Chapter 6
Temporal-Difference Learning

- Temporal-difference (TD) learning
  - A combination of Monte Carlo ideas and dynamic programming ideas.
  - Like MC, TD methods can learn directly from raw experience without a model of the environment
  - Like DP, TD methods update values based on other learned values, without waiting for a final outcome
    - TD is bootstrapping(주어진 상황에서 어떻게든 해결한다)

6.1 TD Prediction

- Recall
  - Constant-α MC method (Nonstationary MC method)

$$v_\pi(S_t) \leftarrow v_\pi(S_t) + \alpha\big(G_t - v_\pi(S_t)\big)$$

Prediction: evaluating the value function of a policy

$$\begin{aligned} A_n &= \frac{1}{n}(B_1 + \cdots + B_n) \\ &= \frac{1}{n}\big((n-1)A_{n-1} + B_n\big) \\ &= A_{n-1} + \frac{1}{n}(B_n - A_{n-1}) \end{aligned}$$

$$A \leftarrow A + \alpha(B_n - A)$$

- TD(0), one-step TD
  - The simplest TD method
  - For a policy π, we estimate the optimal value function v_π as
    $$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$
    where $\alpha \in (0,1]$ is the step size

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\mathit{terminal}) = 0$

Loop for each episode:
    Initialize S
    Loop 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

Example 6.1: Driving Home

Example 6.1: Driving Home 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 week, the weather, and anything else that might be relevant. Say on this Friday you are leaving at exactly 6 o'clock, 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. 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 is thus as follows:

State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

Reward: elapsed time
Value: expected time to go
Discounting: $\gamma = 1$

Estimated value: 2nd column

$S_0 =$ leaving office, friday at 6

$S_1 =$ reach car, raining

$$V(S_0) \leftarrow V(S_0) + \alpha\big(R_1 + \gamma V(S_1) - V(S_0)\big)$$

$$V(S_0) \leftarrow 30 + \alpha(5 + \gamma \cdot 35 - 30)$$

$$\gamma = 1,\ \alpha = 1 \Rightarrow V(S_0) = 40$$

$S_2 =$ exiting highway

$$V(S_1) \leftarrow V(S_1) + \alpha\big(R_2 + \gamma V(S_2) - V(S_1)\big)$$

$$V(S_1) \leftarrow 35 + (15 + 15 - 35) = 30$$

MC method

$v_\pi(S_t) \leftarrow v_\pi(S_t) + \alpha\big(G_t - v_\pi(S_t)\big)$

Red arrow $= G_t - v_\pi(S_t)$

TD method

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

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

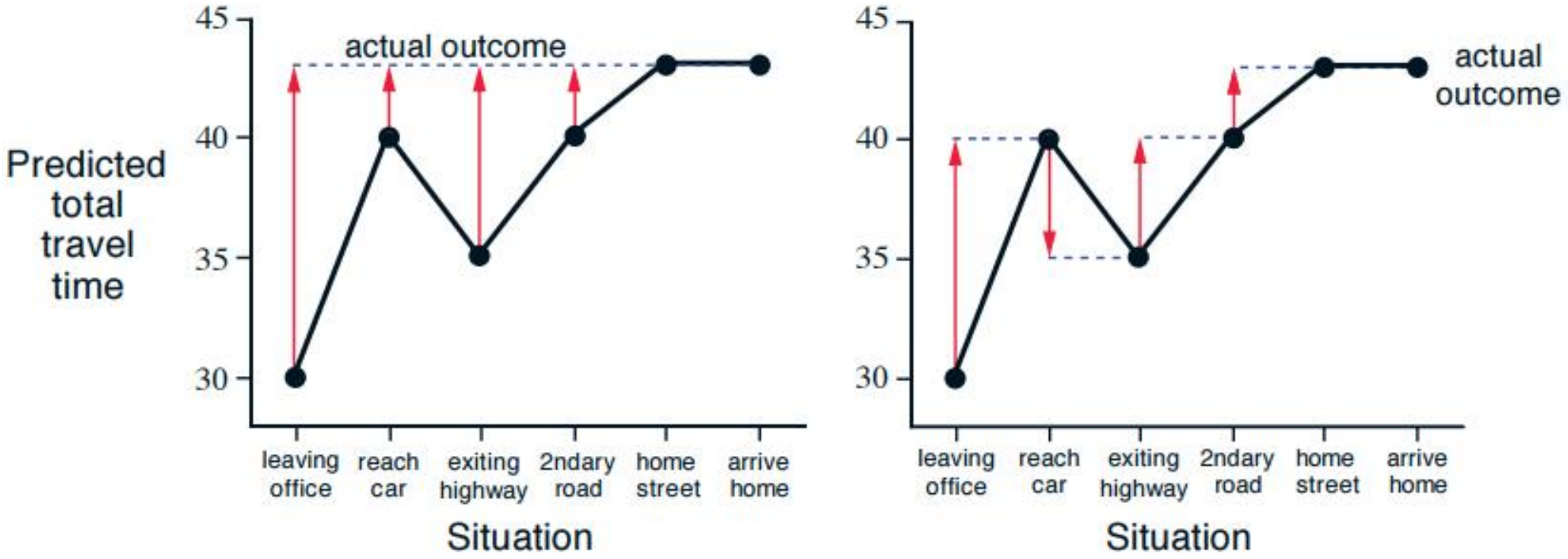


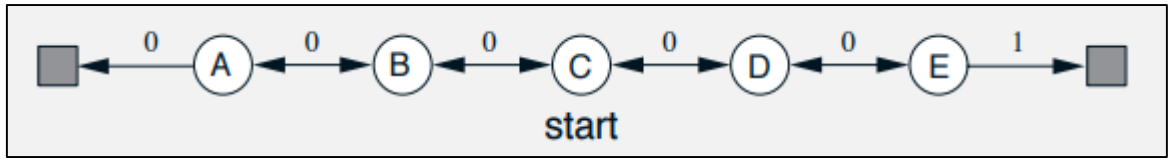
Figure 6.1: Changes recommended in the driving home example by Monte Carlo methods (left) and TD methods (right).

6.2 Advantages of TD Prediction Methods

- TD methods learn a guess from a guess
  - they bootstrap
- Advantages
  - do not require a model of the environment
  - are naturally implemented in an online
    - We wait only one time step
- We can still guarantee convergence to the correct answer

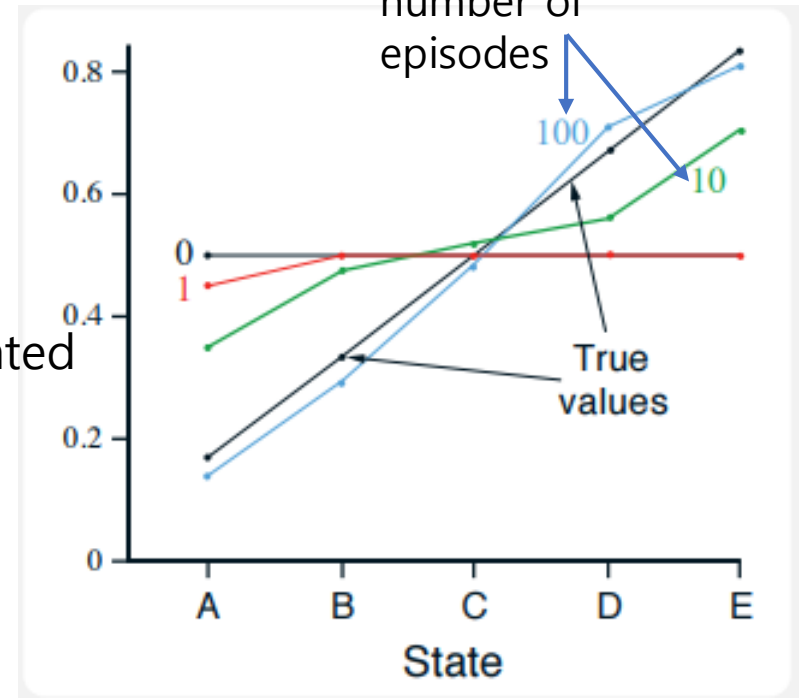
Example 6.2 Random Walk

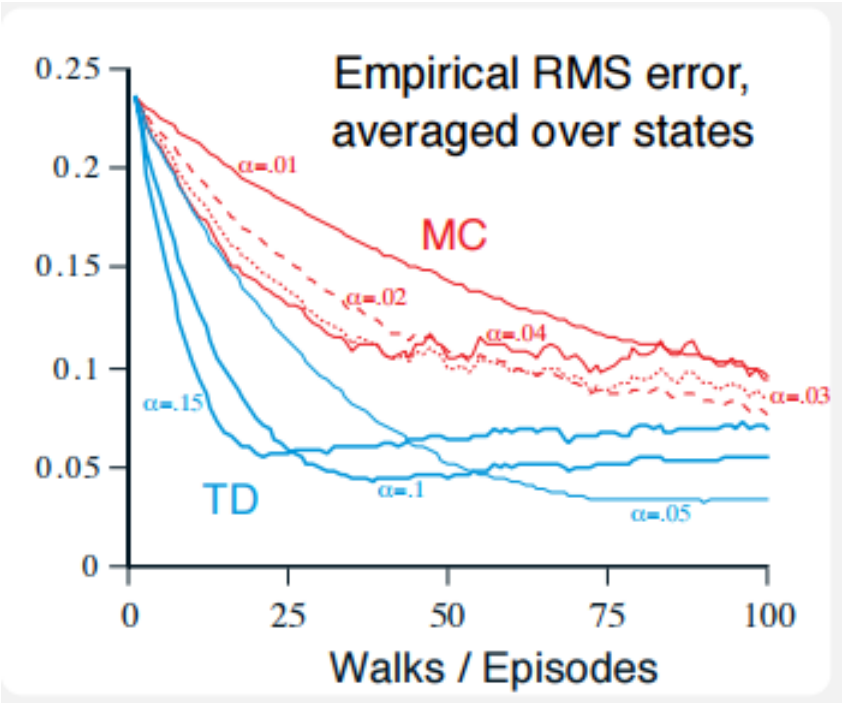
Discounting: $\gamma = 1$
Policy: random



True value
$v_\pi(A) = \frac{1}{6}$, $v_\pi(B) = \frac{2}{6}$,
$v_\pi(C) = \frac{3}{6}$, $v_\pi(D) = \frac{4}{6}$,
$v_\pi(E) = \frac{5}{6}$

TD(0) estimated values





RMS = root mean squared

6.3 Optimality of TD(0)

- Batch updating
  - To use a batch of training data repeatedly
  - To use the same set of experience repeatedly

Example 6.3: Random walk under batch updating

- We use a set of episodes to estimate v_π
- Let V be the estimated value function under batch updating
- By TD(0), V converges to v_π for sufficiently small α
  - V is computed from local information
- By constant-α MC, V converges to a value function($\neq v_\pi$)
  - V is computed from global information

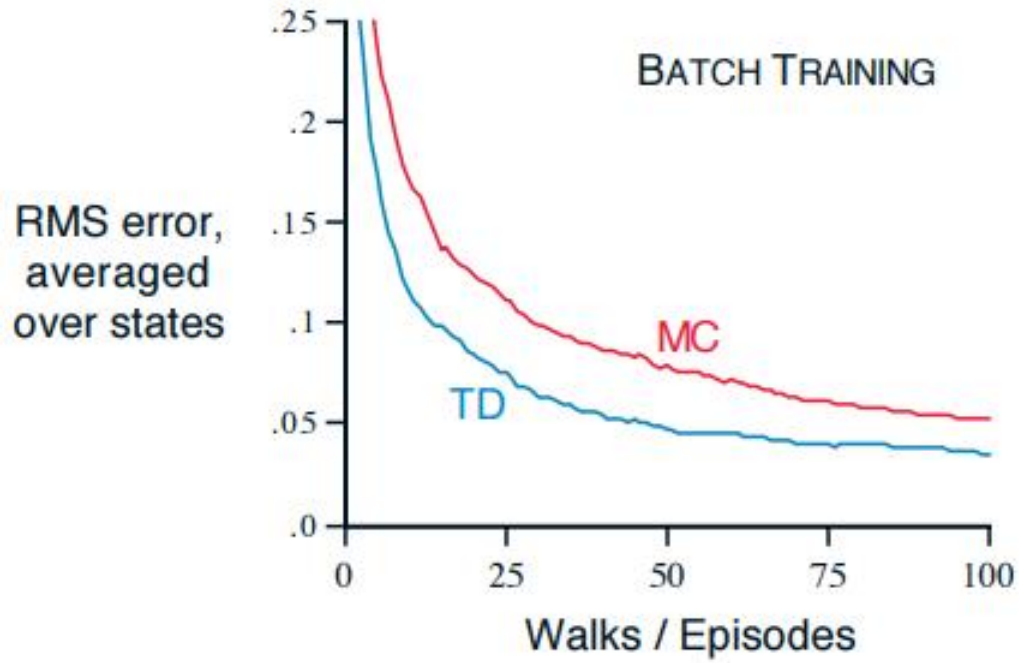
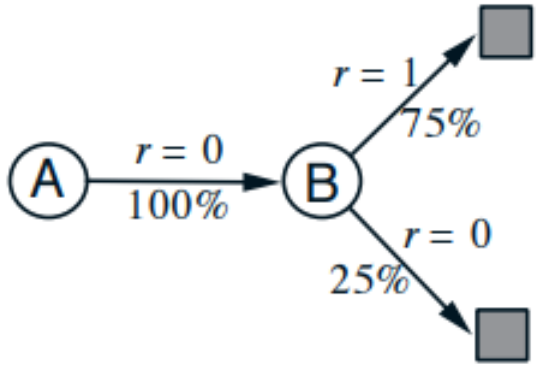


Figure 6.2: Performance of TD(0) and constant-α MC under batch training on the random walk task.

Example 6.4: You are the Predictor



Batch of data

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

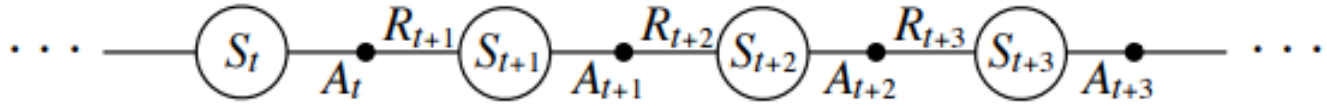
true values

$$v_\pi(A) = v_\pi(B) = \frac{4}{3}$$

TD(0) method
Constant-α MC method

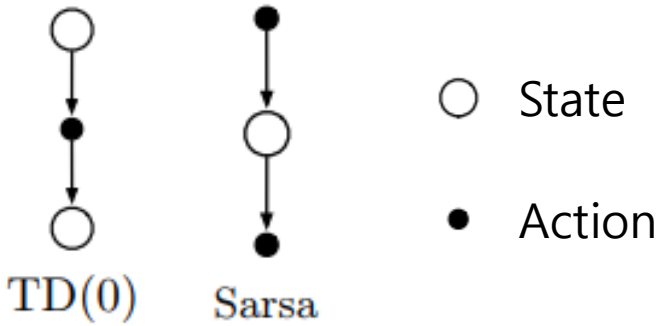
6.4 Sarsa: On-policy TD Control

- Episode



- Sarsa algorithm
  - Estimation of $q_\pi(s,a)$

    $Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1},A_{t+1}) - Q(S_t,A_t)]$
  - The name 'Sarsa' is originated from $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$



- Selection of A_t
  - On-policy
  - ϵ-greedy, i.e.

$$\Pr(A_t) = \begin{cases} 1 - \epsilon + \dfrac{\epsilon}{|\mathcal{A}(S_t)|}, & \text{if } A_t \text{ is maximal} \\ \dfrac{\epsilon}{|\mathcal{A}(S_t)|}, & \text{otherwise} \end{cases}$$

for some $\epsilon > 0$, where $|\mathcal{A}(S_t)|$ is the number of actions at S_t

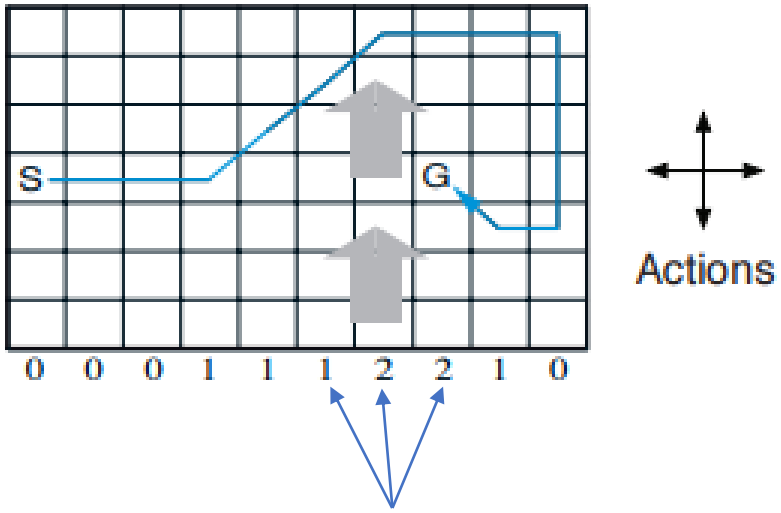
Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

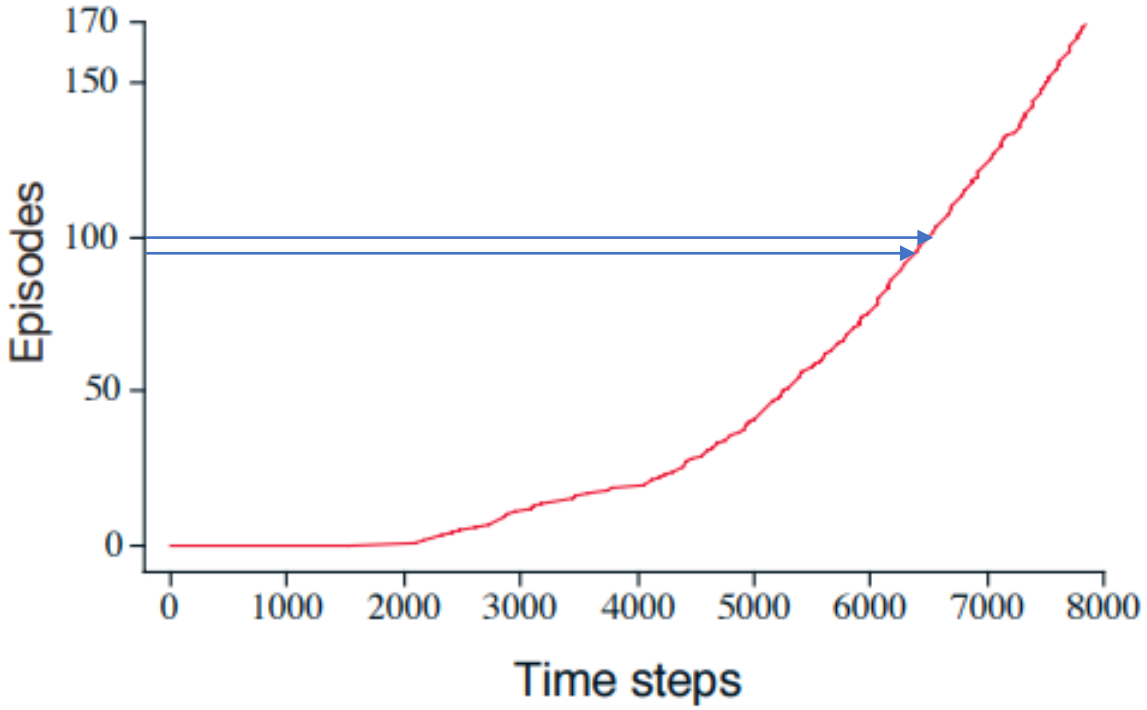
Loop for each episode:
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Loop for each step of episode:
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

Example 6.5: Windy Gridworld

There is a crosswind running upward



ϵ-greedy Sarsa, $\epsilon = 0.1$
$\gamma = 1$, $\alpha = 0.5$, $R_t = -1$

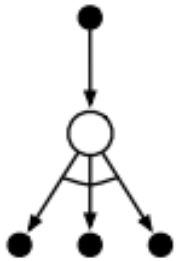


6.5 Q-learning: Off-policy TD Control

- Q-learning algorithm
  - Approximation of q_*, the optimal action-value function

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

  - A_t is chosen by ϵ-greedy
  - No policy is needed

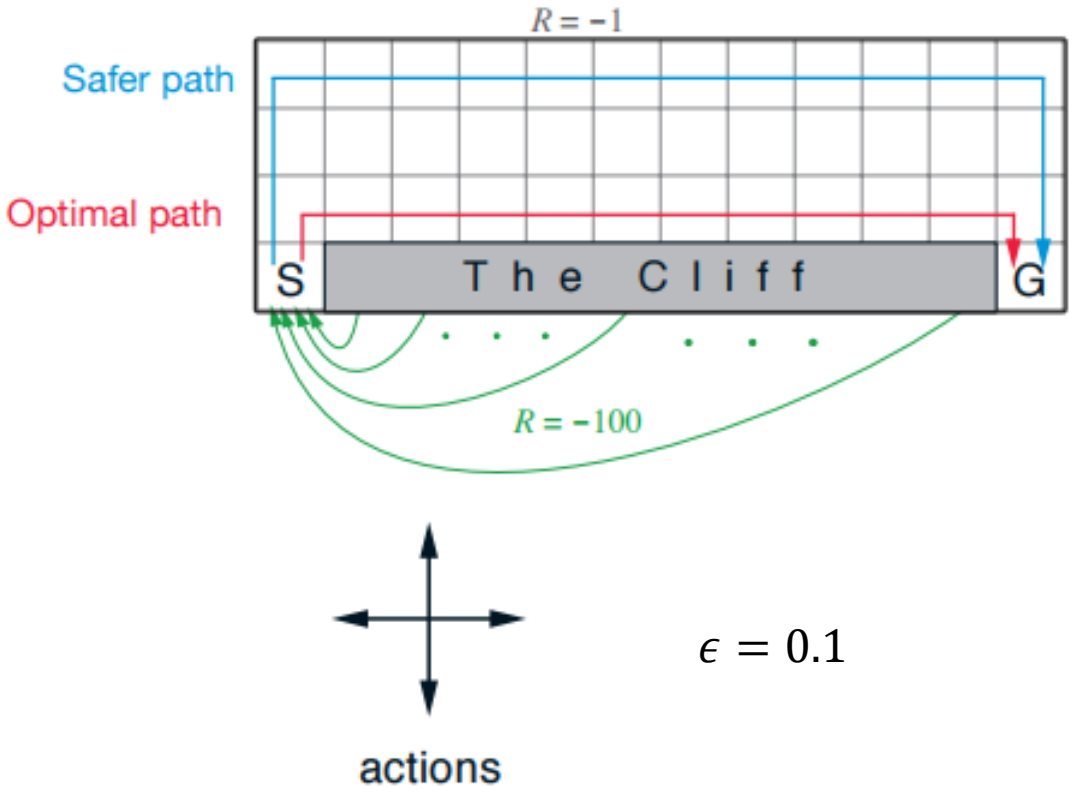


Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

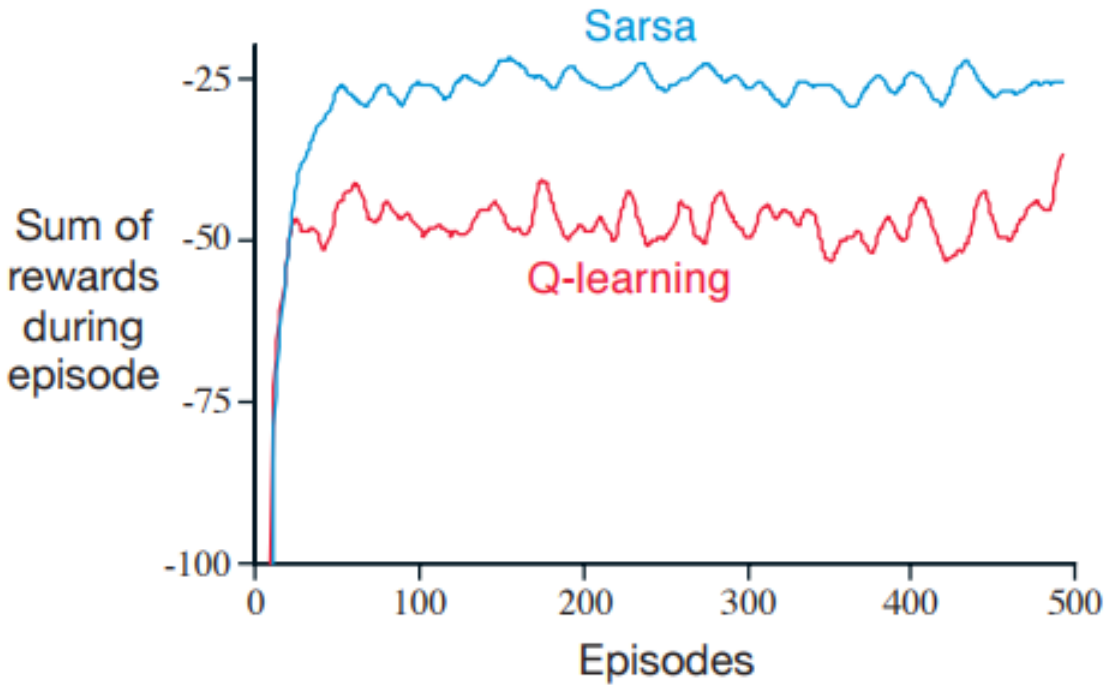
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Example 6.6: Cliff Walking



$\epsilon = 0.1$

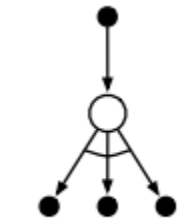


6.6 Expected Sarsa

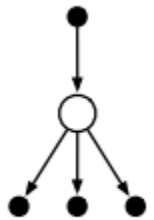
- Algorithm

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, a)|S_{t+1}] - Q(S_t, A_t)]$$

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



Q-learning



Expected Sarsa

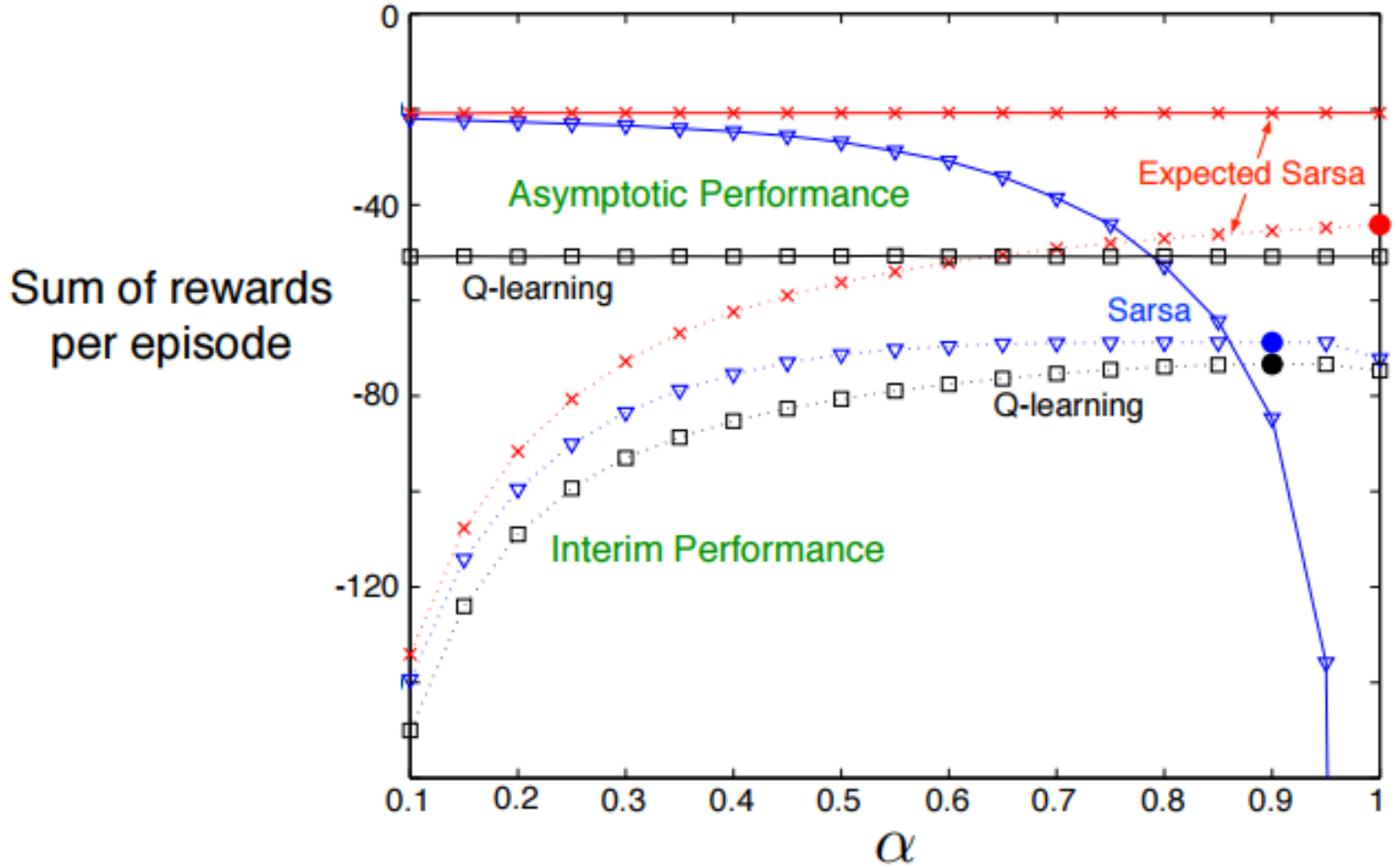
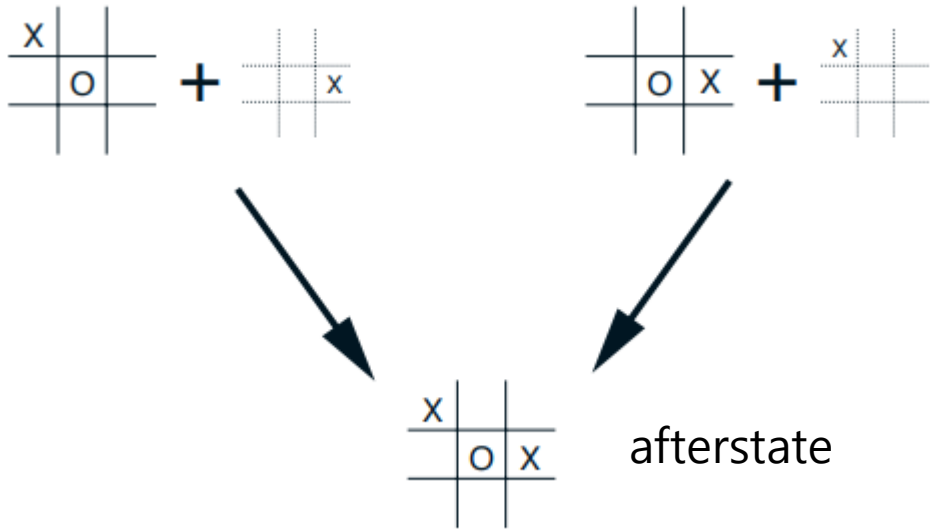


Figure 6.3: Interim and asymptotic performance of TD control methods on the cliff-walking task as a function of α. All algorithms used an ε-greedy policy with $\varepsilon = 0.1$. Asymptotic performance is an average over 100,000 episodes whereas interim performance is an average over the first 100 episodes. These data are averages of over 50,000 and 10 runs for the interim and asymptotic cases respectively. The solid circles mark the best interim performance of each method. Adapted from van Seijen et al. (2009).

6.7 Maximization Bias and Double Learning

6.8 Games, Afterstates, and Other Special Cases

- Afterstate
  - Board position after the agent has made its move
  - afterstate value functions are useful in games like tic-tac-toe



Summary

- TD
  - Policy evaluation - prediction
  - Estimating state values
- Sarsa
  - Update policy - on-policy control
  - Estimating action values by policy
- Q-learning
  - Update policy - off-policy control
  - Estimating action values by maximum
- Expected sarsa
  - Update policy - on-policy control
  - Estimating action values by average