Reinforcement Learning Notes

Learn from trying!

Author: occupymars

Date: June. 20, 2025

Version: 0.1

Contents

Chapter 1 Math of Reinforcement Learning

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 $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ to denote this, $T(s, a, s') = p(s' \mid s, a)$ is the probability of reaching s' given s and a. For $\forall s \in \mathcal{S}$ and $\forall a \in \mathcal{A}$, $\sum_{s' \in S} T(s, a, s') = 1$.

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

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

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

$$p(\tau) = \pi(a_0 \mid s_0) \cdot p(s_1 \mid s_0, a_0) \cdot \pi(a_1 \mid s_1) \cdot p(s_2 \mid s_1, a_1) \cdots$$

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 γ, 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 $a_1, \cdots, a_5$.

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:

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

If we divede the trajectory into two parts, s_0 and τ', 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:

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

we get:

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

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_**:

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

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

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

We can also get

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

combined we get

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

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

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \sum_{s' \in \mathcal{S}} p(s' \mid s,a) \left[r(s') + \gamma V^\pi(s') \right] \tag{1.5}$$

Let

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

rewirte 1.3 into the vector form:

$$V^\pi = r^\pi + \gamma P^\pi V^\pi \tag{1.7}$$

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

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

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

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:

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

The action value function is given as:

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

Then the bellman equation of action value is:

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

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

Rewrite 1.10 into vector form:

$$Q^\pi = \tilde{r} + \gamma P \Pi Q^\pi \tag{1.11}$$

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

1.5 Bellman Optimality Equation

> **Definition 1.1 (Optimal Policy)**
>
> *If $V^{\pi_1}(s) \geq V^{\pi_2}(s), \forall s \in \mathcal{S}$, than π_1 is better than π_2, if π_1 is better than all other policies, it is called **Optimal Policy** π^*.* ♣

Bellman Optimality Equation (BOE) is given by:

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

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

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 beacuse:

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

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

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

where the equality is achieved when

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

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^π and P^π 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 $f : \mathbb{R}^d \to \mathbb{R}^d$. 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:*
>
> $$\|f(x_1) - f(x_2)\| \le \gamma \|x_1 - x_2\|, \forall x_1, x_2 \in \mathbb{R}^d$$
>
> ♣

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, than:*
>
> 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 Cauthy 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:*
>
> $$\|f(V_1) - f(V_2)\|_\infty \le \gamma \|V_1 - V_2\|_\infty$$
>
> *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

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

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

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

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

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

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

define

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

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

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

which imlies:

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

it then follows that:

$$\|f(V_1) - f(V_2)\|_\infty \le \|z\|_\infty \tag{1.13}$$

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

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

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

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

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

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

Substitute it back to 1.13, we have:

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

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 π^* by:

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

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 π^* are optimal, i.e., for any other policy $\pi \in \Pi$, it holds that:

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

♡

Proof

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

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

Theorem 1.5 (Greedy optimal policy)

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

$$\pi(a \mid s) = \begin{cases} 1, & a = a^*, \\ 0, & a \ne a^*. \end{cases} \tag{1.15}$$

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

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

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 invaraince)**
>
> *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 γ, 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):
$$\pi_{k+1}(s) = \arg\max_{\pi(s) \in \Pi(s)} \sum_{a \in \mathcal{A}} \pi(a \mid s)\Big(\sum_r p(r \mid s, a)r + \gamma \sum_{s'} p(s' \mid s, a)V_k(s')\Big)$$

Then the value update is:
$$V_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi_{k+1}(a \mid s)\Big(\sum_r p(r \mid s, a)r + \gamma \sum_{s'} p(s' \mid s, a)V_k(s')\Big)$$
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^{π_k} will not be the exact solution, would this cause problem? No, see the truncated policy iteration algorithm.

And the second question, why π_{k+1} is better than π_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:

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

> **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 π^* 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 $V_k \to V^*$ as $k \to \infty$. 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 $k+1$:

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

Since V_k converges to V^*, V^{π_k} is also converges to V^*.

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

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

and policy improvement:

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

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:

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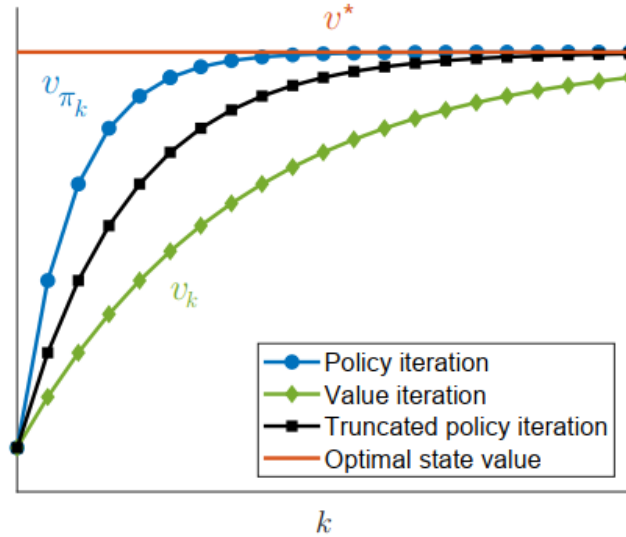\end{aligned}
$$

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

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

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 tranferable 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 ans 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, than the definition of action value is:

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

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 than we get:

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

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 corrleation).

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 $\pi_0(a|s)$, $Q(s,a)$, Returns$(s,a) = 0$, 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 π

3     $g \leftarrow 0$

4   **for** $t = T-1, T-2, \dots, 0$ **do**

5         $g \leftarrow \gamma g + r_{t+1}$

6   Returns$(s_t, a_t) \leftarrow$ Returns$(s_t, a_t) + g$

7   Num$(s_t, a_t) \leftarrow$ 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 ϵ-***greedy Policy***:

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

Where $\pi \in \Pi_\epsilon$ is the set of all ϵ-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:

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

So the algorithm is:

Algorithm 2: MC ϵ-greedy (a variant of MC Exploring Starts)

Input: Initial policy $\pi_0(a|s)$, $Q(s,a)$, Returns$(s,a) = 0$, 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 $s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T$ following π

3 $g \leftarrow 0$

4   **for** $t = T-1, T-2, \ldots, 0$ **do**

5 $g \leftarrow \gamma g + r_{t+1}$

6   Returns$(s_t, a_t) \leftarrow$ Returns$(s_t, a_t) + g$

7   Num$(s_t, a_t) \leftarrow$ 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 Π_ϵ, not in Π, so it is not guaranteed to be optimal, but it is still a good policy (if ϵ 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 $w_k = \frac{1}{k-1}\sum_{i=1}^{k-1} x_i, k = 2, 3, \ldots$, then:

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{k-1}{k}w_k + \frac{1}{k}x_k$$

this update the mean estimate incrementally. We use

$$w_{k+1} = w_k - \alpha_k(w_k - x_k) \tag{1.19}$$

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

Given a noisy observation of $g(w)$:

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

where η is a noise, we can use the stochastic approximation method *Robbins-Monro* algorithm to update the estimate of w:

$$w_{k+1} = w_k - \alpha_k\tilde{g}(w_k, \eta_k), \alpha_k > 0 \tag{1.20}$$

Why this algorithm converge?

Theorem 1.8 (Robbins-Monro theorem)

If

   1. $0 \leq c_1 \leq \nabla_w g(w) \leq c_2 < \infty, \forall w$; *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. $\mathbb{E}[\eta_k \mid \mathcal{H}_k] = 0$ *and* $\mathbb{E}[\eta_k^2 \mid \mathcal{H}_k] < \infty$;

> where $\mathcal{H}_k = \{w_k, w_{k-1}, \dots, \}$, then w_k converges to the root of $g(w)$ almost surely, i.e., $\lim_{k \to \infty} w_k = w^*$, where $g(w^*) = 0$.
>
> ♡

Then let $g(w) = w - \mathbb{E}[X], \tilde{g}(w, \eta) = w - x + \eta, \eta = \mathbb{E}[X] - x$, we can use the Robbins-Monro algorithm to prove that with careful choice of α_k, the estimate w_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 Δ_k converges to 0 almost surely, if:
   1. *$\sum_{k=1}^{\infty} \alpha_k = \infty$, $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$, and $\sum_{k=1}^{\infty} \beta_k^2 < \infty$ uniformly almost surely;*
   2. *$\mathbb{E}[\eta_k \mid \mathcal{H}_k] = 0$ and $\mathbb{E}[\eta_k^2 \mid \mathcal{H}_k] < \infty$ 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, α_k, β_k are deterministic, but in Dvoretzky's theorem, they can be stochastic, and is useful in cases they are function of Δ_k.

♡

Proof Please refer to the book, page 110.

And we can also use this theorem to prove the convergence of the mean estimation, Robbins-Monro algorithm as well, and please refer to the book too.

Theorem 1.10 (Exentsion of Dvoretzky's Theorem)

Consider a finite set $\mathcal{S}$ of real numbers, a stochastic process:

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

$\Delta_k(s)$ converges to 0 almost surely, if:
   1. *$\sum_{k=1}^{\infty} \alpha_k(s) = \infty$, $\sum_{k=1}^{\infty} \alpha_k^2(s) < \infty$, $\sum_{k=1}^{\infty} \beta_k^2(s) < \infty$ and $\mathbb{E}[\beta_k(s) \mid \mathcal{H}_k] \leq \mathbb{E}[\alpha_k(s) \mid \mathcal{H}_k]$ uniformly almost surely.*
   2. *$\|\mathbb{E}[\eta_k \mid \mathcal{H}_k]\|_\infty \leq \gamma \|\Delta_k\|_\infty$, where $\gamma \in (0, 1)$*
   3. *$var[\eta_k \mid \mathcal{H}_k] \leq C(1 + \|\Delta_k\|_\infty)^2$, where C is a constant.*

where $\mathcal{H}_k = \{\Delta_k, \Delta_{k-1}, \dots, \eta_{k-1}, \dots, \alpha_{k-1}, \dots, \beta_{k-1}, \dots, \}$.
   - *The variable s can be viewed as an index, in the context of reinforcement learning, it indicates a state or a state-action pair. The norm is maximum norm.*
   - *This theorem only requires that the expectation and variance are bounded by the error. The convergence requires the conditions are valid for every s.*

♡

We then show that *stochastic gradient descent* (SGD) is a special case of the Robbins-Monro algorithm, and mean estimation is a special case of the SGD. The preliminary knowledge of gradient descent, convex function and the convergence of the GD please refer to the book appendix D.

Given a optimization problem:

$$\min_w J(w) = \mathbb{E}[f(w, X)]$$

where f is a function of parameter w and random vairable X, we can use the gradient descent to solve it:

$$w_{k+1} = w_k - \alpha_k \nabla_{w_k} J(w_k) = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X_k)] \tag{1.21}$$

The expectation is not computable, so we collect a larger number of data:

$$w_{k+1} = w_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i)$$

And we if only use one sample at a time, we can use the stochastic gradient descent (SGD):

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k) \tag{1.22}$$

An interesting **convergence pattern** is that it behaves similarly to the regular gradient descent algorithm when the estimate w_k is far from the optimal solution w^*. Only when they are close, does the convergence of SGD exhibit more randomness.

$$
\begin{aligned}
\delta_k &= \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}\left[\nabla_w f(w_k, X)\right]|}{|\mathbb{E}\left[\nabla_w f(w_k, X)\right]|} \\
&= \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}\left[\nabla_w f(w_k, X)\right]|}{|\mathbb{E}\left[\nabla_w f(w_k, X)\right] - \mathbb{E}\left[\nabla_w f(w^*, X)\right]|} \quad \text{as } \mathbb{E}\left[\nabla_w f(w^*, X)\right] = 0 \\
&= \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}\left[\nabla_w f(w_k, X)\right]|}{|\mathbb{E}\left[\nabla_{\tilde{w}}^2 f(w_k, X)\right](w_k - w^*)|} \quad \text{(by mean value theorem)} \\
&\leq \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}\left[\nabla_w f(w_k, X)\right]|}{c\,|w_k - w^*|}
\end{aligned}
$$

Above we use stochastic formulation, but we can also use the deterministic formulation, the equations are the same, but samples are not random, they are fixed $\{x_i\}_{i=1}^n$. So does the same equations are SGD too? The answer is yes, because if we set X be a random variable that $P(X = x_i) = \frac{1}{n}$, then $\min_w J(w) = \frac{1}{n}\sum f(w, x_i) = \mathbb{E}[f(w, X)]$. It means if we uniformly and independently sample x_k from the $\{x_i\}_{i=1}^n$, it is still SGD.

> **Theorem 1.11 (Convergence of SGD)**
>
> w_k converges to the optimal solution w^* almost surely, where $\mathbb{E}\left[\nabla_w f(w^*, X)\right] = 0$, if:
> 1. $0 < c_1 \leq \nabla_w^2 f(w, X) \leq c_2$;
> 2. $\sum_{k=1}^\infty \alpha_k = \infty$ and $\sum_{k=1}^\infty \alpha_k^2 < \infty$;
> 3. $\{x_k\}_{k=1}^\infty$ is i.i.d.

Proof Let
$$
g(w) = \nabla_w J(w) = \mathbb{E}[\nabla_w f(w, X)]
$$

Then SGD aims to find the root of $g(w) = 0$, and $\tilde{g} = \nabla_w f(w, x)$ where x is a sample of X:
$$
\tilde{g}(w, \eta) = \mathbb{E}[\nabla_w f(w, X)] + \nabla_w f(w, x) - \mathbb{E}[\nabla_w f(w, X)] = \mathbb{E}[\nabla_w f(w, X)] + \eta
$$

Then the RM is:
$$
w_{k+1} = w_k - \alpha_k \tilde{g}(w, \eta) = w_k - \alpha_k \nabla_w f(w, x)
$$

Then we show it follows 3 conditions in 1.8:
1. $\nabla_w g(w) = \nabla_w \mathbb{E}[\nabla_w f(w, X)] = \mathbb{E}[\nabla_w^2 f(w, X)]$, then because $0 < c_1 \leq \nabla_w^2 f(w, X) \leq c_2$, $c_1 \leq \nabla_w g(w) \leq c_2$.
2. second conditions are the same
3. since $\{x_k\}_{k=1}^\infty$ is i.i.d, $\mathbb{E}_{x_k}[\nabla_w f(w, x_k)] = \mathbb{E}[\nabla_w f(w, X)]$ for all k, therefore
$$
\mathbb{E}[\eta_k | \mathcal{H}_k] = \mathbb{E}[\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)] | \mathcal{H}_k] = 0
$$
as $\mathbb{E}[\nabla_w f(w_k, x_k)|\mathcal{H}_k] = \mathbb{E}_{x_k}[\nabla_w f(w_k, x_k)]$ because of independency, $\mathbb{E}[\mathbb{E}[\nabla_w f(w_k, X)]|\mathcal{H}_k] = \mathbb{E}[\nabla_w f(w_k, X)]$ because $\mathbb{E}[\nabla_w f(w_k, X)]$ is function of w_k. Same we can show $\mathbb{E}[\eta_k^2|\mathcal{H}_k] < \infty$.

1.9 Temporal-Difference Methods

The algorithm is:
$$
v_{t+1}(s_t) = v_t(s_t) - \alpha_t(s_t)[v_t(s_t) - (r_{t+1} + \gamma v_t(s_{t+1}))] \tag{1.23}
$$

and $v_{t+1}(s) = v_t(s)$ for all $s \neq s_t$, where $t = 0, 1, 2, \ldots$. Here $v_t(s_t)$ is the estimate of $V^\pi(s_t)$, $\alpha_t(s_t)$ is the learning rate. Only value that are visited will update!

$$
V^\pi(s) = \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] = \mathbb{E}\left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s\right]
$$

Then let
$$g(V^\pi(s_t)) = V^\pi(s_t) - \mathbb{E}\big[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s\big] = 0$$

But we will only get samples so
$$\begin{aligned}
\tilde{g}(V^\pi(s_t)) &= V^\pi(s_t) - [r_{t+1} + \gamma V^\pi(s_{t+1})] \\
&= \underbrace{(V^\pi(s_t) - \mathbb{E}\left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s_t\right])}_{g(V^\pi(s_t))} \\
&\quad + \underbrace{(\mathbb{E}\left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s_t\right] - [r_{t+1} + \gamma V^\pi(s_{t+1})])}_{\eta}.
\end{aligned}$$

then we get RM algorithm as:
$$v_{t+1}(s_t) = v_t(s_t) - \alpha_t(s_t)[v_t(s_t) - (r_{t+1} + \gamma V^\pi(s_{t+1}))]$$

the only difference is V^π replaces $v_t(s_{t+1})$. And we will show this replacement does not affect the convergence of TD in 1.12.

$$\underbrace{v_{t+1}(s_t)}_{\text{new estimate}} = \underbrace{v_t(s_t)}_{\text{current estimate}} - \alpha_t(s_t) \overbrace{[v_t(s_t) - \underbrace{(r_{t+1} + \gamma v_t(s_{t+1}))}_{\text{TD target } \bar{v}_t}]}^{\text{TD error } \delta_t}$$

$\bar{v}_t$ is called TD target because:
$$\begin{aligned}
v_{t+1}(s_t) - \bar{v}_t &= [1 - \alpha_t(s_t)][v_t(s_t) - \bar{v}_t] \\
&\to \text{Get the absolute value} \\
|v_{t+1}(s_t) - \bar{v}_t| &= |1 - \alpha_t(s_t)||v_t(s_t) - \bar{v}_t| \\
&< |v_t(s_t) - \bar{v}_t|
\end{aligned}$$

The expectation of TD error is 0 if $v_t(s_t) = V^\pi(s_t)$, so it reflects the discrepancy between the estimate and the true state value, it can be interpreted as *innovation*, which indicates new information obtained from the experience sample. A comparison of TD and MC can be seen in the book, table 7.1.

> **Theorem 1.12 (Convergence of TD)**
>
> *Given a policy π, in 1.23, $v_t(s)$ converges almost surely to $V^\pi(s)$ as $t \to \infty$ for all $s \in \mathcal{S}$ if $\sum_t \alpha_t(s) = \infty$ and $\sum_t \alpha_t^2(s) < \infty$ for all $s \in \mathcal{S}$. When α_t is constant, it can still be shown that the algorithm converges in the sense of expectation, though it does not follow second condition.* ♡

Proof Deducting $V^\pi(s)$ from both side in 1.23, then show that it follows 1.10.

Then we introduce *Sarsa*, another TD algorithm that can directly estimate action values.
$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t)\big[q_t(s_t, a_t) - (r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1}))\big] \tag{1.24}$$

and $q_{t+1}(s, a) = q_t(s, a)$ for all $(s, a) \neq (s_t, a_t)$, where $t = 0, 1, 2, \ldots$. It is given for solving the Bellman equation of a given policy:
$$Q^\pi(s, a) = \mathbb{E}\big[R + \gamma Q^\pi(S', A') \mid s, a\big], \quad \forall(s, a) \tag{1.25}$$

Let's proof this is a Bellman equation:

Proof
$$\begin{aligned}
Q^\pi(s, a) &= \sum_r r p(r|s, a) + \gamma \sum_{s'} \sum_{a'} Q^\pi(s', a') p(s'|s, a) \pi(a'|s') \\
&= \sum_r r p(r|s, a) + \gamma \sum_{s'} p(s'|s, a) \sum_{a'} Q^\pi(s', a') \pi(a'|s').
\end{aligned}$$

This equation establishes the relationships among the action values. Since
$$\begin{aligned}
p(s', a'|s, a) &= p(s'|s, a) p(a'|s', s, a) \\
&= p(s'|s, a) p(a'|s') \quad \text{due to conditional independence} \\
&= p(s'|s, a) \pi(a'|s')
\end{aligned}$$

Then

$$Q^\pi(s,a) = \sum_r rp(r|s,a) + \gamma \sum_{s'} \sum_{a'} Q^\pi(s',a')p(s',a'|s,a).$$

Algorithm 3: Optimal policy learning by Sarsa

Input: Initial policy $\pi_0(a|s)$, $Q_0(s,a)$, $\forall(s,a)$, $\epsilon \in (0,1]$

1 foreach *episode* **do**

2   **Episode generation:** Generate a_0 at s_0 following $\pi_0(s_0)$

3   **while** $s_t(t = 0, 1, 2, \ldots) \neq s_{target}$ **do**

4   Generate r_{t+1}, s_{t+1} and a_{t+1}, generate a_{t+1} following $\pi_{t+1}(s_t)$, generate r_{t+1}, s_{t+1} by interacting with enviroment

5 $q_{t+1}(s_t,a_t) \leftarrow q_t(s_t,a_t) - \alpha_t(s_t,a_t)[q_t(s_t,a_t) - (r_{t+1} + \gamma q_t(s_{t+1},a_{t+1}))]$

6 $\pi_{t+1}(a|s_t) \leftarrow \begin{cases} 1 - \frac{\epsilon}{|\mathcal{A}(s)|}(|\mathcal{A}(s)| - 1) & a = a^* = \arg\max_a q_{t+1}(s_t,a) \\ \frac{\epsilon}{|\mathcal{A}(s)|} & a \neq a^*. \end{cases}$

7   **end**

8 end

A variant of Sarsa is Expected Sarsa:

$$q_{t+1}(s_t,a_t) = q_t(s_t,a_t) - \alpha_t(s_t,a_t)\big[q_t(s_t,a_t) - (r_{t+1} + \gamma\mathbb{E}[q_t(s_{t+1},A)])\big]$$

where $\mathbb{E}[q_t(s_{t+1},A)] = \sum_a \pi_t(a|s_{t+1})q_t(s_{t+1},a) = v_t(s_{t+1})$. Although calculating the expected value may increase the computational complexity slightly, it is beneficial in the sense that it reduces the estimation variances. This variant can be viewed as solving

$$Q^\pi(s,a) = \mathbb{E}\big[R_{t+1} + \gamma\mathbb{E}[Q^\pi(S_{t+1},A_{t+1}) \mid S_{t+1}] \mid S_t = s, A_t = a\big], \quad \forall(s,a)$$

which is again a Bellman equation as $\mathbb{E}[Q^\pi(S_{t+1},A_{t+1}) \mid S_{t+1}] = \sum_{A'} Q^\pi(S_{t+1},A')\pi(A' \mid S_{t+1}) = V^\pi(S_{t+1})$.

Now we introduce ***n-step Sarsa***, We will see that Sarsa and MC learning are two extreme cases of n-step Sarsa.

$$\begin{aligned} \text{Sarsa} \quad &\leftarrow \quad G_t^{(1)} = R_{t+1} + \gamma Q^\pi(S_{t+1},A_{t+1}) \\ &\qquad\quad G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q^\pi(S_{t+2},A_{t+2}) \\ &\qquad\quad \vdots \\ \text{n-step Sarsa} \quad &\leftarrow \quad G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n Q^\pi(S_{t+n},A_{t+n}) \\ &\qquad\quad \vdots \\ \text{policy iteration} \quad &\leftarrow \quad G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots \end{aligned}$$

It should be noted that $G_t = G_t^{(1)} = G_t^{(2)} = G_t^{(n)} = G_t^{(\infty)}$, where the superscripts merely indicate the different decomposition structures of G_t.

When $n = 1$, we get 1.24, when $n = \infty$, $q_{t+1}(s_t,a_t) = g_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots$. And for the general n:

$$q_{t+1}(s_t,a_t) = q_t(s_t,a_t) - \alpha_t(s_t,a_t)[q_t(s_t,a_t) - (r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^n q_t(s_{t+n},a_{t+n}))]$$

Now, finally we introduce ***Q-learning***, which can directly estimate optimal action values and find optimal policies (Sarsa need the policy improvement step):

$$q_{t+1}(s_t,a_t) = q_t(s_t,a_t) - \alpha_t(s_t,a_t)\big[q_t(s_t,a_t) - (r_{t+1} + \gamma \max_{a \in A(s_{t+1})} q_t(s_{t+1},a))\big] \tag{1.26}$$

Given (s_t,a_t), Sarsa need $(r_{t+1}, s_{t+1}, a_{t+1})$ in every iteration, whereas Q-learning merely requires (r_{t+1}, s_{t+1}). It is for solving the following equation:

$$q(s,a) = \mathbb{E}\big[R_{t+1} + \gamma \max_a q(S_{t+1},a) \mid S_t = s, A_t = a\big] \tag{1.27}$$

we can show this is also a Bellman equation:

$$q(s, a) = \sum_{r} p(r \mid s, a)r + \gamma \sum_{s'} p(s'|s, a) \max_{a \in \mathcal{A}(s')} q(s', a)$$

$$\xrightarrow{\text{taking maximum}}$$

$$\max_{a \in \mathcal{A}(s)} = \max_{a \in \mathcal{A}(s')} \Big[\sum_{r} p(r \mid s, a)r + \gamma \sum_{s'} p(s'|s, a) \max_{a \in \mathcal{A}(s')} q(s', a) \Big]$$

$$\xrightarrow{\text{let } v(s) = \max_{a \in \mathcal{A}(s)} q(s,a)}$$

$$v(s) = \max_{a \in \mathcal{A}(s')} \Big[\sum_{r} p(r \mid s, a)r + \gamma \sum_{s'} p(s'|s, a)v(s') \Big]$$

$$= \max_{\pi} \sum_{a \in A(s)} \pi(a \mid s) \Big[\sum_{r} p(r \mid s, a)r + \gamma \sum_{s'} p(s'|s, a)v(s') \Big]$$

> **Definition 1.3**
>
> - *behavior policy* is the one used to generate experience samples.
> - *target policy* is the one that is constantly updated to converge to an optimal policy.
> - *on-policy*: When the behavior policy is the same as the target policy, such a learning process is called on-policy.
> - *off-policy*: When the behavior policy is **not** the same as the target policy, like Q-learning. The advantage of off-policy learning is that it can learn optimal policies based on the experience samples generated by other policies. So we can choose a exploratory behavior policy and the learning efficiency will be increased. ♣

The fundamental reason why Q-learning is off-policy is that Q-learning aims to solve the Bellman optimality equation rather than the Bellman equation of a given policy.

Algorithm 4: Optimal policy learning via Q-learning (on-policy version)

Input: Initial policy $\pi_0(a|s)$ from q_0, $q_0(s, a), \forall(s, a), \epsilon \in (0, 1]$

Output: Learn an optimal path that can lead the agent to the target state from an initial state s_0

1 **foreach** *episode* **do**
2   **while** $s_t(t = 0, 1, 2, \ldots) \neq s_{target}$ **do**
3   Collect a_t, r_{t+1}, s_{t+1}, generate a_t following $\pi_t(s_t)$, generate r_{t+1}, s_{t+1} by interacting with enviroment
4        $q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t)\big[q_t(s_t, a_t) - (r_{t+1} + \gamma \max_{a \in A(s_{t+1})} q_t(s_{t+1}, a))\big]$
5        $\pi_{t+1}(a|s_t) \leftarrow \begin{cases} 1 - \frac{\epsilon}{|\mathcal{A}(s)|}(|\mathcal{A}(s)| - 1) & a = a^* = \arg\max_a q_{t+1}(s_t, a) \\ \frac{\epsilon}{|\mathcal{A}(s)|} & a \neq a^*. \end{cases}$
6   **end**
7 **end**

Algorithm 5: Optimal policy learning via Q-learning (off-policy version)

Input: Behavior policy $\pi_b(a|s)$, initial guess $q_0(s, a), \forall(s, a)$ and all t

Output: Learn an optimal target policy π_T for all states from the experience samples generated by π_b

1 **foreach** *episode* $\{s_0, a_0, r_1, s_1, a_1, r_2, \ldots\}$ *generated by* π_b **do**
2   **foreach** *step* $t = 0, 1, 2, \ldots$ *of the episode* **do**
3        $q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t)\big[q_t(s_t, a_t) - (r_{t+1} + \gamma \max_{a \in A(s_{t+1})} q_t(s_{t+1}, a))\big]$
4        $\pi_{T,t+1}(a|s_t) \leftarrow \begin{cases} 1 & a = a^* = \arg\max_a q_{t+1}(s_t, a) \\ 0 & a \neq a^*. \end{cases}$
5   **end**
6 **end**

We can use a unified formulation for all the TD algorithm:

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t)\big[q_t(s_t, a_t) - \bar{q}_t\big]$$

Algorithm	Expression of the TD target $\bar{q}_t$ in (7.20)
Sarsa	$\bar{q}_t = r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})$
n-step Sarsa	$\bar{q}_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^n q_t(s_{t+n}, a_{t+n})$
Q-learning	$\bar{q}_t = r_{t+1} + \gamma \max_a q_t(s_{t+1}, a)$
Monte Carlo	$\bar{q}_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots$

Algorithm	Equation to be solved
Sarsa	BE: $q_\pi(s, a) = \mathbb{E}\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \vert S_t = s, A_t = a\right]$
n-step Sarsa	BE: $q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n q_\pi(S_{t+n}, A_{t+n}) \vert S_t = s, A_t = a]$
Q-learning	BOE: $q(s, a) = \mathbb{E}\left[R_{t+1} + \gamma \max_a q(S_{t+1}, a) \big\vert S_t = s, A_t = a\right]$
Monte Carlo	BE: $q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \vert S_t = s, A_t = a]$

Figure 1.2: TD algorithms

1.10 Value Function Methods

Instead of using table to represent the value function, which takes a lot of memory and does not generalize well at unknown state, we now introduce another way - using function estimation. In particular, the objective function is:

$$J(w) = \mathbb{E}\big[(V^\pi(S) - \hat{v}(S, w))^2\big] \tag{1.28}$$

We can use a uniform distribution as the distribution of S, but it does not consider the Markov process property. So we use **stationary distribution**, the long-term behavior, after sufficiently long period, every state will have its stationary probability. Let $\{d_\pi(s)\}_{s \in \mathcal{S}}$ denote the stationary distribution of the Markov process under policy π. The objective is now:

$$J(w) = \sum_{s \in \mathcal{S}} d_\pi(s)(V^\pi(S) - \hat{v}(S, w))^2 \tag{1.29}$$

> **Definition 1.4**
>
> *The probability of the agent transitioning from s_i to s_j using exactly k steps is denoted as*
>
> $$p_{ij}^k = \Pr(S_{t_k} = j \mid S_{t_0} = i)$$
>
> *by the definition of P_π, we have*
>
> $$[P_\pi]_{ij} = p_{ij}^{(1)}$$
>
> *which means $[P_\pi]_{ij}$ is the probability of transitioning from s_i to s_j using a single step. Second, consider P_π^2:*
>
> $$[P_\pi^2]_{ij} = [P_\pi P_\pi]_{ij} = \sum_{q=1}^{n} [P_\pi]_{iq}[P_\pi]_{qj} = p_{ij}^{(2)}$$
>
> *In the same manner:*
>
> $$[P_\pi^n]_{ij} = p_{ij}^{(n)}$$

Let $d_0 \in \mathcal{R}^n$ be a vector representing the probability distribution of the states at the initial time step. Then

$$d_k(s_i) = \sum_{j=1}^{n} d_0(s_j)[P_\pi^k]_{ji}, i = 1, 2, \ldots \tag{1.30}$$

The matrix form is:

$$d_k^T = d_0^T P_\pi^k$$

When we consider the long-term behavior of the Markov process, it holds under certain conditions that

$$\lim_{k \to \infty} P_\pi^k = \mathbf{1}_n d_\pi^T$$

♣

1.11 Appendix: Measure-Theoretic Probability Theory

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

- Ω is a sample space, which is a set of all possible outcomes;
- $\mathcal{F}$ is a σ-algebra, which is a collection of subsets of Ω that includes the empty set and Ω, 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 Ω.
- P is a probability measure, which assigns a probability to each event in $\mathcal{F}$ such that $P(\Omega) = 1$ and $P(\emptyset) = 0$.

For example, the game of dice, $\Omega = \{1, 2, 3, 4, 5, 6\}$, $\mathcal{F} = \mathcal{P}(\Omega)$ is the power set of Ω, we can define one event $A = \{w \in \Omega : w > 3\} = \{4, 5, 6\}$ with $\mathbb{P}(A) = 1/2$.

Definition 1.5 (σ-algebra)

A σ-algebra $\mathcal{F}$ on a set Ω is a collection of subsets of Ω 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 $A_1, A_2, A_3, \ldots \in \mathcal{F}$, then $\bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$ (closed under countable unions).*

♣

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

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

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

Simple random variables are those that take a finite number of values, and they can be expressed as:

$$X(w) = \sum_{x \in \mathcal{X}} x \mathbb{I}_{A_x}(w)$$

where $A_x = \{w \in \Omega \mid X(w) = x\} = X^{-1}(x)$ and

$$\mathbb{I}_{A_x}(w) = \begin{cases} 1 & \text{if } w \in A_x, \\ 0 & \text{otherwise.} \end{cases}$$

With the above preparation, we can define the *Expectation* of a random variable X:

$$\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x \mathbb{P}(A_x) \tag{1.31}$$

Lemma 1.2 (Basic Properties)

1. *$\mathbb{E}[a \mid Y] = a$, where a is a constant;*
2. *$\mathbb{E}[aX + bZ \mid Y] = a\mathbb{E}[X \mid Y] + b\mathbb{E}[Z \mid Y]$, where a, b are constants;*
3. *$\mathbb{E}[X \mid Y] = \mathbb{E}[X]$ if X is independent of Y;*
4. *$\mathbb{E}[Xf(Y) \mid Y] = f(Y)\mathbb{E}[X \mid Y]$, where $f(Y)$ is a function of Y;*

---

5. $\mathbb{E}[f(Y) \mid Y] = f(Y)$;
6. $\mathbb{E}[X \mid Y, f(Y)] = \mathbb{E}[X \mid Y]$;
7. If $X \geq 0$, then $\mathbb{E}[X \mid Y] \geq 0$;
8. If $X \geq Z$, then $\mathbb{E}[X \mid Y] \geq \mathbb{E}[Z \mid Y]$;
9. $\mathbb{E}\big[\mathbb{E}[X \mid Y]\big] = \mathbb{E}[X]$;
10. $\mathbb{E}\big[\mathbb{E}[X \mid Y, Z]\big] = \mathbb{E}[X]$;
11. $\mathbb{E}\big[\mathbb{E}[X \mid Y] \mid Y\big] = \mathbb{E}[X \mid Y]$;

Given a stochastic sequence $\{X_k\} = \{X_1, X_2, \ldots\}$:

Definition 1.6 (Sure Convergence)

$\{X_k\}$ converges *surely* to X if:
$$\lim_{k \to \infty} X_k(w) = X(w), \quad \forall w \in \Omega$$

this can be equivalently written as:
$$A = \Omega \quad where \quad A = \{w \in \Omega \mid \lim_{k \to \infty} X_k(w) = X(w)\}$$

Definition 1.7 (Almost Sure Convergence)

$\{X_k\}$ converges *almost surely* (or almost everywhere or with probability 1) to X if:
$$\mathbb{P}(A) = 1 \quad where \quad A = \{w \in \Omega \mid \lim_{k \to \infty} X_k(w) = X(w)\}$$

The points, for which this limit is invalid, form a set of zero measure. For the sake of simplicity, it is often written as:
$$P\big(\lim_{k \to \infty} X_k = X\big) = 1$$

and denote as $X_k \xrightarrow{a.s.} X$.

Definition 1.8 (Convergence in Probability)

$\{X_k\}$ converges *in probability* to X if $\forall \epsilon > 0$:
$$\lim_{k \to \infty} \mathbb{P}(A_k) = 0 \quad where \mathbb{A}_k = \{w \in \Omega : |X_k(w) - X(w)| > \epsilon\}$$

This means that the probability of the difference between X_k and X being greater than ϵ goes to zero as k increases. It can be written as:
$$\lim_{k \to \infty} \mathbb{P}\big(|X_k - X| > \epsilon\big) = 0$$

Definition 1.9 (Convergence in Mean)

$\{X_k\}$ converges *in mean* (r-th mean or in L^r norm) to X if:
$$\lim_{k \to \infty} \mathbb{E}[|X_k - X|^r] = 0$$

This means that the expected value of the absolute difference between X_k and X goes to zero as k increases.

Definition 1.10 (Convergence in Distribution)

$\{X_k\}$ converges *in distribution* to X if:
$$\lim_{k \to \infty} \mathbb{P}(X_k \leq a) = \mathbb{P}(X \leq a) \quad \forall a \in \mathbb{R}$$

a compact expression is:
$$\lim_{k \to \infty} \mathbb{P}(A_k) = \mathbb{P}(A)$$

> *where*
> $$A_k = \{w \in \Omega : X_k(w) \le a\}, A = \{w \in \Omega : X(w) \le a\}$$
>
> ♣

The relationship between these convergence types is: almost sure convergence $\rightarrow$ convergence in probability $\rightarrow$ convergence in distribution. Convergence in mean $\rightarrow$ convergence in probability $\rightarrow$ convergence in distribution. And convergence in mean does not imply almost sure convergence.

Chapter 2   From LQR to RL

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:

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

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 negitive 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:

$$V(\boldsymbol{x}_T) = \text{const} + \frac{1}{2} \boldsymbol{x}_T^T \boldsymbol{V}_T \boldsymbol{x}_T + \boldsymbol{x}_T^T \boldsymbol{v}_T$$

$$\boldsymbol{V}_T = \boldsymbol{C}_{\boldsymbol{x}_T, \boldsymbol{x}_T} + \boldsymbol{C}_{\boldsymbol{x}_T, \boldsymbol{u}_T} \boldsymbol{K}_T + \boldsymbol{K}_T^T \boldsymbol{C}_{\boldsymbol{u}_T, \boldsymbol{x}_T} + \boldsymbol{K}_T^T \boldsymbol{C}_{\boldsymbol{u}_T, \boldsymbol{u}_T} \boldsymbol{K}_T$$

$$\boldsymbol{v}_T = \boldsymbol{c}_{\boldsymbol{x}_T} + \boldsymbol{C}_{\boldsymbol{x}_T, \boldsymbol{u}_T} \boldsymbol{k}_T + \boldsymbol{K}_T^T \boldsymbol{C}_{\boldsymbol{u}_T} + \boldsymbol{K}_T^T \boldsymbol{C}_{\boldsymbol{u}_T, \boldsymbol{u}_T} \boldsymbol{k}_T$$

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:

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

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

$$Q_{T-1} = C_{T-1} + F_{T-1}^T V_T F_{T-1}$$

$$q_{T-1} = c_{T-1} + F_{T-1}^T V_T f_{T-1} + F_{T-1}^T v_T$$

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

$$u_{T-1} = K_{T-1} x_{T-1} + k_{T-1}$$

$$K_{T-1} = -Q_{u_{T-1},u_{T-1}}^{-1} Q_{u_{T-1},x_{T-1}}$$

$$k_{T-1} = -Q_{u_{T-1},u_{T-1}}^{-1} q_{u_{T-1}}$$

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 wirte the Backward Recursion as algorithm:

Algorithm 6: 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), beacuse the expectation of gaussian is zero for linear and constant for quadratic cost ($\mathbb{E}[x_{t+1}^\top V x_{t+1}] = (Ax_t + Bu_t)^\top V(Ax_t + Bu_t) + tr(VW)$, so when minimizing it is ignored).

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

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

$$c(\mathbf{x}_t, \mathbf{u}_t) \approx c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

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

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*.