Temporal-Difference (TD) Learning: What it is and How to Compute It

What is TD Learning?

Temporal-Difference (TD) learning estimates value functions by *bootstrapping*: it updates the value of the current state using the *observed immediate reward* plus the *current estimate* of the next state's value. Unlike Monte Carlo (MC), TD does not wait for the end of the episode; it can update after each transition.

Given a (fixed) policy π and discount factor $\gamma \in [0, 1]$, the state-value function is

$$V^\pi(s) \;=\; \mathbb{E}_\pi\big[G_t \mid S_t = s\big], \quad \text{where } G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}.$$

TD(0) Update Rule (One-Step TD)

After experiencing a transition

$$S_t = s \xrightarrow{A_t,\, R_{t+1}} S_{t+1} = s'$$

under policy π, TD(0) updates the estimate $V(s)$ as:

$$V(s) \;\leftarrow\; V(s) + \alpha\, \delta_t, \qquad \delta_t \;=\; R_{t+1} + \gamma\, V(s') - V(s),$$

where δ_t is the *TD error* and $\alpha > 0$ is the step size.

Key ideas.

- R_{t+1} is observed now.
- $V(s')$ is your *current* estimate of the next state's value (look it up from your value table or function approximator).
- No need to wait for episode termination (unlike MC).

Pseudocode (On-Policy Evaluation, Tabular TD(0))

```
Initialize V(s) arbitrarily (e.g., 0) for all states s
Repeat (for many episodes):
    Initialize S
    Repeat (for each step of the episode):
        Choose A ~ (·|S)           (the evaluation policy)
        Take action A, observe R, S'
         ← R +  V(S') - V(S)      (TD error)
        V(S) ← V(S) +             (TD update)
        S ← S'
    until S is terminal
```

Fully Worked Numerical Example

Consider three states $A \to B \to$ terminal with rewards:

$$A \xrightarrow{0} B \xrightarrow{1} \text{terminal.}$$

Let $\gamma = 1$ and $\alpha = 0.5$. Initialize $V(A) = V(B) = 0$ (terminal has value 0).

Episode 1

$$\begin{aligned}
\textbf{Step 1: } & S = A,\ R = 0,\ S' = B \\
& \delta = 0 + 1 \cdot V(B) - V(A) = 0 - 0 = 0 \\
& V(A) \leftarrow 0 + 0.5 \cdot 0 = 0. \\
[6pt]\textbf{Step 2: } & S = B,\ R = 1,\ S' = \text{terminal} \\
& \delta = 1 + 1 \cdot 0 - V(B) = 1 - 0 = 1 \\
& V(B) \leftarrow 0 + 0.5 \cdot 1 = 0.5.
\end{aligned}$$

After Episode 1: $V(A) = 0,\ V(B) = 0.5.$

Episode 2

$$\begin{aligned}
\textbf{Step 1: } & S = A,\ R = 0,\ S' = B \\
& \delta = 0 + 1 \cdot V(B) - V(A) = 0.5 - 0 = 0.5 \\
& V(A) \leftarrow 0 + 0.5 \cdot 0.5 = 0.25. \\
[6pt]\textbf{Step 2: } & S = B,\ R = 1,\ S' = \text{terminal} \\
& \delta = 1 + 1 \cdot 0 - V(B) = 1 - 0.5 = 0.5 \\
& V(B) \leftarrow 0.5 + 0.5 \cdot 0.5 = 0.75.
\end{aligned}$$

After Episode 2: $V(A) = 0.25,\ V(B) = 0.75.$

Episode 3

$$
\begin{aligned}
\textbf{Step 1:}\ & S = A,\ R = 0,\ S' = B \\
& \delta = 0 + 1 \cdot 0.75 - 0.25 = 0.5 \\
& V(A) \leftarrow 0.25 + 0.5 \cdot 0.5 = 0.5. \\
[6pt]\textbf{Step 2:}\ & S = B,\ R = 1,\ S' = \text{terminal} \\
& \delta = 1 - 0.75 = 0.25 \\
& V(B) \leftarrow 0.75 + 0.5 \cdot 0.25 = 0.875.
\end{aligned}
$$

After Episode 3: $V(A) = 0.5,\ V(B) = 0.875$.

As learning continues (with sufficient visits), $V(B)$ approaches 1 (the expected return from B), and $V(A)$ approaches 1 as well (since from A you go to B and then receive reward 1 next step).

TD vs. Monte Carlo (MC): Quick Contrast

- **MC:** waits for the full return G_t; updates after episode ends.
  Update (incremental form): $V(s) \leftarrow V(s) + \alpha\big(G_t - V(s)\big)$.
- **TD(0):** updates *during* the episode using one-step bootstrap target $R_{t+1} + \gamma V(s_{t+1})$.

Function Approximation (Optional Note)

If $V(s)$ is represented by a differentiable function $V(s; \boldsymbol{w})$ (e.g., a neural net with weights $\boldsymbol{w}$), TD(0) performs a stochastic gradient step on the squared TD error:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\, \delta_t\, \nabla_{\boldsymbol{w}} V(s_t; \boldsymbol{w}), \quad \delta_t = R_{t+1} + \gamma V(s_{t+1}; \boldsymbol{w}) - V(s_t; \boldsymbol{w}).$$