Reinforcement Learning June 5, 2019

Recitation 12

Lecturer:Yishay Mansour *TA: Lee Cohen*

LQR - Linear Quadratic Regulation

Typically a reinforcement learning problem can be described by a Markov Decision Process (MDP) consisting of a tuple (S, A, T, I, R) where S is the set of states, A is the set of actions, T is the time horizon, $I \subset S$ is the set of initial states. $R : S \times A \to \mathbb{R}$ is the reward (cost) function. As we deal with continuous state and action spaces, the focus is on (discretetime) dynamical systems where the transition dynamics can be described by a (non-linear) function $f : x_{t+1} = f(x_t, u_t)$ for some action $u_t \in A$. A policy π is a mapping from states $x \in S$ to actions $u \in A$. Acting according to policy π yields expected sum of rewards $V(\pi) = \sum_{t=0}^{H} R(x_t, u_t \mid \pi)$. The goal is to find an optimal policy π^* that maximizes the expected sum of rewards: $\pi^* = argmax_\pi \sum_{t=0}^{H} R(x_t, u_t \mid \pi)$

Motivation. The key question from this section is: assume that we know the dynamics and cost functions, how do we derive optimal actions (design controller) to minimize cost or maximize rewards?

We start with Linear Quadratic Regularization, which is a special case of the general MDP framework where the optimal policy can be computed exactly using dynamic programming. Consider discrete-time system with linear dynamics:

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

where $x_t \in S \subset \mathbb{R}^n$, $u_t \in A \subset \mathbb{R}^m$ are continuous state and action (a.k.a control input), with initial condition $x_0 = x^{init}$. For now let's assume $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are independent of time t - this is called Linear Time Invariant (LTI) dynamical system in classical control.

The reward function is given by the quadratic form:

$$J(x_t, u_t) = -x_t^T Q x_t - u_t^T R u_t \tag{2}$$

where $Q = Q^T \succeq 0$ and $R = R^T \succeq 0$ are positive semi-definite matrices. For convenience, we use the terms reward and cost function (negative of reward) interchangeably.

Assume that A, B, Q, R are known. Optimal actions for linear dynamical systems with quadratic cost can be computed efficiently with standard linear algebra operations.

For $t = 0, ..., T$ define the value function $V_t : \mathbb{R}^n \to \mathbb{R}$ by the recursion:

$$V_t(x) = \min_u \left(x^T Q x + u^T R u + V_{t+1}(Ax + Bu) \right) \tag{3}$$

LQR framework is convenient because all value functions take the quadratic form $V_t(x) = x^T P_t x$, for all $x \in \mathbb{R}^n$. This can be proved by backward induction on t.

First note that $V_T(x) = x^T Q x$ is quadratic in x, by definition of the cost function. Then assume $V_t(x) = x^T P_t x$ for all x, we have:

$$V_{t-1}(x) = \min_u \left[x^T Q x + u^T R u + V_t(Ax + Bu) \right] = \tag{4}$$

$$= \min_u \left[x^T Q x + u^T R u + (Ax + Bu)^T P_t (Ax + Bu) \right] = \tag{5}$$

$$= x^T \left(A^T P_t A + Q - (A^T P_t B)(B^T P_t B + R)^{-1}(B^T P_t A) \right) x \tag{6}$$

This enables exact dynamic programming solution to the optimal policy via the Ricatti recursion:

- initialize $P_T = Q$
- for each $t = T, T-1, ..., 1$ set:

  $$P_{t-1} = A^T P_t A + Q - (A^T P_t B)(B^T P_t B + R)^{-1}(B^T P_t A) \tag{7}$$

  This is the 'Algebraic Riccati Equation' (ARE).
- at each step, the optimal action is given by

  $$u_t^* = -(B^T P_t B + R)^{-1} B^T P_t A x_t \tag{8}$$

  it exists for any t if the system is controllable.

This is a well-known result in control theory: optimal policy for the LQR problem is a linear feedback controller, which can be efficiently computed using dynamic programming. The value function for any time step t is given by $V_t(x) = x^T P_t x$.
The ability to reach from any initial state x_0 any final state x_D, after a large enough time T, is called Controllability. Consider how the system evolves:

$$\begin{aligned} x_{t+1} =& Ax_t + Bu_t \\ =& A(Ax_{t-1} + Bu_{t-1}) + Bu_t \\ =& A^2 x_{t-1} + ABu_{t-1} + B_{u_t} \\ =& A^3 x_{t-2} + A^2 Bu_{t-2} + ABu_{t-1} + Bu_t \\ =& A^{t+1} x_0 + \sum_{i=0}^{t} A^i B u_{t-i} \end{aligned}$$

We can define now a matrix C and vector U, where

$$C = [B\ AB\ A^2B \cdots A^tB] \quad U = [u_t^\top\ u_{t-1}^\top \cdots u_0^\top]$$

and we have

$$x_{t+1} = A^{t+1}x_0 + CU$$

Therefore a sufficient condition for controllability is to have C full rank. This will ensure that we can enforce that $x_{t+1} = x_D$ for any x_D (by solving the linear equations).

Exercise 1.
Consider the following state space model for $T = 2$:

$$x_{t+1} = Ax_t + Bu_t \tag{9}$$

$$A = \begin{bmatrix} 0 & 1 \\ -8 & -6 \end{bmatrix},\ B = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{10}$$

We are interested in determining whether this model is controllable.

Solution 1.
First, let's compute the controllability matrix C:
$C = [B\ AB] = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & -6 & -6 \end{bmatrix}$

As you've seen in class, for the system to be controllable, C must have full rank. Keep in mind that an $n \times m$ matrix C (where $n \neq m$) is considered to have full rank if $rank(C) = min(n, m)$. So all we need is for C to have rank 2 - which it does. Thus the system is controllable.

* In general, solving even a simple LQR problem by hand can be hard for a 2-state system, and becomes exceedingly difficult for larger systems. Luckily, Python Control Systems Library can give us the desired gain right away via control.lqr(A,B,Q,R,[,N]).

Exercise 2.

Consider the following LQR problem:

$$A = \begin{bmatrix} -1 & 0 \\ -1 & 0 \end{bmatrix},\ B = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ R = [1] \tag{11}$$

Note that $Q \succcurlyeq 0$, $R \succcurlyeq 0$

We are looking for a unique solution to the ARE which is a symmetric (positive semi-definite) matrix P:

$$P = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

Hence all we need to find are the values of a, b, c.

Solution 2.
The Algebraic Riccati Eequation is

$$P = Q + A^T PA - (A^T PB)(R + B^T PB)^{-1}(B^T PA) \tag{12}$$

$$A^T PA = \begin{bmatrix} -1 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} -1 & 0 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -a-b & 0 \\ -b-c & 0 \end{bmatrix} = \begin{bmatrix} a+2b+c & 0 \\ 0 & 0 \end{bmatrix}$$

$$A^T PB = \begin{bmatrix} -1 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} -a-b \\ 0 \end{bmatrix}$$

$$B^T PB = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = a$$

$$B^T PA = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} -1 & 0 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} -a-b & 0 \\ -b-c & 0 \end{bmatrix} = \begin{bmatrix} -a-b & 0 \end{bmatrix}$$

$$(A^T PB)(R+B^T PB)^{-1}(B^T PA) = \begin{bmatrix} -a-b \\ 0 \end{bmatrix} (1+a)^{-1} \begin{bmatrix} -a-b & 0 \end{bmatrix} = (1+a)^{-1} \begin{bmatrix} (a+b)^2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} a+2b+c & 0 \\ 0 & 0 \end{bmatrix} - (1+a)^{-1} \begin{bmatrix} (a+b)^2 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1+a+2b+c-(1+a)^{-1}(a+b)^2 & 0 \\ 0 & 1 \end{bmatrix}$$

$a = -2$
$b = 0$
$c = 1$

References:

1. Feedback Control Systems, Fall 2010 - Recitation 7 https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-30-feedback-control-systems-fall-2010/recitations/MIT16_30F10_rec07.pdf

2. LQR, Inverse Reinforcement Learning, Learning from Expert Demonstrations http://www.yisongyue.com/courses/cs159/lectures/helicopter_course_notes.pdf