



Reinforcement Learning

Lecture 13: TD Control

9 Sep, 2025

Instructor: Raunak Bhattacharyya



ScAI

YARDI SCHOOL OF ARTIFICIAL INTELLIGENCE
INDIAN INSTITUTE OF TECHNOLOGY DELHI

ϵ - greedy policy:

For all actions to continue to be selected infinitely often,
the policy has to continue to select them

$$\pi_{\text{new}} = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}|} & \text{if } a = \arg \max_a Q(s, a) \\ \frac{\epsilon}{|\mathcal{A}|} & \text{otherwise} \end{cases}$$

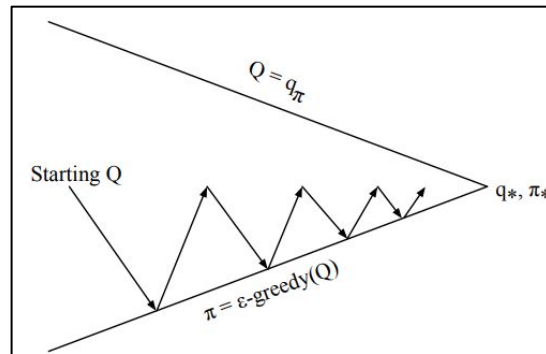
$$\pi(a|s) \geq \frac{\epsilon}{|A(s)|}$$

More general: epsilon-soft policy

GLIE: Greedy in the limit with Inf. Expl.

Conditions to get to the optimal policy

- Explore: Visit all states, try all actions
- Be deterministic ... eventually

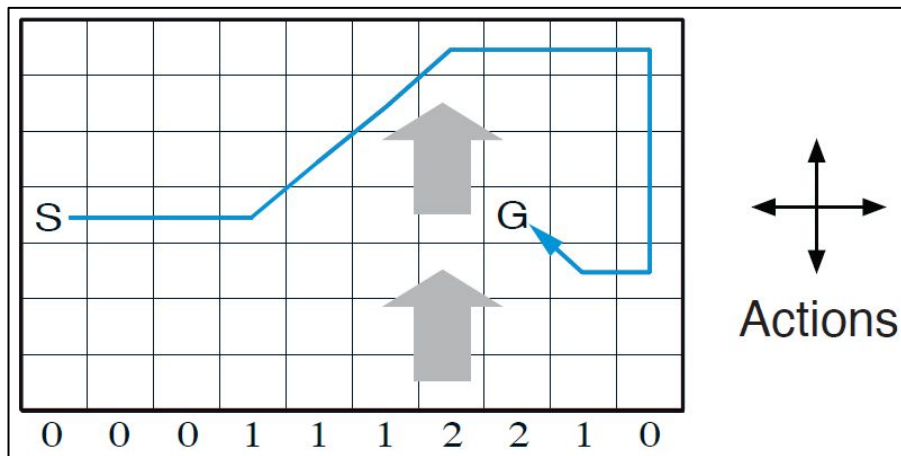


$$q_*(s, a) = \mathbb{E} \left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a \right]$$

Plan

- TD Control Methods
- On-policy & Off-policy
- Exploration & Bandits

Windy Gridworld: Description



- 4 det actions
- Upward crosswind
 - Strength below grid
- Undiscounted episodic task
 - Reward -1 each timestep until G

Why MC may not be suitable in this example?

Reminder: Benefits of TD over MC

- Lower variance
- Online
- Can learn from incomplete sequences
- Idea: GPI with TD, update every timestep

Building Blocks: Dynamic Programming

$$v_{k+1}(s) = \mathbb{E} [R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t \sim \pi(S_t)]$$

$$q_{k+1}(s, a) = \mathbb{E} [R_{t+1} + \gamma q_k(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$

$$v_{k+1}(s) = \max_a \mathbb{E} [R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a]$$

$$q_{k+1}(s, a) = \mathbb{E} \left[R_{t+1} + \gamma \max_{a'} q_k(S_{t+1}, a') \mid S_t = s, A_t = a \right]$$

TD Control: SARSA

$$q_{k+1}(s, a) = \mathbb{E} [R_{t+1} + \gamma q_k(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$

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

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

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)

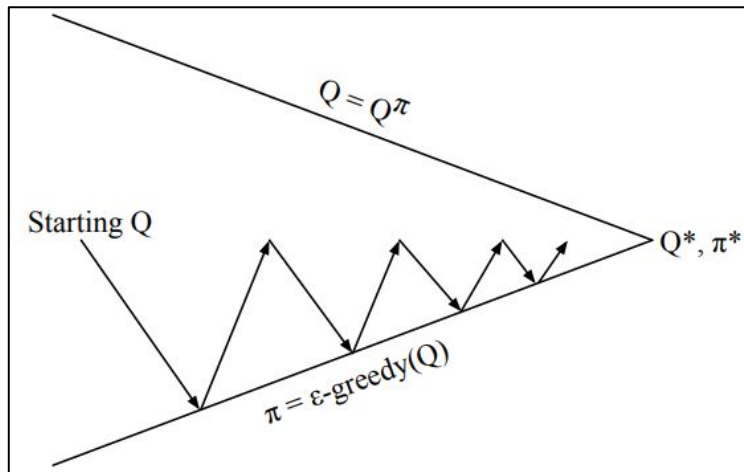
$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$

$S \leftarrow S'; A \leftarrow A';$

 until S is terminal

SARSA in the GPI Framework

Tabular SARSA converges to the optimal action-value function, $q(s, a) \rightarrow q_(s, a)$, if the policy is GLIE*

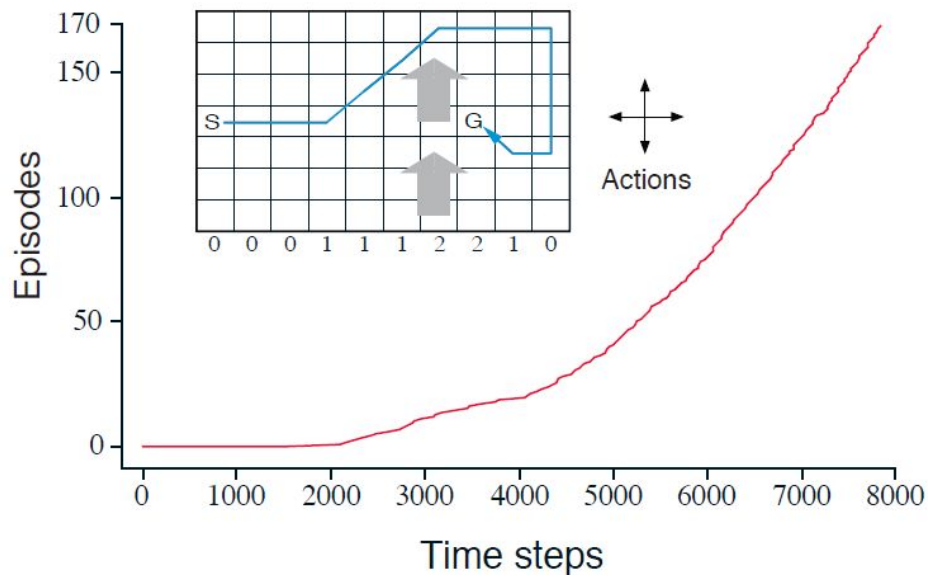


One timestep of evaluation

- GLIE sequence of policies
- Robbins-Monro sequence of step sizes

$$\sum_{t=1}^{\infty} \alpha_t = \infty \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

Windy Gridworld: Solution



- SARSA
 - Eps: 0.1
 - Alpha: 0.5
 - Q init: 0

What does the increasing slope tell us?

Q-Learning

$$q_{k+1}(s, a) = \mathbb{E} \left[R_{t+1} + \gamma \max_{a'} q_k(S_{t+1}, a') \mid S_t = s, A_t = a \right]$$

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

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'

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

 until S is terminal

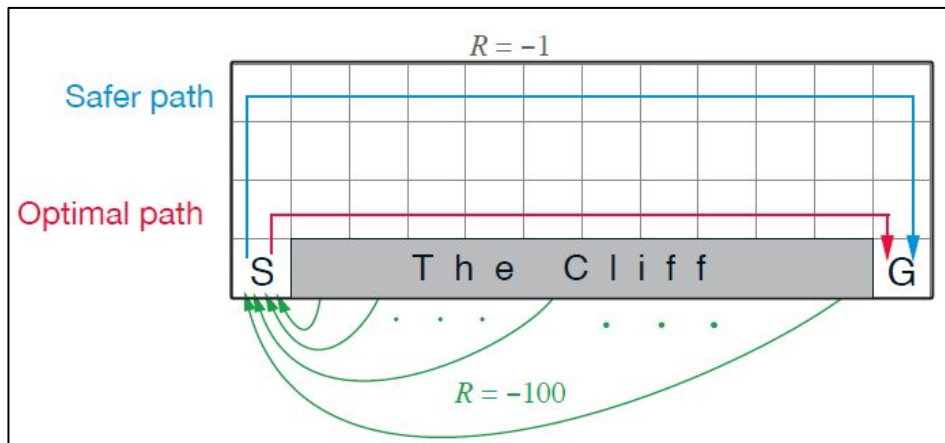
One step walkthrough

Q-Learning: Convergence

Q-learning control converges to the optimal action-value function, $q \rightarrow q^$, as long as we take each action in each state infinitely often.*

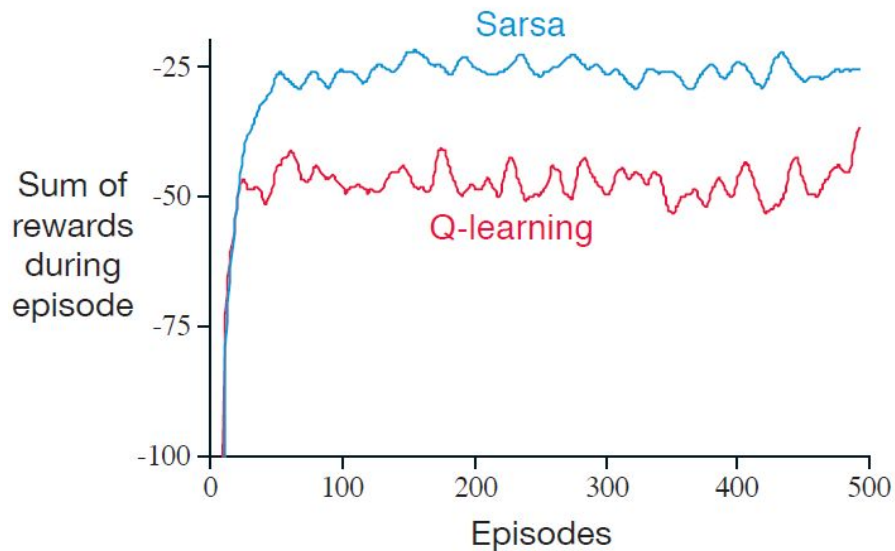
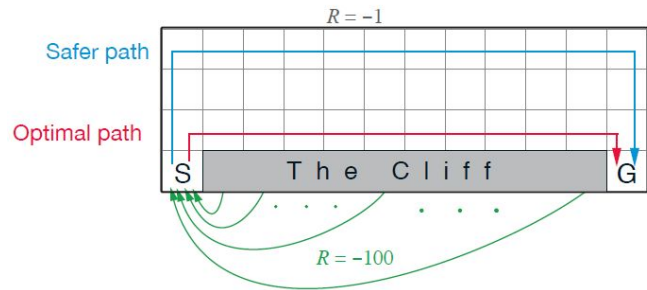
No need for policy to become greedy

Cliff Walking



- 4 det actions
- Undiscounted episodic task
 - Reward -1 each timestep
 - Reward -1 cliff and agent back to S

Cliff Walking



On-policy and Off-policy

- Optimal policy may be deterministic
- But we need stochastic policies to visit state action pair infinitely often
- To estimate the state-action value function

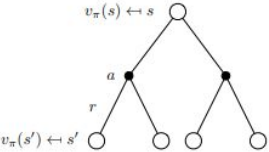
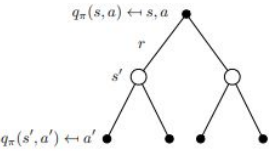
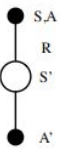
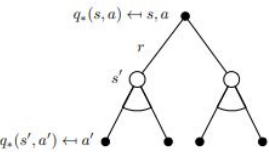
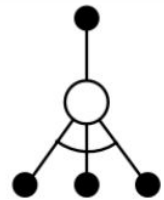
Get to an almost deterministic final policy
(that still explores)

On-policy type of algorithms

Use one policy to explore. Using the exploration, update another (deterministic) policy, which eventually becomes the optimal policy

Off-policy type of algorithms

DP and Sampling

	<i>Full Backup (DP)</i>	<i>Sample Backup (TD)</i>
Bellman Expectation Equation for $v_{\pi}(s)$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $q_{\pi}(s, a)$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $q_*(s, a)$	 Q-Value Iteration	 Q-Learning

DP and Sampling

<i>Full Backup (DP)</i>	<i>Sample Backup (TD)</i>
Iterative Policy Evaluation $V(s) \leftarrow \mathbb{E}[R + \gamma V(S') \mid s]$	TD Learning $V(S) \stackrel{\alpha}{\leftarrow} R + \gamma V(S')$
Q-Policy Iteration $Q(s, a) \leftarrow \mathbb{E}[R + \gamma Q(S', A') \mid s, a]$	Sarsa $Q(S, A) \stackrel{\alpha}{\leftarrow} R + \gamma Q(S', A')$
Q-Value Iteration $Q(s, a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S', a') \mid s, a\right]$	Q-Learning $Q(S, A) \stackrel{\alpha}{\leftarrow} R + \gamma \max_{a' \in \mathcal{A}} Q(S', a')$
where $x \stackrel{\alpha}{\leftarrow} y \equiv x \leftarrow x + \alpha(y - x)$	