_t = L_t\hat{x}_t$

- LQE:
  - Compute belief covariance recursively forward

$$\Sigma'_{t+1} = A(\Sigma'_t - \Sigma'_t C^\intercal(C\Sigma'_t C^\intercal + \Sigma_\psi)^{-1}C\Sigma'_t)A^\intercal + \Sigma_\omega$$

  - Compute Kalman gain:

$$K_t = \Sigma'_t C^\intercal(C\Sigma'_t C^\intercal + \Sigma_\psi)^{-1}$$

  - Belief update: $\hat{x}_t = A\hat{x}_{t-1} + K_t e_t$

Extensive cost-to-go term

- Optimal cost-to-go: $\mathcal{J}_t^*(x_t) = \frac{1}{2}x_t^\intercal S_t x_t + \mathcal{J}_t^*(0)$

- Extensive (linear in T) term:

$$\mathcal{J}_t^*(0) = \frac{1}{2}\sum_{t'=t}^{T}(\underbrace{\operatorname{tr}(Q\Sigma_{t'})} + \underbrace{\operatorname{tr}(S_{t'+1}(\Sigma'_{t'+1} - \Sigma_{t'+1}))})$$

immediate cost of uncertainty in x_t | cost-to-go of uncertainty added by 1-step prediction

- Infinite horizon: $\mathcal{J}^* = \frac{1}{2}\operatorname{tr}(Q\Sigma)+\frac{1}{2}\operatorname{tr}(S(\Sigma' - \Sigma))$
  - S and Σ' are solutions of algebraic Ricatti equation

Today's lecture

Linear–Quadratic–Gaussian control

Planning

iLQR, DDP

Planning

- Planning is finding a good policy when we “know” the MDP
  - 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

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

Deterministic dynamics

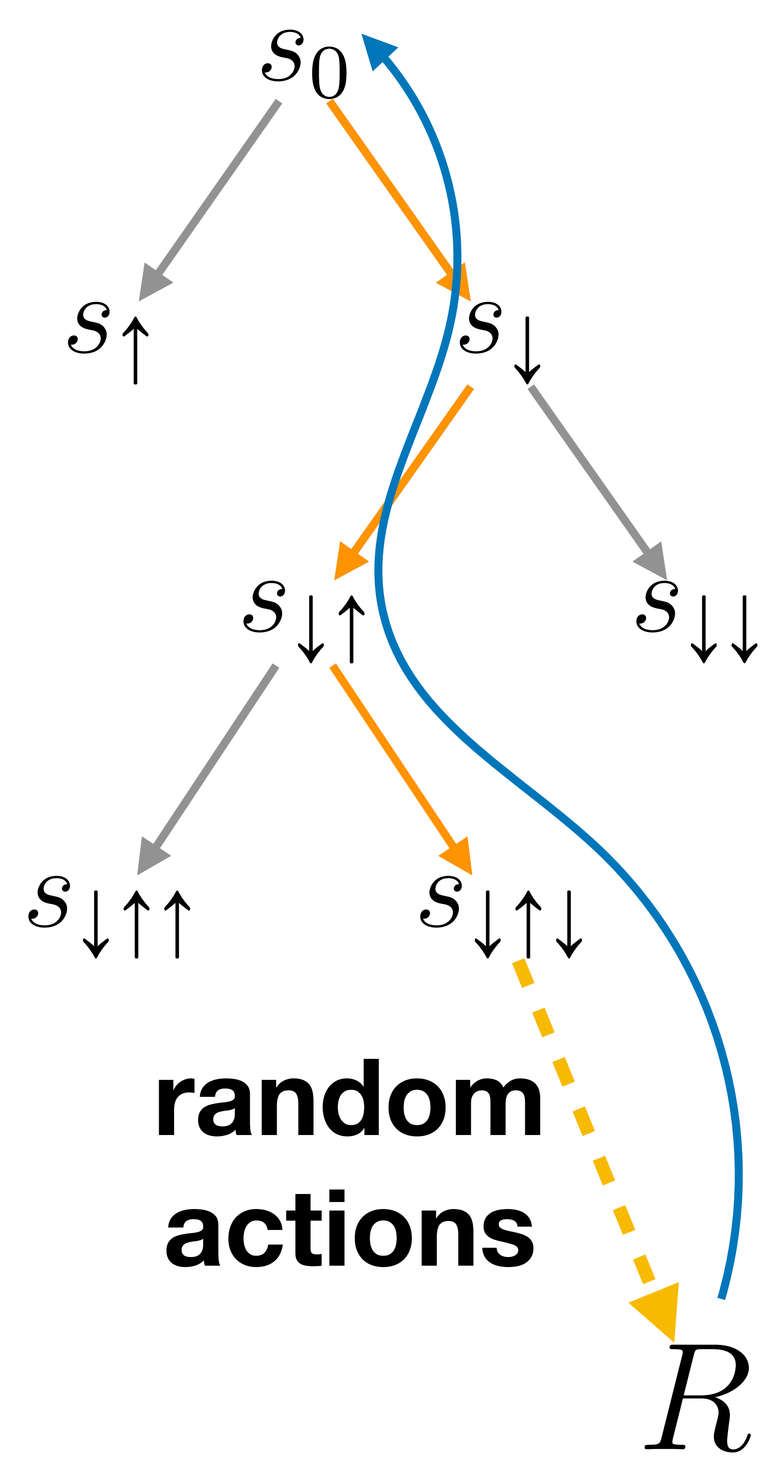
- With deterministic dynamics, policy can be just a sequence of actions

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

- Use any black-box optimizer; e.g., Cross Entropy Method (CEM):
  - Sample $\vec{a}_1, \ldots, \vec{a}_N$ from π
  - Take top $\frac{N}{c}$ "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

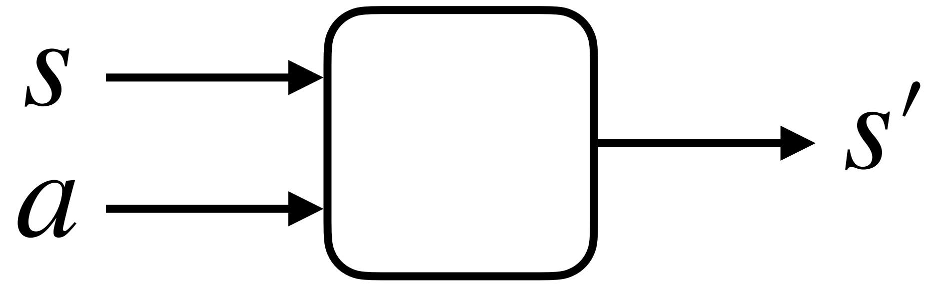


- Selecting a leaf: recursively maximize $\begin{cases} \infty & \text{if } N_{\text{visits}}(\text{child}) = 0 \\ V(\text{child}) + C\sqrt{\dfrac{\log N_{\text{visits}}(\text{self})}{N_{\text{visits}}(\text{child})}} & \text{otherwise} \end{cases}$

How to use an arbitrary-reset simulator

- Arbitrary-reset simulator allows sampling from $s' \mid s, a \sim p$ for any (s, a) we want

- Small state space — Value Iteration with tabular parametrization:



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

- Large state space — Fitted Value Iteration?

$$\mathscr{L}_\theta(s_i) = (\min_a r(s_i, a) + \gamma \mathbb{E}_{s' \mid s_i, a \sim p}[V_{\bar{\theta}}(s')] - V_\theta(s_i))^2$$

- Problem: we need $s_i \sim p_\theta(\xi)$, or we have covariate shift (train–test mismatch)
  - $\implies$ we need to start sampling from s_0, arbitrary-reset is no help
  - Simulator does enable data augmentation: perturb $s_t \sim p_\theta(\xi)$ and see how it evolves

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 for perturbation $(\delta x, \delta u)$ around a trajectory $(\hat{x}, \hat{u})$:

hiding dependence on x_t and u_t

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

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 for perturbation $(\delta x, \delta u)$ around a trajectory $(\hat{x}, \hat{u})$:

linear dependence on x_t and u_t

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

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 for perturbation $(\delta x, \delta u)$ around a trajectory $(\hat{x}, \hat{u})$:

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

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

hiding dependence on x_t and u_t

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 for perturbation $(\delta x, \delta u)$ around a trajectory $(\hat{x}, \hat{u})$:

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

linear dependence on x_t and u_t

optimal control: ∞

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 for perturbation $(\delta x, \delta u)$ around a trajectory $(\hat{x}, \hat{u})$:

NOW we can neglect these

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

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

$$+ \frac{1}{2}(\delta x_t^\intercal \nabla_x^2 \hat{c}_t \delta x_t + \delta u_t^\intercal \nabla_u^2 \hat{c}_t \delta u_t + 2\delta x_t^\intercal \nabla_{xu} \hat{c}_t \delta u_t) + O(\epsilon^3)$$

Iterative LQR (iLQR)

Algorithm 1 iLQR

linearize dynamics around current trajectory $(\hat{x}, \hat{u})$

compute $A, B \leftarrow \nabla_x \hat{f}_t, \nabla_u \hat{f}_t$

quadratic cost approximation around $(\hat{x}, \hat{u})$

compute $Q, R, N, q, r \leftarrow \nabla_x^2 \hat{c}_t, \nabla_u^2 \hat{c}_t, \nabla_{xu} \hat{c}_t, \nabla_x \hat{c}_t, \nabla_u \hat{c}_t$

$\hat{L}_t, \hat{\ell}_t \leftarrow$ LQR on $\delta x_t = x_t - \hat{x}_t$, $\delta u_t = u_t - \hat{u}_t$ ← solve with LQR

$\delta x^*, \delta u^* \leftarrow$ execute policy $\delta u_t = \hat{L}_t \delta x_t + \hat{\ell}_t$ in the simulator / environment

$\hat{x} \leftarrow \hat{x} + \delta x^*$, $\hat{u} \leftarrow \hat{u} + \delta u^*$

roll out to get new trajectory $(\hat{x}, \hat{u})$

repeat to convergence

Newton's method

- Compare iLQR 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 $\min_u \mathcal{J}(u)$
- This would be exact if we expanded the dynamics to 2nd order
  - Differential Dynamic Programming (DDP)

Recap

- 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)

Logistics

assignments

- Assignment 3 to be published this week
  - Due next Friday