6 Temporal Difference Learning

Melih Kandemir

Özyeğin University
Computer Science Department
melih.kandemir@ozyegin.edu.tr

31 Oct 2017

Temporal Difference (TD) Learning

- is most central and novel idea for the RL field.
- is model-free, like MC and unlike DP.
- **bootstraps:** updates estimates based on other estimates.
- TD-MC-DP relationship is at the heart of the RL theory.
- All RL methods differ only in prediction. All perform GPI for control.

Constant-α MC

- Define a simple every-visit MC method as

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[G_t - V(S_t) \Big],$$

  where α is the step size. Denote this method as *constant-α MC*.
- The main drawback of this method is that it can update parameters only after an episode ends.

The temporal difference

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{target} - \underbrace{V(S_t)}_{prediction} \Big]$$

- Update the value estimate every time step, not every episode!
- Convert RL into a supervised learning problem:
  Minimize the error between the target and the prediction!
- (target-prediction) is referred to as the *TD error*.
- This method is called *one-step TD* or *TD(0)*.

The TD(0) algorithm

input: the policy π to be evaluated
initialize $V(s)$ arbitrarily

repeat for each episode
  initialize S
  repeat for each step of episode
    $A \leftarrow$ action given by π for S
    take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\Big[R + \gamma V(S') - V(S)\Big]$
    $S \leftarrow S'$
  until S is terminal

DP vs MC vs TD

$$
\begin{aligned}
v_\pi(s) &\triangleq \mathbb{E}_\pi[G_t | S_t = s] && \text{Target for MC} && (1)\\
&= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] && \text{Target for DP} && (2)\\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] && \text{Target for TD} && (3)
\end{aligned}
$$

- (1) is a target for MC, because $\mathbb{E}_\pi[\cdot]$ is estimated by sampling.
- (2) is a target for DP, because $v_\pi(S_{t+1})$ estimated by $V(S_{t+1})$.
- (3) is a target for TD, because both $\mathbb{E}_\pi[\cdot]$ is sampled and $v_\pi(S_{t+1})$ estimated by $V(S_{t+1})$.

Sample versus full backups

- **Sample backup** looks ahead to a sample successor state (MC and TD)
- **Full backup** updates based on a complete distribution of all possible successor states (DP)

Monte Carlo Backup

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

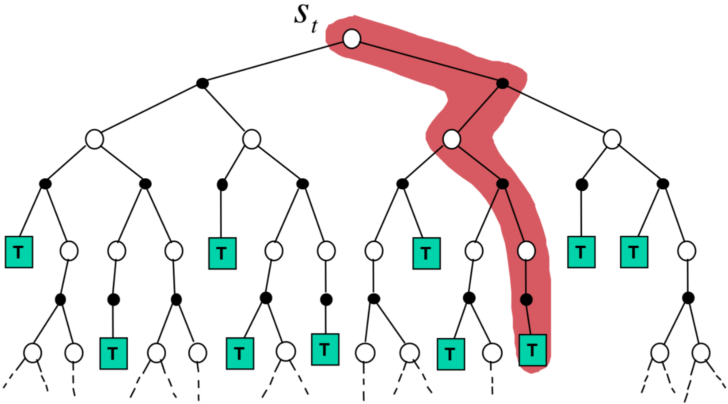


Figure. D. Silver, lecture slides

Temporal Difference Backup

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$

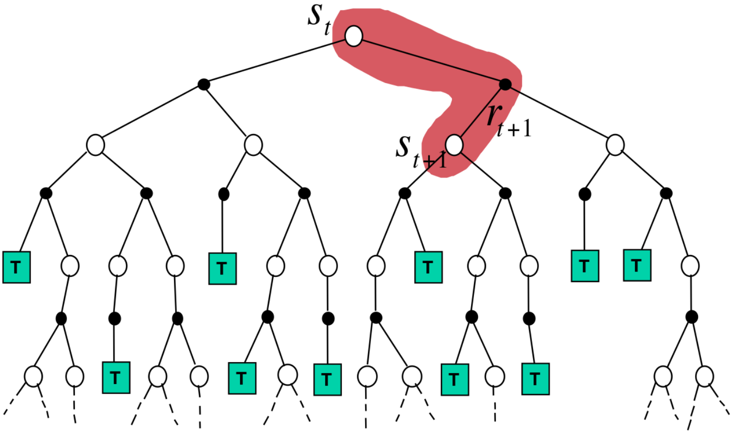


Figure. D. Silver, lecture slides

Dynamic Programming Backup

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1})\right]$$

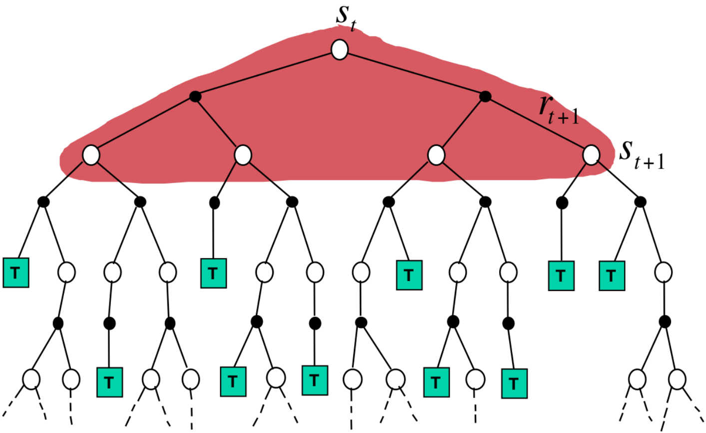


Figure. D. Silver, lecture slides

TD error and MC error

TD error (available at $t+1$)

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

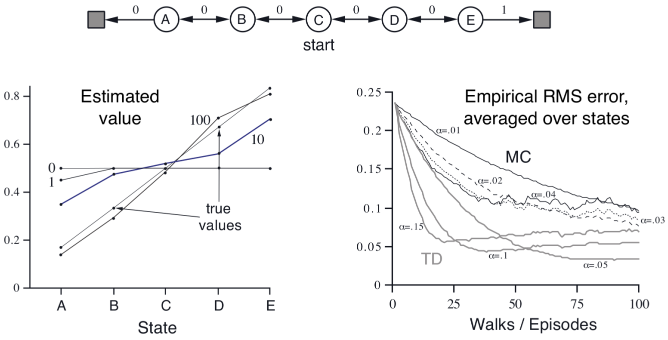
Called TD(0) because, this is the error made at time 0.
If V does not change during the current episode, then the MC error and TD error have the relationship below

$$\begin{aligned}
G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}) \\
&= \delta_t + \gamma (G_{t+1} - V(S_{t+1})) \\
&= \delta_t + \gamma \delta_{t+1} + \gamma^2 (G_{t+2} - V(S_{t+2})) \\
&= \delta_t + \gamma \delta_{t+1} + \gamma^2 \delta_{t+2} + \cdots + \gamma^{T-t} (G_T - V(S_T)) \\
&= \delta_t + \gamma \delta_{t+1} + \gamma^2 \delta_{t+2} + \cdots + \gamma^{T-t} (0 - 0) \\
&= \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k
\end{aligned}$$

Advantages of TD

- TD is model-free
- TD is on-line within the episode
- TD(0) converges to v_π
- Is TD faster than MC? Yes in practice, but no proof yet

Example: Random Walk



Set $\gamma = 1$

$$V(A) = \frac{1}{6}, \quad V(B) = \frac{2}{6}, \quad V(C) = \frac{3}{6}, \quad V(D) = \frac{4}{6}, \quad V(E) = \frac{5}{6}$$

Figure. R. Sutton and A. Barto, MIT Press, 2017

Batch Training with TD(0)

- If observations are limited (e.g. 10 episodes), present them to the RL algorithm repeatedly until convergence.
- Calculate all increments, but change the value function by the sum of them at the end of the full pass on all observations (epoch).
- This is called *batch training*.
- Under batch training, both TD and MC converge to unique but different answers.

Random Walk with Batch Training

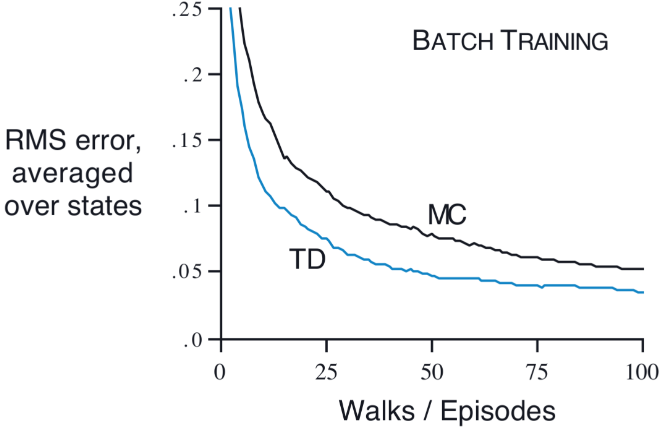


Figure. R. Sutton and A. Barto, MIT Press, 2017

Batch MC vs Batch TD

- Batch MC minimizes MSE on the training set.
- Batch TD estimates the maximum-likelihood model of the Markov Decision Process

$$P(s'|s) = \frac{\#observed\ transitions\ from\ s\ to\ s'}{\#observed\ transitions\ from\ s}$$

$$\mathbb{E}[R] = avg\ rewards\ during\ s\ to\ s'\ transitions$$

Batch MC vs Batch TD

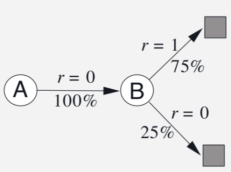


Figure. R. Sutton, A. Barto, MIT Press, 2017

$$\begin{array}{ccc} A,0,B,0 & B,1 & B,1 \\ B,1 & B,1 & B,0 \\ B,1 & B,1 & \end{array}$$

$V(B) = 3/4$, because $R = 1$ in 4 out of 6 cases.

- $V(A) = 0$ for MC, because we haven seen A once and the related return was 0.
- $V(A) = 3/4$, because $A \to B$ is 100% and $V(B) = 3/4$.

Certainty Equivalence

- MC converges to the solution with minimum MSE

$$\sum_{k=1}^{K}\sum_{t=1}^{T_k}(G_t^k - V(s_t^k))^2$$

- TD(0) converges to the MLE of the MDP

$$p(s'|s,a) = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\delta_{s_t^k=s \ \& \ a_t^k=a \ \& \ s_{t+1}^k=s'}$$

$$p(r|s,a) = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\delta_{s_t^k=s \ \& \ a_t^k=a} r_t^k$$

The second is called the **certainty equivalence estimate**, because it assures the value estimate to converge to its exact value, as the MDP converges to its exact value (after infinitely many samples are collected).

On-policy TD control: Sarsa

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$ and $Q(s_{end}, \cdot) = 0$
repeat (for each episode)
  Initialize S
  Choose A from S using policy derived from Q
  (e.g., $\epsilon-$greedy)
  repeat (for each step of episode)
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q
    (e.g., $\epsilon-$greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S', A') - Q(S,A)]$
    $S \leftarrow S';\ A \leftarrow A'$
  until S is terminal

Off-policy TD control: Q-learning

- Avoid importance sampling (introduces variance)
- Choose next action following an $\epsilon-$greedy behavior policy wrt $Q(s,a)$.
- Calculate the target by a greedy policy wrt $Q(s,a)$

$$
\begin{aligned}
& R_{t+1} + \gamma Q(S_{t+1}, A') \\
& \quad = R_{t+1} + \gamma Q(S_{t+1}, \max_{a'} Q(S_{t+1}, a')) \\
& \quad = R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')
\end{aligned}
$$

The Q-learning algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$ and $Q(s_{end}, \cdot) = 0$
repeat (for each episode)
  Initialize S
  repeat (for each step of episode)
    Choose A from S using policy derived from Q
    (e.g., $\epsilon-$greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S,A)]$
    $S \leftarrow S'$
  until S is terminal

Cliff Walk

- Sarsa samples S', which covers the whole set action selection into account, hence chooses the long and safe path.
- Q-learning follows the $\epsilon-$greedy policy policy, hence takes the optimal path but occasionally falls down (has worse online performance).

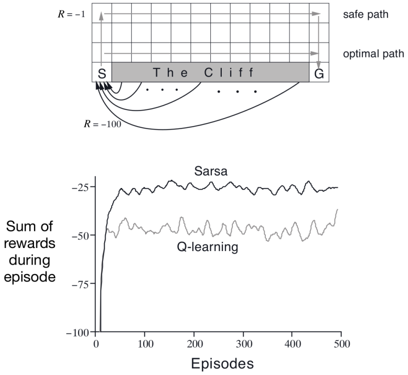


Figure: R. Sutton and A. Barto, MIT Press, 2017

Expected Sarsa

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1})|S_{t+1}] - Q(S_t, A_t)\Big] \\
&\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \underbrace{\sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)}_{in\ place\ of\ \epsilon-greedy} - Q(S_t, A_t)\Big]
\end{aligned}
$$

- (+) Has less variance than sarsa, as A' is no longer chosen at random.
- (+) Much less sensitive to the choice of α. Works well on $\alpha = 1$, while sarsa requires $\alpha << 1$.
- (-) Has more computational complexity, as all actions are swept in every update.

Asymptotic Cliff Walk

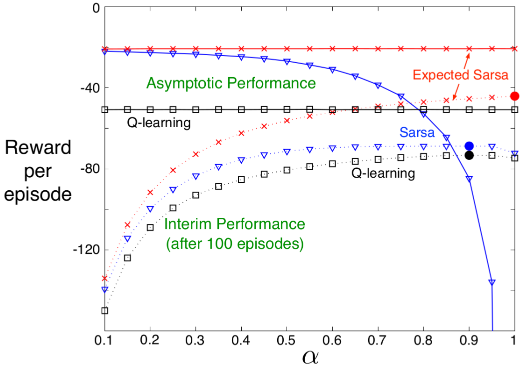


Figure. R. Sutton and A. Barto, MIT Press, 2017

- **Asymptotic performance:** average over 100000 episodes.
- **Interim performance:** average over the first 100 episodes.

Maximization bias and double learning

- Many RL algorithms use the following approximation:

$$\max(\mathbb{E}[a], \mathbb{E}[b]) = \mathbb{E}[\max(a, b)]$$

- A maximum over estimated values are used in place of an estimate of the maximum value.
- As $\max(\mathbb{E}[a], \mathbb{E}[b])$ is prone to generate estimations greater than $\mathbb{E}[\max(a, b)]$, the bias resulting from this approximation is called the **maximization bias**.

Double learning

- The maximization bias problem emerges from using the same samples both to determine the maximizing action and to estimate its value.
- A solution is to use
  - one estimate Q_1 to determine the maximizing action $A^* = \underset{a}{\operatorname{argmax}}\, Q_1(a)$,
  - another estimate Q_2 to estimate its value $Q_2(A^*) = Q_2(\underset{a}{\operatorname{argmax}}\, Q_1(a))$.
- The outcome is an unbiased estimate of the value of the maximizing action $\mathbb{E}[Q_2(A^*)] = q(A^*)$.
- The trick can be used anywhere: Q-learning, Sarsa, Expected Sarsa, etc.

The double Q-learning algorithm

Initialize $Q_1(s, a)$ and $Q_2(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q_1(s_{end}, \cdot) = Q_2(s_{end}, \cdot) = 0$
repeat (for each episode)
  Initialize S
  repeat (for each step of episode)
    Choose A from S using policy derived from Q_1 and Q_2
    (e.g., $\epsilon-$greedy in $Q_1 + Q_2$)
    Take action A, observe R, S'
    With probability 0.5:

$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha(R + \gamma Q_2(S', \operatorname*{argmax}_a Q_1(S', a)) - Q_1(S, A))$$

    else:

$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha(R + \gamma Q_1(S', \operatorname*{argmax}_a Q_2(S', a)) - Q_2(S, A))$$

    $S \leftarrow S'$
  until S is terminal

Q-learning versus Double Q-learning

Q-learning will choose **left** 5% more often even at the asymptote!

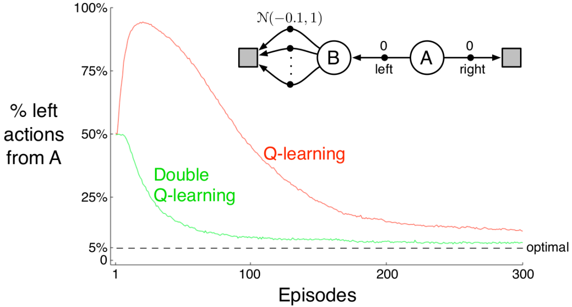


Figure. R. Sutton and A. Barto, MIT Press, 2017

Importance sampling for off-Policy TD

- Choose an arbitrary b (no longer has to be the $\epsilon-$greedy version of π).
- Only one correction required (i.e. much lower variance than MC-IS).
- It suffices for b and π to resemble only on the current time step.

$$V(S_t) \leftarrow V(S_t) + \alpha \left[\frac{\pi(A_t|S_t)}{b(A_t|S_t)} \Big(R_{t+1} - \gamma V(S_{t+1}) \Big) - V(S_t) \right]$$

MC versus TD (1)

- **Goal:** Estimate v_π for a given π
- Incremental MC
  - Update $V(S_t)$ towards **actual** return G_t:

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

- TD(0)
  - Update $V(S_t)$ towards **estimated** return $R_{t+1} + \gamma V(S_{t+1})$:

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

MC versus TD (2)

- TD can learn *before* the final outcome is observed
  - TD learns online from every state transition
  - MC has to wait the episode end to calculate the return
- TD can learn *without* the final outcome
  - TD can learn from incomplete sequences (i.e. works in continuing environments)
  - MC can only learn from complete sequences (i.e. works only in episodic environments)
- TD exploits Markov property, MC does not
  - TD works better if the environment is Markov
  - MC can better handle non-stationarity

MC versus TD (3)

The bias-variance trade-off in RL:

- MC has high variance but zero bias, TD has low variance but bias.
  - TD target depends on *one* random (S, A, R) tuple, MC depends on many.
  - TD is more sensitive to initialization than MC.
- $G_t = R_{t+1} + \gamma R_{t+2} + \cdots \gamma^{T-1} R_T$ is an **unbiased** estimate of $v_\pi(S_t)$.
- $R_{t+1} + \gamma {\color{blue} v_\pi(S_{t+1})}$ is an **unbiased** estimate of $v_\pi(S_t)$.
- $R_{t+1} + \gamma {\color{red} V(S_{t+1})}$ is an **biased** estimate of $v_\pi(S_t)$.

MC versus TD (4)

- **Bootstrapping:** update by estimates
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- **Sampling:** update by sampling from the expectation
  - MC samples
  - DP does not sample
  - TD samples

MC and TD in common

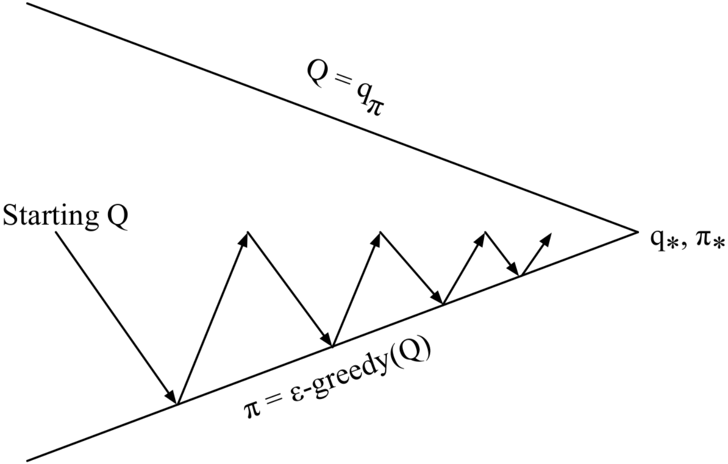


Figure. D. Silver, lecture slides

DP vs TD

Full Backup (DP)	*Sample Backup (TD)*
Iterative Policy Evaluation	TD Learning
$V(s) \leftarrow \mathbb{E}\left[R + \gamma V(S') \mid s\right]$	$V(S) \xleftarrow{\alpha} R + \gamma V(S')$
Q-Policy Iteration	Sarsa
$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma Q(S',A') \mid s,a\right]$	$Q(S,A) \xleftarrow{\alpha} R + \gamma Q(S',A')$
Q-Value Iteration	Q-Learning
$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S',a') \mid s,a\right]$	$Q(S,A) \xleftarrow{\alpha} R + \gamma \max_{a' \in \mathcal{A}} Q(S',a')$

where $x \xleftarrow{\alpha} y \equiv x \leftarrow x + \alpha(y - x)$

Table. D. Silver, lecture slides

Unified view of RL algorithms

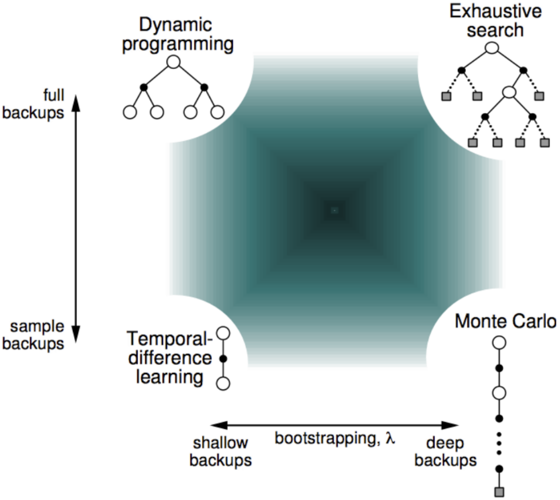


Figure. D. Silver, lecture slides