Optimal Control for Linear Dynamical Systems and Quadratic Cost ("LQR")

Reinforcement Learning

D. Sorokin

April 28, 2025

Bellman's Course of Dimensionality

1. n-dimensional state space
2. Number of states grows exponentially in n (for fixed number of discretizations levels per coordinate)
3. In practice
   - 3.1 Discretizations is considered only computationally feasible up to 5 or 6 dimensional state spaces even when using
     - 3.1.1 Variable resolution discretizations
     - 3.1.2 Highly optimized implementations
   - 3.2 Function approximation might or might not work in practice

This Lecture

- ✓ Optimal Control for Linear Dynamical Systems and Quadratic Cost (aka LQ setting, or LQR setting)
  - Very special case: can solve continous state-space optimal control problem exactly and only requires performing linear algebra operations
  - Running time $O(H \times n^3)$

Note 1: Great reference [optional] Andereson and Moore, Linear Quadratic Methods
Note 2: Strong similarity with Kalman Filtering, which is able to compute the Bayes' filter updates exactly even through in general there are no closed from solutions and numerical solutions scale poorly with dimensionality.

Linear Quadratic Regulator (LQR)

The LQR setting assumes a linear dynamical system:

$$x_{t+1} = Ax_t + Bu_t$$

x_t: state at time t

u_t: action at time t

It assumes a quadratic cost function (i.e -reward):

$$g(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$$

Q, R - symmetric, positive definite ($Q \succ 0, R \succ 0$)

$Q \succ 0 \iff \forall z : z^T Q z > 0$

Reminder: symmetric matrix

symmetric matrix is rotation - rescaling - rotation back:

$Q = V \Lambda V^T$

V - orthogonal i.e. $V^T V = I$

Λ - diagonal

What if Q is not symmetric?

$g(x_t, u_t) = \frac{1}{2} x_t^T Q x_t + \frac{1}{2} u_t^T R u_t$

$x^T Q x = x^T \frac{Q+Q^T}{2} x + x^T \frac{Q-Q^T}{2} x = x^T \frac{Q+Q^T}{2} x$

since $\frac{x^T Q x}{2} - \frac{x^T Q^T x}{2} = \frac{x^T Q x}{2} - \frac{(x Q x^T)^T}{2} = 0$

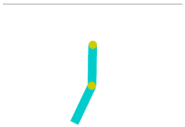
only symmetric part $\frac{Q+Q^T}{2}$ *matters*!

What if Q is not positive definite?

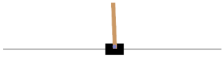
$g(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$

if $z^T Q z < 0$ you can infinitly minimize cost function g so the task does not have an finite optimal solution.

Classic Control



Acrobot



Cart Pole

Mountain Car Continuous

Mountain Car

Pendulum

Value Iteration

Back-up step for i+1 steps to go:

$$J_{i+1}(s) = \min_u g(s,u) + \sum_{s'} P(s'|s,u) J(s')$$

LQR:

$$J_{i+1}(x) = \min_u x^T Q x + u^T R u + J_i(x' = Ax + Bu)$$

LQR value iteration: J_1

Cost for zero steps-to-go: $J_0(x) = x^T P_0 x$

Cost for one step-to-go:

$$J_1(x) = \min_u \left[x^T Q x + u^T R u + (Ax + Bu)^T P_0 (Ax + Bu) \right]$$

$$\nabla_u \left[J_1(x) \right] = 2Ru + 2B^T P_0 (Ax + Bu) = 0$$

$$\rightarrow u = -(R + B^T P_0 B)^{-1} B^T P_0 A x = Kx$$

$$J_1(x) = x^T P_1 x$$

$$P_1 = Q + K_1^T R K_1 + (A + BK_1)^T P_0 (A + BK_1)$$

$$K_1 = -(R + B^T P_0 B)^{-1} B^T P_0 A$$

Cost for N steps will have the same form (!):

$$J_N(x) = x^T P_N x$$

Value iteration solution to LQR

Set $P_0 = 0$

for $i = 1, 2, 3, ...$

$$K_i = -(R + B^T P_{i-1} B)^{-1} B^T P_{i-1} A$$
$$P_i = Q + K_i^T R K_i + (A + BK_i)^T P_{i-1} (A + BK_i)$$

The optimal policy for a i-step horizon is given by: $\pi(x) = K_i x$

The cost-to-go function for a i-step horizon is given by: $J_i(x) = x^T P_i x$

- Fact: Guaranteed to converge to the infinite horizon optimal policy if and only if the dynamics (A, B) is such that there exists a policy that can drive the state to zero.
- Often most convenient to use the steady-state K for all times.

Infinite-horizon, discrete time

$$K = -(R + B^T P B)^{-1} B^T P A$$

$$P = Q + K^T R K + (A + BK)^T P (A + BK) =$$

$$Q + A^T P A + A^T P B K + K^T (R + B^T P B) K + K^T B^T P A =$$

$$P = Q + A^T P A + A^T P B K \qquad \text{(Discrete time Riccati equation)}$$

Infinite-horizon, continuous time

$$\dot{x} = Ax + Bu$$

$$J(x) = \int_0^\infty (x^T Qx + u^T Ru)dt = x_0^T Px_0 + \int_0^\infty (\frac{d}{dt}(x^T Px) + x^T Qx + u^T Ru)dt =$$

$$x_0^T Px_0 + \int_0^\infty (\dot{x}^T Px + x^T P\dot{x} + x^T Qx + u^T Ru)dt =$$

$$x_0^T Px_0 + \int_0^\infty (Ax + Bu)^T Px + x^T P(Ax + Bu) + x^T Qx + u^T Ru)dt =$$

$$x_0^T Px_0 + \int_0^\infty (x^T A^T Px + u^T B^T Px + x^T PAx + x^T PBu + x^T Qx + u^T Ru)dt =$$

$$J(x) = x_0^T Px_0 +$$

$$\int_0^\infty (x^T (A^T P + PA + Q - PBR^{-1}B^T P)x + (u + R^{-1}B^T Px)^T R(u + R^{-1}B^T Px)dt$$

Infinite-horizon, continuous time

$$J(x) = x_0^T P x_0 + \int_0^\infty (x^T (A^T P + PA + Q - PBR^{-1}B^T P)xdt + \int_0^\infty (u + R^{-1}B^T Px)^T R(u + R^{-1}B^T Px)dt$$

Choose matrix P and K such that:

$$u = Kx$$
$$K = -R^{-1}B^T P$$
$$A^T P + PA + Q - PBR^{-1}B^T P = 0 \qquad \text{(Riccati equation)}$$
$$J(x) = x_0^T P x_0$$

LQR assumptions revisited

$$x_{t+1} = Ax_t + Bu_t$$
$$g(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$$

= for keeping a linear system at the all-zeros state while preferring to keep the control input small.

- Extensions make it more generally applicable:
  - Affine systems
  - Systems with stochasticity
  - Regulation around non-zero fixed point for non-linear systems
  - Penalization for change in control inputs
  - Linear time varying (LTV) systems
  - Trajectory following for non-linear systems

LQR Ext0: Affine systems

$$x_{t+1} = Ax_t + Bu_t + c$$
$$g(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$$

- Optimal control policy remains linear, optimal cost-to-go function remains quadratic
- Two avenues to do derivation:
  - 1. Re-derive the update, which is very similar to what we did for standard setting
  - 2. Re-define the state as: $z_t = [x_t; 1]$, then we have:

$$z_{t+1} = \begin{bmatrix} x_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} A & c \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_i \\ 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_t = A' z_t + B' u_t$$

LQR Ext1: stochastic system

$$x_{t+1} = Ax_t + Bu_t + w_t$$
$$g(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$$
w_t, t=0,1,... are zero mean and independent

- Exercise: work through similar derivation as we did for the deterministic case, but which will now have expectations.
- Result:
  - Same optimal control policy
  - Cost-to-go function is almost identical: has one additional term which depends on the variance in the noise (and which cannot be influenced by the choice of control inputs)

LQR Ext2: non-linear systems

Nonlinear system: $x_{t+1} = f(x_t, u_t)$

We can keep the system at the state x^* iff

$$\exists u^* s.t. \quad x^* = f(x^*, u^*)$$

Linearizing the dynamics around x^* gives:

$$x_{t+1} \approx f(x^*, u^*) + \frac{\partial f}{\partial x}(x^*, u^*)(x_t - x^*) + \frac{\partial f}{\partial u}(x^*, u^*)(u_t - u^*)$$

Equivalently:

$$x_{t+1} - x^* \approx A(x_t - x^*) + B(u_t - u^*)$$

Let $z_t = x_t - x^*$, let $v_t = u_t - u^*$, then:

$$z_{t+1} = Az_t + Bv_t, \text{cost} = z_t^T Q z_t + v_t^T R v_t$$

$$v_t = Kz_1 \Rightarrow u_t - u^* = K(x_t - x^*) \Rightarrow u_t = u^* + K(x_t - x^*)$$

LQR Ext3: Penalize for Change in Control Inputs

- Standard LQR:

$$x_{t+1} = Ax_t + Bu_t$$
$$g(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$$

- When run in this format on real systems: often high frequency control inputs get generated. Typically highly undesirable and results in poor control performance.
- Why?
- Solution: frequency shaping of the cost function. Can be done by augmenting the system with a filter and then the filter output can be used in the quadratic cost function. (See, e.g., Anderson and Moore.)
- Simple special case which works well in practice: penalize for change in control inputs. – How ??

LQR Ext3: Penalize for Change in Control Inputs

- Standard LQR:

$$x_{t+1} = Ax_t + Bu_t$$
$$g(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$$

- How to incorporate the change in controls into the cost/reward function?
  - Soln. method A: explicitly incorporate into the state by augmenting the state with the past control input vector, and the difference between the last two control input vectors.
  - Soln. method B: change of control input variables.

LQR Ext3: Penalize for Change inf Control Inputs

- Standard LQR:

$$x_{t+1} = Ax_t + Bu_t$$
$$g(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$$

- Introducing change in controls Δu:

$$\begin{bmatrix} x_{t+1} \\ u_t \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x_t \\ u_{t-1} \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \Delta u_t$$

$$\text{cost} = x'^T Q' x' + \Delta u^T R' \Delta u$$

$$Q' = \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix}, \qquad R' = \text{penalty for change in controls}$$

LQR Ext4: Linear TIme Varying (LTV) Systems

$$x_{t+1} = A_t x_t + B_t u_t$$
$$g(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$$

LQR Ext4: Linear TIme Varying (LTV) Systems

Set $P_0 = 0$

for $i = 1, 2, 3, ...$

$$K_i = -(R_{H-i} + B_{H-i}^T P_{i-1} B_{H-i})^{-1} B_{H-i}^T P_{i-1} A_{H-i}$$

$$P_i = Q_{H-i} + K_i^T R_{H-i} K_i + (A_{H-i} + B_{H-i} K_i)^T P_{i-1} (A_{H-i} + B_{H-i} K_i)$$

The optimal policy for a i-step horizon is given by: $\pi(x) = K_i x$

The cost-to-go function for a i-step horizon is given by: $J_i(x) = x^T P_i x$

LQR Ext5: Trajectory Following for Non-Linear Systems

- A state sequence $x_0^*, x_1^*, ..., x_H^*$ is a feasible target trajectory if and only if

$$\exists u_0^*, u_1^*, ..., u_{H-1}^* : \forall t \in 0, 1, ..., H-1 : x_{t+1}^* = f(x_t^*, u_t*)$$

- Problem statement:

$$\min_{u0, u_1, ..., u_{H-1}} \sum_{t=0}^{H-1} (x_t - x_t^*)^T Q (x_t - x_t^*) + (u_t - u_t^*) R (u_t - u_t^*)$$

$$\text{s.t. } x_{t+1} = f(x_t, u_t)$$

- Transform into linear time varying case (LTV):

$$x_{t+1} \approx f(x_t^*, u_t^*) + \frac{\partial f}{\partial x}(x_t^*, u_t^*)(x_t - x_t^*) + \frac{\partial f}{\partial u}(x_t^*, u_t^*)(u_t - u_t^*)$$

$$x_{t+1} - x_{t+1}^* \approx A_t (x_t - x_t^*) + B_t (u_t - u_t^*)$$

LQR Ext5: Trajectory Following for Non-Linear Systems

- Transform into linear time varying case (LTV):

$$x_{t+1} \approx f(x_t^*, u_t^*) + \frac{\partial f}{\partial x}(x_t^*, u_t^*)(x_t - x_t^*) + \frac{\partial f}{\partial u}(x_t^*, u_t^*)(u_t - u_t^*)$$
$$x_{t+1} - x_{t+1}^* \approx A_t(x_t - x_t^*) + B_t(u_t - u_t^*)$$

- Now we can run the standard LQR back-up iterations.
- Resulting policy at i time-steps from the end:

$$u_{H-i} - u_{H-i}^* = K_i(x_{H-i} - x_{H-i}^*)$$

- The target trajectory need not be feasible to apply this technique, however, if it is infeasible then there will be an offset term in the dynamics:

$$x_{t+1} - x_{t+1}^* = f(x_t, u_t) - x_{t+1}^* + A_t(x_t - x_t^*) + B_t(u_t - u_t^*)$$

Most General Case

- How abount this general optimal control problem?

$$\min_{u_0,\ldots,u_H} \sum_{t=0}^{H} g(x_t, u_t)$$

Iteratively Apply LQR

Initialize the algorithm by picking either (a) A control policy $\pi^{(0)}$ or (b) A sequence of states $x_0^{(0)}, x_1^{(0)}, ..., x_H^{(0)}$ and control inputs $u_0^{(0)}, u_1^{(0)}, ..., u_H^{(0)}$. With initialization (a), start in Step (1). With initialization (b), start in Step (2). Iterate the following:

1. Execute the current policy $\pi^{(i)}$ and record the resulting state-input trajectory $x_0^{(0)}, u_0^{(0)}, x_1^{(0)}, u_1^{(0)}, ..., x_H^{(0)}, u_H^{(0)}$.
2. Compute the LQ approximation of the optimal control problem around the obtained state-input trajectory by computing a first-order Taylor expansion of the dynamics model, and a second-order Taylor expansion of the cost function.
3. Use the LQR back-ups to solve for the optimal control policy $\pi^{(i+1)}$ for the LQ approximation obtained in Step (2).
4. Set $i = i + 1$ and go to Step (1).