Active Reinforcement Learning

Chapter 6, Sutton Barto

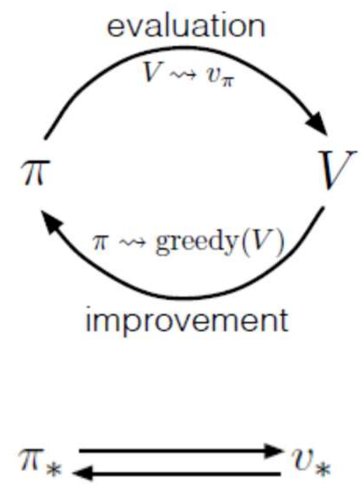
- So far, we have assumed agent with a policy
  - We try to learn how good it is
- Now, suppose agent must learn a good policy (optimal)
  - While acting in uncertain world

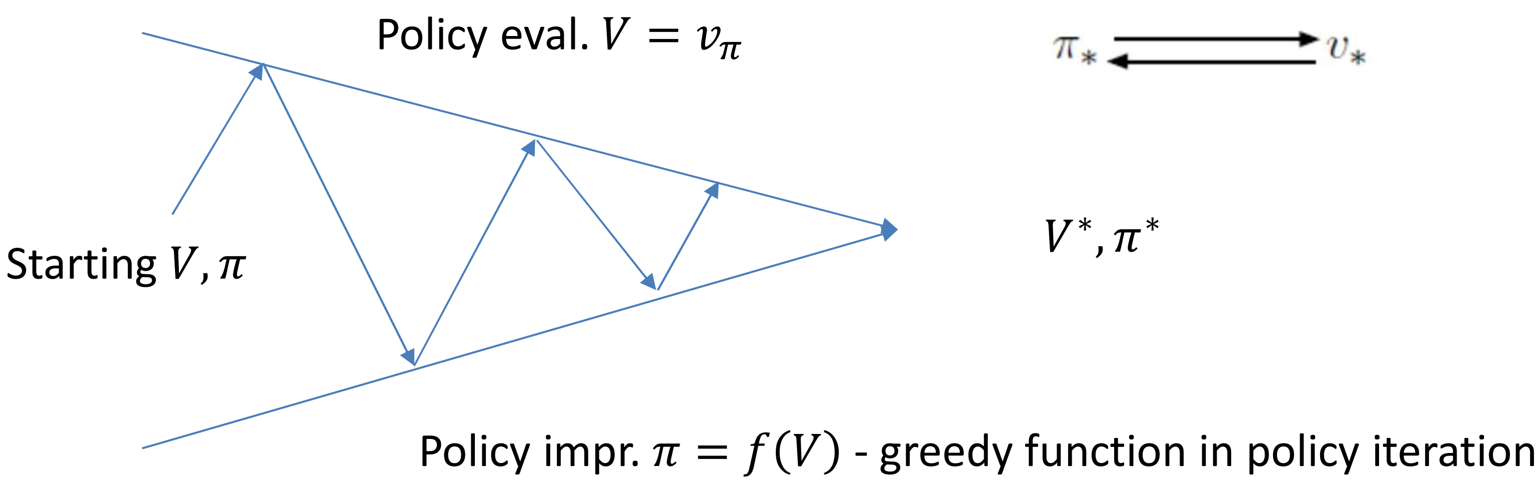
Active Reinforcement Learning

- On-Policy
  - Learn while following a policy (“learning on the job”)
  - Learn about a policy π from experience sampled from π
- Off-Policy
  - Learn outside the policy (“learn from someone else”)
  - Learn about policy π from experience sampled from μ

Policy Iteration

- Policy Evaluation
  - Estimate the value function
- Policy Improvement
  - Find a way to improve the policy

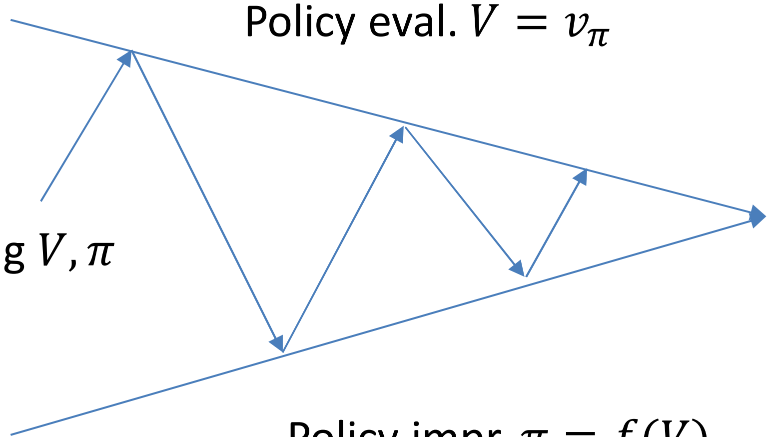




Policy Iteration

- Policy Evaluation
  - Estimate the value function: MC-Evaluation?
- Policy Improvement
  - Find a way to improve the policy?

Model free!
How can we improve policy?



$$\pi'(s) = argmax_{a \in A} \, R_s^a + P_{ss'}^a V(s')$$

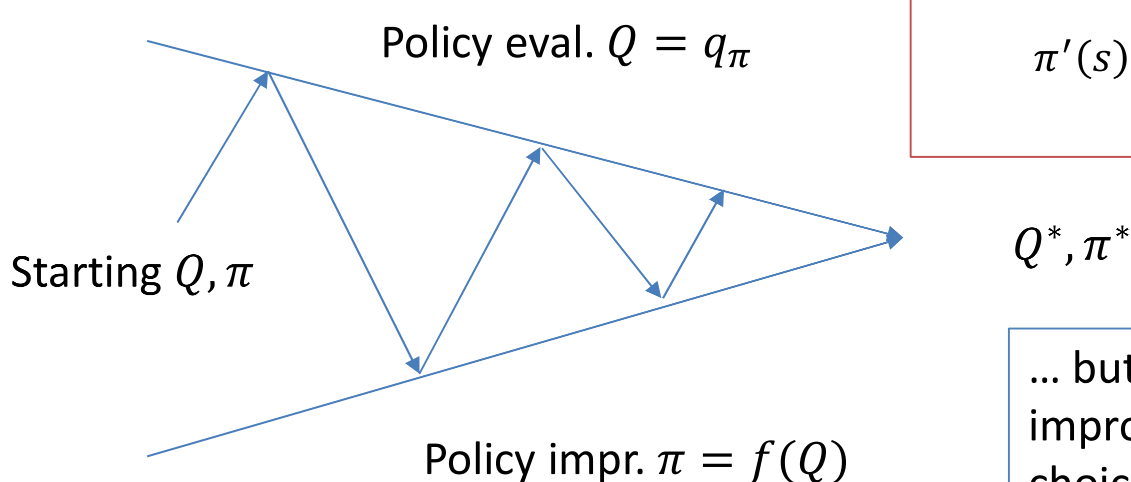
... but no model available

Policy Iteration

- Policy Evaluation
  - Estimate the value function: MC-Evaluation?
- Policy Improvement
  - Find a way to improve the policy?

Model free!
Better to use $Q(s, a)$ …

$$\pi'(s) = argmax_{a \in A}\, Q\,(s, a)$$



… but is the greedy improvement a good choice?

Multi-Armed Bandit Problem

- Action selection:
  – Decide which machines to play

Greedy selection strategy?

1. $R(left) = 0$
2. $R(right) = 1$
3. $R(right) = 2$
4. $R(right) = 1$

We keep going with right, but left was not explored enough

Which is the best strategy?
- We need to explore and exploit, balancing exploration and exploitation

This is a typical RL problem

Exploration versus Exploitation

- Two reasons to take an action in RL
  - **Exploitation**: To try to get reward. We exploit our current knowledge to get a payoff.
  - **Exploration**: Get more information about the world. How do we know if there is not a pot of gold around the corner.
- To explore we typically need to take actions that do not seem best according to our current model.
- Managing the trade-off between exploration and exploitation is a critical issue in RL
- Basic intuition behind most approaches:
  - Explore more when knowledge is weak
  - Exploit more as we gain knowledge

Exploration

- Simple exploration strategy: ϵ-Greedy
  - With prob ϵ select a random action (given m actions)
  - With prob $1-\epsilon$ select the greedy action

$$\pi(a \mid s) = \begin{cases} \frac{\epsilon}{m} + 1 - \epsilon & \text{If } a = argmax_{a \in A} Q(s,a) \\ \frac{\epsilon}{m} & \text{otherwise} \end{cases}$$

Theorem

For any ϵ-Greedy policy π, the ϵ-Greedy policy π' with respect to q_π is an improvement, i.e., $v_{\pi'}(s) \geq v_\pi(s)$

$$q_\pi(s, \pi'(s)) = \sum_{a \in A} \pi'(a|s) q_\pi(s,a) = \frac{\epsilon}{m} \sum_{a \in A} q_\pi(s,a) + (1-\epsilon) \max_{a \in A} q_\pi(s,a) \geq$$

$$\frac{\epsilon}{m} \sum_{a \in A} q_\pi(s,a) + (1-\epsilon) \sum_{a \in A} q_\pi(s,a) \frac{\left(\pi(a|s) - \frac{\epsilon}{m}\right)}{1-\epsilon} = v_\pi(s)$$

Exploration

- Simple exploration strategy: ϵ-Greedy
  - With prob ϵ select a random action (given m actions)
  - With prob $1 - \epsilon$ select the greedy action

$$\pi(a \mid s) = \begin{cases} \frac{\epsilon}{m} + 1 - \epsilon & \text{If } a = argmax_{a \in A} Q(s,a) \\ \frac{\epsilon}{m} & \text{otherwise} \end{cases}$$

Theorem

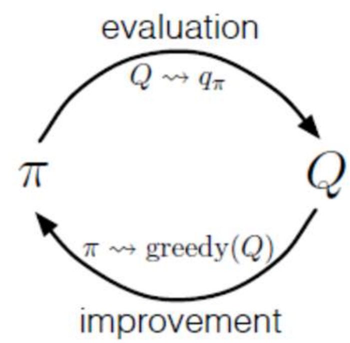
For any ϵ-Greedy policy π, the ϵ-Greedy policy π' with respect to q_π is an improvement, i.e., $v_{\pi'}(s) \geq v_\pi(s)$

$$q_\pi(s, \pi'(s)) = \sum_{a \in A} \pi'(a|s) q_\pi(s,a) = \frac{\epsilon}{m} \sum_{a \in A} q_\pi(s,a) + (1-\epsilon) \max_{a \in A} q_\pi(s,a) \geq$$

$$\frac{\epsilon}{m} \sum_{a \in A} q_\pi(s,a) + (1-\epsilon) \sum_{a \in A} q_\pi(s,a) \frac{\left(\pi(a|s) - \frac{\epsilon}{m}\right)}{1-\epsilon} = v_\pi(s)$$

$$\boxed{\sum_{a \in A} \frac{\left(\pi(a|s) - \frac{\epsilon}{m}\right)}{1-\epsilon} = 1}$$

Policy Iteration



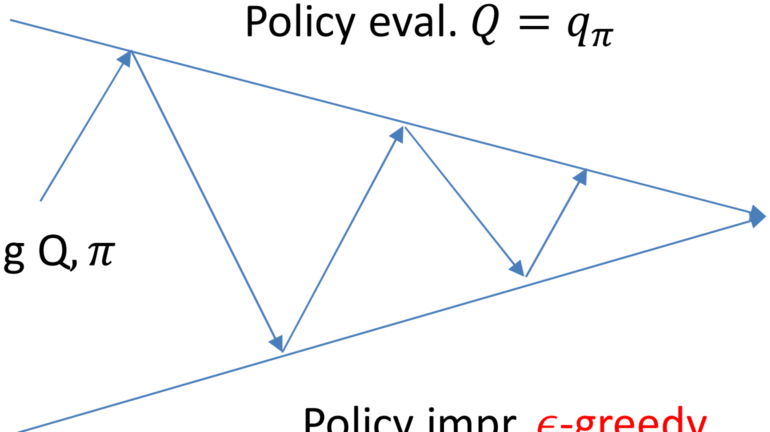
- Policy Evaluation
  - Estimate the value function: MC-Evaluation?
- Policy Improvement
  - Find a way to improve the policy?

Model free!
Better to use $Q(s, a)$ …

Policy eval. $Q = q_\pi$

$\pi'(s) = argmax_{a \in A}\, Q\ (s, a)$

Starting Q, π



Q^*, π^*

Policy impr. ϵ-greedy

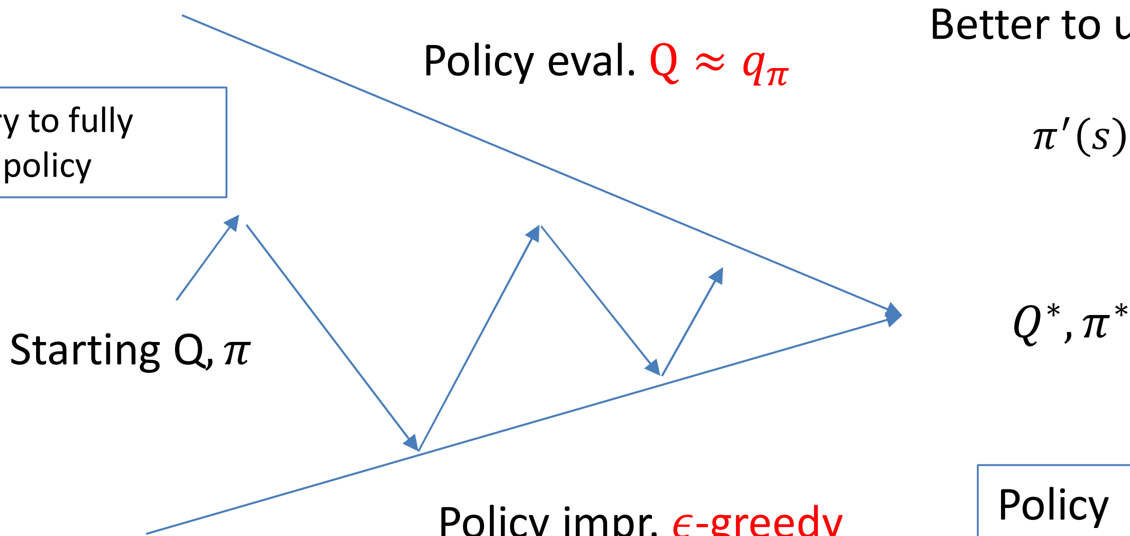
Policy Iteration

- Policy Evaluation
  - Estimate the value function: MC-Evaluation?
- Policy Improvement
  - Find a way to improve the policy?

Model free!
Better to use $Q(s, a)$ …

$$\pi'(s) = argmax_{a \in A}\, Q\,(s, a)$$

Not necessary to fully evaluate the policy



Policy evaluation/improvement can be more frequent

GLIE Exploration

- Exploration policy greedy in the limit of infinite exploration (GLIE) satisfies the following two properties:
  - 1. If a state is visited infinitely often, then each action in that state is chosen infinitely often (with probability 1).

$$\lim_{t \to \infty} N_t(s, a) = \infty$$

  - 2. In the limit (as t → ∞), the learning policy is greedy with respect to the learned Q-function (with probability 1).

$$\lim_{t \to \infty} \pi_t(a \mid s) = 1 \ (i.e., a = argmax_{a' \in A} Q(s, a'))$$

  - For instance, ϵ-Greedy with $\epsilon_t = \frac{1}{t}$

Reduce at each episode

GLIE Monte-Carlo

- Learning with MC
  - Sample k-th episode from π: $\{S_1, A_1, R_2, \ldots, S_T\}$
  - For each state and action in the episode:

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}\left(G_t - Q(S_t, A_t)\right)$$

  - Improve the policy with ϵ-Greedy: $\epsilon_k = \frac{1}{k}$

$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

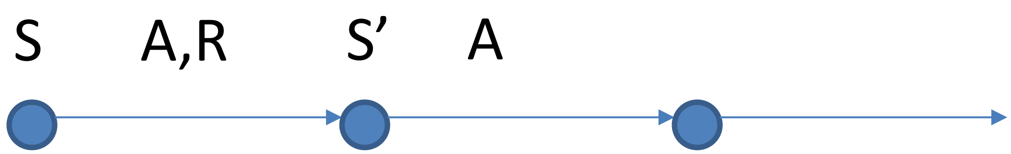
Theorem
The MC-GLIE converges towards the optimal action-value function $Q^*(s, a)$

MC vs TD Learning

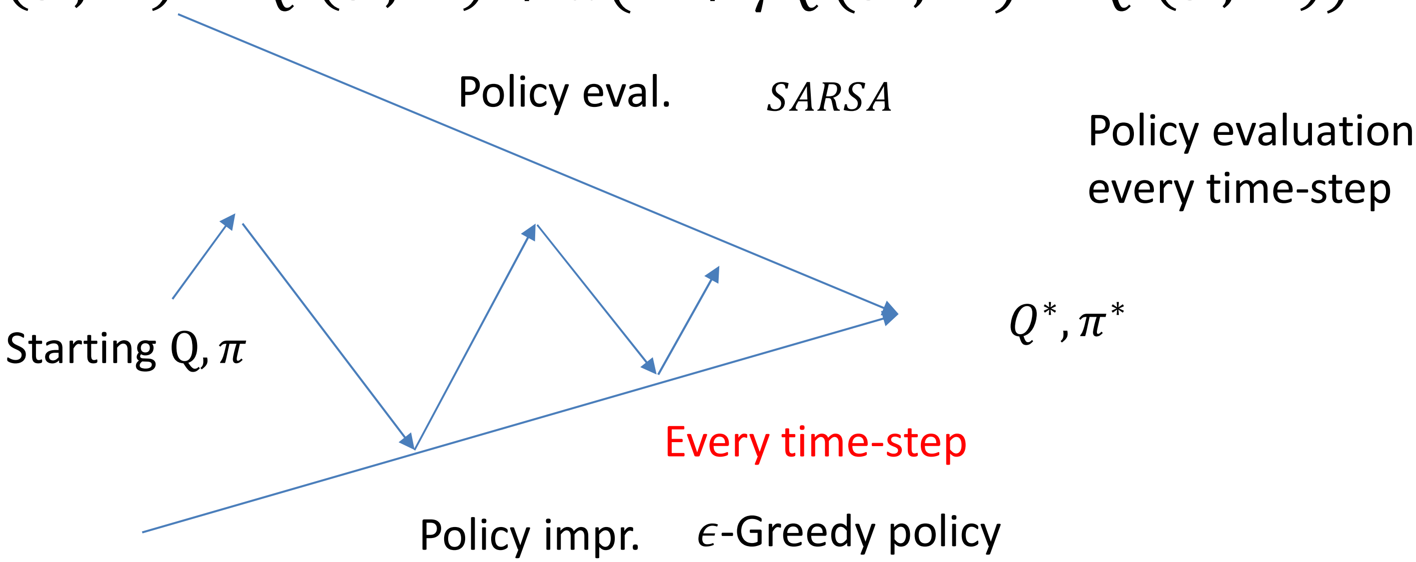
- TD Learning advantages over MC:
  - On-line learning (no termination)
  - Incomplete sequences
  - Exploits Bellman
  - Low variance
- Use TD Learning instead of MC Learning in the control loop
  - Evaluation of $Q(s, a)$
  - Policy improvement with ϵ-greedy
  - Update every step (not after each episode)

TD Learning

- Use TD Learning instead of MC Learning
  - Policy evaluation: evaluation of $Q(s,a)$
  - SARSA action-value update



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



TD-Sarsa

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
    Initialize s
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Repeat (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(s,a) is usually represented as a look-up table

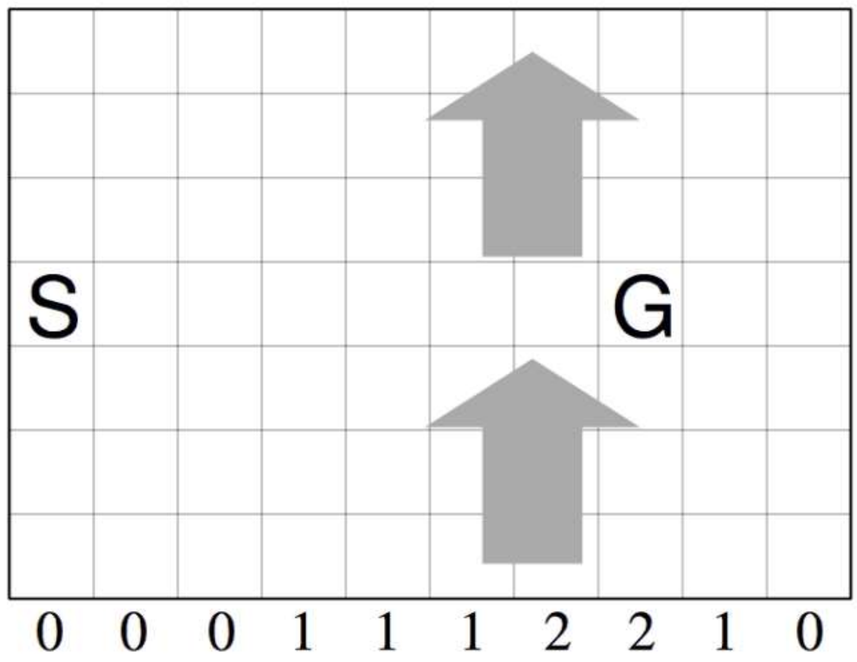
Theorem

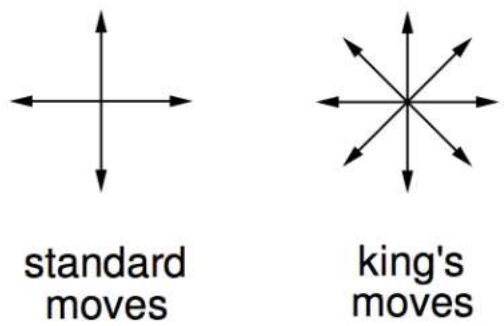
The TD-GLIE converges towards the optimal action-value function under the following conditions:

vanishing oscillation

- GLIE sequence of policies $\pi_t(a \mid s)$
- Robbins-Monro sequence of step α_t: $\lim_{t\to\infty} \sum_{t\in T} \alpha_t = \infty, \sum_{t\in T} \alpha_t^2 < \infty$

Windy Gridworld Example



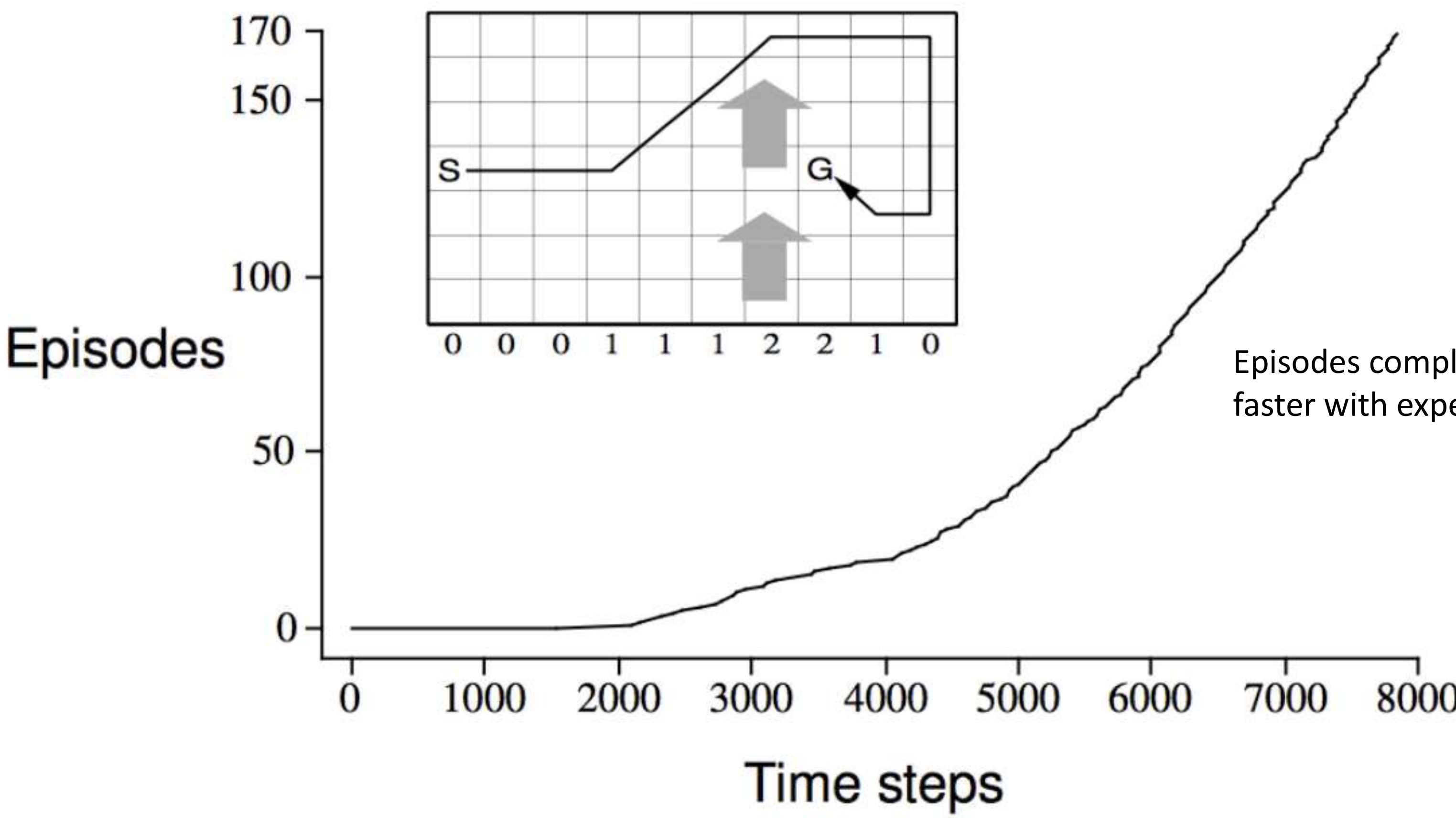


undiscounted, episodic, reward = –1 until goal

R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction

Windy Gridworld Example

ϵ-greedy Sarsa with $\epsilon = 0.1$ and $\alpha = 0.5$, init values $Q(s, a) = 0$



Episodes completed faster with experience

Completed episodes vs time steps

n-step Sarsa

Chapter 7, Sutton Barto
Section 7.1, 7.2

n-step version of SARSA

$n = 1, \quad q_t^1 = R_{t+1} + \gamma Q(S_{t+1})$ SARSA

$n = 2, \quad q_t^2 = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+1})$

…

$n = \infty, \quad q_t^\infty = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^T R_T$ MC

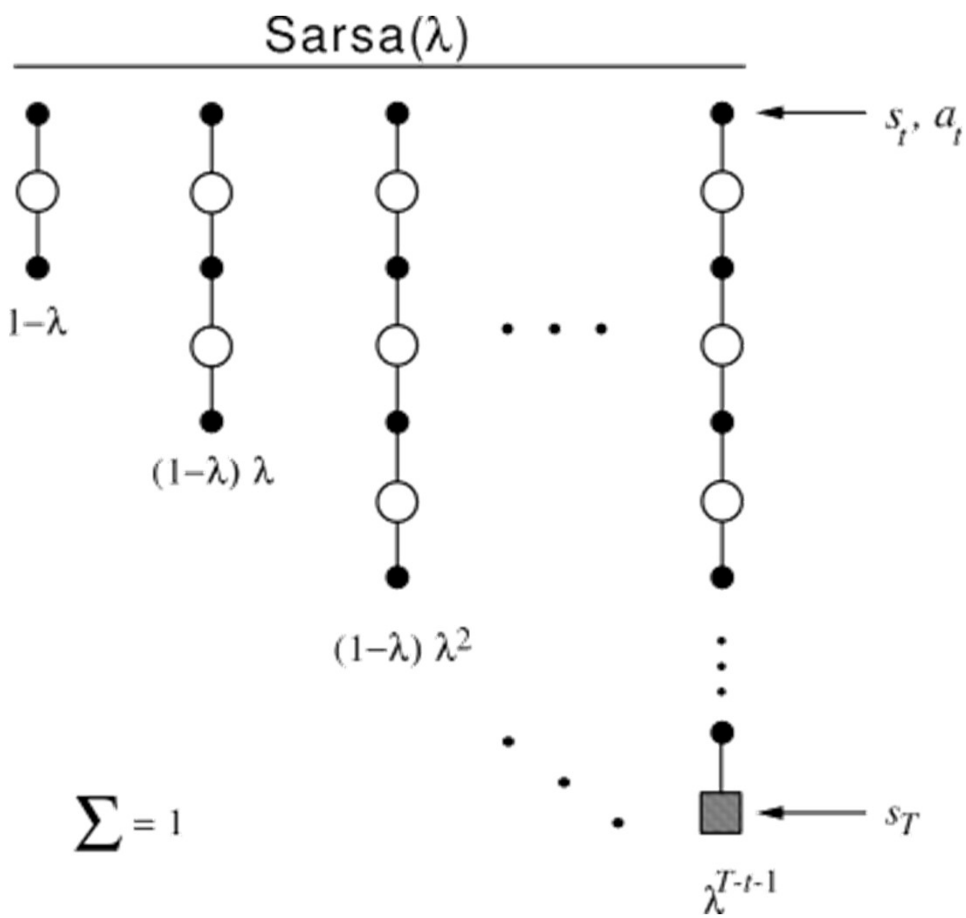
n-step Q-return

$$q_t^n = R_{t+1} + \gamma R_{t+2} + \gamma^{n-1} Q(S_{t+n})$$

n-step SARSA update towards the n-step Q-return

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(q_t^n - Q(S_t, A_t))$$

Forward View Sarsa λ



$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(q_t^\lambda - Q(S_t, A_t))$$

$$q_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} q_t^n$$

Backward View Sarsa λ

Eligibility trace in an online algorithm

Sarsa(λ) has one eligibility trace for each state-action pair

$$E_0(s,a) = 0$$
$$E_t(s,a) = \gamma\lambda E_{t-1}(s,a) + \mathbf{1}(S_t = s, A_t = a)$$

$Q(s,a)$ updated for every state and action:

$$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha\delta_t E_t(s,a)$$

Backward View Sarsa λ

Eligibility trace in an online algorithm

Sarsa(λ) has one eligibility trace for each state-action pair

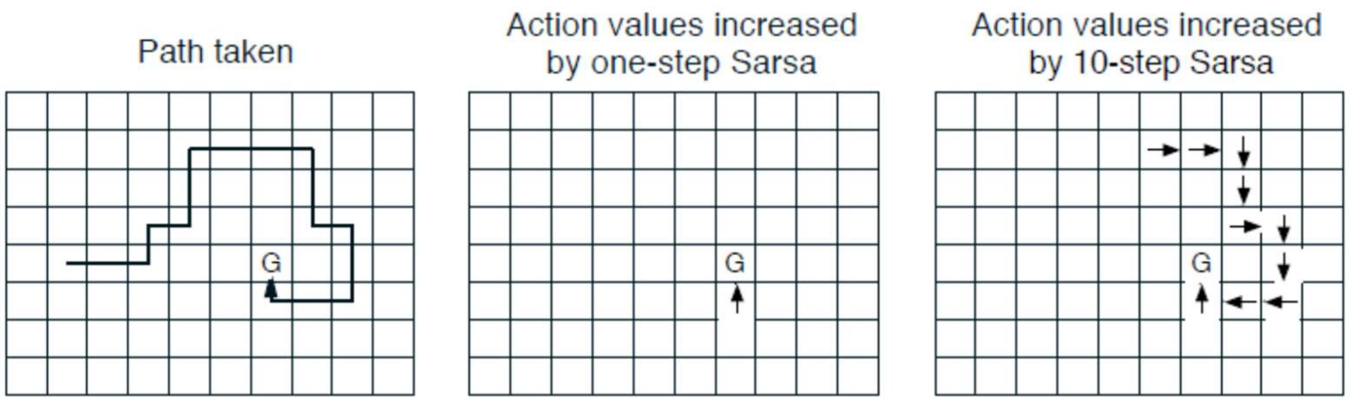
Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Repeat (for each episode):
    $E(s,a) = 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
    Initialize S, A
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $\delta \leftarrow R + \gamma Q(S',A') - Q(S,A)$
        $E(S,A) \leftarrow E(S,A) + 1$
        For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
            $Q(s,a) \leftarrow Q(s,a) + \alpha \delta E(s,a)$
            $E(s,a) \leftarrow \gamma \lambda E(s,a)$
        $S \leftarrow S'$; $A \leftarrow A'$
    until S is terminal

Equivalent to forward view

Backward View Sarsa λ

Eligibility trace in an online algorithm

Sarsa(λ) has one eligibility trace for each state-action pair



Off-Policy Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$ while following another policy $\mu(a \mid s)$

- Learn from observing humans or other agents
- Learn from past policies, re-use experience from old policies
- Learn the *optimal* policy while following an *exploration* policy
- Learn multiple policies while following one policy

Chapter 6, Sutton Barto
Section 6.4, 6.5, 6.6

Off-Policy Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$ while following another policy $\mu(a \mid s)$

- Importance sampling
- Q-learning

Off-Policy Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$ while following another policy $\mu(a \mid s)$

- Importance sampling

Monte-Carlo Off-policy with importance sampling

- Importance along the whole episode

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)\pi(A_{t+1}|S_{t+1}) \ldots \pi(A_T|S_T)}{\mu(A_t|S_t)\mu(A_{t+1}|S_{t+1}) \ldots \mu(A_T|S_T)} G_t$$

- Update towards the correct return

$$V(S_t) \to V(S_t) + \alpha(G_t^{\pi/\mu} - V(S_t))$$

- Not practical, too high variance

Off-Policy Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$ while following another policy $\mu(a \mid s)$

- Importance sampling

TD Off-policy with importance sampling

- Importance sampling correction at each step

$$V(S_t) \rightarrow V(S_t) + \alpha\left(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}(R_{t+1} + \gamma V(S_{t+1})) - V(S_t)\right)$$

- Lower variance than MC importance sampling
- Policies need to be similar over a single step

Off-Policy Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$ while following another policy $\mu(a \mid s)$

- **Q-Learning approach** [Watkins, 1989]
    - Suited for TD(0)
    - No importance sampling
    - Next action using the behavior policy μ, i.e., $A_{t+1} \sim \mu(\cdot \mid S_t)$
    - Assess alternative successor action with policy π, i.e., $A' \sim \pi(\cdot|S_t)$
    - Update $Q(S_t, A_t)$ considering the alternative action

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

Q-Learning

Evaluate target policy $\pi(a \mid s)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$ while following another policy $\mu(a \mid s)$

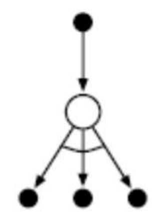
- The target policy π is greedy with respect to $Q(s,a)$

$$\pi(S_{t+1}) = argmax_{a'} Q(S_{t+1}, a')$$

- The behavior policy μ is ϵ –greedy with respect to $Q(s,a)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, argmax_{a'} Q(S_{t+1}, a')) - Q(S_t, A_t))$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a') - Q(S_t, A_t))$$



Q-learning

Theorem

The Q-Learning converges towards the optimal action-value function with GLIE and

$$\lim_{T\to\infty} \sum_{t=1}^{T} \alpha_t = \infty \quad \text{and} \quad \lim_{T\to\infty} \sum_{t=1}^{T} \alpha_t^2 < \infty$$

Q-Learning

1. Start with initial Q-function (e.g., all zeros)
2. Take an action according to an explore/exploit policy (should converge to greedy policy, i.e. GLIE)
3. Perform TD update

   $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a') - Q(S_t, A_t))$$

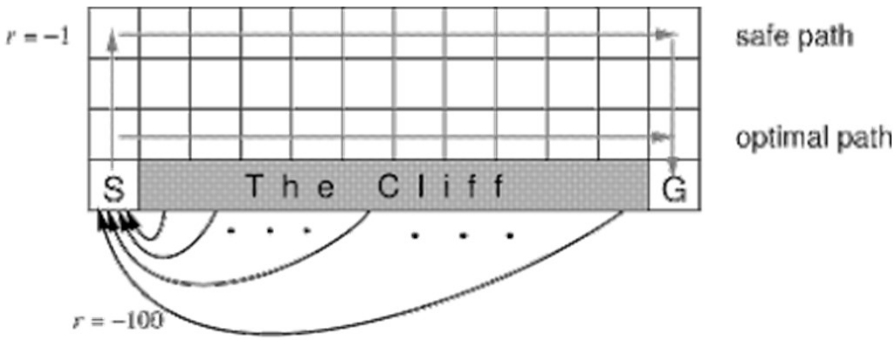
   Q(s,a) is current estimate of optimal Q-function.
4. Goto 2

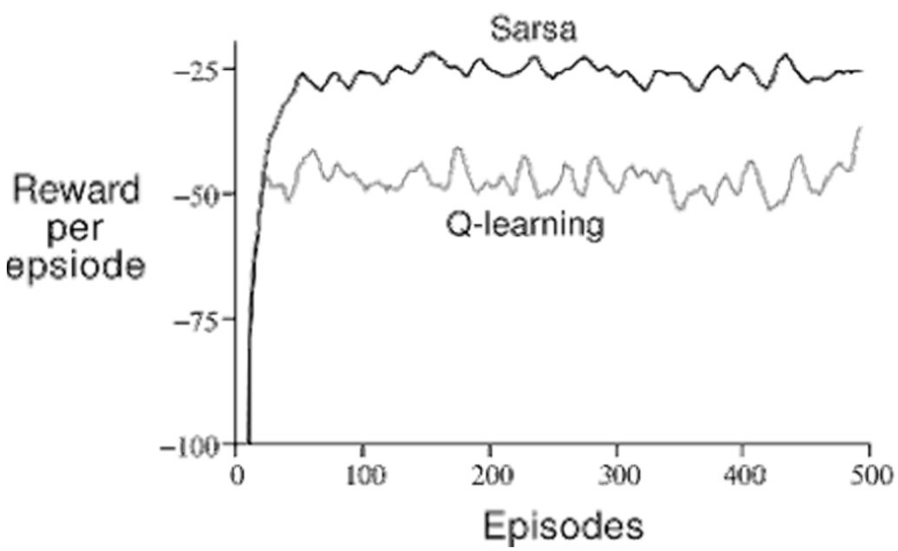
- Does not require model since we learn Q directly
- Uses explicit |S|x|A| table to represent Q
- Explore/exploit policy directly uses Q-values

SARSA vs Q-Learning

Cliff Walking (undiscounted, episodic task)

- ϵ-greedy policy with $\epsilon = 0.1$
- Q-learning off-policy, more risky policy (because of ϵ-gready)





Optimal policy, but lower Reward (off-policy). If ϵ is gradually reduced both policies converge to the optimal one