Subject: Westlake University, Reinforce Learning, Lecture 7, Temporal-Difference Learning

Date: from January 24, 2025 to February 12, 2025

Contents

Lecture 7, Temporal-Difference Learning

Bilibili:Lecture 7, Temporal-Difference Learning

Introduction and Outline

在 Lecture 4 中我们学习了 **model-based** 算法 Value & Policy iteration，随后在 Lecture 5 中学习了 **model-free** 算法 Monte Carlo Learning. 为解决 MC-learning 的 **non-incremental** 的更新模式带来模型算法的下降，在 Lecture 6 中引入了 RM 算法. 本节中将借助 RM 算法的思想，正式引入 **Temporal-Difference Learning(TD learning)**.

$$\text{model-based} \to \text{model-free}; \quad \text{non-incremental} \to \text{incremental}$$

TD learning 是强化学习中的一大类方法，本节将介绍其中四种算法

1. **basic TD** algorithm: 最基础的 TD 算法，用于评价给定 policy 的 state values
2. **Sarsa** algorithm: 用于评价给定 policy 的 action values
3. ***n*-step Sarsa** algorithm: Sarsa 和 MC learning 是 n-step Sarsa 的两种特殊情形
4. **Q-learning** algorithm: 用于直接求解 BOE 以获得最优策略

Motivating examples

Motivating examples

Example 1: mean estimation(review) 对于 mean estimation 问题 $w = \mathbb{E}[X]$，令 $g(w) = w - \mathbb{E}[X]$，转化为求解 $g(w) = 0$，同时为使用 RM 算法，得到采有噪观测为:

$$\tilde{g}(w,\eta) = w - x = (w - \mathbb{E}[X]) + (\mathbb{E}[X] - x) \triangleq g(w) + \eta$$

再使用 RM 算法即可求解 mean estimation 问题:

$$w_{k+1} = w_t - \alpha_t \tilde{g}(w_t, \eta_t) = w_t - \alpha_t(w_t - x_t)$$

Example 2: mean estimation of a function $v(X)$ (a little bit more complex)

此时求解 $w = \mathbb{E}[v(X)]$，其中 $v(\cdot)$ 为函数. 类似地令 $g(w) = w - \mathbb{E}[v(X)]$ 可以得到:

$$\tilde{g}(w,\eta) = w - v(x) = (w - \mathbb{E}[v(X)]) + (\mathbb{E}[v(X)] - v(x)) \triangleq g(w) + \eta$$
$$w_{k+1} = w_t - \alpha_t \tilde{g}(w_t, \eta_t) = w_t - \alpha_t[w_t - v(x_t)]$$

Example 3: mean estimation of a two random variables (complex situation)

此时我们求解 $w = \mathbb{E}[R + \gamma v(X)]$，其中 R, X 均为随机变量，γ 为常数，$v(\cdot)$ 为函数，记 $\{x\}, \{r\}$ 分别为 X, R 的采样，那么类似地令 $g(w) = w - \mathbb{E}[R + \gamma v(X)]$ 可以得到:

$$\tilde{g}(w,\eta) = w - [r + \gamma v(x)] = (w - \mathbb{E}[R + \gamma v(X)]) + (\mathbb{E}[R + \gamma v(X)] - [r + \gamma v(x)]) \triangleq g(w) + \eta$$
$$w_{k+1} = w_t - \alpha_t \tilde{g}(w_t, \eta_t) = w_t - \alpha_t[w_t - (r_t + \gamma v(x_t))]$$

TD Learning

TD Learning of state values

TD Learning of state values – Basic TD

最基础的 TD learning 算法用于 policy evaluation，即估计给定 policy 的 state value.

Data TD learning 算法是 model-free 的，因此依赖于数据/experience，此处将其记为

$$(s_0, r_1, s_1, \ldots, s_t, r_{t+1}, s_{t+1}, \ldots) \text{ or } \{(s_t, r_{t+1}, s_{t+1})\}_t$$

其是由给定的 policy π 生成的 state-reward 序列.

Algorithm 估计 policy π 的 state value 的 TD learning 算法如下：

$$v_{t+1}(s_t) = v_t(s_t) - \alpha_t(s_t)\left[v_t(s_t) - [r_{t+1} + \gamma v_t(s_{t+1})]\right] \tag{1a}$$

$$v_{t+1}(s) = v_t(s), \quad \forall s \neq s_t (\text{未被访问的状态不更新，常省略此式}) \tag{1b}$$

涉及到更新过程的式(1a)可以更详细地写为：

$$\underbrace{v_{t+1}(s_t)}_{\text{new estimate}} = \underbrace{v_t(s_t)}_{\text{current estimate}} - \alpha_t(s_t)\Bigg[\overbrace{v_t(s_t) - \underbrace{[r_{t+1} + \gamma v_t(s_{t+1})]}_{\text{TD target } \bar{v}_t}}^{\text{TD error } \delta_t}\Bigg] \tag{2}$$

其中，定义 **TD target** 为

$$\bar{v}_t \triangleq r_{t+1} + \gamma v(s_{t+1}) \tag{3}$$

定义 **TD error** 为

$$\delta_t \triangleq v(s_t) - [r_{t+1} + \gamma v(s_{t+1})] = v(s_t) - \bar{v}_t \tag{4}$$

由此看出，新的估计 $v_{t+1}(s_t)$ 是当前估计 $v_t(s_t)$ 和 TD error (4) 的组合.

Basic TD – Property analysis

首先详细分析 TD target (3)：将 $\bar{v}_t$ 称为 target 是因为我们迭代的目标就是让 $v(s_t)$ 不断靠近 $\bar{v}_t$.

根据式(2)，可以得到

$$\begin{aligned} &v_{t+1}(s_t) - \bar{v}_t = v_t(s_t) - \bar{v}_t - \alpha_t(s_t)\left[v_t(s_t) - \bar{v}_t\right] = [1 - \alpha_t(s_t)][v_t(s_t) - \bar{v}_t] \\ &\implies |v_{t+1}(s_t) - \bar{v}_t| = |1 - \alpha_t(s_t)||v_t(s_t) - \bar{v}_t| \end{aligned}$$

再由 $\alpha_t(s_t)$ 为一个较小的正数，得到：

$$\alpha_t(s_t) > 0 \Rightarrow 0 < 1 - \alpha_t(s_t) < 1 \Rightarrow |v_{t+1}(s_t) - \bar{v}_t| \leqslant |v_t(s_t) - \bar{v}_t|$$

因此 $v_t(s_t) \to \bar{v}_t$.

下面分析 TD error (4)：

1. Temporal Difference 可以看到 TD error 的定义和算法更新涉及到 t 与 $t+1$ 两个时间步，因此将此算法称为 "temporal difference"(时序差分)，反映了两个时间步间的差异.

2. State value difference 由于 $v_\pi(s_t) = \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s_t\right]$，因此

$$\mathbb{E}[\delta_{\pi,t} | S_t = s_t] = v_\pi(s_t) - \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s_t\right] = 0$$

因此如果 $v_t = v_\pi$，那么 $\delta_t = 0$，否则 $v_t \neq v_\pi$. 故算法迭代的目标是 $v_t \to v_\pi$，TD error 反映了估计值 v_t 与真实状态值 v_π 之间的差异.

TD Learning of state values: The idea of the algorithm

Bellman expectation equation – 首先介绍一个新的 Bellman equation，其与原先介绍的 BE 等价，只不过使用了期望的数学形式，因此看起来更简洁. 根据 state value 的定义

$$v_\pi(s) = \mathbb{E}\left[R + \gamma G \mid S = s\right], \quad s \in S,$$

其中 G 为 discounted return. 由于

$$\mathbb{E}[G | S = s] = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) v_\pi(s') = \mathbb{E}[v_\pi(S') | S = s]$$

其中 S' 为下一 state。这样 BE 就可以重写为

$$v_\pi(s) = \mathbb{E}\left[R + \gamma v_\pi(S') \mid S = s\right], \quad s \in \mathcal{S} \tag{5}$$

RM algorithm for solving the Bellman expectation equation (5)
首先定义求解的目标

$$g(v(s)) = v(s) - \mathbb{E}\left[R + \gamma v_\pi(S') \mid s\right] \to g(v(s)) = 0$$

进而根据 R 和 S' 的采样 r, s' 得到

$$\tilde{g}(v(s)) = \underbrace{\left(v(s) - \mathbb{E}\left[R + \gamma v_\pi(S') \mid s\right]\right)}_{g(v(s))} + \underbrace{\left(\mathbb{E}\left[R + \gamma v_\pi(S') \mid s\right] - \left[r + \gamma v_\pi(s')\right]\right)}_{\eta}.$$

因此求解此问题的 RM 算法为：

$$v_{t+1}(s) = v_t(s) - \alpha_t \tilde{g}(v_t(s)) = v_t(s) - \alpha_t \left(v_t(s) - \left[r_t + \gamma v_{\pi}(s'_t)\right]\right) \tag{6}$$

其中 $v_t(s)$ 为 $v_\pi(s)$ 在第 k 步的估计，r_t, s'_t 分别为 R, S' 在第 t 步的采样.
需要注意的两点是

1. 算法(6)中仅是在对一个 state s 进行估计，在 TD learning 中我们需要将固定的 $\{(s, r, s')\}$ 变为序列 $\{s_t, t_t, s_{t+1}\}$，而没有更新到的 state 就不变（如式1b）
2. 算法(6)中使用了我们估计的目标 v_π 进行迭代，在 TD learning 中由于 v_π 未知，因此需要使用 $v_t(s')$ 替换 $v_\pi(s'_t)$. 此时依然有收敛性，详见 **Theorem 1**.

What does this TD learning alg. do mathematically? – 通过上述分析可以看出，TD learning 在求解给定策略 π 的 Bellman expectation equation.

Basic idea TD learning 的基本思想是根据新获得的信息纠正当前对状态值的估计，从而使得 $v_t \to v_\pi$.

TD Learning of state values: Convergence analysis

Theorem 1 (Convergence of TD learning)**.** 给定 *policy* π，使用 *TD algorithm (1a)(1b)*时，若满足如下条件则 $\forall s \in \mathcal{S}$， $v_t(s) \xrightarrow[t\to\infty]{w.p.1.} v_\pi(s)$:

1. $\sum\limits_t \alpha_t(s) = \infty$（每一个状态 s 都会被访问足够多/无穷次，因为未被访问时$\alpha_t(s) = 0$）
2. $\sum_t \alpha_t^2(s) < \infty, \forall s \in \mathcal{S}$（意味着$\alpha_t \to 0$，实际中设为随着 t 最后变得很小的数）

Note 1. 事实上当 α 为常数时，仍然可以证明算法在期望意义上收敛. ***Theorem 1*** 的证明详见附录*A.1*，其需要使用 ***Lecture 6*** 中 *Dvoretzky's theorem* 的推广版本.

TD Learning of state values: comparison

Comparison between TD learning and MC learning[1]	
TD / Sarsa learning	**MC learning**
Online / Incremental: TD / Sarsa 可以在获得 experience sample 后立即更新 state/action value	**Offline / Non-incremental:** 由于要计算 episode 的 discounted return，因此 MC learning 需要等到 episode 采样完成.
Continuing tasks: 由于 TD / Sarsa 是增量式的，因此可以处理 episodic 或 continuing 任务(可能没有终止状态).	**Episodic tasks:** 只能处理有终止状态(terminal states)的 episodic tasks.
Bootstrapping: TD / Sarsa 的更新依赖之前对 state / action value 的估计，因此需要初始猜测(initial guess).	**Non-bootstrapping:** MC 直接估计 state / action value，无需初始猜测.
Low estimation variance: 由于使用的随机变量少因此估计方差低于 MC. 例如 Sarsa 只需要三个随机变量的样本: $R_{t+1}, S_{t+1}, A_{t+1}$.	**High estimation variance:** 为估计 $q_\pi(s_t, a_t)$，需要 $R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$ 的样本，假设 episode 的长度为 L，那么就有 $\lvert A\rvert^L$ 个可能的 episodes.
Biased to unbiased 由于 TD / Sarsa 数据量大使用由少变多，因此从有偏估计慢慢趋向于无偏估计.	**Unbiased** 由于使用了更多的数据，因此 MC 是无偏的.

[1] 由于 TD learning 与 Sarsa 基本一样，因此放在一起比较.

Table 1: Comparison between TD /Sarsa learning and MC learning

TD Learning of action values

TD Learning of action values: Sarsa

Data Sarsa 算法目的是给定 policy π 的 action value $q_\pi(s, a)$，因此需要的数据/ experience 为 $\{(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})\}_t$.

Algorithm 直接给出 Sarsa 算法如下：

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t) \left[q_t(s_t, a_t) - [r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})]\right] \tag{7a}$$

$$q_{t+1}(s,a) = q_t(s,a), \quad \forall (s,a) \neq (s_t,a_t), t = 0,1,2,\cdots \tag{7b}$$

其中 $q_t(s_t,a_t)$ 是对 $q_\pi(s_t,a_t)$ 的估计，α_t 为 learning rate.

Why called Sarsa? – 源自 state-action-reward-state-action 的首字母缩写.

The relationship between Sarsa and TD learning – Sarsa 是 TD learning 的 action-value version，只需要将 TD learning 中的 $v(s)$ 替换为 $q(s,a)$ 即可.

What does the Sarsa do mathematically? – 类似于 TD learning，Sarsa 也在求解一个 action value 版本的 Bellman equation(证明下式为 BE 可详见附录A.2):

$$q_\pi(s,a) = \mathbb{E}\left[R + \gamma q_\pi(S',A')|s,a\right], \quad \forall s,a \tag{8}$$

Theorem 2 (Convergence of Sarsa learning)**.** 使用 *Srasa algorithm* 时，若满足如下条件则 $\forall s \in \mathcal{S}, a \in \mathcal{A}$，$q_t(s,a) \xrightarrow[t\to\infty]{w.p.1.} q_\pi(s,a)$:

1. $\sum\limits_t \alpha_t(s,a) = \infty$（每对 (s,a) 都会被访问很多/无穷次，因为未被访问时$\alpha_t(s,a)=0$）
2. $\sum_t \alpha_t^2(s,a) < \infty, \forall s \in \mathcal{S}, a \in \mathcal{A}$（意味着$\alpha_t \to 0$，实际中设置为随 t 逐渐变足够小）

Note 2. 收敛性的证明与 *TD learning* 中 ***Theorem 1***类似，因此暂略.

TD Learning of action values: Sarsa - Algorithm

Sarsa 算法只能通过迭代得到给定 policy π 的 action value，后续仍需结合 ploicy improvement 才能得到 optimal policy. 实际上二者结合后也被成为 Sarsa，且常说的 Sarsa 就是指结合后的算法.

Algorithm 1 Optimal policy learning by Sarsa

1: **Initialization:** $\alpha_t(s,a) = \alpha > 0$ for all (s,a) and all t. $\epsilon \in (0,1)$. Initial $q_0(s,a)$ for all (s,a). Initial ϵ-greedy policy π_0 derived from q_0.
2: **Goal:** Learn an optimal policy that can lead the agent to the target state from an initial state s_0.
3: **for** each episode **do**
4:     **if** the current s_t is not the target state **then**
5:         Collect the experience $(s_t,a_t,r_{t+1},s_{t+1},a_{t+1})$: In particular, take action a_t following $\pi_t(s_t)$, generate r_{t+1},s_{t+1}, and then take action a_{t+1} following $\pi_t(s_{t+1})$.
6:         Update q-value**(policy evaluation):**

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

7:         Update policy**(policy improvement):**

$$\pi_{t+1}(a \mid s_t) = \begin{cases} 1 - \frac{\epsilon}{|A|-1} & \text{if } a = \arg\max_a q_{t+1}(s_t,a) \\ \frac{\epsilon}{|A|} & \text{otherwise} \end{cases}$$

8:     **end if**
9: **end for**

Note 3. 注意在做 *policy evalution* 时只做了一次就进行了 *policy improvement* 并继续迭代，并没有准确进行估计，因此是基于 *gneralized policy iteration* 的想法.

TD Learning of action values: Expected Sarsa

Expected Sarsa 算法是在原先的 Sarsa 基础上变形得到的，将 TD target 由 $r_{t+1}+\gamma q_t(s_{t+1},a_{t+1})$ 变为 $r_{t+1}+\gamma\mathbb{E}[q_t(s_{t+1},A)]$，因此 Expected Sarsa 算法如下：

$$q_{t+1}(s_t,a_t)=q_t(s_t,a_t)-\alpha_t(s_t,a_t)\left[q_t(s_t,a_t)-(r_{t+1}+\gamma\mathbb{E}[q_t(s_{t+1},A)])\right] \tag{9a}$$

$$q_{t+1}(s,a)=q_t(s,a),\quad \forall(s,a)\neq(s_t,a_t) \tag{9b}$$

其中

$$\mathbb{E}[q_t(s_{t+1},A)])=\sum_a \pi_t(a|s_{t+1})q_t(s_{t+1},a)\triangleq v_t(s_{t+1})$$

Note 4. 可以看到 *expected Sarsa* 不需要对 a_{t+1} 进行采样，取而代之需要更大的计算量去估计 $\mathbb{E}[q_t(s_{t+1},A)])$，因此相较于 *Sarsa* 计算量增大，但是由于使用的随机变量的个数变少，因此估计方差变小.

What does the expected Sarsa do mathematically? – 实际上在求解如下方程：

$$q_\pi(s,a)=\mathbb{E}\left[R_{t+1}+\gamma\mathbb{E}_{A_{t+1}\sim\pi(S_{t+1})}[q_\pi(S_{t+1},A_{t+1})]|S_t=s,A_t=a\right],\quad \forall s,a. \tag{10}$$

由于

$$\mathbb{E}\left[q_\pi\left(S_{t+1},A_{t+1}\right)\mid S_{t+1}\right]=\sum_{A'}q_\pi\left(S_{t+1},A'\right)\pi\left(A'\mid S_{t+1}\right)=v_\pi\left(S_{t+1}\right)$$

因此式(10)是如下 Bellman equation 的另一种展开式：

$$q_\pi(s,a)=\mathbb{E}\left[R_{t+1}+\gamma v_\pi(S_{t+1})|S_t=s,A_t=a\right]$$

TD Learning of action values: n-step Sarsa

n-step Sarsa 是 Sarsa 的一个变形，也是一个推广，Sarsa 和 Monte Carlo 可以看作是其两种极端情况.

首先，根据定义，action value 写为

$$q_\pi(s,a)=\mathbb{E}[G_t|S_t=s,A_t=a]$$

其中 G_t 为 discounted return，其可以被写为多种不同的形式：

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

Note 5. 注意此处所有含有上标的 $G_t^{(1)},\cdots,G_t^{(\infty)}$ 都是等价的，不同的仅是如何分解.

得到 G_t 不同的分解后，将其代入 action value 定义式中，得到

$$\begin{aligned}
\text{Sarsa to solve:} \quad & q_\pi(s,a) = \mathbb{E}[G_t^{(1)}|s,a] = \mathbb{E}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|s,a] \\
n\text{-step Sarsa:} \quad & q_\pi(s,a) = \mathbb{E}[G_t^{(n)}|s,a] = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n q_\pi(S_{t+n}, A_{t+n})|s,a] \\
\text{MC learning:} \quad & q_\pi(s,a) = \mathbb{E}[G_t^{(\infty)}|s,a] = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots|s,a]
\end{aligned}$$

相应的随机近似算法如下：

$$\begin{aligned}
\text{Sarsa:} \quad & q_{t+1}(s_t,a_t) = q_t(s_t,a_t) - \alpha_t(s_t,a_t)\left[q_t(s_t,a_t) - [r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})]\right] \\
n\text{-step Sarsa:} \quad & q_{t+1}(s_t,a_t) = q_t(s_t,a_t) - \alpha_t(s_t,a_t)\left[q_t(s_t,a_t) - [r_{t+1} + \cdots + \gamma^n q_t(s_{t+n}, a_{t+n})]\right] \\
\text{MC learning:} \quad & q_{t+1}(s_t,a_t) = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^n t_{t+n} + \cdots
\end{aligned}$$

当 n-step Sarsa 中的 $n = 1$ 时，就变为了 Sarsa，当 $\alpha_t = 1, n = \infty$ 时，就变为了 MC learning，因此说 n-step Sarsa 包含了 Sarsa 和 TD learning.当 n 较大时，n-step Sarsa 的性质类似于 TD learning，有比较大的 variance，但较小的 bias；反之 当n 较小时，其性质类似于 Sarsa，有较小的 variance 但较大的 bias.

下面比较这三种算法的不同之处：

① **TD target:** Sarsa 与 n-step Sarsa 的 TD target是不同的：

$$\begin{aligned}
\text{Sarsa:} \quad & \bar{v}_t^{(1)} \triangleq r_{t+1} + \gamma v(s_{t+1}) \\
n\text{-step Sarsa:} \quad & \bar{v}_t^{(2)} \triangleq r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^n q_t(s_{t+n}, a_{t+n})
\end{aligned}$$

② **Data needed:** 由于三种算法都是 model free 的算法，但所需数据不同：

$$\begin{aligned}
\text{Sarsa:} \quad & (s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}) \\
n\text{-step Sarsa:} \quad & (s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, \cdots, r_{t+n}, s_{t+n}, a_{t+n}) \\
\text{TD learning:} \quad & (s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, \cdots, r_{t+n}, s_{t+n}, a_{t+n}, \cdots)
\end{aligned}$$

③ **Online vs. Offline:** Sarsa 为 online 的算法，可以使用当前时间节点的数据立即进行算法迭代，而 MC learning 为 offline 的算法，需要等待到足够多的数据才可以估算 action value. n-step Sarsa 则既不是 online 也不是 offline 的算法，或将其看作是特殊的 online 算法，因为其需要等到 $t+n$ 时刻才可以更新 (s_t, a_t) 的 action value：

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

TD Learning of optimal action values

TD Learning of optimal action values: Q-learning

Sarsa 及其变形都是在估计 action value，需要结合 policy evaluation 才能够进行 policy searching(如Algorithm1所示). 下面介绍非常经典且常用[a]的算法 Q-learning，其直接估计 optimal action value，无需交替进行 policy evaluation 和 policy improvement.

Algorithm 首先直接给出 Q-learning 算法：

$$q_{t+1}(s_t,a_t) = q_t(s_t,a_t) - \alpha_t(s_t,a_t)\left[q_t(s_t,a_t) - [r_{t+1} + \gamma \max_{a\in\mathcal{A}} q_t(s_{t+1}, a)]\right] \tag{11a}$$

$$q_{t+1}(s,a) = q_t(s,a), \quad \forall (s,a) \neq (s_t, a_t) \tag{11b}$$

可以看到 Q-learning 的形式与 Sarsa 的唯一区别在于 TD target:

$$\text{Q-learning:} \quad r_{t+1} + \gamma \max_{a \in \mathcal{A}} q_t(s_{t+1}, a)$$
$$\text{Sarsa:} \quad r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})$$

Data 由于 TD target 不同，导致所需的数据不同：Sarsa 每个 iteration 需要的数据是 $(r_{t+1}, s_{t+1}, a_{t+1})$，Q-learning 是 (r_{t+1}, s_{t+1}).

What does the Q-learning do mathematically? – 与 Sarsa 在求解一个 action value 的 Bellman equation 不同，Q-learning 是在求解一个带有期望的带有 action value 形式的 Bellman optimality equation(BOE). 关于下式(12)是 BOE 证明详见附录A.3

$$q(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_a q(S_{t+1}, a) | S_t = s, A_t = a\right], \quad \forall s, a. \tag{12}$$

[a]现在常用的算法是 deep Q-learning，为 Q-learning 的变形

Off vs. on-policy

首先介绍 TD learning 中存在的两种 policy：

Definition 1 (Two policies in TD learning)**.** *TD learning* 中存在两种策略，分别为 ***behavior policy*** 和 ***target policy***。其中 *behavior policy* 用于生成 *experience samples*，是智能体与环境的交互策略，可以包含探索成分； *target policy* 用于不断迭代至最优策略.

基于上述两种 policy，可以将不同的强化学习算法分为 on-policy 和 off-policy 两类：

Definition 2 (on-policy and off-policy)**.** 当一种学习方法的 *behavior policy* 与 *target policy* 一致时，称之为 ***on-policy***，即智能体在学习过程中是通过执行自己的 *target policy* 来收集数据和更新策略；当 *behavior policy* 与 *target policy* 不一致时，称之为 ***off-policy***，智能体可以通过 *behavior policy* 收集数据，而通过 *target policy* 来进行学习.

Advantages of off-policy learning: off-policy 的好处是其可以使用别的（例如探索性更强的）policy 生成的 experience samples 去搜索 optimal policy，可以加速学习进程.

How to judge On or off-policy? 1. 搞清楚算法在数学上做了什么事情（例如求解BE/BOE?）；2. 检查算法需要什么数据.

Sarsa is on-policy

Sarsa 在每个 iteration 中，第一步需要通过求解给定 policy π 的 Bellman equation 进行 policy evaluation，这需要由 policy π 产生的数据，因此 π 为 behavior policy. 第二步根据 π 的估计值获取改进的策略，因此 π 也是 target policy.

Sarsa 需要的数据为 $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$，其产生的方式为

$$s_t \xrightarrow{\pi_b} a_t \xrightarrow{\text{model}} r_{t+1}, s_{t+1} \xrightarrow{\pi_b} a_{t+1}$$

可以看到数据是通过 behavior policy π_b 产生的，而 Sarsa 的目的是估计 target policy π_T 的 action value，由于 π_T 的估计基于 $(r_{t+1}, s_{t+1}, a_{t+1})$，而 a_{t+1} 的产生依赖于 π_b，故 Sarsa 估计的 policy 又被用于产生数据，因此为 on-policy.

MC learning is on-policy: 首先，MC learning 的目标是求 action value 的期望：

$$q_\pi(s,a) = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \ldots | S_t = s, A_t = a\right], \quad \forall s, a.$$

其次，在算法是使用多个采样来近似估算：

$$q(s,a) \approx r_{t+1} + \gamma r_{t+2} + \dots$$

其过程为：对于当前的策略 π，生成相应的 experience / trajectory，再使用该 experience / trajectory 的 return 得到 action value，再改进 π，即如下流程：

$$\pi \to \text{experience / trajectory} \to q_\pi \to \pi \to \cdots$$

因此这里的 π 既是 behavior policy 也是 target policy.

Q-learning is off-policy: 首先 Q-learning 数学上在求解一个 BOE：

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

其次，在算法上需要的数据为 $(s_t, a_t, r_{t+1}, s_{t+1})$，其中若 (s_t, a_t) 给定，那么 r_{t+1} 和 s_{t+1} 分别依赖于 $\mathbb{P}(r|s,a), \mathbb{P}(s'|s,a)$，而不依赖于 policy.

$$s_t \xrightarrow{\pi_b} a_t \xrightarrow{\text{model}} r_{t+1}, s_{t+1}$$

Q-learning 的 behavior policy 用于从 state s_t 产生 action a_t，其可以是任何 policy，而 target policy 则为 optimal policy，其根据 $q \to q^*$ 来判断相应策略收敛至 optimal policy。因此 Q-learning 是 off-policy 的.

Note 6. 有一些方法可以将 *on-policy* 算法转换为 *off-policy* 算法，例如重要性采样*(importance sampling)*.

Online vs. Offline

容易与 on-policy/off-policy 混淆的两个概念是 online/offline. 在线学习(online)是指代理在与环境交互时更新 value 和 policy，离线学习(offline)是指代理使用预先收集的体验数据更新 value 和 policy 而无需与环境交互.

因此若一个算法是 on-policy 的，那么它只能以 online 的模式进行，不能使用别的 policy 预先产生的数据；若算法是 off-policy 的，那么它 online / offline 的模式都可以使用.

Q-learning – Implementation

由于 Q-learning 是 off-policy 的，其 behavior policy 与 target policy 可以不同也可以相同，分别能够得到如下两种版本的算法：

Algorithm 2 Policy Searching by Q-learning (on-policy version)

for each episode **do**
  if the current s_t is not the target state **then**
    Collect the experience $(s_t, a_t, r_{t+1}, s_{t+1})$: In particular, take action a_t following $\pi_t(s_t)$, generate r_{t+1}, s_{t+1}.
    Update q-value:

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

    Update policy:

$$\pi_{t+1}(a \mid s_t) = \begin{cases} 1 - \frac{\epsilon}{|A|-1} & \text{if } a = \arg\max_a q_{t+1}(s_t, a), \\ \frac{\epsilon}{|A|} & \text{otherwise.} \end{cases}$$

  end if
end for

Algorithm 3 Optimal Policy Search by Q-learning (off-policy version)

for each episode $\{s_0, a_0, r_1, s_1, a_1, r_2, \ldots\}$ generated by π_b **do** ▷ π_b is behavior policy
  for each step $t = 0, 1, 2, \ldots$ of the episode **do**
    Update q-value:

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

    Update target policy:

$$\pi_{T,t+1}(a \mid s_t) = \begin{cases} 1 & \text{if } a = \arg\max_a q_{t+1}(s_t, a), \\ 0 & \text{otherwise.} \end{cases}$$

▷ π_T is target policy
  end for
end for

Note 7. 在 *off-policy version* 中直接使用 *greedy* 策略进行 *target policy* 的更新而无需具备探索性是因为此时数据由 *behavior policy* 生成，只需要每次直接选择最好的 *target policy* 即可；相反在 *on-policy version* 中，由于 *target policy* 也需要再生成数据，因此需要保持一定的探索性，故使用 $\epsilon-$*greedy* 的更新模式.

A unified point of view

本节中所介绍的所有算法实际上都可以表达为一个统一的形式：

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

其中 $\bar{q}_t$ 为 TD target。所有的 TD 算法的不同点就在于 TD target 和数学上的求解目标：

Different TD target in different TD algorithms	
Algorithm	**Expression of $\bar{q}_t$**
Sarsa	$\bar{q}_t = r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})$
n-step Sarsa	$\bar{q}_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^n q_t(s_{t+n}, a_{t+n})$
Expected Sarsa	$\bar{q}_t = r_{t+1} + \gamma \sum_a \pi_t(a \mid s_{t+1}) q_t(s_{t+1}, a)$
Q-learning	$\bar{q}_t = r_{t+1} + \gamma \max_a q_t(s_{t+1}, a)$
Monte Carlo[1]	$\bar{q}_t = r_{t+1} + \gamma r_{t+2} + \ldots$

[1] 只需要将 $\alpha_t(s_t, a_t) = 1$，就可以得到 $q_{t+1}(s_t, a_t) = \bar{q}_t$

Table 2: Different TD target in different TD algorithms

Different equation aim to solve in different TD algorithms	
Algorithm	**Equation aim to solve**
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: $q_\pi(s,a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n q_\pi(s_{t+n}, a_{t+n}) \vert S_t = s, A_t = a]$
Expected Sarsa	BE: $q_\pi(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \mathbb{E}_{A_{t+1}}[q_\pi(S_{t+1}, A_{t+1})] \vert S_t = s, A_t = a\right]$
Q-learning	BOE: $q(s,a) = \mathbb{E}\left[R_{t+1} + \max_a q(S_{t+1}, a) \vert S_t = s, A_t = a\right]$
Monte Carlo	BE: $q_\pi(s,a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \ldots \vert S_t = s, A_t = a]$

Table 3: Different equation aim to solve in different TD algorithms

A Proof

A.1 Proof for TD learning convergence Theorem 1

TD learning convergence

Proposition 1 (Convergence of TD learning)**.** 给定 *policy* π，使用 *TD algorithm (1a)(1b)*时，若满足如下条件则 $\forall s \in \mathcal{S}$， $v_t(s) \xrightarrow[t\to\infty]{w.p.1.} v_\pi(s)$:

1. $\sum\limits_t \alpha_t(s) = \infty$

2. $\sum_t \alpha_t^2(s) < \infty, \forall s \in \mathcal{S}$

Proof. 根据 TD learning 算法，有

$$v_{t+1}(s) = \begin{cases} v_t(s) - \alpha_t(s)\left(v_t(s) - (r_{t+1} + \gamma v_t(s_{t+1}))\right), & \text{若} s = s_t, \\ v_t(s), & \text{若} s \neq s_t. \end{cases} \tag{13}$$

给定 policy π 后，定义 estimation error 为

$$\Delta_t(s) \triangleq v_t(s) - v_\pi(s).$$

这样就得到

$$\Delta_{t+1}(s) = \begin{cases} (1-\alpha_t(s))\,\Delta_t(s) + \alpha_t(s)\underbrace{(r_{t+1} + \gamma v_t\,(s_{t+1}) - v_\pi(s))}_{\eta_t(s)}, & \text{若} s = s_t, \\ \Delta_t(s) = (1-\alpha_t(s))\,\Delta_t(s) + \underbrace{\alpha_t(s)}_{=0}\,\underbrace{\eta_t(s)}_{=0}, & \text{若} s \neq s_t. \end{cases} \tag{14}$$

因此可以统一写为

$$\Delta_{t+1}(s) = (1-\alpha_t(s))\,\Delta_t(s) + \alpha_t(s)\eta_t(s)$$

下面使用 **Lemma 1**，只需要逐条验证其中的三个条件均满足即可.
(a) 根据假设条件，自然满足.
(b) 当 $s \neq s_t$ 时 $\eta_t(s) = 0$，自然满足；当 $s = s_t$ 时，根据马尔可夫性有

$$\mathbb{E}[\eta_t(s_t)|\mathcal{H}_t] = \mathbb{E}[\eta_t(s_t)] = \mathbb{E}[r_{t+1} + \gamma v_t(s_{t+1}) - v_\pi(s_t)|s_t] = \mathbb{E}[r_{t+1} + \gamma v_t(s_{t+1})|s_t] - v_\pi(s_t).$$

由于 $v_\pi(s_t) = \mathbb{E}[r_{t+1} + \gamma v_\pi(s_{t+1})|s_t]$，因此

$$\mathbb{E}[\eta_t(s)|\mathcal{H}_t] = \gamma\mathbb{E}[v_t(s_{t+1}) - v_\pi(s_{t+1})|s_t] = \gamma \sum_{s'\in\mathcal{S}} p(s'|s_t)[v_t(s') - v_\pi(s')].$$

从而有

$$\begin{aligned} |\mathbb{E}[\eta_t(s)]| &= \gamma \left| \sum_{s'\in\mathcal{S}} p(s'|s_t)[v_t(s') - v_\pi(s')] \right| \leqslant \gamma \sum_{s'\in\mathcal{S}} p(s'|s_t) \max_{s'\in\mathcal{S}} |v_t(s') - v_\pi(s')| \\ &= \gamma \max_{s'\in\mathcal{S}} |v_t(s') - v_\pi(s')| = \gamma\|v_t(s') - v_\pi(s')\|_\infty = \gamma\|\Delta_t(s)\|_\infty. \end{aligned}$$

(c) 当 $s \neq s_t$ 时，$\mathrm{var}[\eta_t(s)|\mathcal{H}_t]=0$；当 $s=s_t$ 时，

$$\mathrm{var}[\eta_t(s)|\mathcal{H}_t]=\mathrm{var}[r_{t+1}+\gamma v_t(s_{t+1})-v_\pi(s_t)|s_t]=\mathrm{var}[r_{t+1}+\gamma v_t(s_{t+1})|s_t]$$

由于 r_{t+1} 有界，因此此条件也可以证明满足. □

Lemma 1 (Extension of Dvoretzky's theorem)**.** 设 $\mathcal{S}$ 为有限实数集. 对于随机过程

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

若对于每个 $s \in \mathcal{S}$，满足以下条件，则有 $\Delta_k(s)\to 0$ 几乎必然成立:

(a) $\sum_k \alpha_k(s)=\infty$，$\sum_k \alpha_k^2(s)<\infty$，$\sum_k \beta_k^2(s)<\infty$，并且 $\mathbb{E}[\beta_k(s)\mid\mathcal{H}_k]\leqslant\mathbb{E}[\alpha_k(s)\mid\mathcal{H}_k]$ 几乎必然一致成立;

(b) $\|\mathbb{E}[\eta_k(s)\mid\mathcal{H}_k]\|_\infty\leqslant\gamma\|\Delta_k\|_\infty$，其中 $\gamma\in(0,1)$;

(c) $\mathrm{var}[\eta_k(s)\mid\mathcal{H}_k]\leqslant C(1+\|\Delta_k(s)\|_\infty^2)$，其中 C 为常数。

其中，$\mathcal{H}_k=\{\Delta_k,\Delta_{k-1},\ldots,\eta_{k-1},\ldots,\alpha_{k-1},\ldots,\beta_{k-1},\ldots\}$ 表示历史信息. 符号 $\|\cdot\|_\infty$ 表示最大范数*(maximum norm)*.

A.2 Proof for action value version Bellman equation

TD learning convergence

Proposition 2. 下式*(15)*是关于 *action value* 的 *Bellman equation*

$$q_\pi(s,a)=\mathbb{E}\left[R+\gamma q_\pi(S',A')|s,a\right],\quad \forall s,a \tag{15}$$

Proof. 根据定义，action value 的 Bellman equation 为

$$\begin{aligned} q_\pi(s,a)&=\sum_r rp(r\mid s,a)+\gamma\sum_{s'}\sum_{a'}q_\pi\left(s',a'\right)p\left(s'\mid s,a\right)\pi\left(a'\mid s'\right)\\ &=\sum_r rp(r\mid s,a)+\gamma\sum_{s'}p\left(s'\mid s,a\right)\sum_{a'}q_\pi\left(s',a'\right)\pi\left(a'\mid s'\right). \end{aligned} \tag{16}$$

根据条件概率公式有

$$p(s',a'|s,a)=p(s'|s,a)p(a'|s',s,a)=p(s'|s,a)p(a'|s')\triangleq p(s'|s,a)\pi(a'|s'),$$

因此

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

根据期望的定义，即有式(15)成立. □

A.3 Proof for action value version Bellman optimal equation

TD learning convergence

Proposition 3. 下式*(17)*是带有期望的 *action value* 形式的 *Bellman optimality equation(BOE):*

$$q(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a} q(S_{t+1},a) | S_t = s, A_t = a\right], \quad \forall s, a. \tag{17}$$

Proof. 根据期望的定义，式可写为

$$q(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a) \max_{a\in\mathcal{A}(s')} q(s',a). \tag{18}$$

对两边同时取 max 得到

$$\max_{a\in\mathcal{A}(s)} q(s,a) = \max_{a\in\mathcal{A}(s)} \left[\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a) \max_{a\in\mathcal{A}(s')} q(s',a)\right]. \tag{19}$$

记 $v(s) \triangleq \max_{a\in\mathcal{A}(s)} q(s,a)$，上式可以重新写为

$$\begin{aligned} v(s) &= \max_{a\in\mathcal{A}(s)} \left[\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a) v(s')\right] \\ &= \max_{\pi} \sum_{a\in\mathcal{A}(s)} \pi(a|s) \left[\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a) v(s')\right], \end{aligned} \tag{20}$$

因此式(12)是 BOE. □

First updated: February 12, 2025
Last updated: May 20, 2025

References