Lecture 7B: Review of Model-free Methods: Monte Carlo, SARSA (n-step) and Q-Learning

Dr. Wen Fuxi

Introduction

"If one had to identify one idea as central and novel to RL, it would undoubtedly be TD learning."

Richard Sutton

- Can update value function only by a single sample pair (In comparison, MC must update at the end of each episode)
- Can be implemented in an online, fully incremental fashion
- Bootstrapping is helpful to increase estimate accuracy with only a few samples and accelerate convergence

1. Comparison of MC, Sarsa and Q-learning

2. On-policy and Off-policy

Deriving the Bellman equation

Consider a random trajectory:

$$S_t \xrightarrow{A_t} R_{t+1}, S_{t+1} \xrightarrow{A_{t+1}} R_{t+2}, S_{t+2} \xrightarrow{A_{t+2}} R_{t+3}, \ldots$$

The return G_t can be written as

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots, \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \ldots \right), \\ &= R_{t+1} + \gamma G_{t+1}, \end{aligned}$$

Then, it follows from the definition of the state value that

$$\begin{aligned} v_\pi(s) &= \mathbb{E}\left[G_t \mid S_t = s \right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right] \\ &= \mathbb{E}\left[R_{t+1} \mid S_t = s \right] + \gamma \mathbb{E}\left[G_{t+1} \mid S_t = s \right] \end{aligned}$$

TD learning of state values - The idea of the algorithm

First, a new expression of the Bellman equation.

The definition of the state value of π is

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s], \quad s \in \mathcal{S} \tag{1}$$

where G is the discounted return. Since

$$\mathbb{E}[G_{t+1} \mid S = s] = \sum_a \pi(a \mid s) \sum_{s'} p\left(s' \mid s, a\right) v_\pi\left(s'\right) = \mathbb{E}\left[v_\pi\left(S_{t+1}\right) \mid S_t = s\right],$$

where $S_{t+1} = s'$ is the next state, we can rewrite (1) as

$$v_\pi(s) = \mathbb{E}\left[R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) \mid S_t = s\right], \quad s \in \mathcal{S}. \tag{2}$$

Equation (2) is another expression of the Bellman equation. It is sometimes referred to as the Bellman expectation equation, an important tool for designing and analyzing TD algorithms.

TD Policy Evaluation

Bootstrapping technique in TD

$$v^{\pi}(s_t) \leftarrow (1-\alpha)v^{\pi}(s_t) + \alpha \cdot G_t$$

Bootstrapping

Return : $G_t = \sum_{i=0}^{+\infty} \gamma^i r_{t+i}$

- Return G_t comes from interaction with the environment

- Recall the self-consistency condition

$$\mathbb{E}_{\pi}\{G_t|s\} = v^{\pi}(s) = \mathbb{E}_{\pi}\{r + \gamma v^{\pi}(s')\}$$

Sample return to replace the actual expected return

- Use estimate $V^{\pi}(s)$ to replace true value $v^{\pi}(s)$

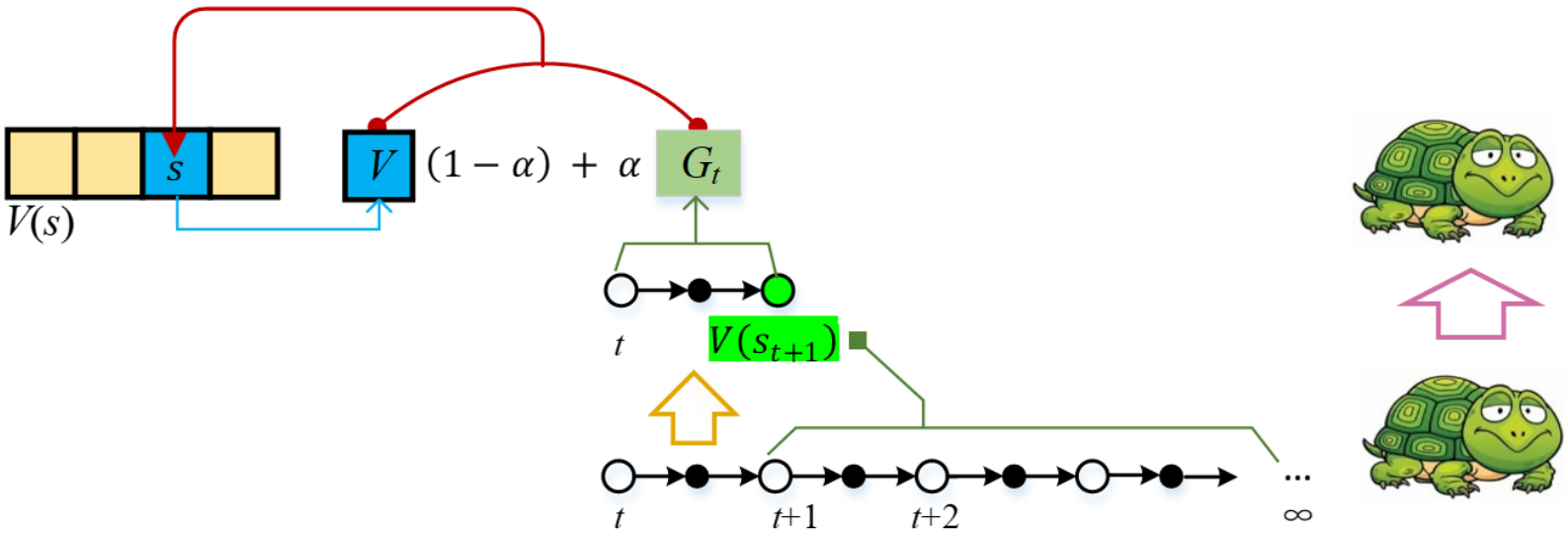
$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha\big(\underbrace{r_{t+1} + \gamma V^{\pi}(s_{t+1})}_{G_t} - V^{\pi}(s_t)\big)$$

New estimate

Reward Prediction Error

TD Policy Evaluation

One-step TD for the State-value function



Use new sample $(s_t, a_t, r_{t+1}, s_{t+1})$ to update state-value $V(s)$

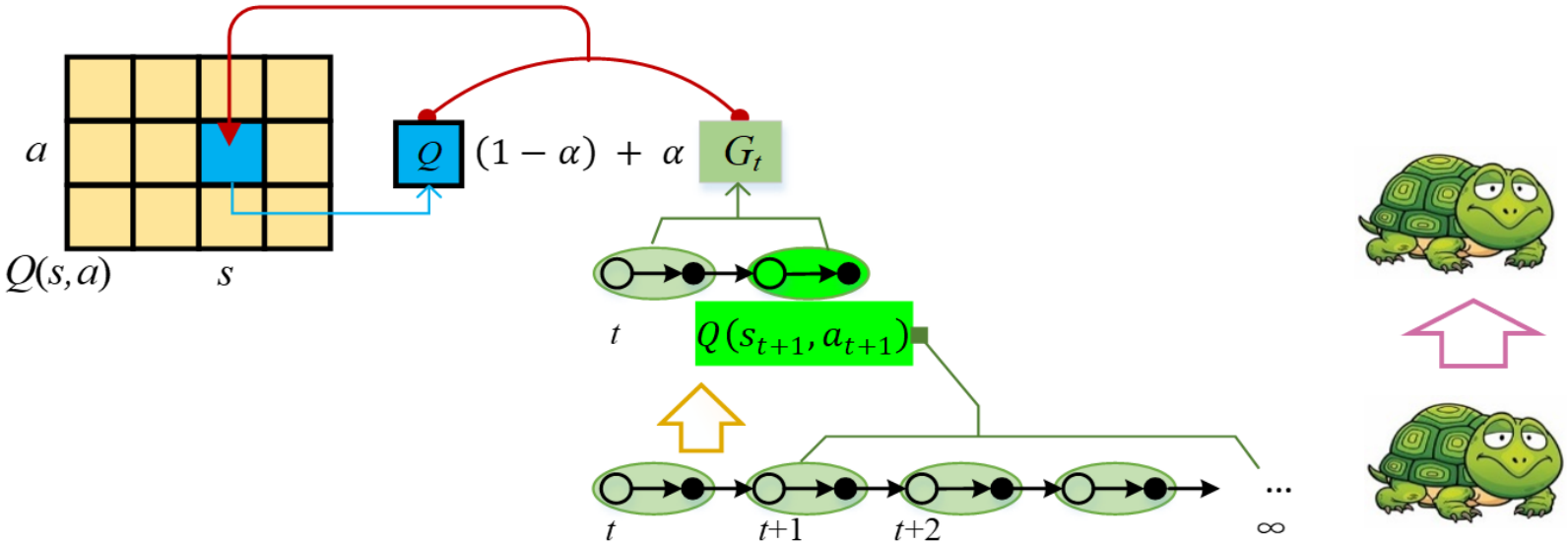
$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha\left(r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)\right)$$

New Experience

History estimate

TD Policy Evaluation

One-step TD for the Action-value function



Use new sample $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ to update state-value $Q(s,a)$

$$Q^{\pi}(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \alpha\left(r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) - Q^{\pi}(s_t, a_t)\right)$$

New Experience

History estimate

Sarsa - Algorithm

Sarsa (State-action-reward-state-action)

- On-policy one-step TD
- Update whenever encounter a 5-tuple ($s_t, a_t, r_t, s_{t+1}, a_{t+1}$)

Policy Evaluation (PEV)

Estimate action-value function $Q(s, a)$ with N samples

Loop N times

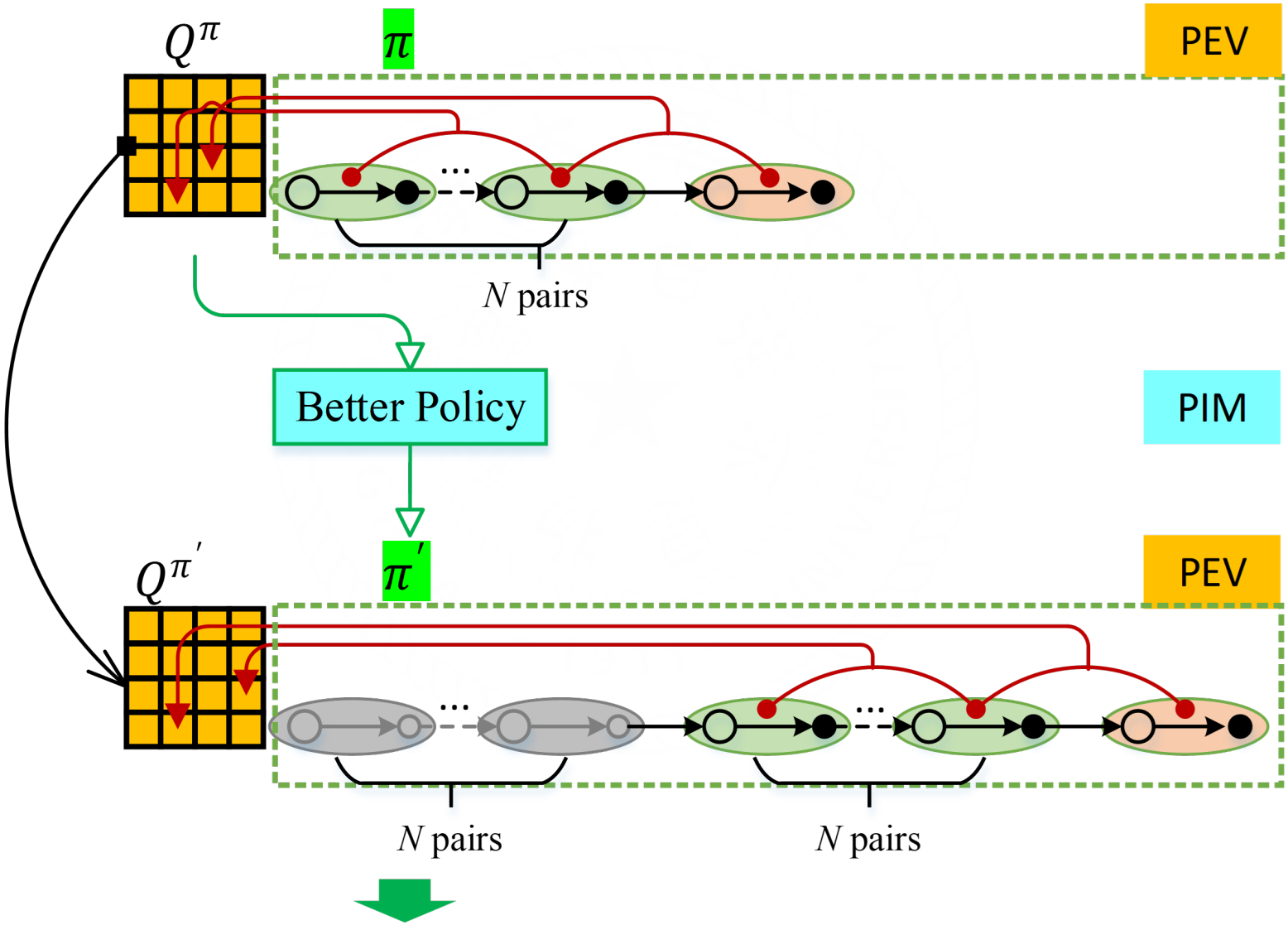
$$Q^{\pi}(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \alpha\left(r_t + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) - Q^{\pi}(s_t, a_t)\right)$$

End

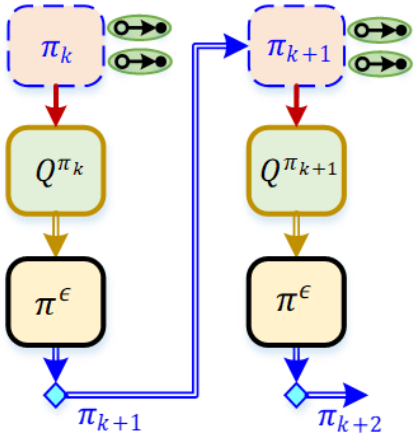
Policy Improvement (PIM)

- Find a better policy satisfying the policy improvement theorem
- Update ϵ-greedy policy for action-value function

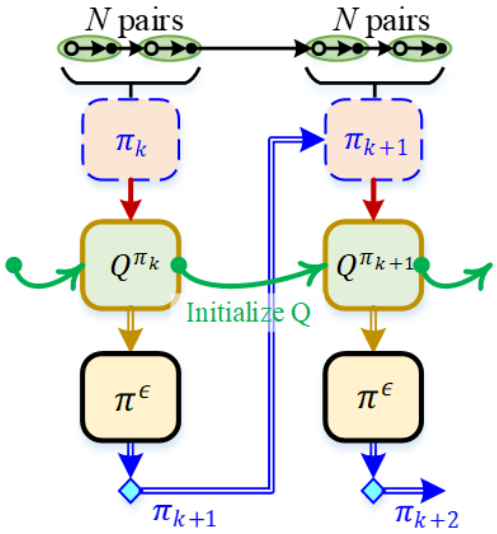
Flow chat of Sarsa



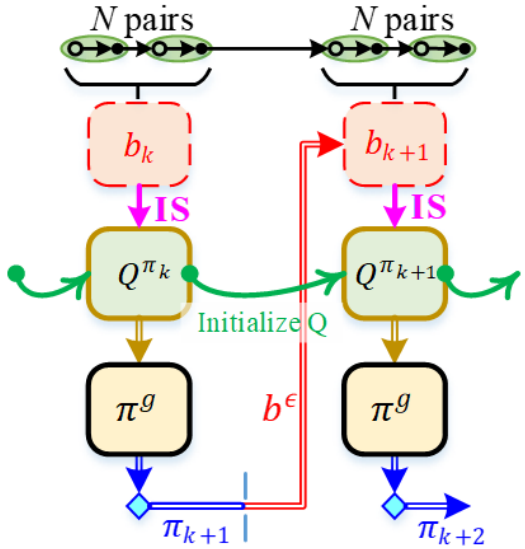
MC, Sarsa and Q-learning



- **MC**
- Use episodes in PEV
- On-policy/off-policy
- Often no initialization

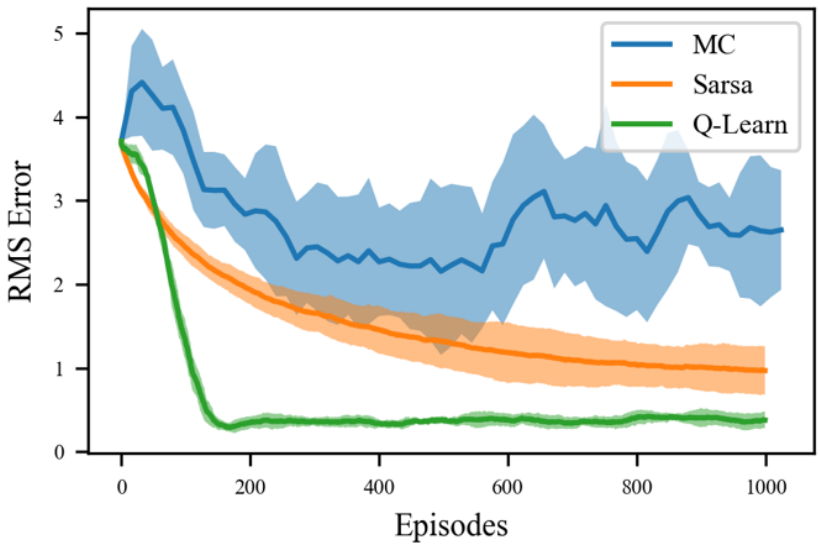


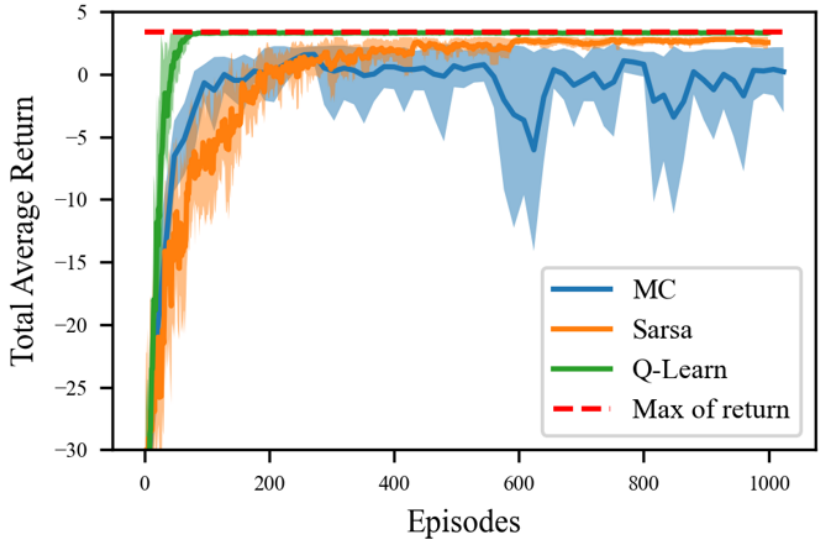
- **Sarsa**
- Use N pairs/PEV
- On-policy
- With initialization



- **Off-policy TD**
- Q-learning = **1** pair/PEV
- Use pairs
- With initialization

Comparison of MC, Sarsa and Q-learning





Mostly, Q-learning outperforms Sarsa and MC

TD learning of action values: n-step Sarsa

n-step Sarsa can unify Sarsa and Monte Carlo learning

The definition of action value is

$$q_\pi(s,a) = \mathbb{E}\left[G_t \mid S_t = s, A_t = a\right].$$

The discounted return G_t can be written in different forms as

$$\begin{aligned}
\text{Sarsa} \longleftarrow \quad & G_t^{(1)} = R_{t+1} + \gamma q_\pi\left(S_{t+1}, A_{t+1}\right), \\
& G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 q_\pi\left(S_{t+2}, A_{t+2}\right), \\
& \vdots \\
\text{n-step Sarsa} \longleftarrow \quad & G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n q_\pi\left(S_{t+n}, A_{t+n}\right), \\
& \vdots \\
\text{MC} \longleftarrow \quad & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots
\end{aligned}$$

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.

TD learning of action values: n-step Sarsa

- Sarsa aims to solve

$$q_\pi(s,a) = \mathbb{E}\left[G_t^{(1)} \mid s,a\right] = \mathbb{E}\left[R_{t+1} + \gamma q_\pi\left(S_{t+1}, A_{t+1}\right) \mid s,a\right].$$

- MC learning aims to solve

$$q_\pi(s,a) = \mathbb{E}\left[G_t^{(\infty)} \mid s,a\right] = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid s,a\right].$$

- An intermediate algorithm called n-step Sarsa aims to solve

$$q_\pi(s,a) = \mathbb{E}\left[G_t^{(n)} \mid s,a\right] = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n q_\pi\left(S_{t+n}, A_{t+n}\right) \mid s,a\right].$$

- The algorithm of n-step Sarsa is

$$\begin{aligned} q_{t+1}\left(s_t,a_t\right) &= q_t\left(s_t,a_t\right) \\ &- \alpha_t\left(s_t,a_t\right)\left[q_t\left(s_t,a_t\right) - \left[r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^n q_t\left(s_{t+n},a_{t+n}\right)\right]\right]. \end{aligned}$$

  - n-step Sarsa becomes the (one-step) Sarsa algorithm when $n = 1$.
  - n-step Sarsa becomes the MC learning algorithm when $n = \infty$.

Q-learning - Algorithm

The Q-learning algorithm is

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

$$q_{t+1}(s, a) = q_t(s, a), \quad \forall (s, a) \neq (s_t, a_t),$$

Q-learning is very similar to SARSA. They are different only in terms of the TD target:

- The TD target in Q-learning is $r_{t+1} + \gamma \max_{a \in \mathcal{A}} q_t(s_{t+1}, a)$
- The TD target in Sarsa is $r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})$

Q-learning - Algorithm

What does Q-learning do mathematically?

It aims to solve

$$q(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_a q\left(S_{t+1}, a\right) \mid S_t = s, A_t = a\right], \quad \forall s, a$$

This is the Bellman optimality equation expressed in terms of action values.

As a result, Q-learning can directly estimate the optimal action values, rather than the action values of a given policy.

Off-policy vs on-policy

Before further studying Q-learning, we first introduce two important concepts:

On-policy learning and off-policy learning.

There exist two policies in a TD learning task:

- The behavior policy is used to generate experience samples.
- The target policy is constantly updated toward an optimal policy.

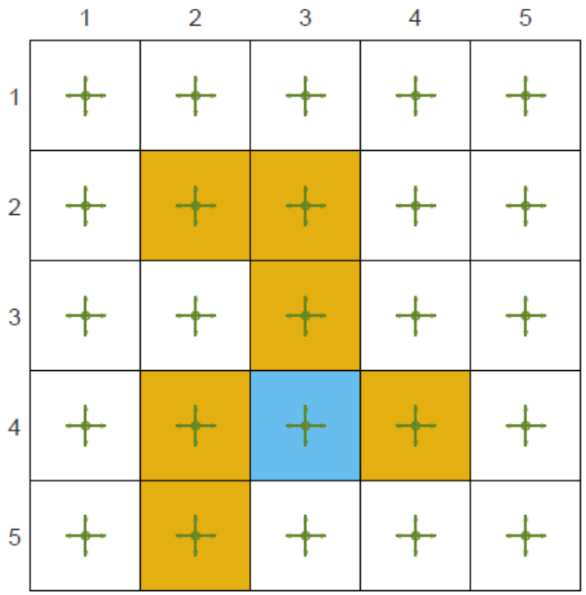
On-policy vs off-policy

- When the behavior policy is the same as the target policy, this kind of learning is called on-policy.
- When they are different, the learning is referred to as off-policy.

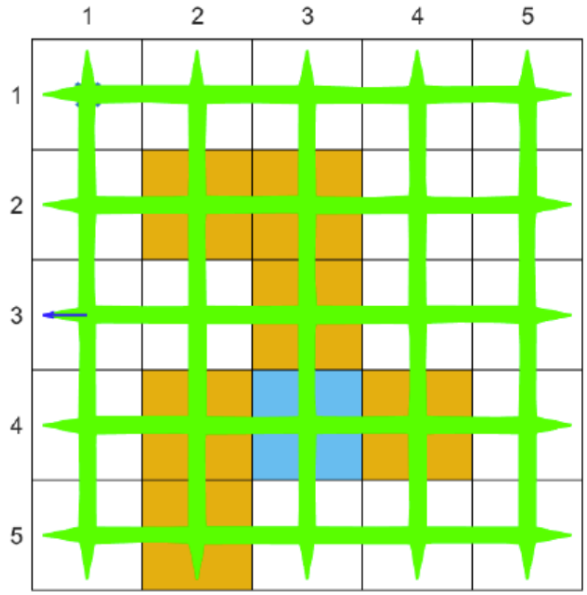
Off-policy vs on-policy

Advantages of off-policy learning:

- It can search for optimal policies based on the experience samples generated by any other policies.
- Example: The behavior policy is exploratory, so that we can generate episodes visiting every state-action pair sufficiently many times.



(a) Exploratory behavior policy



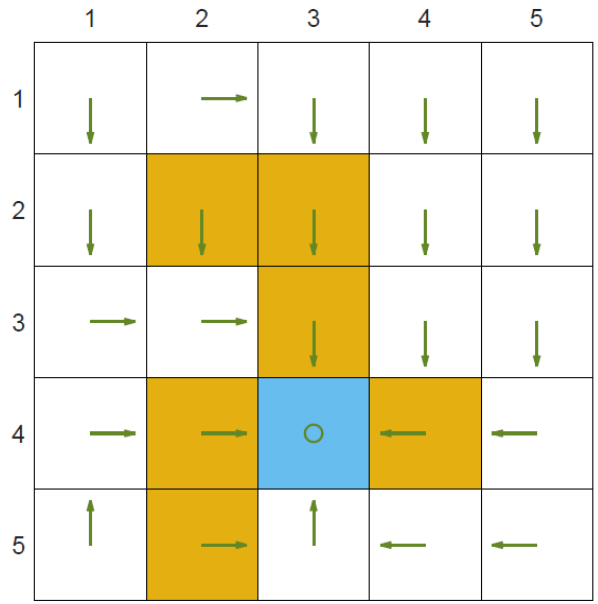
(b) Generated episode

Q-learning - Examples

Task description:

- The task in these examples is to find an optimal policy for all the states.
- The reward setting is $r_{\text{boundary}} = r_{\text{forbidden}} = -1$, and $r_{\text{target}} = 1$. The discount rate is $\gamma = 0.9$. The learning rate is $\alpha = 0.1$.

Ground truth: an optimal policy and the corresponding optimal state values.



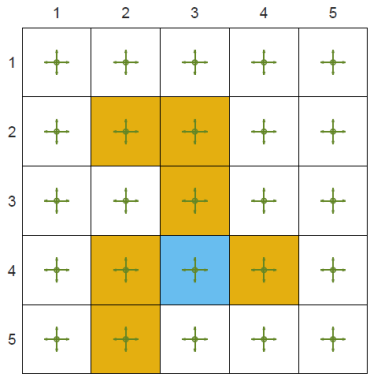
(a) Optimal policy

	1	2	3	4	5
1	5.8	5.6	6.2	6.5	5.8
2	6.5	7.2	8.0	7.2	6.5
3	7.2	8.0	10.0	8.0	7.2
4	8.0	10.0	10.0	10.0	8.0
5	7.2	9.0	10.0	9.0	8.1

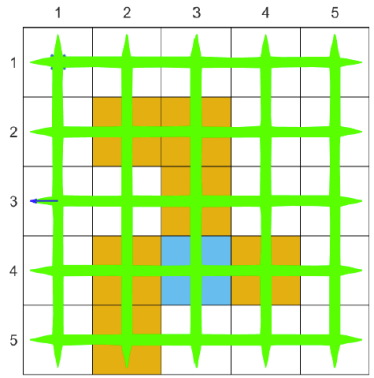
(b) Optimal state value

Q-learning - Examples

The behavior policy and the generated experience (10^5 steps):

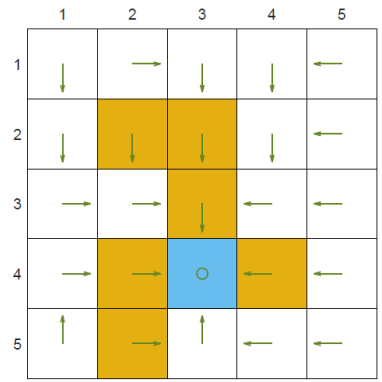


(a) Behavior policy

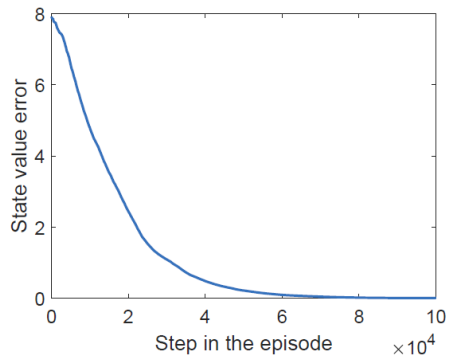


(b) Generated episode

The policy found by off-policy Q-learning:



(a) Estimated policy

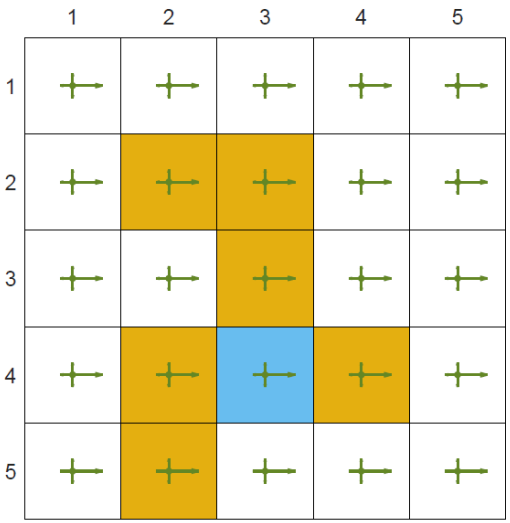


(b) State value error

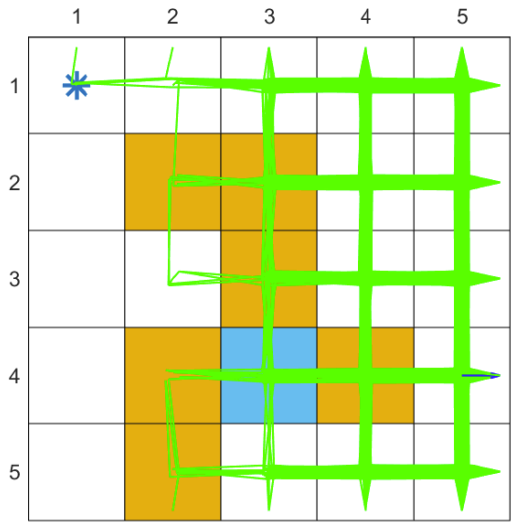
Q-learning - Examples

The importance of exploration: episodes of 10^5 steps

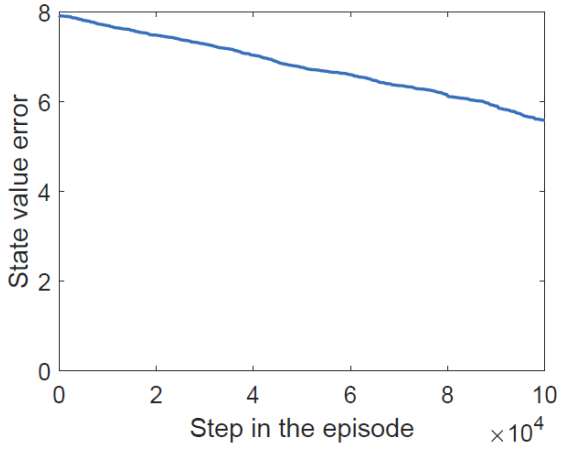
If the policy is not sufficiently exploratory, the samples are not good.



(a) Behavior policy $\epsilon = 0.5$

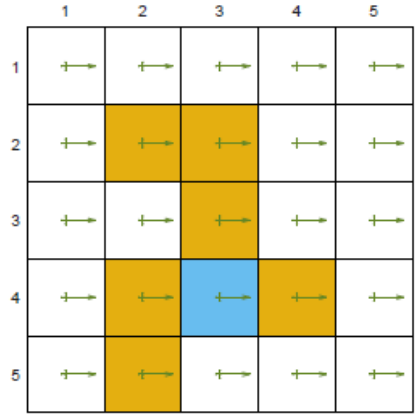


(b) Generated episode



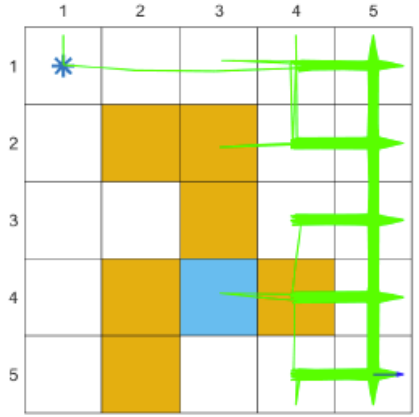
(c) Q-learning result

Q-learning - Examples

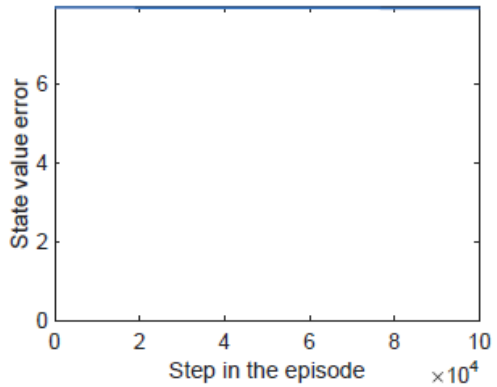


(a) Behavior policy

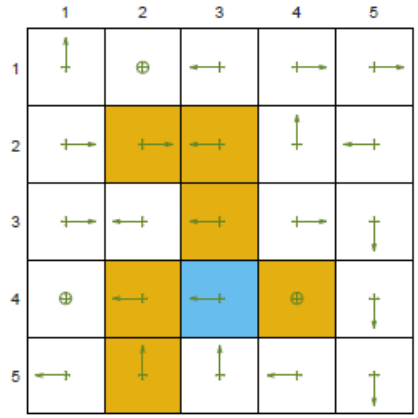
$\epsilon = 0.1$



(b) Generated episode

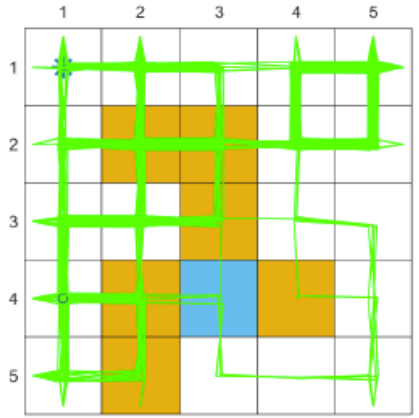


(c) Q-learning result

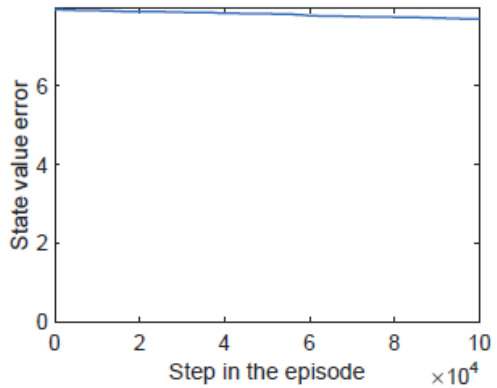


(a) Behavior policy

$\epsilon = 0.1$



(b) Generated episode



(c) Q-learning result

On-policy and Off-policy

On-policy strategy

- Use the same policy for sampler and learner
- Can use ϵ-greedy policy to increase exploitation
- Can shift to the optimal policy by gradually reducing ϵ

Off-policy strategy

- Use different policies in the sampler and the learner

  - Behavior policy b for more exploration in the environment

  - Target policy π to determine the optimal policy

- Use the importance sampling (IS) technique to estimate the value function under π by using samples from b

On-policy and Off-policy

Off-policy RL provides a better compromise between exploitation and exploration.

The importance sampling (IS) transformation allows the estimation of the value function under the target policy π from the data samples of the behavior policy b.

For the state-value function, the current state is known, i.e., $S_t = s$, and its following state and action are random variables.

Suppose that we have already recorded a sample trajectory from time t to time $t+1$.

The probabilities of generating such a trajectory under π and b are

$$d_\pi\left(A_t, S_{t+1}\right) = \Pr\left\{A_t = a, S_{t+1} = s' \mid \pi, s\right\} = \pi(a \mid s) p\left(s' \mid s, a\right)$$
$$d_b\left(A_t, S_{t+1}\right) = \Pr\left\{A_t = a, S_{t+1} = s' \mid b, s\right\} = b(a \mid s) p\left(s' \mid s, a\right)$$

where A_t and S_{t+1} are two random variables, and $d\left(A_t, S_{t+1}\right)$ is the probability mass function of their bivariate distribution.

Importance sampling in Off-policy TD

Define one-step IS ratio as

$$\rho_{t:t} = \frac{d_\pi\left(a_t, s_{t+1}\right)}{d_b\left(a_t, s_{t+1}\right)} = \frac{\pi\left(a_t \mid s_t\right)}{b\left(a_t \mid s_t\right)}$$

Expected return under π can be estimated by using samples generated by b

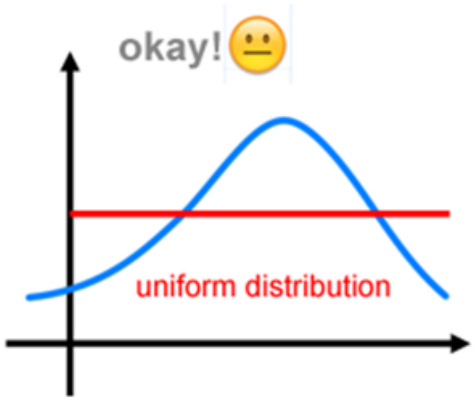
$$\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi\left\{r_t + \gamma v_\pi\left(s_{t+1}\right)\right\} \\
&= \sum_{\left(a_t, s_{t+1}\right) \in \mathcal{A} \times \mathcal{S}} d_\pi\left(a_t, s_{t+1}\right)\left(r_t + \gamma v_\pi\left(s_{t+1}\right)\right) \\
&= \sum_{\left(a_t, s_{t+1}\right) \in \mathcal{A} \times \mathcal{S}} \frac{d_\pi\left(a_t, s_{t+1}\right)}{d_b\left(a_t, s_{t+1}\right)} d_b\left(a_t, s_{t+1}\right)\left(r_t + \gamma v_\pi\left(s_{t+1}\right)\right) \\
&= \sum_{\left(a_t, s_{t+1}\right) \in \mathcal{A} \times \mathcal{S}} d_b\left(a_t, s_{t+1}\right)\left[\rho_{t:t}\left(r_t + \gamma v_\pi\left(s_{t+1}\right)\right)\right] \\
&= \mathbb{E}_b\left\{\rho_{t:t}\left(r_t + \gamma v_\pi\left(s_{t+1}\right)\right)\right\}
\end{aligned}$$

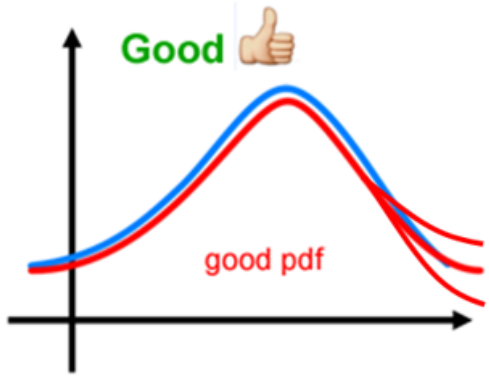
Importance sampling in Off-policy TD

One-step TD prediction with IS ratio

$$V_\pi(s_t) \leftarrow V_\pi(s_t) + \alpha\Big[\rho_{t:t}\big(r_t + \gamma V_\pi(s_{t+1})\big) - V_\pi(s_t)\Big]$$

- Often has low quality because of its high variance
- A short explanation: $\rho_{t:t} \to \infty$ if $b(a \mid s) \to 0$
  - Sample becomes very important, and its noise is infinitely amplified.
  - Bootstrapping mechanism loses effect.





Importance sampling in Off-policy TD

Method for variance reduction

$$V_\pi(s_t) \leftarrow V_\pi(s_t) + \alpha\Big[\rho_{t:t}\big(r_t + \gamma V^\pi(s_{t+1})\big) - V_\pi(s_t)\Big]$$

$$V_\pi(s_t) \leftarrow V_\pi(s_t) + \alpha\Big[\rho_{t:t}\big(r_t + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)\big)\Big]$$

$\rho_{t:t}\times$ One-step TD error

The amplification effect from the uncertain IS ratio is decreased.

Key of Proof

$$\mathbb{E}_b\{V_\pi(s)\} = \mathbb{E}_b\{\rho_{t:t} V_\pi(s)\}$$

Importance sampling in Off-policy TD

$$\mathbb{E}_b\{V_\pi(s)\} = \mathbb{E}_b\{\rho_{t:t} V_\pi(s)\}$$

Proof:

- $\mathbb{E}_b\{\cdot\}$ is an abbreviation of $\mathbb{E}_b\{\cdot \mid s\}$ for brevity
- $V_\pi(s)$ can be viewed as a constant because s is known

$$\begin{aligned}
\mathbb{E}_b\{\rho_{t:t} V_\pi(s)\} &= \mathbb{E}_{a\sim b}\left\{\frac{\pi(a \mid s)}{b(a \mid s)}\right\} \mathbb{E}_{a\sim b}\{V_\pi(s)\} \\
&= \sum_a b(a \mid s) \frac{\pi(a \mid s)}{b(a \mid s)}\Bigg|_s \cdot \mathbb{E}_{a\sim b}\{V_\pi(s)\} \\
&= \sum_a \pi(a \mid s)\, \mathbb{E}_{a\sim b}\{V_\pi(s)\} \\
&= 1 \cdot \mathbb{E}_b\{V_\pi(s)\}
\end{aligned}$$