Notes on Reinforcement Learning

Randy Sun

March 11, 2018

1 Markov Decision Process

This section introduces several basic elements of Markov Decision Process. Also, some important equations would be presented.

1.1 Fundamental elements

Markov Decision Process (MDP) is a tuple, $(S, A, P, \pi, R, \gamma)$.

- S is a finite set which contains a set of states, $\{s_1, s_2, ..., s_N\}$. $S(s, a) \subseteq S$ is a set of states which are available after state s and action a.
- A is an action set which contains a set of actions, $\{a_1, a_2, ..., a_M\}$. $A(s) \subseteq A$ is a set of actions which are available after action s.
- P are transition probability measures.
$$P^a_{s,s'} = \Pr\left(\mathrm{s_{t+1} = s'|s_t = s, a_t = a}\right). \tag{1.1}$$
- π is policy. $\pi(a_t = a|s_t = s)$ is the probability of taking an action, a, under state s and policy, π. Moreover,
$$\pi(s_{t+1} = s'|s_t = s) = \sum_{a\in A(s)} \pi(a_t = a|s_t = s) P^a_{s,s'}. \tag{1.2}$$
- R are rewards.
$$R_{t+1} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3}... = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \tag{1.3}$$
where
$$r_{t+1} = r(s_t, a_t, s_{t+1}) \tag{1.4}$$
depend on the current state, s_{t+1}, last state, s_t, and last action, a_t.
$$R^a_s = E_\pi\left[r_{t+1}|s_t = s, a_t = a\right] = \sum_{s'\in S(s,a)} P^a_{s,s'} r_{t+1}(s_t = s, a_t = a, s_{t+1} = s'). \tag{1.5}$$

- γ is the discount rate, which is smaller than 1.

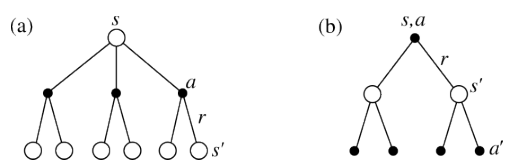
There are two important functions, which play the crucial roles in reinforcement learning, *state-value function for policy* π, V^π and *action-value function for policy* π, Q^π.

$$\begin{aligned} V^\pi(s) =& E_\pi\left[R_{t+1}|s_t = s\right] \\ =& \sum_{a\in A(s)} \pi\left(a_t = a|s_t = s\right) Q^\pi\left(s,a\right) \end{aligned} \tag{1.6}$$

means the expected reward given the current state is s under policy π, and

$$\begin{aligned} Q^\pi(s,a) =& E_\pi\left[R_{t+1}|s_t = s, a_t = a\right] \\ =& \sum_{s'\in S(s,a)} P^a_{s,s'}\left(r\left(s_t = s, a_t = a, s_{t+1} = s'\right) + \gamma V^\pi\left(s'\right)\right) \\ =& R^a_s + \gamma \sum_{s'\in S(s,a)} P^a_{s,s'} V^\pi\left(s'\right) \end{aligned} \tag{1.7}$$

means the expected reward given the current state is s and current action a under policy π. As the state-value function, action-value function can be written as the linear combination of state-value function, either.



Equations (7) and (8) could be unstood through this graph.

Reinforce Learning cares about looking for the *best policy*, π^*. It aims to estimate

$$\begin{aligned} \max_\pi V^\pi\left(s\right) =& V^{\pi^*}\left(s\right) \\ =& \max_{a\in A(s)} Q^{\pi^*}\left(s,a\right) \end{aligned} \tag{1.8}$$

and

$$\max_\pi Q^\pi\left(s,a\right) = Q^{\pi^*}\left(s,a\right). \tag{1.9}$$

It's intuitively to write down the last equation in (9), since the best policy under a state, s, would be taking the action with the maximum expected reward.

1.2 Bellman Equation

To estimate, we rewrite $V^{\pi},\ Q^{\pi},\ V^{\pi^*},\ Q^{\pi^*}$ by the form of Bellman equation.

$$
\begin{aligned}
V^{\pi}(s) =& E_{\pi}\left[R_{t+1}|s_t=s\right] \\
=& E_{\pi}\left[r_{t+1}+\gamma R_{t+2}|s_t=s\right] \\
=& \sum_{a,s'} \mathrm{Pr}_{\pi}\left(\mathrm{a_t=a,s_{t+1}=s'|s_t=s}\right)\left(\mathrm{E}_{\pi}\left[\mathrm{r_{t+1}+\gamma R_{t+2}|s_t=s,a_t=a,s_{t+1}=s'}\right]\right) \\
=& \sum_{a,s'} \mathrm{Pr}_{\pi}\left(\mathrm{a_t=a,s_{t+1}=s'|s_t=s}\right)\left(\mathrm{r_{t+1}}\left(\mathrm{s_t=s,a_t=a,s_{t+1}=s'}\right)+\mathrm{E}_{\pi}\left[\mathrm{\gamma R_{t+2}|s_t=s,a_t=a,s_{t+1}=s'}\right]\right) \\
=& \sum_{a,s'} \mathrm{Pr}_{\pi}\left(\mathrm{a_t=a,s_{t+1}=s'|s_t=s}\right)\left(\mathrm{r_{t+1}}\left(\mathrm{s_t=s,a_t=a,s_{t+1}=s'}\right)+\mathrm{E}_{\pi}\left[\mathrm{\gamma R_{t+2}|s_{t+1}=s'}\right]\right) \\
=& \sum_{a,s'} \mathrm{Pr}_{\pi}\left(\mathrm{a_t=a,s_{t+1}=s'|s_t=s}\right)\left(\mathrm{r_{t+1}}\left(\mathrm{s_t=s,a_t=a,s_{t+1}=s'}\right)+\gamma \mathrm{V}^{\pi}\left(\mathrm{s'}\right)\right) \\
=& \sum_{a\in A(s)} \pi\left(a_t=a|s_t=s\right) \sum_{s'\in S(s,a)} P^{a}_{s,s'}\left(r_{t+1}\left(s_t=s,a_t=a,s_{t+1}=s'\right)+\gamma V^{\pi}\left(s'\right)\right)
\end{aligned}
\tag{1.10}
$$

and

$$
\begin{aligned}
Q^{\pi}(s,a) =& E_{\pi}\left[R_{t+1}|s_t=s,a_t=a\right] \\
=& E_{\pi}\left[r_{t+1}+\gamma R_{t+2}|s_t=s,a_t=a\right] \\
=& \sum_{s',a'} \mathrm{Pr}_{\pi}\left(\mathrm{a_{t+1}=a',s_{t+1}=s'|s_t=s,a_t=a}\right)\left(\mathrm{E}_{\pi}\left[\mathrm{r_{t+1}+\gamma R_{t+2}|s_t=s,a_t=a,s_{t+1}=s',a_{t+1}=a'}\right]\right) \\
=& \sum_{s',a'} \mathrm{Pr}_{\pi}\left(\mathrm{a_{t+1}=a',s_{t+1}=s'|s_t=s,a_t=a}\right)\left(\mathrm{R^{a}_{s,s'}}+\gamma \mathrm{Q}^{\pi}\left(\mathrm{s',a'}\right)\right). \\
=& R^{a}_{s} + \sum_{s'\in S(s,a)} P^{a}_{s,s'}\gamma V^{\pi}\left(s'\right).
\end{aligned}
\tag{1.11}
$$

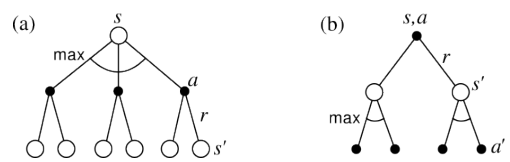
Actually, the equations (12) and (13) could be obtained directly from equations (7) and (8).

To have the Bellman equation form of V^{π^*}, we can move on from equation (9),

$$
\begin{aligned}
V^{\pi^*}(s) =& \max_a Q^{\pi^*}(s,a) \\
=& \max_a \left(R^{a}_{s}+\gamma \sum_{s'\in S(s,a)} P^{a}_{s,s'} V^{\pi^*}\left(s'\right)\right).
\end{aligned}
\tag{1.12}
$$

As for Q^{π^*}, since equation (12) holds for any policy, including best policy π^*, combing equations (12) and (13) could obtain the result.

$$
\begin{aligned}
Q^{\pi^*}(s,a) =& R_s^a + \gamma \sum_{s' \in S(s,a)} P_{s,s'}^a V^{\pi^*}(s') \\
=& R_s^a + \gamma \sum_{s' \in S(s,a)} P_{s,s'}^a \max_a Q^{\pi^*}(s',a)\,.
\end{aligned}
\tag{1.13}
$$



Equations (13) and (14) could be unstood through this graph.

2 Model-based Dynamic Programming

In this section, we will introduce policy iteration and value iteration. Moreover, we will introduce policy improvement theorem.

2.1 Policy iteration

Policy iteration is an model-based algortihm to find the best policy and its corresponding state-value function. It needs $P_{s,s'}^a$ as the input for the algorithm, so it is classified to the model-based category. Policy iteration algorithm consists of two parts: *Policy evaluation* and *Policy improvement*.

Given a policy π, the algorithm of Policy evaluation aims to estimate $V^\pi(s)\ \forall s \in S$. It esitmates the value iteratively by model-based dynamic programming.

Algorithm 1 Policy evaluation

Input: A policy π, transition probabilities $P_{s,s'}^a$, expected rewards R_s^a, and discount rate γ.

Output: state-value function $V^\pi(s)\ \forall s \in S$.

Initialize $V^\pi(s) = 0\ \forall s \in S$.
while True **do**
    for $s \in S$ **do**
        $V^\pi(s) = \sum_{a \in A(s)} \pi(a_t = a | s_t = s) \left(R_s^a + \gamma \sum_{s' \in S(s,a)} P_{s,s'}^a V^\pi(s') \right)$
    end for
    if $V_{k+1}^\pi(s) = V_k^\pi(s)\ \forall s \in S$ **then**
        break
    end if
end while
Return $V^\pi(s)\ \forall s \in S$.

Now, we are able to evaluate a policy. We want to improve the policy. By the policy improvement theorem(we leave the proof in subsection 3), if

$$V^{\pi}(s) \leq Q^{\pi}(s, \pi'(s)) \tag{2.1}$$

for all s,

$$V^{\pi}(s) \leq V^{\pi'}(s) \tag{2.2}$$

for all s. So, to have better policy, we use equation (1.7) to find the better policy π'. Let

$$\pi'(s) = \underset{a}{\operatorname{argmax}} \sum_{s' \in S(s,a)} P^{a}_{s,s'} \left(r\left(s_t = s, a_t = a, s_{t+1} = s'\right) + \gamma V^{\pi}(s')\right) \tag{2.3}$$

for each s. The algorithm is as follows:

Algorithm 2 Policy improvement

while True **do**
    policy stable = True
    for each s in S **do**
        Let $b = \pi(s)$.
        Let $\pi(s) = \underset{a}{\operatorname{argmax}} \sum_{s' \in S(s,a)} P^{a}_{s,s'} \left(r\left(s_t = s, a_t = a, s_{t+1} = s'\right) + \gamma V^{\pi}(s')\right)$
        if $b \neq \pi(s)$ **then**
            policy stable = False
        end if
    end for
    if *policy stable* **then**
        break
    end if
end while

Now, we are able to improve the policy. Then, we could combine the policy evalution and policy improvement to find the best policy. The whole process of

Algorithm 3 Policy iteration

Initialize $\pi(s) \in A(s)$ and $V^{\pi}(s)$ arbitrarily.
while True **do**
    Policy evaluation on π.
    Policy improvement on π and get π'.
    if $\pi = \pi'$ **then**
        break
    end if
    $\pi = \pi'$
end while

policy iteration could be unstood as the following graph:

$$\pi_0 \xrightarrow{\text{E}} V^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} V^{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} V^*,$$

2.2 Value iteration

Although there are some garauntees that the policy iteration would converge to the best policy, it sometimes suffers from protractive procedure. There are other ways to find the best policy. Instead of iteratively finding the best policy, one could try to find the state-value function directly in first, then use the state-value function to find the best policy.

Value iteration uses the following way to update the state-value function:

$$V_{n+1}\left(s\right) = \max_{a} \sum_{s' \in S(s,a)} P^{a}_{s,s'} \left(r\left(s_t = s, a_t = a, s_{t+1} = s'\right) + \gamma V^{\pi}\left(s'\right)\right) \tag{2.4}$$

Then, it would converge to the value-state function of the best policy. Lastly, use equation (1.6) to find the best policy. Note that the difference between the

---

Input: θ small enough.

Initialize $V\left(s\right) = 0 \ \forall s \in S$.
while True **do**
    $d = 0$.
    for $s \in S$ **do**
        $v = V\left(s\right)$.
        $V\left(s\right) = \max\limits_{a} \sum_{s' \in S(s,a)} P^{a}_{s,s'} \left(r\left(s_t = s, a_t = a, s_{t+1} = s'\right) + \gamma V^{\pi}\left(s'\right)\right)$.
        $d = \max\left(\mathrm{d}, \left|\mathrm{v} - \mathrm{V}\left(\mathrm{s}\right)\right|\right)$.
    end for
    if $d \leq \theta$ **then**
        break
    end if
end while
Let $\pi\left(s\right) = \operatorname*{argmax}\limits_{a} \sum_{s' \in S(s,a)} P^{a}_{s,s'} \left(r\left(s_t = s, a_t = a, s_{t+1} = s'\right) + \gamma V^{\pi}\left(s'\right)\right)$ $\forall s \in S$.

---

two update steps of value iteration and policy iteration could be understood through figure(1)(a) and figure(2)(a).

2.3 Policy Improvement Theorem

Given that $\forall s \in S,\ V^{\pi}(s) \leq Q^{\pi}(s, \pi'(s))$, then,

$$
\begin{aligned}
V^{\pi}(s) \leq & Q^{\pi}(s, \pi'(s)) \\
= & \sum_{s' \in S(s,\pi'(s))} P_{s,s'}^{\pi'(s)} \left(R_{s,s'}^{\pi'(s)} + \gamma V^{\pi}(s') \right) \\
\leq & \sum_{s' \in S(s,\pi'(s))} P_{s,s'}^{\pi'(s)} \left(R_{s,s'}^{\pi'(s)} + \gamma Q^{\pi}(s', \pi'(s')) \right) \\
= & \sum_{s' \in S(s,\pi'(s))} P_{s,s'}^{\pi'(s)} \left(R_{s,s'}^{\pi'(s)} + \gamma \sum_{s'' \in S(s',\pi'(s'))} P_{s',s''}^{\pi'(s')} \left(R_{s',s''}^{\pi'(s)} + \gamma V^{\pi}(s'') \right) \right) \\
= & \sum_{s' \in S(s,\pi'(s))} P_{s,s'}^{\pi'(s)} r_{t+1}(s, \pi'(s), s') + \gamma \sum_{s'' \in S(s',\pi'(s'))} P_{s',s''}^{\pi'(s')} r_{t+2}(s', \pi'(s'), s'') + \ldots \\
\leq & \cdots\cdots \\
\leq & \cdots\cdots \\
= & \sum_{a,s',a',s'',\ldots} \mathrm{Pr}_{\pi'}\left(\mathrm{a_t = a,\ s_{t+1} = s',\ a_{t+1} = a', \ldots | s_t = s}\right) \left(\mathrm{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots}\right) \\
= & V^{\pi'}(s).
\end{aligned}
\tag{2.5}
$$

That is to say, if the expected reward under a state s and an action $a = \pi'(s)$ is greater than expected reward under state s, $\forall s \in S$, the state value function of the new policy π' would be grater than the old one.

3 Monte Carlo estimate

Monte Carlo control is an approach to estimate the value-state function or action-state function. It is based on sampling many episodes and try to estimate it by averaging all the returns. In general, there are two ways to estimate, *first-visit MC estimate* and *every-visit MC estimate*. They share some crucial properties but have slightly different theoretical properties. Here, we consider the former one.

Algorithm 4 First-visit MC control for estimate state-value function

Initialize policy π, state-value function V^{π} and empty list $L\left(s\right) \forall s \in S$.
while True **do**
    Generate an episode with π.
    for each state s in the episode **do**
        Let G be the return of the state s.
        Append G to $L\left(s\right)$.
        $V\left(s\right) = \text{average}\left(\text{L (s)}\right)$.
    end for
end while

3.1 On-policy control

Before we move on to on-policy approach, we should discuss some thoughts about policy. We intriduce several common stochastic strategies below,

1. *Greedy strategy*

$$\pi\left(a_t = a | s_t = s\right) = \begin{cases} 1 & if\ a = \underset{a}{\operatorname{argmax}}\ Q^{\pi}\left(s, a\right) \\ 0 & if\ a \neq \underset{a}{\operatorname{argmax}}\ Q^{\pi}\left(s, a\right) \end{cases} \tag{3.1}$$

Greedy strategy selects the action with the largest reward, it is *deterministic*.

2. ϵ*-greedy strategy*

$$\pi\left(a_t = a | s_t = s\right) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|} & if\ a = \underset{a}{\operatorname{argmax}}\ Q^{\pi}\left(s, a\right) \\ \frac{\epsilon}{|A(s)|} & if a \neq \underset{a}{\operatorname{argmax}}\ Q^{\pi}\left(s, a\right) \end{cases} \tag{3.2}$$

ϵ-greedy strategy strikes a balanse between *exploitation* and *exploration*. It makes sure that all actions would have the chances to be executed.

Here, *exploitation* means always selecting the action with the largest expected return(greedy strategy). And the *exploration* means we should select other possible actions so that we may find out a better action than the action we see it as the best one.

Basically, there are two types of MC control: *with explorating starts* and *without explorating starts*. We first introduce the former algorithm.

Algorithm 5 Monte Carlo control with explorating starts.

1: Initialize policy π, action-value function Q^{π} and empty list $L\left(a,s\right)\forall s \in S\ and\ a \in A\left(s\right)$.
2: **while** True **do**
3: Choose s_0 and a_0 s.t all pairs have probabilities > 0.
4: Generate all pairs from s_0 and a_0.
5: **for** each pair s and a in the episode **do**
6: Let G be the return of first occurance of the pair s and a.
7: Append G to $L\left(s,a\right)$.
8: $Q\left(s,a\right) = \mathrm{average}\left(\mathrm{L}\left(\mathrm{s},\mathrm{a}\right)\right)$.
9: **end for**
10: **for** each state s in the episode **do**
11: $\pi\left(s\right) = \underset{a}{\mathrm{argmax}}\ Q^{\pi}\left(s,a\right)$.
12: **end for**
13: **end while**

Note that step 3 makes sure all the pairs would appear as the beginings of episodes.

Now, could we avoid the human efforts in step 3 and 4? Apparently, if we use greedy strategy, it won't work. So, we should make our policy non-greedy, i.e, we hope our policy to be $\epsilon - soft$.

Algorithm 6 Monte Carlo control without explorating starts.

1: Initialize policy π to be $\epsilon -$ greedy, state-value function V^{π} and empty list $L\left(a,s\right)\forall s \in S\ and\ a \in A\left(s\right)$.
2: **while** True **do**
3: Generate an episode with π.
4: **for** each pair s and a in the episode **do**
5: Let G be the return of first occurance of the pair s and a.
6: Append G to $L\left(s,a\right)$.
7: $Q\left(s,a\right) = \mathrm{average}\left(\mathrm{L}\left(\mathrm{s},\mathrm{a}\right)\right)$.
8: **end for**
9: **for** each state s in the episode **do**
10: $a^* = \underset{a}{\mathrm{argmax}}\ Q^{\pi}\left(s,a\right)$.
11:
12: **for** $a \in A$ **do**
13: $\pi\left(a|s\right) = \begin{cases} 1 - \epsilon + \dfrac{\epsilon}{|A|} & \text{if a} = a^* \\ \dfrac{\epsilon}{|A|} & \text{if a} \neq a^* \end{cases}$
14: **end for**
15: **end for**
16: **end while**

This seems to be a good strategy. However, could it better the original policy? All we want to know is if $Q^{\pi}\left(s, \pi'\left(s\right)\right) \geq V^{\pi}\left(s\right) \forall s \in S$?

$$
\begin{aligned}
Q^{\pi}\left(s, \pi'\left(s\right)\right) =& \sum_{a} \pi'\left(a|s\right) Q^{\pi}\left(s, a\right) \\
=& \frac{\epsilon}{|A|} \sum_{a} Q^{\pi}\left(s, a\right) + \left(1-\epsilon\right) \underset{a}{max} Q^{\pi}\left(s, a\right) \\
\geq& \frac{\epsilon}{|A|} \sum_{a} Q^{\pi}\left(s, a\right) + \left(1-\epsilon\right) \sum_{a} \frac{\pi\left(a|s\right) - \frac{\epsilon}{|A|}}{1-\epsilon} Q^{\pi}\left(s, a\right) \\
=& \frac{\epsilon}{|A|} \sum_{a} Q^{\pi}\left(s, a\right) - \frac{\epsilon}{|A|} \sum_{a} Q^{\pi}\left(s, a\right) + \sum_{a} \pi\left(a|s\right) Q^{\pi}\left(s, a\right) \\
=& V^{\pi}\left(s\right)
\end{aligned}
\tag{3.3}
$$

So, this algorithm works.

3.2 Off-policy control

As stated previously, we estimate the target policy, π, while following another policy, π'. This strategy could be successful under the idea of *important sampling*. In general, we may want to estimate $E\left(f(X)\right)$ for any possible function f. Nevertheless, sometimes it's hard to sample, π, of the random variable X. Thus, we use another distribution π' to estimate it.

$$
\begin{aligned}
E_{\pi}\left(f(X)\right) =& \int_{\Omega} f\left(x\right) \pi\left(x\right) dx \\
=& \int_{\Omega} f\left(x\right) \frac{\pi\left(x\right)}{\pi'\left(x\right)} \pi'\left(x\right) dx \\
=& E_{\pi'}\left(f\left(X\right) \frac{\pi\left(X\right)}{\pi'\left(X\right)}\right) \\
\underset{N \longrightarrow \infty}{\approx}& \sum_{i=1}^{N} \frac{f\left(x^{i}\right) \frac{\pi\left(x^{i}\right)}{\pi'\left(x^{i}\right)}}{N} \text{ where } \mathrm{x^{i}} \sim \pi'.
\end{aligned}
\tag{3.4}
$$

Define

$$
w_i = \frac{\pi\left(x^{i}\right)}{\pi'\left(x^{i}\right)} \tag{3.5}
$$

then equation (3.1) could be written as

$$
\sum_{i=1}^{N} \frac{f\left(x^{i}\right) w_i}{N} \tag{3.6}
$$

Obviously, this is an unbiased estimator of the target. However, variance of the estimation would be pretty large if the π' is not selected well. Generally, we would try to estimate the target by

$$E_{\pi}\left(f\left(X\right)\right) \approx \sum_{i=1}^{N} \frac{w_i}{\sum_{j=1}^{N} w_j} f\left(x^i\right) \tag{3.7}$$

This could effectively reduce the estimated variance.
Back to MDP, since under policy, π

$$\begin{aligned}
\Pr\left(\mathrm{a_t}, \mathrm{s_{t+1}}, \ldots \mathrm{s_T} | \mathrm{s_t}\right) &= \prod_{k=t}^{T-1} \pi\left(a_k|s_k\right) \Pr\left(\mathrm{s_{k+1}}|\mathrm{s_k}, \mathrm{a_k}\right) \\
&= \prod_{k=t}^{T-1} \pi\left(a_k|s_k\right) P_{s_k, s_{k+1}}^{a_k}
\end{aligned} \tag{3.8}$$

Similarly, under policy, π'

$$\Pr\left(\mathrm{a_t}, \mathrm{s_{t+1}}, \ldots \mathrm{s_T} | \mathrm{s_t}\right) = \prod_{\mathrm{k=t}}^{\mathrm{T-1}} \pi'\left(\mathrm{a_k}|\mathrm{s_k}\right) \mathrm{P}_{\mathrm{s_k}, \mathrm{s_{k+1}}}^{\mathrm{a_k}}. \tag{3.9}$$

Now, the weight then becomes

$$w = \frac{\prod_{k=t}^{T-1} \pi\left(a_k|s_k\right)}{\prod_{k=t}^{T-1} \pi'\left(a_k|s_k\right)}. \tag{3.10}$$

It has a remarkable property, i.e it doesn't need the model(conditional probability).

To estimate $V^{\pi}\left(s\right)$, let w_i be the weight of the ith episode which contains state s and the R^i represents the discount reward of the ith episode. Then the off-policy estimates $V^{\pi}\left(s\right) = \frac{\sum_i w_i R^i}{\sum_i w_i}$.

In practice, when we want to estimate a deterministic policy π, we would use the above approach to estimate $Q^{\pi}\left(s, a\right)$, and then use following algorithm to estimate a deterministic policy π:

Algorithm 7 Off-policy Monte Carlo Control

Input: behavior policy π'.

Output: deterministic target policy π.

```
1: ∀s ∈ S, a ∈ A, initialize Q^π(s,a) = 0, N(s,a) = 0, D(s,a) = 0 and π be
   any arbitrary deterministic policy.
2: while True do
3: Use policy π' to generate an episode : s_0, a_0, r_1, s_1, a_1..., r_T, s_T.
4: Let τ be the latest time t where a_t ≠ π(s_t)
5:     for each pair s, a appearing in the episode after τ do
6:         Let t be the first occurence of s, a after τ.
7:         w = ∏_{k=t+1}^{T-1} 1/π'(a_k|s_k).
8:         N(s,a) = N(s,a) + wR_{t+1}.
9:         D(s,a) = D(s,a) + w.
10:        Q(s,a) = N(s,a)/D(s,a)
11:    end for
12:    for each s ∈ S do
13:        Let π(s) = argmax_a Q(s,a)
14:    end for
15: end while
```

Note that step 6 indicates we have to consider the process after s and a, so that we are actually estimating $Q^{\pi}(s,a)$ not $V^{\pi}(s)$.

3.3 Incremental Implementation

In last subsection, we state that we could use this estimation, $V^{\pi} = \frac{\sum w_i R^i}{\sum w_i}$, to estimate the value-state function. However, in practice, we could use incremental implementation to write down the estimation as

$$V_{n+1}^{\pi} = V_n^{\pi} + \frac{w_{n+1}}{W_{n+1}} \left[R^{n+1} - V_n^{\pi} \right] \tag{3.11}$$

where

$$W_{n+1} = W_n + w_{n+1} \tag{3.12}$$

Thus Algorithm 2 could be modified as

Algorithm 8 Off-policy Monte Carlo Control

Input: behavior policy π'.
Output: deterministic target policy π.

```
∀s ∈ S, a ∈ A, initialize Q^{pi}(s,a) = 0, N(s,a) = 0, D(s,a) = 0 and π be
  any arbitrary deterministic policy.
while True do
Use policy π' to generate an episode : s_0, a_0, r_1, s_1, a_1..., r_T, s_T.
Let w = 1 and R = 0.
    for t = T−1, T−2, ...0 do
        R = γR + r_{t+1}.
        D(s_t, a_t) = D(s_t, a_t) + w.
        Q(s_t, a_t) = Q(s_t, a_t) + W / D(s_t, a_t) [R − Q(s_t, a_t)]
        π(s_t) = argmax_a Q(s_t, a_t)
        if a_t ≠ π(s_t) then
            break
        end if
        W = W 1 / π(a_t|s_t)
    end for
end while
```

4 Temporal Difference

Recall that Monte Carlo approach estimate the target by *sampling(Monte Carlo)* and dynamic programming approach estimate the target by *bootstrapping*. Temporal difference(TD) approach combines the two methods to estimate the target.

In section 3, we see that the estimate could be done by

$$V^{\pi}(s) = V^{\pi}(s) + \alpha\left[R_{t+1} - V^{\pi}(s)\right] \tag{4.1}$$

for some α. Monte Carlo approach uses the *sampled return* to update the value-state function. Different from Monte Carlo approach, temporal difference method use the result estimated before to estimate, i.e.

$$V^{\pi}(s_t) = V^{\pi}(s_t) + \alpha\left[r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)\right]. \tag{4.2}$$

Theoretically, the difference between temporal difference and monte carlo approach could be identified by the following equations.

$$\begin{aligned} V^{\pi}(s_{t+1}) =& E_{\pi}\left[R_{t+1} | s_t = s\right] \\ =& E_{\pi}\left[r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s\right] \end{aligned} \tag{4.3}$$

Monte Carlo approach estimates the function by the first equation of (4.3); Nevertheless, the temporal difference approach estimate it by the second one. The complete algorithm is as follows:

Algorithm 9 Temporal Difference

Input: target policy π'.

Initialize $V(s)$ arbitrary.
Initialize s.
for each step, s, of an episode **do**
    Let a be actopm given by state s and π.
    Take action a and observe s' and r.
    $V(s) = V(s) + \alpha [r + \gamma V(s') - V(s)]$
    $s' = s$
end for

4.1 On-policy TD control–Sarsa algorithm

Sarsa algorithm uses the temporal difference to estimate the action-value function Q^{π} by the same way. And it uses the $\epsilon - greedy$ strategy.

Algorithm 10 Sarsa algorithm

Input: arbitrary target policy π and α.

Initialize $Q(s, a)$ arbitrarily.
while True **do**
    Initialize s.
    Choose a from s and π derived from Q.
    while s is not terminal state **do**
        Take action a and observe s' and r'.
        Choose a' from s' and π derived from Q.
        $Q(s, a) = Q(s, a) + \alpha [r' + \gamma Q(s', a') - Q(s, a)]$
        $s = s'$ and $a = a'$.
    end while
end while

4.2 Off-policy TD control–Q–learning

Sarsa algorithm follows the same policy π for every step. Hoever, Q–learning directly estimate Q^*, instead of Q^{π}(Sarsa algorithm). Although both algorithm converge to optimal policy π^*, Q–learning dramatically simplifies the analysis of the algorithm.

Algorithm 11 Q–learning algorithm

Input: arbitrary target policy π and α.

```
Initialize Q(s, a) arbitrarily.
while True do
    Initialize s.
    while s is not terminal state do
        Choose a from s and π derived from Q.
        Take action a and observe s' and r'.
        Q(s, a) = Q(s, a) + α [r' + γ max_{a'} Q(s', a') − Q(s, a)].
        s = s'.
    end while
end while
```