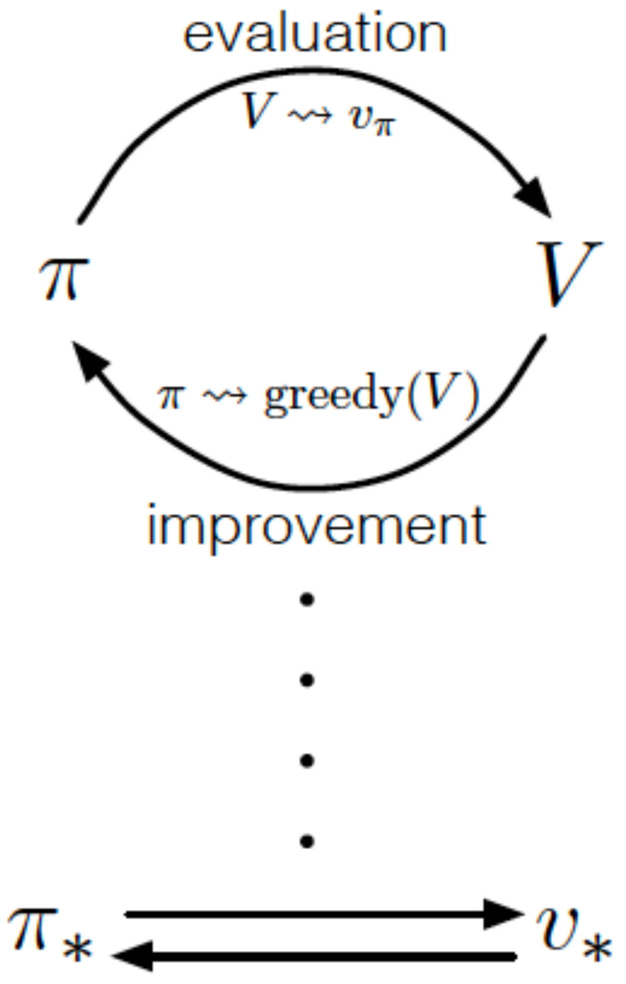
Temporal-Difference Learning

- Combination of learning from experience without a model (like Monte Carlo methods) and bootstrapping (like DP methods)
- Based on GPI and differences between DP, TD and MC are in their approaches to the prediction problem (value function evaluation).



TD prediction

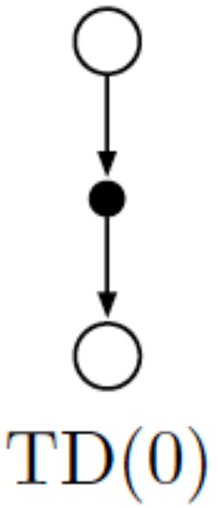
- While MC methods must wait until the end of the episode to calculate return for the value function update…

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

- TD methods need to wait only until the next time step, and the update for TD(0) or one-step TD is

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

- Bootstrapping: updating from an existing estimate



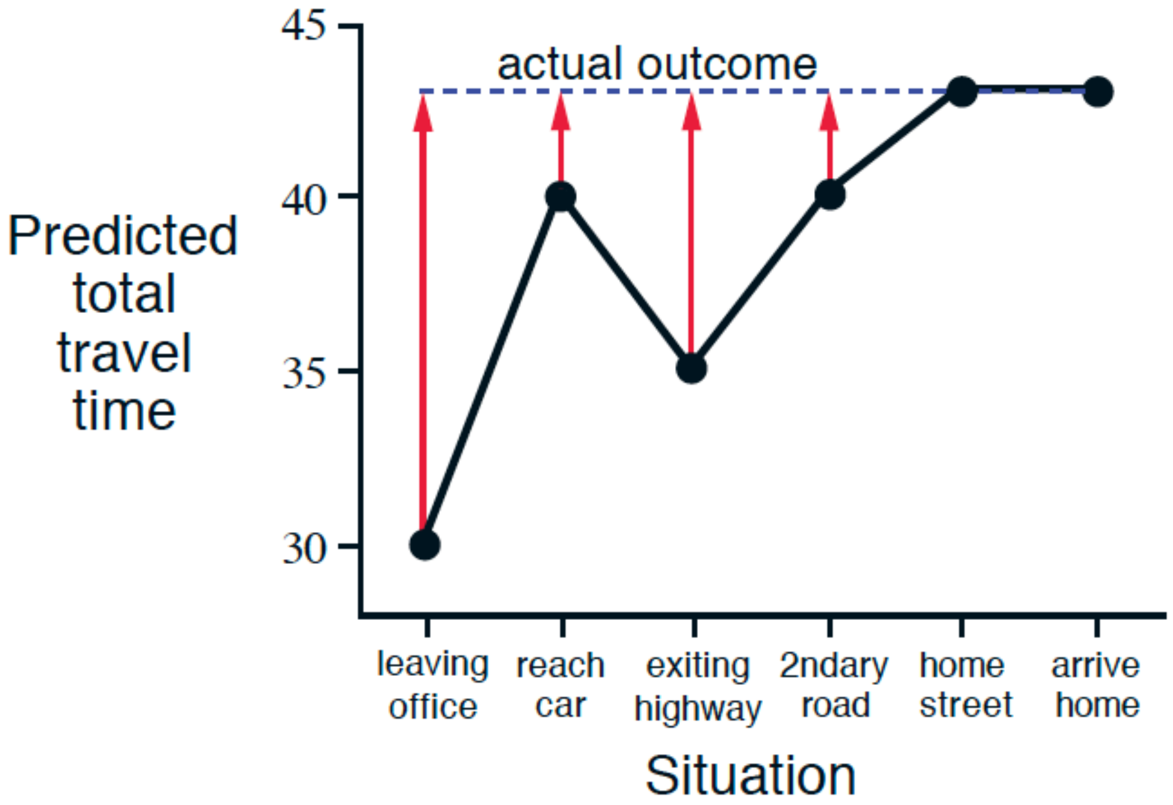
Tabular TD(0) for estimating v_π

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 $V(terminal) = 0$

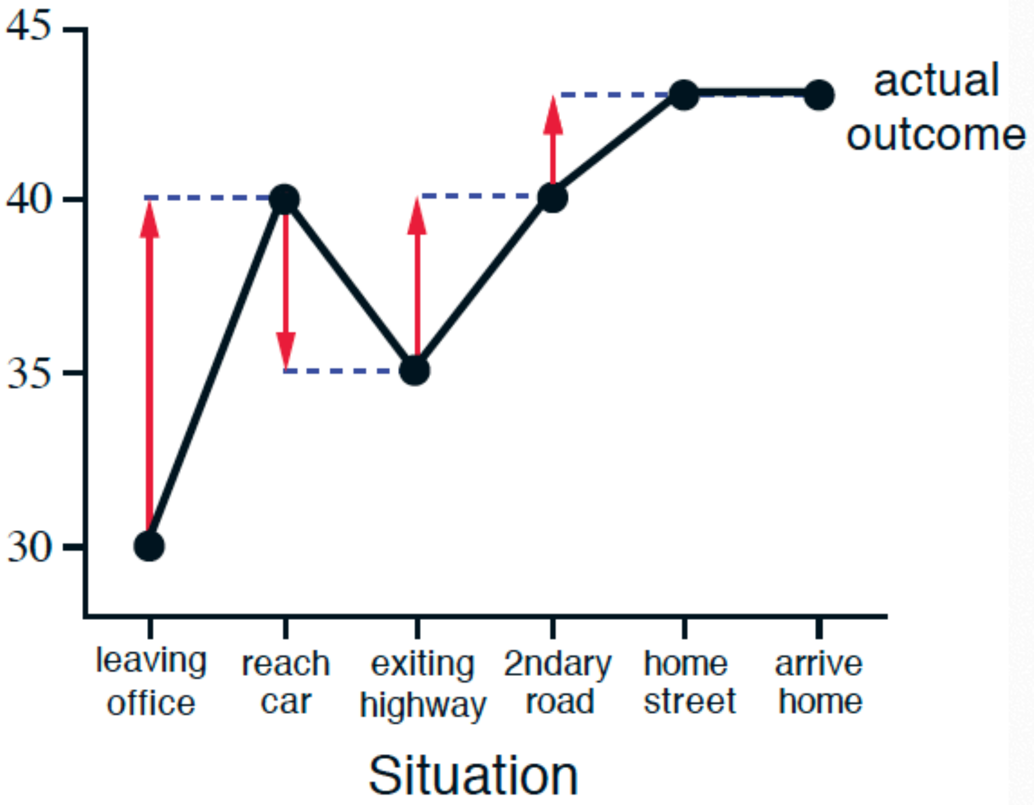
Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

Driving Home example

Monte Carlo



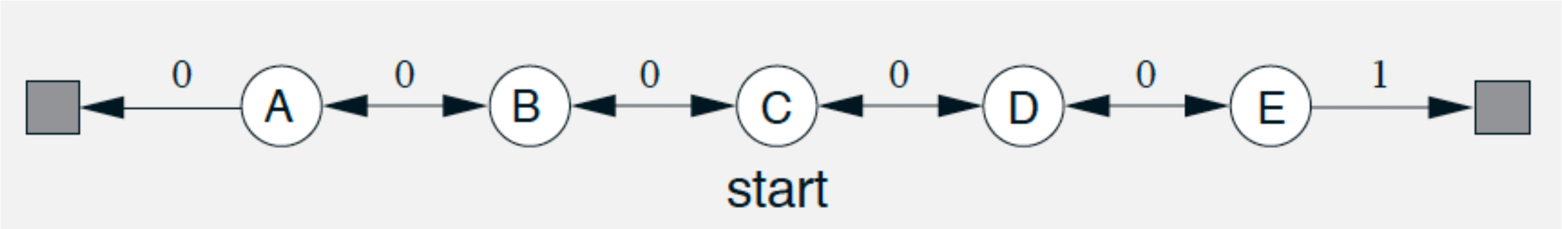
TD methods



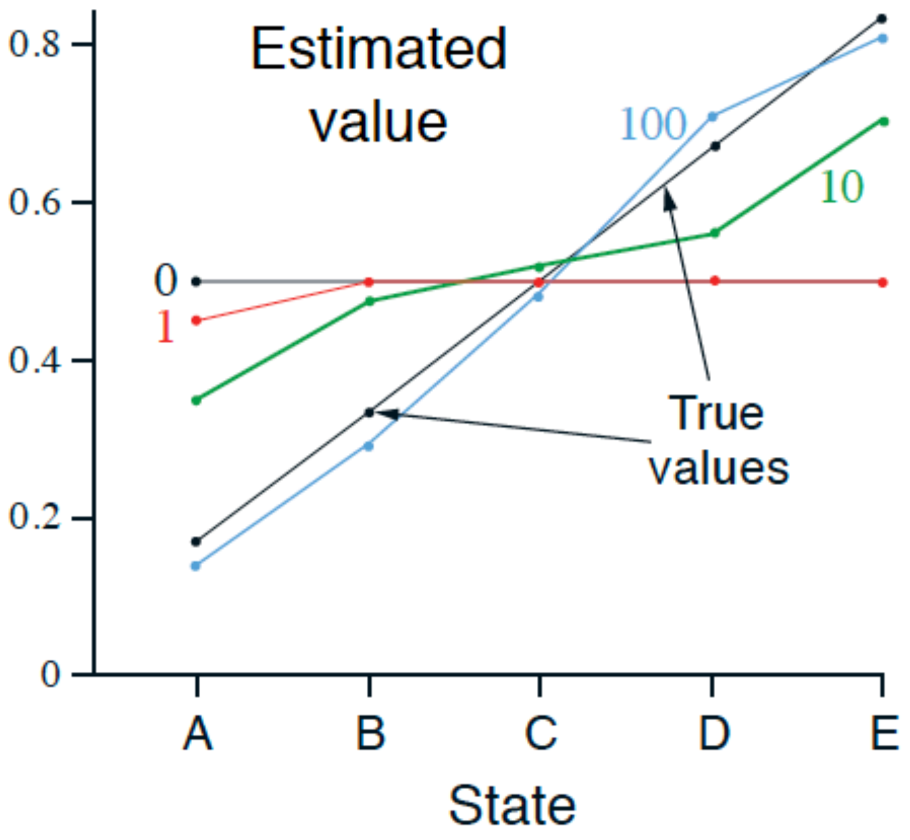
Advantages of TD Prediction Methods

- Implemented in an online, fully incremental fashion so suitable for very long episodes or continuous tasks
- For any fixed policy TD(0) prediction converges to the correct value function (with decreasing step-size)
- In practice TD methods converge faster than MC method (not proved)

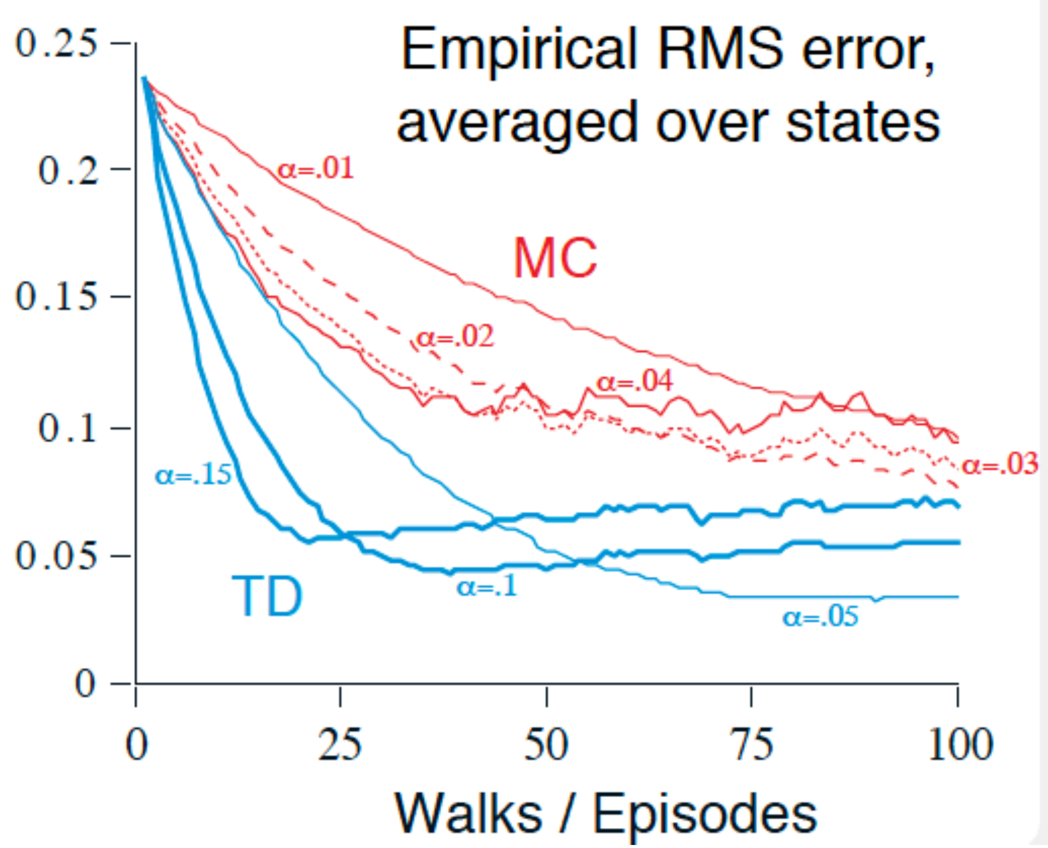
Random walk



0
A
0
B
0
C
0
D
0
E
1
start



Estimated value
0.8
0.6
0.4
0.2
0
100
10
0
1
True values
A
B
C
D
E
State

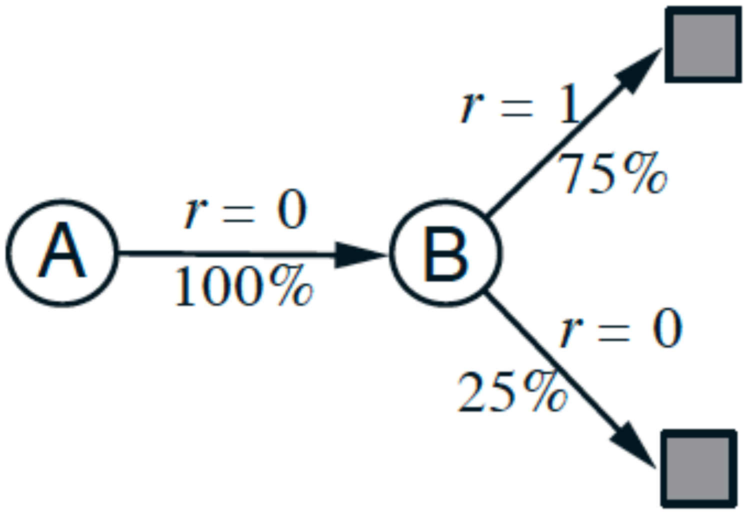


Empirical RMS error, averaged over states
0.25
0.2
0.15
0.1
0.05
0
α=.01
MC
α=.02
α=.04
α=.03
α=.15
TD
α=.1
α=.05
0
25
50
75
100
Walks / Episodes

Optimality of TD(0)

- Batch updating: the value function is updated only once per batch, by the sum of all the increments in the batch
- TD(0) and MC methods both converge deterministically but to different answers
- Example:

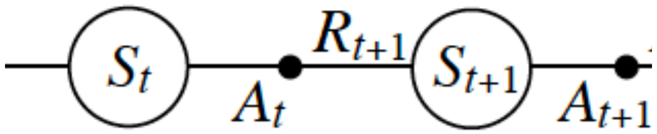
A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0

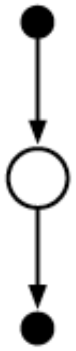


	V(A)	V(B)
MC (minimum MSE)	0	3/4
TD (certainty-equivalence - maximum likelihood estimate)	3/4	3/4

Sarsa: On-policy TD Control

- GPI with TD prediction considering transitions from state–action pairs to state–action pairs (with the corresponding reward)





Sarsa

- Learning an action value function with the following update rule

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

- Policy improvement: ε-greedy or ε-soft

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 $Q(\mathit{terminal}, \cdot) = 0$

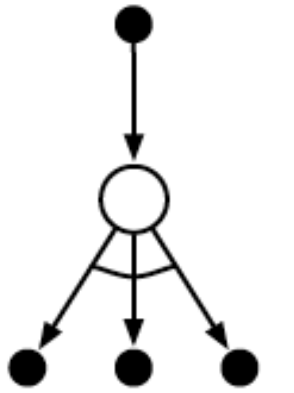
Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') -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

Q-learning: Off-policy TD Control

- Learned action-value function directly approximates the optimal action-value function, independent of the policy being followed.

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

- Convergence if all pairs continue to be updated and step-size decay



Q-learning

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 $Q(\mathit{terminal}, \cdot) = 0$

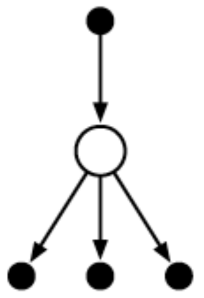
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Expected Sarsa

- Like Q-learning except that instead of the maximum over next state–action pairs it uses the expected value

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

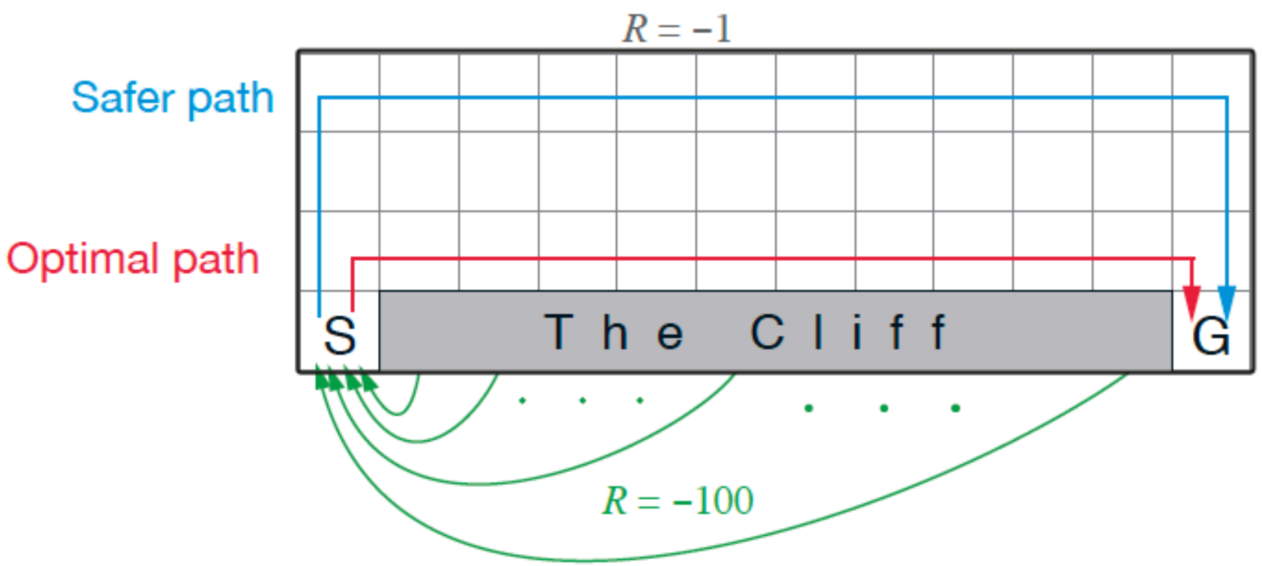
- Eliminates the variance of random action selection A_{t+1} of Sarsa

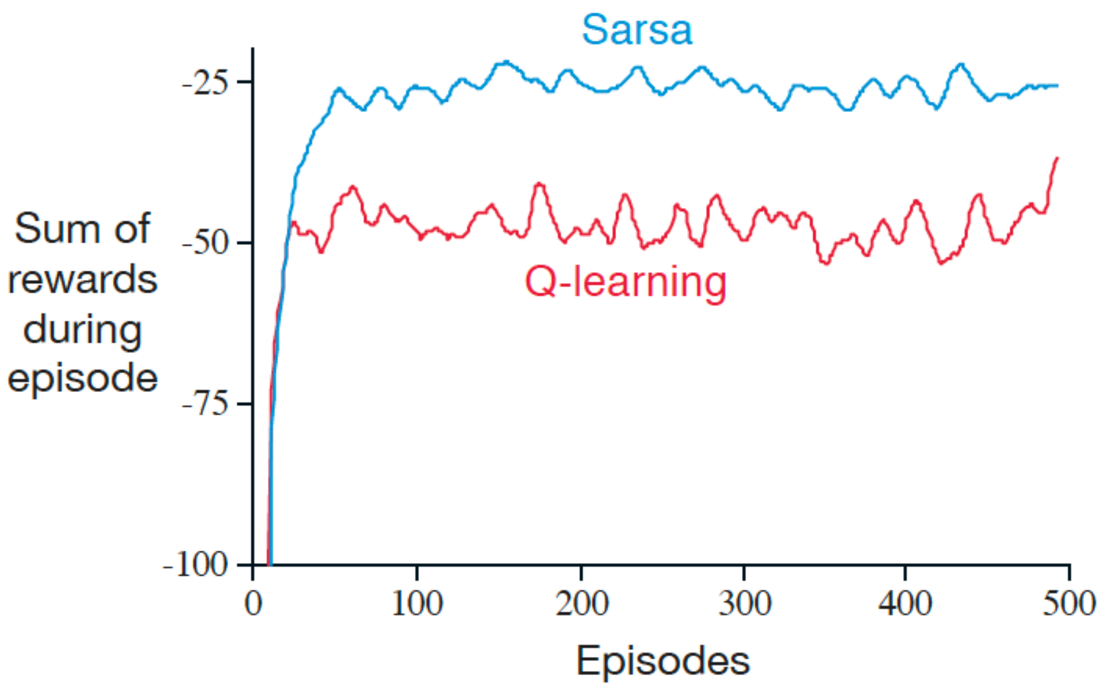


Expected Sarsa

Cliff walking example

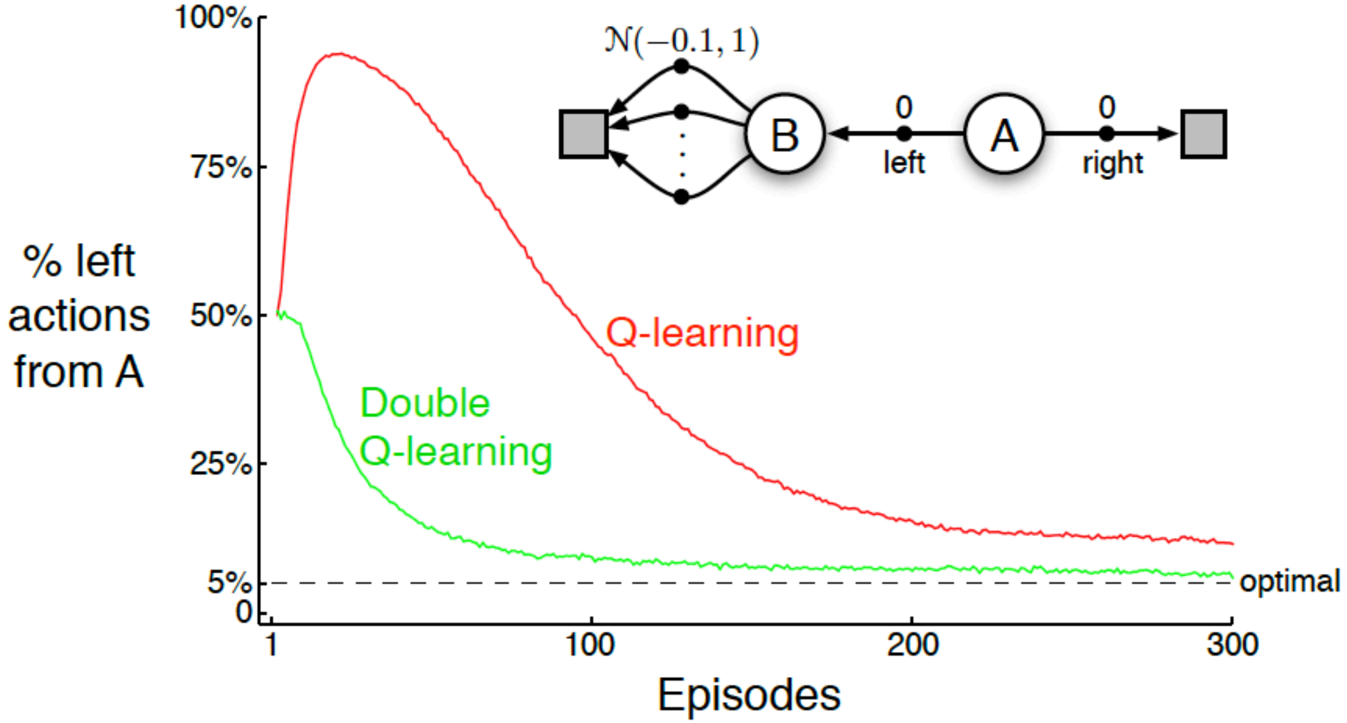
- ε-greedy with $\varepsilon = 0.1$





Maximization Bias and Double Learning

- Maximization (Q-learning target or ε-greedy action selection) leads to positive bias
- Double learning: 2 Q-functions where Q_1 determines the maximizing action and Q_2 estimates its value (target)



$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma Q_2\big(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)\big) - Q_1(S_t, A_t) \Big].$$

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:

$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$$

    else:

$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$$

    $S \leftarrow S'$
  until S is terminal

Summary

Algorithm	Sarsa	Q-learning	Expected sarsa
Target	$R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$	$R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$	$R_{t+1} + \gamma \sum_a \pi(a \| S_{t+1}) Q(S_{t+1}, a)$
Backup diagram