Algorithmic and Theoretical Foundations of RL

Temporal-Difference (TD) Learning

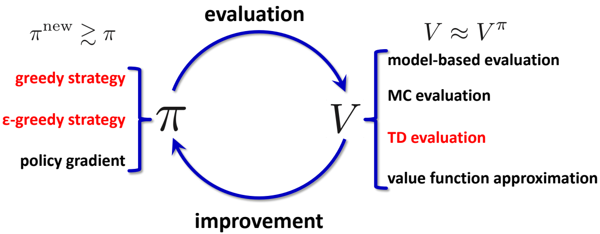
Ke Wei
School of Data Science
Fudan University

Table of Contents

TD Policy Evaluation (Prediction)

TD Learning (Control)

Recap



- Model-based evaluation: Solve Bellman equation accurately based on model;
- MC evaluation: Value estimation via sample mean;
- 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

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

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

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

Given samples $\{(s,a,s')\}_{s\in\mathcal{S}}$, RM replaces expectation by $r(s,a,s') + \gamma V^t(s')$,

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

- 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 π 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 a 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 $V^\pi(s_{t+1}) \neq V^t(s_{t+1})$.

MC vs TD(0)

- MC evaluation:
  - model-free, first visit estimator 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 $r_t + \gamma V^t(s_{t+1})$ is biased for $V^{\pi}(s_t)$;
  - lower variance: TD target relies 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

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

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

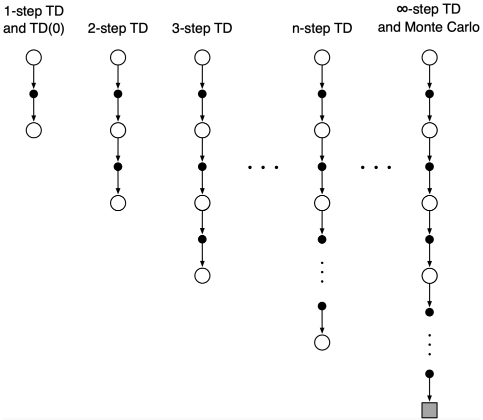
$$G^n(s) = \sum_{k=0}^{n-1} \gamma^k r_k + \gamma^n V^t(s_n).$$

It is easy to see 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.

Remark

The definition of $(\mathcal{T}^\pi)^n$ coincides with applying $\mathcal{T}^\pi$ repeatedly n times. For simplicity, consider the case $n = 2$:

$$
\begin{aligned}
&[\mathcal{T}^\pi(\mathcal{T}^\pi V)](s) \\
&= \mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma[\mathcal{T}^\pi V](s')\right] \\
&= \mathbb{E}_{a\sim\pi(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma\mathbb{E}_{a'\sim\pi(\cdot|s')}\mathbb{E}_{s''\sim P(\cdot|s',a')}\left[r(s',a',s'') + \gamma V(s'')\right]\right] \\
&= \mathbb{E}_\pi[r(s,a,s') + \gamma r(s',a',s'') + \gamma^2 V(s'')] \\
&= [(\mathcal{T}^\pi)^2 V](s).
\end{aligned}
$$

Noting that $(\mathcal{T}^\pi)^n V \to V^\pi$ when $n \to \infty$, compared TD(0) which follows target $\mathcal{T}^\pi V$, n-step TD follows target that is more accurate (thus less biased). However, variance estimating $(\mathcal{T}^\pi)^n V$ using random samples is higher that estimating $\mathcal{T}^\pi V$. To this end, TD(λ) seeks a bias-variance tradeoff by combining all n-step TD target in a suitable way.

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 $(\mathcal{T}^\pi)^n$.

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 as

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

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

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

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 $G^\lambda(s)$

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

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

Table of Contents

TD Policy Evaluation (Prediction)

TD Learning (Control)

TD Policy Evaluation of Action Values

While we focus on TD evaluation of state values, the TD evaluation of action values/Q-values can be similarly derived. The Bellman equation for Q-values is

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

The Bellman iteration for computing Q-values is given by

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

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, π_0, $a_0 \sim \pi_0(\cdot|s_0)$

for $t=0,1,2,\dots$ **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 ϵ_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).

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

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

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^* via 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, 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 $t = 0, 1, 2, \dots$ **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):

$$Q^{t+1}(s_t, a_t) = Q^t(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \cdot \max_{a' \in \mathcal{A}} Q^t\left(s_{t+1}, a'\right) - Q^t(s_t, a_t)\right)$$

end

Remark

- Q-Learning is off-policy since the behavior policy is different with the target policy (explicitly expressed via Q^t). It does not require importance sampling since behavior policy only play the role of selecting which state-action pairs will be updated, or it does not need to sample a_{t+1} for the evaluation of $\max_{a'} Q^t(s_{t+1}, a')$, or the action $a_{t+1} = \operatorname{argmax}_{a'} Q^t(s_{t+1}, a')$ is sampled from the greedy target policy and $\mathbb{E}_{a' \sim \pi(\cdot|s_{t+1})} \left[Q^t(s_{t+1}, a')\right] = Q^t(s_{t+1}, a_{t+1})$ is evaluated exactly.

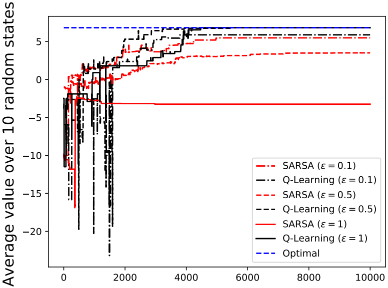
Remark

- In population version, both SARSA and Q-learning follow policy evaluation (via one-step Bellman iteration) and policy improvement update rule:

$$Q \to \pi \to \mathcal{F}^{\pi} Q \to \pi'$$

  For Q-learning, extracted policy (not defined explicitly) is greedy thus $\mathcal{F}^{\pi} Q = \mathcal{F} Q$. For SARSA, since state-action pairs are collected following target policy, to enhance exploration, we need to modify target policy from greedy one to ϵ-greedy one. In this case, $\mathcal{F}^{\pi} Q \neq \mathcal{F} Q$.

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**!*

Maximization Bias

Lemma 3

Let $\{\hat{\theta}_a\}_{a=1}^n$ *be unbiased estimators of* $\{\theta_a\}_{a=1}^n$*, respectively. Then,* $\max_a \hat{\theta}_a$ *is a biased estimator of* $\max_a \theta_a$*. More precisely, there holds*

$$\mathbb{E}\left[\max_a \hat{\theta}_a\right] \geq \max_a \theta_a.$$

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 $\mathbb{E}\left[\max(X_1, X_2)\right] > 1$.

Solution: Double Estimator

Lemma 4

Let $\{\hat{\theta}_a^A\}_{a=1}^n$ and $\{\hat{\theta}_a^B\}_{a=1}^n$ be two independent sets of unbiased estimators of $\{\theta_a\}_{a=1}^n$, respectively. Define $a^* = \operatorname*{argmax}_a \hat{\theta}_a^A$. Then

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

- 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 $t = 0, 1, 2, \dots$ **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 ϵ-greedy policy with respect to $(Q^{t,A} + Q^{t,B})/2$)

  if (with 0.5 probability) **then**

    Define $a^* = \operatorname*{argmax}_a Q^{t,A}(s_{t+1}, a)$

    $Q^{t+1,A}(s_t, a_t) = Q^{t,A}(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \cdot Q^{t,B}(s_{t+1}, a^*) - Q^{t,A}(s_t, a_t)\right)$

  else

    Define $b^* = \operatorname*{argmax}_a Q^{t,B}(s_{t+1}, a)$

    $Q^{t+1,B}(s_t, a_t) = Q^{t,B}(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \cdot Q^{t,A}(s_{t+1}, b^*) - Q^{t,B}(s_t, a_t)\right)$

  end

end

Questions?