

强化学习

Reinforcement learning

第5节

无模型控制 (model-free control)

张世周

Outlines

- 5.1 简介
- 5.2 同策 (On-Policy) 蒙特卡洛控制
- 5.3 同策 (On-Policy) 时序差分学习
- 5.4 异策 (Off-Policy) 学习
- 5.5 总结

无模型的强化学习

上一节课我们讲了无模型的预测和估计未知MDP的价值函数

这节课我们讲无模型的控制和优化未知MDP的价值函数

使用无模型的控制

这些例子可以被建模为MDPs，比如：

电梯、平行泊车、船舶操纵、生物反应器、直升机、飞机后勤、机器人足球、雷神之锤、投资组合管理、蛋白质折叠、机器人行走、围棋等。

对于大多数的这些例子：

- 1、要不MDPs模型是未知的，但是我们可以从已有的经验中获取样本
- 2、要不MDPs模型是已知的，但是太大了无法使用，还是需要从已有的经验中获取样本

同策/异策学习

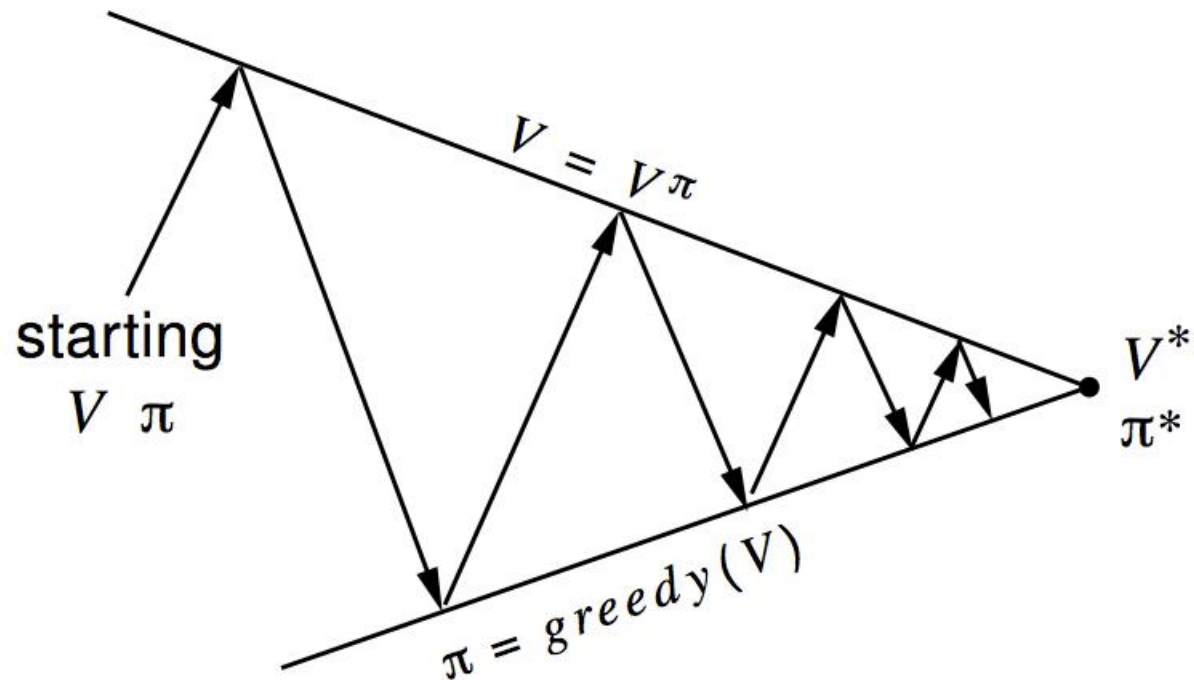
同策学习：是指智能体已经有了一个策略 π ，并且遵循这个策略进行采样，对这个策略迭代评估进行优化。待优化的策略（**目标策略**）就是采样遵循的策略（**采样策略**），称为同策学习。

Learn about policy Π from experience sampled from Π

异策学习：是指智能体已经有了一个策略 π ，但是通过采样策略 μ 进行交互产生经验。这种学习方式类似于“站在别人的肩膀上可以看得更远”。**目标策略与采样策略不同**，称为异策学习。

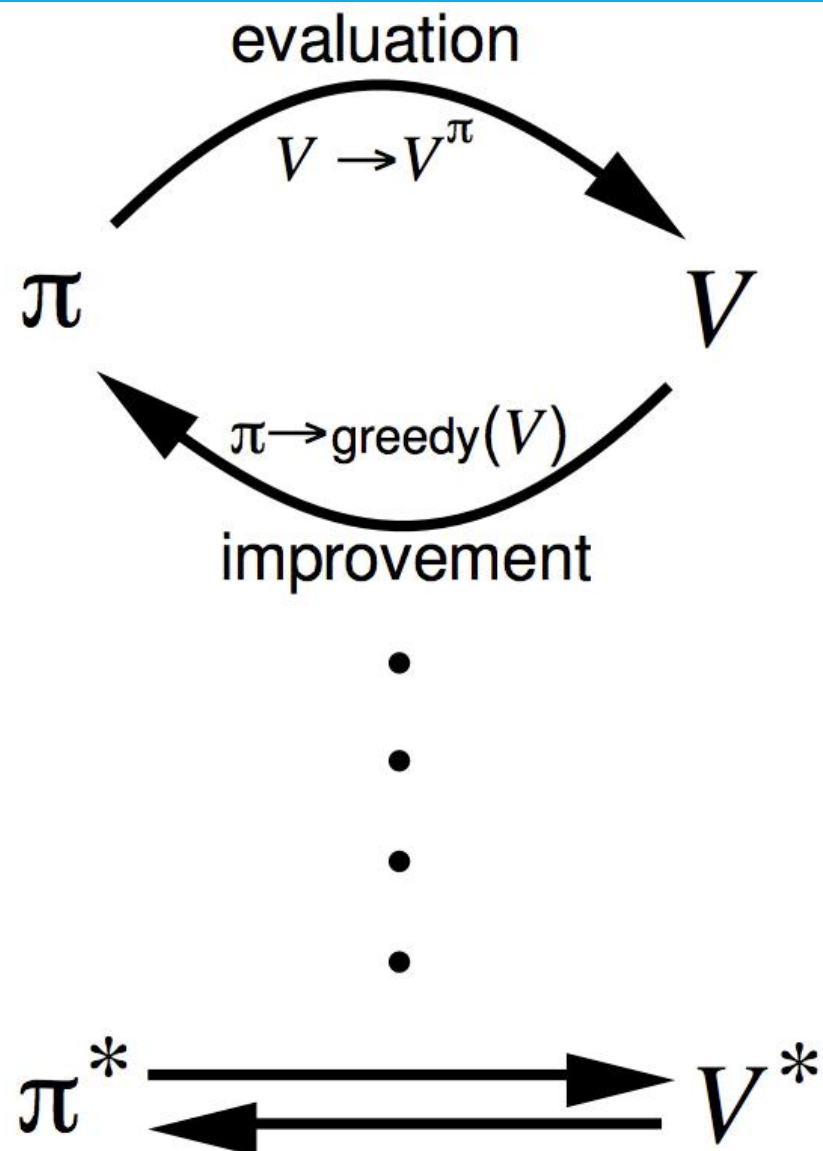
Learn about policy Π from experience sampled from μ

广义策略迭代（回顾）

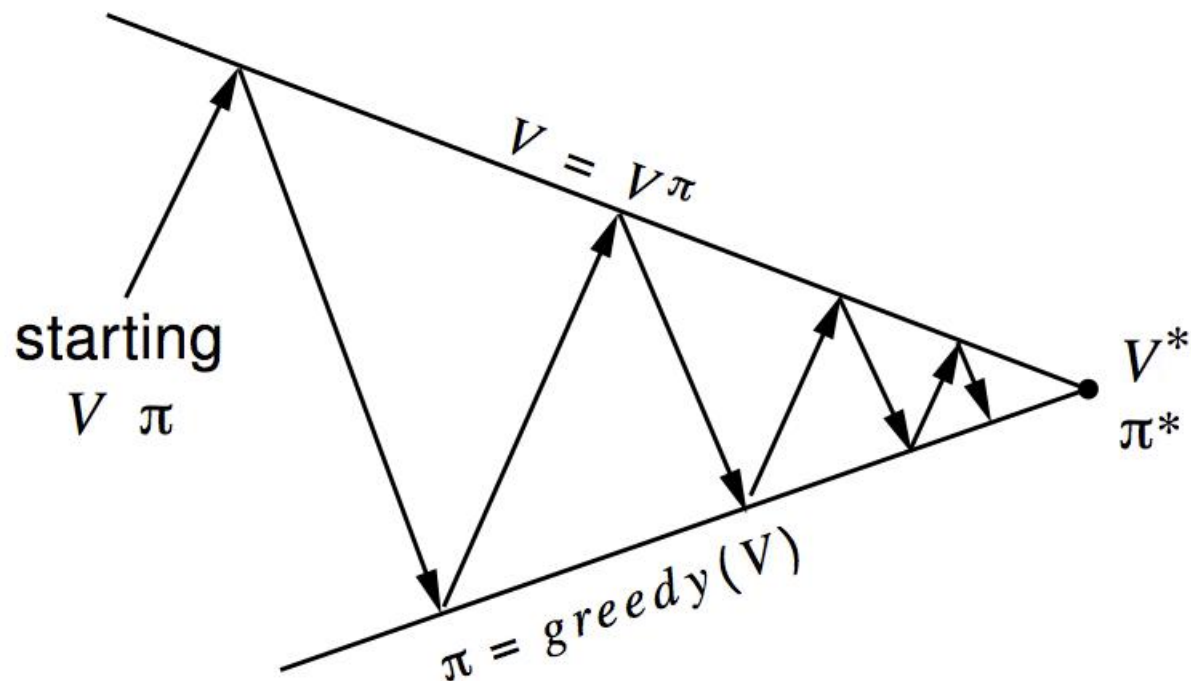


策略评估： 即估计 v_π 。比如迭代策略评估

策略改进： 产生 $\pi' > \pi$ 。比如贪婪策略改进



基于蒙特卡洛评估的广义策略迭代



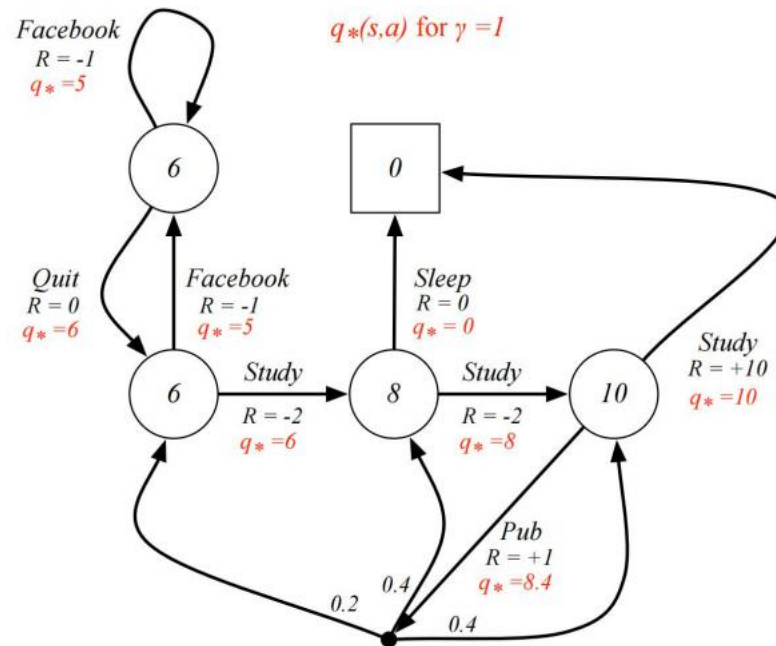
策略评估：基于蒙特卡洛的策略评估： $V = v_{\pi}$?

策略改进：依然是使用贪婪策略改进？

无模型的策略迭代使用动作价值函数

- 基于价值函数 $V(s)$ 的贪婪策略提升需要MDP的模型

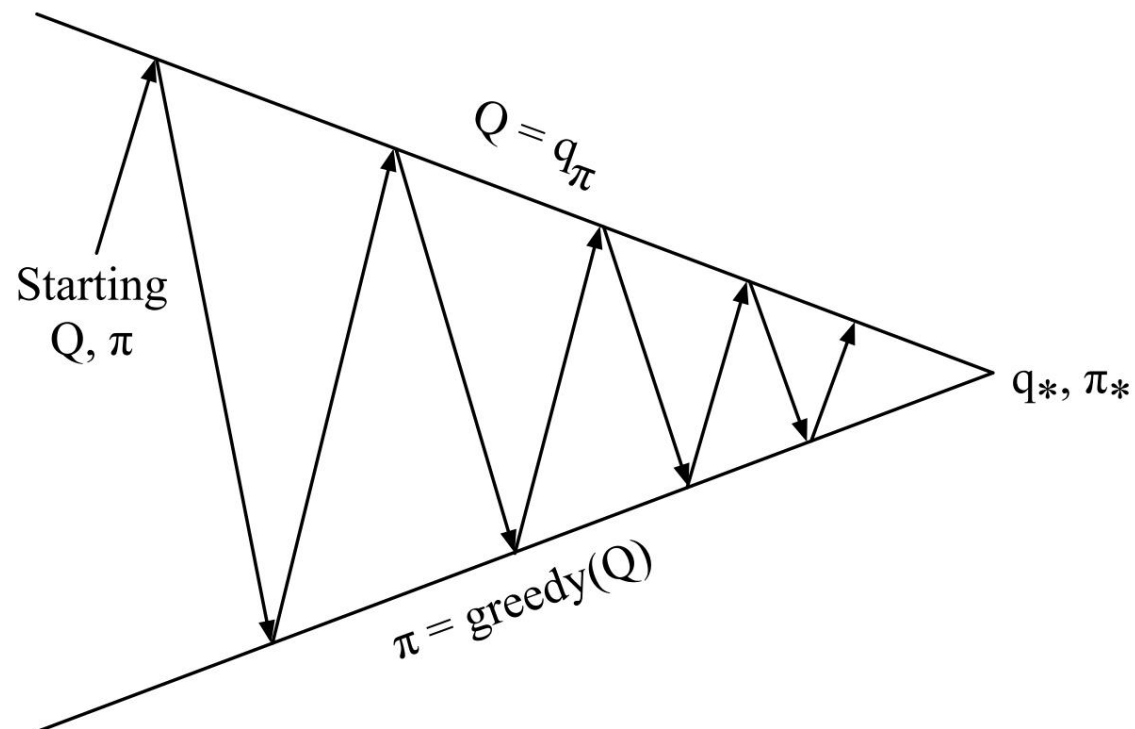
$$\pi'(s) = \operatorname{argmax}_{a \in \mathcal{A}} \mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')$$



- 基于动作价值函数 $Q(a,s)$ 的贪婪策略提升是不需要MDP的模型的

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

基于动作价值函数的广义策略迭代



策略评估：基于蒙特卡洛的策略评估是使 $Q = q_\pi$

策略改进：依然是使用**贪婪策略改进**吗？

贪婪动作选择的例子



"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

在你面前有两扇门：

- 当你打开左边那扇门的时候获得reward=0
即 $V(\text{left})=0$
- 当你打开右边那扇门的时候获得reward=+1
即 $V(\text{right})=+1$
- 当你打开右边那扇门的时候获得reward=+3
即 $V(\text{right})=+2$
- 当你打开右边那扇门的时候获得reward=+2
即 $V(\text{right})=+2$

你确定你选择的是最优的那扇门？

ϵ -贪婪试探 (Exploration)

- ϵ -贪婪试探：确保持续试探的最简单想法
- 所有的 m 个动作都以非零概率被选择
- 以 $1-\epsilon$ 的概率去选择贪婪动作
- 以 ϵ 的概率去选择一个随机动作

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

ϵ -贪婪策略改进

策略改进定理

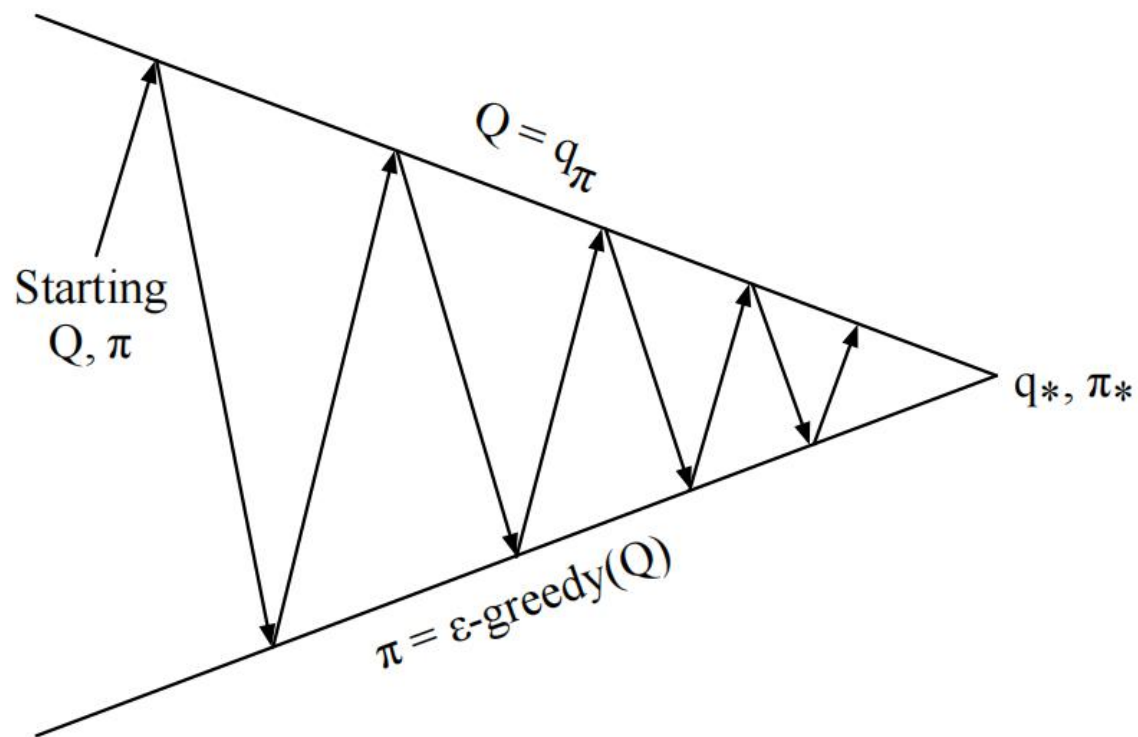
对于任意的 ϵ -贪婪策略 π ，这个 ϵ -贪婪策略 π' 对应的 q_π 是提升的，即 $v_{\pi'}(s) \geq v_\pi(s)$

$$\begin{aligned} q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s, a) \\ &= \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \max_{a \in \mathcal{A}} q_\pi(s, a) \\ &\geq \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \epsilon/m}{1 - \epsilon} q_\pi(s, a) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a) = v_\pi(s) \end{aligned}$$

q_π 最大值大于等于
任意分布的 q_π 平均
值

因此根据策略提升定理， $v_{\pi'}(s) \geq v_\pi(s)$

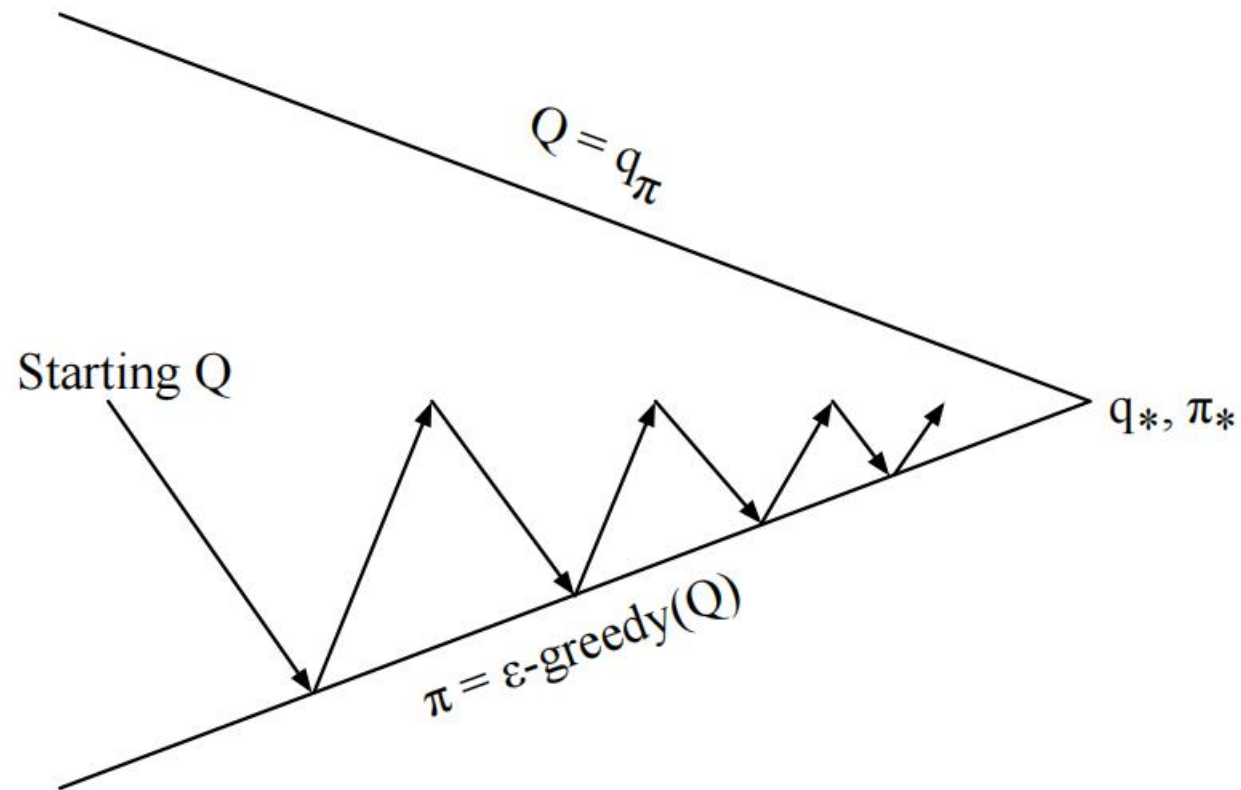
蒙特卡洛策略迭代



策略评估：基于蒙特卡洛的策略评估是使 $Q = q_\pi$

策略改进：使用 ϵ -贪婪策略改进

蒙特卡洛控制



对于每一个episode

策略评估：基于蒙特卡洛的策略评估是使 $Q \approx q_\pi$

策略改进：使用 ϵ -贪婪策略改进

定义

Greedy in the Limit with Infinite Exploration (GLIE)

- 所有状态-动作对都被无限次地探索

$$\lim_{k \rightarrow \infty} N_k(s, a) = \infty$$

- 这样的策略趋同于贪婪的策略

$$\lim_{k \rightarrow \infty} \pi_k(a | s) = 1(a = \arg \max_{a' \in A} Q_k(s, a'))$$

举个例子：如果 ϵ 满足 $\epsilon_k = \frac{1}{k}$ ϵ -贪婪策略就是满足GLIE的

GLIE蒙特卡洛控制

基于GLIE的蒙特卡洛控制流程如下：

1、对于给定策略 π ，采样第 k 个Episode: $\{S_1, A_1, R_2, \dots, S_T\} \sim \pi$

2、对于该Episode里出现的每一个状态行为对 S_t 和 A_t ,更其计数和 Q 函数

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

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

3、基于新的 Q 函数改善以如右方式改善策略：

$$\epsilon \leftarrow 1/k$$

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

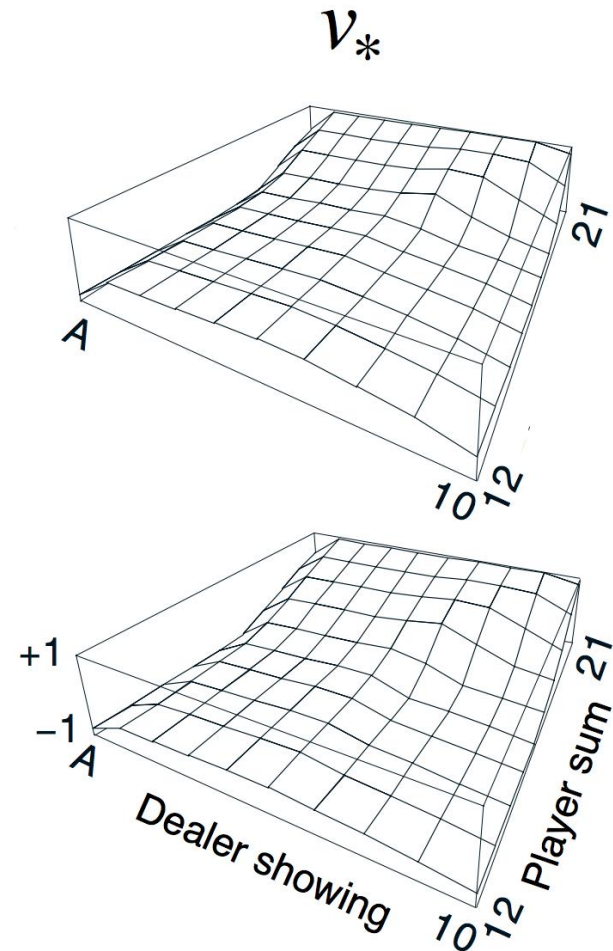
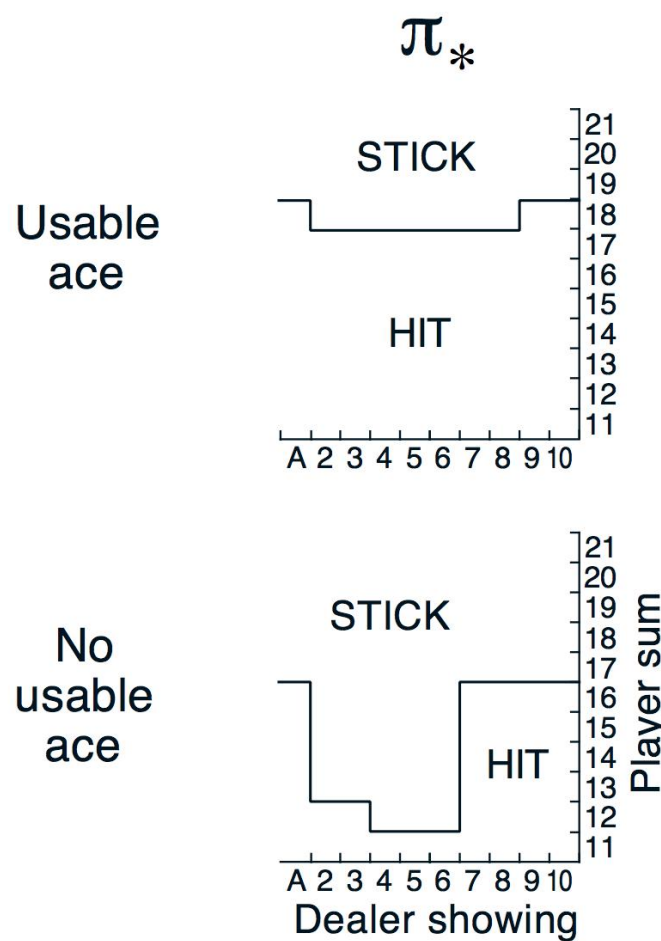
定理：GLIE蒙特卡洛控制收敛于最优动作价值函数，即 $Q(s, a) \rightarrow q_*(s, a)$

回到Blackjack例子



在blackjack问题中使用蒙特卡洛控制

- 当你手上有可用 A 时，大多数情况下当你的牌面和达到**17**或**18**时停止要牌，如果庄家可见的牌面在**2-9**之间，你选择**17**，其它条件选择**18**；
- 当你手上没有 A 时，最优策略提示大多数情况下牌面和达到**16**就要停止叫牌，当庄家可见的牌面在**2-7**时，这一数字更小至**13**甚至**12**。这种极端情况下，宁愿停止叫牌等待让庄家的牌爆掉。



MC控制 vs. TD 控制

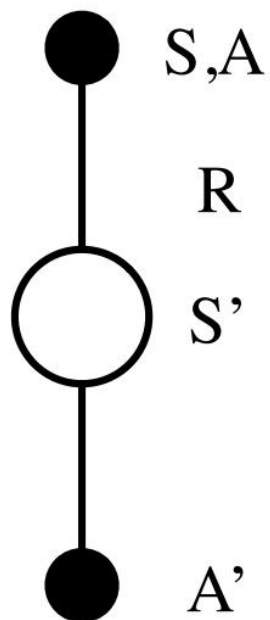
时序差分学习与蒙特卡洛学习相比有如下几个优势：

- 1、低方差
- 2、在线学习
- 3、可以从不完整的episode中学习

自然的想法：使用TD替代MC来进行控制，即：

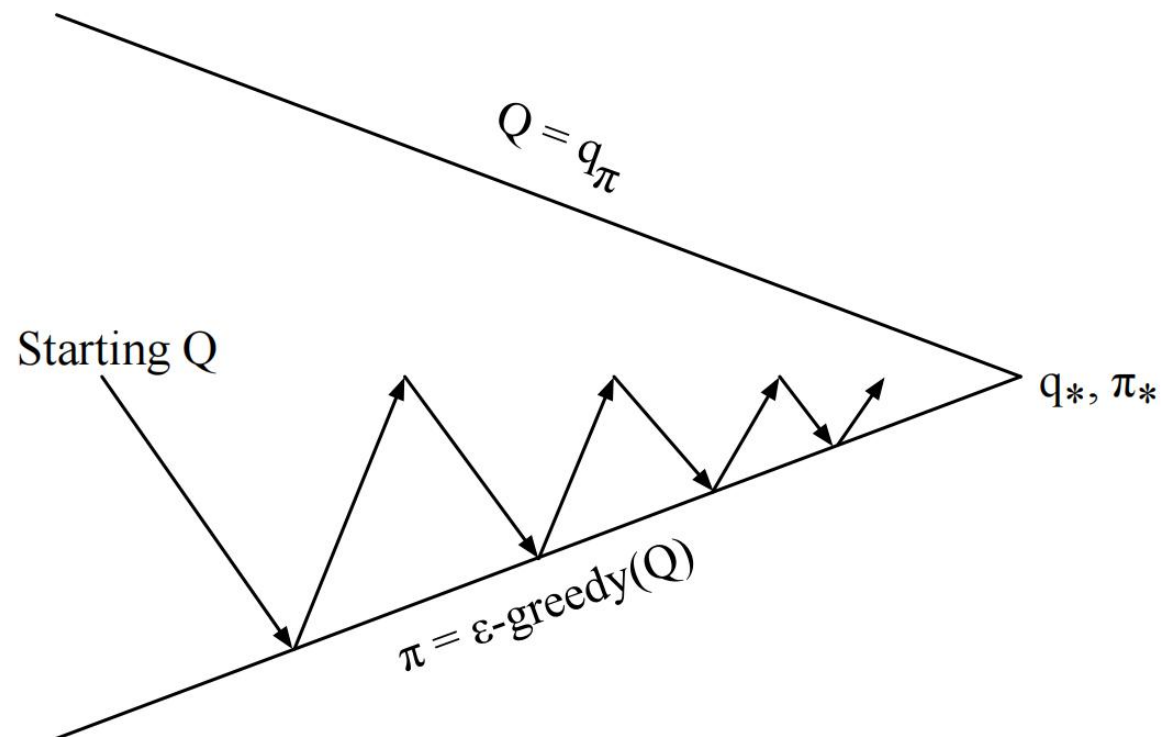
- 1、应用TD来更新 $Q(s,a)$
- 2、使用 ϵ -贪婪策略进行改进
- 3、每个时间步都进行更新

使用SARSA算法更新动作价值函数



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

基于SARSA算法的同策控制



对于每一个时间步 (time-step)

策略评估: **SARSA**, $Q \approx q_\pi$

策略改进: 使用 ϵ -贪婪策略改进

基于SARSA算法的在线控制

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

SARSA算法的收敛

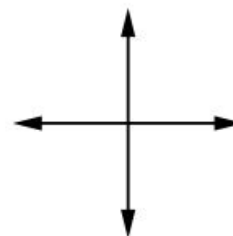
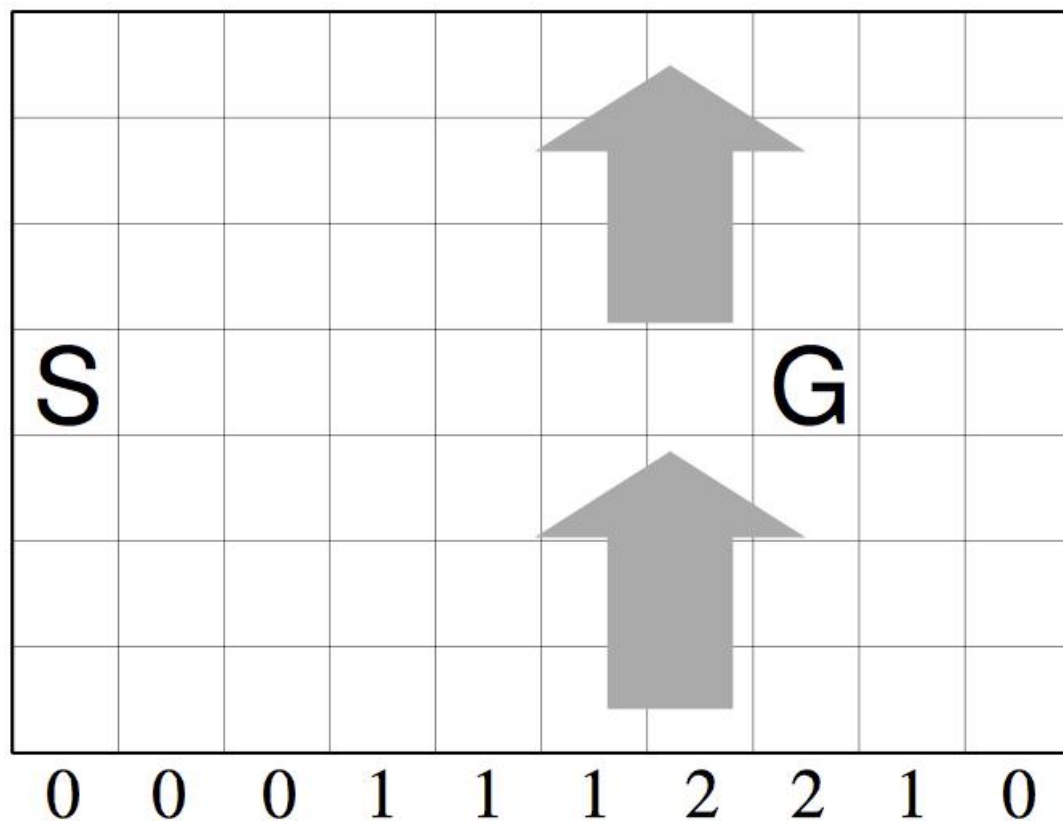
定理

SARSA在满足以下条件时收敛到最优动作价值函数 $Q(s, a) \rightarrow q_*(s, a)$

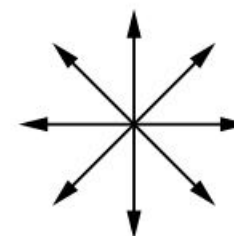
条件一：任何时候的策略 $\pi_t(a | s)$ 符合**GLIE**特性；

条件二：步长系数 α_t 满足： $\sum_{t=1}^{\infty} \alpha_t = \infty$ 且 $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$

示例——有风格子世界



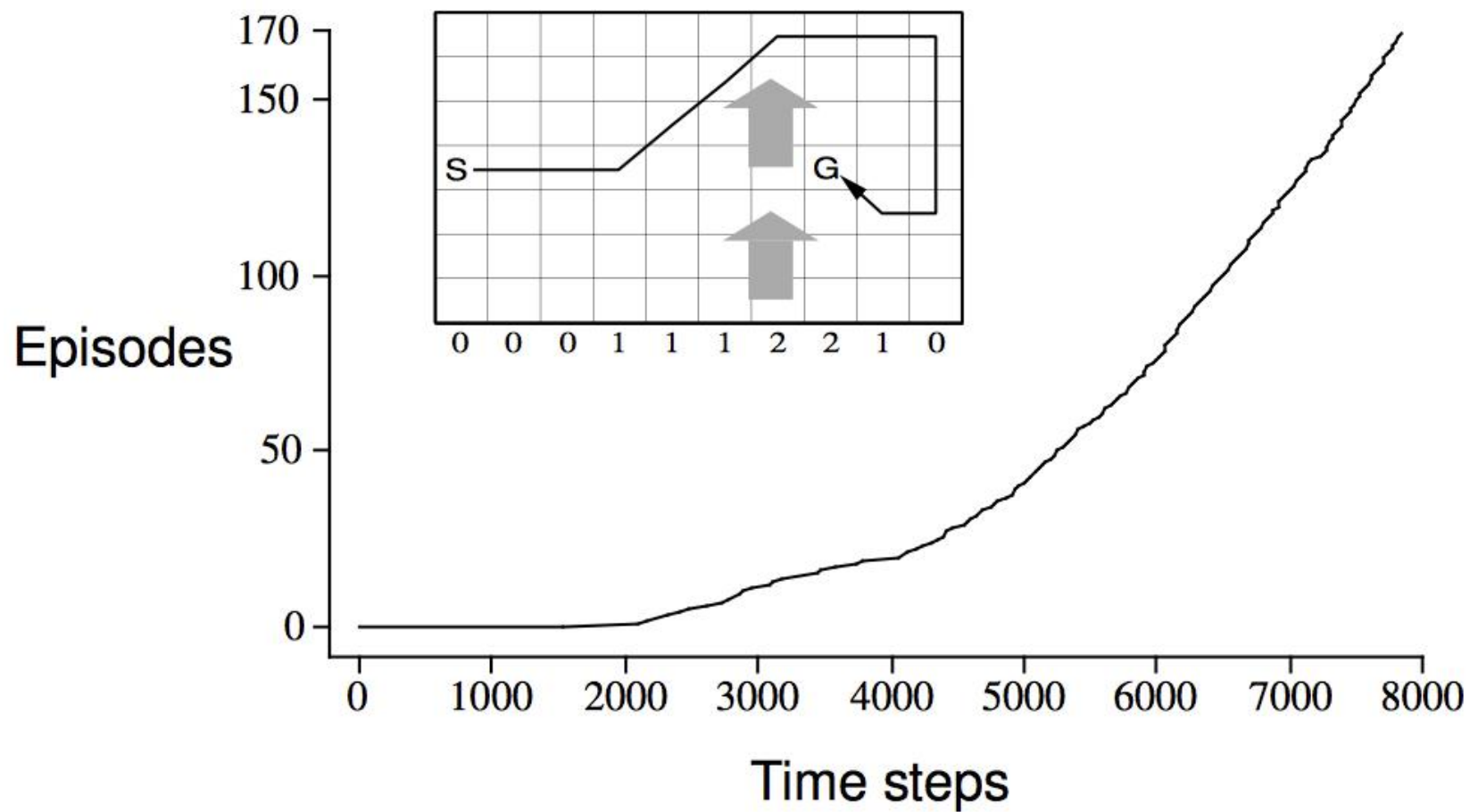
standard
moves



king's
moves

在到达终点之前每走一步reward=-1
无折扣率

SARSA算法在有风格子世界中的应用



n-Step Sarsa

- 下面是当 $n=0,1,2...\infty$ 时的 n 步回报

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

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

\vdots

\vdots

$$n = \infty \quad (\text{MC}) \quad q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

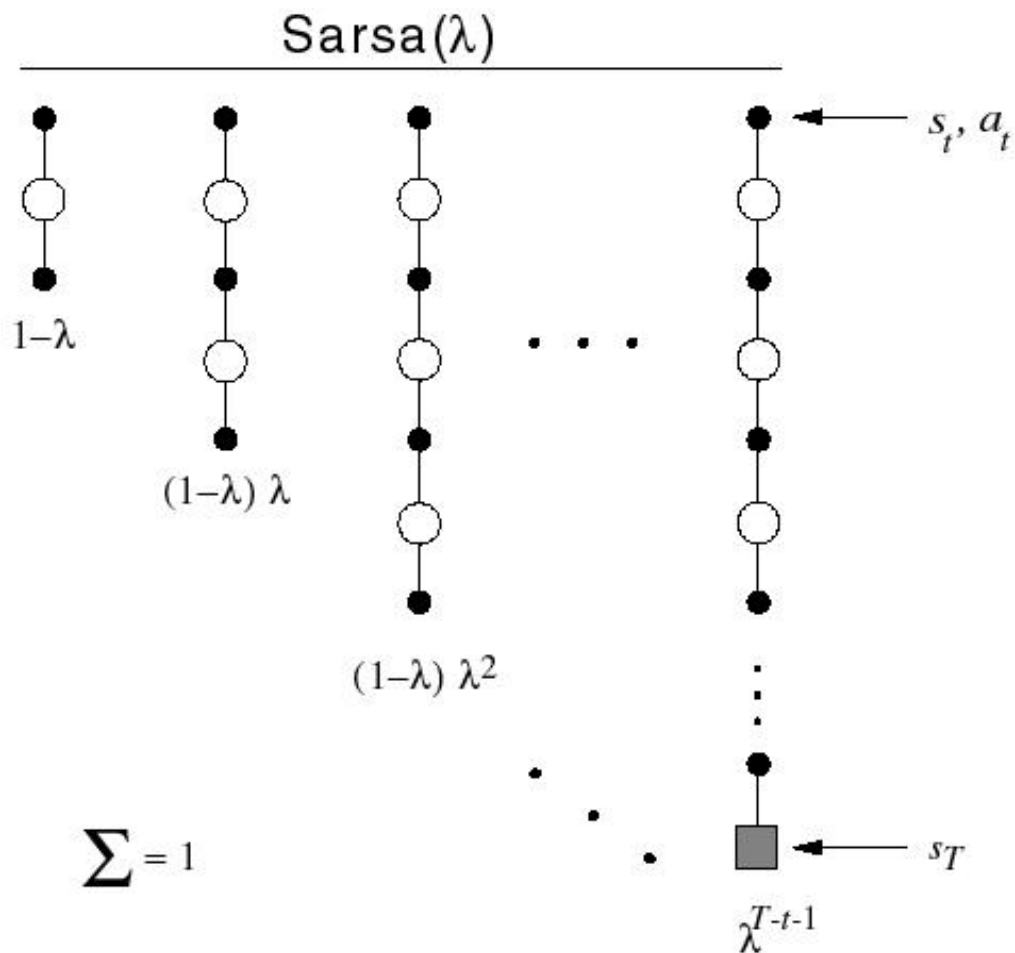
- 定义 n 步Q回报为:

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

- 使用 n 步Q回报来更新 $Q(s,a)$

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

前向视角SARSA(λ)



- q^λ 回报包含了所有的n步Q回报 $q_t^{(n)}$

- 使用权重 $(1-\lambda)\lambda^{n-1}$ 使

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

- 前向视角**SARSA(λ)**更新函数为:

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

后向视角SARSA(λ)

- 就像TD(λ)算法一样，为在线更新算法引入资格迹
- SARSA算法对于每个状态-动作对来说有一个资格迹

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

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

- $Q(s,a)$ 更新于每个状态 s 和动作 a 之后
- 更新量与TD误差 δ_t 与资格迹 $E_t(s, a)$ 成正比

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

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

SARSR(λ)算法流程

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

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

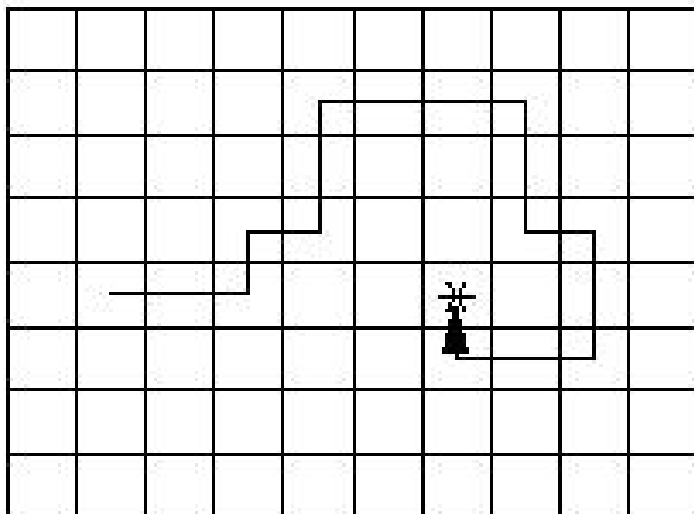
$E(s, a) \leftarrow \gamma \lambda E(s, a)$

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

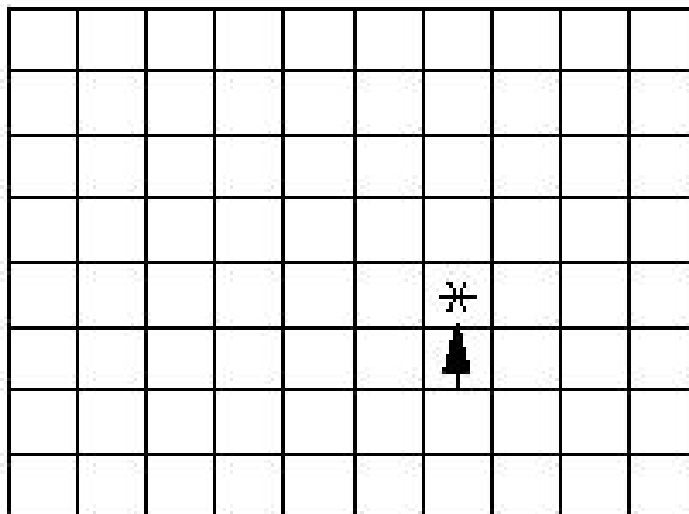
until S is terminal

SARSA算法在有风格子世界中的应用

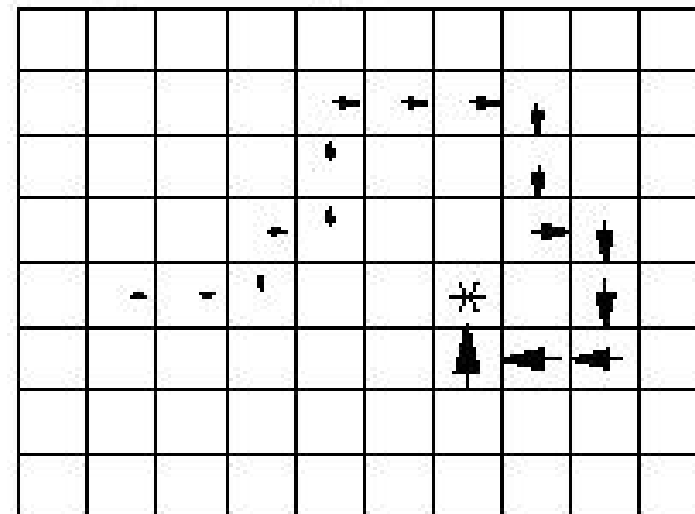
Path taken



Action values increased
by one-step Sarsa



Action values increased
by Sarsa(λ) with $\lambda=0.9$



异策学习

评估目标策略 $\pi(a|s)$ 以计算 $v_\pi(s)$ 或 $q_\pi(s, a)$ 时遵循策略 $\mu(a|s)$

$$\{S_1, A_1, R_2, \dots, S_T\} \sim \mu$$

为什么异策学习方法很重要？

- 可以从人类或其他的智能体经验中学习
- 重用旧策略的经验
- 在使用一个探索性策略的同时学习一个确定性策略
- 使用一个策略采样，同时学习多个策略

重要性抽样

估计不同分布的期望值：

$$\begin{aligned}\mathbb{E}_{X \sim P}[f(X)] &= \sum P(X)f(X) \\ &= \sum Q(X) \frac{P(X)}{Q(X)} f(X) \\ &= \mathbb{E}_{X \sim Q} \left[\frac{P(X)}{Q(X)} f(X) \right]\end{aligned}$$

异策蒙特卡洛方法的重要性抽样

- 使用由 μ 产生的回报值来评估 π
- 根据策略之间的相似性对 G_t 进行加权
- 沿着整个episode对重要性抽样的权重进行连乘

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

- 根据修正过的返回值来更新价值函数

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

- 当 π 为非零时，且 μ 为零，则无法使用
- 重要性抽样会显著增加方差

异策TD方法的重要性抽样

- 使用由 μ 产生的TD目标值来评估 π
- 通过重要性抽样对TD目标 $R+\gamma V(S')$ 进行加权
- 只需要一次重要抽样校正

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

- 方差比蒙特卡罗重要性抽样低得多
- 策略只需要在一个时间步上相似

智能体处在状态 S_t 中，基于策略 μ 产生了一个行为 A_t ，执行该行为后进入新的状态 S_{t+1} ，那么在当前策略下如何根据新状态的价值调整原来状态的价值呢？**异策学习的方法就是：在状态 S_t 时比较目标策略 π 和采样策略 μ 产生行为 A_t 的概率大小**，如果策略 π 得到的概率值与当前策略 μ 得到的概率值接近，说明根据状态 S_{t+1} 价值来更新 S_t 的价值同时得到两个策略的支持，这一更新操作比较有说服力。同时也说明在状态 S_t 时，两个策略有接近的概率选择行为 A_t 。假如这一概率比值很小，则表明如果依照评估策略，选择 A_t 的机会很小，这时候我们在更新 S_t 价值的时候就不能过多的考虑基于采样策略得到的状态 S_{t+1} 的价值。同样概率比值大于1时的道理也类似。这就相当于借鉴被评估策略的经验来更新我们自己的策略。

Q学习 (Q-Learning)

- 现在我们考虑动作价值函数 $Q(S,A)$ 的异策学习。
- Q学习不需要重要性抽样 (why?)
- 根据采样策略 μ 选择的下一个动作 $A_{t+1} \sim \mu(\cdot | S_t)$
- 但是我们考虑替代后继动作 $A' \sim \pi(\cdot | S_t)$
- 并根据替代动作的价值更新 $Q(S_t, A_t)$

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

Q学习的异策控制

- 我们现在允许同时改善采样策略 μ 和目标策略 π

- 目标策略 π 设为对应于 $Q(s,a)$ 的贪婪策略

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

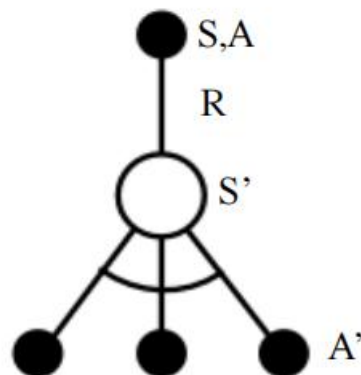
- 采样策略 μ 设为对应于 $Q(s,a)$ 的 ϵ -贪婪策略

- Q-learning目标简化为:

$$\begin{aligned} & R_{t+1} + \gamma Q(S_{t+1}, A') \\ &= R_{t+1} + \gamma Q(S_{t+1}, \operatorname{argmax}_{a'} Q(S_{t+1}, a')) \\ &= R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a') \end{aligned}$$

重要性采样的因子 π/μ 分两种情况。对于某个状态 s , π 策略选择动作 a 的概率要么是1, 要么是0。当为1时, μ 选择该动作的概率也较大 (因为 ϵ 一般比较小), 所以 $\pi/\mu \approx 1$; 当为0时 $\pi/\mu = 0$ 。

Q学习控制算法



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

定理

Q学习控制算法收敛到最优动作价值函数 $Q(s, a) \rightarrow q_*(s, a)$

异策控制的Q学习算法

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'

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

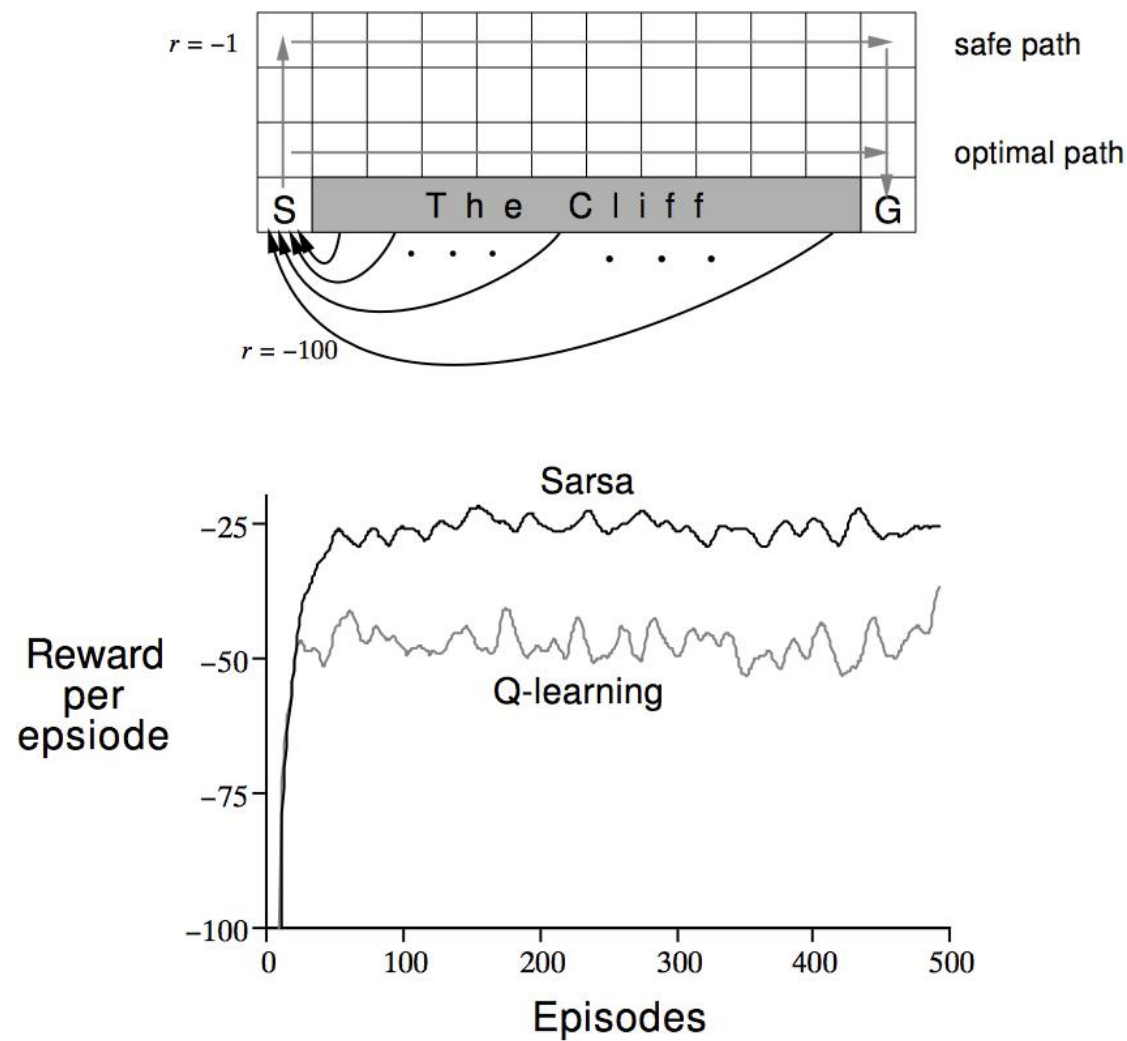
$S \leftarrow S'$;

until S is terminal

此时选择的动作只会参与价值函数的更新，不会真正的执行。价值函数更新后，新的执行动作需要基于状态 S' ，用 ϵ -贪婪法重新选择得到。

悬崖行走示例

右图显示的是在使用 ϵ -贪心方法 ($\epsilon=0.1$) 来选择动作时, **Sarsa**和**Q学习**方法的表现。训练一小段时间后, **Q学习**学到了最优策略, 即沿着悬崖边上走的策略。不幸的是, 由于动作是通过 ϵ -贪心的方式选择的, 因此在执行这个策略时, 智能体会偶尔掉入悬崖。与之对比, **Sarsa**则考虑了动作被选取的方式, 学到了一条通过网格的上半部分的路径, 这条路径虽然更长但更安全。虽然**Q学习**实际上学到了最优策略的价值, 其在线性能却比学到迂回策略的**Sarsa**更差。当然如果 ϵ 逐步减小, 那么两种方法都会渐进地收敛到最优策略。

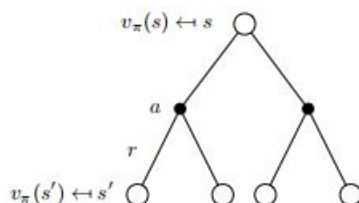

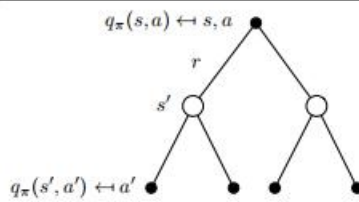

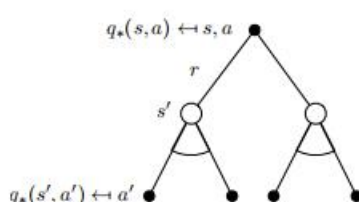
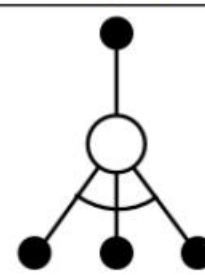


SARSA v.s. Q-Learning

Q-learning 直接学习的是最优策略，而 **SARSA** 在学习最优策略的同时还在做探索。这导致我们在学习最优策略的时候，如果用 **SARSA**，为了保证收敛，需要制定一个策略，使 ϵ — greedy 法的超参数 ϵ 在迭代的过程中逐渐变小。**Q-learning** 没有这个烦恼。另外一个就是 **Q-learning** 直接学习最优策略，但是最优策略会依赖于训练中产生的一系列数据，所以受样本数据的影响较大，因此受到训练数据方差的影响很大，甚至会影响 Q 函数的收敛。

Q-learning 的深度强化学习版 **Deep Q-Learning** 也有这个问题。在学习过程中，**SARSA** 在收敛的过程中鼓励探索，这样学习过程会比较平滑，不至于过于激进，导致出现像 **Q-learning** 可能遇到一些特殊的最优“陷阱”。在实际应用中，如果我们是在模拟环境中训练强化学习模型，推荐使用 **Q-learning**，如果是在线生产环境中训练模型，则推荐使用 **SARSA**。

DP与TD

	<i>Full Backup (DP)</i>	<i>Sample Backup (TD)</i>
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>

DP与TD (2)

<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 $x \stackrel{\alpha}{\leftarrow} y \equiv x \leftarrow x + \alpha(y - x)$

The End