1. Sarsa State-Action-Reward-State-Action (SARSA)

Derive TD Target

- Assume R_t depends on (S_t, A_t, S_{t+1}).

为什么？在t时刻的奖励R_t 在给定t时刻状态s_t并做出动作a_t后，不就已经得到了吗？为什么依赖于t+1时刻的状态s_{t+1}？

基于policy π的 action-value function

- $Q_\pi(s_t, a_t) = \mathbb{E}[U_t | s_t, a_t]$

$$= \mathbb{E}[R_t + \gamma \cdot U_{t+1} | s_t, a_t]$$

$$= \mathbb{E}[R_t | s_t, a_t] + \gamma \cdot \mathbb{E}[U_{t+1} | s_t, a_t] \quad ?$$

$$= \mathbb{E}[R_t | s_t, a_t] + \gamma \cdot \mathbb{E}[Q_\pi(S_{t+1}, A_{t+1}) | s_t, a_t].$$

Identity: $Q_\pi(s_t, a_t) = \mathbb{E}[\underline{R_t} + \gamma \cdot \underline{Q_\pi(S_{t+1}, A_{t+1})}]$, for all π.

- We do not know the expectation.
- Approximate it using Monte Carlo (MC).

$\approx r_t$ $\quad\quad \approx Q_\pi(s_{t+1}, a_{t+1})$

$$\approx \boxed{r_t + \gamma \cdot Q_\pi(s_{t+1}, a_{t+1})}$$

TD target y_t

TD learning: Encourage $Q_\pi(s_t, a_t)$ to approach y_t.

Sarsa: Tabular Version

- We want to learn $Q_{\boxed{\pi}}(s, a)$.
- Suppose the numbers of states and actions are finite.
- Draw a table and learn the entries.

	Action a_1	Action a_2	Action a_3	Action a_4	⋯
State s_1					
State s_2					
State s_3					
⋮					

Algorithm

- Use $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ for updating Q_π.
- State-Action-Reward-State-Action (SARSA).

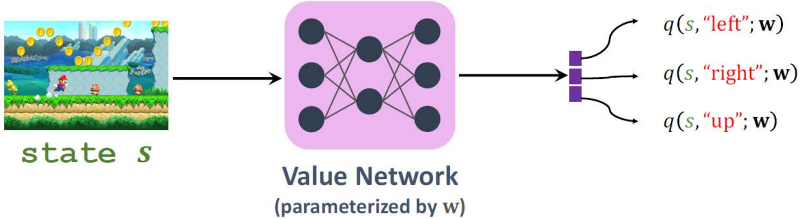
- Observe a transition (s_t, a_t, r_t, s_{t+1}).
- Sample $a_{t+1} \sim \pi(\cdot \mid s_{t+1})$, where π is the policy function.
- TD target: $y_t = r_t + \gamma \cdot Q_\pi(s_{t+1}, a_{t+1})$.

  注意！ t+1时刻的Q是基于一个随机采样的a_{t+1};
  在接下来的Q learning中， t+1时刻的Q是所有Q值中(a取值不同)最大的;
- TD error: $\delta_t = Q_\pi(s_t, a_t) - y_t$.
- Update: $Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) - \alpha \cdot \delta_t$. 直接更新表格，减小error

Sarsa: Neural Network Version

Actor-critic method中，value net就是这么训练的

- Approximate $Q_\pi(s, a)$ by the value network, $q(s, a; \mathbf{w})$.



- q is used as the critic who evaluates the actor. (Actor-Critic Method.)
- We want to learn the parameter, $\mathbf{w}$.

TD Error & Gradient

- TD target: $y_t = r_t + \gamma \cdot q(s_{t+1}, a_{t+1}; \mathbf{w})$.
- TD error: $\delta_t = q(s_t, a_t; \mathbf{w}) - y_t$.
- Loss: $\delta_t^2/2$.
- Gradient: $\frac{\partial\, \delta_t^2/2}{\partial\, \mathbf{w}} = \delta_t \cdot \frac{\partial\, q(s_t, a_t; \mathbf{w})}{\partial\, \mathbf{w}}$.
- Gradient descent: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \delta_t \cdot \frac{\partial\, q(s_t, a_t; \mathbf{w})}{\partial\, \mathbf{w}}$.

2. Q-Learning

Derive TD Target

- We have proved that for all π,

$$Q_\pi(s_t, a_t) = \mathbb{E}[R_t + \gamma \cdot Q_\pi(S_{t+1}, A_{t+1})].$$

- If π is the optimal policy $\pi^\star$, then

$$Q_{\pi^\star}(s_t, a_t) = \mathbb{E}[R_t + \gamma \cdot Q_{\pi^\star}(S_{t+1}, A_{t+1})].$$

- $Q_{\pi^\star}$ and $Q^\star$ both denote *the optimal action-value function.*

Identity: $Q^\star(s_t, a_t) = \mathbb{E}[R_t + \gamma \cdot Q^\star(S_{t+1}, A_{t+1})].$

- The action A_{t+1} is computed by

$$A_{t+1} = \underset{a}{\operatorname{argmax}}\, Q^\star(S_{t+1}, a).$$

A_{t+1} 是随机变量，当S_{t+1} 取值不同时， A_{t+1} 取值会不同。

- Thus $Q^\star(S_{t+1}, A_{t+1})$ 只由S_{t+1}取值来决定。Q* 就是在s_{t+1}条件下，变换不同action得到的最大的Q值

$$Q^\star(S_{t+1}, A_{t+1}) = \max_a Q^\star(S_{t+1}, a).$$

Identity: $Q^\star(s_t, a_t) = \mathbb{E}\left[\underline{R_t + \gamma \cdot \max_a Q^\star(S_{t+1}, a)}\right].$

We do not know the expectation.
Approximate it using Monte Carlo (MC).

$$\approx \boxed{r_t + \gamma \cdot \max_a Q^\star(s_{t+1}, a)}$$

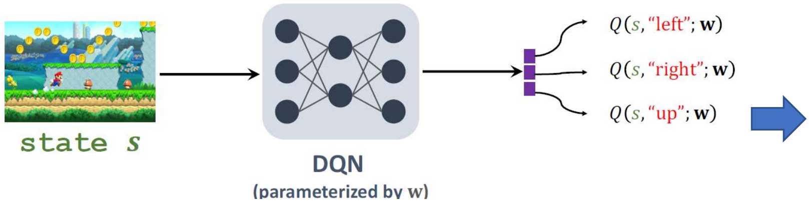
TD target y_t

Q-Learning: Tabular Version

- Observe a transition (s_t, a_t, r_t, s_{t+1}).
- TD target: $y_t = r_t + \gamma \cdot \max_a Q^\star(s_{t+1}, a)$. 查表。虽然当前的表可能还不够好。
- TD error: $\delta_t = Q^\star(s_t, a_t) - y_t$.
- Update: $Q^\star(s_t, a_t) \leftarrow Q^\star(s_t, a_t) - \alpha \cdot \delta_t$. 将更新值写入表中。

Q-Learning: DQN Version

- Approximate $Q^\star(s, a)$ by DQN, $Q(s, a; \mathbf{w})$.



- DQN controls the agent by: $a_t = \operatorname{argmax} Q(s_t, a; \mathbf{w})$.
- We seek to learn the parameter, $\mathbf{w}$.

Algorithm

- Observe a transition (s_t, a_t, r_t, s_{t+1}).
- TD target: $y_t = r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \mathbf{w})$.
- TD error: $\delta_t = Q(s_t, a_t; \mathbf{w}) - y_t$.
- Update: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \delta_t \cdot \frac{\partial\, Q(s_t, a_t; \mathbf{w})}{\partial\, \mathbf{w}}$.

Sarsa VS Q-Learning

- Sarsa is for training action-value function, $Q_\pi(s, a)$.
- TD target: $y_t = r_t + \gamma \cdot Q_\pi(s_{t+1}, a_{t+1})$.
- We used Sarsa for updating value network (critic).

- Q-learning is for training the optimal action-value function, $Q^\star(s, a)$.
- TD target: $y_t = r_t + \gamma \cdot \max_a Q^\star(s_{t+1}, a)$.
- We used Q-learning for updating DQN.

3. Multi-Step TD Target

Multi-Step Return

Identity: $U_t = \sum_{i=0}^{m-1} \gamma^i \cdot R_{t+i} + \gamma^m \cdot U_{t+m}$.

$$U_t = R_t + \gamma \cdot R_{t+1} + \gamma^2 \cdot R_{t+2} + \gamma^3 \cdot U_{t+3}.$$

- m-step TD target for **Sarsa**:

$$y_t = \sum_{i=0}^{m-1} \gamma^i \cdot r_{t+i} + \gamma^m \cdot Q_\pi(s_{t+m}, a_{t+m}).$$

- m-step TD target for **Q-learning**:

$$y_t = \sum_{i=0}^{m-1} \gamma^i \cdot r_{t+i} + \gamma^m \cdot \max_a Q^\star(s_{t+m}, a).$$

One-Step versus Multi-Step

- One-step TD target uses only one reward: r_t.
- m-step TD target uses m rewards: $r_t, r_{t+1}, r_{t+2}, \cdots, r_{t+m-1}$.
- If m is suitably tuned, m-step target works better than one-step target [1].

Reference:

1. Hossel et al. Rainbow: combining improvements in deep reinforcement learning. In *AAAI*, 2018.