



# Reinforcement Learning Notes

**Learn from trying!**

**Author:** occupymars

**Date:** June. 20, 2025

**Version:** 0.1

# Contents

<b>Chapter 1 Math of Reinforcement Learning</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 . . . . .	3
1.5 Bellman Optimality Equation . . . . .	3
1.6 Value Iteration and Policy Iteration . . . . .	6
1.7 Monte Carlo Method . . . . .	8
1.8 Stochastic Approximation . . . . .	10
1.9 Appendix: Measure-Theoretic Probability Theory . . . . .	11
<b>Chapter 2 From LQR to RL</b>	<b>12</b>
2.1 LQR and Value function . . . . .	12

# Chapter 1 Math of Reinforcement Learning

## 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 trajetory to goal.

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

**Remark** In this book, the reward to a state that is not forbidden and is not a boundary is set to 0. There are five actions: up, right, down, left, and stay, with  $a_1, \dots, a_5$ .

## 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 divide 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  $E_{s_1 \sim p(s_1 | s_0, a_0)}$  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

$$\begin{aligned} r^\pi(s) &= \sum_{a \in \mathcal{A}} \pi(a | s) \sum_r p(r | s, a) r \\ p^\pi(s' | s) &= \sum_{a \in \mathcal{A}} \pi(a | s) p(s' | s, a) \end{aligned} \quad (1.6)$$

rewirte 1.3 into the vector form:

$$V^\pi = r^\pi + \gamma P^\pi V^\pi \quad (1.7)$$

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

**Proof** Define the error as  $\delta_k = V_k - V^\pi$ , substitute  $V_{k+1} = \delta_{k+1} + V^\pi$  and  $V_k = \delta_k + V^\pi$  into the equation:

$$\delta_{k+1} + V^\pi = r^\pi + \gamma P^\pi (\delta_k + V^\pi)$$

Rearrange it:

$$\begin{aligned} \delta_{k+1} &= r^\pi + \gamma P^\pi V^\pi + \gamma P^\pi \delta_k - V^\pi \\ &= \gamma P^\pi V^\pi + r^\pi + \gamma P^\pi \delta_k - V^\pi \\ &= \gamma P^\pi \delta_k \end{aligned}$$

As a result,  $\delta_{k+1} = \gamma P^\pi \delta_k = (\gamma P^\pi)^2 \delta_{k-1} = \dots = (\gamma P^\pi)^{k+1} \delta_0$ . Since every entry of  $P^\pi$  is nonnegative and no greater than 1, and  $\gamma < 1$ , we have  $\|(\gamma P^\pi)^{k+1}\| \rightarrow 0$  as  $k \rightarrow \infty$ , and the error  $\|\delta_k\| \rightarrow 0$  as  $k \rightarrow \infty$ .

## 1.4 Action Value Function

Similarly to value function, **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:

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a | s) Q^\pi(s, a) \quad (1.8)$$

The action value function is given as:

$$\begin{aligned} Q^\pi(s_0, a_0) &= r(s_0, a_0) + \mathbb{E}_{a_t \sim \pi(s_t)} [\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t)] \\ &= r(s_0, a_0) + \gamma \mathbb{E}_{a_t \sim \pi(s_t)} [\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t)] \\ &= r(s_0, a_0) + \gamma \mathbb{E}_{a_t \sim \pi(s_t)} [R(\tau')] \\ &= r(s_0, a_0) + \gamma \mathbb{E}_{s_1 \sim p(s_1 | s_0, a_0)} [\mathbb{E}_{a_t \sim \pi(s_t)} [R(\tau') | s_1]] \\ &= r(s_0, a_0) + \gamma \mathbb{E}_{s_1 \sim p(s_1 | s_0, a_0)} [V^\pi(s_1)] \\ &= r(s_0, a_0) + \gamma \mathbb{E}_{s_1 \sim p(s_1 | s_0, a_0)} [\sum_{a_1 \in \mathcal{A}} \pi(a_1 | s_1) Q^\pi(s_1, a_1)] \end{aligned} \quad (1.9)$$

Then the bellman equation of action value is:

$$\begin{aligned} Q^\pi(s, a) &= r(s, a) + \gamma \sum_{s'} p(s' | s, a) V^\pi(s') \\ &= r(s, a) + \gamma \sum_{s'} p(s' | s, a) \sum_{a' \in \mathcal{A}} \pi(a' | s') Q^\pi(s', a') \end{aligned} \quad (1.10)$$

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 1.10 into vector form:

$$Q^\pi = \tilde{r} + \gamma P \Pi Q^\pi \quad (1.11)$$

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

## 1.5 Bellman Optimality Equation

### Definition 1.1 (Optimal Policy)

If  $V^{\pi_1}(s) \geq V^{\pi_2}(s), \forall s \in \mathcal{S}$ , then  $\pi_1$  is better than  $\pi_2$ , if  $\pi_1$  is better than all other policies, it is called **Optimal Policy**  $\pi^*$ .



**Bellman Optimality Equation** (BOE) is given by:

$$\begin{aligned} V(s) &= \max_{\pi(s) \in \Pi(s)} \sum_{a \in \mathcal{A}} \pi(a | s) (\sum_r p(r | s, a) r + \gamma \sum_{s'} p(s' | s, a) V(s')) \\ &= \max_{\pi(s) \in \Pi(s)} \sum_{a \in \mathcal{A}} \pi(a | s) Q(s, a) \end{aligned} \quad (1.12)$$

There are two unknowns in the equation,  $V(s)$  and  $\pi(a | s)$ , we can first consider the right hand side, to compute the  $\pi(a | s)$ .

**Example 1.1** Consider  $\sum_1^3 c_i q_i$ , where  $c_1 + c_2 + c_3 = 1$  and they are all greater than 0, without loss of generality, we can assume  $q_3 \geq q_1, q_2$ , then the maximum is achieved when  $c_3 = 1, c_1 = 0, c_2 = 0$ . This is because:

$$q_3 = (c_1 + c_2 + c_3) q_3 = c_1 q_3 + c_2 q_3 + c_3 q_3 \geq c_1 q_1 + c_2 q_2 + c_3 q_3$$

Inspired by the example, since  $\sum_a \pi(a | s) = 1$ , we have:

$$\sum_{a \in \mathcal{A}} \pi(a | s) Q(s, a) \leq \sum_{a \in \mathcal{A}} \pi(a | s) \max_{a \in \mathcal{A}} Q(s, a) = \max_{a \in \mathcal{A}} Q(s, a)$$

where the equality is achieved when

$$\pi(a | s) = \begin{cases} 1, & a = a^*, \\ 0, & a \neq a^*. \end{cases}$$

here  $a^* = \arg \max_{a \in \mathcal{A}} Q(s, a)$ .

Then the matrix form of BOE is:

$$V = \max_{\pi \in \Pi} (r^\pi + \gamma P^\pi V) = f(V)$$

the  $r^\pi$  and  $P^\pi$  are the same before in normal Bellman equation 1.6.

In order to solve this nonlinear equation, we first need to introduce **Contraction Mapping** theorem or Fixed Point theorem:

### Definition 1.2 (Contraction Mapping)

Consider function  $f(x)$ , where  $x \in \mathbb{R}^d$  and  $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ . A point  $x^*$  is called a fixed point if  $f(x^*) = x^*$ , and the function is a contraction mapping if there exists  $\gamma \in (0, 1)$  such that:

$$\|f(x_1) - f(x_2)\| \leq \gamma \|x_1 - x_2\|, \forall x_1, x_2 \in \mathbb{R}^d$$



The relation between a fixed point and the contraction property is characterized by:

### Theorem 1.1 (Banach's Fixed Point Theorem)

For any equation that has the form  $x = f(x)$  where  $x$  and  $f(x)$  are real vectors, if  $f$  is a contraction mapping, then:

1. *Existence:* There exists a fixed point  $x^*$  such that  $f(x^*) = x^*$ .
2. *Uniqueness:* There exists a unique fixed point  $x^*$  such that  $f(x^*) = x^*$ .
3. *Algorithm:* For any initial point  $x_0$ , the sequence  $x_{k+1} = f(x_k)$  converges to the fixed point  $x^*$ . Moreover, the convergence rate is exponentially fast.



The proof of the theorem can be found in the book, it is based on Cauchy sequence. Then we need to show the right hand side of the BOE is a contraction mapping:

### Theorem 1.2 (Contraction Property of right-hand side of BOE)

For any  $V_1, V_2 \in \mathbb{R}^{|\mathcal{S}|}$ , it holds that:

$$\|f(V_1) - f(V_2)\|_\infty \leq \gamma \|V_1 - V_2\|_\infty$$

where  $\gamma \in (0, 1)$  is the discount factor,  $\|\cdot\|_\infty$  is the maximum norm, which is the maximum absolute value of the elements of a vector.



### Proof

$$f(V_1) = \max_{\pi} (r^\pi + \gamma P^\pi V_1) = r^{\pi_1^*} + \gamma P^{\pi_1^*} V_1 \geq r^{\pi_2^*} + \gamma P^{\pi_2^*} V_1$$

$$f(V_2) = \max_{\pi} (r^\pi + \gamma P^\pi V_2) = r^{\pi_2^*} + \gamma P^{\pi_2^*} V_2 \geq r^{\pi_1^*} + \gamma P^{\pi_1^*} V_2$$

where  $\geq$  is elementwise comparison, as a result:

$$\begin{aligned} f(V_1) - f(V_2) &= (r^{\pi_1^*} - r^{\pi_2^*}) + \gamma P^{\pi_1^*} V_1 - \gamma P^{\pi_2^*} V_2 \\ &\leq (r^{\pi_1^*} - r^{\pi_1^*}) + \gamma P^{\pi_1^*} V_1 - \gamma P^{\pi_1^*} V_2 \\ &= \gamma P^{\pi_1^*} (V_1 - V_2) \end{aligned}$$

similarly we can get  $f(V_2) - f(V_1) \leq \gamma P^{\pi_2^*} (V_2 - V_1)$ , so we have:

$$\gamma P^{\pi_2^*} (V_1 - V_2) \leq f(V_1) - f(V_2) \leq \gamma P^{\pi_1^*} (V_1 - V_2)$$

define

$$z = \max \{ |\gamma P^{\pi_1^*} (V_1 - V_2)|, |\gamma P^{\pi_2^*} (V_1 - V_2)| \} \in \mathbb{R}^{|\mathcal{S}|}$$

all the operations are elementwise,  $z \geq 0$ , then we have:

$$-z \leq \gamma P^{\pi_2^*} (V_1 - V_2) \leq f(V_1) - f(V_2) \leq \gamma P^{\pi_1^*} (V_1 - V_2) \leq z$$

which implies:

$$|f(V_1) - f(V_2)| \leq z$$



it then follows that:

$$\|f(V_1) - f(V_2)\|_\infty \leq \|z\|_\infty \quad (1.13)$$

suppose  $z_i, p_i^T, q_i^T$  are  $i$ th entry of  $z, P^{\pi_1^*}, P^{\pi_2^*}$ , then:

$$z_i = \max \{ |\gamma p_i^T (V_1 - V_2)|, |\gamma q_i^T (V_1 - V_2)| \}$$

since  $p_i$  sums up to 1 and nonnegative, we have:

$$|p_i^T (V_1 - V_2)| \leq p_i^T |V_1 - V_2| \leq \|V_1 - V_2\|_\infty$$

similarly we have  $|q_i^T (V_1 - V_2)| \leq \|V_1 - V_2\|_\infty$ , therefore  $z_i \leq \gamma \|V_1 - V_2\|_\infty$ , and hence

$$\|z\|_\infty = \max_i |z_i| \leq \gamma \|V_1 - V_2\|_\infty$$

Substitute it back to 1.13, we have:

$$\|f(V_1) - f(V_2)\|_\infty \leq \gamma \|V_1 - V_2\|_\infty$$

Then we can use this to solve an optimal policy from the BOE. Since  $V^* = \max_{\pi \in \Pi} (r^\pi + \gamma P^\pi V^*)$ , so it is clearly a fixed point.

### Theorem 1.3 (Existence, Uniqueness and Algorithm of Optimal Policy)

The optimal policy  $V^*$  exists and is unique, and the sequence  $V_{k+1} = f(V_k)$  converges to the optimal policy  $V^*$  exponentially fast given any initial guess  $V_0$  with the iteration algorithm:

$$V_{k+1} = f(V_k) = \max_{\pi \in \Pi} (r^\pi + \gamma P^\pi V_k)$$



The proof follows the proof of the contraction mapping theorem, and the iteration algorithm is called **Value Iteration**. Once we have the optimal value function  $V^*$ , we can get the optimal policy  $\pi^*$  by:

$$\pi^* = \arg \max_{\pi \in \Pi} (r^\pi + \gamma P^\pi V^*) \quad (1.14)$$

substitute this into the BOE yields  $V^* = r^{\pi^*} + \gamma P^{\pi^*} V^*$ , which is the optimal value function.

### Theorem 1.4 (Optimality of Value Function and Policy)

The solution  $V^*$  and the policy  $\pi^*$  are optimal, i.e., for any other policy  $\pi \in \Pi$ , it holds that:

$$V^* = V^{\pi^*} \geq V^\pi$$



### Proof

$$V^* - V^\pi = r^{\pi^*} + \gamma P^{\pi^*} V^* - r^\pi - \gamma P^\pi V^\pi \geq r^{\pi^*} + \gamma P^{\pi^*} V^* - r^\pi - \gamma P^{\pi^*} V^\pi = \gamma P^{\pi^*} (V^* - V^\pi)$$

iteratively we have  $V^* - V^\pi \geq \lim_{n \rightarrow \infty} (\gamma P^{\pi^*})^n (V^* - V^\pi)$ .

### Theorem 1.5 (Greedy optimal policy)

For any  $s \in \mathcal{S}$ , the deterministic greedy policy:

$$\pi(a | s) = \begin{cases} 1, & a = a^*, \\ 0, & a \neq a^*. \end{cases} \quad (1.15)$$

is an optimal policy, where  $a^* = \arg \max_{a \in \mathcal{A}} Q^*(s, a)$ , where

$$Q^*(s, a) = \sum_{r \in \mathcal{R}} p(r | s, a) r + \gamma \sum_{s'} p(s' | s, a) V^*(s')$$

remember that even though  $V^*$  is unique, the optimal policy may not be unique, and there always exists a greedy optimal policy.




We can talk about the impact of the reward values:

**Theorem 1.6 (Optimal policy invariance)**

If every reward  $r \in \mathcal{R}$  is changed by an affine transformation to  $\alpha r + \beta$ , where  $\alpha, \beta \in \mathbb{R}$  and  $\alpha > 0$ , then the corresponding optimal state value  $V'$  is also an affine transformation of  $V^*$ :

$$V' = \alpha V^* + \frac{\beta}{1 - \gamma} \mathbf{1}$$

Consequently, the optimal policy derived from  $V'$  is invariant to the affine transformation of the reward values. 

And with the discount factor  $\gamma$ , the optimal policy will not take any meaningless detour.

## 1.6 Value Iteration and Policy Iteration

The algorithm of *Value Iteration* is:

$$V_{k+1} = \max_{\pi \in \Pi} (r^\pi + \gamma P^\pi V_k)$$

and there are two steps in one iteration, first one is called *Policy Update*:

$$\pi_{k+1} = \arg \max_{\pi \in \Pi} (r^\pi + \gamma P^\pi V_k)$$

the second one is called *Value Update*:

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

With the elementwise form, the policy update is (if there are same actions that takes the maximum, we can choose any of them):

$$\pi_{k+1}(s) = \arg \max_{\pi(s) \in \Pi(s)} \sum_{a \in \mathcal{A}} \pi(a | s) \left( \sum_r p(r | s, a) r + \gamma \sum_{s'} p(s' | s, a) V_k(s') \right)$$

Then the value update is:

$$V_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi_{k+1}(a | s) \left( \sum_r p(r | s, a) r + \gamma \sum_{s'} p(s' | s, a) V_k(s') \right)$$

We can use the greedy deterministic policy, which is then:

$$V_{k+1}(s) = \max_a Q_k(s, a)$$

We should know that  $V_k$  is not a state value though it converges to the optimal state value, it is not ensured to satisfy the Bellman equation. So the  $Q_k$  is also not a action value, they are all intermediate values.

The algorithm of *Policy Iteration* has also two steps, first one is called *Policy Evaluation*:

$$V^{\pi_k} = r^{\pi_k} + \gamma P^{\pi_k} V^{\pi_k}$$


second one is called *Policy Improvement*:

$$\pi_{k+1} = \arg \max_{\pi \in \Pi} (r^{\pi_k} + \gamma P^{\pi_k} V^{\pi_k})$$

Here comes the first question, how to solve the policy evaluation? We can use the iterative method introduced in 1.3, and this results an iteration algorithm inside an iteration algorithm. We will not do infinite iteration here, so the  $V^{\pi_k}$  will not be the exact solution, would this cause problem? No, see the truncated policy iteration algorithm.

And the second question, why  $\pi_{k+1}$  is better than  $\pi_k$ ?

**Lemma 1.1 (Policy Improvement)**

If  $\pi_{k+1} = \arg \max_{\pi \in \Pi} (r^{\pi_k} + \gamma P^{\pi_k} V^{\pi_k})$ , then  $V^{\pi_{k+1}}(s) \geq V^{\pi_k}(s), \forall s \in \mathcal{S}$ . 

**Proof** We know that:

$$r^{\pi_{k+1}} + \gamma P^{\pi_{k+1}} V^{\pi_{k+1}} \geq r^{\pi_k} + \gamma P^{\pi_k} V^{\pi_k}$$



Then

$$\begin{aligned}
 V^{\pi_k} - V^{\pi_{k+1}} &= r^{\pi_k} + \gamma P^{\pi_k} V^{\pi_k} - (r^{\pi_{k+1}} + \gamma P^{\pi_{k+1}} V^{\pi_{k+1}}) \\
 &\leq r^{\pi_{k+1}} + \gamma P^{\pi_{k+1}} V^{\pi_k} - (r^{\pi_{k+1}} + \gamma P^{\pi_{k+1}} V^{\pi_{k+1}}) \\
 &\leq \gamma P^{\pi_{k+1}} (V^{\pi_k} - V^{\pi_{k+1}})
 \end{aligned}$$

again iteratively we have:

$$V^{\pi_k} - V^{\pi_{k+1}} \leq \lim_{n \rightarrow \infty} (\gamma P^{\pi_{k+1}})^n (V^{\pi_k} - V^{\pi_{k+1}}) = 0$$

### Theorem 1.7 (Convergence of policy iteration)

The state value sequence  $\{V^{\pi_k}\}_{k=0}^{\infty}$  converges to the optimal state value  $V^*$ , and the policy sequence  $\{\pi_k\}_{k=0}^{\infty}$  converges to the optimal policy  $\pi^*$  in policy iteration algorithm.

**Proof** We introduce another sequence  $\{V_k\}_{k=0}^{\infty}$  generated by:

$$V_{k+1} = f(V_k) = \max_{\pi} (r^{\pi} + \gamma P^{\pi} V_k)$$

which is exactly the value iteration algorithm, and we know that  $V_k \rightarrow V^*$  as  $k \rightarrow \infty$ . For  $k = 0$ , we can always find a  $V_0$  such that  $V_0 \leq V^{\pi_0}$ , then we use induction: for  $k \geq 0$ , if  $V_k \leq V^{\pi_k}$ , then for  $k + 1$ :

$$\begin{aligned}
 V^{\pi_{k+1}} - V_{k+1} &= (r^{\pi_{k+1}} + \gamma P^{\pi_{k+1}} V^{\pi_{k+1}}) - \max_{\pi} (r^{\pi} + \gamma P^{\pi} V_k) \\
 &\geq (r^{\pi_{k+1}} + \gamma P^{\pi_{k+1}} V_k) - \max_{\pi} (r^{\pi} + \gamma P^{\pi} V_k) \\
 &\quad (\text{because } V^{\pi_{k+1}} \geq V_k \text{ by Lemma 1.1 and } P^{\pi_{k+1}} \geq 0) \\
 &= (r^{\pi'_k} + \gamma P^{\pi'_k} V_k) - (r^{\pi'_k} + \gamma P^{\pi'_k} V_k) \\
 &\quad (\text{suppose } \pi'_k = \arg \max_{\pi} (r^{\pi} + \gamma P^{\pi} V_k)) \\
 &\geq (r^{\pi'_k} + \gamma P^{\pi'_k} V_k) - (r^{\pi'_k} + \gamma P^{\pi'_k} V_k) \\
 &\quad (\text{because } \pi_{k+1} = \arg \max_{\pi} (r^{\pi} + \gamma P^{\pi} V_k)) \\
 &= \gamma P^{\pi'_k} (V_{k+1} - V_k) \geq 0
 \end{aligned}$$

Since  $V_k$  converges to  $V^*$ ,  $V^{\pi_k}$  is also converges to  $V^*$ .

Then the elementwise form of the policy iteration algorithm is, policy evaluation:

$$V_{(j+1)}^{\pi_k}(s) = \sum_{a \in \mathcal{A}} \pi_{k+1}(a | s) \left( \sum_r p(r | s, a) r + \gamma \sum_{s'} p(s' | s, a) V_{(j)}^{\pi_k}(s') \right)$$

and policy improvement:

$$\pi_{k+1}(s) = \arg \max_{\pi(s) \in \Pi(s)} \sum_{a \in \mathcal{A}} \pi(a | s) \left( \sum_r p(r | s, a) r + \gamma \sum_{s'} p(s' | s, a) V^{\pi_k}(s') \right)$$

Next we introduce **Truncated Policy Iteration** algorithm, which is a combination of value iteration and policy iteration. We will see that the value iteration and policy iteration algorithms are two special cases of the truncated policy iteration algorithm.

If we start from  $V_0^{\pi_1} = V_0$ , we get:

$$\begin{array}{lll}
 & & V_{\pi_1}^{(0)} = V_0 \\
 \text{value iteration} & \leftarrow V_1 \leftarrow & V_{(1)}^{\pi_1} = r^{\pi_1} + \gamma P^{\pi_1} V_{(0)}^{\pi_1} \\
 & & V_{(2)}^{\pi_1} = r^{\pi_1} + \gamma P^{\pi_1} V_{(1)}^{\pi_1} \\
 & & \vdots \\
 \text{truncated policy iteration} & \leftarrow \bar{V}_1 \leftarrow & V_{(j)}^{\pi_1} = r^{\pi_1} + \gamma P^{\pi_1} V_{(j-1)}^{\pi_1} \\
 & & \vdots \\
 \text{policy iteration} & \leftarrow V^{\pi_1} \leftarrow & V_{(\infty)}^{\pi_1} = r^{\pi_1} + \gamma P^{\pi_1} V_{(\infty)}^{\pi_1}
 \end{array}$$

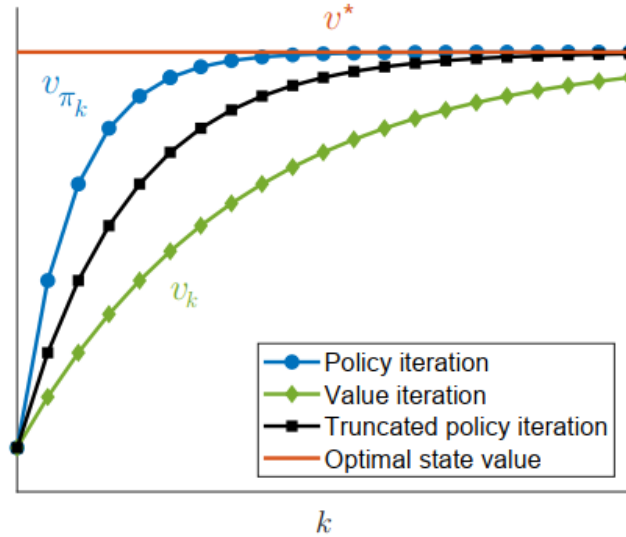


Figure 1.1: 3 iteration methods

**Proposition 1.1 (Value Improvement)**

In the policy evaluation step if the initial guess is selected as  $V_{(0)}^{\pi_k} = V^{\pi_{k-1}}$ , then it holds that:

$$V_{(j+1)}^{\pi_k} \geq V_{(j)}^{\pi_k}$$

**Proof**

$$V_{(j+1)}^{\pi_k} - V_{(j)}^{\pi_k} = \gamma P^{\pi_k} (V_{(j)}^{\pi_k} - V_{(j-1)}^{\pi_k}) = \dots = \gamma^j P^{\pi_k} (V_{(1)}^{\pi_k} - V_{(0)}^{\pi_k})$$

we have

$$V_{(1)}^{\pi_k} = r^{\pi_k} + \gamma P^{\pi_k} V^{\pi_{k-1}} \geq r^{\pi_{k-1}} + \gamma P^{\pi_{k-1}} V^{\pi_{k-1}} = V^{\pi_{k-1}} = V_{(0)}^{\pi_k}$$

where the inequality is due to  $\pi_k = \arg \max_{\pi} (r^{\pi} + \gamma P^{\pi} V^{\pi_{k-1}})$ , substitute  $V_{(1)}^{\pi_k} \geq V_{(0)}^{\pi_k}$  into the first equation, we have  $V_{(j+1)}^{\pi_k} \geq V_{(j)}^{\pi_k}$ .

But we have to note that this is based on the assumption that the initial guess is  $V_{(0)}^{\pi_k} = V^{\pi_{k-1}}$ , and  $V^{\pi_{k-1}}$  is not always available, like in the truncated policy iteration algorithm, we do not do the iteration until convergence, but only do a few steps, so the initial value is approximation. And in Deep Reinforcement Learning, we do not have the exact value function, even if we have it may not be transferable or meaningful.

## 1.7 Monte Carlo Method

If we sample from a independent and identically distributed (i.i.d.) distribution, we can use the Monte Carlo method to do mean estimate, as value function and action value function are all mean estimates.

In the policy iteration, the action value lies in the core of these two steps, we first use Bellman equation to compute the value function, then use it to compute action value, and then update the policy. If we do not have a model, then the definition of action value is:

$$Q^{\pi_k}(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a] = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s, A_t = a]$$

we use  $R$  to denote the stochastic reward, and  $G_t$  is the return at time  $t$ .

If we run the policy for  $n$  episodes then we get:

$$Q^{\pi_k}(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a] \approx \frac{1}{n} \sum_{i=1}^n G_{(i)}^{\pi_k}(s, a) \quad (1.16)$$

instead of calculation of value function, we directly get action value from samples, this is the simplest Monte Carlo method (we call it MC Basic).

In the example of the book, we can see that if the length of the episodes are too short, it will get zero action value, because it can not get to the target. This is caused by the *Sparse Reward*.

We next show how to efficiently use the samples. Every time a state-action pair appears in an episodes, it is called a *visit*. Given an episodes:

$$s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots$$

1. Initial visit, only the first pair is evaluated, like in the MC Basic, which is not efficient.
2. First-visit, only use the samples to estimate the action value of the first visit.
3. Every visit, use the samples to estimate the action value of every visit, which is more efficient, but may cause bias (because they are subsequence, there are correlation).

Then we consider how to efficiently update the policy.

1. In the policy evaluation step, to collect all the episodes starting from the same state-action pair and then approximate the action value using the average return of these episodes, which is in MC Basic. The drawback of this strategy is that the agent must wait until all the episodes have been collected before the estimate can be updated.
2. Use the return of a single episode to approximate the corresponding action value. In this way, we can immediately obtain a rough estimate when we receive an episode. Then, the policy can be improved in an episode-by-episode fashion. In fact, this strategy falls into the scope of *Generalized Policy Iteration* introduced in the last chapter, we can still update the policy even if the value estimate is not sufficiently accurate.

So the new algorithm is called *MC Exploring Starts*, based on the exploring starts condition, which means that the agent can start from any state and take any action (MC Basic assume this too).

---

**Algorithm 1:** MC Exploring Starts (efficient variant of MC Basic)

---

**Input:** Initial policy  $\pi_0(a|s)$ ,  $Q(s, a)$ ,  $\text{Returns}(s, a) = 0$ ,  $\text{Num}(s, a) = 0, \forall (s, a)$

```

1 foreach episode do
2   Episode generation: Select  $(s_0, a_0)$  (exploring-starts), then generate  $s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T$ 
   following  $\pi$ 
3    $g \leftarrow 0$ 
4   for  $t = T - 1, T - 2, \dots, 0$  do
5      $g \leftarrow \gamma g + r_{t+1}$ 
6      $\text{Returns}(s_t, a_t) \leftarrow \text{Returns}(s_t, a_t) + g$ 
7      $\text{Num}(s_t, a_t) \leftarrow \text{Num}(s_t, a_t) + 1$ 
8      $Q(s_t, a_t) \leftarrow \frac{\text{Returns}(s_t, a_t)}{\text{Num}(s_t, a_t)}$ 
9      $\pi(a|s_t) \leftarrow \begin{cases} 1 & \text{if } a = \arg \max_a Q(s_t, a), \\ 0 & \text{otherwise} \end{cases}$ 
10  end
11 end
```

---

We can see that the algorithm starts from the last state of the episode, and then update the action value and policy in a **reverse order**.

However, the exploring starts condition is not always satisfied, for example in the real world involving physical interactions, we need to remove this condition. A policy is **soft** if it has a positive probability of taking any action at any state, with a soft policy, a single episode that is sufficiently long can visit every state-action pair many times. Thus we do not need the exploring starts condition.

One type of soft policy is  *$\epsilon$ -greedy Policy*:

$$\pi(a | s) = \begin{cases} 1 - \frac{\epsilon}{|\mathcal{A}(s)|} (|\mathcal{A}(s)| - 1), & \text{for the greedy action,} \\ \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{for the other actions.} \end{cases} \quad (1.17)$$

Where  $\pi \in \Pi_\epsilon$  is the set of all  $\epsilon$ -greedy policies, and  $|\mathcal{A}(s)|$  is the number of actions available at state  $s$ . The probability of taking the greedy action is always greater than that of taking any other action.

With this we can change the policy improvement step:

$$\pi(a | s) = \begin{cases} 1 - \frac{\epsilon}{|\mathcal{A}(s)|} (|\mathcal{A}(s)| - 1), & a = a^*, \\ \frac{\epsilon}{|\mathcal{A}(s)|}, & a \neq a^*. \end{cases} \quad (1.18)$$

So the algorithm is:

---

**Algorithm 2:** MC  $\epsilon$ -greedy (a variant of MC Exploring Starts)

---

**Input:** Initial policy  $\pi_0(a|s)$ ,  $Q(s, a)$ ,  $\text{Returns}(s, a) = 0$ ,  $\text{Num}(s, a) = 0, \forall (s, a). \epsilon \in (0, 1]$

```

1 foreach episode do
2   Episode generation: Select  $(s_0, a_0)$ , then generate  $s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T$  following  $\pi$ 
3    $g \leftarrow 0$ 
4   for  $t = T - 1, T - 2, \dots, 0$  do
5      $g \leftarrow \gamma g + r_{t+1}$ 
6      $\text{Returns}(s_t, a_t) \leftarrow \text{Returns}(s_t, a_t) + g$ 
7      $\text{Num}(s_t, a_t) \leftarrow \text{Num}(s_t, a_t) + 1$ 
8      $Q(s_t, a_t) \leftarrow \frac{\text{Returns}(s_t, a_t)}{\text{Num}(s_t, a_t)}$ 
9      $\pi(a|s_t) \leftarrow \begin{cases} 1 - \frac{\epsilon}{|\mathcal{A}(s)|} (|\mathcal{A}(s)| - 1) & a = a^*, \\ \frac{\epsilon}{|\mathcal{A}(s)|} & a \neq a^*. \end{cases}$ 
10  end
11 end
```

---

The policy is optimal in  $\Pi_\epsilon$ , not in  $\Pi$ , so it is not guaranteed to be optimal, but it is still a good policy (if  $\epsilon$  is sufficiently small).

## 1.8 Stochastic Approximation

For now, we only learned non-incremental algorithms, which may take a logn time to compute, not good for online learning. Suppose  $\omega_k = \frac{1}{k-1} \sum_{i=1}^{k-1} x_i, k = 2, 3, \dots$ , then:

$$\omega_{k+1} = \frac{1}{k} \sum_{i=1}^k x_i = \frac{k-1}{k} \omega_k + \frac{1}{k} x_k$$

this update the mean estimate incrementally. We use

$$\omega_{k+1} = \omega_k - \alpha_k (\omega_k - x_k) \quad (1.19)$$

$\alpha_k$  may not be given, though above we give it as  $\frac{1}{k}$ .

Given a noisy observation of  $g(\omega)$ :

$$\tilde{g}(\omega, \eta) = g(\omega) + \eta$$

where  $\eta$  is a noise, we can use the stochastic approximation method **Robbins-Monro** algorithm to update the estimate of  $\omega$ :


$$\omega_{k+1} = \omega_k - \alpha_k \tilde{g}(\omega_k, \eta_k), \alpha_k > 0 \quad (1.20)$$

Why this algorithm converge?

**Theorem 1.8 (Robbins-Monro theorem)**

If

1.  $0 \leq c_1 \leq \nabla_\omega g(\omega) \leq c_2 < \infty, \forall \omega$ ; So  $g$  is monotonic and bounded;
2.  $\sum_{k=1}^{\infty} \alpha_k = \infty$  and  $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$ ; It should converge to zero, but not too fast, one common choice is  $\alpha_k = \frac{1}{k}$ ;
3.  $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$  and  $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] < \infty$ ;

where  $\mathcal{H}_k = \{\omega_k, \omega_{k-1}, \dots\}$ , then  $\omega_k$  converges to the root of  $g(\omega)$  almost surely, i.e.,  $\lim_{k \rightarrow \infty} \omega_k = \omega^*$ , where  $g(\omega^*) = 0$ . 

Then let  $g(\omega) = \omega - \mathbb{E}[X]$ ,  $\tilde{g}(\omega, \eta) = \omega - x + \eta$ ,  $\eta = \mathbb{E}[X] - x$ , we can use the Robbins-Monro algorithm to prove that with careful choice of  $\alpha_k$ , the estimate  $\omega_k$  converges to the mean  $\mathbb{E}[X]$  almost surely.

#### Theorem 1.9 (Dvoretzky's Theorem)


Consider a stochastic sequence:

$$\Delta_{k+1} = (1 - \alpha_k)\Delta_k + \beta_k \eta_k$$

where  $\alpha_k, \beta_k \geq 0$ , then  $\Delta_k$  converges to 0 almost surely, if:

1.  $\sum_{k=1}^{\infty} \alpha_k = \infty$  and  $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$ , and  $\sum_{k=1}^{\infty} \beta_k^2 < \infty$  uniformly almost surely;
2.  $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$  and  $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] < \infty$  almost surely.

where  $\mathcal{H}_k = \{\Delta_k, \Delta_{k-1}, \dots, \eta_{k-1}, \dots, \alpha_{k-1}, \dots, \beta_{k-1}, \dots\}$ .

In RM,  $\alpha_k, \beta_k$  are deterministic, but in Dvoretzky's theorem, they can be stochastic, and is useful in cases they are function of  $\Delta_k$ . 


## 1.9 Appendix: Measure-Theoretic Probability Theory

A **Probability Triple** is a tuple  $(\Omega, \mathcal{F}, P)$ , where:

- $\Omega$  is a sample space, which is a set of all possible outcomes;
- $\mathcal{F}$  is a  $\sigma$ -algebra, which is a collection of subsets of  $\Omega$  that includes the empty set and  $\Omega$ , and is closed under complementation and countable unions; An element in  $\mathcal{F}$  is called an event, denoted as  $A$ . An elementary event refers to a single outcome in  $\Omega$ .
- $P$  is a probability measure, which assigns a probability to each event in  $\mathcal{F}$  such that  $P(\Omega) = 1$  and  $P(\emptyset) = 0$ .

#### Definition 1.3 ( $\sigma$ -algebra)

A  **$\sigma$ -algebra**  $\mathcal{F}$  on a set  $\Omega$  is a collection of subsets of  $\Omega$  that satisfies the following properties:

1.  $\emptyset \in \mathcal{F}$  and  $\Omega \in \mathcal{F}$ ;
2. If  $A \in \mathcal{F}$ , then  $A^c = \Omega \setminus A \in \mathcal{F}$  (closed under complementation);
3. If  $A_1, A_2, A_3, \dots \in \mathcal{F}$ , then  $\bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$  (closed under countable unions). 

A **Random Variable** is a measurable function  $X(\omega) : \Omega \rightarrow \mathbb{R}$  that maps outcomes in  $\Omega$  to real numbers. Not all the mappings are random variables, the formal definition is:

$$A = \{\omega \in \Omega \mid X(\omega) \leq x\} \in \mathcal{F}, \forall x \in \mathbb{R}$$

This definition indicates that  $X$  is a random variable only if  $X(\omega) \leq x$  is an event in  $\mathcal{F}$ .

## 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 3:** 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*.