Reinforcement Learning

Temporal Difference Learning

Jordi Casas Roma

jordi.casas.roma@uab.cat

October 7, 2025

UAB
Universitat Autònoma
de Barcelona

Table of Contents

Table of Contents

Introduction

Preliminaries

Temporal Difference (TD) learning:

- TD learning is a combination of Monte Carlo (MC) methods and dynamic programming (DP) methods.
- TD learning methods can learn directly from experience, without knowing a model of the dynamics of the environment (like Monte Carlo methods).
- TD learning methods update estimates based on other estimation values, without waiting for the final result (like DP methods).
  - This phase or process is known as **bootstrap**.

Introduction

Preliminaries

Temporal Difference (TD) learning: Advantages and strengths

- **Relative to DP methods**, the main advantage of TD methods is that they do not require a model of the environment, the return function and the probability transition function.
- **Relative to Monte Carlo methods**, TD methods are implemented as an incremental algorithm.
    - Monte Carlo methods must wait until the end of a complete episode, because the return is needed; while in TD methods only the next step is required.

Table of Contents

Table of Contents

Prediction

One-Step TD or TD(0)

One-Step TD or TD(0):

- The simplest TD method makes updates according to the following formula:

  $$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{1}$$

  That is, right at the transition $V(S_{t+1})$ and after receiving R_{t+1}.
- We can state that the objective function for Monte Carlo updates is G_t, while for TD methods it is $R_{t+1} + \gamma V(S_{t+1})$.
- α is the learning rate, which determines how much the algorithm updates the Q-values based on new information.

Prediction

One-Step TD or TD(0)

Algorithm 1 Prediction TD(0) method to estimate v_π

```
Require: Policy $\pi$ to evaluate
Require: Step $\alpha \in [0, 1)$
Initialize $V(s), \forall s \in S^+$ randomly, except $V(terminal) = 0$
for all episode do
    $S$ = Initial state
    for all each step, $t = 0$ to $T - 1$ do
        $A$ = select_action($S$) using policy $\pi$
        $R, S'$ = step($S$, $A$)
        $V(S) = V(S) + \alpha[R + \gamma V(S') - V(S)]$
        $S \leftarrow S'$
    end for
end for
return $V$
```

Prediction

One-Step TD or TD(0)

One-Step TD or TD(0):

- We notice that the value in brackets in the TD(0) updates is a kind of error, which measures the difference between the estimated value of S_t and $R_{t+1} + \gamma V(S_{t+1})$.
- This value, called TD error, is defined as:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{2}$$

- The TD error at each instant is the error of the estimate at that instant.
- Since the TD error depends on the next state and the next reward value, it is not available until the next time step. That is, δ_t is the error of the estimate of $V(S_t)$, available at time instant $t + 1$.

Table of Contents

Control

Introduction

TD learning control methods:

- Similarly to the case of Monte Carlo methods, there is the problem of intermediate solution between exploration and exploitation and, also, we have two types of approaches to address these problems:
  - **On-policy**
  - **Off-policy**
- The main TD methods control methods are:
  - **SARSA** (or TD learning control *on-policy*)
  - **Q-learning** (or TD learning control *off-policy*)

Table of Contents

Control

On-policy: SARSA

SARSA:

- The **first step** in this method is to learn an action value function (rather than a state value function).
- In particular, we must estimate $q_\pi(s, a)$ for policy π and for all states s and actions a.
- This can be carried out using the same TD method described in the previous sections to learn $v_\pi(s)$.

Control

On-policy: SARSA

SARSA:

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

- This update is done after each transition from a non-terminal state S_t.
    - If S_{t+1} is a terminal state, then $Q(S_{t+1}, A_{t+1})$ is set to zero.
- This rule uses each element of the quintuple of events $(S_r, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, which determines the transition from a state-action pair to the next.
- This quintuplet is responsible for providing the name to the algorithm, i.e. SARSA.

Control

On-policy: SARSA

SARSA:

- To deal with the exploitation vs exploration problem, we can use *ϵ-greedy* or *ϵ-soft* policies.
- SARSA converges with probability 1 to an optimal policy and an optimal action value function as long as all state-action pairs have been visited an infinite number of times.

Control

On-policy: SARSA

ϵ-**greedy and** ϵ-**soft policies**:

- A **soft** policy implies that $\pi(a|s) > 0$ for all $s \in \mathcal{S}$ and for all $a \in \mathcal{A}(s)$.
- The ϵ-**greedy policies** behave like a *greedy* policy most of the time (selecting maximum values of action value function $q_\pi(s,a)$), but with probability ε select an action at random:

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|} & \textit{if} \quad a = \textit{arg}\max_a Q(s,a) \\ \frac{\epsilon}{|\mathcal{A}(s)|} & \textit{if} \quad a \neq \textit{arg}\max_a Q(s,a) \end{cases} \tag{4}$$

where $|\mathcal{A}(s)|$ is the number of available actions for state s.

Control

On-policy: SARSA

An example of ε-greedy (soft) policy

- Let's suppose the following parameters:
  - $\varepsilon = 0.1$
  - $|\mathcal{A}(s)| = 10$
- From the equation 4 we get:

$$\pi(a|s) = \begin{cases} 1 - 0.1 + \frac{0.1}{10} = 0.91 & \textit{if} \quad a = \textit{arg}\max_a Q(s,a) \\ \frac{0.1}{10} = 0.01 & \textit{if} \quad a \neq \textit{arg}\max_a Q(s,a) \end{cases}$$

- Thus, there is a 91% chance of selecting the best action, and only a 1% chance of selecting each of the other 9 actions.

Control

On-policy: SARSA

Algorithm 2 SARSA algorithm (TD *on-policy* control method) to estimate $Q \approx q^*$

```
Require: Step $\alpha \in [0, 1)$
Require: $\epsilon > 0$ (small)
Initializer $Q(s, a), \forall s \in S^+, a \in A(s)$ randomly, except $Q(terminal, \cdot) = 0$
for all episode do
    $S$ = Initial state
    $A$ = select_action($S$) using the policy derived from $Q$ (e.g., $\epsilon$-greedy)
    for all each step, $t = 0$ to $T - 1$ do
        $R, S'$ = step($S$, $A$)
        $A'$ = select_action($S'$) using the policy derived from $Q$ (e.g., $\epsilon$-greedy)
        $Q(S, A) = Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$
        $S = S'$
        $A = A'$
    end for
end for
return $Q$
```

Control

On-policy: Expected SARSA

Expected SARSA:

- **Expected SARSA** is a variant of the SARSA algorithm that uses the expected value based on the policy instead of using the value function of a future random action selected based on the policy to be evaluated.

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha\big[R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1})|S_{t+1}] - Q(S_t, A_t)\big]$$

- By substituting the expected value, the update rule becomes:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha\big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\big]$$

Control

On-policy: Expected SARSA

Algorithm 3 Expected SARSA algorithm (TD *on-policy* control method) to estimate $Q \approx q^*$

```
Require: Step $\alpha \in [0, 1)$
Require: $\epsilon > 0$ (small)
Initializer $Q(s, a), \forall s \in S^+, a \in A(s)$ randomly, except $Q(terminal, \cdot) = 0$
for all episode do
    $S =$ Initial state
    for all each step, $t = 0$ to $T - 1$ do
        $A =$ select_action($S$) using the policy derived from $Q$ (e.g., $\epsilon$-greedy)
        $R, S' =$ step($S$, $A$)
        $Q(S, A) = Q(S, A) + \alpha[R + \gamma \sum_a \pi(a|S') Q(S', a) - Q(S, A)]$
        $S = S'$
    end for
end for
return $Q$
```

Table of Contents

Control

Off-policy: Q-learning

Q-learning:

- The *off-policy* control method known as **Q-learning** represents one of the greatest successes of all TD methods.
- The *Q-learning* method is based on the following formula:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (5)$$

- The action value function, Q, **directly approximates** q_*, the optimal action value function, regardless of the policy being followed.
  - The policy still has the effect of determining which state-action pairs are visited and updated.
- This greatly simplifies the analysis of the algorithm and allows it to converge more quickly.

Control

Off-policy: Q-learning

Algorithm 4 Q-learning (TD control method *off-policy*) to estimate $\pi = \pi^*$

```
Require: Step $\alpha \in (0, 1]$
Require: $\epsilon > 0$ (small)
Initializer $Q(s, a), \forall s \in S^+, a \in A(s)$ randomly, except $Q(terminal, \cdot) = 0$
for all episode do
    $S$ = Initial state
    for all each step, $t = 0$ to $T - 1$ do
        $A$ = select_action($S$) using the policy derived from $Q$ (e.g., $\epsilon$-greedy)
        $R, S'$ = step($S$, $A$)
        $Q(S, A) = Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
        $S = S'$
    end for
end for
return $Q$
```

Table of Contents

Biliography

References

Some relevant references:

1. **R. S. Sutton, A. G. Barto**. (2018). *Reinforcement Learning: An Introduction (Second edition)*. MIT Press, Cambridge, MA.
2. **M. Lapan**. (2024). *Deep Reinforcement Learning Hands-On (Third Edition)*. Packt Publishing.