Recitation 10: Quiz 3 Review

Nicholas Toldalagi, Yafei Hu, and Melrose Roderick

Quiz 3 Topics

Part 1 of Quiz 3- Regular questions like Quiz 1 & 2 (Mainly lectures 20+)

Majority of the topics are among lectures after Quiz 2

- Optimal control: LQR, iLQR
- Curiosity driven exploration
- Sim-2-Real transfer, domain adaptation
- Off-policy, Offline RL

We will also cover materials before Quiz 2.

Part 2 of Quiz 3- Paper reading and question answering

LQR (**L**inear **Q**uadratic **R**egulator)

What is Linear, and What is Quadratic?

Consider a Infinite-horizon, continuous-time case,

State transition *f* is Linear w.r.t. state **x** and action **u**

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = [\mathbf{A}, \mathbf{B}] \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} = \mathbf{F} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix}$$

Cost function *J* is Quadratic w.r.t. state **x** and action **u** (functions of t)

$$\begin{aligned} J &= \int_0^\infty \left[\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{u}^\top \mathbf{R}\mathbf{u}\right] dt \\ &= \int_0^\infty \left\{ [\mathbf{x}^\top, \mathbf{u}^\top] \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} \right\} dt \\ &= \int_0^\infty \left\{ [\mathbf{x}^\top, \mathbf{u}^\top] \mathbf{C} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} \right\} dt \end{aligned}$$

$$\mathbf{Q} = \mathbf{Q}^\top \succeq \mathbf{0},\ \mathbf{R} = \mathbf{R}^\top \succ \mathbf{0}$$

Q: symmetric positive semi-definite
R: symmetric positive definite

LQR (**L**inear **Q**uadratic **R**egulator)

Solve LQR:

A bit preliminary:
The Hamilton-Jacobi-Bellman (HJB) Equation (**not required**, just to help with LQR derivation)

$$0 = \min_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) \right],$$
$$\pi^*(\mathbf{x}) = \text{argmin}_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) \right].$$

Then we have the HJB equation for LQR:

$$0 = \min_{\mathbf{u}} \left[\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} + \frac{\partial J^*}{\partial \mathbf{x}} (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}) \right]$$

Next let’s solve optimal policy for LQR!

LQR (**L**inear **Q**uadratic **R**egulator)

Solve LQR:

Assume: $J^*(\mathbf{x}) = \mathbf{x}^T\mathbf{S}\mathbf{x}, \quad \mathbf{S} = \mathbf{S}^T \succeq 0.$ Matrix **S** is currently unknown and need to be solved!

Then $\dfrac{\partial J^*}{\partial \mathbf{x}} = 2\mathbf{x}^T\mathbf{S}.$ $\dfrac{\partial}{\partial \mathbf{u}} = 2\mathbf{u}^T\mathbf{R} + 2\mathbf{x}^T\mathbf{S}\mathbf{B} = 0.$

Optimal policy solved as:

$$\mathbf{u}^* = \pi^*(\mathbf{x}) = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S}\mathbf{x} = -\mathbf{K}\mathbf{x}.$$

Remember S is still unknown! What next?
Substitute **u*** back to HJB equation!

$$0 = \mathbf{x}^T\left[\mathbf{Q} - \mathbf{S}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S} + 2\mathbf{S}\mathbf{A}\right]\mathbf{x}.$$

Which yields the algebraic Ricatti equation (Recall HW 4) to solve **S**

$$0 = \mathbf{S}\mathbf{A} + \mathbf{A}^T\mathbf{S} - \mathbf{S}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S} + \mathbf{Q}$$

LQR (**L**inear **Q**uadratic **R**egulator)

Local stabilization of nonlinear systems

Given a goal point, or stabilizable operating point $(\mathbf{x}_0, \mathbf{u}_0)$ with $f(\mathbf{x}_0, \mathbf{u}_0) = 0$, $\mathbf{u}_0$ is typically **0**

$$\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0, \quad \bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}_0$$

with first-order Taylor expansion

$$\dot{\bar{\mathbf{x}}} \approx f(\mathbf{x}_0, \mathbf{u}_0) + \frac{\partial f(\mathbf{x}_0, \mathbf{u}_0)}{\partial \mathbf{x}}(\mathbf{x} - \mathbf{x}_0) + \frac{\partial f(\mathbf{x}_0, \mathbf{u}_0)}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}_0) = \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}}$$

Thus we get the optimal action to reach goal point with **S** solved previously

$$\bar{\mathbf{u}}^* = -\mathbf{K}\bar{\mathbf{x}}$$

iLQR (**I**terative **L**inear **Q**uadratic **R**egulators)

Why Iterative?

LQR: Use linearized transition model

iLQR: extend to non-linear transition model

Algorithm

- Get initial guess of trajectory
- Take First-order approximation of *f* and second-order (quadratic) approximation of cost function *J*

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

$$J(\mathbf{x}_t, \mathbf{u}_t) \approx J(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t} J(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}^\top \nabla^2_{\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t} J(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

- Run time-variant LQR to get action
- Get new states
- Repeat until convergence

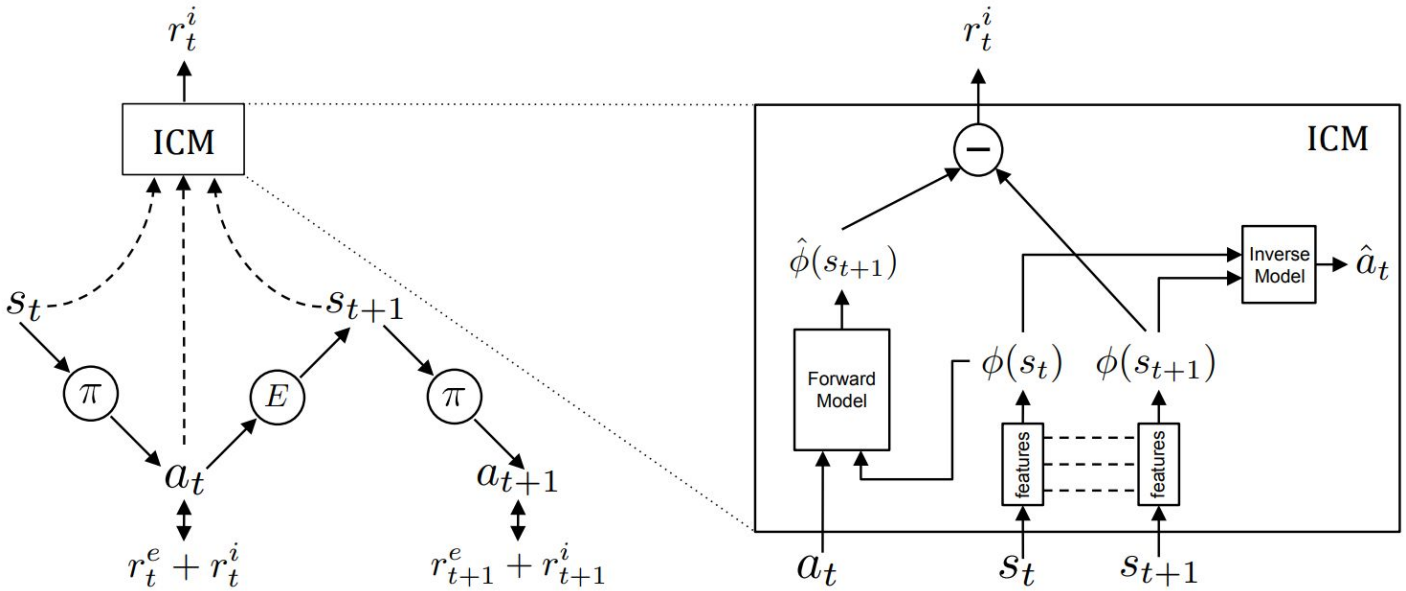
LQR and iLQR, practice questions

- ➔ Do LQR and iLQR require state transition model?
- ➔ Briefly tell the difference between LQR and iLQR?
- ➔ What is the constraints of LQR and what is the motivation(s) of iLQR?
- ➔ How would you compare LQR/iLQR with Dynamic Programming (Policy Iteration, Value Iteration) and graph search algorithms (BFS, Dijkstra, A*)?

Curiosity driven exploration

Many Scenarios: Extrinsic rewards are sparse

Possible solution: use curiosity as intrinsic rewards



Pathak et.al. Curiosity-driven Exploration by Self-supervised Prediction, in ICML 2017

Curiosity driven exploration

Total rewards is the sum of intrinsic and extrinsic rewards (0 most time steps)

$$r_t = r_t^i + r_t^e$$

Find policy with max expected rewards

$$\max_{\theta_P} \mathbb{E}_{\pi(s_t;\theta_P)}[\Sigma_t r_t]$$

Inverse model (given states, estimate action), and its loss

$$\hat{a}_t = g\Big(s_t, s_{t+1}; \theta_I\Big) \quad \min_{\theta_I} L_I(\hat{a}_t, a_t)$$

Forward model, and its loss

$$\hat{\phi}(s_{t+1}) = f\Big(\phi(s_t), a_t; \theta_F\Big) \quad L_F\Big(\phi(s_t), \hat{\phi}(s_{t+1})\Big) = \frac{1}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2$$

Final optimization objective

$$\min_{\theta_P, \theta_I, \theta_F} \Bigg[-\lambda \mathbb{E}_{\pi(s_t;\theta_P)}[\Sigma_t r_t] + (1-\beta) L_I + \beta L_F \Bigg]$$

Pathak et.al. Curiosity-driven Exploration by Self-supervised Prediction, in ICML 2017

Curiosity driven exploration, practice questions

- ➔ Why would we use curiosity as intrinsic reward?
- ➔ Can you come up with other solutions to deal with the sparse reward problem?
- ➔ Can you describe the components of ICM in *Pathak et.al. Curiosity-driven Exploration by Self-supervised Prediction*?

Sim2Real Transfer

Offline learning often requires the use of a simulator to a gather sufficient samples

Simulators often under-model, failing to capture the true complexity and diversity of an environment and are hard to hand-build with the precisely correct parameters

- How can we accurately approximate *real world* dynamics for training an agent?
- Answer: Try to simulate many types of environment conditions

Sim2Real Transfer

How to properly generalize a policy in simulation:

- **Domain Randomization:** vary aspects of your simulated training environments to help the agent/model generalize better to the variance of the real world
- **Adaptive Domain Randomization**: Evaluate the performance of your model using the current distribution and adjust the upper and lower bounds of environment parameters based on whether the policy performs well or not

Off Policy, Offline RL

On Policy: Training a policy that is also used to collect data to improve its own performance
Off Policy: Training a policy with data that is collected using a different policy

- E.g. The behavior policy might be an older version of the current policy, meaning transitions are still correlated with current policy
- Batch RL is truly off-policy
    - Generally fails due to a distribution shift

Off Policy, Offline RL

How to solve the extrapolation error problem?

- Batch constrained RL: learn a model to generate similar actions that in the batch
  - Stochasticity of the generation allows for the agent to evaluate new, yet similar states/actions (VAE)