LQR: The Analytic MDP

Linear Quadratic Regulator

In the previous chapter we defined MDPs and investigated how to recursively compute the value function at any state with Value Iteration. While the examples in the previous chapter involved discrete state and action spaces, one of the most important applications of the basic algorithms and theory of MDPs is problems where both states and actions are continuous. Perhaps the simplest such problem is the linear quadratic regulator (LQR) problem.

The LQR is one of the most effective and widely used methods in control systems design. The basic problem is to identify a mapping from states to controls that minimizes the quadratic cost of a linear (possibly time invariant) system. A quadratic cost has the form

$$c(x, u) = x^T Q x + u^T R u, \tag{1}$$

where $x \in \mathbb{R}^n$ is the state of the system, and $u \in \mathbb{R}^k$ is the control. To avoid infinite control, Q should be positive semi-definite and R should be positive definite.

Continuous Control of a Discrete-Time System

An example of a continuous time-invariant system with quadratic cost is the problem of balancing a simple inverted pendulum. The pendulum is illustrated in Figure 8. The simple pendulum consists of a bob, modeled as a point mass, and attached to a massless rigid rod. Let the mass of the bob be m, the length of the rod be l, and gravity be g. The angle between the pendulum and the y-axis θ is controlled by the torque τ exerted at the origin. The dynamics of this system is given by

$$\begin{aligned} ml^2\ddot{\theta} &= mgl\sin\theta + \tau \\ \implies \ddot{\theta} &= \frac{g}{l}\sin\theta + \frac{1}{ml^2}\tau \\ &\approx \frac{g}{l}\theta + \frac{1}{ml^2}\tau \end{aligned} \tag{2}$$

To find the control policy of the system, we first linearize it about

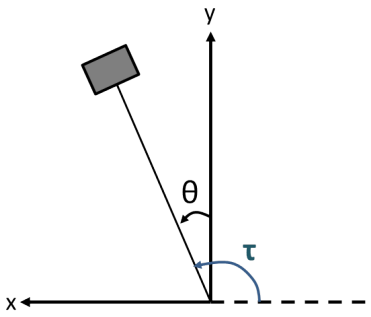

Figure 8: An inverted pendulum.

the up-right configuration. Let $c = g/l$, and assume $ml^2 = 1$. The state space equations become

$$\begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}_{t+1} = \begin{bmatrix} 1 + \frac{1}{2}\Delta t^2 \cdot c & \Delta t \\ c \cdot \Delta t & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}_t + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} \tau \tag{3}$$

The optimal control policy can be found by formulating an MDP. For the linearized simple pendulum,

- state: $x_t = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}_t$,
- action: $u_t = \tau$,
- cost: $c(x, u) = x^T Q x + u^T R u$,
- dynamics: $x_{t+1} = Ax_t + Bu_t$,
  where $A = \begin{bmatrix} 1 + \frac{1}{2}\Delta t^2 \cdot c & \Delta t \\ c \cdot \Delta t & 1 \end{bmatrix}$ and $B = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix}$.

We already know how to solve this problem: Value Iteration! Let's look at this more closely.

Value Iteration for Linear Quadratic MDPs

Let the value function of the MDP for a finite-horizon problem with horizon T be be $J(x_t, t)$. Recall the **Bellman Equation**:

$$\begin{aligned} J(x_t, t) &= \min_{u_t} c(x_t, u_t) + \gamma \mathbb{E}[J(x_{t+1}, t+1)] \\ &= \min_{u_t} ({x_t}^T Q x_t + {u_t}^T R u_t) + \gamma \mathbb{E}[J(x_{t+1}, t+1)]. \end{aligned} \tag{4}$$

Let's consider the recursive formulation for solving this problem.
Time T − 1:

At the last time step $t = T - 1$, the solution to Equation 4 is $u_{T-1} = \mathbf{0}$. This is due to the fact that when the system is in the

goal state, *any* action will increase the cost. Also, since we cannot alter the cost influenced by the state or the value of the next time step, minimizing (4) is essentially minimizing $u_{T-1}^T R u_{T-1}$. By definition, R is a positive definite matrix, and therefore setting $u_{T-1} = 0$ can result in minimum cost at $t = T - 1$.

Now let calculate the value function $J(x_{T-1}, T-1)$. Since ${u_{T-1}}^T R u_{T-1} = 0$ and $J(x_T, T) = 0$ as the goal state is reached,

$$J(x_{T-1}, T-1) = {x_{T-1}}^T Q x_{T-1} := {x_{T-1}}^T V_{T-1} x_{T-1}, \tag{5}$$

where V_{T-1} is the *value matrix*.

In summary, at the last time step, we have a zero control and a value that is quadratic in the state.

Time T − 2:

The value function at $t = T - 2$ is

$$J(x_{T-2}, T-2) = \min_{u_{T-2}} c(x_{T-2}, u_{T-2}) + J(x_{T-1}, T-1) \tag{6}$$

$$= \min_{u_{T-2}} \left({x_{T-2}}^T Q x_{T-2} + {u_{T-2}}^T R u_{T-2} + {x_{T-1}}^T V_{T-1} x_{T-1} \right). \tag{7}$$

Let $x = x_{T-2}$ and $u = u_{T-2}$. From the dynamics of the system, $x_{T-1} = Ax + Bu$.

$$J(x, T-2) = \min_u \left\{ x^T Q x + u^T R u + (Ax)^T V_{T-1} (Ax) + 2(Ax)^T V_{T-1} (Bu) + (Bu)^T V_{T-1} (Bu) \right\} \tag{8}$$

Taking the partial derivative of the function to be minimized with respect to u and setting it to o yields

$$\begin{aligned} 2Ru + 2B^T V_{T-1} Ax + 2B^T V_{T-1} Bu &= 0 \\ (R + B^T V_{T-1} B) u &= -B^T V_{T-1} Ax \\ u &= -(R + B^T V_{T-1} B)^{-1} B^T V_{T-1} Ax \end{aligned} \tag{9}$$

The solution to u always exists because the inverse of $R + B^T V_{T-1} B$ exists since R is positive definite and $B^T V_{T-1} B$ is at least positive semidefinite. Let $K_{T-2} = -(R + B^T V_{T-1} B)^{-1} B^T V_{T-1} A$,

$$u_{T-2} = K_{T-2} x_{T-2}. \tag{10}$$

The control u_{T-2} is a linear function of state x_{T-2} with control matrix K_{T-2}. The value function at $t = T - 2$ can be found as

$$\begin{aligned} J(x_{T-2}, T-2) &= {x_{T-2}}^T Q x_{T-2} + {x_{T-2}}^T K_{T-2}^T R K_{T-2} x_{T-2} + {x_{T-2}}^T (A + BK_{T-2})^T V_{T-1} (A + BK_{T-2}) x_{T-2} \\ &= {x_{T-2}}^T (Q + K_{T-2}^T R K_{T-2} + (A + BK_{T-2})^T V_{T-1} (A + BK_{T-2})) x_{T-2} \\ &= {x_{T-2}}^T V_{T-2} x_{T-2}. \end{aligned} \tag{11}$$

Observe that in this time step, the value is *also* quadratic in state. Therefore, we can derive similar results of linear control and quadratic value for every time step prior to $t = T - 2$:

$$\begin{aligned} K_t &= -(R + B^T V_{t+1} B)^{-1} B^T V_{t+1} A \\ V_t &= \underbrace{Q}_{\text{current cost}} + \underbrace{K_t^T R K_t}_{\text{cost of action at t}} + \underbrace{(A + BK_t)^T V_{t+1} (A + BK_t)}_{\text{cost to go}} \\ J(x_t, t) &= {x_t}^T V_t x_t \end{aligned} \tag{12}$$

Algorithm 6 summarizes value iteration for LQRs:

Algorithm `OptimalValue`(A, B, Q, R, t, T)

```
if t = T − 1 then
    return Q
end
else
    V_{t+1} = OptimalValue(A, B, Q, R, t + 1, T)
     K_t = −(B^T V_{t+1} B + R)^{−1} B^T V_{t+1} A
    return V_t = Q + K_t^T R K_t + (A + BK_t)^T V_{t+1} (A + BK_t)
end
```

Algorithm 6: LQR value Iteration

The complexity of the above algorithm is a function of the horizon T, the dimensionality of the state space n, and the dimensionality of the action space k: $O(T(n^3 + k^3))$.

Convergence of Value Iteration

K_t and V_t converge if the system is *stabilizable*, and the solution to them is the Discrete Algebraic Ricatti Equation (DARE):

$$\begin{aligned} V &= Q + K^T R K + (A + BK)^T V (A + BK) \\ K &= -(R + B^T V B)^{-1} B^T V A \end{aligned} \tag{13}$$

We can view V as a combination of current state, control and future cost. If the system is not stabilizable, for example, a system of two motors controlling two inverted pendulums with one of the motors broken, K_t and V_t no longer converge. However, the value iteration will still return the policy that can get the system work as well as possible by stabilizing the good motor.

LQR Tracking

The method described in Algorithm 6 will not work for a pendulum swing up problem, since the system dynamics at $\theta = 0^\circ$ (unstable) and $\theta = 180^\circ$ (stable) are qualitatively different.

Given a expert trajectory (x_t, u_t) from $\theta = 180^\circ$ to $\theta = 0^\circ$ (see Fig 9), one might imagine that you could simply replay it. But this wont work in practice due to modeling error (the same sequence of controls is unlikely to produce exactly the same behavior when played twice on a real system). However, a reference trajectory can still be useful. One way to use expert trajectory in presence of uncertainties, is to use LQR *tracking*, which we will describe below.

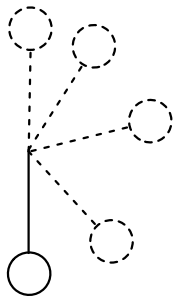

Figure 9: Solving inverted pendulum swing up using LQR tracking.

To describe how tracking works, we first introduce LQRs for *Linear Time Varying* dynamical systems, and *Affine Quadratic Regulation*.

LQR for Linear Time-Varying Dynamical Systems

Thus far, we have assumed that we were modeling a linear time invariant system. As we will see, we might be interested in systems that are linear, but time varying

$$x_{t+1} = A_t x_t + B_t u_t \tag{14}$$

$$c(x_t, u_t) = {x_t}^T Q_t x_t + u_t^T R_t u + t \tag{15}$$

In this case, the LQR equations are simply updated to

$$K_t = -(B_t^T V_{t+1} B_t + R_t)^{-1} B_t^T V_{t+1} A_t \tag{16}$$

$$V_t = Q_t + K_t^T R_t K_t + (A_t + B_t K_t)^T V_{t+1} (A_t + B_t K_t) \tag{17}$$

Affine Quadratic Regulation

Let's now consider a generic affine system with time varying dynamics A_t and B_t and a state *offset* x_t^{off}:

$$x_{t+1} = A_t x_t + B_t u_t + x_t^{off}. \tag{18}$$

Affine problems can be converted to linear problems by using the homogeneous coordinates:

$$\tilde{x} = \begin{pmatrix} x \\ 1 \end{pmatrix} \tag{19}$$

$$\tilde{x}_{t+1} = \begin{pmatrix} A_t & x_t^{off} \\ 0 & 1 \end{pmatrix} \tilde{x}_t + \begin{pmatrix} B_t \\ 0 \end{pmatrix} u_t \tag{20}$$

Given the new state and dynamics, and a redefinition of the cost as $c(\tilde{x}_t, u_t) = \tilde{x}_t^T \tilde{Q}_t \tilde{x}_t + u_t^T R_t u_t$, the Affine Quadratic Regulation problem can be solved in exactly the same way as the LQR problem.

Tracking

There are two possible formulations for a tracking cost function:

$$C(x_t, u_t) = (x_t - x_t^*)^T Q (x_t - x_t^*) + (u_t - u_t^*)^T R (u_t - u_t^*) \tag{21}$$

$$C(x_t, u_t) = (x_t - x_t^*)^T Q (x_t - x_t^*) + (u_t)^T R (u_t) \tag{22}$$

where x_t^* and u_t^* are the nominal trajectory and nominal control input obtained from the expert. Q penalizes the deviation from the nominal trajectory and R penalizes either the deviation from the nominal controls or is just a penalty on the control (e.g. lots of actuation is bad).

Expanding the term corresponding to state error in the cost function:

$$\begin{aligned}(x_t - x_t^*)^T Q (x_t - x_t^*) &= x_t^T Q x_t + \underbrace{x_t^{*T} Q x_t^*}_{\text{constant at time } t} - \underbrace{2 x_t^{*T} Q}_{\text{constant at time } t} x_t \\ &= x_t^T Q x_t + c_t - 2 q_t x_t,\end{aligned}$$

where $c_t \equiv x_t^{*T} Q x_t^*$ and $q_t \equiv x_t^{*T} Q$. Next, we choose a $\tilde{Q}_t$ defined as:

$$\tilde{Q}_t = \begin{pmatrix} Q & -q_t \\ -q_t^T & c_t \end{pmatrix},$$

such that the state error term of the cost function can be formulated as $\tilde{x}_t^T \tilde{Q}_t \tilde{x}_t$. Note that c is a constant, which only shifts the cost function in some uninteresting way. Rather than defining c in terms of Q and the nominal trajectory, typically a large value of c is selected in an attempt to keep $\tilde{Q}$ positive semidefinite.

Expanding the control error term of cost function described in Eq. 21:

$$\begin{aligned}(u_t - u_t^*)^T R (u_t - u_t^*) &= u_t^T R u_t - 2 u_t^{*T} R u_t + u_t^{*T} R u_t^* \\ &= u_t^T R u_t + 2 r^T u_t, \quad \text{where } r^T = -u_t^{*T} R.\end{aligned}$$

For cost functions of this form (Quadratic + linear), let $\widetilde{u}_t = (u_t - u_t^*)$, so that the corresponding term of the cost function can be modified as $\widetilde{u}_t^T R \widetilde{u}_t$. In order to use $\widetilde{u}_t$ instead of u_t in the cost function defined as in Eq. 21, the dynamics needs to be modified as follows:

$$\widetilde{x}_{t+1} = \begin{pmatrix} A_t & x_t^{off} + B_t u_t^* \\ 0 & 1 \end{pmatrix} \widetilde{x}_t + \begin{pmatrix} B \\ 0 \end{pmatrix} \widetilde{u}_t. \tag{23}$$

The new cost function is:

$$C(\widetilde{x}_t, \widetilde{u}_t) = \widetilde{x}_t^T \widetilde{Q}_t \widetilde{x}_t + \widetilde{u}_t^T R \widetilde{u}_t. \tag{24}$$

Solving the LQR for the system using the above cost function

$$\widetilde{u}_t = -K_t \begin{pmatrix} x_t \\ 1 \end{pmatrix}$$

Subsequently u_t is obtained as $u_t = \widetilde{u}_t + u_t^*$.

Iterative LQR (iLQR) (for non linear systems)

1. Propose some trajectory $\{x_t, u_t\}_{t=0,\cdots,T-1}$
2. Linearize the dynamics, f about trajectory:

$$\left.\frac{\partial f}{\partial x}\right|_{x_t} = A, \quad \left.\frac{\partial f}{\partial u}\right|_{u_t} = B$$

   Linearization can be obtained by three methods:

   (a) Analytical
   (b) Numerical
   (c) Statistical: Collect data, fit linear model

3. Quadraticize cost function, $C(x_t, u_t)$ (compute second order Taylor series expansion for C around x_t and u_t).
4. Now that we have $\{A, B, Q, R, q, r\}_t$, solve the LQR control problem (Affine version) and obtain the sequence of controls $\{u_t\}_{t=0,\cdots,T-1}$.
5. Forward simulate the full non-linear model $f(x, u)$ using the computed controls $\{u_t\}_{t=0,\cdots,T-1}$ to obtain new states $\{x_t\}_{t=0,\cdots,T-1}$.
6. Using the newly obtained $\{x_t, u_t\}_{t=0,\cdots,T-1}$ repeat steps from 2.

This looks a lot like Newton's method. We first approximate the value function to a quadratic and then minimize it and iterate till convergence.

Issues

- Q and R can be indefinite when the cost function is not convex. Hack:
  - Projection: $Q = U \underbrace{\Sigma}_{\text{set negative Eigenvalues to 0}} U^T$
  - Regularize: Boost up the diagonal values till Q becomes positive definite: $Q = Q + \lambda I$
- Trust region: Sometimes the approximation of the cost function is poor and in such cases its a good idea to restrict the step size while executing the control. This can be done in the following ways:
  - interpolate between the control at current and previous steps
  - modify cost to $\widetilde{C} = \alpha \times C + (1 - \alpha) \times (\text{penalty for deviation from last trajectory})$
- Trivia :)
  LQR got some bad name in 1970s due to failure in systems at high frequencies by invoking resonant frequency and breaking. So to prevent high frequency control from being generated we could do:
  - Penalize change in control from previous control. This is to ensure that the control is smooth. Higher order of smoothness can be obtained by passing the control signal through a filter and then using the output of that as control input for the system.
  - Or include delay in the system dynamics as below:

$$\begin{pmatrix} x_t \\ u_{t-1} \end{pmatrix} = \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_{t-1} \\ u_{t-2} \end{pmatrix} + \begin{pmatrix} 0 \\ u_{t-1} \end{pmatrix}. \tag{25}$$