Temporal Difference

Combines Monte Carlo methods and Dynamic Programming

- MC: Agent generates the trajectory and learn q-value estimates from the examples.
- MC: Agent does not have the model of the environmental dynamics, and thus you can't use the state V values but only state-action Q values.
- DP: Like Policy iteration, agent bootstraps Q values from the older Q table, unlike MC which learns using the whole trajectory.
- DP: Agent updates Q values at every iteration, unlike MC which only learns at the end of episode.

$$\begin{aligned} q_\pi(s,a) &= \mathrm{E}_\pi[G_t \mid S_t = s, A_t = a] \\ &= \sum_{r,s'} p(r,s' \mid s,a)[r + \gamma v_\pi(s')] \\ &= \sum_{r,s'} p(r,s' \mid s,a)[r + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s',a')] \end{aligned} \tag{1}$$

From above equation, we can see sampling of experience/trajectory just like in DP by following current policy π, and we bootstrap the Q value for the chosen action by the policy:

$$q_\pi(S_t, A_t) = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) \tag{2}$$

Note that Value iteration is not applicable in TD-learning, as Value Iteration does not update it's policy, π, every iteration and thus is not able to help with sampling trajectory/experience for bootstrap.

Policy Updates

Updates are based on the Temporal Difference Error:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{3}$$

where $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$ is the Temporal Difference Error.

You can redistribute the α and reexpress the update equation as below:

$$Q(S_t, A_t) \leftarrow (1 - \alpha) Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})] \tag{4}$$

which tells us that the new value to update is the **weighted average by alpha** between the old Q value and newly derived Q value via the immediate reward and the bootstrapping from the next state.

SARSA

On-Policy TD-learning which uses five elements $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ to update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]. \tag{5}$$

Since SARSA is On-Policy, the trajectory is generated following policy π, and the next action A_{t+1} is picked using ϵ-greedy policy π for bootstrapping.

Q-Learning

Off-Policy TD-learning that uses Exploratory policy b to select action with added stochasticity and Target policy π for bootstrapping:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, \pi(S_{t+1})) - Q(S_t, A_t)] \tag{6}$$

where $Q(S_t, A_t)$ is updated based on the best available action $A_{t+1} = \pi(S_{t+1}) = \arg\max_a Q(S_{t+1}, a)$ instead of stochastic ϵ-greedy (or even totally uniform random!) selection.