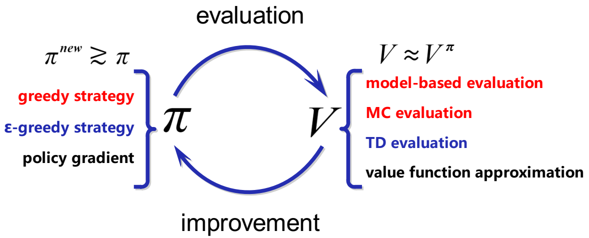
Algorithmic and Theoretical Foundations of RL

Temporal-Difference (TD) Learning

Table of Contents

Recap



- Model-based evaluation: Solve Bellman equation accurately based on model;
- MC evaluation: the simplest idea: value = mean return;
- TD evaluation: Solve Bellman equation in a stochastic and online manner.

TD(0) Policy Evaluation from Perspective of RM Algorithm

For a policy π, recall that the Bellman equation is given by

$$v_\pi(s) = \mathcal{T}_\pi v_\pi(s) = \mathbb{E}_\pi\left[r(s,a,s') + \gamma v_\pi(s')\right], \quad s \in \mathcal{S}.$$

The Bellman iteration for computing $v_\pi(s)$ is given by

$$\begin{aligned} v_{t+1}(s) &= \mathbb{E}_\pi\left[r(s,a,s') + \gamma v_t(s')\right] \\ &= v_t(s) + \alpha_t(s)\left(\mathbb{E}_\pi\left[r(s,a,s') + \gamma v_t(s')\right] - v_t(s)\right), \quad s \in \mathcal{S}. \end{aligned}$$

Given samples $\{(s,a,s')\}_{s\in\mathcal{S}}$, the RM algorithm replaces $\mathbb{E}_{a,s'}\left[r(s,a,s') + \gamma v_t(s')\right]$ by $r(s,a,s') + \gamma v_t(s')$ and yields

$$v_{t+1}(s) = v_t(s) + \alpha_t(s)\left(\left[r(s,a,s') + \gamma v_t(s')\right] - v_t(s)\right), \quad s \in \mathcal{S}.$$

- Need to repeatedly sample from every s simultaneously? Online update.

TD(0) Policy Evaluation

Algorithm 1: TD(0) Policy Evaluation

```
Initialization: $v_0(s) = 0 \; \forall \, s \in \mathcal{S}$, target policy $\pi$ and initial state $s_0$.
for $t = 0, 1, 2, \ldots$ do
    Sample a tuple $(s_t, a_t, r_t, s_{t+1}) \sim \pi$ from $s_t$
    $v_{t+1}(s_t) = v_t(s_t) + \alpha_t(s_t)\big([r_t + \gamma v_t(s_{t+1})] - v_t(s_t)\big)$
end
```

- TD(0) is a stochastic and online Bellman iteration. Note that at s_t, only a short episode is provided and there is no future information. TD complements the missing information in depth using an estimate of the next state value from previous iteration instead of the true next state value.
- At time t, only the value of the visited state s_t is updated whereas the values of the unvisited states remain unchanged (or updated using stepsize 0). If trajectory terminates, may need to restart the algorithm.
- $r_t + \gamma v_t(s_{t+1})$ is referred to as TD target while $\delta_t = [r_t + \gamma v_t(s_{t+1})] - v_t(s_t)$ is referred to as TD error. Note that $r_t + \gamma v_t(s_{t+1})$ is an unbiased estimator of $\mathcal{T}_\pi v_t(s_t)$, but a biased estimator of $v_\pi(s_t)$ since $v_\pi(s_{t+1}) \neq v(s_{t+1})$.

Convergence of TD(0)

Theorem 1. If the stepsizes α_t satisfy

$$\sum_{t=0}^{\infty} \alpha_t(s) = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2(s) < \infty$$

for all $s \in \mathcal{S}$, then $\lim_{t\to\infty} v_t = v_\pi$ with probability one.

- Note that if $s_t \neq s$ at time t, then $\alpha_t(s)$ is zero (thus the state is not updated). The conditions hold for example if in Algorithm 1 letting

$$\alpha_t(s_t) = \frac{1}{\#\mathrm{visit}(s_t)}.$$

- The proof of the theorem relies on an extension of Dvoretzky's Theorem, which is presented next.

Bounded reward is always assumed in convergence analysis.

An extension of Dvoretzky's Theorem

Theorem 2. Consider a stochastic process $(\alpha_t, \Delta_t, F_t)$, $t \geq 0$, where α_t, Δ_t, F_t: $\mathcal{X} \to \mathbb{R}$ satisfies the equation

$$\Delta_{t+1}(x) = (1 - \alpha_t(x))\,\Delta_t(x) + \alpha_t(x)\,F_t(x)\,, x \in \mathcal{X}, t = 0, 1, 2, \ldots$$

Let $\mathcal{F}_t = \sigma(\alpha_0, \ldots, \alpha_t, \Delta_0, \ldots, \Delta_t, F_0, \ldots, F_{t-1})$ be the information upto time t. Assume that the following hold:

- the set $\mathcal{X}$ is finite;
- $0 \leq \alpha_t(x) \leq 1,\ \sum_t \alpha_t(x) = \infty,\ \sum_t \alpha_t^2(x) < \infty$;
- $\|\mathbb{E}[F_t(\cdot)\,|\mathcal{F}_t]\|_\infty \leq \kappa\,\|\Delta_t\|_\infty + c_t$ where $\kappa \in [0, 1)$ and c_t converges to zero;
- $\mathrm{Var}[F_t(x)\,|\mathcal{F}_t] \leq K\left(1 + \|\Delta_t\|_\infty\right)^2$, where K is some constant.

Then Δ_t converges to zero with probability one.

MC vs TD(0)

- MC evaluation:
  - model-free, first visit G_t is unbiased for $v_\pi(s_t)$;
  - high variance: return relies on many random actions, transitions, rewards;
  - does not exploit MDP structure;
  - learns from complete episodes, no bootstrapping based on estimates that are already learned.
- TD(0) evaluation:
  - model free, TD target $r_t + v(s_{t+1})$ is biased for $v_\pi(s_t)$;
  - lower variance: TD target depends on one random action, transition, reward;
  - exploits MDP structure, usually more efficient;
  - learns from incomplete episodes (after every time step) by bootstrapping.

n-Step TD Policy Evaluation

For a policy π and $n \in \mathbb{N}^+$, define $\mathcal{T}_\pi^n$ as

$$\mathcal{T}_\pi^n v(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{n-1} \gamma^k r_k + \gamma^n v(s_n) \,|s_0 = s \right].$$

Lemma 1. For any policy π, $\mathcal{T}_\pi^n$ is a contraction with factor γ^n. Moreover, v_π is a fixed point of $\mathcal{T}_\pi^n$, i.e., $\mathcal{T}_\pi^n v_\pi = v_\pi$.

The fixed point iteration for computing v_π based on $\mathcal{T}_\pi^n$ is given by

$$\begin{aligned} v_{t+1}(s) &= T_\pi^n v_t(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{n-1} \gamma^k r_k + \gamma^n v_t(s_n) \,|s_0 = s \right] \\ &= v_t(s) + \alpha_t(s) \left(\mathbb{E}_\pi \left[\sum_{k=0}^{n-1} \gamma^k r_k + \gamma^n v_t(s_n) \,|s_0 = s \right] - v_t(s) \right), \quad s \in \mathcal{S}. \end{aligned}$$

n-Step TD Policy Evaluation

Given an episode $(s_0, a_0, r_0, s_1, a_1, r_1, \cdots, s_{n-1}, a_{n-1}, r_{n-1}, s_n) \sim \pi$ with $s_0 = s$. Define the n-step return as

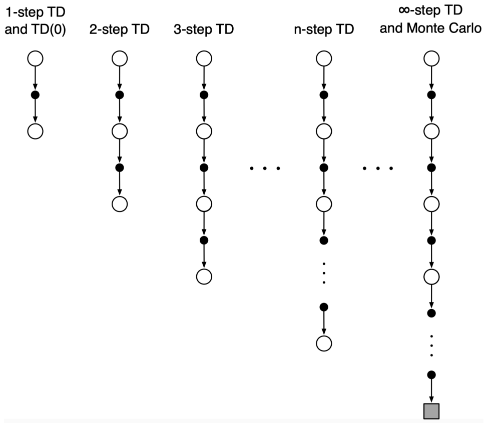
$$G^n(s) = \sum_{k=0}^{n-1} \gamma^k r_k + \gamma^n v_t\left(s_n\right).$$

Then it is easily seen that $G^n(s)$ is unbiased estimator of $\mathcal{T}_\pi^n v_t(s)$. The n-step TD method is a stochastic and online Bellman iteration associated with $\mathcal{T}_\pi^n v_t(s)$:

$$v_{t+1}(s) = v_t(s) + \alpha_t(s)\left(G^n(s) - v_t(s)\right).$$

n-Step TD Policy Evaluation

- Information may propagate back slowly in TD(0); while in MC information propagates faster, but the updates are noisier;
- n-step TD goes between TD and MC by looking n steps into the future. MC can be seen as ∞-step TD.



"Reinforcement learning: an Introduction" by Sutton and Barto, 2018.

TD(λ) Policy Evaluation

For a policy π, define the TD(λ) operator $\mathcal{T}_\pi^\lambda$ as

$$\mathcal{T}_\pi^\lambda := (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\mathcal{T}_\pi^n,$$

which is a weighted average of T_π^n.

Lemma 2. For any policy π, $\mathcal{T}_\pi^\lambda$ is a contraction with factor:

$$\frac{(1-\lambda)\,\gamma}{1-\lambda\gamma} \in (0,\gamma]\,.$$

Moreover, v_π is a fixed point of $\mathcal{T}_\pi^\lambda$, i.e., $\mathcal{T}_\pi^\lambda v_\pi = v_\pi$.

The fixed point iteration for computing v_π based on $\mathcal{T}_\pi^\lambda$ is given by

$$\begin{aligned} v_{t+1}(s) &= (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\mathcal{T}_\pi^n v_t(s) \\ &= v_t(s) + \alpha_t(s)\left((1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\mathcal{T}_\pi^n v_t(s) - v_t(s)\right), \quad s \in \mathcal{S}. \end{aligned}$$

TD(λ) Policy Evaluation

Given a trajectory $(s_0, a_0, r_0, s_1, a_1, r_1, \cdots) \sim \pi$ with $s_0 = s$, define the λ-return at s as

$$G^\lambda(s) = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G^n(s).$$

Then $G^\lambda(s)$ is unbiased estimator of $\mathcal{T}_\pi^\lambda$. The TD(λ) method is a stochastic and online Bellman iteration associated with $\mathcal{T}_\pi^\lambda$:

$$v_{t+1}(s) = v_t(s) + \alpha_t(s)\left(G^\lambda(s) - v_t(s)\right).$$

- As an episode goes, it is natural to explore the depth of the episode to update the estimate of the state value by using n-Step TD. While n-Step TD can potentially reduces bias, it suffers from high variance as n becomes large. TD(λ) provides a bias-variance tradeoff by combining all n-step information.

Here we only discuss the forward-view of TD(λ) which seems to suggest λ-return can only be computed from complete episodes. There is a backward-view of TD(λ) which provides the mechanism to update in online manner, see "Reinforcement learning with replacing eligibility traces" by Singh and Sutton, 1996.

TD(λ) Policy Evaluation

More on $G^\lambda(s)$

$$
\begin{aligned}
G^\lambda(s) &= (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G^n(s) = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\left(\sum_{k=0}^{n-1}\gamma^k r_k + \gamma^n v_t\left(s_n\right)\right)\\
&= (1-\lambda)\sum_{k=0}^{\infty}\sum_{n=k+1}^{\infty}\lambda^{n-1}\gamma^k r_k + (1-\gamma)\sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n v_t\left(s_n\right)\\
&= \sum_{t=0}^{\infty}(\lambda\gamma)^t r_t + \gamma\sum_{n=0}^{\infty}\lambda^n\gamma^n v_t(s_{n+1}) - \sum_{n=1}^{\infty}\lambda^n\gamma^n v_t(s_n)\\
&= \sum_{k=0}^{\infty}(\lambda\gamma)^k\big(\underbrace{r_k + \gamma v_t(s_{k+1}) - v_t(s_k)}_{\delta_k}\big) + v_t(s).
\end{aligned}
$$

- If $\lambda = 0$, $G^\lambda(s) = r_0 + \gamma v_t(s_1)$. That is why one-step TD is called TD(0).
- If $\lambda \to 1$, $G^\lambda(s) = \sum_{k=0}^{\infty}\gamma^k r_k$. That is why MC evaluation is also known as TD(1).

Table of Contents

TD Policy Evaluation of Action Values

While we focus on TD evaluation of state values, the TD evaluation of action values can be similarly derived. Recall that the Bellman equation for q-values is

$$\begin{aligned} q_\pi(s,a) = \mathcal{F}q_\pi(s,a) &= \sum_{s'} P(s' \mid s,a)\Big(r(s,a,s') + \gamma \sum_{a' \in \mathcal{A}} \pi(a' \mid s') q_\pi(s',a')\Big) \\ &= \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a), a' \sim \pi}\left[r(s,a,s') + \gamma q_\pi(s',a')\right], \quad (s,a) \in \mathcal{S} \times \mathcal{A}. \end{aligned}$$

The Bellman iteration for computing q-values is given by

$$\begin{aligned} q_{t+1}(s,a) &= \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a), a' \sim \pi}\left[r(s,a,s') + \gamma q_t(s',a')\right] \\ &= q_t(s,a) + \alpha_t(s,a)\left(\mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a), a' \sim \pi}\left[r(s,a,s') + \gamma q_t(s',a')\right] - q_t(s,a)\right). \end{aligned}$$

Given a random sample (s,a,r,s',a'), the RM algorithm is

$$q_{t+1}(s,a) = q_t(s,a) + \alpha_t(s,a)\left(r(s,a,s') + \gamma q_t(s',a') - q_t(s,a)\right).$$

TD(0) evaluation of actions values implements this in an online manner.

SARSA: On Policy TD Learning

```
Algorithm 2: SARSA
Initialization: $q_0(s,a)=0$, $s_0$, $\pi_0$, $a_0 \sim \pi_0(s_0)$
for $t = 0, 1, 2, \ldots$ do
    Sample a tuple $(s_t, a_t, r_t, s_{t+1}, a_{t+1}) \sim \pi_t$ from $(s_t, a_t)$
    $q_{t+1}(s_t, a_t) = q_t(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma q_t(s_{t+1}, a_{t+1}) - q_t(s_t, a_t)\right)$
    Update policy of visited state via $\epsilon_t$-greedy:
        $\pi_{t+1}(a|s_t) = \begin{cases} 1 - \epsilon_t + \frac{\epsilon_t}{|\mathcal{A}|} & \text{if } a = \operatorname{argmax}_{a'} q_{t+1}(s_t, a'), \\ \frac{\epsilon_t}{|\mathcal{A}|} & \text{otherwise.} \end{cases}$
end
```

- Sarsa is the abbreviation of “state-action-reward-state-action”, and it is an on policy algorithm which updates the policy after every time step; SARSA(λ) can also be developed based on TD(λ).

Convergence of SARSA

Theorem 3. SARSA for finite-state and finite-action MDPs converges to the optimal action-value, i.e., $q_t \to q^*$, under the following conditions:

- the policy sequence $\pi_t(a|s)$ satisfies the condition of GLIE;
- the stepsizes α_t satisfy

$$\sum_{t=0}^{\infty} \alpha_t(s, a) = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2(s, a) < \infty$$

for all (s, a).

Q-Learning: Off-Policy TD-Learning

Recall that the optimal state-action values q^* is the fixed point of the Bellman optimality operator $\mathcal{F}$ where

$$\mathcal{F}q(s,a) = \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a)} \left(r(s,a,s') + \gamma \cdot \max_{a' \in \mathcal{A}} q(s',a') \right), \quad (s,a) \in \mathcal{S} \times \mathcal{A}.$$

It can be shown that $\mathcal{F}$ is a contraction with factor γ. Assuming the model (probability transition model) is known we can find q^* by q-value iteration:

$$\begin{aligned} q_{t+1}(s,a) &= \mathcal{F}q_t(s,a) \\ &= q_t(s,a) + \alpha_t(s,a)(\mathcal{F}q_t(s,a) - q_t(s,a)), \quad (s,a) \in \mathcal{S} \times \mathcal{A}. \end{aligned}$$

Q-learning is a model free and online implementation of q-value iteration: Sample a tuple (s,a,r,s') via a behavior policy b, noting that

$$r + \gamma \cdot \max_{a' \in \mathcal{A}} q_t(s',a')$$

is an unbiased estimator of $\mathcal{F}q_t(s,a)$, we can update action-value at (s,a) by

$$q_{t+1}(s,a) = q_t(s,a) + \alpha_t(s,a) \left(r + \gamma \cdot \max_{a' \in \mathcal{A}} q_t(s',a') - q_t(s,a) \right).$$

Q-Learning: Off-Policy TD-Learning

```
Algorithm 3: Q-Learning
Initialization: $q_0(s, a) = 0$, $s_0$
for $t = 0, 1, 2, \ldots$ do
    Sample a tuple $(s_t, a_t, r_t, s_{t+1}) \sim b_t$ from $s_t$, where $b_t$ is a behavior policy
    Update $q$-value at visited state-action pair $(s_t, a_t)$:
        $q_{t+1}(s_t, a_t) = q_t(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \cdot \max_{a' \in \mathcal{A}} q_t(s_{t+1}, a') - q_t(s_t, a_t)\right)$
end
```

Remark 1. Q-Learning is off-policy since it solves for optimal action values directly which is policy irrelevant. It does not require importance sampling since behavior policy only play the role of selecting which state-action pairs will be updated. In contrast, SARSA needs data associated with the target policy to do policy evaluation in order to update the policy.

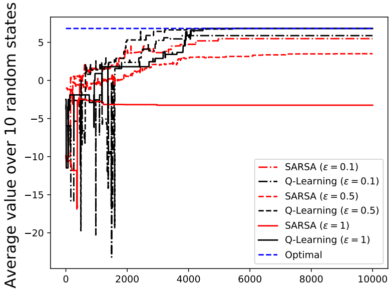
Convergence of Q-Learning

Theorem 4. Q-Learning for finite-state and finite-action MDPs converges to the optimal action-value, i.e., $q_t \to q^*$ with probability one if the stepsizes α_t satisfy

$$\sum_{t=0}^{\infty} \alpha_t(s,a) = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2(s,a) < \infty$$

for all (s,a).

Illustrative Example



For the 10 × 10 gridworld problem mentioned in Lecture 4. Stepsize is set to 0.1 in SARSA and Q-learning.

Double Q-Learning

Motivation

In Q-learning, $q_t(s, a)$ is used to approximate $q^*(s, a)$. Recall its main update is

$$q_{t+1}(s, a) = q_t(s, a) + \alpha_t(s, a) \left(r + \gamma \cdot \max_{a'} q_t(s', a') - q_t(s, a) \right).$$

A natural question is (after simplifying notation):

– *Whether $\max_a q_t(s, a)$ is an unbiased estimator of $\max_a q^*(s, a)$ if $q_t(s, a)$ is an unbiased estimator of $q^*(s, a)$?*

– *The answer is **No**!*

Double Q-Learning

Maximization Bias

Lemma 3. Let $\{\hat{\theta}_k\}_{k=1}^n$ be unbiased estimators of $\{\theta_k\}_{k=1}^n$, respectively. Then, $\max_k \hat{\theta}_k$ is a biased estimator of $\max_k \theta_k$. More precisely, there holds

$$\mathbb{E}\left[\max_k \hat{\theta}_k\right] \geq \max_k \theta_k.$$

Proof. Apply Jensen's inequality.

Example 1. Define the following two random variables:

	1/2	1/2
X_1	0	2
X_2	1	-1

It is easy to verify that $\mathbb{E}[\max(X_1, X_2)] > 1$.

Double Q-Learning

Solution: Double Estimator

Lemma 4. Let $\{\hat{\theta}_k^A\}_{k=1}^n$ and $\{\hat{\theta}_k^B\}_{k=1}^n$ be two independent sets of unbiased estimators of $\{\theta_k\}_{k=1}^n$, respectively. Define $a^* = \underset{k}{\mathrm{argmax}}\, \hat{\theta}_k^A$. Then

$$\mathbb{E}\left[\hat{\theta}_{a^*}^B\right] = \theta_{a^*} \leq \max_k \theta_k.$$

- Double Q-learning: maintain two q-tables, alternatively use one to select the action to update q-values of the other.

Double Q-Learning

```
Algorithm 4: Double Q-Learning
Initialization: $q_0^A(s,a) = q_0^B(s,a) = 0$, $s_0$
for $t = 0, 1, 2, \ldots$ do
    Sample a tuple $(s_t, a_t, r_t, s_{t+1}) \sim b_t$ from $s_t$, where $b_t$ is a behavior policy
     (e.g., $b_t$ is a $\epsilon$-greedy policy with respect to $(q_t^A + q_t^B)/2$)
    if (with $0.5$ probability) then
        Define $a^* = \underset{a}{\mathrm{argmax}}\ q_t^A(s_{t+1}, a)$
        $q_{t+1}^A(s_t, a_t) = q_t^A(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \cdot q_t^B(s_{t+1}, a^*) - q_t^A(s_t, a_t)\right)$
    else
        Define $b^* = \underset{a}{\mathrm{argmax}}\ q_t^B(s_{t+1}, a)$
        $q_{t+1}^B(s_t, a_t) = q_t^B(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \cdot q_t^A(s_{t+1}, b^*) - q_t^B(s_t, a_t)\right)$
    end
end
```

Questions?