Temporal-Difference Learning

Machine Learning

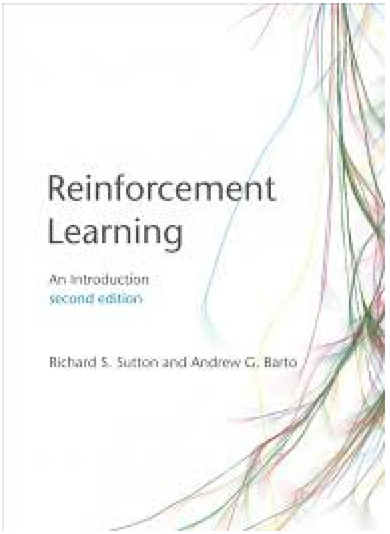
Daniele Loiacono

POLITECNICO
MILANO 1863

Outline and References

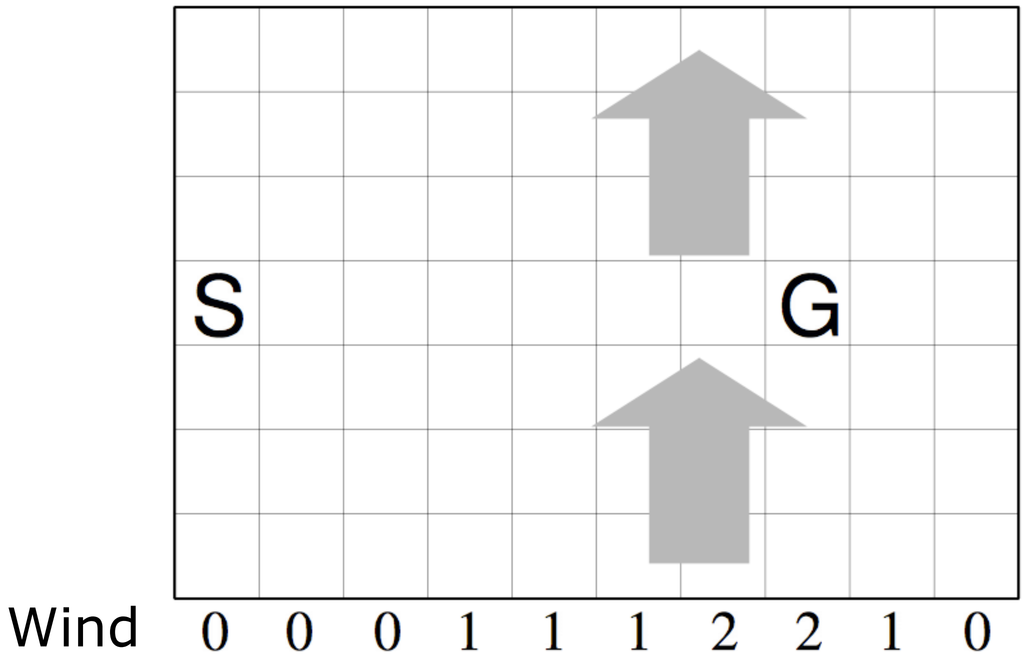
- Outline
  - TD(0)
  - SARSA
  - Q-Learning

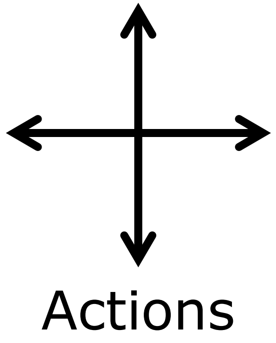
- References
  - Reinforcement Learning: An Introduction [RL Chapter 6 and 7]
  - Sample-based Learning Methods (Coursera)



Why temporal-difference?

- DP requires the knowlege of the MDP dynamics
- MC learns from experience but requires complete episodes to perform the updates
  - It can be applied only to episodic task
  - Even in episodic task it might have problems:





$\gamma = 1$

$R_t = -1$

TD(0)

Temporal-Difference Policy Evaluation – TD(0)

- Temporal-Difference combines MC (**model-free**) with DP (**bootstrapping**):

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

Temporal-Difference Policy Evaluation – TD(0)

- Temporal-Difference combines MC (**model-free**) with DP (**bootstrapping**):

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

$$G_t \approx R_{t+1} + \gamma V(S_{t+1})$$

$$\boxed{V(S_t) \leftarrow V(S_t) + \alpha\,[\underbrace{R_{t+1} + \gamma V(S_{t+1}) - V(S_t)}_{\text{Temporal-Difference Error } (\delta_t)}]}$$

TD(0) Policy Evaluation

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

TD(0) Policy Evaluation

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

S,A,R,S,A,S,A,R,S,A,S,A,R,...

TD(0) Policy Evaluation

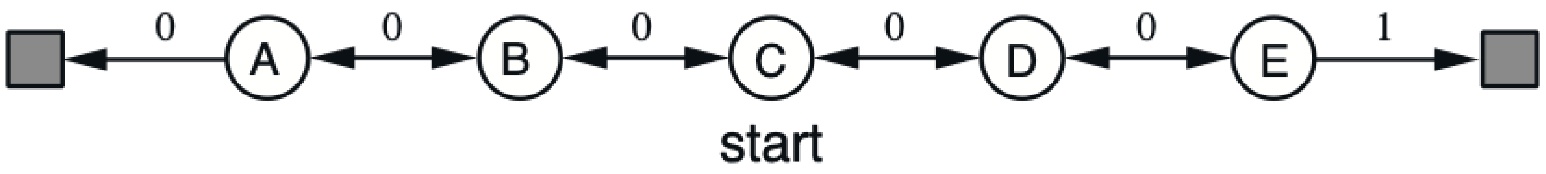
Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

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

update

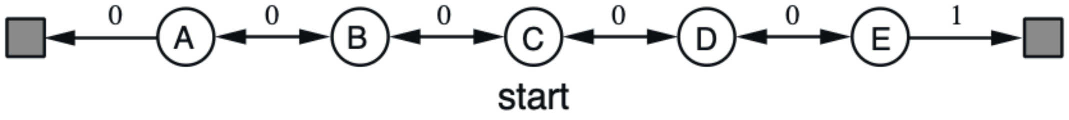
S,A,R,S,A,S,A,R,S,A,S,A,R,...

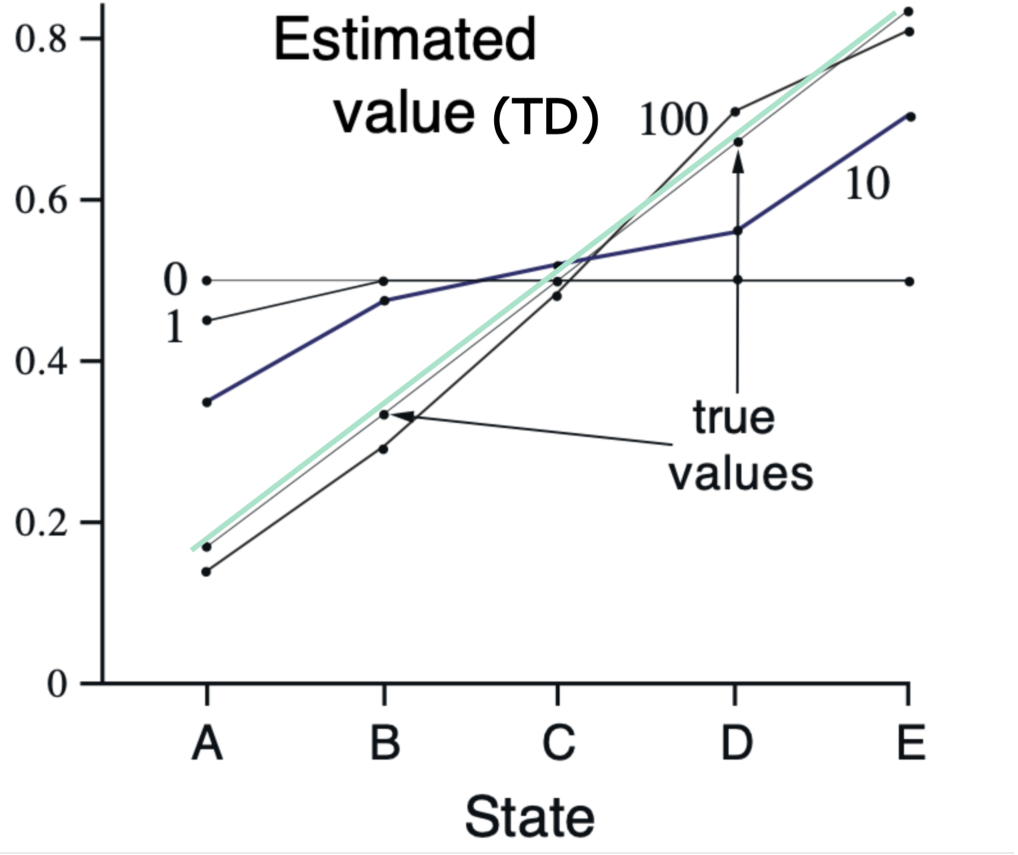
Random Walk Example

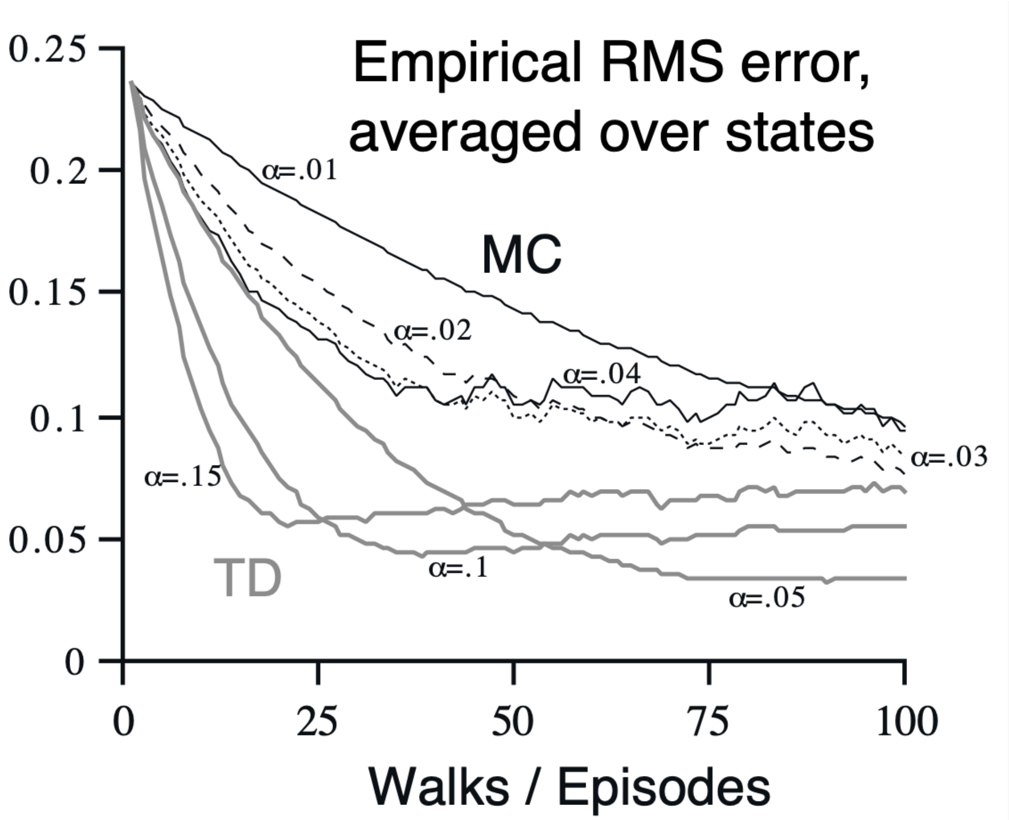


- Let C be the starting state
- Let π be left or right with equal probability in all states (random policy)
- Reward +1 on **right** termination, 0 otherwise
- Assuming $\gamma = 1$, $V_\pi(s)$ represents the probability of ending on the right side

Random Walk Example – Value Function







MC vs TD

- TD can learn **before** knowing the final outcome
  - TD can learn **after every step**
  - MC must wait **until end of episode** before return is known
- TD can learn **without** the final outcome
  - TD can learn from **incomplete sequences**
  - MC can only learn form **complete sequences**
- TD works in continuing (non–terminating) environments MC only works for episodic (terminating) environments

MC vs TD: Bias-Variance

- MC target has lower bias
  - MC return $R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_{t+T}$ is an **unbiased** estimate of $V_\pi(S_t)$
  - TD target $R_{t+1} + \gamma \mathrm{V}(\mathrm{S}_{t+1})$ is a **biased** estimate of $V_\pi(S_t)$, as $\mathrm{V}(\mathrm{S}_{t+1}) \neq V_\pi(S_{t+1})$
- TD target has lower variance
  - Return depends on many random actions, transitions, rewards
  - TD target depends on one random action, transition, reward
- Overall
  - MC works well with function approximation and it is not very sensitive to initial values
  - TD has problem with function approximation and it is more sensitive to initial values

Sampling and Bootstrapping: overview

- **Bootstrapping**: update involves an estimate
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- **Sampling**: update does not involve an expected value
  - MC samples
  - DP does not sample
  - TD samples

SARSA

Let's go back to MC Policy Iteration

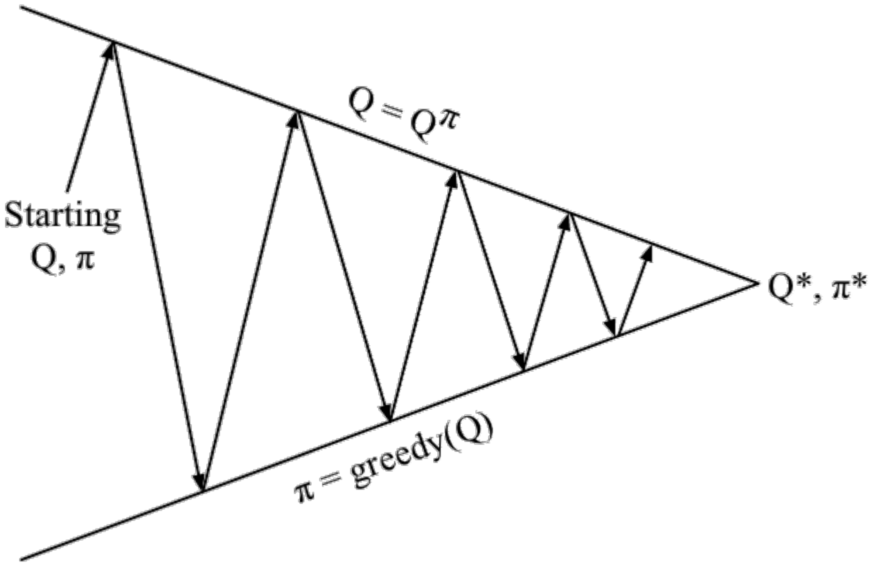
- Evaluation

$$G_t \approx R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$$

$$Q_\pi(s,a) \approx average[G_t | S_t = s, A_t = a]$$

- Improvement

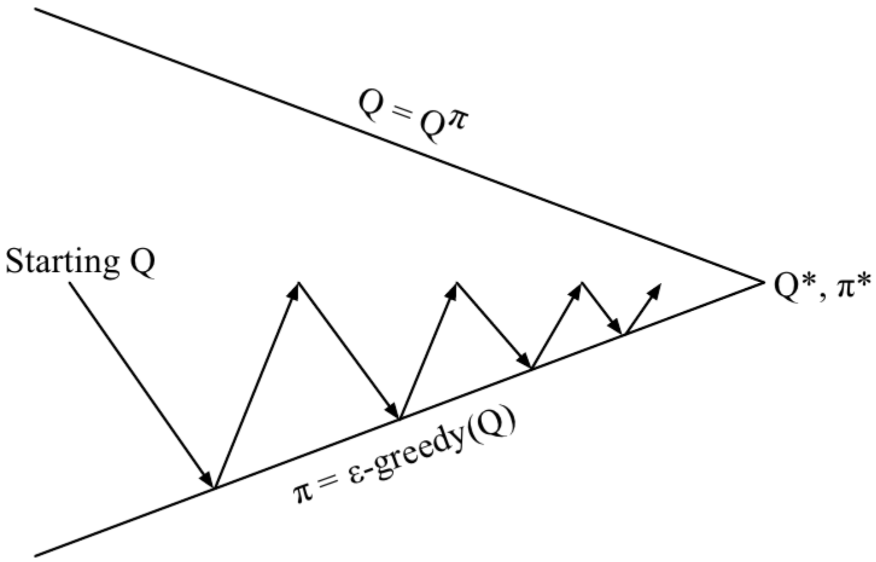
$$\pi'(s) = \arg\max_a Q_\pi(s,a)$$



On–Policy Control with SARSA

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

- Evaluation
  - SARSA
- Improvement
  - ε-Greedy Policy Improvement



On–Policy Control with SARSA

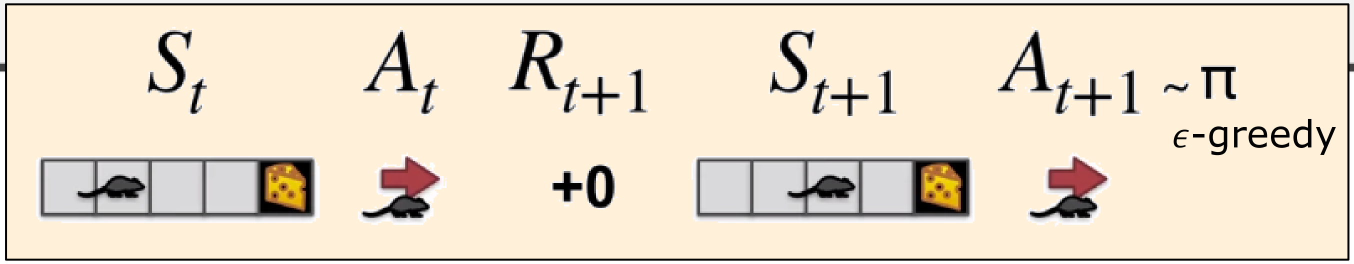
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

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

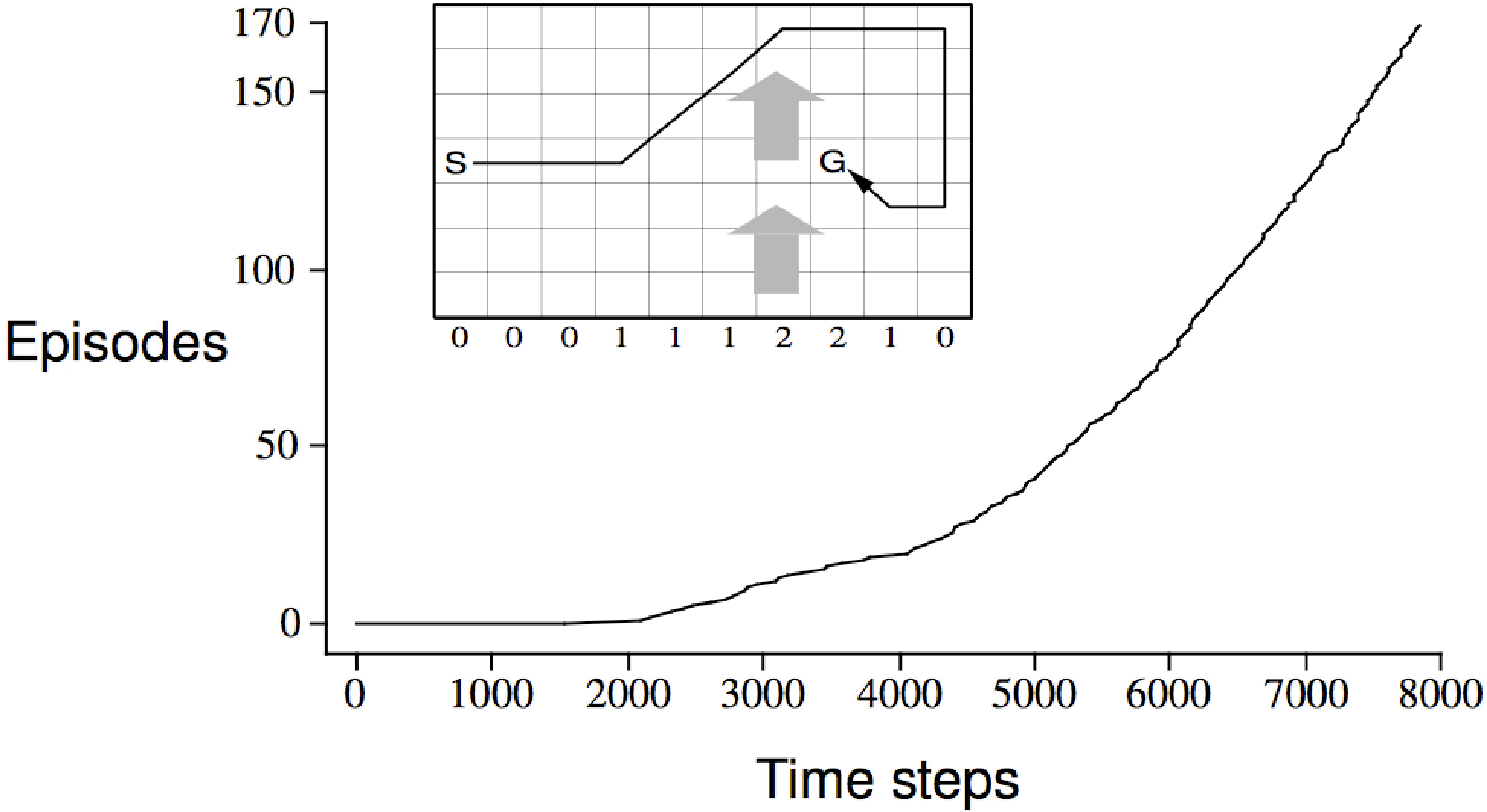
On–Policy Control with SARSA

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 $Q(terminal, \cdot) = 0$

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: SARSA on Windy Gridworld



Q-Learning

Q-Learning: Off-Policy TD Control

- SARSA, as Policy Iteration in DP, is based on Bellman Expectation Equation

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

$$\boxed{Q_\pi(s,a) = \sum_{s',r} p(s',r|s,a) \left(r + \gamma \sum_{a'} \pi(a'|s') Q_\pi(s',a') \right)}$$

- Q-Learning, as Value Iteration in DP, is based on Bellman Optimality Equation

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

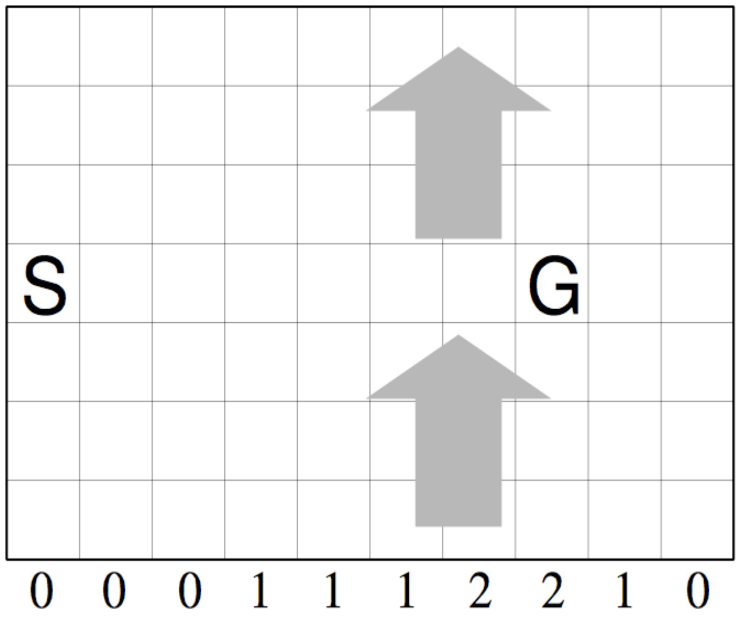
$$\boxed{Q^*(s,a) = \sum_{s',r} p(s',r|s,a) \left(r + \gamma \max_{a'} Q^*(s',a') \right)}$$

Q-Learning: Off-Policy TD Control

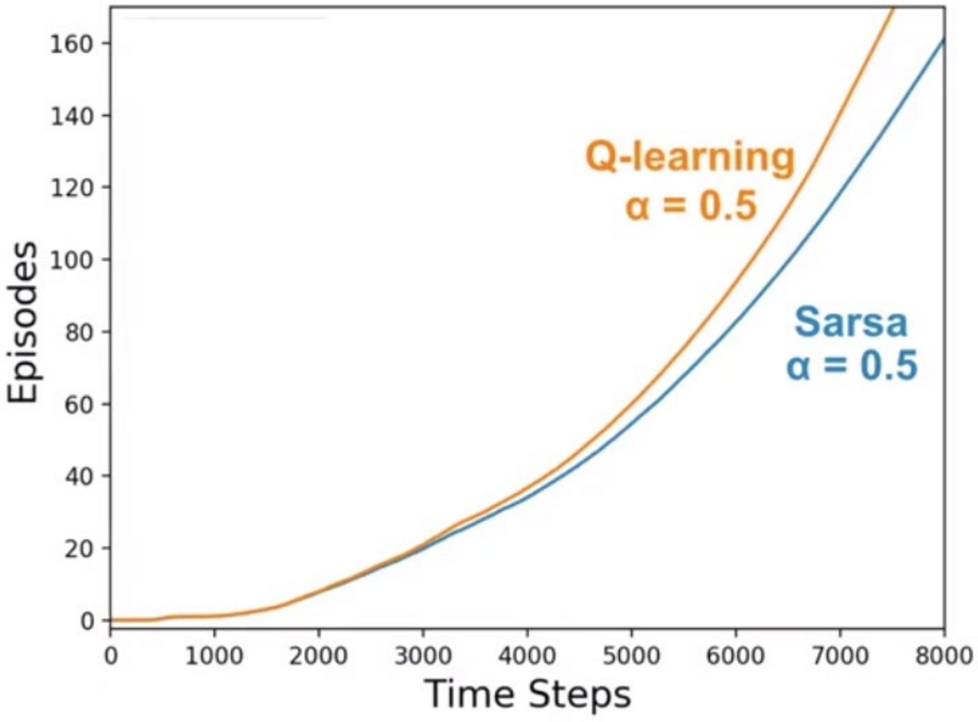
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

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

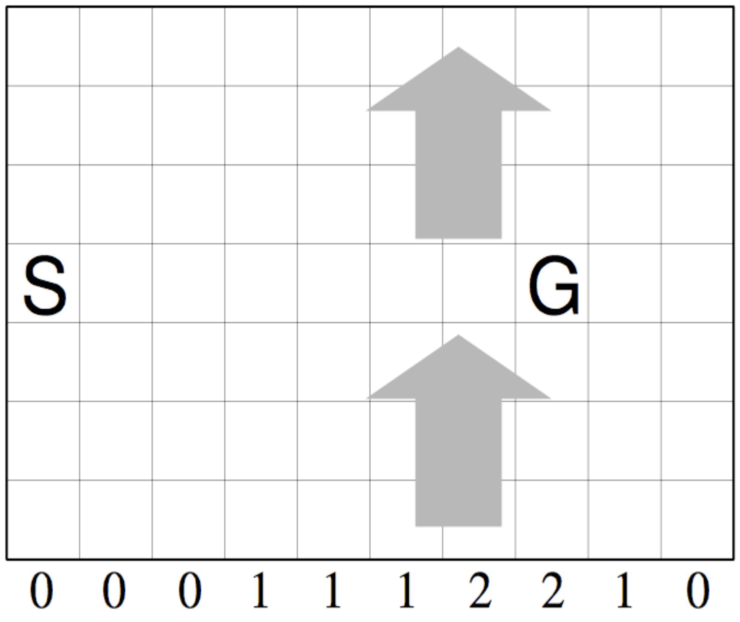
Q-Learning vs SARSA: Windy Gridworld



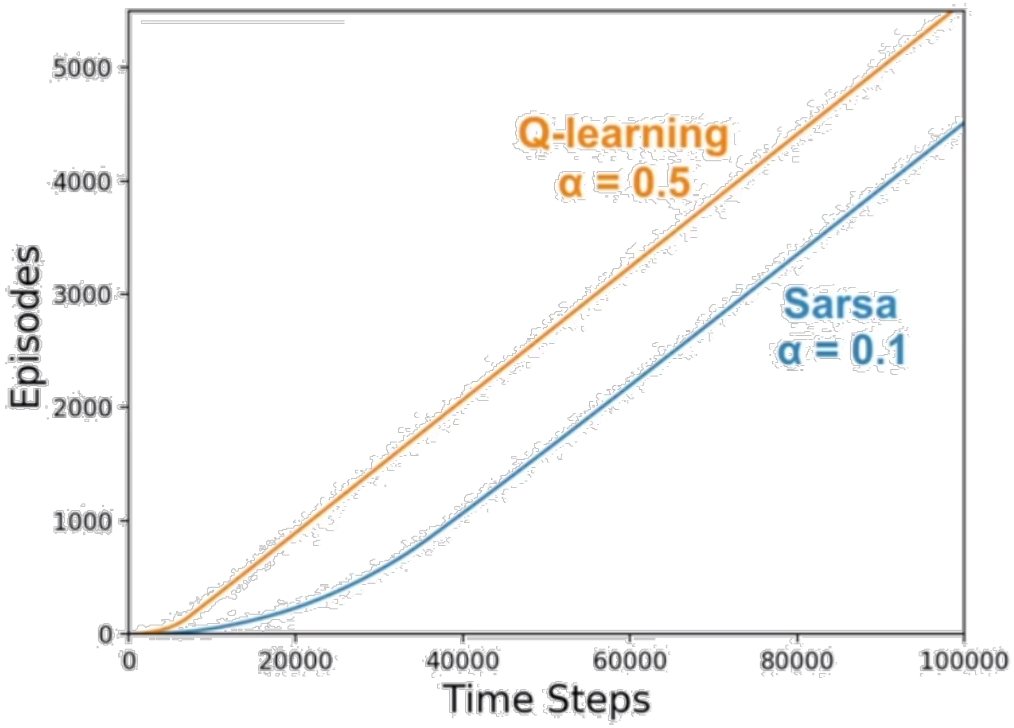
$\epsilon = 0.1$



Q-Learning vs SARSA: Windy Gridworld



$\epsilon = 0.1$



Q-Learning vs SARSA: Cliff Walking Environment