Lecture 6: The Linear Quadratic Regulator

Last time, we discussed finite horizon MDPs with continuous state & action spaces. We also introduced linear dynamics (transitions).

Today we consider a continuous MDP problem with linear dynamics and quadratic costs: the Linear Quadratic Regulator.

1) LQR \[ LQR(A,B,Q,R) \]

For this continuous MDP,

\[ S = \mathbb{R}^{ns}, \quad A = \mathbb{R}^{na}, \quad f(s_t,a_t,w_t) = A s_t + B a_t + w_t \]

\[ w_t \sim N(0,\sigma^2I) \]

\[ c(s,u) = s^T Q s + u^T R u \]

quadratic costs, \( Q, R \succ 0 \)

Horizon \( H \) & initial distribution \( M_0 \).

Putting this all together, Optimal Control Problem:

\[
\min_{\tau} \quad \mathbb{E} \left[ s_H^T Q s_H + \sum_{t=0}^{H-1} s_t^T Q s_t + a_t^T R a_t \right]
\]

\[ s_{t+1} = A s_t + B a_t + w_t, \quad x_0 \sim M_0 \]

\[ a_t = \Pi_t(s_t), \quad w_t \sim N(0,\sigma^2I) \]

Example: 1d robot from last lecture

\[ s_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix} \] position  \[ s_{t+1} = \begin{bmatrix} 1 & 1 \end{bmatrix} s_t + \begin{bmatrix} 0 \\ m \end{bmatrix} a_t \]

\[ \text{goal: move to goal position } p=0 \text{ and be still } v=0 \text{ without using too much force} \]
can use quadratic cost:
\[ C(s, a) = \gamma_p p^2 + \gamma_v v^2 + \gamma_a a^2 \]
\[ = s^T \begin{bmatrix} \gamma_p & 0 \\ 0 & \gamma_v \end{bmatrix} s + \gamma_a a^2 \]

Depending on the relative weighting of \( \gamma_p, \gamma_v, \) and \( \gamma_a, \) optimal policy will be more or less aggressive.

Value and Q functions "cost to go"

\[ V_t^\pi(s) = \mathbb{E} \left[ s_{t+1}^T Q_{t+1} s_{t+1} + \sum_{k=t}^{t+1} s_k^T Q_k s_k + \alpha_k^T R_k s_k \right] \]

\[ Q_t^\pi(s, a) = \mathbb{E} \left[ s_{t+1}^T Q_{t+1} s_{t+1} + \sum_{k=t}^{t+1} s_k^T Q_k s_k + \alpha_k^T R_k s_k \right] \]

\[ V_t^\pi(s) = Q_t^\pi(s, \pi(s)) \]

Notice that due to terminal cost, \( V_t^\pi(s) \) is nonzero.
2) Optimal LQR Policy

We can derive the optimal LQR Policy via Dynamic Programming.

\[ \Pi^* = (\Pi_{H-1}^*, \Pi_{H-2}^*, \ldots, \Pi_1^*) \]

Start: \( V_H^*(s) = c_H(s) \) (due to terminal cost, initialize \( V_H^*(s) \neq 0 \) unlike before, but the algorithm is the same).

\[ \text{for } t = H-1, H-2, \ldots, 0: \]
\[ Q_t^*(s, a) = c_t(s, a) + \mathbb{E} \left[ V_{t+1}^*(s') \right] \]
\[ \Pi_t^* = \arg\min_a Q_t^*(s, a) \]

**Theorem (LQR optimal value fn. & Policy):**

For LQR(\( A, B, Q, R \)), the optimal value function is quadratic:
\[ V_t^*(s) = s^T P_t s + p_t \]
and the optimal policy is linear
\[ \Pi_t^*(s) = -K_t^* s \]
where \( (P_t, p_t, K_t^*) \) can be computed exactly from \( (A, B, Q, R) \).
Proof: We prove by induction, using DP.

Claim 1: (Base case) $V^*_H(s) = S^TP_HS + P_H$ is quadratic \forall S

Claim 2: (induction) Assume $V^*_{t+1}(s) = S^TP_{t+1}S + P_{t+1}$ \forall S. Then

1) $Q^*_t(s,a)$ is quadratic in $s,a$

2) $\Pi^*_t(a)^\text{argmin}_{a}Q_t^*(s,a)$ is linear in $s$

Therefore, $V^*_t(s) = S^TP_tS + P_{t+1}$ is quadratic.

Then by induction, $V$ is quadratic & $\Pi$ linear.

**Proof of Claim 1:**

$V^*_H(s) = C_H(s) = S^TQ_S \cdot S$. \[ (symmetric) \]

**Proof of Claim 2:**

**Part 1:** $Q^*_t(s,a) = S^TQ_S + a^T R_a + \mathbb{E}_{s \sim P_{t}(a)} [V^*_{t+1}(s')]$

\[ \mathbb{E}_{s \sim P_{t}(a)} [V^*_{t+1}(s')] = \mathbb{E}_{w \sim N(0,1)} [V^*_{t+1}(As + Ba + w)] \]

$V^*_{t+1}(As + Ba + w) = (As)^T P_{t+1} (As) + (As)^T P_{t+1} Ba + (As)^T P_{t+1} W$

$+ (Ba)^T P_{t+1} As + (Ba)^T P_{t+1} Ba + (Ba)^T P_{t+1} W$

$+ W^T P_{t+1} As + W^T P_{t+1} Ba + W^T P_{t+1} W + P_{t+1}$

Once we take expectation, many terms $= 0$ because $\mathbb{E}W = 0$.

\[ \mathbb{E}_{s \sim P_{t}(a)} [V^*_{t+1}(s')] = S^T A^T P_{t+1} As + S^T A^T P_{t+1} Ba + a^T R_a P_{t+1} Ba \]

$+ \mathbb{E}W[W^T P_{t+1} W] + P_{t+1}$
to simplify the remaining expectation, recall
\[ W^T P W = \text{Tr}(W^T P W) = \text{Tr}(P W W^T). \]
\[ \mathbb{E} [W^T P W] = \mathbb{E} [\text{Tr}(P W W^T)] = \text{Tr}(P \mathbb{E}[WW^T]) \]
\[ = \sigma^2 \text{Tr}(P) \]

Finally,
\[ Q_t^*(s, a) = s^T (R + A^T P_{t+1} A) s + a^T (R + B^T P_{t+1} B) a \]
\[ + 2 s^T A^T P_{t+1} B a + \sigma^2 \text{Tr}(P) + P_{t+1} \quad \checkmark \]

Done with part 1 because this is a quadratic function.

**Part 2**

\[ \pi_t^* (s) = \arg \min_a Q_t^* (s, a) \]

First let’s derive the minimization for a generic quadratic function.

\[ Q(s, a) = s^T M_1 s + a^T M_2 a + 2 s^T M_3 a + c \]

Minimum must occur at a critical point.

\[ \nabla_a Q(s, a) = \nabla_a(s^T M_1 s) + \nabla_a(a^T M_2 a) + \nabla_a(s^T M_3 a) \]
\[ = 0 + 2 M_2 a + 2 M_3^T s \]

Then \( \nabla_a Q(s, a) = 0 \) when \( M_2 a = -M_3 s \)

for now assuming invertibility, \( a = -M_2^{-1} M_3^T s \)

Going back to \( Q_t^* (s, a) \)

\[ M_2 = R + B^T P_{t+1} B \quad \text{&} \quad M_3 = A^T P_{t+1} B \]

\( M_2 \) \text{ invertible}

Therefore, \( \pi_t^* (s) = - (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A \)

Define \( K_t^* \).
Last piece: check that \( V_t^*(s) = q_t^*(s) \Pi_t^*(s) \)
is quadratic & derive equations for \( P_t \) and \( p_t \)

Rather than plug in directly, recall our general quadratic function
\[
q(s, a^*) = s^T M_1 s + s^T M_3^{-1} M_2 M_2^T M_3^{-1} M_2^T s
- 2 s^T M_3^{-1} M_3^T s + c
= s^T (M_1 - M_3^{-1} M_3^T) s + c
\]

Therefore, this is the form of \( V_t^*(s) \).
Plugging \( M_1, M_2, M_3, c \) in, we have

\[
V_t^*(s) = s^T \left( q + \bar{A}^T P_{t+1} A - \bar{A}^T P_{t+1} B (R + B^T P_{t+1} B) B^T P_{t+1} A \right) s
+ \sigma^2 \text{tr}(P_{t+1}) + p_{t+1}
\]

This concludes the proof of Claim 2 and therefore the proof of Theorem.

Collecting the iterative definitions together:

\[
\begin{align*}
P_{t+1} &= Q, \quad P_{tt} = 0 \\
\text{for } t = t-1, \ldots, 0: \\
P_t &= Q + \bar{A}^T P_{t+1} A - \bar{A}^T P_{t+1} B (R + B^T P_{t+1} B) B^T P_{t+1} A \\
P_t &= p_{t+1} + \sigma^2 \text{tr}(P_{t+1}) \\
K_t &= (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A
\end{align*}
\]
Some straightforward Extensions:

1) time-varying costs/dynamics
   e.g. \( S_{t+1} = A_t S_t + B_t a_t + W_t \)
   \( c_t(s, a) = S_t^T Q_t S_t + A_t^T R_t A_t \)

2) non-stochastic disturbance
   \( S_{t+1} = A S_t + B a_t + W_t + V_t \)
   where \( V_t \) is known a priori

3) trajectory tracking
   \( c_t(s, a) = (s - S^*_t)^T Q (s - S^*_t) + (a - a^*_t)^T R (a - a^*_t) \)
   for desired trajectory \((s^*_0, a^*_0, ...)\) known a priori.
   (This case can be reduced to case 2 if substitute
   \( s \leftarrow s - S^*_t \) and \( a \leftarrow a - a^*_t \))