Lecture 05: Temporal-Difference Learning

Davit Ghazaryan

February 27, 2025

Temporal-Difference (TD) Learning

If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning. TD learning is a combination of Monte Carlo ideas and dynamic programming (DP) ideas.

Temporal-Difference (TD) Learning

If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning. TD learning is a combination of Monte Carlo ideas and dynamic programming (DP) ideas. Like Monte Carlo methods, TD methods can learn directly from raw experience without a model of the environment's dynamics.

Temporal-Difference (TD) Learning

If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning. TD learning is a combination of Monte Carlo ideas and dynamic programming (DP) ideas. Like Monte Carlo methods, TD methods can learn directly from raw experience without a model of the environment's dynamics. Like DP, TD methods update estimates based in part on other learned estimates, without waiting for a final outcome (they bootstrap).
(source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018)

Temporal-difference learning and the previous methods

Temporal-difference (TD) learning combines the previous ideas introduced in DP and MC:

- From Monte Carlo (MC) methods: Learns directly from experience.
- From dynamic programming (DP): Updates estimates based on other learned estimates (bootstrap).

Temporal-difference learning and the previous methods

Temporal-difference (TD) learning combines the previous ideas introduced in DP and MC:

- From Monte Carlo (MC) methods: Learns directly from experience.
- From dynamic programming (DP): Updates estimates based on other learned estimates (bootstrap).

Hence, TD characteristics are:

- Allows model-free prediction and control in unknown MDPs.

Temporal-difference learning and the previous methods

Temporal-difference (TD) learning combines the previous ideas introduced in DP and MC:

- From Monte Carlo (MC) methods: Learns directly from experience.
- From dynamic programming (DP): Updates estimates based on other learned estimates (bootstrap).

Hence, TD characteristics are:

- Allows model-free prediction and control in unknown MDPs.
- Updates policy evaluation and improvement in an online fashion (i.e., not per episode) by bootstrapping.

Temporal-difference learning and the previous methods

Temporal-difference (TD) learning combines the previous ideas introduced in DP and MC:

- From Monte Carlo (MC) methods: Learns directly from experience.
- From dynamic programming (DP): Updates estimates based on other learned estimates (bootstrap).

Hence, TD characteristics are:

- Allows model-free prediction and control in unknown MDPs.
- Updates policy evaluation and improvement in an online fashion (i.e., not per episode) by bootstrapping.
- Still assumes finite MDP problems (or problems close to that).

Table of contents

General TD prediction updates

Recap the every-visit MC update rule (4.3) for non-stationary problems:

$$\hat{v}(s_t) \leftarrow \hat{v}(s_t) + \alpha \left[G_t - \hat{v}(s_t) \right]. \tag{5.1}$$

- $\alpha \in \{\mathbb{R} | 0 < \alpha < 1\}$ is the forgetting factor / step size.
- G_t is the target of the incremental update rule.
- To execute the update (5.1) one has to wait until the episode's termination since only then G_t is available (MC requirement).

General TD prediction updates

Recap the every-visit MC update rule (4.3) for non-stationary problems:

$$\hat{v}(s_t) \leftarrow \hat{v}(s_t) + \alpha \left[G_t - \hat{v}(s_t) \right]. \tag{5.1}$$

- $\alpha \in \{\mathbb{R} | 0 < \alpha < 1\}$ is the forgetting factor / step size.
- G_t is the target of the incremental update rule.
- To execute the update (5.1) one has to wait until the episode's termination since only then G_t is available (MC requirement).

One-step TD / TD(0) update

$$\hat{v}(s_t) \leftarrow \hat{v}(s_t) + \alpha \left[R_{t+1} + \gamma \hat{v}(s_{t+1}) - \hat{v}(s_t) \right]. \tag{5.2}$$

- Here, the TD target is $R_{t+1} + \gamma \hat{v}(s_{t+1})$.
- TD is bootstrapping: estimate $\hat{v}(s_t)$ based on $\hat{v}(s_{t+1})$.
- Delay time of one step and no need to wait until the episode's end.

Algorithmic implementation: TD-based prediction

input: a policy π to be evaluated
output: estimate of $\boldsymbol{v}_{\mathcal{S}}^{\pi}$ (i.e., value estimates for all states $s \in \mathcal{S}$)
init: $\hat{v}(s) \, \forall \, s \in \mathcal{S}$ arbitrary except $v_0(s) = 0$ if s is terminal
for $j = 1, \ldots, J$ *episodes* **do**
    Initialize s_0;
    for $t = 0, 1, 2 \ldots$ *time steps* **do**
        $a_t \leftarrow$ apply action from $\pi(s_t)$;
        Observe s_{t+1} and R_{t+1};
        $\hat{v}(s_t) \leftarrow \hat{v}(s_t) + \alpha \left[R_{t+1} + \gamma \hat{v}(s_{t+1}) - \hat{v}(s_t) \right]$;
        Exit loop if s_{t+1} is terminal;

Algo. 5.1: Tabular TD(0) prediction

- Note that the algorithm can be directly adapted to action-value prediction as it will be used for the later TD-based control approaches.

TD error

Fig. 5.1: Back up diagram for TD(0)

- TD as well as MC use sample updates.
- Looking ahead to a sample successor state including its value and the reward along the way to compute a backed up value estimate.

TD error

Fig. 5.1: Back up diagram for TD(0)

- TD as well as MC use sample updates.
- Looking ahead to a sample successor state including its value and the reward along the way to compute a backed up value estimate.

The TD error is:

$$\delta_t = R_{t+1} + \gamma \hat{v}(s_{t+1}) - \hat{v}(s_t). \tag{5.3}$$

- δ_t is available at time step $t + 1$.
- Iteratively δ_t converges towards zero.

TD error and its relation to the MC error

Let's assume that the TD(0) estimate $\hat{v}(s)$ is not changing over one episode as it would be for MC prediction:

$$\underbrace{G_t - \hat{v}(s_t)}_{\text{MC-error}} = R_{t+1} + \gamma G_{t+1} - \hat{v}(s_t) + \gamma \hat{v}(s_{t+1}) - \gamma \hat{v}(s_{t+1}), \tag{5.4}$$

TD error and its relation to the MC error

Let's assume that the TD(0) estimate $\hat{v}(s)$ is not changing over one episode as it would be for MC prediction:

$$\begin{aligned}\underbrace{G_t - \hat{v}(s_t)}_{\text{MC-error}} &= R_{t+1} + \gamma G_{t+1} - \hat{v}(s_t) + \gamma\hat{v}(s_{t+1}) - \gamma\hat{v}(s_{t+1}),\\ &= \delta_t + \gamma(G_{t+1} - \hat{v}(s_{t+1})),\end{aligned} \tag{5.4}$$

TD error and its relation to the MC error

Let's assume that the TD(0) estimate $\hat{v}(s)$ is not changing over one episode as it would be for MC prediction:

$$\begin{aligned}\underbrace{G_t - \hat{v}(s_t)}_{\text{MC-error}} &= R_{t+1} + \gamma G_{t+1} - \hat{v}(s_t) + \gamma\hat{v}(s_{t+1}) - \gamma\hat{v}(s_{t+1}),\\ &= \delta_t + \gamma(G_{t+1} - \hat{v}(s_{t+1})),\\ &= \delta_t + \gamma\delta_{t+1} + \gamma^2(G_{t+2} - \hat{v}(s_{t+2})),\end{aligned} \tag{5.4}$$

TD error and its relation to the MC error

Let's assume that the TD(0) estimate $\hat{v}(s)$ is not changing over one episode as it would be for MC prediction:

$$\begin{aligned}\underbrace{G_t - \hat{v}(s_t)}_{\text{MC-error}} &= R_{t+1} + \gamma G_{t+1} - \hat{v}(s_t) + \gamma\hat{v}(s_{t+1}) - \gamma\hat{v}(s_{t+1}),\\ &= \delta_t + \gamma(G_{t+1} - \hat{v}(s_{t+1})),\\ &= \delta_t + \gamma\delta_{t+1} + \gamma^2(G_{t+2} - \hat{v}(s_{t+2})),\\ &= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \gamma^3(G_{t+3} - \hat{v}(s_{t+3})) = \cdots,\end{aligned} \tag{5.4}$$

TD error and its relation to the MC error

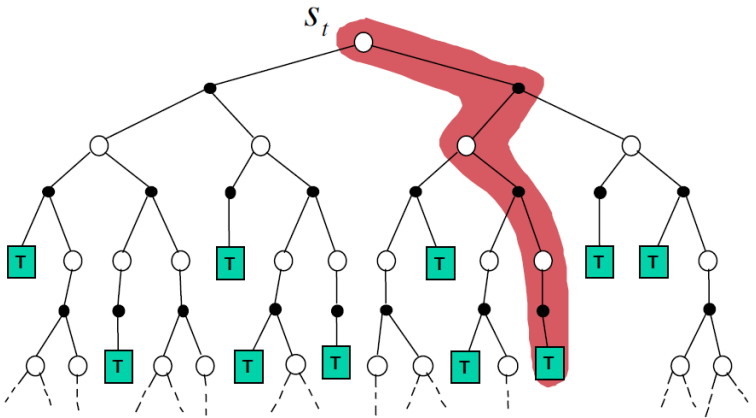
Let's assume that the TD(0) estimate $\hat{v}(s)$ is not changing over one episode as it would be for MC prediction:

$$\begin{aligned}
\underbrace{G_t - \hat{v}(s_t)}_{\text{MC-error}} &= R_{t+1} + \gamma G_{t+1} - \hat{v}(s_t) + \gamma\hat{v}(s_{t+1}) - \gamma\hat{v}(s_{t+1}), \\
&= \delta_t + \gamma(G_{t+1} - \hat{v}(s_{t+1})), \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2(G_{t+2} - \hat{v}(s_{t+2})), \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \gamma^3(G_{t+3} - \hat{v}(s_{t+3})) = \cdots, \\
&= \sum_{i=t}^{T-1} \gamma^{i-t}\delta_i.
\end{aligned} \tag{5.4}$$

- MC error is the discounted sum of TD errors in this simplified case.
- If $\hat{v}(s)$ is updated during an episode (as expected in TD(0)), the above identity only holds approximately.

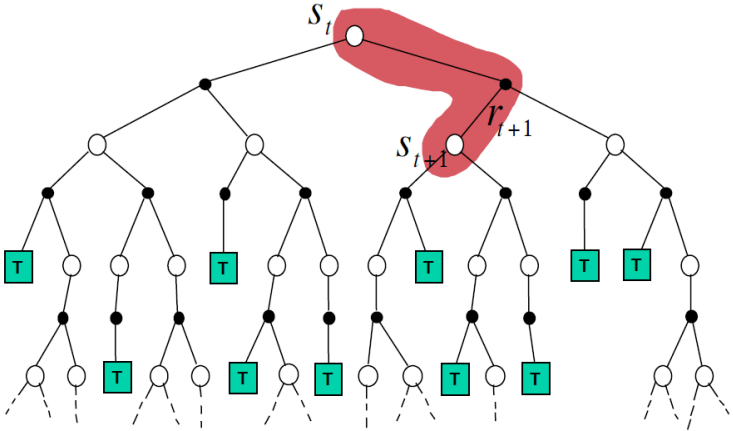
Monte Carlo Backup

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



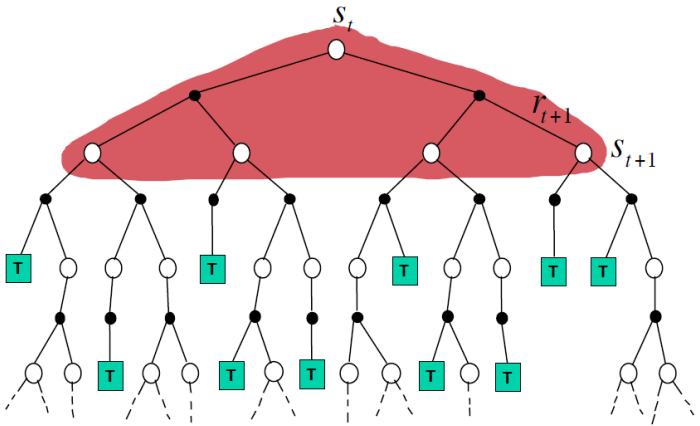
Temporal Difference Learning Backup

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



Dynamic Programming Backup

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



Overview of the RL methods considered so far

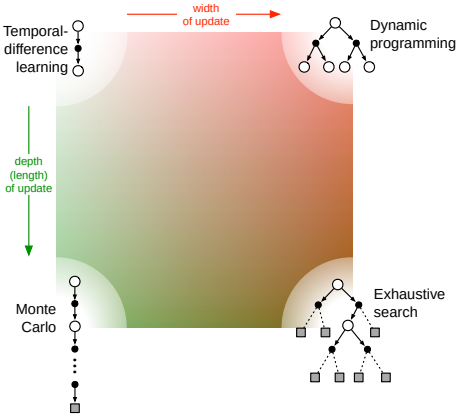


Fig. 5.2: Comparison of the RL methods considered so far with regard to the update rules (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018)

Driving home example

Each day as you drive home from work, you try to predict how long it will take to get home. When you leave your office, you note the time, the day of the week, the weather, and anything else that might be relevant.

Driving home example

Each day as you drive home from work, you try to predict how long it will take to get home. When you leave your office, you note the time, the day of the week, the weather, and anything else that might be relevant. Say on this Friday you are leaving at exactly 6, and you estimate that it will take 30 minutes to get home.

Driving home example

Each day as you drive home from work, you try to predict how long it will take to get home. When you leave your office, you note the time, the day of the week, the weather, and anything else that might be relevant. Say on this Friday you are leaving at exactly 6, and you estimate that it will take 30 minutes to get home. As you reach your car it is 6:05, and you notice it is starting to rain.

Driving home example

Each day as you drive home from work, you try to predict how long it will take to get home. When you leave your office, you note the time, the day of the week, the weather, and anything else that might be relevant. Say on this Friday you are leaving at exactly 6, and you estimate that it will take 30 minutes to get home. As you reach your car it is 6:05, and you notice it is starting to rain. The traffic is often slower in the rain, so you reestimate that it will take 35 minutes from then, or a total of 40 minutes.

Driving home example

Each day as you drive home from work, you try to predict how long it will take to get home. When you leave your office, you note the time, the day of the week, the weather, and anything else that might be relevant. Say on this Friday you are leaving at exactly 6, and you estimate that it will take 30 minutes to get home. As you reach your car it is 6:05, and you notice it is starting to rain. The traffic is often slower in the rain, so you reestimate that it will take 35 minutes from then, or a total of 40 minutes. Fifteen minutes later you have completed the highway portion of your journey in good time.

Driving home example

Each day as you drive home from work, you try to predict how long it will take to get home. When you leave your office, you note the time, the day of the week, the weather, and anything else that might be relevant. Say on this Friday you are leaving at exactly 6, and you estimate that it will take 30 minutes to get home. As you reach your car it is 6:05, and you notice it is starting to rain. The traffic is often slower in the rain, so you reestimate that it will take 35 minutes from then, or a total of 40 minutes. Fifteen minutes later you have completed the highway portion of your journey in good time. As you exit onto a secondary road you cut your estimate of total travel time to 35 minutes.

Driving home example

Each day as you drive home from work, you try to predict how long it will take to get home. When you leave your office, you note the time, the day of the week, the weather, and anything else that might be relevant. Say on this Friday you are leaving at exactly 6, and you estimate that it will take 30 minutes to get home. As you reach your car it is 6:05, and you notice it is starting to rain. The traffic is often slower in the rain, so you reestimate that it will take 35 minutes from then, or a total of 40 minutes. Fifteen minutes later you have completed the highway portion of your journey in good time. As you exit onto a secondary road you cut your estimate of total travel time to 35 minutes. Unfortunately, at this point you get stuck behind a slow truck, and the road is too narrow to pass.

Driving home example

Each day as you drive home from work, you try to predict how long it will take to get home. When you leave your office, you note the time, the day of the week, the weather, and anything else that might be relevant. Say on this Friday you are leaving at exactly 6, and you estimate that it will take 30 minutes to get home. As you reach your car it is 6:05, and you notice it is starting to rain. The traffic is often slower in the rain, so you reestimate that it will take 35 minutes from then, or a total of 40 minutes. Fifteen minutes later you have completed the highway portion of your journey in good time. As you exit onto a secondary road you cut your estimate of total travel time to 35 minutes. Unfortunately, at this point you get stuck behind a slow truck, and the road is too narrow to pass. You end up having to follow the truck until you turn onto the side street where you live at 6:40.

Driving home example

Each day as you drive home from work, you try to predict how long it will take to get home. When you leave your office, you note the time, the day of the week, the weather, and anything else that might be relevant. Say on this Friday you are leaving at exactly 6, and you estimate that it will take 30 minutes to get home. As you reach your car it is 6:05, and you notice it is starting to rain. The traffic is often slower in the rain, so you reestimate that it will take 35 minutes from then, or a total of 40 minutes. Fifteen minutes later you have completed the highway portion of your journey in good time. As you exit onto a secondary road you cut your estimate of total travel time to 35 minutes. Unfortunately, at this point you get stuck behind a slow truck, and the road is too narrow to pass. You end up having to follow the truck until you turn onto the side street where you live at 6:40. Three minutes later you are home. The sequence of states, times, and predictions are in the next slide:
(source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018)

Driving home example

State	Elapsed Time (minutes)	Predicted Time to Go	Predicted Total Time
leaving office	0	30	30
reach car, raining	5	35	40
exit highway	20	15	35
behind truck	30	10	40
home street	40	3	43
arrive home	43	0	43

Driving home example

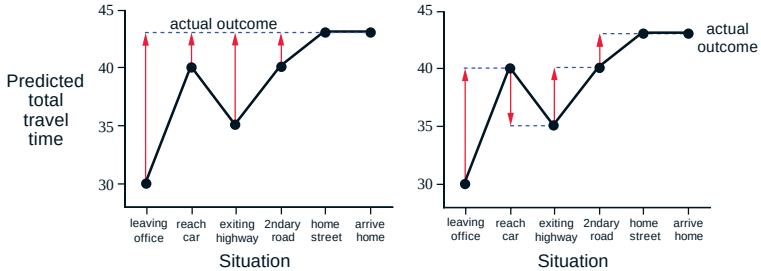


Fig. 5.3: Updates by MC (left) and TD (right) for $\alpha = 1$ (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018)

- TD can learn before knowing the final outcome.
  - TD learns after every step.
  - MC must wait until the episode's end.
- TD could learn without a final outcome.
  - MC is only applicable to episodic tasks.

TD(0) prediction example: forest tree MDP (1)

Let's reuse the forest tree MDP example with *fifty-fifty policy* and discount factor $\gamma = 0.8$ plus disaster probability $\alpha = 0.2$:

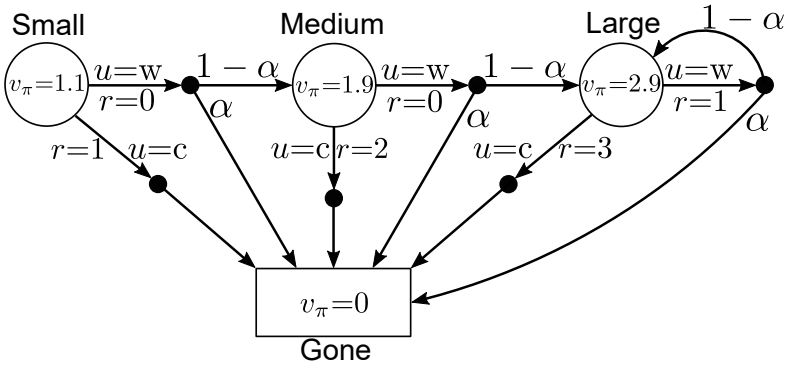


Fig. 5.4: Forest MDP with fifty-fifty-policy including state values

TD(0) prediction example: forest tree MDP (2)

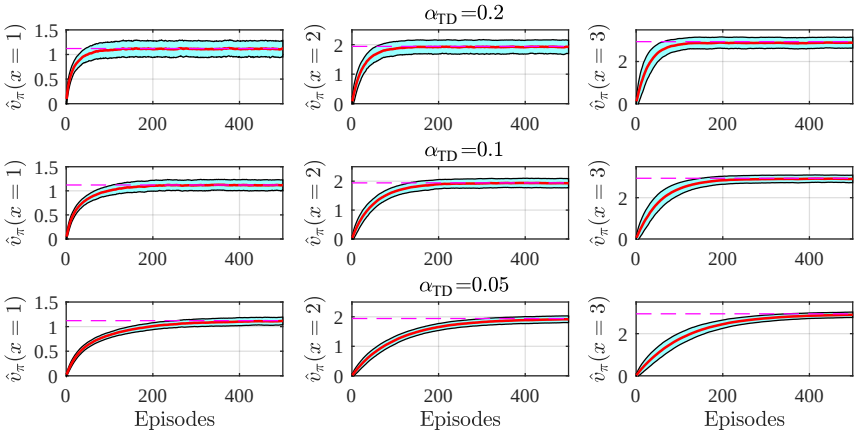


Fig. 5.5: State-value estimate of forest tree MDP using TD(0) prediction over the number of episodes being evaluated (mean and standard deviation are calculated based on 2000 independent runs)

TD(0) vs. MC prediction example: forest tree MDP (1)

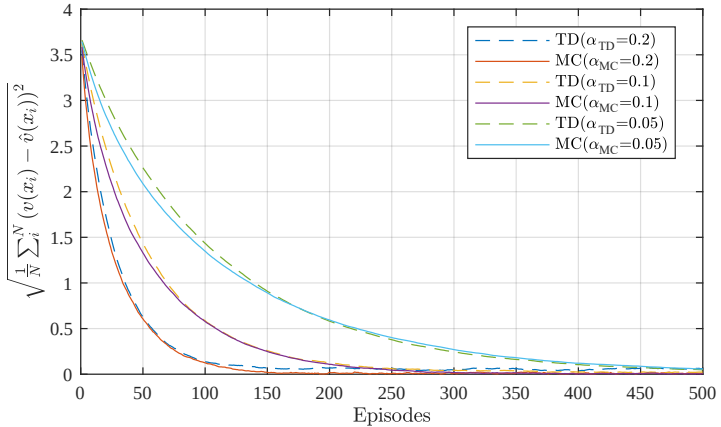


Fig. 5.6: Averaged mean of state-value estimates of forest tree MDP using TD(0) and MC over 1000 independent runs with $\hat{v}_0(s) = 0 \, \forall s \in \mathcal{S}$

TD(0) vs. MC prediction example: forest tree MDP (2)

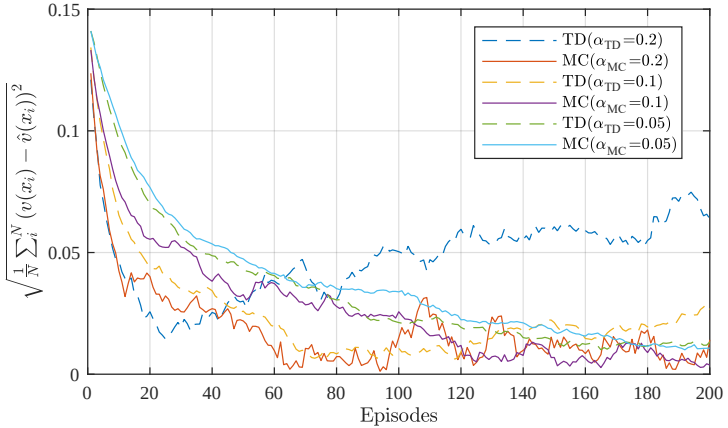


Fig. 5.7: Averaged mean of state-value estimates of forest tree MDP using TD(0) and MC over 1000 independent runs with $\hat{v}_0(s) \approx v(s) \forall s \in \mathcal{S}$

Convergence of TD(0)

Theorem 5.1: Convergence of TD(0)

Given a finite MDP and a fixed policy π the state-value estimate of TD(0) converges to the true v_π

- in the mean for a constant but sufficiently small step-size α and
- with probability 1 if the step-size holds the condition

$$\sum_{t=1}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty. \tag{5.5}$$

Above t is the sample index (i.e., how often the TD update was applied).

Convergence of TD(0)

Theorem 5.1: Convergence of TD(0)

Given a finite MDP and a fixed policy π the state-value estimate of TD(0) converges to the true v_π

- in the mean for a constant but sufficiently small step-size α and
- with probability 1 if the step-size holds the condition

$$\sum_{t=1}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty. \tag{5.5}$$

Above t is the sample index (i.e., how often the TD update was applied).

- In particular, $\alpha_t = \frac{1}{t}$ meets the condition (5.5).

Convergence of TD(0)

Theorem 5.1: Convergence of TD(0)

Given a finite MDP and a fixed policy π the state-value estimate of TD(0) converges to the true v_π

- in the mean for a constant but sufficiently small step-size α and
- with probability 1 if the step-size holds the condition

$$\sum_{t=1}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty. \tag{5.5}$$

Above t is the sample index (i.e., how often the TD update was applied).

- In particular, $\alpha_t = \frac{1}{t}$ meets the condition (5.5).
- Often TD(0) converges faster than MC, but there is no guarantee.

Convergence of TD(0)

Theorem 5.1: Convergence of TD(0)

Given a finite MDP and a fixed policy π the state-value estimate of TD(0) converges to the true v_π

- in the mean for a constant but sufficiently small step-size α and
- with probability 1 if the step-size holds the condition

$$\sum_{t=1}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty. \tag{5.5}$$

Above t is the sample index (i.e., how often the TD update was applied).

- In particular, $\alpha_t = \frac{1}{t}$ meets the condition (5.5).
- Often TD(0) converges faster than MC, but there is no guarantee.
- TD(0) can be more sensitive to bad initializations $\hat{v}_0(s)$ compared to MC.

Batch training

- If experience $\to \infty$ both MC and TD converge $\hat{v}(s) \to v(s)$.
- But how to handle limited experience, i.e., a finite set of episodes

$$
\begin{aligned}
&s_{1,1}, a_{1,1}, R_{2,1}, \ldots, s_{T_1,1},\\
&s_{1,2}, a_{1,2}, R_{2,2}, \ldots, s_{T_2,2},\\
&\vdots\\
&s_{1,j}, a_{1,j}, R_{2,j}, \ldots, s_{T_j,j},\\
&\vdots\\
&s_{1,J}, a_{1,J}, R_{2,J}, \ldots, s_{T_J,J}.
\end{aligned}
$$

Batch training

- If experience $\to \infty$ both MC and TD converge $\hat{v}(s) \to v(s)$.
- But how to handle limited experience, i.e., a finite set of episodes

$$
\begin{aligned}
&s_{1,1}, a_{1,1}, R_{2,1}, \ldots, s_{T_1,1},\\
&s_{1,2}, a_{1,2}, R_{2,2}, \ldots, s_{T_2,2},\\
&\vdots\\
&s_{1,j}, a_{1,j}, R_{2,j}, \ldots, s_{T_j,j},\\
&\vdots\\
&s_{1,J}, a_{1,J}, R_{2,J}, \ldots, s_{T_J,J}.
\end{aligned}
$$

Batch training

- Process all available episodes $j \in [1, J]$ repeatedly to MC and TD.
- If the step size α is sufficiently small both will converge to certain steady-state values.

Batch training: AB-example (1)

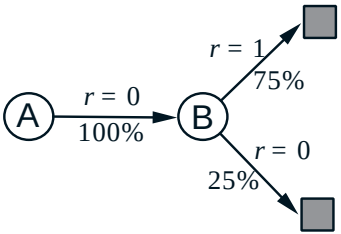


Fig. 5.8: Example environment (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018)

- Only two states: A, B
- No discounting
- 8 episodes of experience available (see Tab. 5.1)
- What is $\hat{v}(A)$ and $\hat{v}(B)$ using batch training TD(0) and MC?

A, 0, B, 0	B,1
B,1	B,1
B,1	B,1
B,1	B,0

Tab. 5.1: Example state-reward sequences for Fig. 5.8

Batch training: AB-example (2)

First, recap MC and TD(0) update rules:

$$\text{MC}: \quad \hat{v}(s_t) \leftarrow \hat{v}(s_t) + \alpha \left[G_t - \hat{v}(s_t) \right],$$
$$\text{TD}: \quad \hat{v}(s_t) \leftarrow \hat{v}(s_t) + \alpha \left[R_{t+1} + \gamma \hat{v}(s_{t+1}) - \hat{v}(s_t) \right].$$

Batch training: AB-example (2)

First, recap MC and TD(0) update rules:

$$\begin{aligned}
\text{MC}: &\quad \hat{v}(s_t) \leftarrow \hat{v}(s_t) + \alpha \left[G_t - \hat{v}(s_t) \right], \\
\text{TD}: &\quad \hat{v}(s_t) \leftarrow \hat{v}(s_t) + \alpha \left[R_{t+1} + \gamma \hat{v}(s_{t+1}) - \hat{v}(s_t) \right].
\end{aligned}$$

Then, in steady state one receives:

$$\begin{aligned}
\text{MC}: &\quad 0 = \alpha \left[G_t - \hat{v}(s_t) \right] = G_t - \hat{v}(s_t), \\
\text{TD}: &\quad 0 = \alpha \left[R_{t+1} + \gamma \hat{v}(s_{t+1}) - \hat{v}(s_t) \right] = R_{t+1} + \gamma \hat{v}(s_{t+1}) - \hat{v}(s_t).
\end{aligned}$$

Batch training: AB-example (2)

First, recap MC and TD(0) update rules:

$$\begin{aligned}\text{MC}: &\quad \hat{v}(s_t) \leftarrow \hat{v}(s_t) + \alpha\left[G_t - \hat{v}(s_t)\right],\\ \text{TD}: &\quad \hat{v}(s_t) \leftarrow \hat{v}(s_t) + \alpha\left[R_{t+1} + \gamma\hat{v}(s_{t+1}) - \hat{v}(s_t)\right].\end{aligned}$$

Then, in steady state one receives:

$$\begin{aligned}\text{MC}: &\quad 0 = \alpha\left[G_t - \hat{v}(s_t)\right] = G_t - \hat{v}(s_t),\\ \text{TD}: &\quad 0 = \alpha\left[R_{t+1} + \gamma\hat{v}(s_{t+1}) - \hat{v}(s_t)\right] = R_{t+1} + \gamma\hat{v}(s_{t+1}) - \hat{v}(s_t).\end{aligned}$$

Considering a batch learning sweep over $j = 1, \ldots, J$ episodes:

$$\begin{aligned}\text{MC}: &\quad 0 = \sum_{j=1}^{J} G_{t,j} - \hat{v}(s_{t,j}),\\ \text{TD}: &\quad 0 = \sum_{j=1}^{J} R_{t+1,j} + \gamma\hat{v}(s_{t+1,j}) - \hat{v}(s_{t,j}).\end{aligned}$$

Batch training: AB-example (3)

Apply the previous equations first to state B. Since B is a terminal state, $\hat{v}(s_{t+1}) = 0$ and $G_{t,j} = R_{t+1,j}$ apply, i.e., the MC and TD updates are identical for B:

$$\text{MC}|_{s=\text{B}}: \quad 0 = \sum_{j=1}^{J} G_{t,j} - \hat{v}(s_{t,j}) \quad \Leftrightarrow \quad \hat{v}(\text{B}) = \frac{1}{J} \sum_{j=1}^{J} G_{t,j},$$

$$\text{TD}|_{s=\text{B}}: \quad 0 = \sum_{j=1}^{J} R_{t+1,j} - \hat{v}(s_{t,j}) \quad \Leftrightarrow \quad \hat{v}(\text{B}) = \frac{1}{J} \sum_{j=1}^{J} G_{t,j}.$$

Batch training: AB-example (3)

Apply the previous equations first to state B. Since B is a terminal state, $\hat{v}(s_{t+1}) = 0$ and $G_{t,j} = R_{t+1,j}$ apply, i.e., the MC and TD updates are identical for B:

$$\text{MC}|_{s=\text{B}}: \quad 0 = \sum_{j=1}^{J} G_{t,j} - \hat{v}(s_{t,j}) \quad \Leftrightarrow \quad \hat{v}(\text{B}) = \frac{1}{J} \sum_{j=1}^{J} G_{t,j},$$

$$\text{TD}|_{s=\text{B}}: \quad 0 = \sum_{j=1}^{J} R_{t+1,j} - \hat{v}(s_{t,j}) \quad \Leftrightarrow \quad \hat{v}(\text{B}) = \frac{1}{J} \sum_{j=1}^{J} G_{t,j}.$$

This is the average return of the available episodes from Tab. 5.1 , i.e., 6×1 and 2×0:

$$\hat{v}(\text{B})|_{\text{MC}} = \hat{v}(\text{B})|_{\text{TD}} = \frac{6}{8} = 0.75\,. \tag{5.6}$$

Batch training: AB-example (4)

Now consider state A assuming the steady state of batch learning process:

- The instantaneous reward is always $R = 0$.
- The TD bootstrap estimate of B is $\hat{v}(s_{t+1,j}) = \hat{v}(B) = \frac{3}{4}$.

Batch training: AB-example (4)

Now consider state A assuming the steady state of batch learning process:

- The instantaneous reward is always $R = 0$.
- The TD bootstrap estimate of B is $\hat{v}(s_{t+1,j}) = \hat{v}(B) = \frac{3}{4}$.

$$\text{MC}: \quad 0 = \sum_{j=1}^{J} G_{t,j} - \hat{v}(s_{t,j}) = \sum_{j=1}^{J} G_{t,j} - \hat{v}(A),$$

$$\text{TD}: \quad 0 = \sum_{j=1}^{J} R_{t+1,j} + \gamma \hat{v}(s_{t+1,j}) - \hat{v}(s_{t,j}) = \sum_{j=1}^{J} \gamma \hat{v}(B) - \hat{v}(A).$$

Batch training: AB-example (4)

Now consider state A assuming the steady state of batch learning process:

- The instantaneous reward is always $R = 0$.
- The TD bootstrap estimate of B is $\hat{v}(s_{t+1,j}) = \hat{v}(B) = \frac{3}{4}$.

$$\text{MC}: \quad 0 = \sum_{j=1}^{J} G_{t,j} - \hat{v}(s_{t,j}) = \sum_{j=1}^{J} G_{t,j} - \hat{v}(A),$$

$$\text{TD}: \quad 0 = \sum_{j=1}^{J} R_{t+1,j} + \gamma \hat{v}(s_{t+1,j}) - \hat{v}(s_{t,j}) = \sum_{j=1}^{J} \gamma \hat{v}(B) - \hat{v}(A).$$

Looking at Tab. 5.1 there is only one episode visiting state A, where the sample return is $G_{t,j} = 0$. Hence, it follows:

$$\hat{v}(\text{A})|_{\text{MC}} = 0, \quad \hat{v}(\text{A})|_{\text{TD}} = \gamma \hat{v}(B) = \frac{3}{4}.$$

Where does this mismatch between the MC and TD estimates come from?

Certainty equivalence

- MC batch learning converges to the least squares fit of the sampled returns:

$$\sum_{j=1}^{J} \sum_{t=1}^{T_j} \left(G_{t,j} - \hat{v}(s_{t,j})\right)^2 . \tag{5.7}$$

Certainty equivalence

- MC batch learning converges to the least squares fit of the sampled returns:

$$\sum_{j=1}^{J}\sum_{t=1}^{T_j}\left(G_{t,j}-\hat{v}(s_{t,j})\right)^2. \tag{5.7}$$

- TD batch learning converges to the maximum likelihood estimate such that $\left\langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}, \hat{\mathcal{R}}, \gamma \right\rangle$ explains the data with highest probability:

$$\begin{aligned}\hat{p}^a_{ss'} &= \frac{1}{n(s,a)}\sum_{j=1}^{J}\sum_{t=1}^{T_j}1(S_{t+1}=s'|S_t=s,A_t=a),\\ \hat{\mathcal{R}}^a_s &= \frac{1}{n(s,a)}\sum_{j=1}^{J}\sum_{t=1}^{T_j}1(S_t=s|A_t=a)R_{t+1,j}.\end{aligned} \tag{5.8}$$

- Here, TD assumes a MDP problem structure and is absolutely certain that its internal model concept describes the real world perfectly (so-called certainty equivalence).

Table of contents

Applying Generalized Policy Iteration (GPI) to TD control

The GPI concept is directly applied to the TD framework using action values:

$$\pi_0 \to \hat{q}_{\pi_0} \to \pi_1 \to \hat{q}_{\pi_1} \to \cdots \pi^* \to \hat{q}_{\pi^*} \,. \tag{5.9}$$

Applying Generalized Policy Iteration (GPI) to TD control

The GPI concept is directly applied to the TD framework using action values:

$$\pi_0 \to \hat{q}_{\pi_0} \to \pi_1 \to \hat{q}_{\pi_1} \to \cdots \pi^* \to \hat{q}_{\pi^*} \,. \tag{5.9}$$

One-step TD / TD(0) action-value update (SARSA)

The TD(0) action-value update is:

$$\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + \alpha \left[R_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}) - \hat{q}(s_t, a_t) \right] . \tag{5.10}$$

SARSA: State, Action, Reward, (next) State, (next) Action evaluation

Applying Generalized Policy Iteration (GPI) to TD control

The GPI concept is directly applied to the TD framework using action values:

$$\pi_0 \to \hat{q}_{\pi_0} \to \pi_1 \to \hat{q}_{\pi_1} \to \cdots \pi^* \to \hat{q}_{\pi^*} \,. \tag{5.9}$$

One-step TD / TD(0) action-value update (SARSA)

The TD(0) action-value update is:

$$\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + \alpha \left[R_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}) - \hat{q}(s_t, a_t) \right] . \tag{5.10}$$

SARSA: State, Action, Reward, (next) State, (next) Action evaluation

- In contrast to MC: continuous online updates of policy evaluation and improvement.

Applying Generalized Policy Iteration (GPI) to TD control

The GPI concept is directly applied to the TD framework using action values:

$$\pi_0 \to \hat{q}_{\pi_0} \to \pi_1 \to \hat{q}_{\pi_1} \to \cdots \pi^* \to \hat{q}_{\pi^*} \,. \tag{5.9}$$

One-step TD / TD(0) action-value update (SARSA)

The TD(0) action-value update is:

$$\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + \alpha \left[R_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}) - \hat{q}(s_t, a_t) \right] . \tag{5.10}$$

SARSA: State, Action, Reward, (next) State, (next) Action evaluation

- In contrast to MC: continuous online updates of policy evaluation and improvement.
- On-policy approach requires exploration, e.g., by an ε-greedy policy:

$$\pi_i(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}|, & a = \tilde{a}, \\ \varepsilon/|\mathcal{A}|, & a \neq \tilde{a}. \end{cases} \tag{5.11}$$

TD-based on-policy control (SARSA)

parameter: $\varepsilon \in \{\mathbb{R}|0 < \varepsilon << 1\}$, $\alpha \in \{\mathbb{R}|0 < \alpha < 1\}$
init: $\hat{q}(s,a)$ arbitrarily (except terminal states) $\forall\ \{s \in \mathcal{S}, a \in \mathcal{A}\}$
for $j = 1, 2, \ldots$ *episodes* **do**
    Initialize s_0;
    Choose a_0 from s_0 using a soft policy (e.g., ε-greedy) derived from $\hat{q}(s,a)$;
    $t \leftarrow 0$;
    repeat
        Take action a_t, observe R_{t+1} and s_{t+1};
        Choose a_{t+1} from s_{t+1} using a soft policy derived from $\hat{q}(s,a)$;
        $\hat{q}(s_t,a_t) \leftarrow \hat{q}(s_t,a_t) + \alpha\left[R_{t+1} + \gamma\hat{q}(s_{t+1},a_{t+1}) - \hat{q}(s_t,a_t)\right]$;
        $t \leftarrow t + 1$;
    until s_t *is terminal*;

Algo. 5.2: TD-based on-policy control (SARSA)

Convergence properties are comparable to MC-based on-policy control:

- Policy improvement theorem holds.
- Greedy in the limit with infinite exploration (GLIE) and step-size requirements in Theo. 5.1 apply.

SARSA example: forest tree MDP (1)

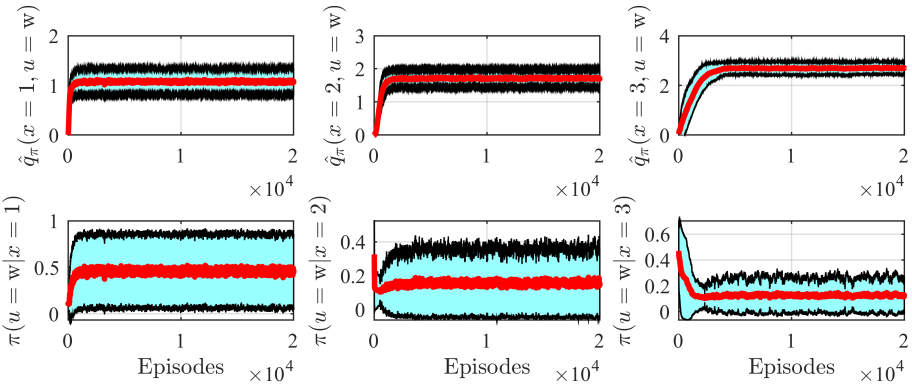


Fig. 5.9: SARSA-based control with $\alpha_{\mathrm{SARSA}} = 0.2$ and ε-greedy policy with $\varepsilon = 0.2$ of forest tree MDP over the number of episodes being evaluated (mean and standard deviation are calculated based on 2000 independent runs)

SARSA example: forest tree MDP (2)

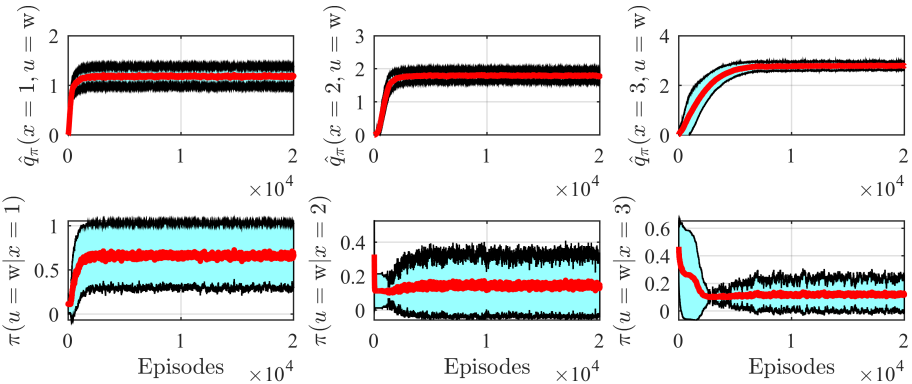


Fig. 5.10: SARSA-based control with $\alpha_{\mathrm{SARSA}} = 0.1$ and ε-greedy policy with $\varepsilon = 0.2$ of forest tree MDP over the number of episodes being evaluated (mean and standard deviation are calculated based on 2000 independent runs)

SARSA example: forest tree MDP (3)

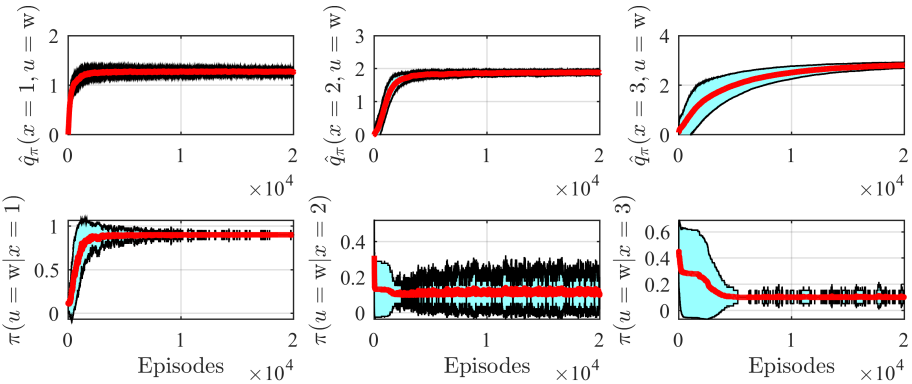


Fig. 5.11: SARSA-based control with adaptive $\alpha_{\mathrm{SARSA}} = \frac{1}{\sqrt{j}}$ (j =episode) and ε-greedy policy with $\varepsilon = 0.2$ of forest tree MDP over the number of episodes being evaluated (mean and standard deviation are calculated based on 2000 independent runs)

Table of contents

Q-learning approach

Similar to SARSA updates, but Q-learning directly estimates q^*:

Q-learning action-value update

The Q-learning action-value update is:

$$\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + \alpha \left[R_{t+1} + \gamma \max_a \hat{q}(s_{t+1}, a) - \hat{q}(s_t, a_t) \right]. \tag{5.12}$$

This is an off-policy update, since the optimal action-value function is updated independent of a given behavior policy.

Q-learning approach

Similar to SARSA updates, but Q-learning directly estimates q^*:

Q-learning action-value update

The Q-learning action-value update is:

$$\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + \alpha \left[R_{t+1} + \gamma \max_a \hat{q}(s_{t+1}, a) - \hat{q}(s_t, a_t) \right]. \tag{5.12}$$

This is an off-policy update, since the optimal action-value function is updated independent of a given behavior policy.

Requirement for Q-learning control:

- Coverage: behavior policy b has nonzero probability of selecting actions that might be taken by the target policy π.
- Consequence: behavior policy b is soft (e.g., ε-soft).
- Step-size requirements (5.5) regarding α apply.

TD-based off-policy control (Q-learning)

parameter: $\varepsilon \in \{\mathbb{R}|0 < \varepsilon << 1\}$, $\alpha \in \{\mathbb{R}|0 < \alpha < 1\}$
init: $\hat{q}(s,a)$ arbitrarily (except terminal states) $\forall\ \{s \in \mathcal{S}, a \in \mathcal{A}\}$
for $j = 1, 2, \ldots$ *episodes* **do**
    Initialize s_0;
    $t \leftarrow 0$;
    repeat
        Choose a_t from s_t using a soft behavior policy;
        Take action a_t, observe R_{t+1} and s_{t+1};
        $\hat{q}(s_t, a_t) \leftarrow \hat{q}(s_t, a_t) + \alpha\left[R_{t+1} + \gamma \max_a \hat{q}(s_{t+1}, a) - \hat{q}(s_t, a_t)\right]$;
        $t \leftarrow t + 1$;
    until s_t *is terminal*;

Algo. 5.3: TD-based off-policy control (Q-learning)

- As discussed with MC-based off-policy control: avoidance of the exploration-optimality trade-off for on-policy methods.
- No importance sampling required as for off-policy MC-based control.

Q-learning control example: cliff walking

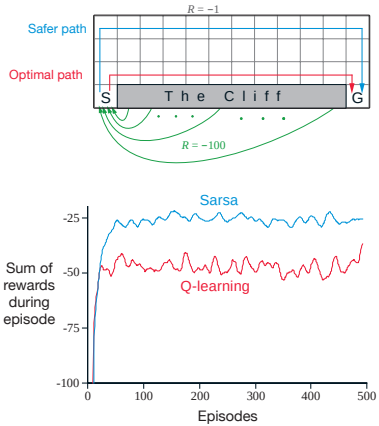


Fig. 5.12: Cliff walking environment (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018)

- $r = -1$ per time step
- Large penalty if you fall off the cliff
- No discounting
- $\varepsilon = 0.1$

Q-learning control example: cliff walking

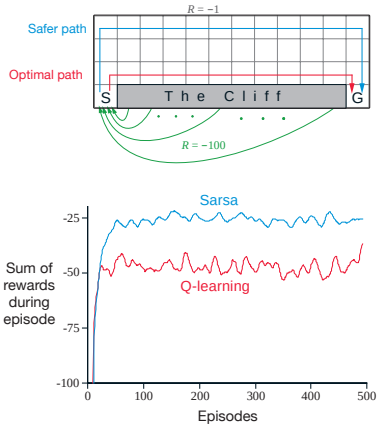


Fig. 5.12: Cliff walking environment (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018)

- $r = -1$ per time step
- Large penalty if you fall off the cliff
- No discounting
- $\varepsilon = 0.1$

- Why is SARSA better in this example?
- And what policy's performance is shown here in particular?

Table of contents

Maximization bias

All control algorithms discussed so far involve maximization operations:

- Q-learning: target policy is greedy and directly uses $\max$ operator for action-value updates.
- SARSA: typically uses an ε-greedy framework, which also involves $\max$ updates during policy improvement.

Maximization bias

All control algorithms discussed so far involve maximization operations:

- Q-learning: target policy is greedy and directly uses $\max$ operator for action-value updates.
- SARSA: typically uses an ε-greedy framework, which also involves $\max$ updates during policy improvement.

This can lead to a significant positive bias:

- Maximization over sampled values is used implicitly as an estimate of the maximum value.
- This issue is called maximization bias.

Maximization bias

All control algorithms discussed so far involve maximization operations:

- Q-learning: target policy is greedy and directly uses $\max$ operator for action-value updates.
- SARSA: typically uses an ε-greedy framework, which also involves $\max$ updates during policy improvement.

This can lead to a significant positive bias:

- Maximization over sampled values is used implicitly as an estimate of the maximum value.
- This issue is called maximization bias.

Small example:

- Consider a single state s with multiple possible actions a.
- The true action values are all $q(s, a) = 0$.

Maximization bias

All control algorithms discussed so far involve maximization operations:

- Q-learning: target policy is greedy and directly uses $\max$ operator for action-value updates.
- SARSA: typically uses an ε-greedy framework, which also involves $\max$ updates during policy improvement.

This can lead to a significant positive bias:

- Maximization over sampled values is used implicitly as an estimate of the maximum value.
- This issue is called maximization bias.

Small example:

- Consider a single state s with multiple possible actions a.
- The true action values are all $q(s, a) = 0$.
- The sampled estimates $\hat{q}(s, a)$ are uncertain, i.e., randomly distributed. Some samples are above and below zero.

Maximization bias

All control algorithms discussed so far involve maximization operations:

- Q-learning: target policy is greedy and directly uses $\max$ operator for action-value updates.
- SARSA: typically uses an ε-greedy framework, which also involves $\max$ updates during policy improvement.

This can lead to a significant positive bias:

- Maximization over sampled values is used implicitly as an estimate of the maximum value.
- This issue is called maximization bias.

Small example:

- Consider a single state s with multiple possible actions a.
- The true action values are all $q(s, a) = 0$.
- The sampled estimates $\hat{q}(s, a)$ are uncertain, i.e., randomly distributed. Some samples are above and below zero.
- Consequence: The maximum of the estimate is positive.

Double learning approach

Split the learning process:

- Divide sampled experience into two sets.
- Use sets to estimate independent estimates $\hat{q}_1(s, a)$ and $\hat{q}_2(s, a)$.

Double learning approach

Split the learning process:

- Divide sampled experience into two sets.
- Use sets to estimate independent estimates $\hat{q}_1(s, a)$ and $\hat{q}_2(s, a)$.

Assign specific tasks to each estimate:

- Estimate the maximizing action:

$$a^* = \arg\max_a \hat{q}_1(s, a). \tag{5.13}$$

Double learning approach

Split the learning process:

- Divide sampled experience into two sets.
- Use sets to estimate independent estimates $\hat{q}_1(s, a)$ and $\hat{q}_2(s, a)$.

Assign specific tasks to each estimate:

- Estimate the maximizing action:

$$a^* = \arg\max_a \hat{q}_1(s, a). \tag{5.13}$$

- Estimate corresponding action value:

$$q(s, a^*) \approx \hat{q}_2(s, a^*) = \hat{q}_2(s, \arg\max_a \hat{q}_1(s, a)). \tag{5.14}$$

Double Q-learning algorithm

parameter: $\varepsilon \in \{\mathbb{R}|0 < \varepsilon << 1\}$, $\quad \alpha \in \{\mathbb{R}|0 < \alpha < 1\}$
init: $\hat{q}_1(s,a)$, $\hat{q}_2(s,a)$ arbitrarily (except terminal states) $\forall\ \{s \in \mathcal{S}, a \in \mathcal{A}\}$
for $j = 1, 2, \ldots$ *episodes* **do**
    Initialize s_0;
    $t \leftarrow 0$;
    repeat
        Choose a_t from s_t using the policy ε-greedy based on $\hat{q}_1(s,a) + \hat{q}_2(s,a)$;
        Take action a_t, observe R_{t+1} and s_{t+1};
        if $n \sim \mathcal{N}(\mu = 0, \sigma) > 0$ **then**
            $\hat{q}_1(s_t, a_t) \leftarrow \hat{q}_1(s_t, a_t) + \alpha\left[R_{t+1} + \gamma \hat{q}_2(s_{t+1}, \arg\max_a \hat{q}_1(s_{t+1}, a)) - \hat{q}_1(s_t, a_t)\right]$;
        else
            $\hat{q}_2(s_t, a_t) \leftarrow \hat{q}_2(s_t, a_t) + \alpha\left[R_{t+1} + \gamma \hat{q}_1(s_{t+1}, \arg\max_a \hat{q}_2(s_{t+1}, a)) - \hat{q}_2(s_t, a_t)\right]$;
        $t \leftarrow t + 1$;
    until s_t *is terminal*;

Algo. 5.4: TD-based off-policy control with double learning

- Doubles memory demand while computational demand per episode remains unchanged
- Less sample efficient than regular Q-learning (samples are split between two estimators)

Maximization bias example

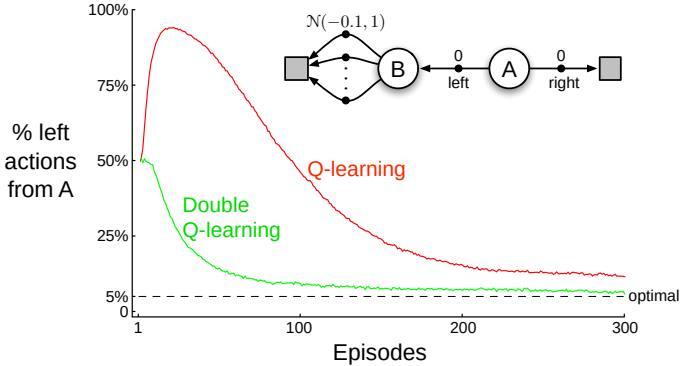


Fig. 5.13: Comparison of Q-learning and double Q-learning on a simple episodic MDP. Q-learning initially learns to take the left action much more often than the right action, and always takes it significantly more often than the 5% minimum probability enforced by ε-greedy action selection with $\varepsilon = 0.1$. In contrast, double Q-learning is essentially unaffected by maximization bias. These data are averaged over 10,000 runs. The initial action-value estimates were zero. (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018)

Summary

- TD unites two key characteristics from DP and MC:
  - From MC: Sample-based updates (i.e., operating in unknown MDPs).
  - From DP: Update estimates based on other estimates (bootstrapping).

Summary

- TD unites two key characteristics from DP and MC:
  - From MC: Sample-based updates (i.e., operating in unknown MDPs).
  - From DP: Update estimates based on other estimates (bootstrapping).
- TD allows certain simplifications and improvements compared to MC:
  - Updates are available after each step and not after each episode.

Summary

- TD unites two key characteristics from DP and MC:
  - From MC: Sample-based updates (i.e., operating in unknown MDPs).
  - From DP: Update estimates based on other estimates (bootstrapping).
- TD allows certain simplifications and improvements compared to MC:
  - Updates are available after each step and not after each episode.
  - Off-policy learning comes without importance sampling.

Summary

- TD unites two key characteristics from DP and MC:
  - From MC: Sample-based updates (i.e., operating in unknown MDPs).
  - From DP: Update estimates based on other estimates (bootstrapping).
- TD allows certain simplifications and improvements compared to MC:
  - Updates are available after each step and not after each episode.
  - Off-policy learning comes without importance sampling.
  - Exploits MDP formalism by maximum likelihood estimates.

Summary

- TD unites two key characteristics from DP and MC:
  - From MC: Sample-based updates (i.e., operating in unknown MDPs).
  - From DP: Update estimates based on other estimates (bootstrapping).
- TD allows certain simplifications and improvements compared to MC:
  - Updates are available after each step and not after each episode.
  - Off-policy learning comes without importance sampling.
  - Exploits MDP formalism by maximum likelihood estimates.
  - Hence, TD prediction and control exhibit a high applicability for many problems.

Summary

- TD unites two key characteristics from DP and MC:
  - From MC: Sample-based updates (i.e., operating in unknown MDPs).
  - From DP: Update estimates based on other estimates (bootstrapping).
- TD allows certain simplifications and improvements compared to MC:
  - Updates are available after each step and not after each episode.
  - Off-policy learning comes without importance sampling.
  - Exploits MDP formalism by maximum likelihood estimates.
  - Hence, TD prediction and control exhibit a high applicability for many problems.
- Batch training can be used when only limited experience is available, i.e., the available samples are re-processed again and again.

Summary

- TD unites two key characteristics from DP and MC:
  - From MC: Sample-based updates (i.e., operating in unknown MDPs).
  - From DP: Update estimates based on other estimates (bootstrapping).
- TD allows certain simplifications and improvements compared to MC:
  - Updates are available after each step and not after each episode.
  - Off-policy learning comes without importance sampling.
  - Exploits MDP formalism by maximum likelihood estimates.
  - Hence, TD prediction and control exhibit a high applicability for many problems.
- Batch training can be used when only limited experience is available, i.e., the available samples are re-processed again and again.
- Greedy policy improvements can lead to maximization biases and, therefore, slow down the learning process.

Summary

- TD unites two key characteristics from DP and MC:
  - From MC: Sample-based updates (i.e., operating in unknown MDPs).
  - From DP: Update estimates based on other estimates (bootstrapping).
- TD allows certain simplifications and improvements compared to MC:
  - Updates are available after each step and not after each episode.
  - Off-policy learning comes without importance sampling.
  - Exploits MDP formalism by maximum likelihood estimates.
  - Hence, TD prediction and control exhibit a high applicability for many problems.
- Batch training can be used when only limited experience is available, i.e., the available samples are re-processed again and again.
- Greedy policy improvements can lead to maximization biases and, therefore, slow down the learning process.
- TD requires careful tuning of learning parameters:
  - Step size α: how to tune convergence rate vs. uncertainty / accuracy?
  - Exploration vs. exploitation: how to visit all state-action pairs?