Temporal Difference Learning: Algorithms and Optimality

A Tutorial on Core Reinforcement Learning Concepts

October 22, 2025

Abstract

Temporal Difference (TD) Learning is a model-free reinforcement learning approach that updates value function estimates by 'bootstrapping' from existing estimates rather than waiting for final outcomes. This document details the fundamental TD prediction method, its extensions for control (SARSA and Q-Learning), the concept of R-Learning for continuous tasks, and the theoretical optimality of TD(1).

Contents

1 Temporal Difference (TD) Prediction

TD Prediction aims to estimate the state-value function $V^\pi(s)$ for a given policy π. It is a model-free method that learns from experience.

1.1 TD(0) Update Rule

The simplest form, TD(0), updates the value of the current state S_t based on the immediate reward R_{t+1} and the estimated value of the next state $V(S_{t+1})$.

The update equation is:

$$V(S_t) \leftarrow V(S_t) + \alpha \left[\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{TD Target}} - V(S_t) \right]$$

Where:

- $\mathbf{V(S_t)}$: The current value estimate for state S_t.

- α: The learning rate ($0 < \alpha \leq 1$).
- γ: The discount factor ($0 \leq \gamma \leq 1$).
- The term in brackets is the **TD Error** (δ_t):

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

2 Optimality of TD(λ)

TD(λ) interpolates between TD(0) and the Monte Carlo method (MC). The parameter $\lambda \in [0, 1]$ controls the extent of the credit assignment across past states.

2.1 TD(1) as Monte Carlo (MC)

When $\lambda = 1$, TD(1) is equivalent to the Monte Carlo method.

The MC target for the value of state S_t is the **actual return** G_t, where $G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$.

$$V(S_t) \leftarrow V(S_t) + \alpha \left[G_t - V(S_t)\right]$$

2.2 Optimality in the Batch Setting

In batch learning (repeatedly processing a fixed dataset):

- **Batch MC (TD(1))**: Converges to the value function that minimizes the **mean-squared error** with respect to the observed sample returns (G_t). This is the optimal estimate against the training data itself.
- **Batch TD(0)**: Converges to the **certainty-equivalence estimate**. This is the value function that is exactly correct for the maximum-likelihood model of the underlying Markov Decision Process (MDP) given the observed data. This is often a more robust prediction for future data.

3 Temporal Difference Control Algorithms

TD control algorithms find the optimal policy π^* by estimating the action-value function $Q(s, a)$.

3.1 SARSA (On-Policy Control)

SARSA is an **on-policy** algorithm, meaning it evaluates the policy π that is used to select actions (the behavior policy). The update uses the action A_{t+1} **actually taken** in the next state S_{t+1}.

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

The sequence of events is $(\mathbf{S_t}, \mathbf{A_t}, \mathbf{R_{t+1}}, \mathbf{S_{t+1}}, \mathbf{A_{t+1}})$, where A_t and A_{t+1} are chosen using the same ϵ-greedy policy.

3.2 Q-Learning (Off-Policy Control)

Q-Learning is an **off-policy** algorithm. It learns the optimal action-value function Q^* regardless of the policy used for exploration. The target is the maximum possible Q-value for the next state, representing the optimal action.

$$\mathbf{Q(S_t, A_t)} \leftarrow \mathbf{Q(S_t, A_t)} + \alpha \left[\mathbf{R_{t+1}} + \gamma \max_{\mathbf{a}} \mathbf{Q(S_{t+1}, a)} - \mathbf{Q(S_t, A_t)}\right]$$

The action A_t is chosen via an ϵ-greedy policy (the behavior policy), but the update target (bootstrapping term) uses the optimal action $\max_a Q(S_{t+1}, a)$ (the target policy).

4 R-Learning (Average Reward Setting)

R-Learning is designed for **continuing tasks** (non-episodic) where the goal is to maximize the average reward per time step, ρ.

It simultaneously estimates the average reward ρ and the **differential value function** $D(s, a)$, which is the relative value of a state-action pair compared to the average reward.

4.1 Differential Value Update

The TD error is computed with respect to the average reward ρ:

$$D(S_t, A_t) \leftarrow D(S_t, A_t) + \alpha \left[R_{t+1} - \rho + \max_a D(S_{t+1}, a) - D(S_t, A_t) \right]$$

4.2 Average Reward Update

The average reward ρ is updated only when the estimated differential value indicates an improvement (i.e., a better-than-average step):

$$\rho \leftarrow \rho + \beta \left[R_{t+1} - \rho + \max_a D(S_{t+1}, a) - \max_a D(S_t, a) \right]$$

Where β is a separate, usually smaller, positive learning rate.

5 Games and Function Approximation

For complex environments, such as games, or those with continuous state spaces, TD methods are often combined with **function approximation** to handle large state-action spaces.

5.1 General Function Approximation

The action-value function is approximated by a parameterized function, typically a deep neural network, with parameters $\mathbf{w}$:

$$Q(s, a) \approx \hat{Q}(s, a, \mathbf{w})$$

The update involves performing stochastic gradient descent on the squared TD error:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[R_{t+1} + \gamma \max_a \hat{Q}(S_{t+1}, a, \mathbf{w}_t) - \hat{Q}(S_t, A_t, \mathbf{w}_t) \right] \nabla_{\mathbf{w}} \hat{Q}(S_t, A_t, \mathbf{w}_t)$$

This forms the basis for algorithms like **Deep Q-Networks (DQN)**.