



# Reinforcement Learning Notes

**Learn from trying!**

**Author:** occupymars

**Date:** June. 20, 2025

**Version:** 0.1

# Contents

<b>Chapter 1 Reinforcement Learning Basics</b>	<b>1</b>
1.1 Markov Decision Process . . . . .	1
1.2 Value Function . . . . .	1
1.3 Solving Value Function . . . . .	2
1.4 Action Value Function . . . . .	2
1.5 Bellman Optimality Equation . . . . .	3
<b>Chapter 2 From LQR to RL</b>	<b>4</b>
2.1 LQR and Value function . . . . .	4

# Chapter 1 Reinforcement Learning Basics

## Introduction

- Markov Decision Process
- Value Function
- Solving Value Function

- Action Value Function
- Bellman Optimality Equation

## 1.1 Markov Decision Process

**State** and **Action** can describe a robot state respect to the enviroment and actions to move around,  $\mathcal{S}, \mathcal{A}$  are states and actions a robot can take, when taking an action, state after may not be deterministic, it has a probability. We use a transition function  $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$  to denote this,  $T(s, a, s') = P(s' | s, a)$  is the probability of reaching  $s'$  given  $s$  and  $a$ . For  $\forall s \in \mathcal{S}$  and  $\forall a \in \mathcal{A}$ ,  $\sum_{s' \in \mathcal{S}} T(s, a, s') = 1$ .

**Reward**  $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ ,  $r(s, a)$  depends on current state and action. And the reward may also be stochastic, given state and action, the reward has probability  $p(r | s, a)$ .

**Policy**  $\pi(a | s)$  tells agent which actions to take at every state,  $\sum_a \pi(a | s) = 1$ .

This can build a Markov Decision Process,  $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r)$  from the **Trajectory**  $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \dots)$ , which has probability of:

$$P(\tau) = \pi(a_0 | s_0) \cdot P(s_1 | s_0, a_0) \cdot \pi(a_1 | s_1) \cdot P(s_2 | s_1, a_1) \cdots$$

We then define **Return** as the total reward  $R(\tau) = \sum_t r_t$ , the goal of reinforcement learning is to find a trajectory that has the largest return. The trajectory might be infinite, so in order for a meaningful formular of its return, we introduce a discount factor  $\gamma < 1$ ,  $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$ . For large  $\gamma$ , the robot is encouraged to explore, for small one to take a short trajectory to goal.

Markov system only depend on current state and action, not the history one (but we can always augment the system).

## 1.2 Value Function

**Value Function** is the value of a state, from that state, the expected sum reward (return).

The formular of value function is:

$$V^\pi(s_0) = \mathbb{E}_{a_t \sim \pi(s_t)}[R(\tau)] = \mathbb{E}_{a_t \sim \pi(s_t)} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \quad (1.1)$$

If we divded the trajectory into two parts,  $s_0$  and  $\tau'$ , we get the return:

$$R(\tau) = r(s_0, a_0) + \gamma \sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t) = r(s_0, a_0) + \gamma R(\tau')$$

Put it back into the value function, using law of total expectation:

$$\mathbb{E}[X] = \sum_a \mathbb{E}[X | A = a] p(a) = \mathbb{E}_a [\mathbb{E}[X | A = a]]$$

we get:

$$\begin{aligned} V^\pi(s_0) &= \mathbb{E}_{a_t \sim \pi(s_t)}[r(s_0, a_0) + \gamma R(\tau')] \\ &= \mathbb{E}_{a_0 \sim \pi(s_0)}[r(s_0, a_0)] + \gamma \mathbb{E}_{a_t \sim \pi(s_t)}[R(\tau')] \\ &= \mathbb{E}_{a_0 \sim \pi(s_0)}[r(s_0, a_0)] + \gamma \mathbb{E}_{a_0 \sim \pi(s_0)} [\mathbb{E}_{s_1 \sim P(s_1 | a_0, s_0)} [\mathbb{E}_{a_t \sim \pi(s_t)}[R(\tau') | s_1, a_0]]] \\ &= \mathbb{E}_{a_0 \sim \pi(s_0)}[r(s_0, a_0)] + \gamma \mathbb{E}_{a_0 \sim \pi(s_0)} [\mathbb{E}_{s_1 \sim P(s_1 | a_0, s_0)} [V^\pi(s_1)]] \\ &= \mathbb{E}_{a \sim \pi(s)} [r(s_0, a_0) + \gamma \mathbb{E}_{s_1 \sim P(s_1 | a_0, s_0)} [V^\pi(s_1)]] \end{aligned} \quad (1.2)$$

before we put  $s_1$  to the right as the condition, it is stochastic, inside the  $\mathbb{E}_{s_1 \sim P(s_1 | s_0, a_0)}$  scope it is deterministic, then we can get  $V^\pi(s_1)$ , as it needs the state to be deterministic.

The discrete formular is (get rid of the notation of time) so called **Bellman Equation**:

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a | s) \left[ r(s, a) + \gamma \sum_{s'} P(s' | s, a) V^\pi(s') \right], \forall s \in \mathcal{S} \quad (1.3)$$

And if we write  $r(s, a)$  as  $\sum_r p(r | s, a)r$ , then

$$p(r | s, a) = \sum_{s' \in \mathcal{S}} p(s', r | s, a)$$

We can also get

$$p(s' | s, a) = \sum_{r \in \mathcal{R}} p(s', r | s, a)$$

combined we get

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a | s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a) [r + \gamma V^\pi(s')] \quad (1.4)$$

If the reward depend solely on the next state  $s'$ , then

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a | s) \sum_{s' \in \mathcal{S}} P(s' | s, a) [r(s') + \gamma V^\pi(s')] \quad (1.5)$$

Let

$$\begin{aligned} r^\pi(s) &= \sum_{a \in \mathcal{A}} \sum_r p(r | s, a)r \\ p^\pi(s' | s) &= \sum_{a \in \mathcal{A}} p(s' | s, a) \end{aligned}$$

rewirte 1.3 into the vector form:

$$V^\pi = r^\pi + \gamma P^\pi V^\pi \quad (1.6)$$

where  $V^\pi = [V^\pi(s_1), \dots, V^\pi(s_n)]^\top \in \mathbb{R}^n$ ,  $r^\pi = [r^\pi(s_1), \dots, r^\pi(s_n)]^\top \in \mathbb{R}^n$ , and  $P^\pi \in \mathbb{R}^{n \times n}$  with  $P_{ij}^\pi = p^\pi(s_j | s_i)$ .

## 1.3 Solving Value Function

Next, we need to solve the value function, first way is closed-form solution:

$$V^\pi = (I - \gamma P^\pi)^{-1} r^\pi$$

Some properties:  $I - \gamma P^\pi$  is invertible,  $(I - \gamma P^\pi)^{-1} \geq I$  which means every element of this inverse is nonnegative. For every vector  $r \geq 0$ , it holds that  $(I - \gamma P^\pi)^{-1} r^\pi \geq r \geq 0$ , so if  $r_1 \geq r_2$ ,  $(I - \gamma P^\pi)^{-1} r_1^\pi \geq (I - \gamma P^\pi)^{-1} r_2^\pi$

However, this method need to calculate the inverse of the matrix, that need some numerical algorithms. We can use a iterative solution:

$$V_{k+1} = r^\pi + \gamma P^\pi V_k$$

as  $k \rightarrow \infty$ ,  $V_k \rightarrow V^\pi = (I - \gamma P^\pi)^{-1} r^\pi$ .

## 1.4 Action Value Function

Similarly to value funtion, **Action Value Function** is the value of an action at state  $s$ , from that state, take that action, the expected sum reward (return). We use  $V^\pi(s)$  to denote value function, and  $Q^\pi(s, a)$  to denote action value, their connection is:

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a | s) Q^\pi(s, a) \quad (1.7)$$

The action value function is given as:

$$\begin{aligned}
Q^\pi(s_0, a_0) &= r(s_0, a_0) + \mathbb{E}_{a_t \sim \pi(s_t)} [\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t)] \\
&= r(s_0, a_0) + \gamma \mathbb{E}_{a_t \sim \pi(s_t)} [\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t)] \\
&= r(s_0, a_0) + \gamma \mathbb{E}_{a_t \sim \pi(s_t)} [R(\tau')] \\
&= r(s_0, a_0) + \gamma \mathbb{E}_{s_1 \sim p(s_1 | s_0, a_0)} [\mathbb{E}_{a_t \sim \pi(s_t)} [R(\tau') | s_1]] \\
&= r(s_0, a_0) + \gamma \mathbb{E}_{s_1 \sim p(s_1 | s_0, a_0)} [V^\pi(s_1)] \\
&= r(s_0, a_0) + \gamma \mathbb{E}_{s_1 \sim p(s_1 | s_0, a_0)} [\sum_{a_1 \in \mathcal{A}} \pi(a_1 | s_1) Q^\pi(s_1, a_1)]
\end{aligned} \tag{1.8}$$

Then the bellman equation of action value is:

$$\begin{aligned}
Q^\pi(s, a) &= r(s, a) + \gamma \sum_{s'} P(s' | s, a) V^\pi(s') \\
&= r(s, a) + \gamma \sum_{s'} P(s' | s, a) \sum_{a' \in \mathcal{A}} \pi(a' | s') Q^\pi(s', a')
\end{aligned} \tag{1.9}$$

Note that we can always write  $r(s, a)$  as  $\sum_r p(r | s, a) r$  if it is stochastic, and it follows the same notation in the book *Math of Reinforcement Learning*.

Rewrite 1.9 into vector form:

$$Q^\pi = \tilde{r}^\pi + \gamma P^\pi \Pi^\pi Q^\pi \tag{1.10}$$

where  $\tilde{r}_{(s,a)}^\pi = \sum_r p(r | s, a) r$ ,  $P_{(s,a),s'}^\pi = p(s' | s, a)$ ,  $\Pi_{s',(s',a')}^\pi = \pi(a' | s')$ .

## 1.5 Bellman Optimality Equation

*Bellman Optimality Equation* is given by:

$$\begin{aligned}
V(s) &= \max_{\pi(s) \in \Pi(s)} \sum_{a \in \mathcal{A}} \pi(a | s) [\sum_r p(r | s, a) r + \gamma \sum_{s'} P(s' | s, a) V^\pi(s')] \\
&= \max_{\pi(s) \in \Pi(s)} \sum_{a \in \mathcal{A}} \pi(a | s) Q(s, a)
\end{aligned} \tag{1.11}$$

## Chapter 2 From LQR to RL

### Introduction

□ LQR Problem

□ Reinforcement Learning

□ iLQR and DDP

### 2.1 LQR and Value function

Given a linear model  $x_{t+1} = f(x_t, u_k) = A_t x_t + B_t u_t + C_t$ . We want to optimize:

$$\min_{u_1, \dots, u_T} c(x_1, u_1) + c(f(x_1, u_1), u_2) + \dots + c(f(f(\dots)), u_T)$$

where we denotes

$$c(x_t, u_t) = \frac{1}{2} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T C_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T c_t$$

and

$$f(x_t, u_t) = F_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + f_t$$

We first do the **Backward Recursion**, solve for  $u_T$  only, then the action value function (or the negative cost function, here we take them with same sign) is:

$$Q(x_T, u_T) = \text{const} + \frac{1}{2} \begin{bmatrix} x_T \\ u_T \end{bmatrix}^T C_T \begin{bmatrix} x_T \\ u_T \end{bmatrix} + \begin{bmatrix} x_T \\ u_T \end{bmatrix}^T c_T$$

Get the derivative respect to  $u_T$ , which is:

$$\nabla_{u_T} Q(x_T, u_T) = C_{u_T, x_T} x_T + C_{u_T, u_T} u_T + c_{u_T}^T = 0$$

so we can get  $K_T = -C_{u_T, u_T}^{-1} C_{u_T, x_T}$ ,  $k_T = -C_{u_T, u_T}^{-1} c_{u_T}$ .

And we get the policy (which is a linear policy):  $u_T = K_T x_T + k_T$ . Because  $u_T$  is fully determined by  $x_T$ , we can eliminate it via substitution:

$$V(x_T) = \text{const} + \frac{1}{2} \begin{bmatrix} x_T \\ K_T x_T + k_T \end{bmatrix}^T C_T \begin{bmatrix} x_T \\ K_T x_T + k_T \end{bmatrix} + \begin{bmatrix} x_T \\ K_T x_T + k_T \end{bmatrix}^T c_T$$

Open the equation:

$$\begin{aligned} V(x_T) &= \text{const} + \frac{1}{2} \mathbf{x}_T^T \mathbf{V}_T \mathbf{x}_T + \mathbf{x}_T^T \mathbf{v}_T \\ \mathbf{V}_T &= \mathbf{C}_{\mathbf{x}_T, \mathbf{x}_T} + \mathbf{C}_{\mathbf{x}_T, \mathbf{u}_T} \mathbf{K}_T + \mathbf{K}_T^T \mathbf{C}_{\mathbf{u}_T, \mathbf{x}_T} + \mathbf{K}_T^T \mathbf{C}_{\mathbf{u}_T, \mathbf{u}_T} \mathbf{K}_T \\ \mathbf{v}_T &= \mathbf{c}_{\mathbf{x}_T} + \mathbf{C}_{\mathbf{x}_T, \mathbf{u}_T} \mathbf{k}_T + \mathbf{K}_T^T \mathbf{c}_{\mathbf{u}_T} + \mathbf{K}_T^T \mathbf{C}_{\mathbf{u}_T, \mathbf{u}_T} \mathbf{k}_T \end{aligned}$$

Use  $x_{T-1}$  and  $u_{T-1}$  to substitute the action value equation:

$$Q(x_{T-1}, u_{T-1}) = \text{const} + \frac{1}{2} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T C_{T-1} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix} + \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T c_{T-1} + V(f(x_{T-1}, u_{T-1}))$$

And the value function of  $x_T$  can be written as:

$$V(x_T) = \text{const} + \frac{1}{2} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T F_{T-1}^T V_T F_{T-1} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix} + \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T F_{T-1}^T V_T f_{T-1} + \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T F_{T-1}^T \mathbf{v}_T$$

So the action value function is:

$$Q(x_{T-1}, u_{T-1}) = \text{const} + \frac{1}{2} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T Q_{T-1} \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix} + \begin{bmatrix} x_{T-1} \\ u_{T-1} \end{bmatrix}^T q_{T-1}$$

where

$$\begin{aligned} Q_{T-1} &= C_{T-1} + F_{T-1}^T V_T F_{T-1} \\ q_{T-1} &= c_{T-1} + F_{T-1}^T V_T f_{T-1} + F_{T-1}^T v_T \end{aligned}$$

get the derivative:

$$\nabla_{u_{T-1}} Q(x_{T-1}, u_{T-1}) = Q_{u_{T-1}, x_{T-1}} x_{T-1} + Q_{u_{T-1}, u_{T-1}} u_{T-1} + q_{u_{T-1}}^T = 0$$

where

$$\begin{aligned} u_{T-1} &= K_{T-1} x_{T-1} + k_{T-1} \\ K_{T-1} &= -Q_{u_{T-1}, u_{T-1}}^{-1} Q_{u_{T-1}, x_{T-1}} \\ k_{T-1} &= -Q_{u_{T-1}, u_{T-1}}^{-1} q_{u_{T-1}} \end{aligned}$$

So we can continue the substitution process to the first control  $u_0$ , with the information of  $x_0$ , we start a **Forward Recursion**,  $u_t = K_t x_t + k_t$ ,  $x_{t+1} = f(x_t, u_t)$  to get the future states.

We can write the Backward Recursion as algorithm:

---

**Algorithm 1:** Backward Pass for Value Function Computation

---

```

1 for  $t = T$  to 1 do
2    $\mathbf{Q}_t = \mathbf{C}_t + \mathbf{F}_t^T \mathbf{V}_{t+1} \mathbf{F}_t$ ;
3    $\mathbf{q}_t = \mathbf{c}_t + \mathbf{F}_t^T \mathbf{V}_{t+1} \mathbf{f}_t + \mathbf{F}_t^T \mathbf{v}_{t+1}$ ;
4    $Q(x_t, u_t) = \text{const} + \frac{1}{2} \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \mathbf{Q}_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T \mathbf{q}_t$ ;
5    $u_t \leftarrow \arg \min_{u_t} Q(x_t, u_t) = \mathbf{K}_t x_t + \mathbf{k}_t$ ;
6    $\mathbf{K}_t = -\mathbf{Q}_{u_t, u_t}^{-1} \mathbf{Q}_{u_t, x_t}$ ;
7    $\mathbf{k}_t = -\mathbf{Q}_{u_t, u_t}^{-1} \mathbf{q}_{u_t}$ ;
8    $\mathbf{V}_t = \mathbf{Q}_{x_t, x_t} + \mathbf{Q}_{x_t, u_t} \mathbf{K}_t + \mathbf{K}_t^T \mathbf{Q}_{u_t, x_t} + \mathbf{K}_t^T \mathbf{Q}_{u_t, u_t} \mathbf{K}_t$ ;
9    $\mathbf{v}_t = \mathbf{q}_{x_t} + \mathbf{Q}_{x_t, u_t} \mathbf{k}_t + \mathbf{K}_t^T \mathbf{q}_{u_t} + \mathbf{K}_t^T \mathbf{Q}_{u_t, u_t} \mathbf{k}_t$ ;
10   $V(x_t) = \text{const} + \frac{1}{2} x_t^T \mathbf{V}_t x_t + x_t^T \mathbf{v}_t$ ;
11 end
```

---

We can generalize it to stochastic case, where system dynamic with a gaussian noise (control is still deterministic), because the expectation of gaussian is zero for linear and constant for quadratic cost ( $\mathbb{E}[x_{t+1}^T V x_{t+1}] = (Ax_t + Bu_t)^T V (Ax_t + Bu_t) + \text{tr}(VW)$ , so when minimizing it is ignored).

For the nonlinear case, we linearize it with reference point:

$$\begin{aligned} f(\mathbf{x}_t, \mathbf{u}_t) &\approx f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix} \\ c(\mathbf{x}_t, \mathbf{u}_t) &\approx c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix} \\ &\quad + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}^T \nabla_{\mathbf{x}_t, \mathbf{u}_t}^2 c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix} \end{aligned}$$

and use

$$\bar{f}(\delta \mathbf{x}_t, \delta \mathbf{u}_t) = \mathbf{F}_t \begin{bmatrix} \delta \mathbf{x}_t \\ \delta \mathbf{u}_t \end{bmatrix}, \quad \bar{c}(\delta \mathbf{x}_t, \delta \mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \delta \mathbf{x}_t \\ \delta \mathbf{u}_t \end{bmatrix}^T \mathbf{C}_t \begin{bmatrix} \delta \mathbf{x}_t \\ \delta \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \delta \mathbf{x}_t \\ \delta \mathbf{u}_t \end{bmatrix}^T \mathbf{c}_t$$

Now we can run LQR with it, this is called **iLQR**, we get  $u_t = K_t(x_t - \hat{x}_t) + k_t + \hat{u}_t$ , this is an approximation of Newton's method for solving the entire cost function over the horizon. And if we linearize the dynamic with second order information:

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \delta \mathbf{x}_t \\ \delta \mathbf{u}_t \end{bmatrix} + \frac{1}{2} \left( \nabla_{\mathbf{x}_t, \mathbf{u}_t}^2 f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \delta \mathbf{x}_t \\ \delta \mathbf{u}_t \end{bmatrix} \right) \begin{bmatrix} \delta \mathbf{x}_t \\ \delta \mathbf{u}_t \end{bmatrix}$$

it is called *DDP*.