无模型的强化学习

学习一个MDP模型

Episode1：
$$s_0^{(1)} \xrightarrow{a_0^{(1)},R(s_0)^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)},R(s_1)^{(1)}} s_2^{(1)} \xrightarrow{a_2^{(1)},R(s_2)^{(1)}} s_3^{(1)} \cdots s_T^{(1)}$$

Episode2：
$$s_0^{(2)} \xrightarrow{a_0^{(2)},R(s_0)^{(2)}} s_1^{(2)} \xrightarrow{a_1^{(2)},R(s_1)^{(2)}} s_2^{(2)} \xrightarrow{a_2^{(2)},R(s_2)^{(2)}} s_3^{(2)} \cdots s_T^{(2)}$$

⋮ ⋮

- 从“经验”中学习一个MDP模型
  - 学习状态转移概率$P_{sa}(s')$

$$P_{sa}(s') = \frac{\text{在}s\text{下采取动作}a\text{并转移到}s'\text{的次数}}{\text{在}s\text{下采取动作}a\text{的次数}}$$

  - 学习奖励函数$R(s)$，也就是立即奖赏期望

$$R(s) = average\{R(s)^{(i)}\}$$

无模型的强化学习（Model-free RL）

- 在现实问题中，通常没有明确地给出状态转移和奖励函数
  - 例如，我们仅能观察到部分片段（episodes）

Episode 1: $s_0^{(1)} \xrightarrow[R(s_0)^{(1)}]{a_0^{(1)}} s_1^{(1)} \xrightarrow[R(s_1)^{(1)}]{a_1^{(1)}} s_2^{(1)} \xrightarrow[R(s_2)^{(1)}]{a_2^{(1)}} s_3^{(1)} \ldots s_T^{(1)}$

Episode 2: $s_0^{(2)} \xrightarrow[R(s_0)^{(2)}]{a_0^{(2)}} s_1^{(2)} \xrightarrow[R(s_1)^{(2)}]{a_1^{(2)}} s_2^{(2)} \xrightarrow[R(s_2)^{(2)}]{a_2^{(2)}} s_3^{(2)} \ldots s_T^{(2)}$

- 模型无关的强化学习直接从经验中学习值（value）和策略（policy），而无需构建马尔可夫决策过程模型（MDP）
- 关键步骤：（1）估计值函数；（2）优化策略

值函数估计

- 在基于模型的强化学习（MDP）中，值函数能够通过动态规划计算获得

$$V^{\pi}(s) = \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi]$$
$$= R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')$$

- 在模型无关的强化学习中
  - 我们无法直接获得 P_{sa} 和 R
  - 但是，我们拥有一系列可以用来估计值函数的经验

Episode 1: $s_0^{(1)} \xrightarrow[R(s_0)^{(1)}]{a_0^{(1)}} s_1^{(1)} \xrightarrow[R(s_1)^{(1)}]{a_1^{(1)}} s_2^{(1)} \xrightarrow[R(s_2)^{(1)}]{a_2^{(1)}} s_3^{(1)} \ldots s_T^{(1)}$

Episode 2: $s_0^{(2)} \xrightarrow[R(s_0)^{(2)}]{a_0^{(2)}} s_1^{(2)} \xrightarrow[R(s_1)^{(2)}]{a_1^{(2)}} s_2^{(2)} \xrightarrow[R(s_2)^{(2)}]{a_2^{(2)}} s_3^{(2)} \ldots s_T^{(2)}$

蒙特卡洛价值预测

蒙特卡洛价值估计

- 目标：从策略 π下的经验片段学习 V^π

$$s_0^{(i)} \xrightarrow[R_1^{(i)}]{a_0^{(i)}} s_1^{(i)} \xrightarrow[R_2^{(i)}]{a_1^{(i)}} s_2^{(i)} \xrightarrow[R_3^{(i)}]{a_2^{(i)}} s_3^{(i)} \ldots s_T^{(i)} \sim \pi$$

- 回顾：累计奖励（return）是总折扣奖励

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots \gamma^{T-1} R_T$$

- 回顾：值函数（value function）是期望累计奖励

$$\begin{aligned} V^\pi(s) &= \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi] \\ &= \mathbb{E}[G_t | s_t = s, \pi] \\ &\simeq \frac{1}{N} \sum_{i=1}^{N} G_t^{(i)} \end{aligned}$$

- 使用策略π从状态s采样N个片段
- 计算平均累计奖励

- 蒙特卡洛策略评估使用经验均值累计奖励而不是期望累计奖励

蒙特卡洛值估计

思路： $$V(S_t) \simeq \frac{1}{N}\sum_{i=1}^{N} G_t^{(i)}$$

实现： $$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

- 蒙特卡洛方法：直接从经验片段进行学习
- 蒙特卡洛是模型无关的：未知马尔可夫决策过程的状态转移/奖励
- 蒙特卡洛从完整的片段中进行学习：没有使用bootstrapping的方法
- 蒙特卡洛采用最简单的思想：值（value）= 平均累计奖励（mean return）
- 注意：只能将蒙特卡洛方法应用于有限长度的马尔可夫决策过程中
  - 即，所有的片段都有终止状态

重要性采样

- 估计一个不同分布的期望

$$
\begin{aligned}
\mathbb{E}_{x\sim p}[f(x)] &= \int_x p(x)f(x)dx \\
&= \int_x q(x)\frac{p(x)}{q(x)}f(x)dx \\
&= \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right]
\end{aligned}
$$

- 将每个实例的权重重新分配为 $\beta(x) = \frac{p(x)}{q(x)}$

使用重要性采样的离线策略蒙特卡洛

- 使用策略μ产生的累计奖励评估策略π
- 根据两个策略之间的重要性比率（importance ratio）对累计奖励G_t加权
- 每个片段乘以重要性比率

$$\{s_1, a_1, r_2, s_2, a_2, \ldots, s_T\} \sim \mu$$

$$G_t^{\pi/\mu} = \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)} \frac{\pi(a_{t+1}|s_{t+1})}{\mu(a_{t+1}|s_{t+1})} \cdots \frac{\pi(a_T|s_T)}{\mu(a_T|s_T)} G_t$$

时序差分学习（Temporal Difference Learning）

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = R_{t+1} + \gamma V(S_{t+1})$$

$$V(S_t) \leftarrow V(S_t) + \alpha\big(\underbrace{R_{t+1}}_{\text{观测值}} + \underbrace{\gamma V(S_{t+1})}_{\text{对未来的猜测}} - V(S_t)\big)$$

- 时序差分方法直接从经验片段中进行学习
- 时序差分是模型无关的
  - 不需要预先获取马尔可夫决策过程的状态转移/奖励
- 通过bootstrapping，时序差分从不完整的片段中学习
- 时序差分更新当前预测值使之接近估计累计奖励（非真实值）

蒙特卡洛 vs. 时序差分（MC vs. TD)

相同的目标：从策略π下的经验片段学习V^{π}

- 增量地进行每次蒙特卡洛过程（MC）
  - 更新值函数$V(S_t)$使之接近准确累计奖励G_t

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

- 最简单的时序差分学习算法（TD）：
  - 更新$V(S_t)$使之接近估计累计奖励$R_{t+1} + \gamma V(S_{t+1})$

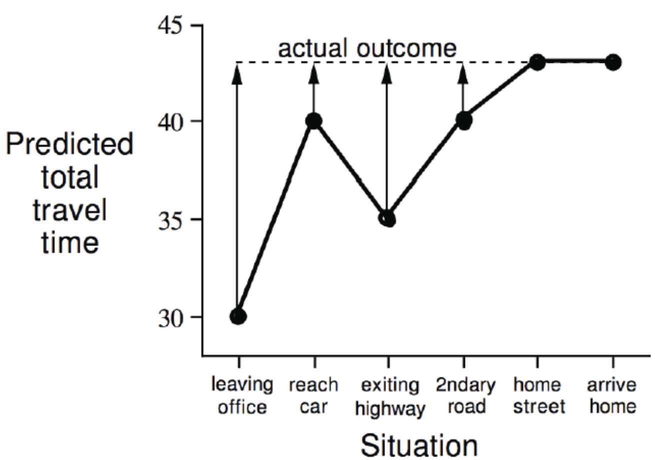
$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$

  - 时序差分目标：$R_{t+1} + \gamma V(S_{t+1})$
  - 时序差分误差：$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

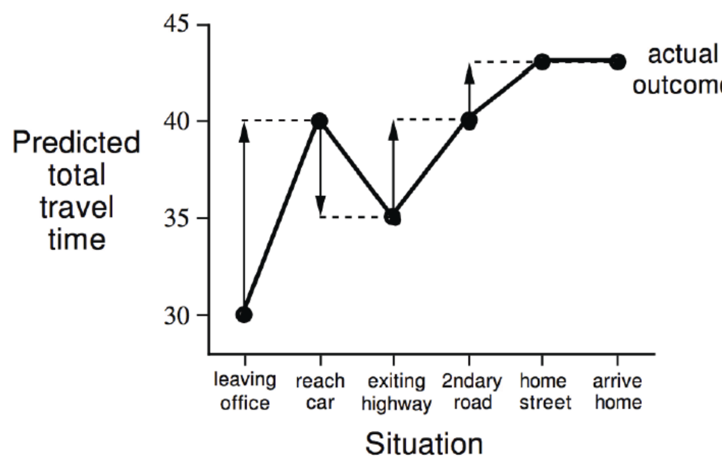
驾车回家的例子（MC vs. TD)

状态	经过的时间(分钟)	预计所剩时间	预计总时间
离开公司	0	30	30
开始驾车，下雨	5	35	40
离开高速公路	20	15	35
卡车后跟车	30	10	40
到达家所在街道	40	3	43
直奔家门	43	0	43

Changes recommended by Monte Carlo methods (α=1)



Changes recommended by TD methods (α=1)



蒙特卡洛（MC）和时序差分（TD）的优缺点

- 时序差分：能够在知道最后结果之前进行学习
  - 时序差分能够在每一步之后进行在线学习
  - 蒙特卡洛必须等待片段结束，直到累计奖励已知

- 时序差分：能够无需最后结果地进行学习
  - 时序差分能够从不完整的序列中学习
  - 蒙特卡洛只能从完整序列中学习
  - 时序差分在连续（无终止的）环境下工作
  - 蒙特卡洛只能在片段化的（有终止的）环境下工作

偏差（Bias）/方差（Variance）的权衡

- 累计奖励$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$是$V^{\pi}(S_t)$的无偏估计
- 时序差分真实目标$R_{t+1} + \gamma V^{\pi}(S_{t+1})$是$V^{\pi}(S_t)$的无偏估计
- 时序差分目标$R_{t+1} + \gamma \underbrace{V(S_{t+1})}_{\text{当前估计}}$是$V^{\pi}(S_t)$的有偏估计
- 时序差分目标具有比累计奖励更低的方差
  - 累计奖励——取决于多步随机动作，多步状态转移和多步奖励
  - 时序差分目标——取决于单步随机动作，单步状态转移和单步奖励

蒙特卡洛（MC）和时序差分（TD）的优缺点（2）

MC:

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

蒙特卡洛具有高方差，无偏差

- 良好的收敛性质
  - 使用函数近似时依然如此
- 对初始值不敏感
- 易于理解和使用

TD:

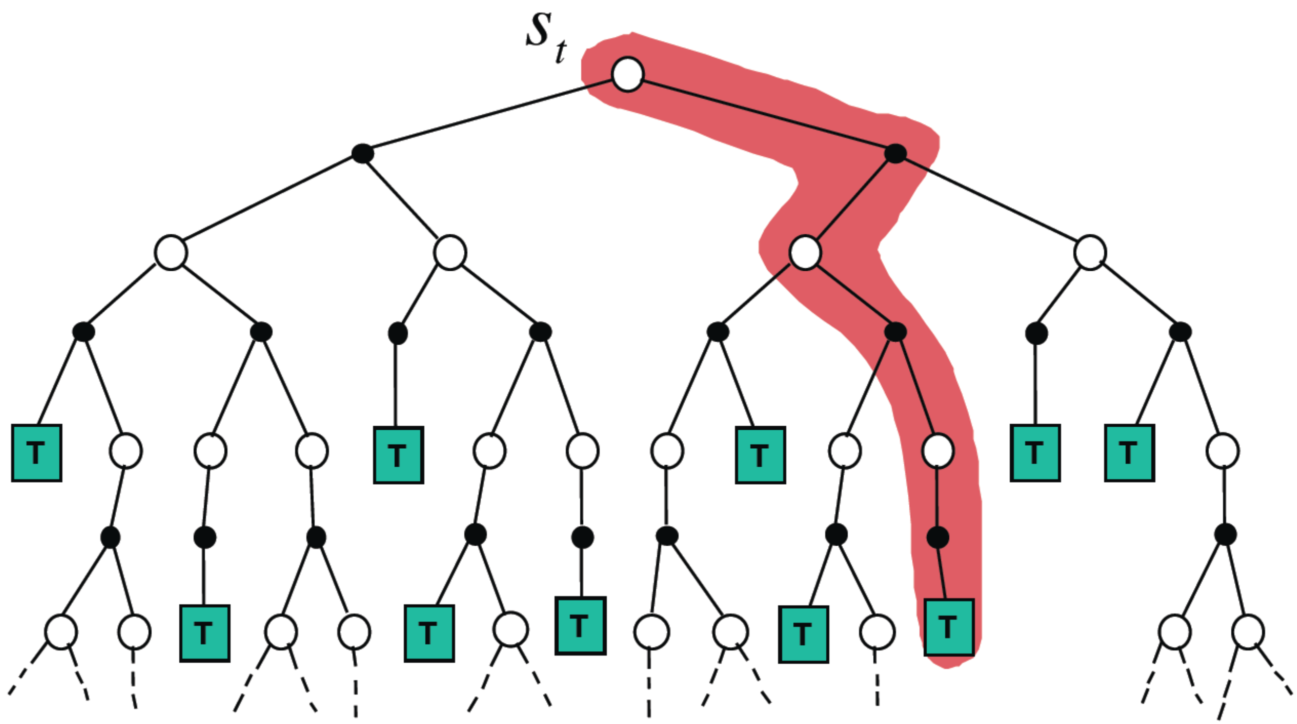
$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$

时序差分具有低方差，有偏差

- 通常比蒙特卡洛更加高效
- 时序差分最终收敛到$V^{\pi}(S_t)$
  - 但使用函数近似并不总是如此
- 比蒙特卡洛对初始值更加敏感

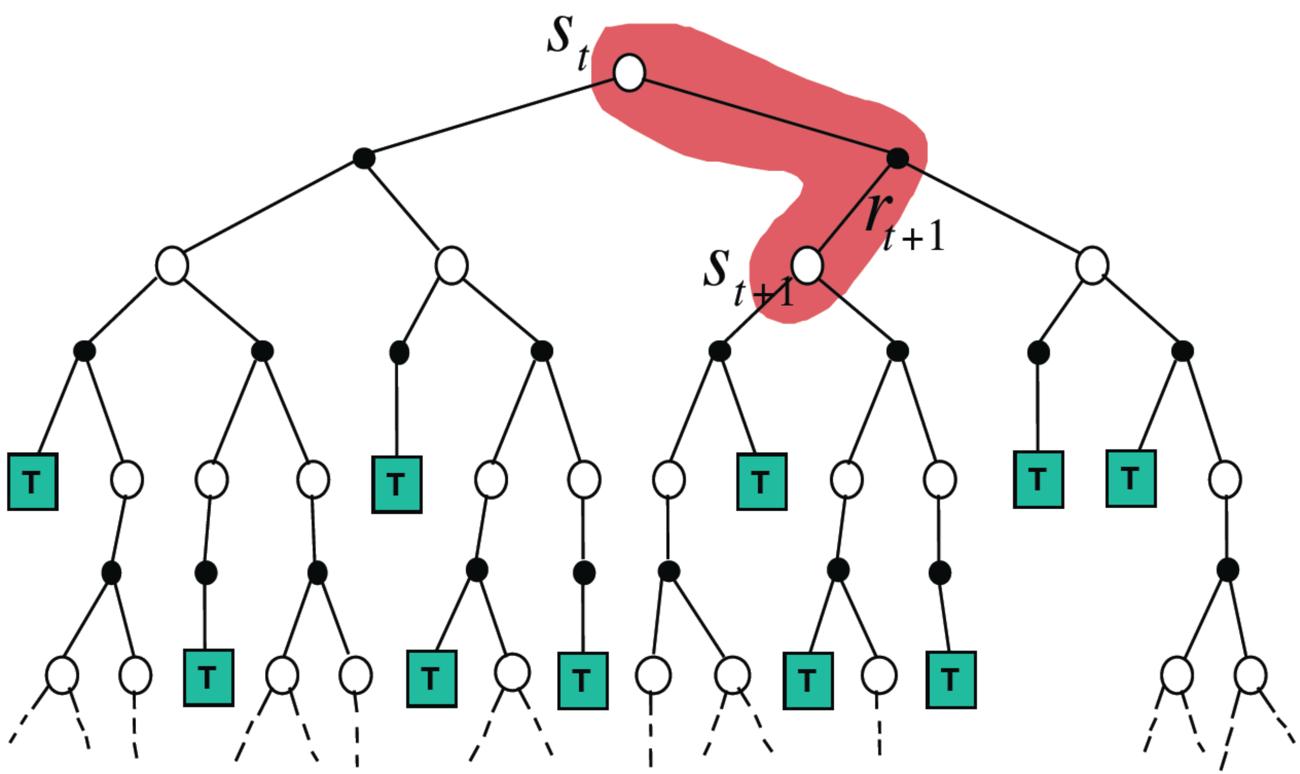
蒙特卡洛反向传播（Backup）

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$



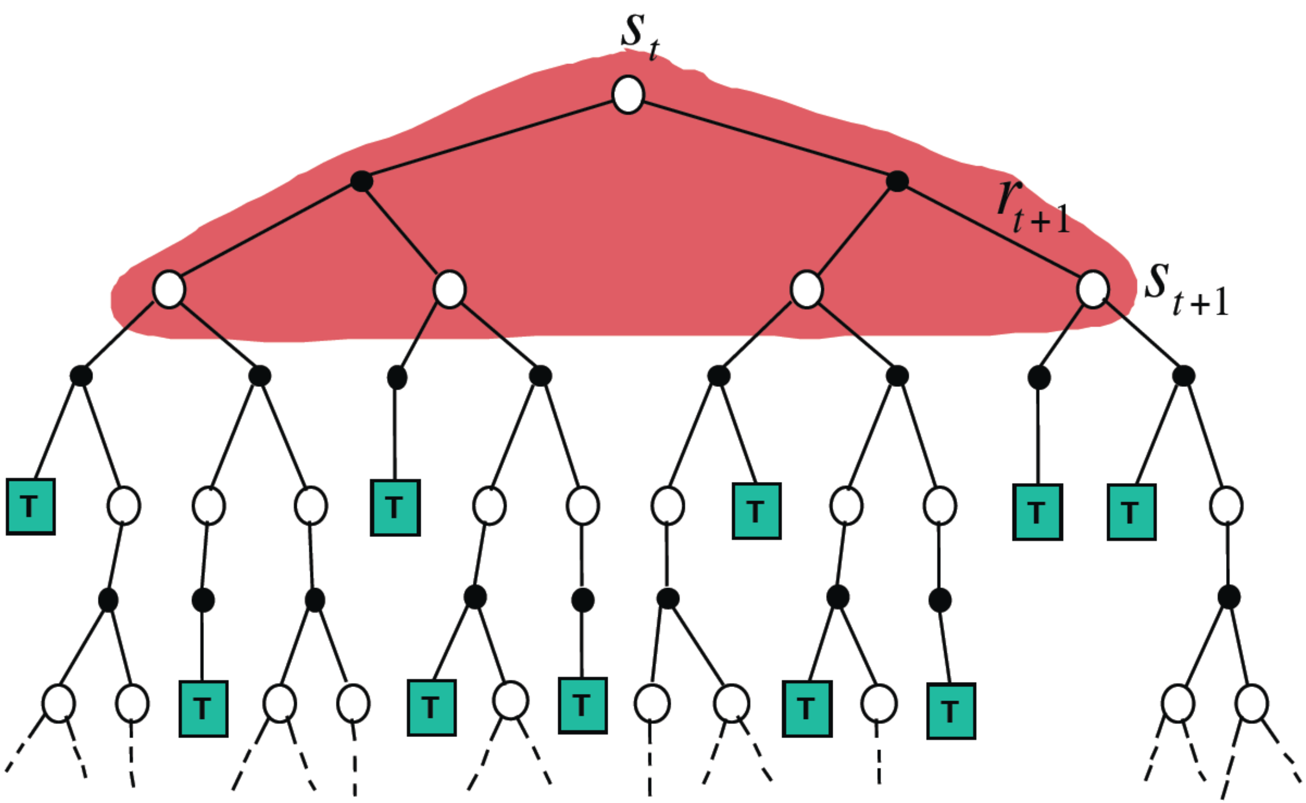
时序差分反向传播（Backup）

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$



动态规划反向传播（Backup）

$$V(S_t) \leftarrow \mathbb{E}[R_{t+1} + \gamma V(S_{t+1})]$$



多步时序差分学习

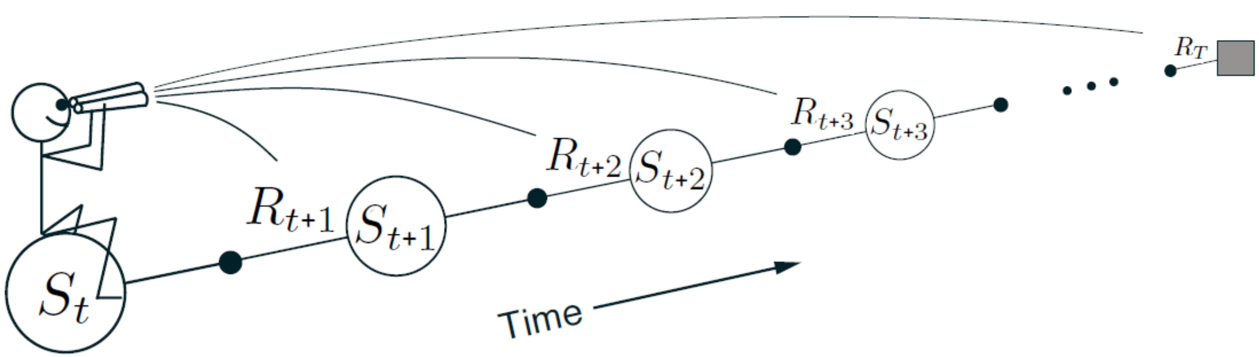
- 对于有时间约束的情况，我们可以跳过n步预测的部分，直接进入模型无关的控制

- 定义n步累计奖励

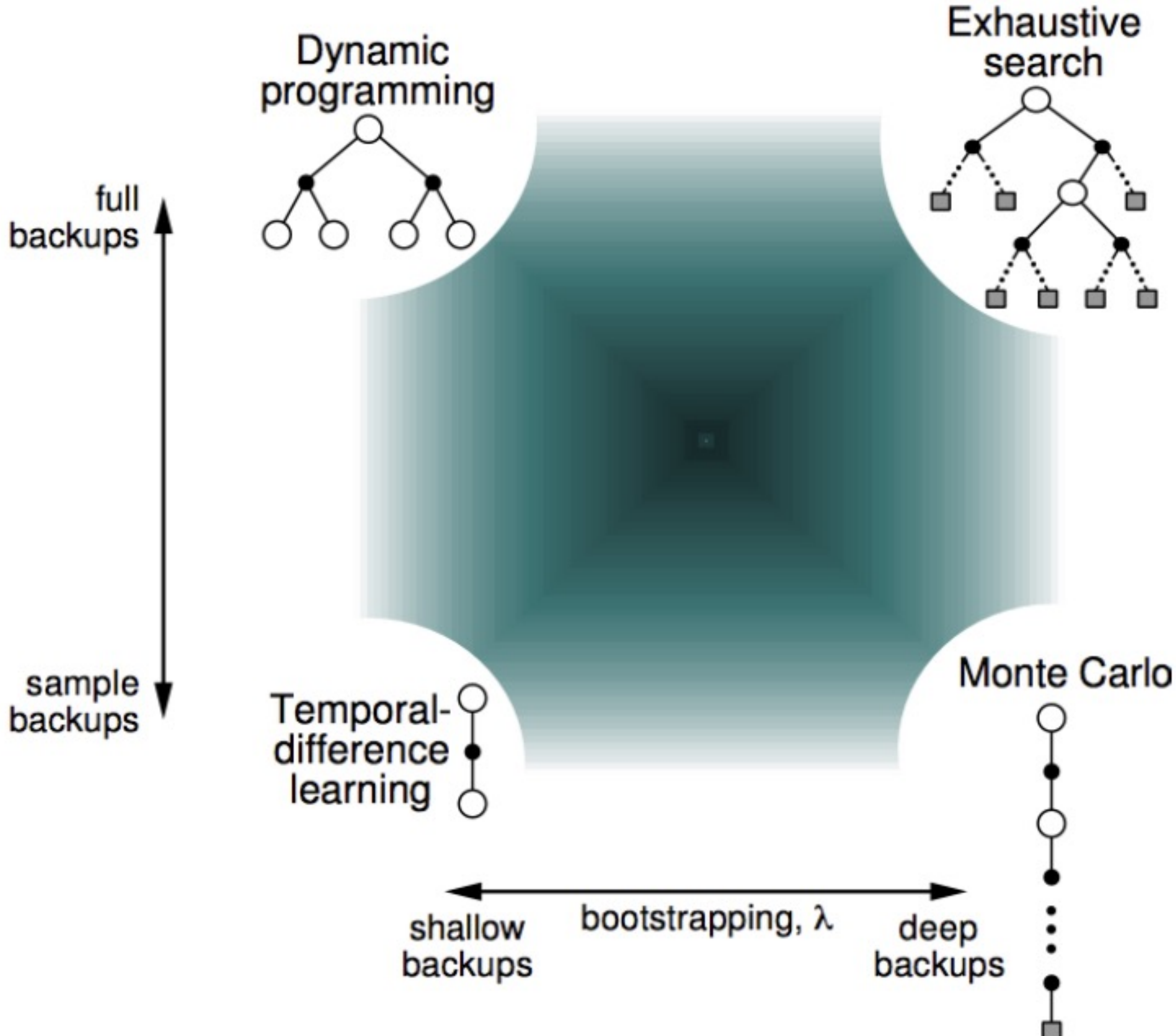
$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- n步时序差分学习

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



总览强化学习值函数估计多种方法



Slide credit: David Silver

值函数估计总结

- 无模型的强化学习在黑盒环境下使用
- 要优化智能体策略，首要任务则是精准、高效地估计状态或者(状态、动作)的价值
- 在黑盒环境下，值函数的估计方法主要包括蒙特卡洛方法和时序差分法
- 蒙特卡洛方法通过采样到底的方式直接估计价值函数
- 时序差分学习通过下一步的价值估计来更新当前一步的价值估计
- 实际使用中，时序差分方法更加常见

涉及知识点：
SARSA、Q学习算法及其收敛性、多步自助法

无模型控制方法

从知道什么是好的，到如何做好行动

- 从知道什么是好的：估计$V^{\pi}(S_t)$

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$

- 基于V函数，如何选择好的行动？

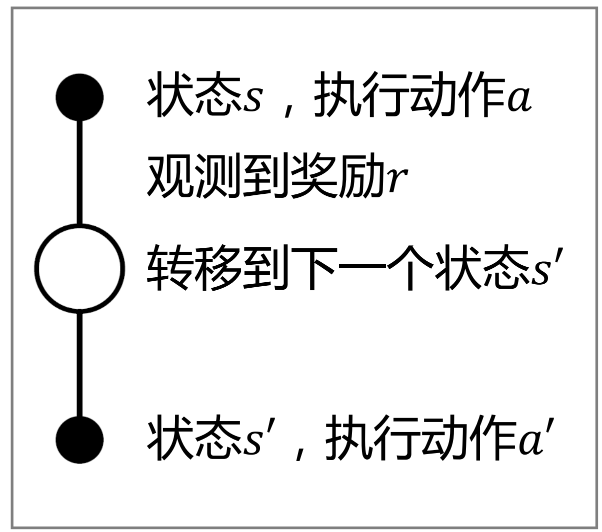
$$\pi(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s')V(s')$$

需要知道环境模型

- 基于Q函数，如何选择好的行动？

$$\pi(s) = \arg\max_{a \in A} Q(s,a)$$

  - 因此，估计Q函数对直接做行动（控制）有直接的作用



SARSA

SARSA

- 对于当前策略执行的每个（状态-动作-奖励-状态-动作）元组

  状态s，执行动作a

  观测到奖励r

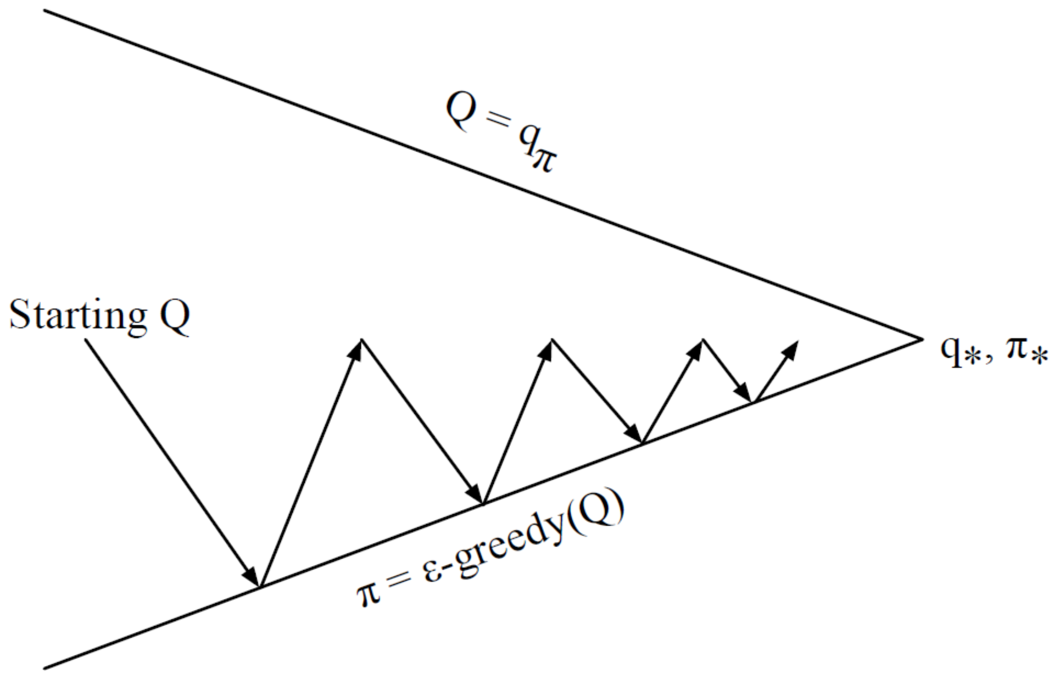
  转移到下一个状态s'

  状态s'，执行动作a'

- SARSA更新状态-动作值函数为

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

使用SARSA的在线策略控制



- 每个时间步长：
  - 策略评估：SARSA $Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$
  - 策略改进：ε-greedy策略改进

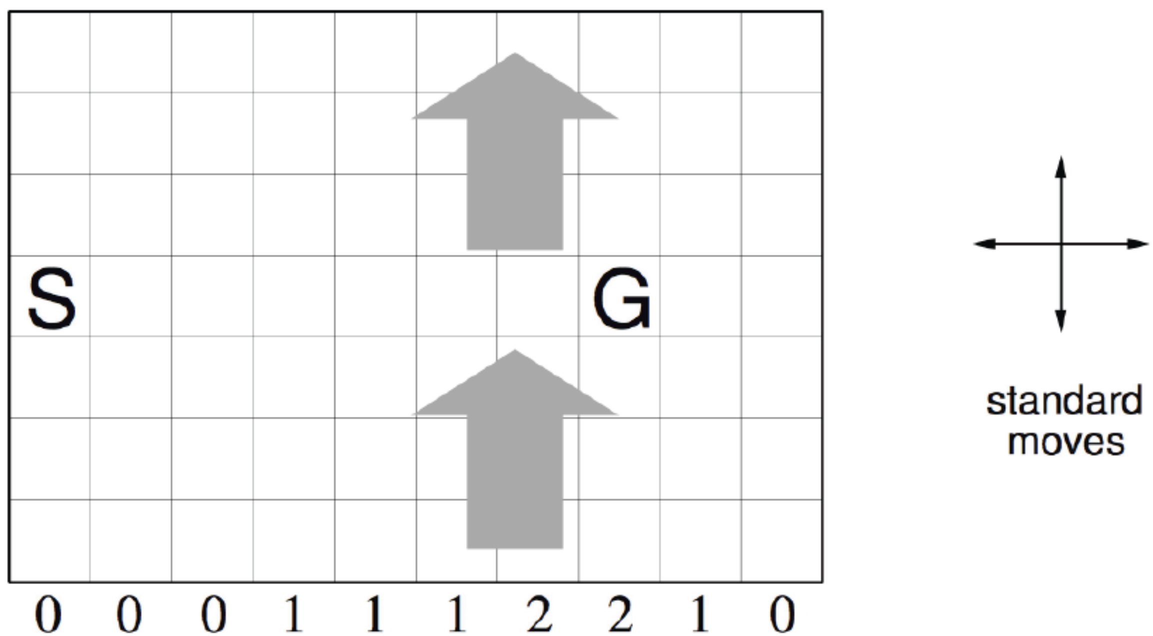
SARSA算法

Sarsa: An on-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
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)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

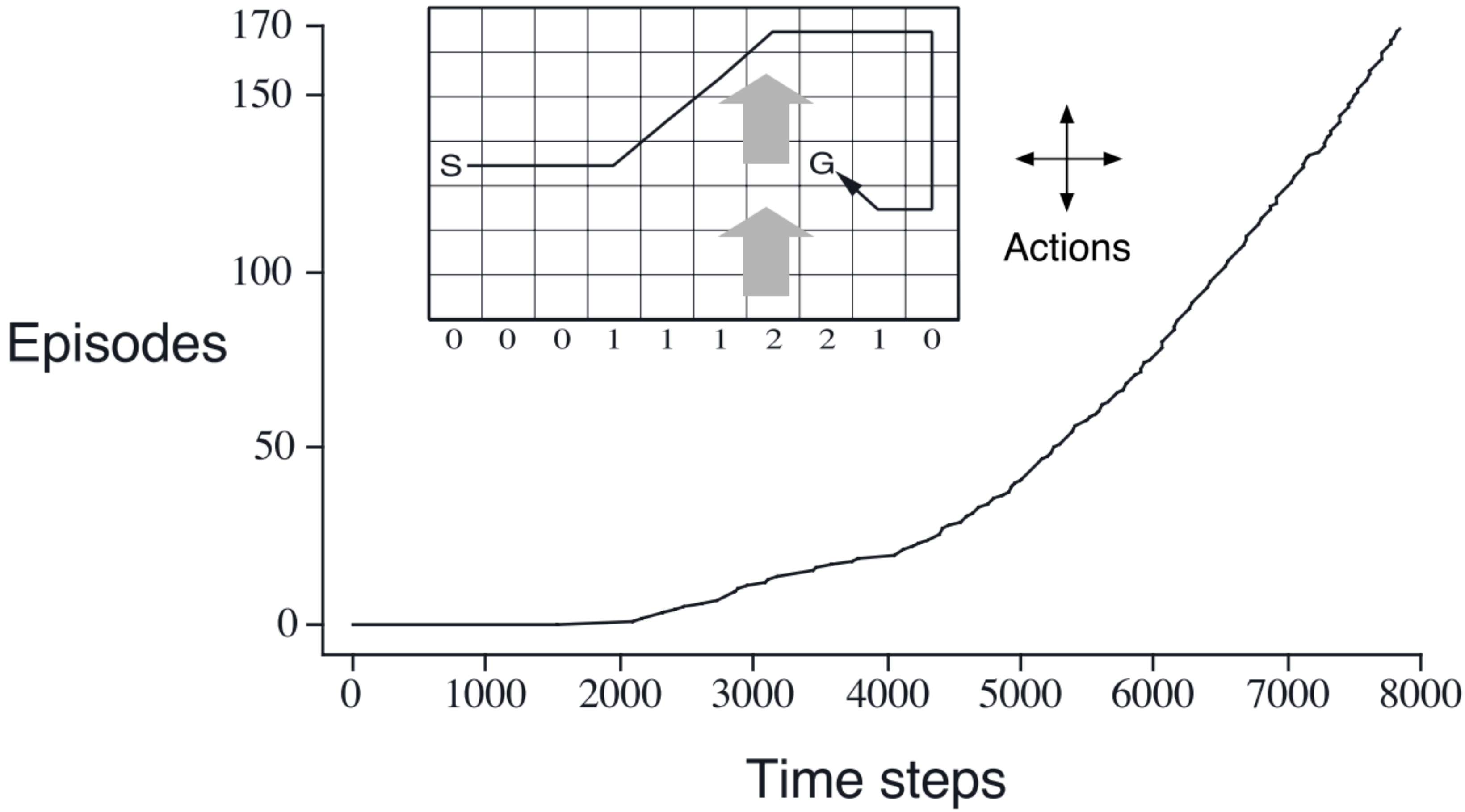
注：在线策略时序差分控制（on-policy TD control）使用当前策略进行动作采样。即，SARSA算法中的两个“A”都是由当前策略选择的

SARSA示例：Windy Gridworld



- 每步的奖励 = -1，直到智能体抵达目标网格
- 无折扣因子

SARSA示例：Windy Gridworld



注意：随着训练的进行，SARSA策略越来越快速地抵达目标

Q 学习算法

Q 学习

- 学习状态-动作值函数 $Q(s,a) \in \mathbb{R}$，不直接优化策略
- 一种离线策略（off-policy）学习方法

策略函数，一般是给定的策略，$\mu(\cdot|\, s_t,) \in \mathbb{R}^{|A|}$

$$Q(s_t,a_t) = \sum_{t=0}^{T} \gamma^t R(s_t,a_t), a_t \sim \mu(s_t)$$

动作空间，$a \sim A$

奖励函数，$R(s_t,a_t) \in \mathbb{R}$

迭代式：$Q(s_t,a_t) = R(s_t,a_t) + \gamma Q(s_{t+1},a_{t+1})$

离线策略学习

什么是离线策略学习

- 目标策略 $\pi(a|s)$ 进行值函数评估（$V^{\pi}(s)$或$Q^{\pi}(s,a)$）
- 行为策略 $\mu(a|s)$ 收集数据：$\{s_1, a_1, r_2, s_2, a_2, \dots, S_T\} \sim \mu$

为什么使用离线策略学习

- 平衡探索（exploration）和利用（exploitation）
- 通过观察人类或其他智能体学习策略
- 重用旧策略所产生的经验
- 遵循探索策略时学习最优策略
- 遵循一个策略时学习多个策略
- 在剑桥MSR研究时的一个例子
  - Collective Noise Contrastive Estimation for Policy Transfer Learning. AAAI 2016

Q 学习

- 无需重要性采样（为什么？）
- 根据行为策略选择动作 $a_t \sim \mu(\cdot\,|s_t)$
- 根据目标策略选择后续动作 $a'_{t+1} \sim \pi(\cdot\,|s_t)$
  - 目标$Q^\star(s_t, a_t) = r_t + \gamma Q(s_{t+1}, a'_{t+1})$
- 更新$Q(s_t, a_t)$的值以逼近目标状态-动作值

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \boxed{Q(s_{t+1}, a'_{t+1})} - Q(s_t, a_t)\right)$$

策略π的动作，而非策略μ

使用Q 学习的离线策略控制

- 允许行为策略和目标策略都进行改进
- 目标策略π是关于$Q(s,a)$的贪心策略

$$\pi(s_{t+1}) = \arg\max_{a'} Q(s_{t+1}, a')$$

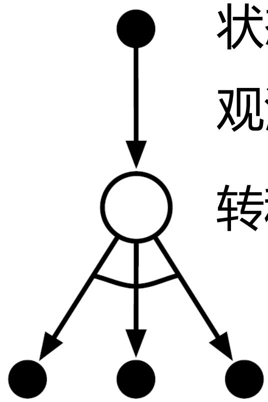
- 行为策略μ是关于$Q(s,a)$的ε-贪心策略
- Q-学习目标函数可以简化为

$$\begin{aligned} r_{t+1} + \gamma Q(s_{t+1}, a'_{t+1}) &= r_{t+1} + \gamma Q(s_{t+1}, \arg\max_{a'_{t+1}} Q(s_{t+1}, a'_{t+1})) \\ &= r_{t+1} + \gamma \max_{a'_{t+1}} Q(s_{t+1}, a'_{t+1}) \end{aligned}$$

- Q-学习更新方式

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'_{t+1}} Q(s_{t+1}, a'_{t+1}) - Q(s_t, a_t))$$

Q 学习控制算法



状态s，执行动作a

观测到奖励r

转移到下一状态s'

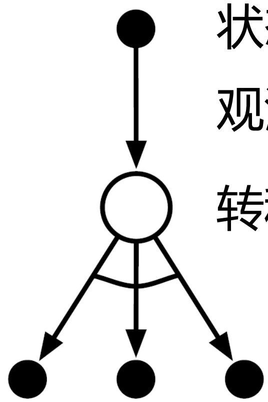
状态s'，执行动作 $\arg\max_{a'} Q(s', a')$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'_{t+1}} Q(s_{t+1}, a'_{t+1}) - Q(s_t, a_t))$$

- 定理：Q-学习控制收敛到最优状态-动作值函数

$$Q(s, a) \rightarrow Q^*(s, a)$$

Q 学习控制算法



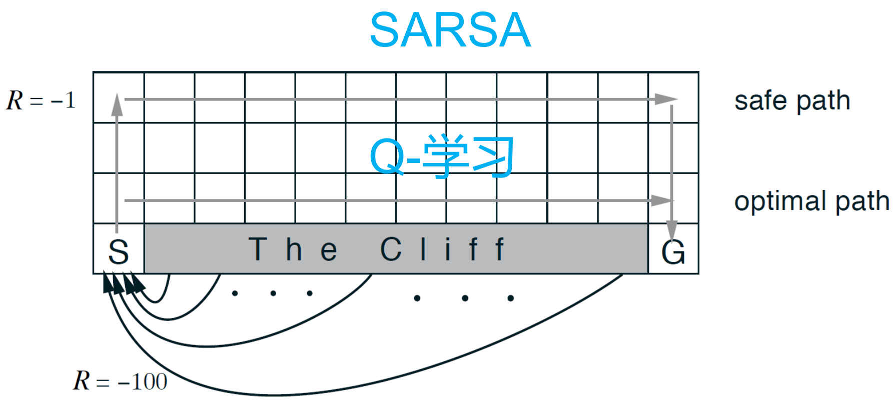
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$

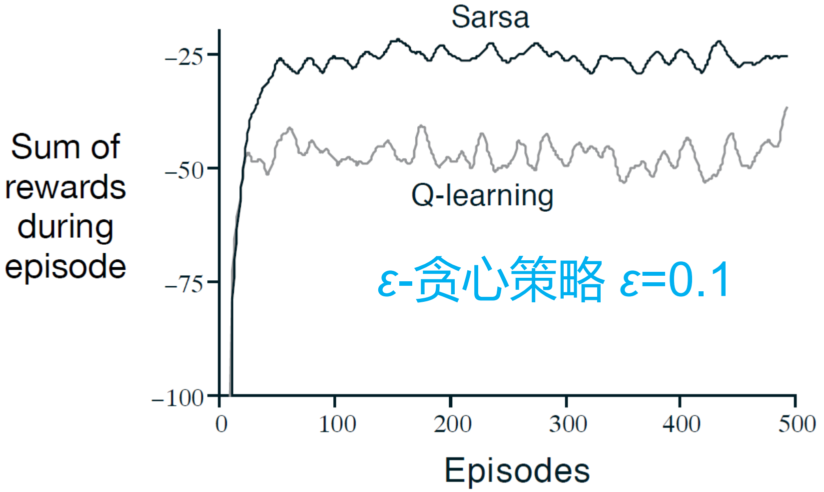
- 为什么不需要重要性采样？
  - 使用了状态-动作值函数而不是使用状态值函数

我们正在更新一个状态-行为对时，我们不必关心我们选择行为的可能性有多大；既然我们已经选择了它，我们希望完全从接下来发生的事情中学习，只对随后的行动进行重要性抽样。

SARSA与Q 学习对比实验

- 悬崖边行走（Cliff-walking）
  - 无折扣的奖励
  - 片段式的任务
  - 所有移动奖励 = -1
  - 踏入悬崖区域会产生-100奖励并将智能体送回开始处
- 为什么会有图示结果？

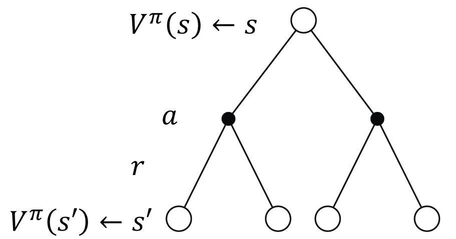
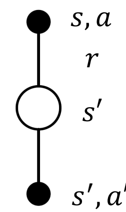
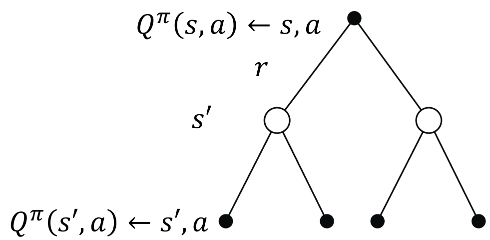
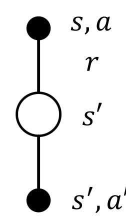
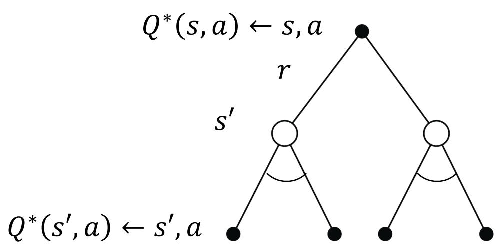
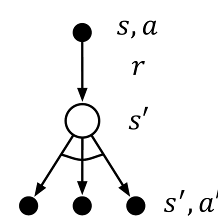




01

多步时序差分预测

回顾：动态规划（DP）和时序差分（TD）的关系

	完全反向传播(DP)	采样反向传播(TD)
状态值函数的贝尔曼期望方程 $V^{\pi}(s)$	 迭代的策略评估	 时序差分学习
状态-动作值函数的贝尔曼期望方程 $Q^{\pi}(s,a)$	 $Q-$策略迭代	 SARSA
状态-动作值函数的贝尔曼最优方程 $Q^{*}(s,a)$	 $Q-$价值迭代	 $Q-$学习

回顾：动态规划（DP）和时序差分（TD）的关系

完全反向传播(DP)	采样反向传播(TD)
迭代的策略评估 $V(s) \leftarrow \mathbb{E}[r + \gamma V(s') \mid s]$	时序差分学习 $V(s) \overset{\alpha}{\leftarrow} r + \gamma V(s')$
$Q -$ 策略迭代 $Q(s,a) \leftarrow \mathbb{E}[r + \gamma Q(s',a') \mid s,a]$	SARSA $Q(s,a) \overset{\alpha}{\leftarrow} r + \gamma Q(s',a')$
$Q -$ 价值迭代 $Q(s,a) \leftarrow \mathbb{E}\left[r + \gamma \max_{a'} Q(s',a') \,\middle|\, s,a\right]$	$Q -$ 学习 $Q(s,a) \overset{\alpha}{\leftarrow} r + \gamma \max_{a'} Q(s',a')$

其中 $x \overset{\alpha}{\leftarrow} y \equiv x \leftarrow x + \alpha(y - x)$

回顾：蒙特卡洛方法&时序差分法

蒙特卡洛方法

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

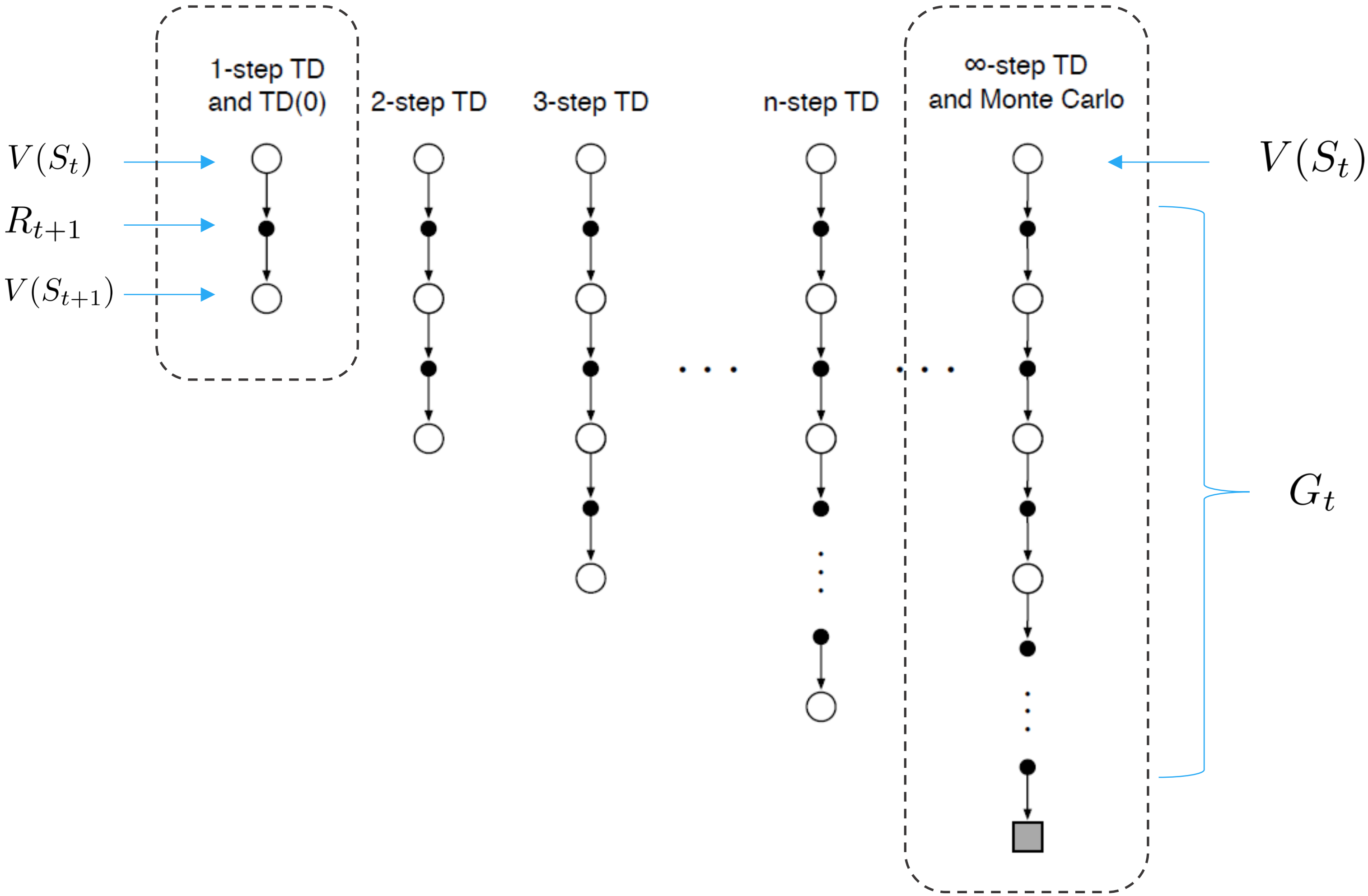
$$G_t = R_{t+1} + \gamma R_{t_2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T$$

时序差分法

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

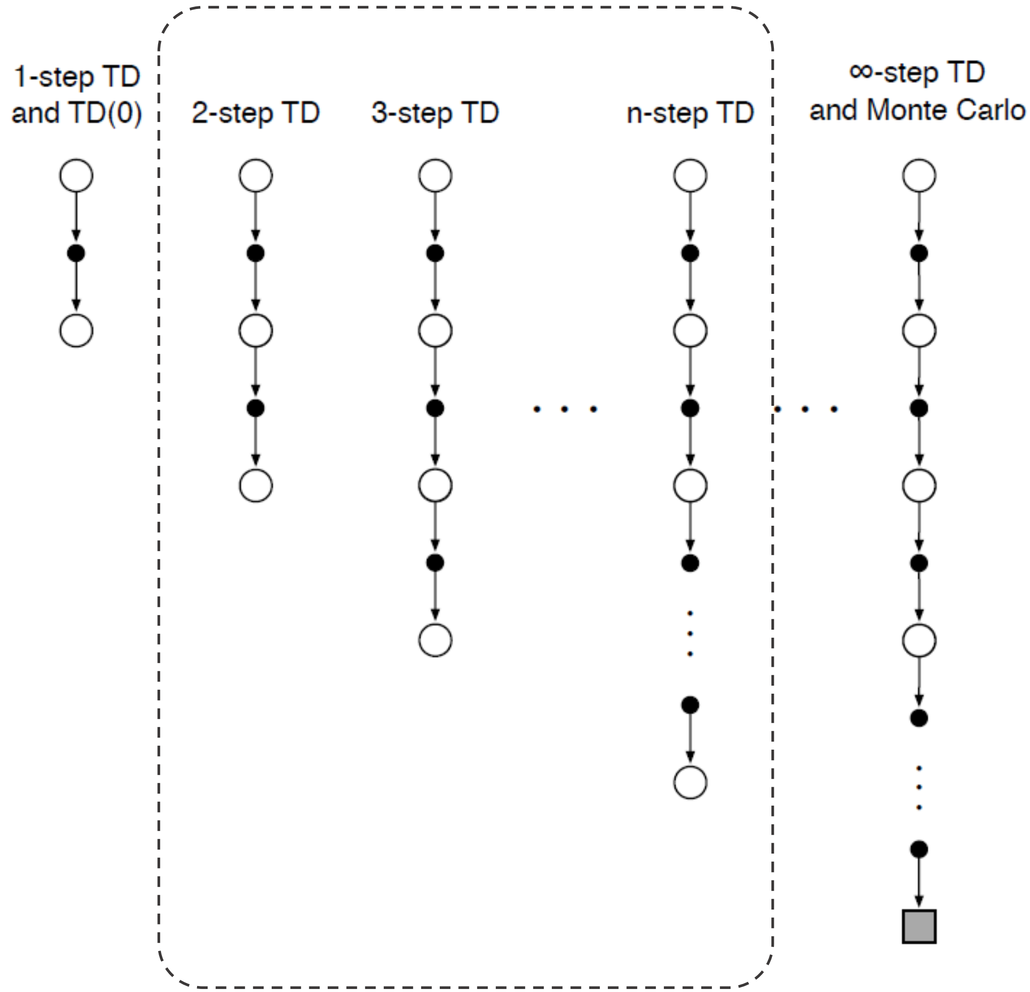
有没有方法介于两者之间？

多步时序差分预测



Reinforcement Learning: An Introduction
Richard S.Sutton and Andrew G.Barto

多步时序差分预测


1-step TD
and TD(0)
2-step TD
3-step TD
n-step TD
∞-step TD
and Monte Carlo

n 步累计奖励

- 考虑下列 n 步累计奖励，$n = 1, 2, \ldots, \infty$

$n = 1$ （TD） $G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$

$n = 2$ $G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$

$\vdots$ $\vdots$

$n = \infty$ （MC） $G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$

- 定义 n 步累计奖励

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- n 步时序差分学习

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

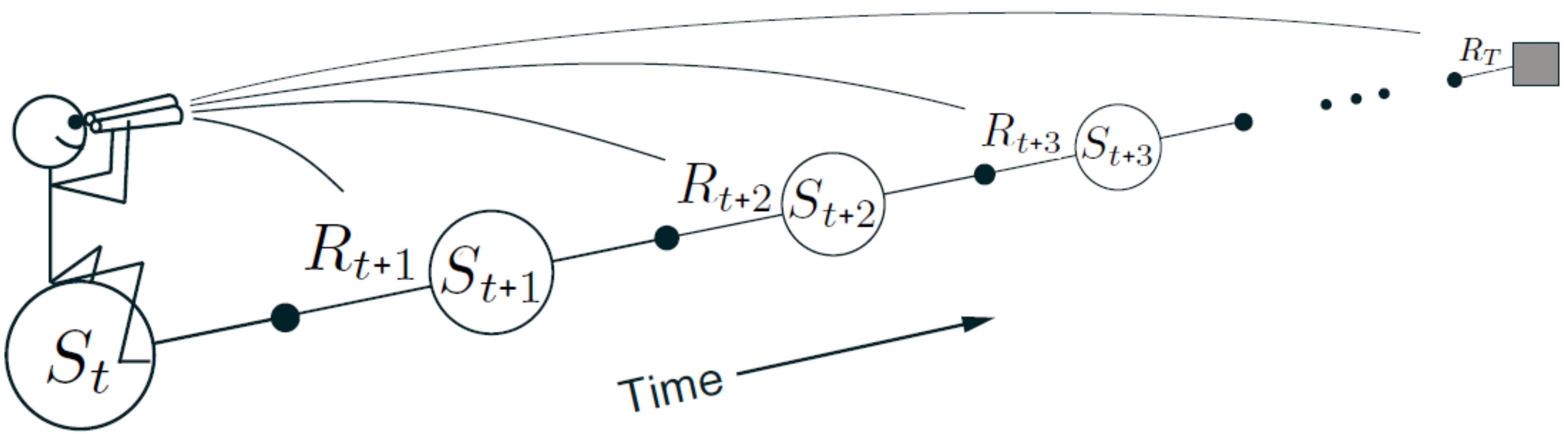
n 步累计奖励

- 定义 n 步累计奖励

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- n 步时序差分学习

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{(n)} - V(S_t))$$



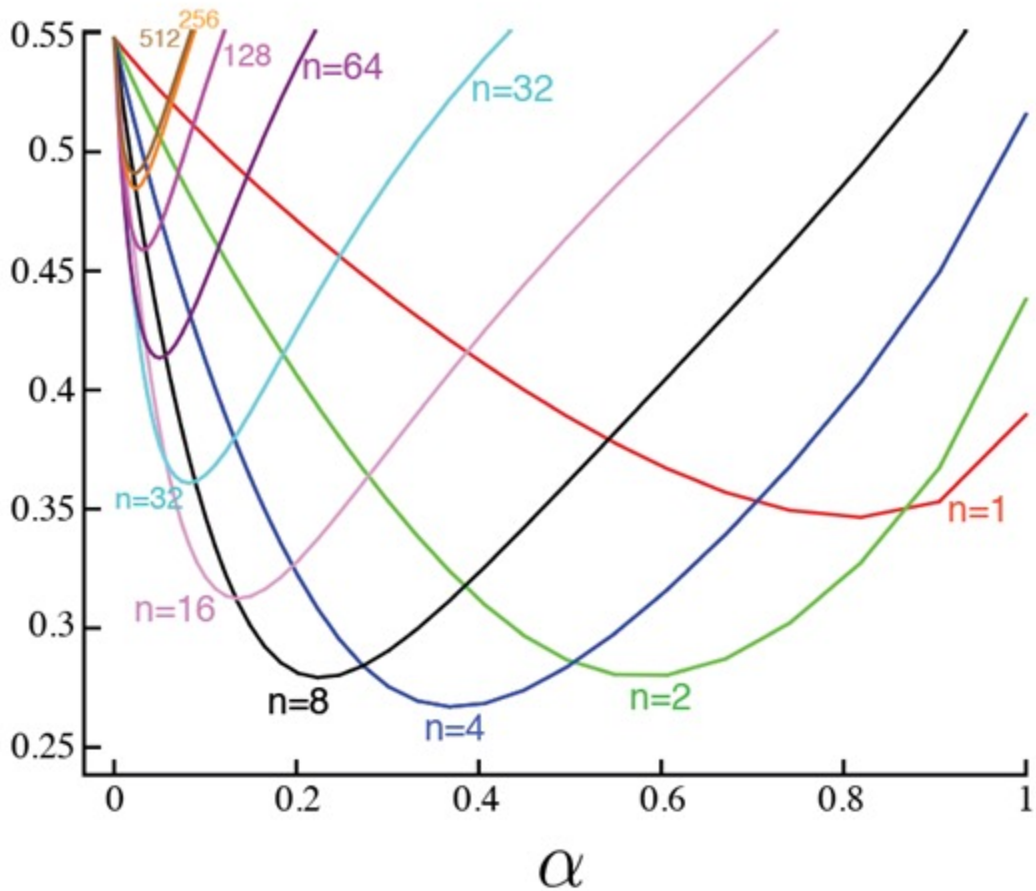
随机游走的例子里使用 n 步时序差分

随机游走

- 每个片段都从中间状态 C 开始。
- 每一时刻都有均等的概率向左走或者向右走。
- 到最左侧或者最右侧时，片段结束。
- 如果走到最右侧，得到的奖励为1；如果走到最左侧，得到的奖励为0。

多步时序差分法的表现

拥有19个状态的随机游走游戏中，在10个片段(episode)结束时的RMS误差

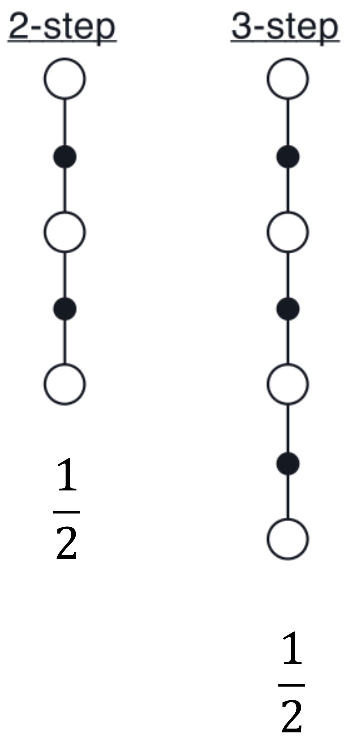


Reinforcement Learning: An Introduction
Richard S.Sutton and Andrew G.Barto

平均 n 步累计奖励

- 我们可以进一步对不同 n 下的 n 步累计奖励求平均值
- 例如，求 2 步和 3 步时的平均累计奖励

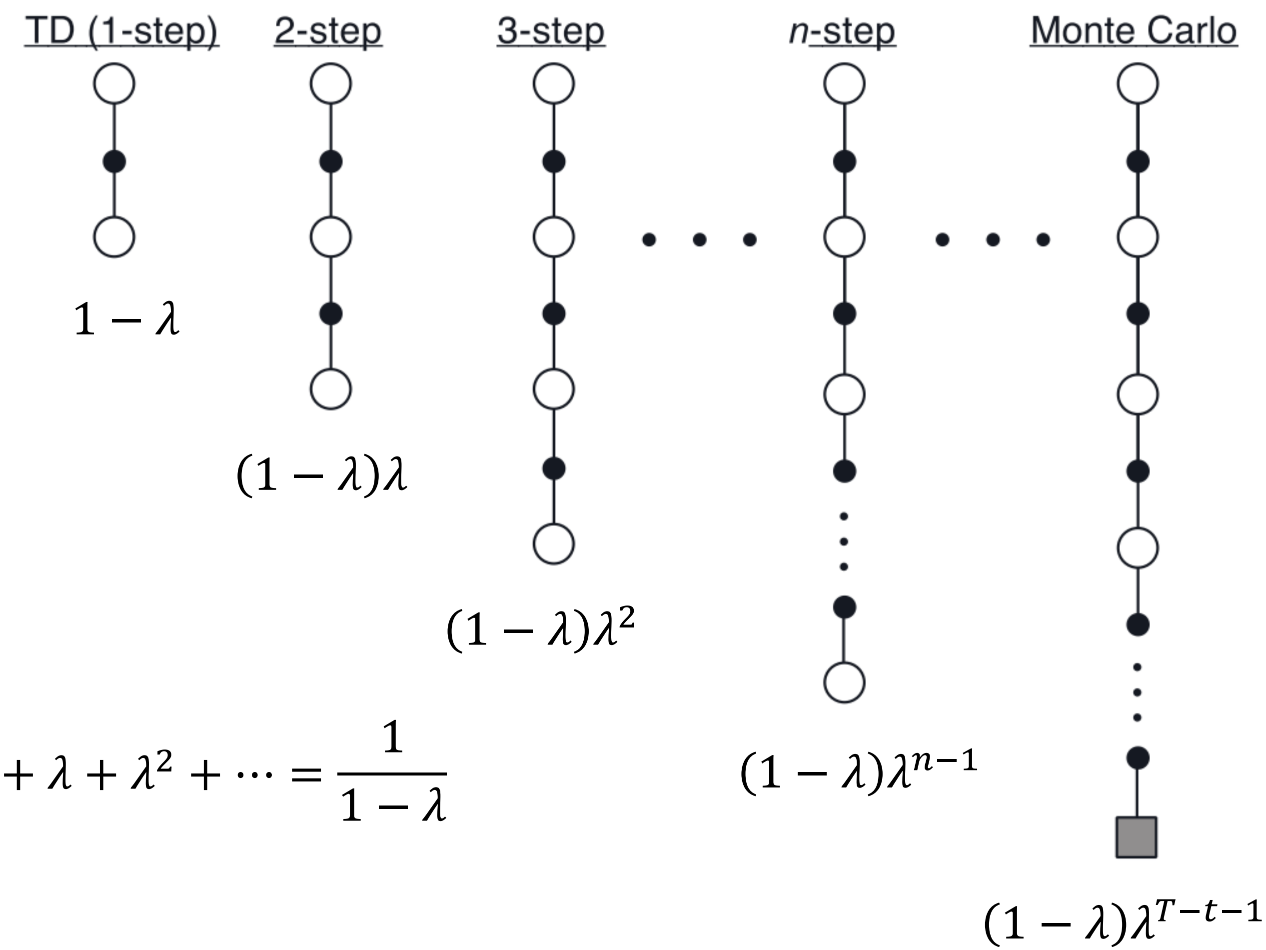
$$\frac{1}{2}G^{(2)}+\frac{1}{2}G^{(3)}$$



- 上式结合两种不同时间步长的信息
- 我们是否能够结合所有不同时间步长的信息呢？

使用平均 n 步累计奖励的 TD(λ) 算法

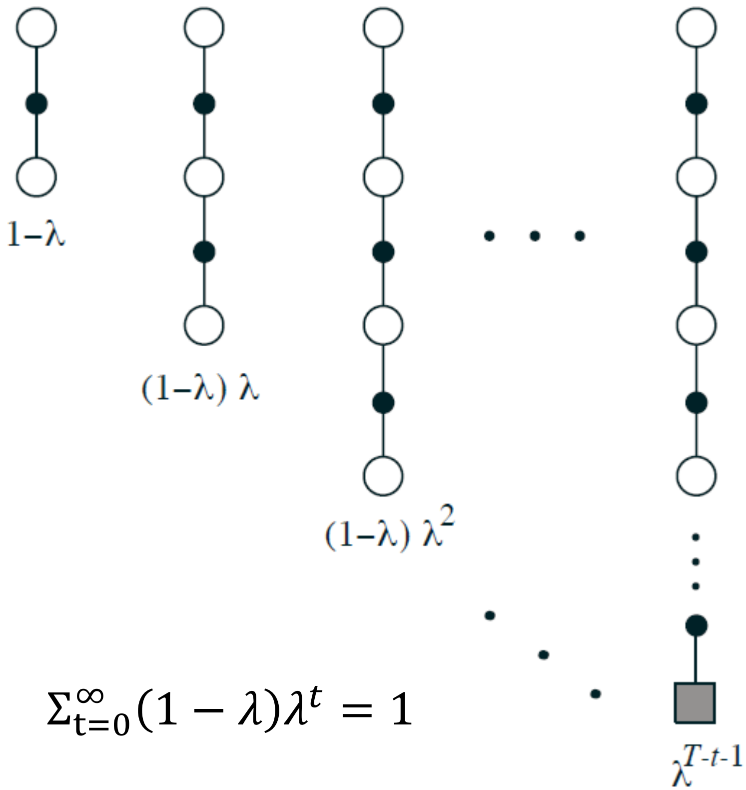
TD(λ), λ −累计奖励



$$1+\lambda+\lambda^2+\cdots=\frac{1}{1-\lambda}$$

使用平均 n 步累计奖励的 TD(λ) 算法

TD(λ), λ – 累计奖励



$\Sigma_{t=0}^{\infty}(1-\lambda)\lambda^{t} = 1$

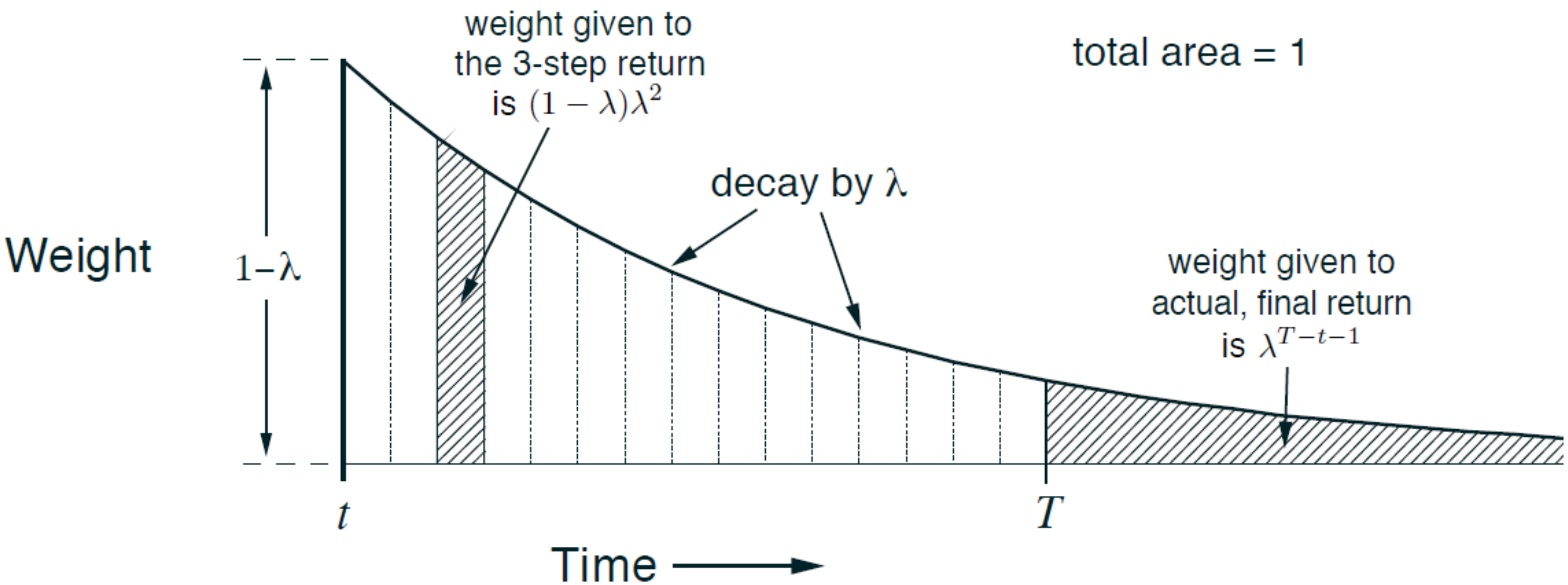
- λ – 累计奖励 G_t^{λ} 结合了所有 n 步累计奖励 $G_t^{(n)}$
- 使用权重 $(1-\lambda)\lambda^{n-1}$

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

- TD(λ)

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t^{\lambda} - B(S_t)\right)$$

使用平均 n 步累计奖励的 TD(λ) 算法

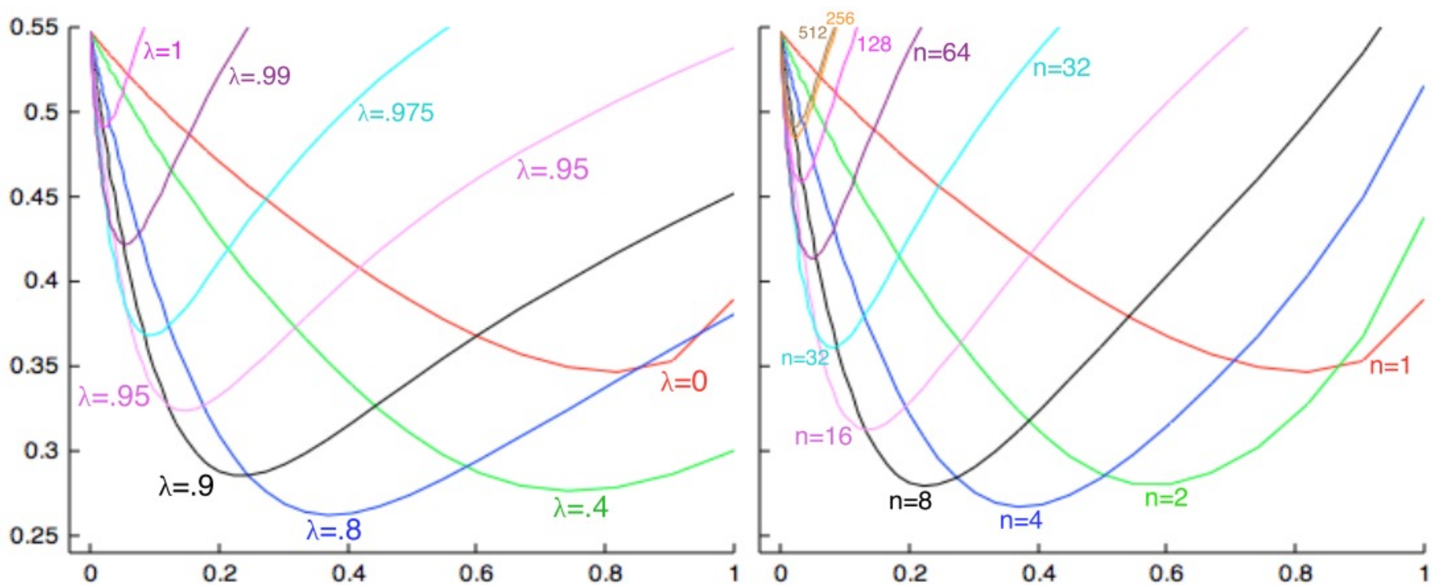


$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}G_t^{(n)} + \lambda^{T-t-1}G_t$$

- 当 $\lambda = 1$ 时，$G_t^{\lambda} = G_t$，相当于蒙特卡洛方法
- 当 $\lambda = 0$ 时，$G_t^{\lambda} = G_t^{(1)}$，相当于单步时序差分方法

TD(λ) vs. n 步时序差分

前10个片段
(episode)
结束时的
Offline
RMS误差



19个状态的随机游走实验结果

- 在使用最佳的 α 与 λ 值时，离线 $\lambda-$ 累计奖励算法具有稍好一些的实验结果

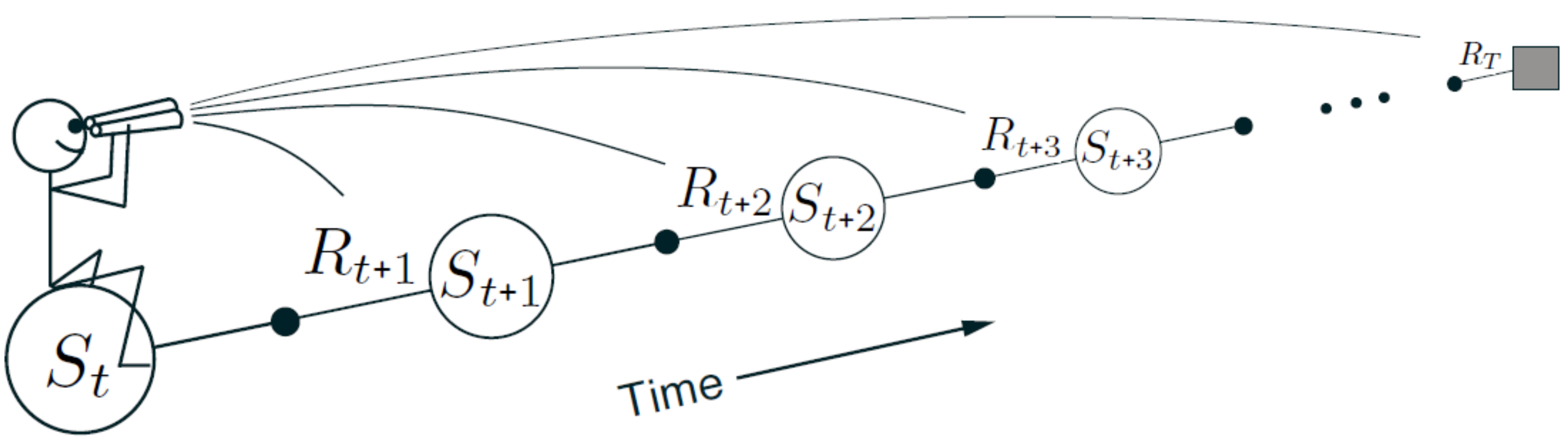
n 步累计奖励

- 定义 n 步累计奖励

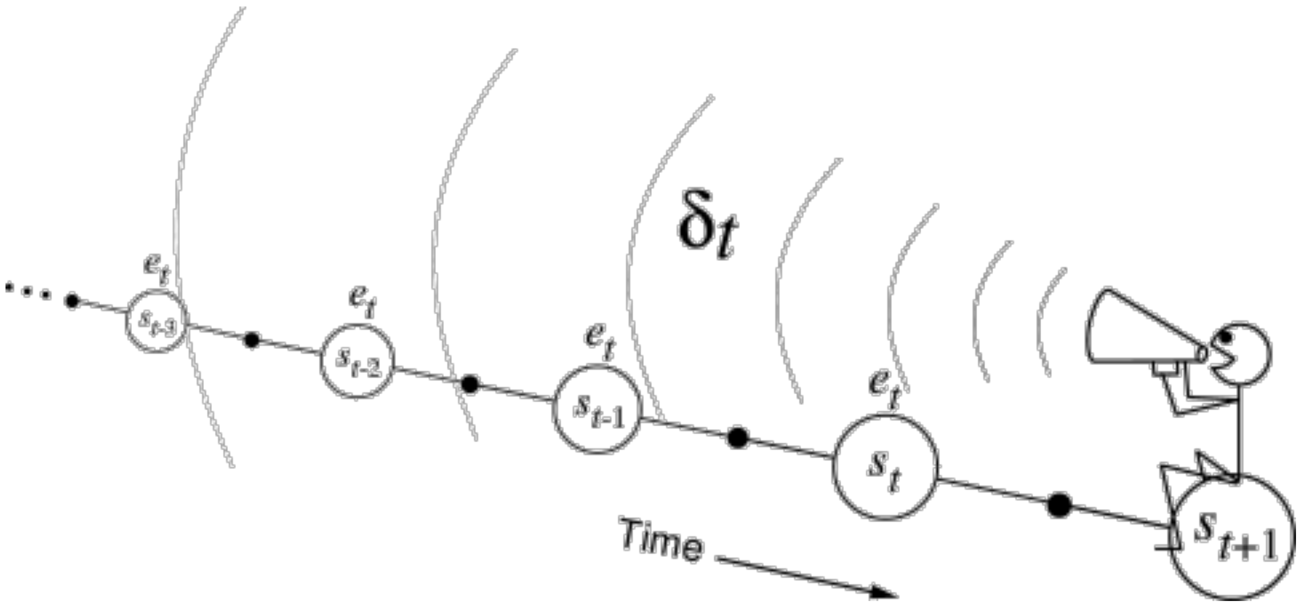
$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- n 步时序差分学习

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{(n)} - V(S_t))$$

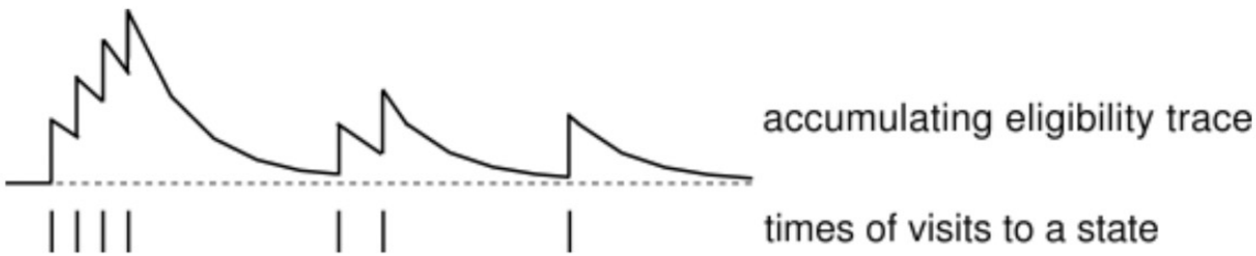


The Backward View of TD(λ)



资格迹（ Eligibility Trace ）

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{if } s \neq s_t; \\ \gamma\lambda e_{t-1}(s) + 1 & \text{if } s = s_t, \end{cases}$$



$$\delta_t \;=\; r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t).$$

$$\Delta V_t(s) = \alpha \delta_t e_t(s), \qquad \text{for all } s \in \mathcal{S}.$$

Online TD (λ)

Initialize $V(s)$ arbitrarily and $e(s) = 0$, for all $s \in \mathcal{S}$

Repeat (for each episode):

- Initialize s
- Repeat (for each step of episode):
  - $a \leftarrow$ action given by π for s
  - Take action a, observe reward, r, and next state, s'
  - $\delta \leftarrow r + \gamma V(s') - V(s)$
  - $e(s) \leftarrow e(s) + 1$
  - For all s:
    - $V(s) \leftarrow V(s) + \alpha \delta e(s)$
    - $e(s) \leftarrow \gamma \lambda e(s)$
  - $s \leftarrow s'$
- until s is terminal

02

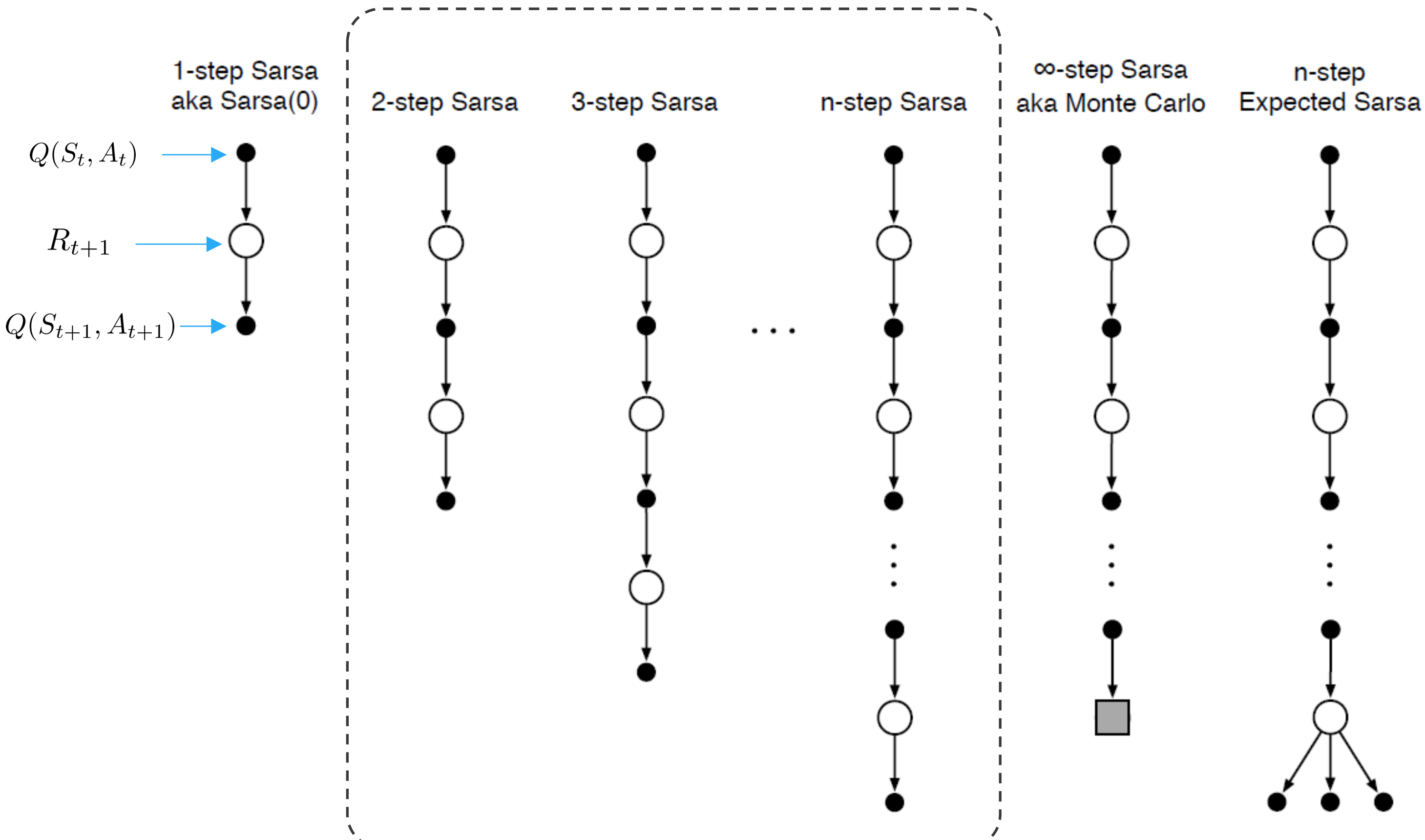
多步Sarsa

ElitesAI

多步Sarsa

- n 步的思想是否只能用在预测上？我们能不能把这种思想放在控制上？

将这种思想用在 Sarsa 上， 我们得到了 n 步 Sarsa算法



多步Sarsa

- 类似与 n 步TD，我们有：

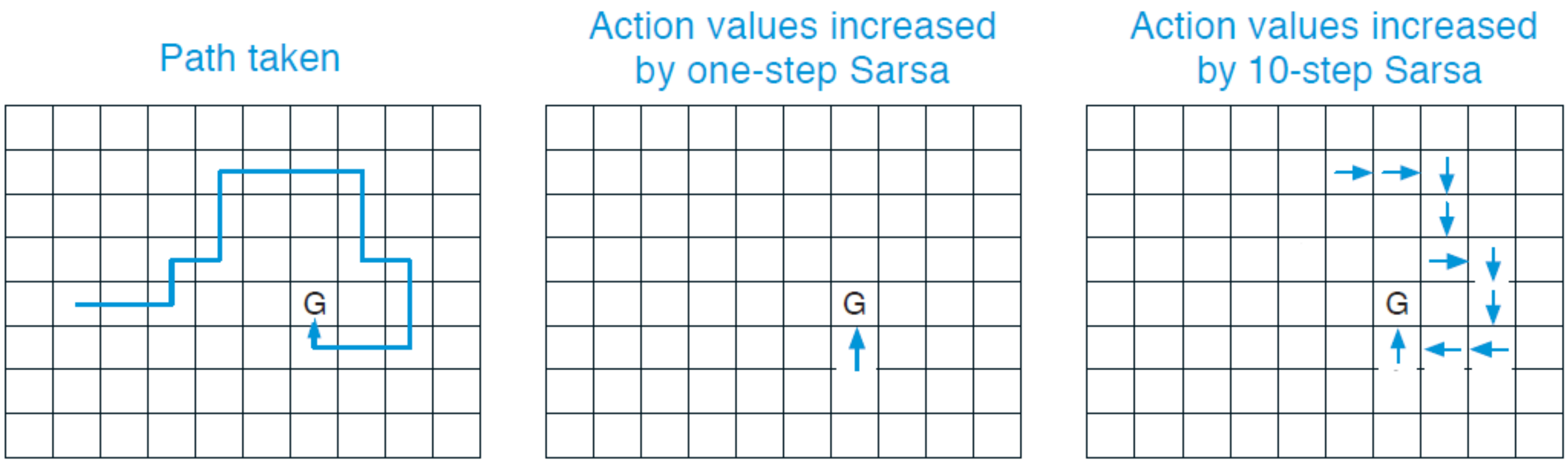
$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^{n} Q_{t+n-1}\left(S_{t+n}, A_{t+n}\right)$$

$$where, n \geq 1, 0 \leq t < T - n$$

$$Q_{t+n}\left(S_t, A_t\right) \doteq Q_{t+n-1}\left(S_t, A_t\right) + \alpha\left[G_{t:t+n} - Q_{t+n-1}\left(S_t, A_t\right)\right]$$

$$where, 0 \leq t < T$$

多步Sarsa的例子


Path taken
Action values increased by one-step Sarsa
Action values increased by 10-step Sarsa
G
G
G

THANK YOU