

# ECE586RL: MDPs and Reinforcement Learning

## A Note on Linear Quadratic Regulator

Instructor: Bin Hu

February 22, 2020

In this note, we review some basic facts related to the linear quadratic regulator (LQR) which is used as a benchmark problem for reinforcement learning.

### 1 LQR without Process Noise

We start from a simple setup. Consider the following linear dynamical system

$$x_{t+1} = Ax_t + Bu_t \tag{1.1}$$

where  $A$  is the state matrix,  $B$  is the input matrix,  $u_t$  is the control action, and  $x_t$  is the system state. In the class, we start with the case where there is a stochastic process noise  $w_t$  in the dynamics. For simplicity, here we start with the case where there is no process noise  $w_t$ .

The objective is to choose  $\{u_t\}$  to minimize the following cost

$$\mathcal{C} = \mathbb{E}_{x_0 \sim \mathcal{D}} \sum_{t=0}^{\infty} (x_t^\top Q x_t + u_t^\top R u_t) \tag{1.2}$$

where  $Q$  and  $R$  are positive definite matrices. Since there is no process noise, the only randomness stems from the randomness in  $x_0$ . So there is an initial distribution  $\mathcal{D}$  where  $x_0$  is sampled from. The choices of  $Q$  and  $R$  reflect the conflicting design objectives in control: We want to achieve small tracking error by using small control inputs. The above setup is a little different from the discounting setup covered in the class. One main difference is that there is no discounting factor in the cost (1.2). Since there is no process noise  $w_t$ , we do not need a discounting factor  $\gamma$  to make  $\mathcal{C}$  finite. When there is the process noise term  $w_t$ , the cost in (1.2) is never finite for  $\gamma = 1$  due to the fact that  $x_t$  does not converge to 0. In that case, a discounting factor is needed. Nevertheless, the above LQR problem can still be viewed as a MDP on a continuous state space. A few key features are summarized as follows.

1. Continuous state space:  $x_t$  is a real vector and hence can take any values in  $\mathbb{R}^x$ .
2. Continuous action space:  $u_t$  is also a real vector.
3. Transition dynamics: Given  $x_t$  and  $u_t$ , then  $x_{t+1}$  is also known due to (1.1). The transition dynamics can be viewed a stochastic kernel centering at  $(Ax_t + Bu_t)$  with probability 1 .

4. Additive structure of cost function:  $\mathcal{C}$  is a sum of cost values at different  $t$ . The one-step cost depends on both the state and the input at that step.
5. No discounting factor in cost  $\mathcal{C}$ .

Given an initial condition  $x_0$ , we denote the state value function as  $V(x_0) = \sum_{t=0}^{\infty} (x_t^\top Q x_t + u_t^\top R u_t)$ . Therefore, we have  $\mathcal{C} = \mathbb{E}_{x \sim \mathcal{D}} V(x)$ .

## 1.1 Policy Parameterization

As discussed in the class, it is reasonable to design  $u_t$  using the past information of the system. This type of closed-loop design is more robust to disturbance and model uncertainty. When the state  $x_t$  can be measured, it is reasonable to just consider the stationary policy in the form of state feedback, i.e.  $u_t = \mu(x_t)$ . For the finite state space case,  $\mu$  can be represented as a table. Here,  $\mu$  is a function mapping from  $\mathbb{R}^x$  to  $\mathbb{R}^u$ . We want to design  $\mu$  to minimize  $\mathcal{C}$ . In practice, we have to confine the search of  $\mu$  within certain tractable classes of functions. Here are a few examples.

1. Linear policy:  $\mu$  is parameterized by a matrix  $K$ , and we have  $x_t = -K u_t$ . Eventually the problem becomes finding  $K$  to minimize  $\mathcal{C}$ .
2. Neural network:  $\mu$  can be parameterized by a neural network, i.e.  $u_t = W_L \phi(W_{L-1} \dots W_2 \phi(W_1 x_t))$  where  $L$  is the depth of the network. This is a more popular choice for complex nonlinear control tasks. Then the problem becomes minimizing  $\mathcal{C}$  over the weights  $(W_1, W_2, \dots, W_L)$ .

## 1.2 Policy Evaluation

Notice the cost (1.2) provides a measure for the performance of the policy  $\mu$ . The smaller the cost  $\mathcal{C}$  is, the better the policy  $\mu$  performs. Policy evaluation refers to the calculation of  $\mathcal{C}$  for a given policy  $\mu$ .

**Stability issue.** Unlike the finite state space case,  $\mathcal{C}$  may not be finite for a given  $\mu$ . So there is a stability issue here. If  $\mu$  stabilizes the system (1.1), then the cost (1.2) is finite.

**Parameterization of value function** In this note, we focus the policy evaluation for a linear policy  $u_t = -K x_t$ . Substituting  $u_t = -K x_t$  into (1.1) leads to  $x_{t+1} = (A - BK)x_t$ . Hence we have

$$x_t = (A - BK)^t x_0 \quad (1.3)$$

We denote the spectral radius as  $\rho$ . If  $K$  stabilizes the system (1.1), then  $\rho(A - BK) < 1$  and  $x_t \rightarrow 0$  at a geometric rate. The cost (1.2) can be rewritten as

$$\mathcal{C}(K) = \mathbb{E}_{x_0 \sim \mathcal{D}} x_0^\top \left( \sum_{t=0}^{\infty} ((A - BK)^\top)^t (Q + K^\top R K) (A - BK)^t \right) x_0 \quad (1.4)$$

When  $\rho(A - BK) \geq 1$ , the above cost blows up to infinity. It makes sense to restrict the policy search within the class of stabilizing  $K$ . When  $\rho(A - BK) < 1$ , we know  $\sum_{t=0}^{\infty} ((A - BK)^\top)^t (Q + K^\top R K) (A - BK)^t$  will converge to a fixed constant matrix. We denote this matrix by  $P_K$ . Therefore, it is reasonable to parameterize the value function as  $x_0^\top P_K x_0$  which is a quadratic function of  $x_0$ . When a nonlinear policy is used, we typically need to parameterize the value function as a neural network.

**Bellman equation for policy evaluation.** From the above discussion, we have already known  $P_K = \sum_{t=0}^{\infty} ((A - BK)^T)^t (Q + K^T RK) (A - BK)^t$ . The bellman equation can be derived as follows.

$$\begin{aligned}
x_0^T P_K x_0 &= \sum_{t=0}^{\infty} x_t^T (Q + K^T RK) x_t \\
&= x_0^T (Q + K^T RK) x_0 + \sum_{t=1}^{\infty} x_t^T (Q + K^T RK) x_t \\
&= x_0^T (Q + K^T RK) x_0 + x_1^T P_K x_1 \\
&= x_0^T (Q + K^T RK) x_0 + x_0^T (A - BK)^T P_K (A - BK) x_0 \\
&= x_0^T \left( Q + K^T RK + (A - BK)^T P_K (A - BK) \right) x_0
\end{aligned}$$

Therefore, the Bellman equation takes the following form:

$$P_K = Q + K^T RK + (A - BK)^T P_K (A - BK) \quad (1.5)$$

For any fixed  $K$ , the above equation is a linear equation of  $P_K$ . Hence the existence and uniqueness of the solution to the above Bellman equation can be established using linear equation theory.

To obtain a closed-form solution for  $P_K$ , we need to introduce the Kronecker product and the vectorization operation. The Kronecker product of two matrices  $A \in \mathbb{R}^{m \times n}$  and  $B \in \mathbb{R}^{p \times q}$  is denoted by  $A \otimes B$  and given by:

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}.$$

where  $a_{ij}$  is the  $(i, j)$ -th entry of  $A$ . Clearly, we have  $A \otimes B \in \mathbb{R}^{pm \times qn}$ . Notice  $(A \otimes B)^T = A^T \otimes B^T$  and  $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$  when the matrices have compatible dimensions.

Next, let  $\text{vec}$  denote the standard vectorization operation that stacks the columns of a matrix into a vector. For example, we have

$$\text{vec} \left( \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 2 \\ 4 \\ 6 \end{bmatrix}.$$

An important fact is that we always have  $\text{vec}(AXB) = (B^T \otimes A) \text{vec}(X)$ . Therefore, we have

$$\text{vec} \left( (A - BK)^T P_K (A - BK) \right) = \left( (A - BK)^T \otimes (A - BK)^T \right) \text{vec}(P_K)$$

Then we can vectorize both sides of the Bellman equation (1.5) to obtain

$$\text{vec}(P_K) = \text{vec}(Q + K^T RK) + \left( (A - BK)^T \otimes (A - BK)^T \right) \text{vec}(P_K)$$

which can be easily solved for  $P_K$ :

$$\text{vec}(P_K) = \left( I - (A - BK)^\top \otimes (A - BK)^\top \right)^{-1} \text{vec}(Q + K^\top RK)$$

Now we have a closed-form solution for  $P_K$ . Using properties of the Kronecker product, one can show  $(I - (A - BK)^\top \otimes (A - BK)^\top)$  is nonsingular under the assumption  $\rho(A - BK) < 1$ . We skip the details here. The key message here is that for any stabilizing  $K$ , we can solve (1.5) to obtain  $P_K$  and then the value function for  $K$  is  $V(x) = x^\top P_K x$ .

For general nonlinear policy, the existence and uniqueness conditions for Bellman equation are much more complicated.

### 1.3 Optimal Bellman Equation: Riccati Equation

For the above LQR problem, the optimal Bellman equation becomes the following Riccati equation

$$P = A^\top P A + Q - A^\top P B (B^\top P B + R)^{-1} B^\top P A \quad (1.6)$$

When  $(A, B)$  is stabilizable, the above equation has a unique positive definite<sup>1</sup> stabilizing solution  $P$  such that  $\rho(A - B(B^\top P B + R)^{-1} B^\top P A) < 1$ . Then the optimal action is given by a linear policy, i.e.  $u_t = -(B^\top P B + R)^{-1} B^\top P A x_t$ . This linear policy is optimal among all (potentially nonlinear) policies. This can be shown using completion of square. We can rewrite the cost as follows.

$$\begin{aligned} \sum_{t=0}^{\infty} (x_t^\top Q x_t + u_t^\top R u_t) &= x_0^\top P x_0 + \sum_{t=0}^{\infty} (x_t^\top Q x_t + u_t^\top R u_t + x_{t+1}^\top P x_{t+1} - x_t^\top P x_t) \\ &= x_0^\top P x_0 + \sum_{t=0}^{\infty} \left( x_t^\top Q x_t + u_t^\top R u_t + (A x_t + B u_t)^\top P (A x_t + B u_t) - x_t^\top P x_t \right) \\ &= x_0^\top P x_0 + \sum_{t=0}^{\infty} \left( x_t^\top (Q + A^\top P A - P) x_t + u_t^\top (R + B^\top P B) u_t + x_t^\top A^\top P B u_t + u_t^\top B^\top P A x_t \right) \end{aligned}$$

By the Riccati equation, we have  $Q + A^\top P A - P = A^\top P B (B^\top P B + R)^{-1} B^\top P A$ . Hence we have

$$\begin{aligned} &\sum_{t=0}^{\infty} (x_t^\top Q x_t + u_t^\top R u_t) \\ &= x_0^\top P x_0 + \sum_{t=0}^{\infty} \left( x_t^\top A^\top P B (B^\top P B + R)^{-1} B^\top P A x_t + u_t^\top (R + B^\top P B) u_t + x_t^\top A^\top P B u_t + u_t^\top B^\top P A x_t \right) \\ &= x_0^\top P x_0 + \sum_{t=0}^{\infty} \left( (B^\top P B + R) u_t + B^\top P A x_t \right)^\top (B^\top P B + R)^{-1} \left( (B^\top P B + R) u_t + B^\top P A x_t \right) \end{aligned}$$

Notice  $B^\top P B + R$  is positive definite. Therefore, the cost achieves its minimal value  $x_0^\top P x_0$  if the action is chosen to satisfy  $(B^\top P B + R) u_t + B^\top P A x_t = 0$ . We have skipped a few technical details but the above calculation roughly explain why the optimal policy is given by  $K = (B^\top P B + R)^{-1} B^\top P A$ .

---

<sup>1</sup>For simplicity, we assume  $Q$  is positive definite in this note. Hence the solution  $P$  is positive definite. In general, it is possible to allow  $Q$  to be only positive semidefinite and the solution  $P$  can also become positive semidefinite. Some extra observability assumption is then needed.

**How does the Riccati equation arise?** Usually one first looks at LQR on a finite horizon and then extends the result there to the infinite horizon setup. Instead of doing that, we will provide an alternative informal derivation that starts from the assumption that the optimal policy is linear. Under this assumption, the optimal value function is quadratic and takes the form of  $x^\top Px$ . The optimal Bellman equation states the following fact.

$$x^\top Px = \min_u (x^\top Qx + u^\top Ru + (Ax + Bu)^\top P(Ax + Bu)) \quad (1.7)$$

Clearly, the right side is a quadratic function in  $u$ . Taking the gradient of the right hand side with respect to  $u$  leads to the following equation:

$$(R + B^\top PB)u + B^\top PAx = 0$$

The optimal  $u$  is given as  $u = -(R + B^\top PB)^{-1}B^\top PAx$ . We can substitute this relation to the right side of (1.7) and obtain

$$P = Q + A^\top PB(R + B^\top PB)^{-1}R(R + B^\top PB)^{-1}B^\top PA \\ + \left(A - B(B^\top PB + R)^{-1}B^\top PA\right)^\top P \left(A - B(B^\top PB + R)^{-1}B^\top PA\right)$$

Simplifying the above equation leads to the Riccati equation (1.6).

**Key message.** When writing out the optimal Bellman equation, we need to know how to parameterize the optimal value function. In the LQR problem, we know we can parameterize the optimal value function as a quadratic function. Therefore, eventually the optimal Bellman equation (1.7) is equivalent to an algebraic Riccati equation on  $P$ . If we just use a general value function  $V(x)$ , then (1.7) becomes

$$V(x) = \min_u (x^\top Qx + u^\top Ru + V(Ax + Bu)) \quad (1.8)$$

How to figure out  $V$  becomes a difficult task. For nonlinear control, the situation is even worse. Suppose the dynamics become  $x_{t+1} = f(x_t, u_t)$  and the one-stage cost is  $c(x, u)$  where  $c$  is some complicated function. Then the optimal Bellman equation is in the following nonlinear form:

$$V(x) = \min_u (c(x, u) + V(f(x, u))) \quad (1.9)$$

which is extremely difficult to solve. One has to approximate  $V$  using neural networks.

## 1.4 Value Iteration

Suppose we decide to parameterize the value function as  $V(x) = x^\top Px$ . The Bellman operator  $T$  maps  $P$  to  $P'$  as follows

$$x^\top P'x = \min_u (x^\top Qx + u^\top Ru + (Ax + Bu)^\top P(Ax + Bu)) \quad (1.10)$$

Clearly, the optimal Bellman equation (1.7) is equivalent to  $P = T(P)$  and just gives the fixed point for the above Bellman operator. Again, the right side of (1.10) is a quadratic function of  $u$ , and minimizing over  $u$  gives

$$P' = A^\top PA + Q - A^\top PB(B^\top PB + R)^{-1}B^\top PA \quad (1.11)$$

Therefore, we can apply the Bellman operator recursively and obtain a value iteration algorithm:

$$P^{n+1} = A^\top P^n A + Q - A^\top P^n B (B^\top P^n B + R)^{-1} B^\top P^n A \quad (1.12)$$

We can clearly see, for the continuous state space case, we need to parameterize the value function and then iteratively update these parameters.

## 1.5 Policy Iteration

For policy iteration, we need to parameterize the policy and recursively update these policy parameters. Let's say we use a linear policy  $K$ . At step  $n$ , the policy update is  $K^n$ . Then we evaluate the value for this policy by solving the Bellman equation  $(A - BK^n)^\top P^n (A - BK^n) + Q + (K^n)^\top R K^n = P^n$ . Once  $P^n$  is obtained, we update  $x = -K^{n+1}u$  by solving

$$\arg \min_u (x^\top Q x + u^\top R u + (Ax + Bu)^\top P^n (Ax + Bu)) \quad (1.13)$$

Minimizing the above cost over  $u$  leads to  $u = -(B^\top P^n B + R)^{-1} B^\top P^n A$ , and hence we have  $K^{n+1} = (B^\top P^n B + R)^{-1} B^\top P^n A$  where  $P_n$  solves  $(A - BK^n)^\top P^n (A - BK^n) + Q + (K^n)^\top R K^n = P^n$ .

**Differences between PI and VI.** In VI, the optimal  $u$  for (1.10) is substituted back to the Bellman operator to obtain  $P^{n+1}$ . In PI,  $P^{n+1}$  is solved by evaluating the value for this  $u$  (hence we need to solve a fixed point here).

## 2 LQR with Process Noise

In this case, the dynamics become

$$x_{t+1} = Ax_t + Bu_t + w_t \quad (2.1)$$

where  $w_t$  is an IID Gaussian noise. Given any  $x_0$ , the state  $x_t$  does not converge to 0 and the cost in (1.2) is not finite. A fix is to use the discounting factor. Now we consider the cost function

$$\mathcal{C} = \mathbb{E} \sum_{t=0}^{\infty} \gamma^t (x_t^\top Q x_t + u_t^\top R u_t) \quad (2.2)$$

where  $0 < \gamma < 1$  is the discounting factor. Again, this is a MDP problem. A key fact is that the probability distribution of  $x_{t+1}$  is completely known if  $x_t$  and  $u_t$  are seen. This is due to the IID nature of  $w_t$ . When  $w_t$  is Gaussian, the transition density will also be Gaussian.

Now we briefly highlight some differences introduced by this noise term.

1. Policy evaluation: Given a linear policy  $K$ , the cost becomes

$$\mathcal{C}(K) = r_K + \mathbb{E}_{x_0 \sim \mathcal{D}} x_0^\top \left( \sum_{t=0}^{\infty} \gamma^t ((A - BK)^\top)^t (Q + K^\top R K) (A - BK)^t \right) x_0 \quad (2.3)$$

where  $r_K$  is some extra term introduced by the noise  $w_t$ . Therefore, we can parameterize the value function as  $x^\top P_K x + r_K$ . Then the Bellman equation becomes

$$P_K = Q + K^\top R K + \gamma (A - BK)^\top P_K (A - BK)$$

2. Optimal Bellman equation: The Riccati equation now depends on  $\gamma$  and becomes

$$P = Q + \gamma A^\top P A - \gamma^2 A^\top P B (R + \gamma B^\top P B)^{-1} B^\top P A$$

The optimal policy becomes  $u_t = -\gamma(R + \gamma B^\top P B)^{-1} B^\top P A$ .

3. Value iteration: Algorithm (1.12) should be modified as

$$P^{n+1} = \gamma A^\top P^n A + Q - \gamma^2 A^\top P^n B (\gamma B^\top P^n B + R)^{-1} B^\top P^n A \quad (2.4)$$

4. Policy iteration: In this case, the PI algorithm iterates as  $K^{n+1} = \gamma(\gamma B^\top P^n B + R)^{-1} B^\top P^n A$  where  $P_n$  solves the Bellman equation  $\gamma(A - BK^n)^\top P^n (A - BK^n) + Q + (K^n)^\top R K^n = P^n$ .

You should try to derive the above results yourself! All you need to do is to modify the derivations in Section 1.