

强化学习及其应用

Reinforcement Learning and Its Applications

第四章 策略控制

Policy Control

授课人：周晓飞

zhouxiaofei@iie.ac.cn

2018-6-26

第四章 策略控制

4.1 策略优化

4.2 蒙特卡洛策略控制

4.3 时序差分策略控制

4.4 算法总结

第四章 策略控制

4.1 策略优化

4.2 蒙特卡洛策略控制

4.3 时序差分策略控制

4.4 算法总结

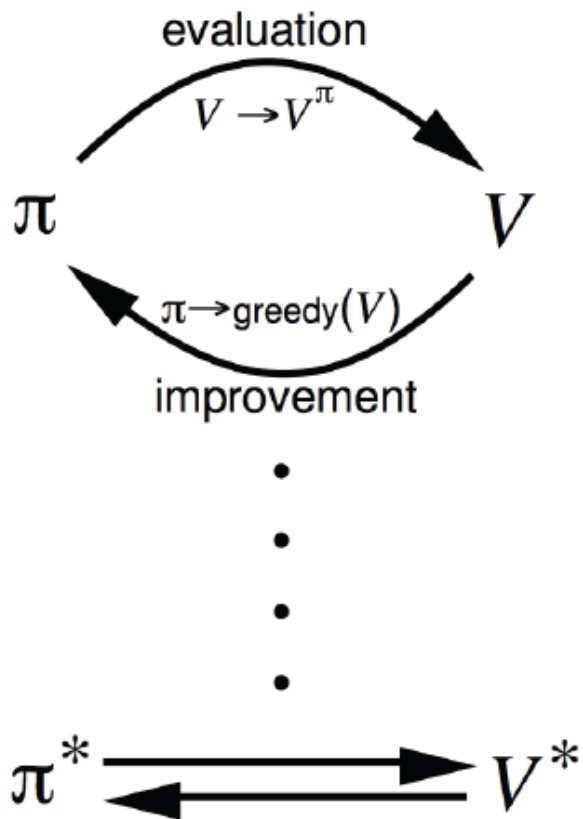
策略优化

问题描述

- For prediction:
 - Input: MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and policy π
 - or: MRP $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$
 - Output: value function v_π
- Or for control:
 - Input: MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$
 - Output: optimal value function v_*
 - and: optimal policy π_*

策略优化

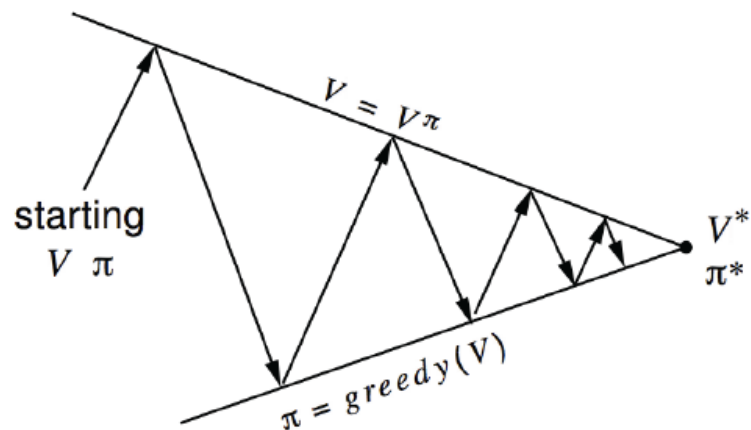
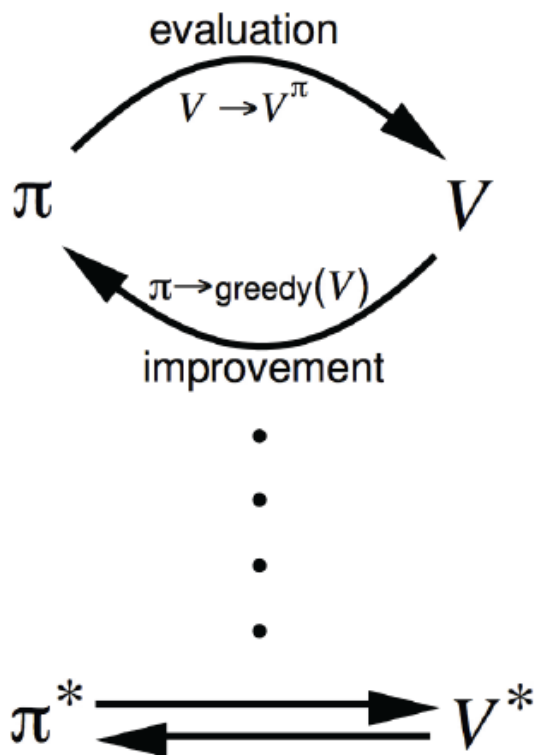
策略迭代



策略优化

策略迭代

两个步骤：Evaluation & Improvement



Policy evaluation Estimate v_π

e.g. Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$

e.g. Greedy policy improvement

策略优化

最优的行动问题：最大的 Action-Value (Q值) 问题

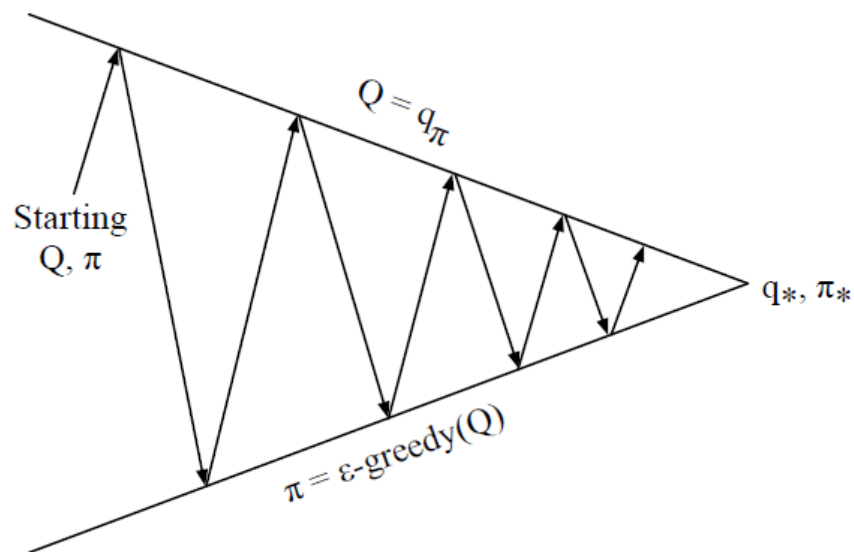
- Greedy policy improvement over $Q(s, a)$ is model-free

$$\pi'(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q(s, a)$$

策略优化

策略优化

类似V迭代的过程，Q替代V值



$$\begin{bmatrix} \text{orange bar} \\ \text{orange bar} \\ \vdots \\ \text{orange bar} \end{bmatrix} = \begin{bmatrix} \text{orange bar} \\ \text{orange bar} \\ \vdots \\ \text{orange bar} \end{bmatrix} + \gamma \begin{bmatrix} \text{blue square} \\ \text{blue square} \\ \vdots \\ \text{blue square} \end{bmatrix} \begin{bmatrix} \text{orange bar} \\ \text{orange bar} \\ \vdots \\ \text{orange bar} \end{bmatrix}$$

$\pi \quad Q \quad R \quad P \quad \pi \quad Q$

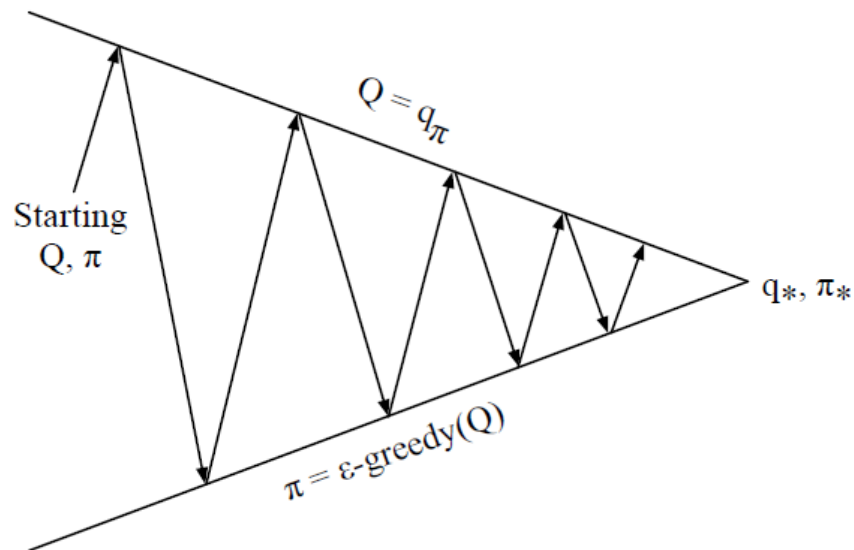
Below the equation, there are arrows indicating the flow of information: a downward arrow from π to Q , and an upward arrow from Q to π .

迭代收敛时，满足 **bellman** 优化方程

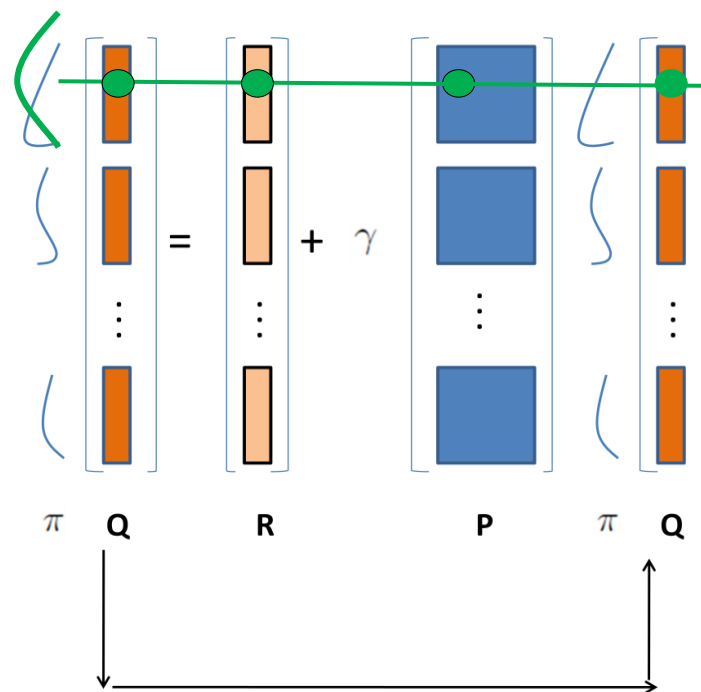
策略优化

策略优化

类似V迭代的过程，Q替代V值



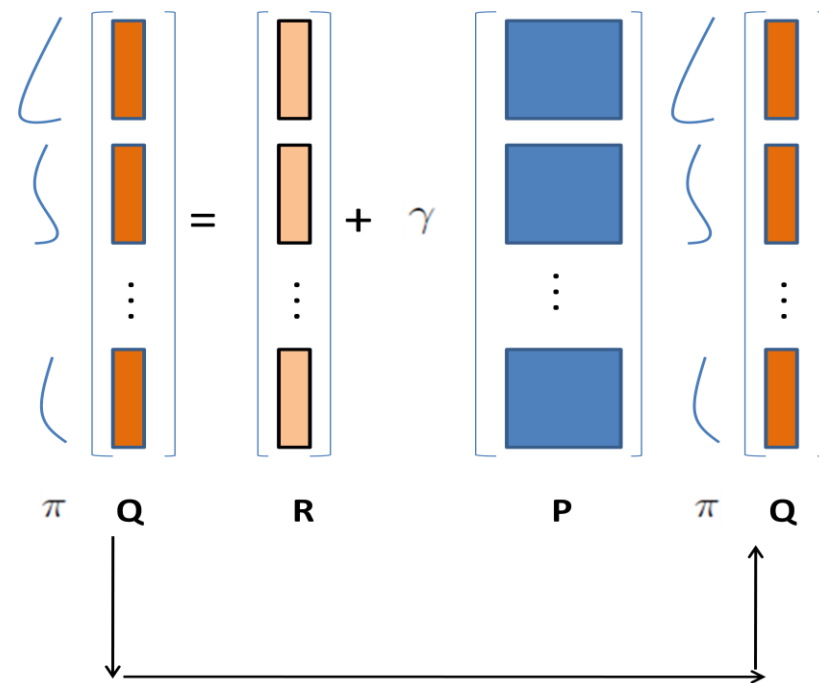
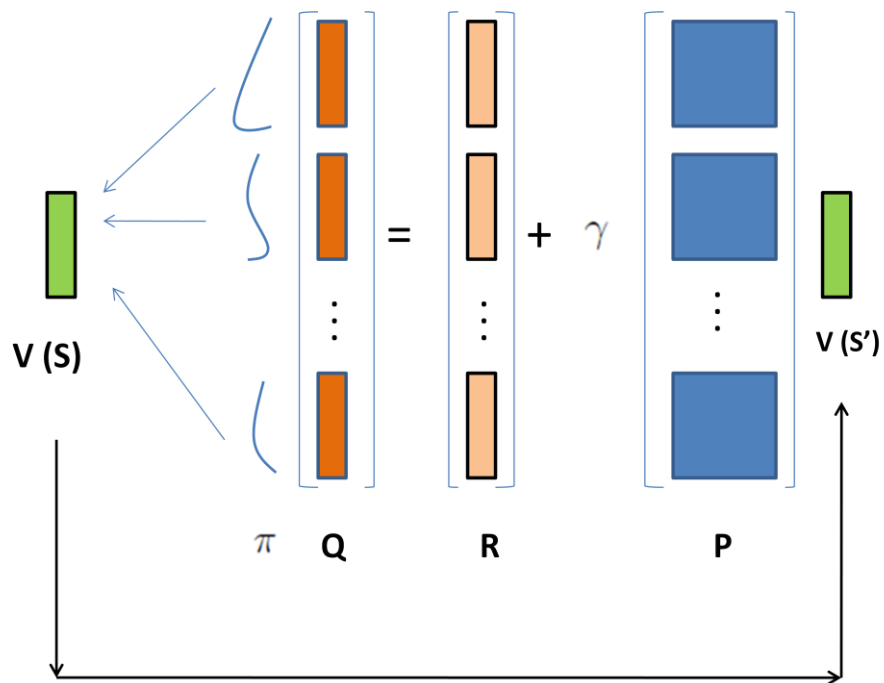
一次随机的优化过程



策略优化

策略优化

比较 V 迭代和 Q 迭代



策略优化

如何改善 Policy ?

ϵ -greedy policy improvement

- Simplest idea for ensuring continual exploration
- All m actions are tried with non-zero probability
- With probability $1 - \epsilon$ choose the greedy action
- With probability ϵ choose an action at random

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \operatorname{argmax}_{a \in \mathcal{A}} Q(s, a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

始终让最优的 a 具有最大的概率，策略分布近似 one-hot 分布。

第四章 策略控制

4.1 策略优化

4.2 蒙特卡洛策略控制

4.3 时序差分策略控制

4.4 算法总结

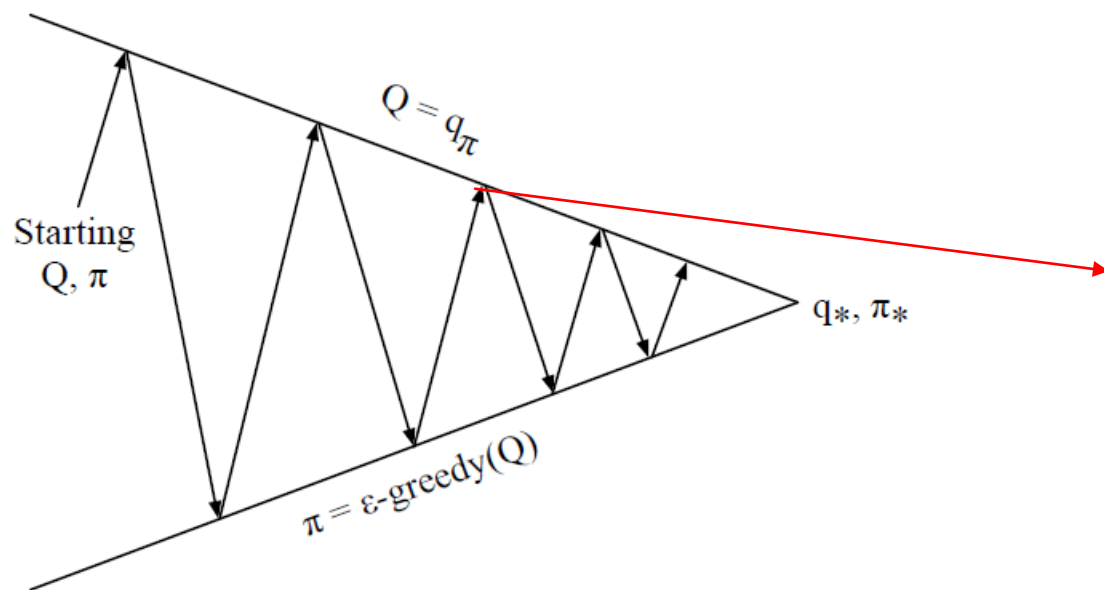
蒙特卡洛策略控制

MC Policy Iteration

两个部分：

Policy evaluation Monte-Carlo policy evaluation, $Q = q_\pi$

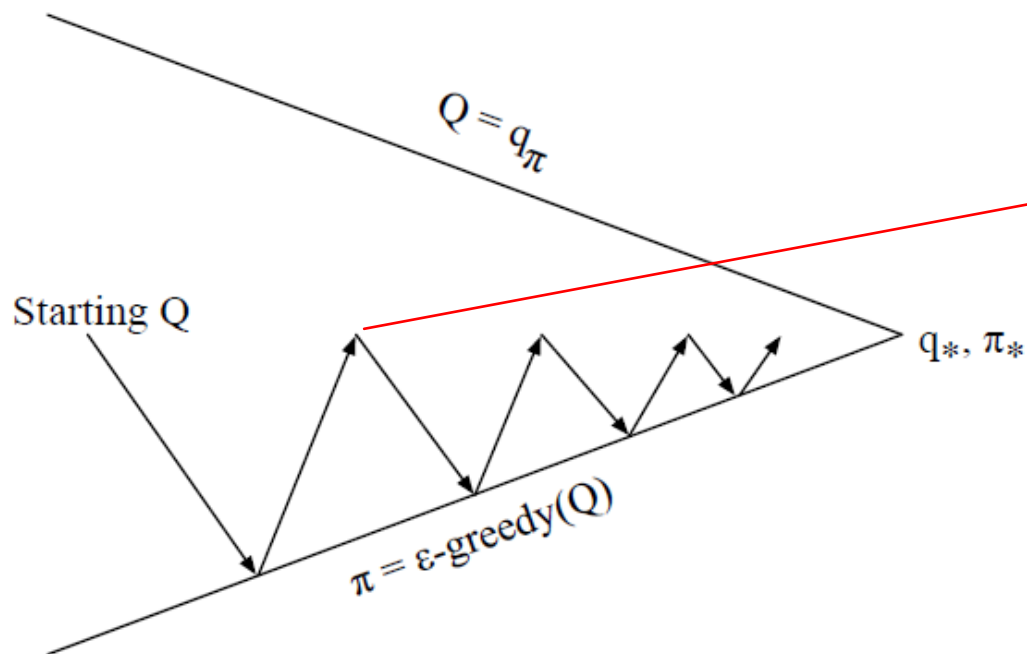
Policy improvement ϵ -greedy policy improvement



每次 policy 的提升, 都充分的采样了

蒙特卡洛策略控制

MC Control



每次 **Policy** 的提升, 没有充分采样,
都是随机的逼近

Every episode:

Policy evaluation Monte-Carlo policy evaluation, $Q \approx q_\pi$

Policy improvement ϵ -greedy policy improvement

GLIE MC Control

- Sample k th episode using π : $\{S_1, A_1, R_2, \dots, S_T\} \sim \pi$
- For each state S_t and action A_t in the episode,

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

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

- Improve policy based on new action-value function

$$\epsilon \leftarrow 1/k$$

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

Theorem

GLIE Monte-Carlo control converges to the optimal action-value function, $Q(s, a) \rightarrow q_(s, a)$*

蒙特卡洛策略控制

GLIE MC Control

- Sample k th episode using π : $\{S_1, A_1, R_2, \dots, S_T\} \sim \pi$
- For each state S_t and action A_t in the episode,

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

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

MC Evaluation

- Improve policy based on new action-value function

$$\epsilon \leftarrow 1/k \longrightarrow \text{收敛技巧}$$

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

ϵ -greedy policy improvement

Theorem

GLIE Monte-Carlo control converges to the optimal action-value function, $Q(s, a) \rightarrow q_(s, a)$*

第四章 策略控制

4.1 策略优化

4.2 蒙特卡洛策略控制

4.3 时序差分策略控制

4.4 算法总结

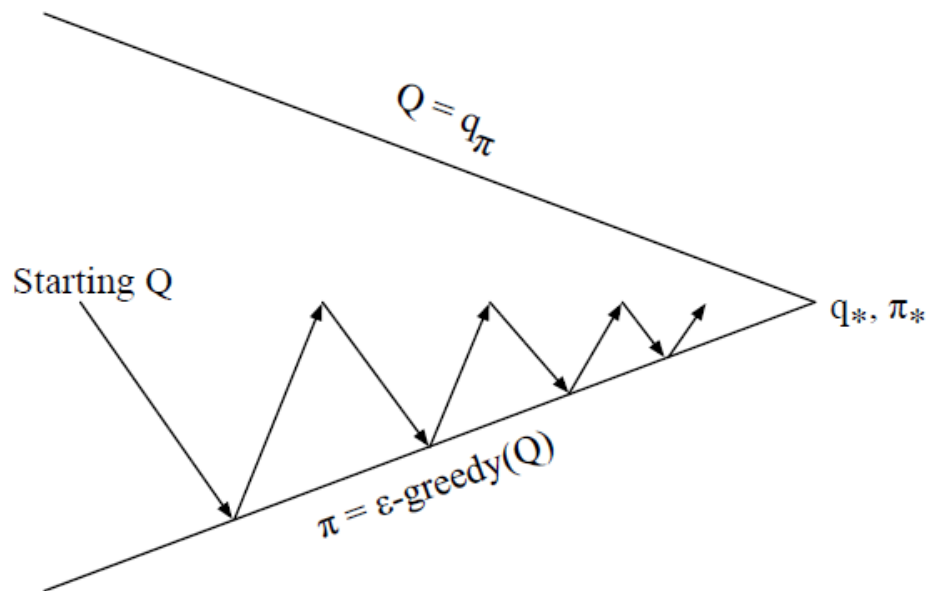
时序差分策略控制

TD for Policy Iteration

- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
 - Lower variance
 - Online
 - Incomplete sequences
- Natural idea: use TD instead of MC in our control loop
 - Apply TD to $Q(S, A)$
 - Use ϵ -greedy policy improvement
 - Update every time-step

时序差分策略控制

Sarsa Control



Every **time-step**:

Policy evaluation **Sarsa**, $Q \approx q_\pi$

Policy improvement ϵ -greedy policy improvement

Sarsa Control

■ Action-Value Evaluation

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

■ Policy Improvement

ϵ -greedy policy improvement

Sarsa Control

■ Sarsa Algorithm

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$

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)

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

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

 until S is terminal

时序差分策略控制

Sarsa (λ)

■ n-step returns

- Consider the following n -step returns for $n = 1, 2, \infty$:

$$\begin{array}{ll} n = 1 & \text{(Sarsa)} \quad q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}) \\ n = 2 & q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}) \\ & \vdots \\ n = \infty & \text{(MC)} \quad q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T \end{array}$$

- Define the n -step Q-return

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

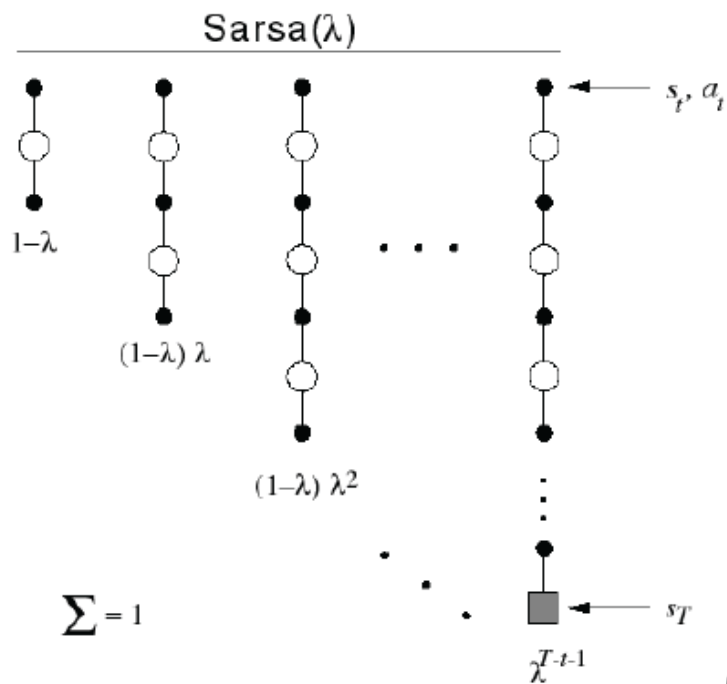
- n -step Sarsa updates $Q(s, a)$ towards the n -step Q-return

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(q_t^{(n)} - Q(S_t, A_t) \right)$$

时序差分策略控制

Sarsa (λ)

■ Forward Sarsa(λ)



- The q^λ return combines all n -step Q-returns $q_t^{(n)}$

- Using weight $(1-\lambda)\lambda^{n-1}$

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

- Forward-view Sarsa(λ)

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

时序差分策略控制

Sarsa (λ)

■ Backward Sarsa(λ)

- Just like TD(λ), we use **eligibility traces** in an online algorithm
- But Sarsa(λ) has one eligibility trace for each state-action pair

$$E_0(s, a) = 0$$

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

- $Q(s, a)$ is updated for every state s and action a
- In proportion to TD-error δ_t and eligibility trace $E_t(s, a)$

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

$$Q(s, a) \leftarrow Q(s, a) + \alpha \delta_t E_t(s, a)$$

时序差分策略控制

Sarsa (λ)

■ Backward Sarsa(λ) Algorithm

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)

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

$E(S, A) \leftarrow E(S, A) + \delta$

 For all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:

$Q(s, a) \leftarrow Q(s, a) + \alpha \delta E(s, a)$

$E(s, a) \leftarrow \gamma \lambda E(s, a)$

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

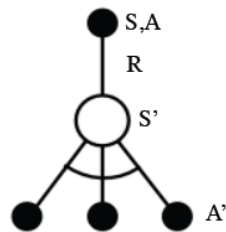
 until S is terminal

时序差分策略控制

Q-Learning

■ Q-Learning Control (本质原理可参看重要性采样)

$$Q(S, A) \leftarrow Q(S, A) + \alpha \left(R + \gamma \max_{a'} Q(S', a') - Q(S, A) \right)$$



Theorem

*Q-learning control converges to the optimal action-value function,
 $Q(s, a) \rightarrow q_*(s, a)$*

时序差分策略控制

Q-Learning

■ Q-Learning Algorithm

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$

Repeat (for each episode):

Initialize S

Repeat (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

第四章 策略控制

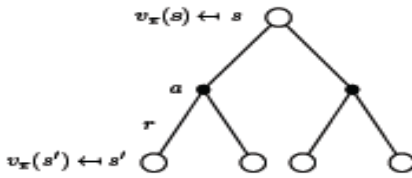

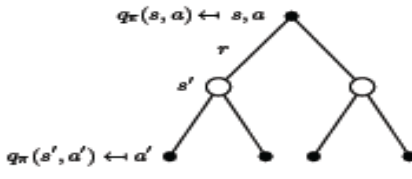
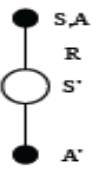
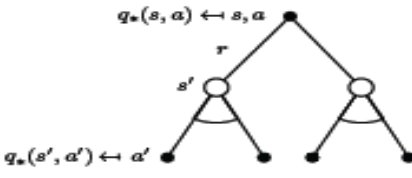

4.1 策略优化

4.2 蒙特卡洛策略控制

4.3 时序差分策略控制

4.4 算法总结

TD V.S. DP

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

TD V.S. DP

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

本讲参考文献

1. Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. (Second edition, in progress, draft).
2. David Silver, Slides@ 《Reinforcement Learning: An Introduction》, 2016.