

Reinforcement Learning China Summer School



RLChina 2021

强化学习简介

Introduction to Reinforcement Learning

Weinan Zhang (张伟楠)

John Hopcroft Center, Shanghai Jiao Tong University

<http://wnzhang.net/>

Aug. 17, 2021



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY



JOHN HOPCROFT
CENTER FOR
COMPUTER SCIENCE





张伟楠 – 上海交通大学 副教授

电院John中心、计算机系、APEX实验室

2016年UCL博士，2011年交大ACM班本科，主要研究强化学习和信息检索

动手学强化学习 hrl.boyuai.com



动手学强化学习



动手学

反馈

视频教程

GitHub

前言

强化学习基础篇

多臂老虎机

• 马尔可夫决策过程

动态规划算法

时序差分算法

Dyna-Q算法

强化学习前沿篇

DQN 算法

DQN 改进算法

策略梯度算法

Actor-Critic 算法

敬请期待

强化学习前沿篇

敬请期待

写在最后

价值函数 (Value Function)

在马尔可夫奖励过程中，一个状态的期望回报被称为这个状态的价值 (value)。所有状态的价值就组成了价值函数，价值函数的输入为某个状态，输出为这个状态的价值。我们将价值函数写成 $V(s) = \mathbb{E}[G_t | S_t = s]$ 。我们将其展开为

$$\begin{aligned} V(s) &= \mathbb{E}[G_t | S_t = s] \\ &= \mathbb{E}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots | S_t = s] \\ &= \mathbb{E}[R_t + \gamma(R_{t+1} + \gamma R_{t+2} + \dots) | S_t = s] \\ &= \mathbb{E}[R_t + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}[R_t + \gamma V(S_{t+1}) | S_t = s] \end{aligned}$$

上式的最后一个等号中，一方面，即时奖励的期望就等于即时奖励，即 $\mathbb{E}[R_t | S_t = s] = r(s)$ ，另一方面，式子中剩余部分 $\mathbb{E}[\gamma V(S_{t+1}) | S_t = s]$ 可以根据从状态 s 出发的转移概率得到，即我们可以得到

$$V(s) = r(s) + \gamma \sum_{s' \in S} p(s' | s) V(s')$$

上式就是马尔可夫奖励过程中非常有名的贝尔曼方程 (Bellman Equation)。上式的贝尔曼方程对每一个状态都成立。若一个马尔可夫奖励过程一共有 n 个状态，即 $S = \{s_1, s_2, \dots, s_n\}$ ，我们将所有状态的价值表示成一个列向量 $V = [V(s_1), V(s_2), \dots, V(s_n)]^T$ ，同理奖励函数写成一个列向量 $R = [r(s_1), r(s_2), \dots, r(s_n)]^T$ 。于是我们可以将贝尔曼方程写成矩阵的形式

$$\begin{aligned} V &= R + \gamma P V \\ \begin{bmatrix} V(s_1) \\ V(s_2) \\ \dots \\ V(s_n) \end{bmatrix} &= \begin{bmatrix} r(s_1) \\ r(s_2) \\ \dots \\ r(s_n) \end{bmatrix} + \gamma \begin{bmatrix} p(s_1|s_1) & p(s_2|s_1) & \dots & p(s_n|s_1) \\ p(s_1|s_2) & p(s_2|s_2) & \dots & p(s_n|s_2) \\ \dots & \dots & \dots & \dots \\ p(s_1|s_n) & p(s_2|s_n) & \dots & p(s_n|s_n) \end{bmatrix} \begin{bmatrix} V(s_1) \\ V(s_2) \\ \dots \\ V(s_n) \end{bmatrix} \end{aligned}$$

于是我们可以直接根据矩阵运算求解得到以下解析解：

$$\begin{aligned} V &= R + \gamma P V \\ (I - \gamma P) V &= R \\ V &= (I - \gamma P)^{-1} R \end{aligned}$$

但实际上，解析解的计算复杂度是 $O(n^3)$ ， n 是状态个数，所以这种方法只适用很小的马尔可夫奖励过程。求解较大规模的马尔可夫奖励过程中的价值函数，我们可以使用动态规划 (Dynamic Programming)，蒙特卡洛法 (Monte-Carlo method)，时序差分法 (Temporal Difference)，这些方法我们将会在之后学到。

我们接下来将求解价值函数的解析解方法用代码写出来，并据此计算上图 MRP 中所有状态的价值。

```
def compute(P, rewards, gamma, states_num):
    ''' 利用贝尔曼方程矩阵形式计算解析解, states_num是MRP的状态数 '''
    rewards = np.array(rewards).reshape((-1,1)) #rewards写成列向量形式
    value = np.dot(np.linalg.inv(np.eye(states_num, states_num) - gamma * P), rewards)
    return value
```

```
V = compute(P, rewards, gamma, 6)
print("MRP中每个状态价值分别为\n", V)
```

简介

马尔可夫过程 (Markov Process)

随机过程 (Stochastic Process)

马尔可夫性质 (Markov Property)

马尔可夫过程 (Markov Process)

马尔可夫奖励过程 (Markov Reward...

回报 (Return)

价值函数 (Value Function)

马尔可夫决策过程 (Markov Decisio...

策略 (Policy)

状态价值函数 (State-value Functi...

动作价值函数 (Action-value Func...

贝尔曼期望方程 (Bellman Expect...

蒙特卡洛方法 (Monte-Carlo metho...

占用度量 (Occupancy Measure)

最优策略 (Optimal Policy)

贝尔曼最优方程(Bellman Optimali...

总结

伯禹ElitesAI作为本节课辅助视频学习平台

伯禹人工智能学院
Make AI Education for All

请输入想学习的知识点

问题反馈 雨滴 51780 云朵 0.1

课程大纲

全部展开

马尔科夫决策过程

基于动态规划的强化学习

代码实践：基于动态规划的强化...

基于模型的强化学习

蒙特卡洛方法

蒙特卡洛价值预测

模型无关控制方法

重要性采样

时序差分学习

SARSA

Q学习算法及其收敛性

代码实践：Q-learning和SARSA...

多步自助法

马尔科夫决策过程

ElitesAI-强

马尔科夫性质

"The future is independent of the past given the present"

定义：

- 状态 S_t 是马尔科夫的，当且仅当
$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

性质：

- 状态从历史 (history) 中捕获了所有相关信息
- 当状态已知的时候，可以抛开历史不管
- 也就是说，当前状态是未来的充分统计量

Boyu.AI

讨论区(26条讨论) 全部 精华 提问 笔记 闲聊

请输入想搜索的内容

西云佳 7天之前 加精 删除

实际上很多现实场景都是没有MDP性质的，但是强化学习还是通过各种方法尽量抽象成MDP，是因为大多数算法都是基于MDP推导的吗？目前有没有不基于MDP的方法呢？

1 回复 链接

Jerry Jin 7天之前 加精 删除

强化学习和一种兼顾控制论的机器学习理论的一种方法，其建立是基于控制论状态转移进行建模的，所以提出了相关MDP的定义和性质。当然在现实场景中很多都不符合严格MDP的定义，但是可能符合MDP的变体，比如POMDP(Partially Observable Markov Decision Process)，就可以有效的只有Environment中的State具备的MDP的情况扩展为Agent中的Observation也有类似MDP的性质。大多数RL的相关的算法是基于MDP进行推导的，不基于MDP的推导的证明往往是与过程性无关，比如一些Value Function的Lipschitz连续的保证和证明，但是涉及到多步的证明一般需要使用MDP的性质。

1 回复 链接

堂本源 7天之前 加精 删除

请问对于状态在绝大情况下并不重复的问题，此时experience的积累似乎不再有用，是否仍然符合MDP的设定？MF方法是否还有用？此时应该用RL的什么方法或技巧来解决呢？

1 回复 链接

Suzy 7天之前 加精 删除

你指的“状态几乎不重复”是什么意思呢，有具体的例子吗？是在单局里面状态几乎不重复的意思吗，这跟能不能建造成MDP应该没有直接的关系

MDP的定义关键点在于是否满足马尔科夫性质，也就是说状态转移只与当前状态有关，与之前经历过的状态无关

1 回复 链接

堂本源 7天之前 加精 删除

您好！状态几乎不重复，是指譬如股票系统里状态的设定，见过的文章中总以：(日期、开盘价、收盘价、最高、最低、持股数量、经济学指标等)等多个数字作为状态的设定，我认为这样的设定在经过一次后，后期不会再有同样的状态，但作者仍然使用了experience replay方法（会有影响？），或者用Istm。这样的状态设定是具有马尔科夫性质的latent variable吗？

1 回复 链接

Suzy 7天之前 加精 删除

<https://www.boyuai.com/elites/course/xVqhU42F5IDky94x>

上海交通大学强化学习课程大纲（16周）

强化学习基础部分

1. 强化学习、探索与利用
2. MDP和动态规划
3. 值函数估计
4. 无模型控制方法
5. 参数化的值函数和策略
6. 规划与学习
7. 深度强化学习价值方法
8. 深度强化学习策略方法

本节课
内容

强化学习前沿部分

9. 基于模型的深度强化学习
10. 离线强化学习
11. 模仿学习
12. 参数化动作空间
13. 多智能体强化学习基础
14. 多智能体强化学习前沿
15. 强化学习的应用
16. 技术与交流与回顾

强化学习基础

1. 强化学习
2. MDP
3. 动态规划
4. 值函数估计
5. 无模型控制方法
6. 参数化的值函数
7. 参数化的策略

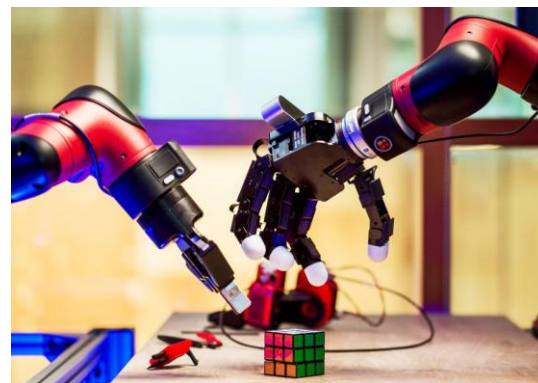
两种人工智能任务类型

□ 预测型任务

- 根据数据预测所需输出（有监督学习）
- 生成数据实例（无监督学习）

□ 决策型任务

- 在动态环境中采取行动（强化学习）
 - 转变到新的状态
 - 获得即时奖励
 - 随着时间的推移最大化累计奖励
 - Learning from interaction in a trial-and-error manner

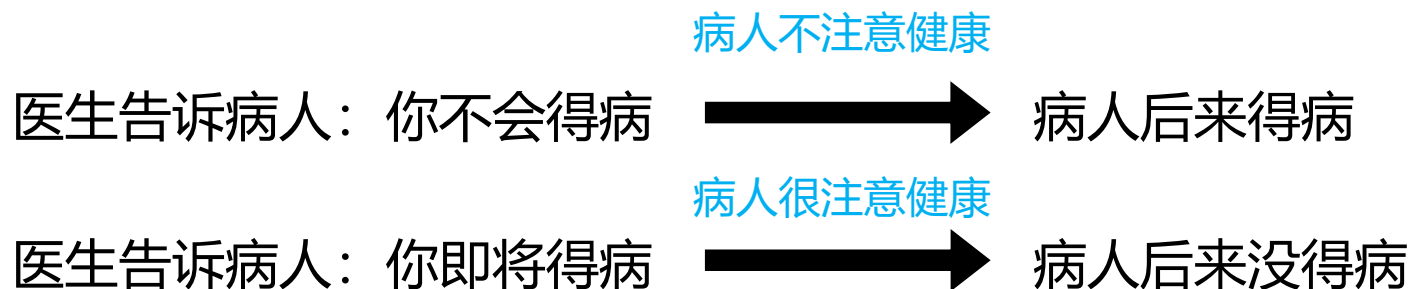


决策和预测的不同

- 决策亲自改变世界
 - 医生或者AI直接给病人下达治疗方案
- 预测辅助别人改变世界
 - AI告诉医生病人可能的得病预测，医生综合各方面判断最后给病人下达治疗方案



医生预判悖论



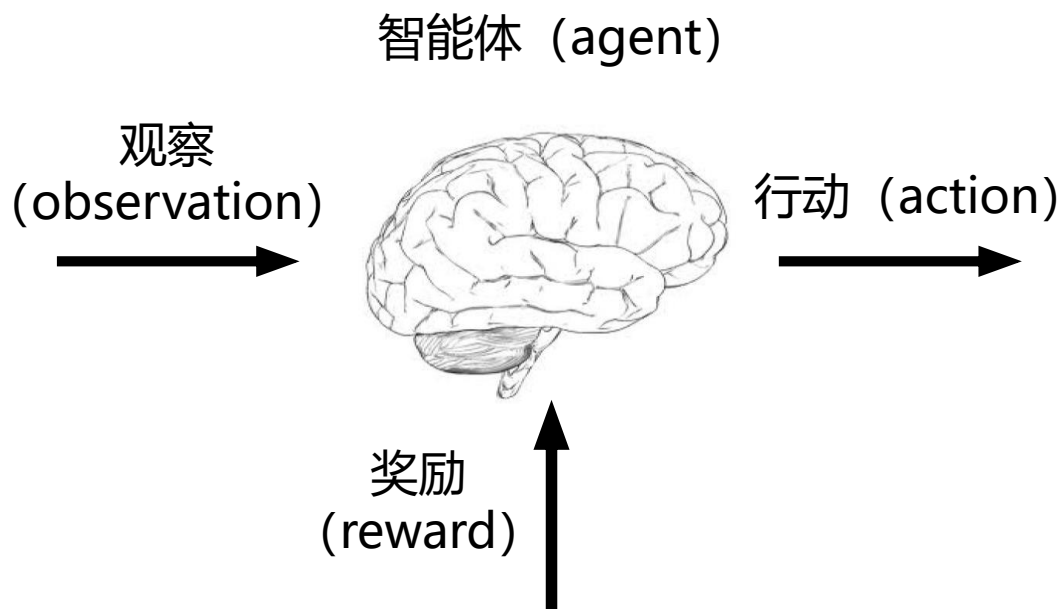
序列决策 (Sequential Decision Making)



只要是序列决策问题，就可以用强化学习来解

强化学习定义

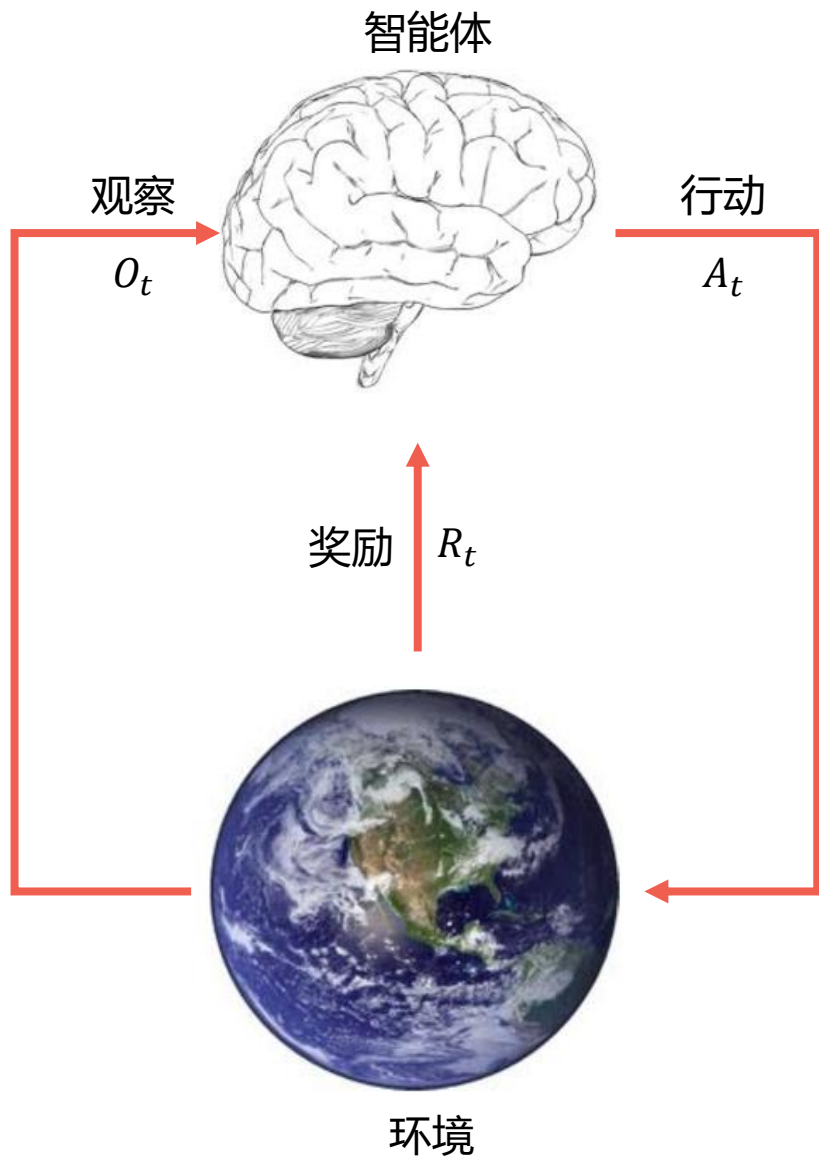
- 通过从交互中学习来实现目标的计算方法



- 三个方面:

- 感知: 在某种程度上感知环境的状态
- 行动: 可以采取行动来影响状态或者达到目标
- 目标: 随着时间推移最大化累积奖励

强化学习交互过程



□ 在每一步 t , 智能体:

- 获得观察 O_t
- 获得奖励 R_t
- 执行行动 A_t

□ 环境:

- 获得行动 A_t
- 给出观察 O_{t+1}
- 给出奖励 R_{t+1}

□ t 在环境这一步增加

在与动态环境的交互中学习

有监督、无监督学习

Model ←



Fixed Data

强化学习

Agent ↔



Agent不同，交互出的数据也不同！

Dynamic Environment

强化学习基础

1. 强化学习
- 2. MDP**
3. 动态规划
4. 值函数估计
5. 无模型控制方法
6. 参数化的值函数
7. 参数化的策略

随机过程

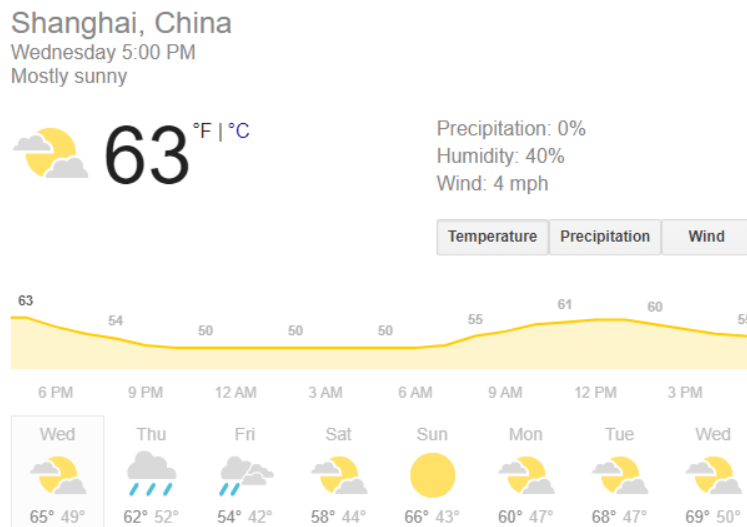
- 随机过程是一个或多个事件、随机系统或者随机现象随时间发生演变的过程

$$\mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

- 概率论研究静态随机现象的统计规律
- 随机过程研究动态随机现象的统计规律



布朗运动



天气变化

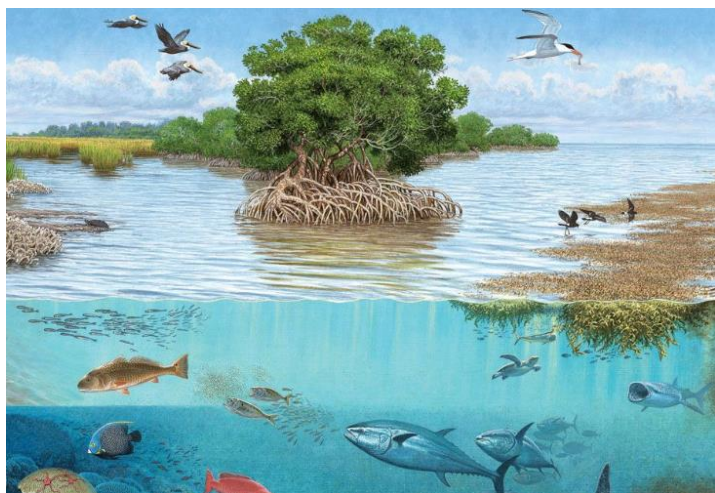
随机过程



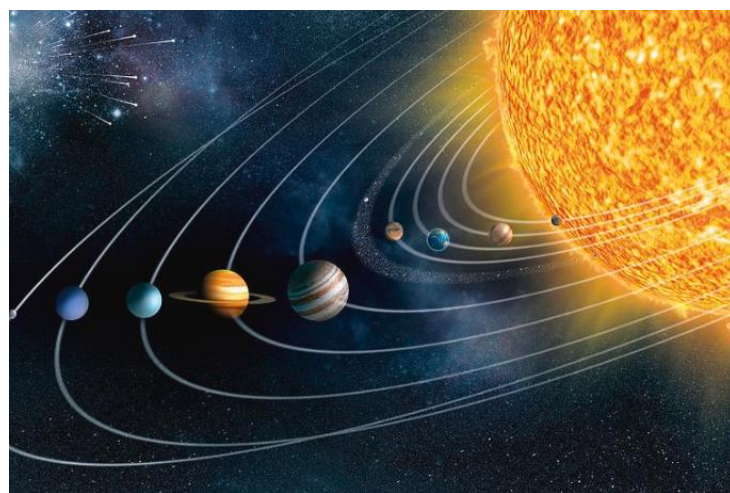
足球比赛



城市交通



生态系统



星系

马尔可夫过程

- 马尔可夫过程 (Markov Process) 是具有马尔可夫性质的随机过程

“The future is independent of the past given the present”

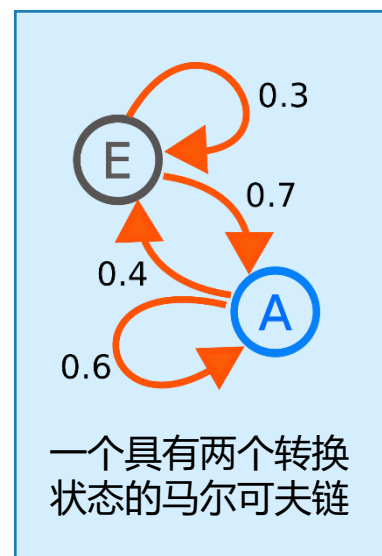
- 定义:

- 状态 S_t 是马尔可夫的, 当且仅当

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

- 性质:

- 状态从历史 (history) 中捕获了所有相关信息
- 当状态已知的时候, 可以抛开历史不管
- 也就是说, 当前状态是未来的充分统计量



马尔可夫决策过程

□ 马尔可夫决策过程 (Markov Decision Process, MDP)

- 提供了一套为在结果部分随机、部分在决策者的控制下的决策过程建模的数学框架

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

$$\mathbb{P}[S_{t+1}|S_t, A_t]$$

□ MDP形式化地描述了一种强化学习的环境

- 环境完全可观测
- 即，当前状态可以完全表征过程 (马尔可夫性质)

MDP五元组

□ MDP可以由一个五元组表示 $(S, A, \{P_{sa}\}, \gamma, R)$

- S 是状态的集合
 - 比如，迷宫中的位置，Atari游戏中的当前屏幕显示
- A 是动作的集合
 - 比如，向N、E、S、W移动，手柄操纵杆方向和按钮
- P_{sa} 是状态转移概率
 - 对每个状态 $s \in S$ 和动作 $a \in A$ ， P_{sa} 是下一个状态在 S 中的概率分布
- $\gamma \in [0,1]$ 是对未来奖励的折扣因子
- $R: S \times A \mapsto \mathbb{R}$ 是奖励函数
 - 有时奖励只和状态相关

MDP的动态性

□ MDP的动态如下所示:

- 从状态 s_0 开始
 - 智能体选择某个动作 $a_0 \in A$
 - 智能体得到奖励 $R(s_0, a_0)$
 - MDP随机转移到下一个状态 $s_1 \sim P_{s_0 a_0}$
- 这个过程不断进行

$$s_0 \xrightarrow{a_0, R(s_0, a_0)} s_1 \xrightarrow{a_1, R(s_1, a_1)} s_2 \xrightarrow{a_2, R(s_2, a_2)} s_3 \dots$$

- 直到终止状态 s_T 出现为止, 或者无止尽地进行下去
- 智能体的总回报为

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots$$

MDP的动态性

□ 在大部分情况下，奖励只和状态相关

- 比如，在迷宫游戏中，奖励只和位置相关
- 在围棋中，奖励只基于最终所围地盘的大小有关

□ 这时，奖励函数为 $R(s): S \mapsto \mathbb{R}$

□ MDP的过程为

$$s_0 \xrightarrow{a_0, R(s_0)} s_1 \xrightarrow{a_1, R(s_1)} s_2 \xrightarrow{a_2, R(s_2)} s_3 \cdots$$

□ 累积奖励为

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$$

REVIEW: 在与动态环境的交互中学习

有监督、无监督学习

Model ←



Fixed Data

强化学习

Agent ↔



Dynamic Environment

和动态环境交互产生的数据分布



- 给定同一个动态环境（即MDP），不同的策略采样出来的(状态-行动)对的分布是不同的
- 占用度量（Occupancy Measure）

$$\rho^{\pi}(s, a) = \mathbb{E}_{a \sim \pi(s), s' \sim p(s, a)} \left[\sum_{t=0}^T \gamma^t p(s_t = s, a_t = a) \right]$$

占用度量和策略

- 占用度量 (Occupancy Measure)

$$\rho^\pi(s, a) = \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim p(\cdot|s, a)} \left[\sum_{t=0}^T \gamma^t p(s_t = s, a_t = a) \right]$$

- 定理1: 和同一个动态环境交互的两个策略 π_1 和 π_2 得到的占用度量 ρ^{π_1} 和 ρ^{π_2} 满足

$$\rho^{\pi_1} = \rho^{\pi_2} \text{ 当且仅当 } \pi_1 = \pi_2$$

- 定理2: 给定一占用度量 ρ , 可生成该占用度量的唯一策略是

$$\pi_\rho = \frac{\rho(s, a)}{\sum_{a'} \rho(s, a')}$$

占用度量和策略

- 占用度量 (Occupancy Measure)

$$\rho^\pi(s, a) = \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim p(\cdot|s, a)} \left[\sum_{t=0}^T \gamma^t p(s_t = s, a_t = a) \right]$$

- 状态占用度量

$$\begin{aligned} \rho^\pi(s) &= \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim p(\cdot|s, a)} \left[\sum_{t=0}^T \gamma^t p(s_t = s) \right] \\ &= \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim p(\cdot|s, a)} \left[\sum_{t=0}^T \gamma^t p(s_t = s) \sum_{a'} p(a_t = a' | s_t = s) \right] \\ &= \sum_{a'} \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim p(\cdot|s, a)} \left[\sum_{t=0}^T \gamma^t p(s_t = s, a_t = a') \right] \\ &= \sum_{a'} \rho^\pi(s, a') \end{aligned}$$

占用度量和累计奖励

- 占用度量 (Occupancy Measure)

$$\rho^\pi(s, a) = \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim p(\cdot|s, a)} \left[\sum_{t=0}^T \gamma^t p(s_t = s, a_t = a) \right]$$

- ▣ 策略的累积奖励为

$$\begin{aligned} V(\pi) &= \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim p(\cdot|s, a)} [R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots] \\ &= \sum_{s, a} \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim p(\cdot|s, a)} \left[\sum_{t=0}^T \gamma^t p(s_t = s, a_t = a) \right] R(s, a) \\ &= \sum_{s, a} \rho^\pi(s, a) R(s, a) = \mathbb{E}_\pi [R(s, a)] \end{aligned}$$

↑
强化学习中的简写

强化学习基础

1. 强化学习
2. MDP
- 3. 动态规划**
4. 值函数估计
5. 无模型控制方法
6. 参数化的值函数
7. 参数化的策略

MDP中智能体的目标和策略

- 目标：选择能够最大化累积奖励期望的动作

$$\mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots]$$

- $\gamma \in [0,1]$ 是未来奖励的折扣因子，使得和未来奖励相比起来智能体更重视即时奖励
 - 以金融为例，今天的\$1比明天的\$1更有价值

- 给定一个特定的策略 $\pi(s): S \rightarrow A$

- 即，在状态 s 下采取动作 $a = \pi(s)$

- 给策略 π 定义价值函数

$$V^\pi(s) = \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s, \pi]$$

- 即，给定起始状态和根据策略 π 采取动作时的累积奖励期望

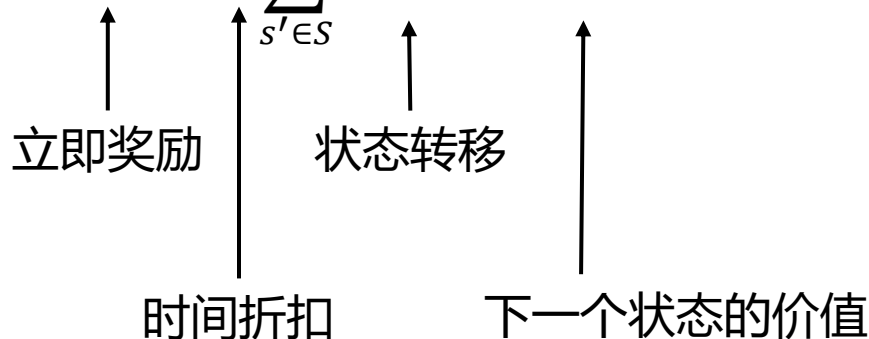
价值函数的Bellman等式

□ 给策略 π 定义价值函数

$$V^\pi(s) = \mathbb{E}[R(s_0) + \underbrace{\gamma R(s_1) + \gamma^2 R(s_2) + \cdots}_{\gamma V^\pi(s_1)} | s_0 = s, \pi]$$

$$= R(s) + \gamma \sum_{s' \in \mathcal{S}} P_{s\pi(s)}(s') V^\pi(s')$$

Bellman等式



立即奖励

时间折扣

状态转移

下一个状态的价值

最优价值函数

- 对状态 s 来说的最优价值函数是所有策略可获得的最大可能折扣奖励的和

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

- 最优价值函数的Bellman等式

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s')$$

- 最优策略

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

- 对状态 s 和策略 π

$$V^*(s) = V^{\pi^*}(s) \geq V^{\pi}(s)$$

价值迭代和策略迭代

□ 价值函数和策略相关

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s')$$

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

□ 可以对最优价值函数和最优策略执行迭代更新

- 价值迭代
- 策略迭代

价值迭代

- 对于一个动作空间和状态空间有限的MDP

$$|S| < \infty, |A| < \infty$$

- 价值迭代过程

1. 对每个状态 s , 初始化 $V(s) = 0$
2. 重复以下过程直到收敛 {

对每个状态, 更新

$$V(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V(s')$$

}

注意：在以上的计算中没有明确的策略

价值迭代例子：最短路径

g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

V_7

策略迭代

- 对于一个动作空间和状态空间有限的MDP

$$|S| < \infty, |A| < \infty$$

- 策略迭代过程

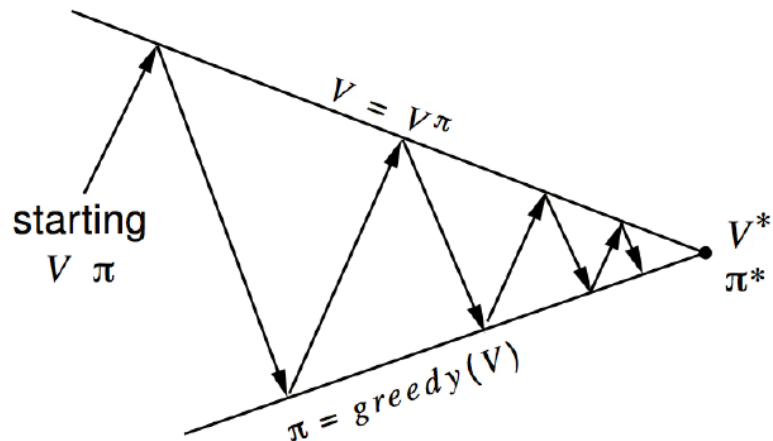
1. 随机初始化策略 π
2. 重复以下过程直到收敛{
 - a) 让 $V := V^\pi$
 - b) 对每个状态, 更新

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

}

更新价值函数会很耗时

策略迭代

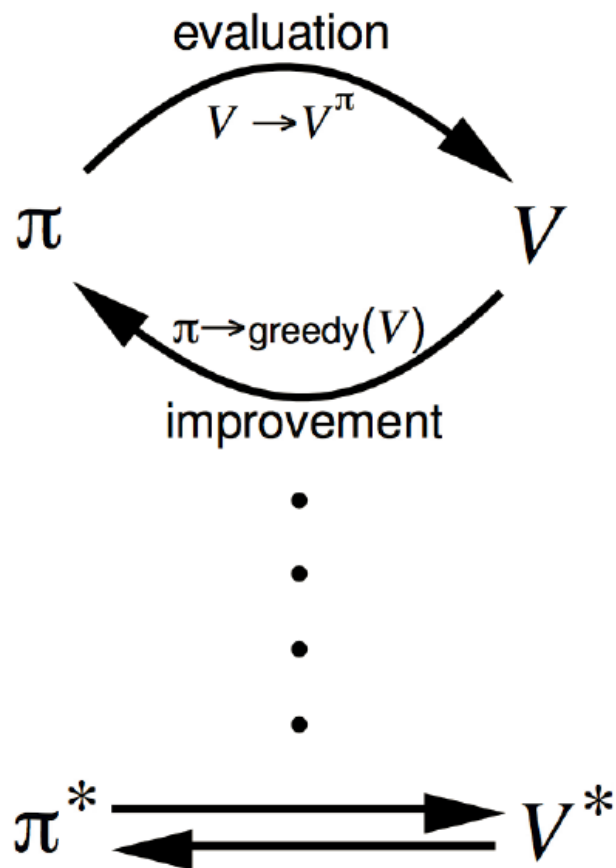


□ 策略评估

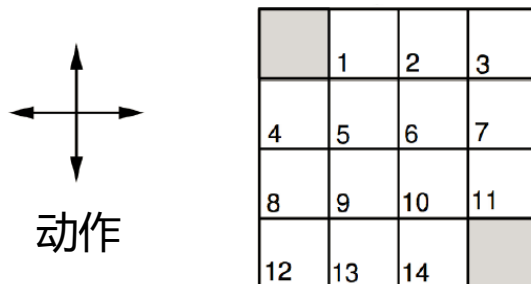
- 估计 V^π
- 迭代的评估策略

□ 策略改进

- 生成 $\pi' \geq \pi$
- 贪心策略改进



举例：策略评估



- 非折扣MDP ($\gamma = 1$)
- 非终止状态：1, 2, ..., 14
- 两个终止状态（灰色方格）
- 如果动作指向所有方格以外，则这一步不动
- 奖励均为-1，直到到达终止状态
- 智能体的策略为均匀随机策略

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

举例：策略评估

K=0

随机策略的 V_k

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

V_k 对应的贪心策略

	↕↕↕	↕↕↕	↕↕↕
↕↕↕	↕↕↕	↕↕↕	↕↕↕
↕↕↕	↕↕↕	↕↕↕	↕↕↕
↕↕↕	↕↕↕	↕↕↕	

K=1

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

	←	↕↕↕	↕↕↕
↑	↕↕↕	↕↕↕	↕↕↕
↕↕↕	↕↕↕	↕↕↕	↓
↕↕↕	↕↕↕	→	

K=2

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

	←	←	↕↕↕
↑	↖	↕↕↕	↓
↑	↕↕↕	↘	↓
↕↕↕	→	→	

举例：策略评估

$K=3$

随机策略的 V_k

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

V_k 对应的贪心策略

	←	←	↙
↑	↖	↘	↓
↑	↗	↘	↓
↖	→	→	

$K=10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

	←	←	↙
↑	↖	↘	↓
↑	↗	↘	↓
↖	→	→	

$K=\infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

	←	←	↙
↑	↖	↘	↓
↑	↗	↘	↓
↖	→	→	

$V := V^\pi$
最优策略

价值迭代 vs. 策略迭代

价值迭代

1. 对每个状态 s , 初始化 $V(s) = 0$

2. 重复以下过程直到收敛 {

对每个状态, 更新

$$V(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V(s')$$

}

策略迭代

1. 随机初始化策略 π

2. 重复以下过程直到收敛 {

a) 让 $V := V^\pi$

b) 对每个状态, 更新

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

}

备注:

1. 价值迭代是贪心更新法
2. 策略迭代中, 用Bellman等式更新价值函数代价很大
3. 对于空间较小的MDP, 策略迭代通常很快收敛
4. 对于空间较大的MDP, 价值迭代更实用 (效率更高)
5. 如果没有状态转移循环, 最好使用价值迭代

强化学习基础

1. 强化学习
2. MDP
3. 动态规划
- 4. 值函数估计**
5. 无模型控制方法
6. 参数化的值函数
7. 参数化的策略

模型无关的强化学习

□ 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，我们仅能观察到交互采样出来的轨迹 (episodes)

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

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

□ 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

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

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

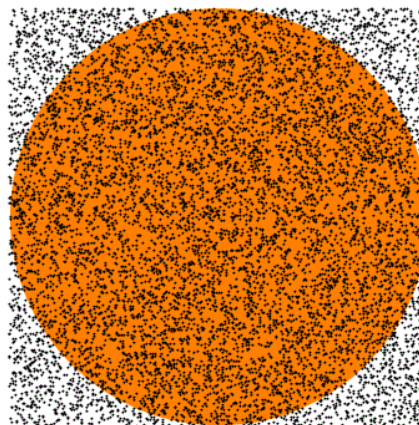
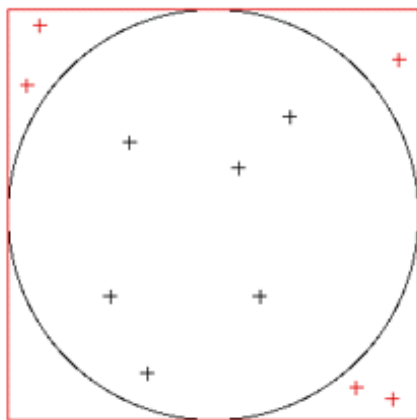
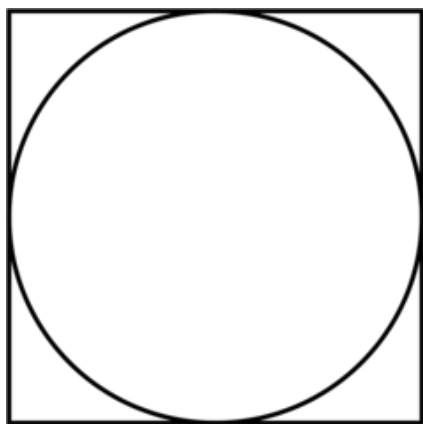
Episode 2: $s_0^{(2)} \xrightarrow[R(s_0)^{(2)}]{a_0^{(2)}} s_1^{(2)} \xrightarrow[R(s_1)^{(2)}]{a_1^{(2)}} s_2^{(2)} \xrightarrow[R(s_2)^{(2)}]{a_2^{(2)}} s_3^{(2)} \dots s_T^{(2)}$

蒙特卡罗方法

□ 蒙特卡罗方法 (Monte-Carlo methods) 是一类依赖于重复随机采样来获得数值结果的计算算法。

- 应用十分普遍，生活中处处都是MC方法。

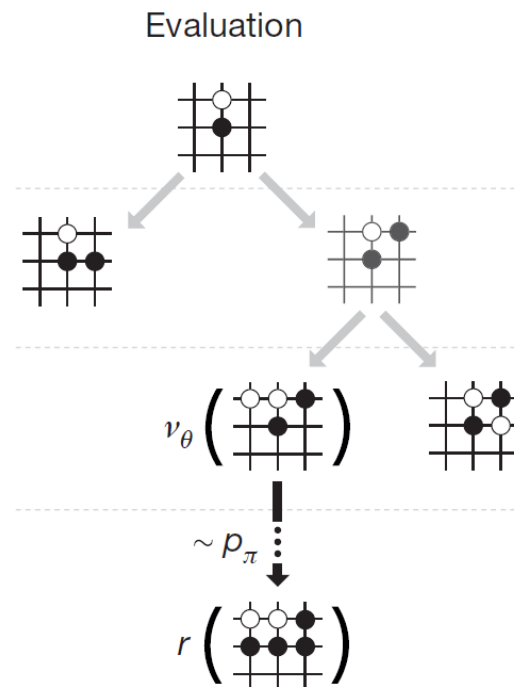
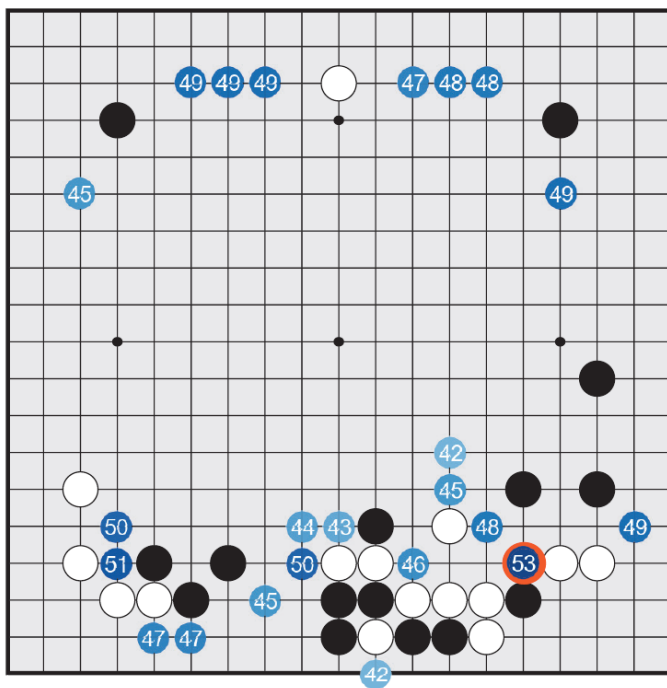
□ 例如，计算圆的面积



$$\text{Circle Surface} = \text{Square Surface} \times \frac{\text{\#points in circle}}{\text{\#points in total}}$$

蒙特卡罗方法

围棋对弈：估计当前状态下的胜率



$$\text{Win Rate}(s) = \frac{\text{\#win simulation cases started from } s}{\text{\#simulation cases started from } s \text{ in total}}$$

蒙特卡罗价值估计

- 目标：从策略 π 下的经验片段学习 V^π

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

- 回顾：累计奖励 (**return**) 是总折扣奖励

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

- 回顾：值函数 (**value function**) 是期望累计奖励

$$\begin{aligned} V^\pi(s) &= \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s, \pi] \\ &= \mathbb{E}[G_t | s_t = s, \pi] \end{aligned}$$

$$\simeq \frac{1}{N} \sum_{i=1}^N G_t^{(i)}$$

- 使用策略 π 从状态 s 采样 N 个片段
- 计算平均累计奖励

- 蒙特卡罗策略评估使用经验均值累计奖励而不是期望累计奖励

蒙特卡罗价值估计

□ 实现

- 使用策略 π 采样片段

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

- 在一个片段中的每个时间步长 t 的状态 s 都被访问
 - 增量计数器 $N(s) \leftarrow N(s) + 1$
 - 增量总累计奖励 $S(s) \leftarrow S(s) + G_t$
 - 价值被估计为累计奖励的均值 $V(s) = S(s)/N(s)$
 - 由大数定律有

$$V(s) \rightarrow V^\pi(s) \text{ as } N(s) \rightarrow \infty$$

增量蒙特卡罗更新

- 每个片段结束后逐步更新 $V(s)$
- 对于每个状态 S_t 和对应累计奖励 G_t

$$N(S_t) \leftarrow N(S_t) + 1$$
$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)} (G_t - V(S_t))$$

- 对于非稳定的问题（即，环境会随时间发生变化），我们可以跟踪一个现阶段的平均值（即，不考虑过久之前的片段）

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

- 蒙特卡罗采用最简单的思想：值（value）= 平均累计奖励（mean return）
- 注意：只能将蒙特卡罗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

时序差分学习

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

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

↑ ↑
观测值 对未来的猜测

- 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)

蒙特卡罗 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罗过程 (MC)

- 更新值函数 $V(S_t)$ 使之接近准确累计奖励 G_t

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

□ 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 $R_{t+1} + \gamma V(S_{t+1})$

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

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

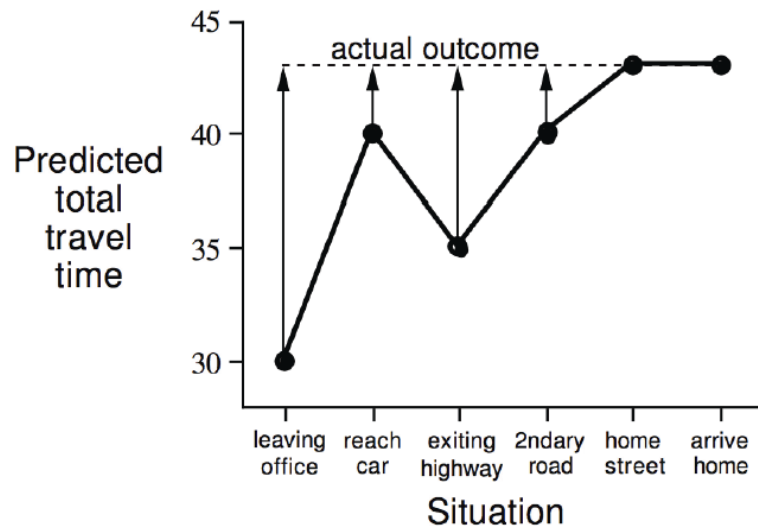
驾车回家的例子



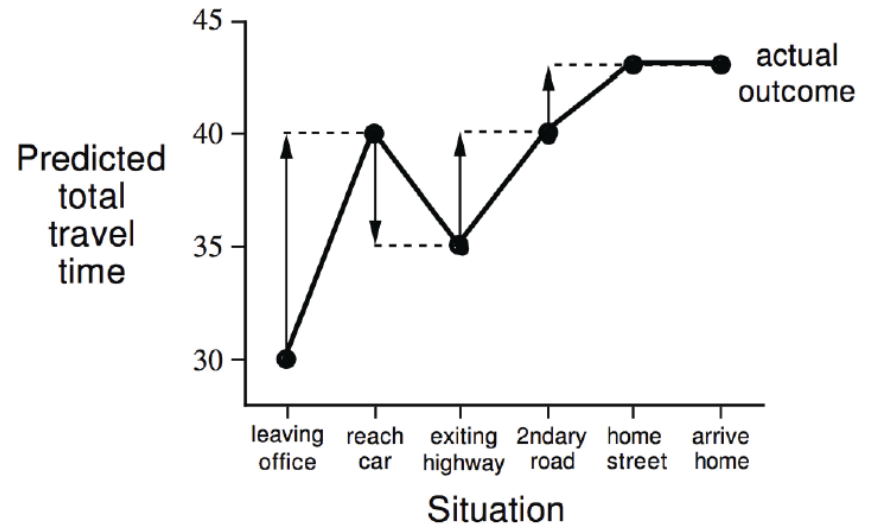
状态	经过的时间 (分钟)	预计所剩时间	预计总时间
离开公司	0	30	30
开始驾车, 下雨	5	35	40
离开高速公路	20	15	35
卡车后跟车	30	10	40
到达家所在街道	40	3	43
直奔家门	43	0	43

驾车回家的例子 (MC vs. TD)

Changes recommended by Monte Carlo methods ($\alpha=1$)



Changes recommended by TD methods ($\alpha=1$)



蒙特卡罗 (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
- 蒙特卡罗必须等待片段结束，直到累计奖励已知

□ 蒙特卡罗：能够无需最后结果地进行学习

- 蒙特卡罗能够从不完整的序列中学习
- 时序差分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
- 蒙特卡罗只能在片段化的（有终止的）环境下工作

蒙特卡罗 (MC) 和时序差分 (TD) 的优缺点 (2)

MC:

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

蒙特卡罗具有高方差，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

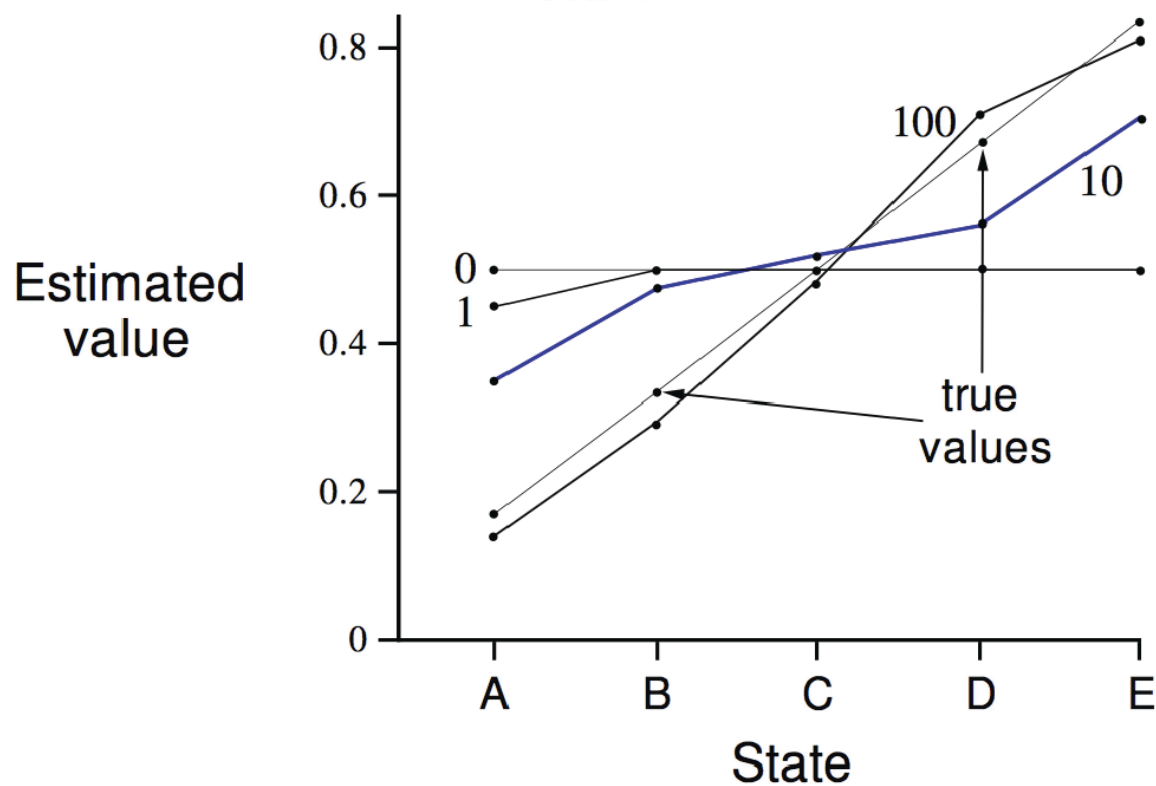
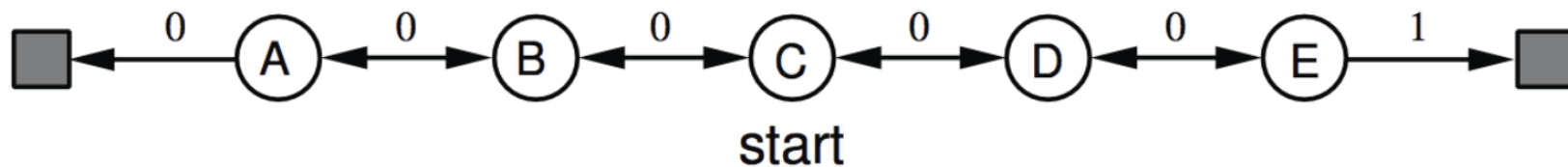
TD:

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

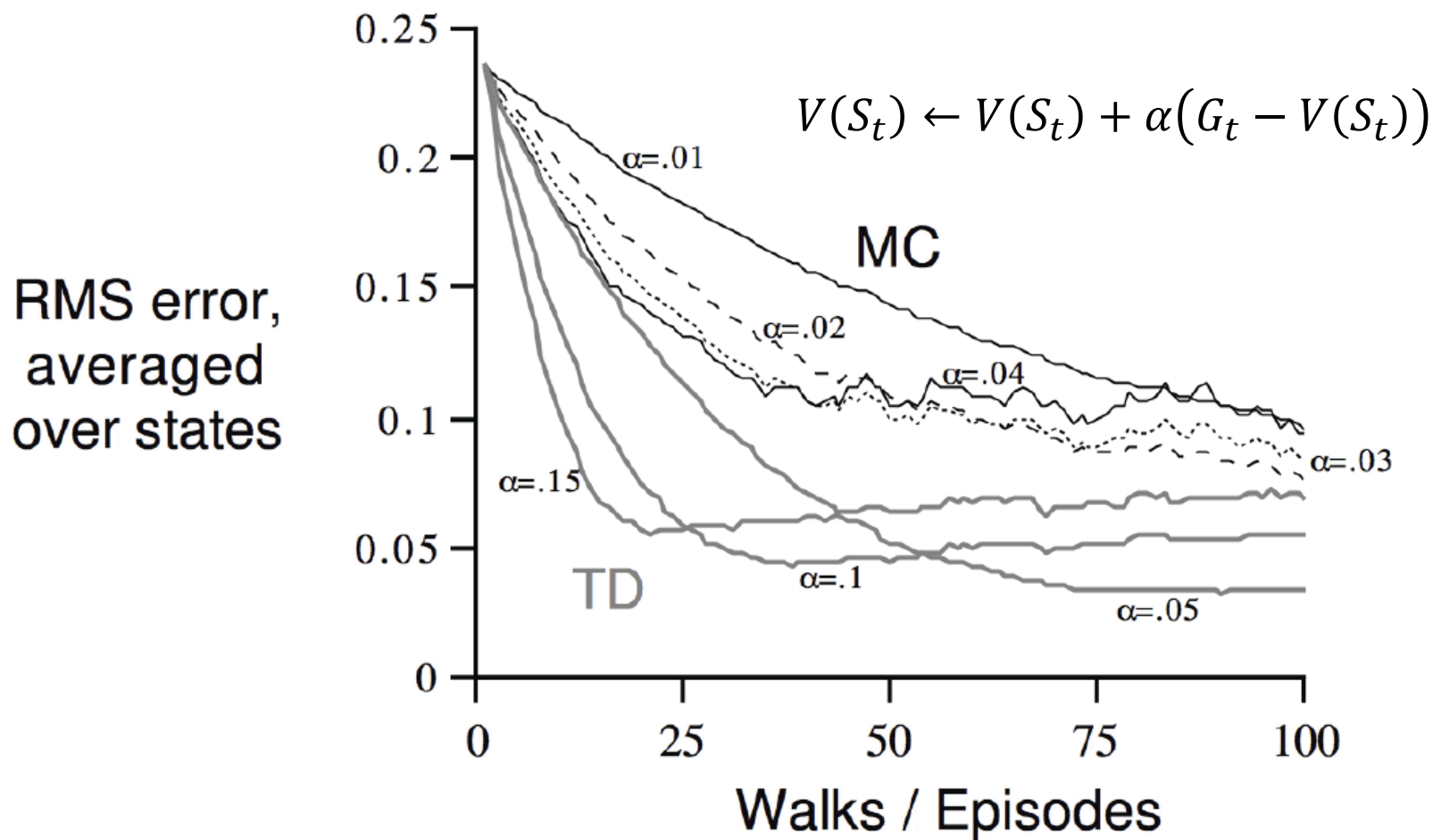
时序差分具有低方差，有偏差

- 通常比蒙特卡罗更加高效
- 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
- 比蒙特卡罗对初始值更加敏感

随机游走的例子

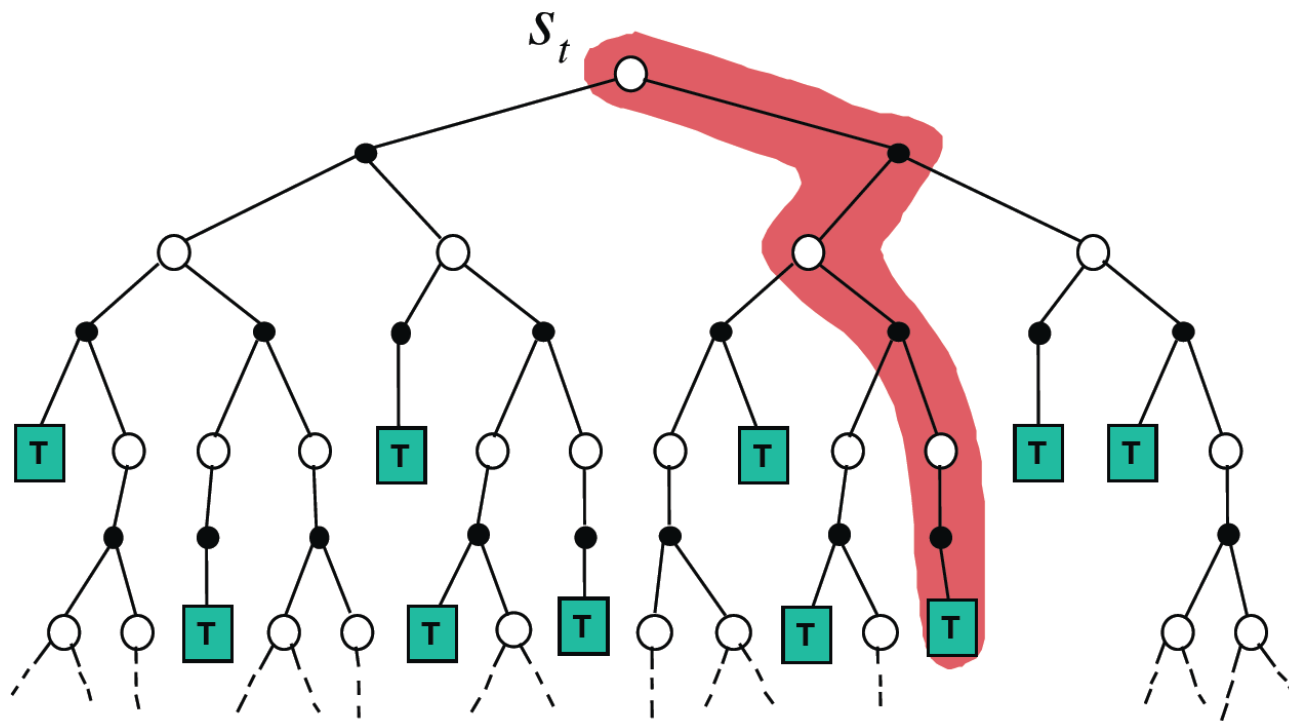


随机游走的例子



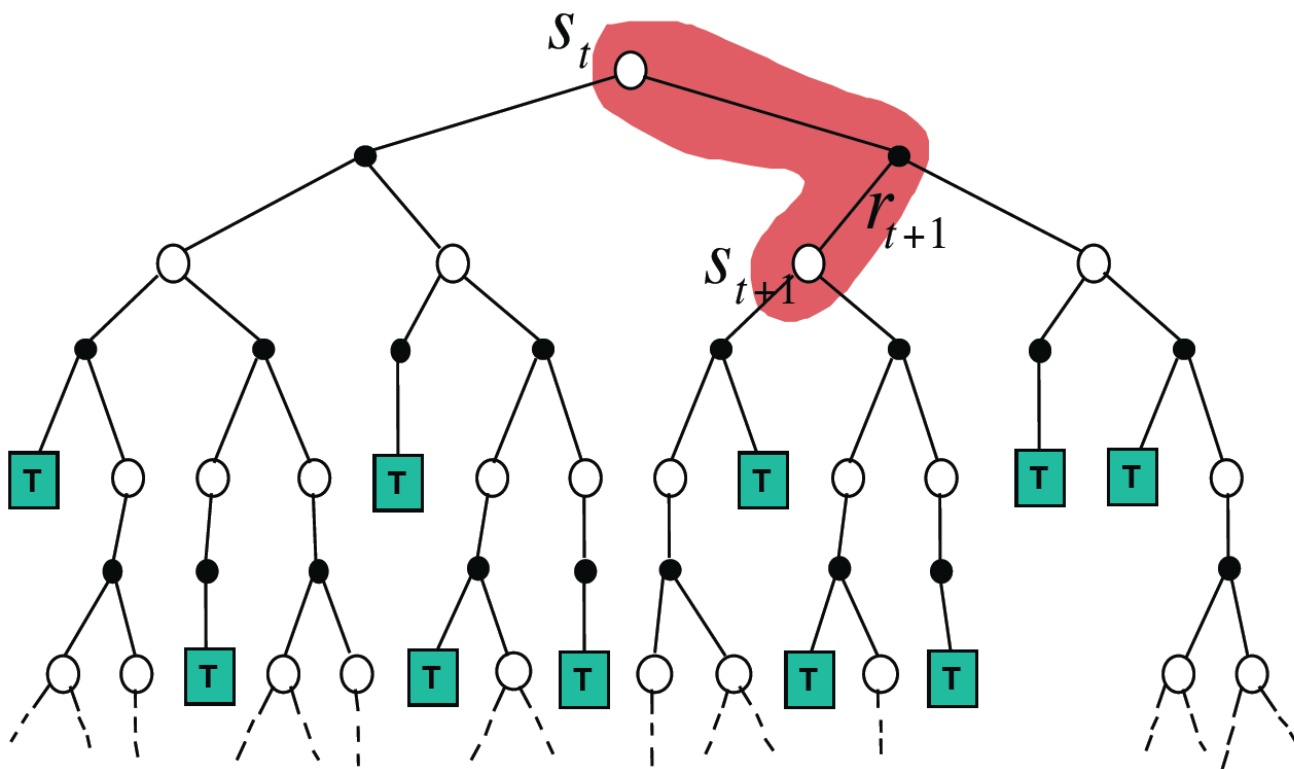
蒙特卡罗的值更新

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



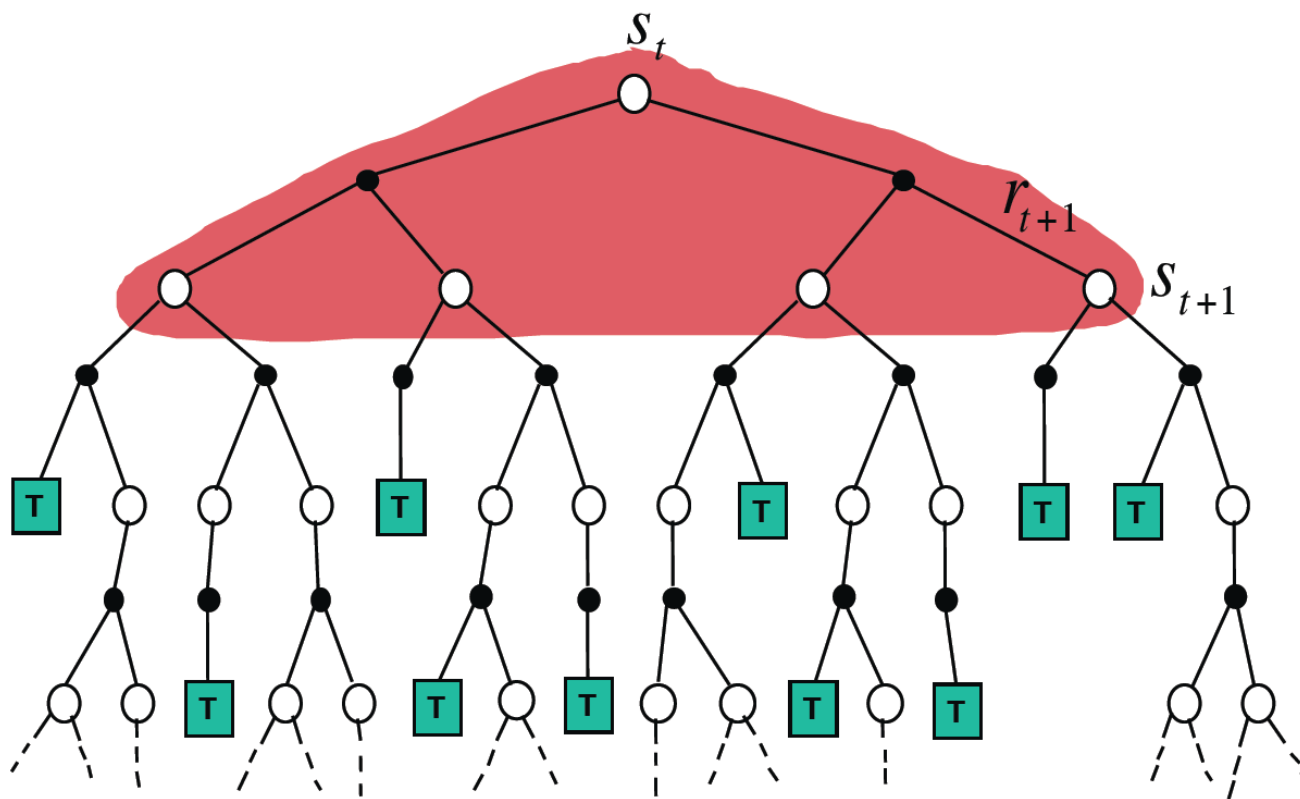
时序差分的值更新

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



动态规划的值更新

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



强化学习基础

1. 强化学习
2. MDP
3. 动态规划
4. 值函数估计
- 5. 无模型控制方法**
6. 参数化的值函数
7. 参数化的策略

动作值函数Q

- 之前我们学习了状态值函数 $V(s)$

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

- 相应的，我们可以定义动作值函数 $Q(s, a)$

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, a_0 = a, \pi] \\ &= \mathbb{E}[R(s_0) + \gamma V(s_1) | s_0 = s, a_0 = a, \pi] \\ &= \mathbb{E}[R(s_0) + \gamma Q(s_1, a_1) | s_0 = s, a_0 = a, \pi] \end{aligned}$$

- 在无模型控制条件下，Q函数更适用

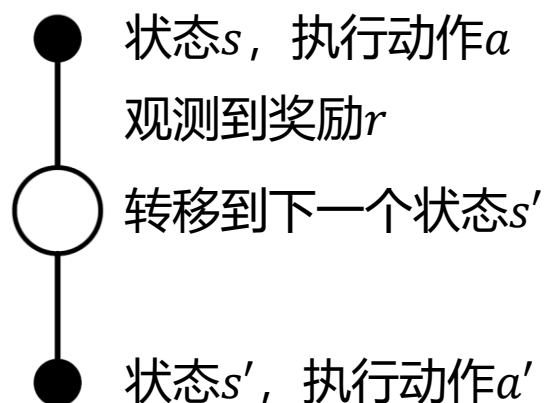
$$\begin{aligned} \pi(s) &= \operatorname{argmax}_{a \in A} \left\{ R(s, a) + \gamma \sum_{s' \in S} P_{sa}(s') V^\pi(s') \right\} \\ \pi(s) &= \operatorname{argmax}_{a \in A} Q^\pi(s, a) \end{aligned}$$

↑
转移概率并不可知

所以接下来的无模型控制部分，我们以Q函数为主

SARSA

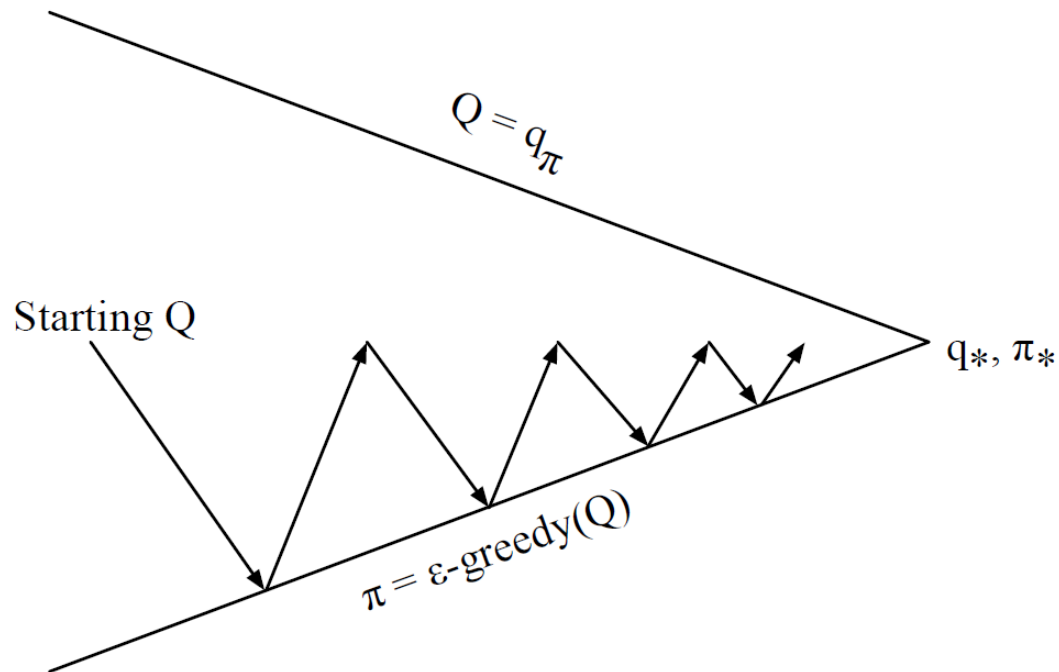
- 对于当前策略执行的每个（状态-动作-奖励-状态-动作）元组



- 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 $Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma Q(s', a') - Q(s, a))$
- 策略改进: ϵ -greedy策略改进
 - 以 $1 - \epsilon$ 概率, 选择 $Q(s, a)$ 最大的行动
 - 以 ϵ 概率, 随机选择任一行动

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

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)

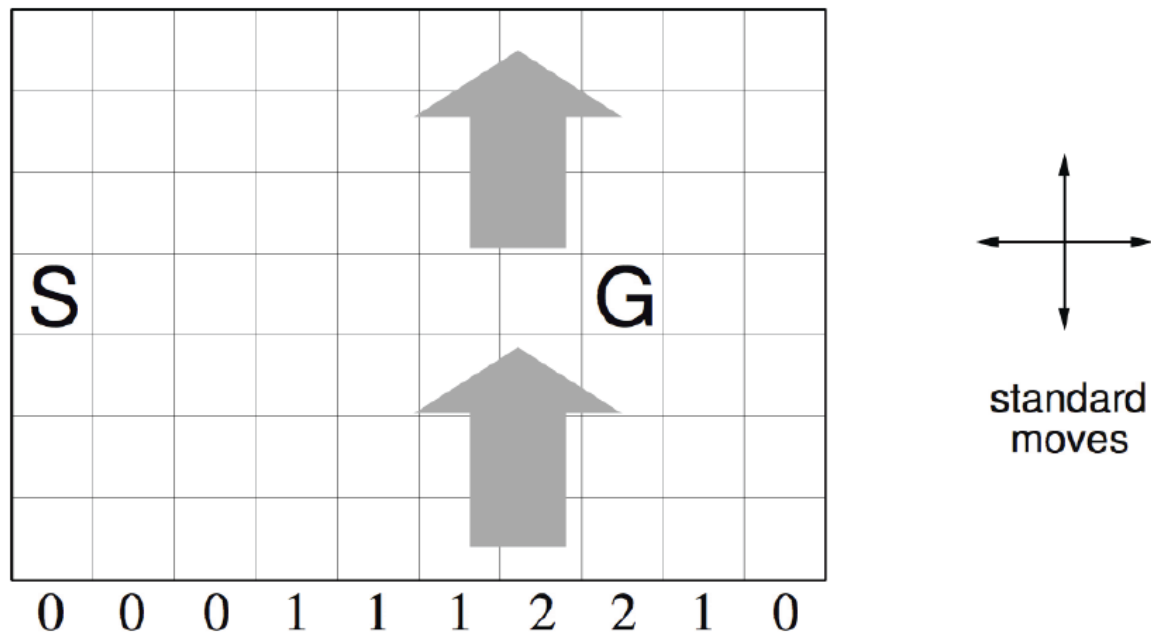
$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$

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

 until S is terminal

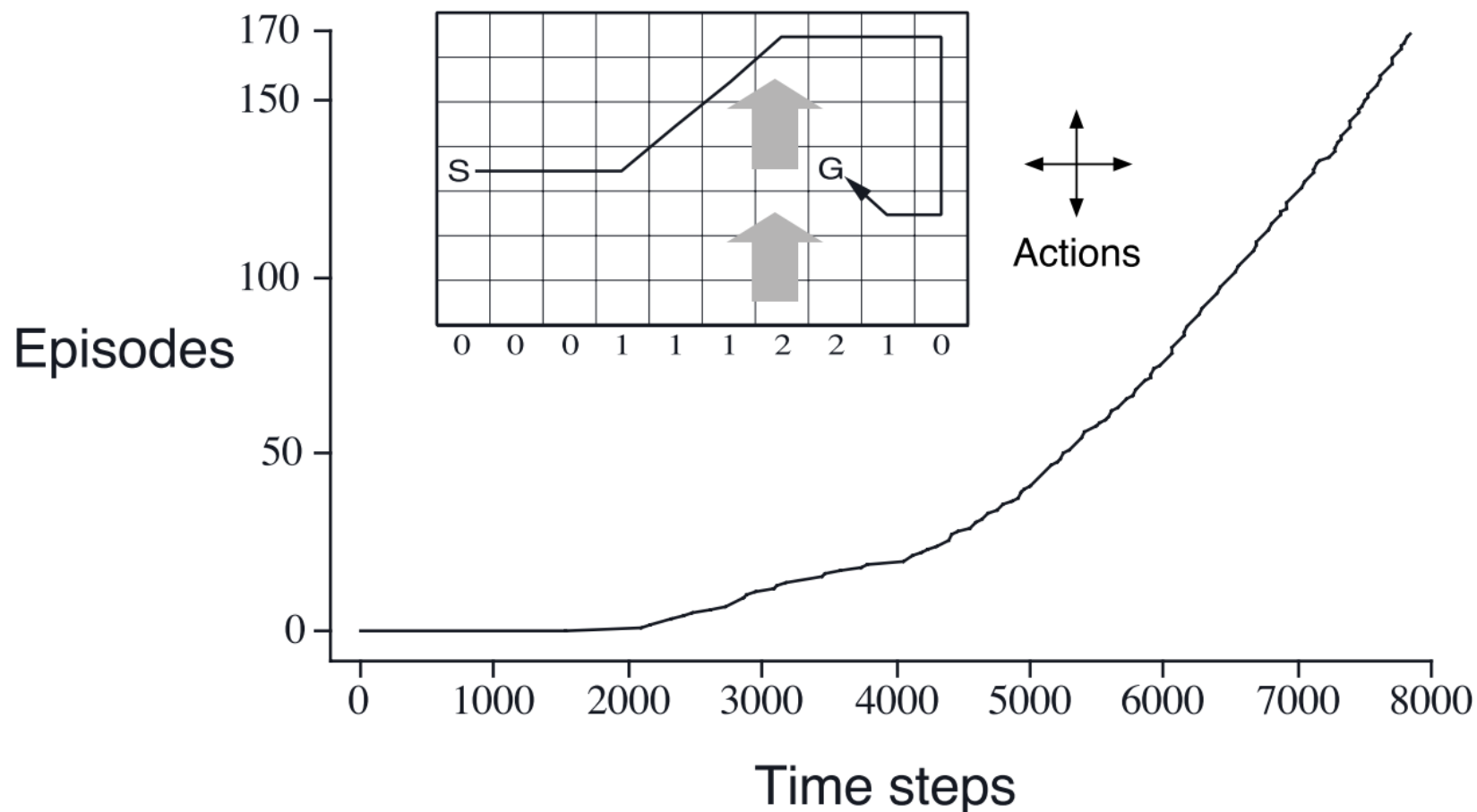
注：在线策略时序差分控制（on-policy TD control）使用当前策略进行动作采样。即，SARSA算法中的两个“A”都是由当前策略选择的

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

- 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， $\mu(\cdot | s_t) \in \mathbb{R}^{|A|}$

动作空间， $a \sim A$

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

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

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 $Q^*(s_t, a_t) = r_t + \gamma Q(s_{t+1}, a'_{t+1})$

- 更新 $Q(s_t, a_t)$ 的值以逼近目标状态-动作值

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



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

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

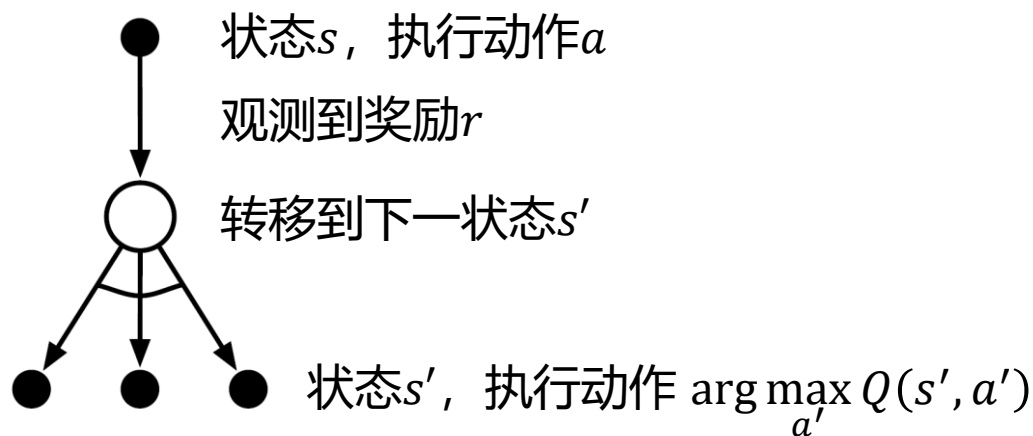
- 行为策略 μ 是关于 $Q(s, a)$ 的 ϵ -贪心策略
- Q-学习目标函数可以简化为

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

- 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

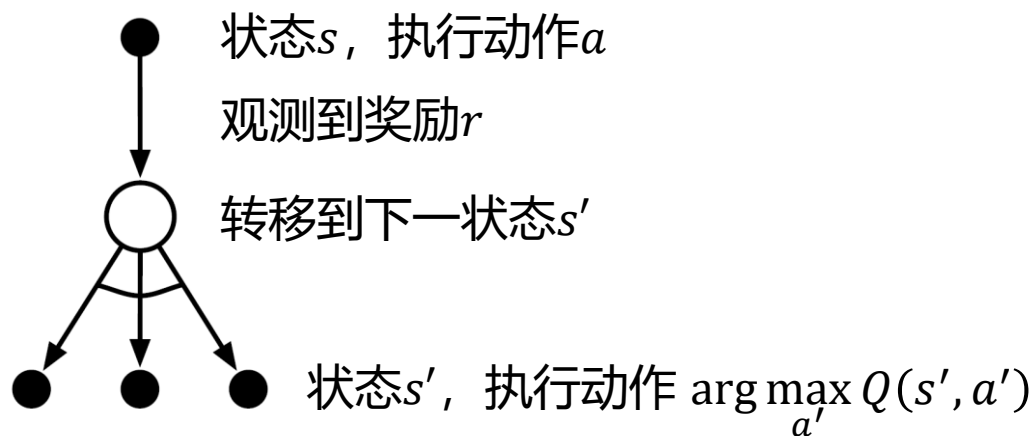


$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'_{t+1}} Q(s_{t+1}, a'_{t+1}) - Q(s_t, a_t))$$

□ 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



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

□ 为什么不需要重要性采样？

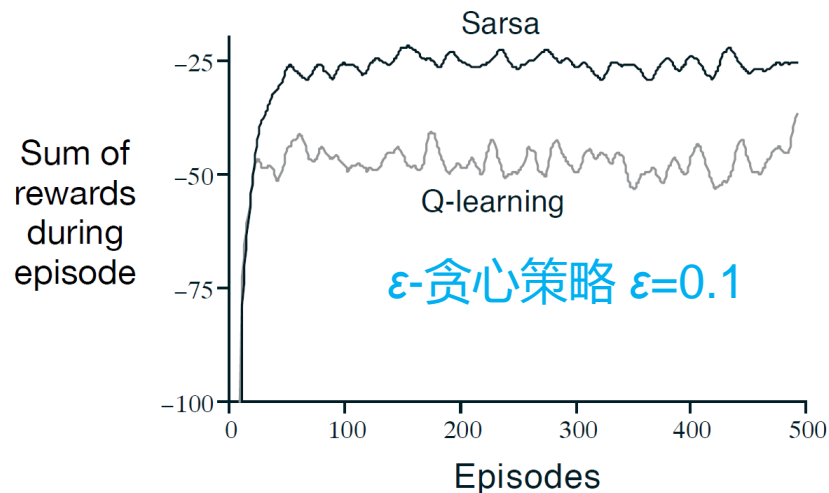
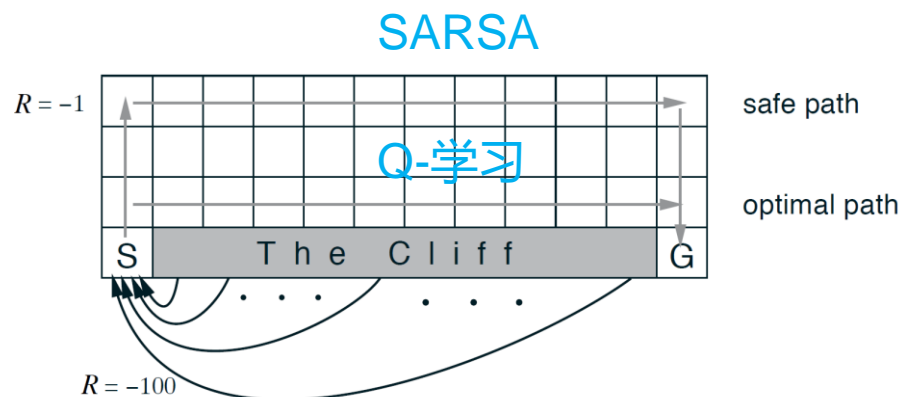
- 使用了状态-动作值函数而不是使用状态值函数

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?



强化学习基础

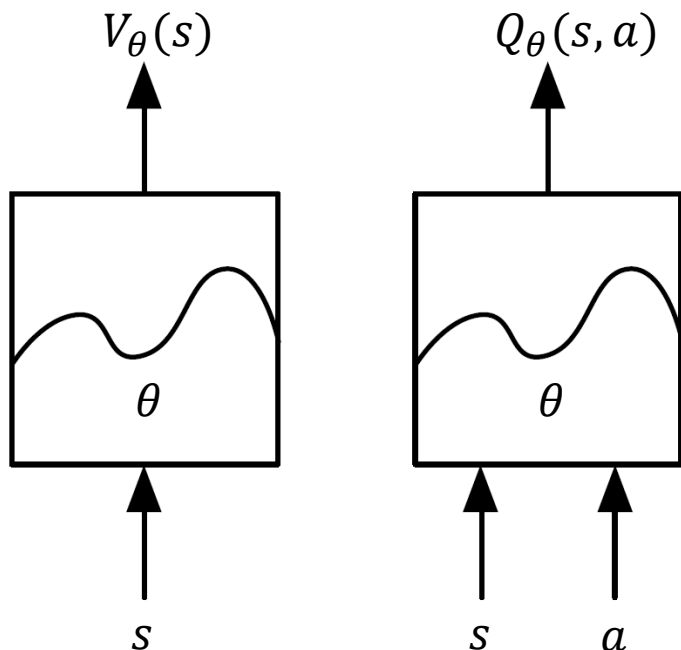
1. 强化学习
2. MDP
3. 动态规划
4. 值函数估计
5. 无模型控制方法
- 6. 参数化的值函数**
7. 参数化的策略

参数化值函数近似

□ 构建参数化（可学习的）函数来近似值函数

$$V_{\theta}(s) \simeq V^{\pi}(s)$$
$$Q_{\theta}(s, a) \simeq Q^{\pi}(s, a)$$

- θ 是近似函数的参数，可以通过强化学习进行更新
- 参数化的方法将现有可见的状态泛化到没有见过的状态上



□ 可微函数

- （一般的）线性模型
- 神经网络

□ 我们希望模型适合在非稳态的，非独立同分布的数据上训练

基于随机梯度下降 (SGD) 的值函数近似

- 目标：找到参数向量 θ 最小化值函数近似值与真实值之间的均方误差

$$J(\theta) = \mathbb{E}_{\pi} \left[\frac{1}{2} (V^{\pi}(s) - V_{\theta}(s))^2 \right]$$

- 误差减小的梯度方向

$$-\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi} \left[(V^{\pi}(s) - V_{\theta}(s)) \frac{\partial V_{\theta}(s)}{\partial \theta} \right]$$

- 单次采样进行随机梯度下降

$$\begin{aligned} \theta &\leftarrow \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \\ &= \theta + \alpha (V^{\pi}(s) - V_{\theta}(s)) \frac{\partial V_{\theta}(s)}{\partial \theta} \end{aligned}$$

特征化状态

- 用一个特征向量表示状态

$$x(s) = \begin{bmatrix} x_1(s) \\ \vdots \\ x_k(s) \end{bmatrix}$$

- 以直升机控制问题为例

- 3D位置
- 3D速度（位置的变化量）
- 3D加速度（速度的变化量）



线性状态值函数近似

- 用特征的线性组合表示价值函数

$$V_{\theta}(s) = \theta^T x(s)$$

- 目标函数是参数 θ 的二次函数

$$J(\theta) = \mathbb{E}_{\pi} \left[\frac{1}{2} (V^{\pi}(s) - \theta^T x(s))^2 \right]$$

- 因而随机梯度下降能够收敛到全局最优解上

$$\begin{aligned} \theta &\leftarrow \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \\ &= \theta + \alpha \underbrace{(V^{\pi}(s) - V_{\theta}(s))}_{\text{预测误差}} \underbrace{x(s)}_{\text{特征值}} \end{aligned}$$

步长 步长 步长

蒙特卡罗状态值函数近似

$$\theta \leftarrow \theta + \alpha(V^\pi(s) - V_\theta(s))x(s)$$

- 我们用 $V^\pi(s)$ 表示真实的目标价值函数
- 在“训练数据”上运用监督学习对价值函数进行预测

$$\langle s_1, G_1 \rangle, \langle s_2, G_2 \rangle, \dots, \langle s_T, G_T \rangle$$

- 对于每个数据样本 $\langle s_t, G_t \rangle$

$$\theta \leftarrow \theta + \alpha(G_t - V_\theta(s))x(s_t)$$

- 蒙特卡罗预测至少能收敛到一个局部最优解
 - 在价值函数为线性的情况下可以收敛到全局最优

时序差分状态值函数近似

$$\theta \leftarrow \theta + \alpha(V^\pi(s) - V_\theta(s))x(s)$$

□ 时序差分算法的目标 $r_{t+1} + \gamma V_\theta(s_{t+1})$ 是真实目标价值 $V_\pi(s_t)$ 的有偏采样

□ 在“训练数据”上运用监督学习

$$\langle s_1, r_2 + \gamma V_\theta(s_2) \rangle, \langle s_2, r_3 + \gamma V_\theta(s_3) \rangle, \dots, \langle s_T, r_T \rangle$$

□ 对于每个数据样本 $\langle s_t, r_{t+1} + \gamma V_\theta(s_{t+1}) \rangle$

$$\theta \leftarrow \theta + \alpha(r_{t+1} + \gamma V_\theta(s_{t+1}) - V_\theta(s))x(s_t)$$

□ 线性情况下时序差分学习（接近）收敛到全局最优解

状态-动作值函数近似

- 对动作-状态值函数进行近似

$$Q_{\theta}(s, a) \simeq Q^{\pi}(s, a)$$

- 最小均方误差

$$J(\theta) = \mathbb{E}_{\pi} \left[\frac{1}{2} (Q^{\pi}(s, a) - Q_{\theta}(s, a))^2 \right]$$

- 在单个样本上进行随机梯度下降

$$\begin{aligned} \theta &\leftarrow \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \\ &= \theta + \alpha (Q^{\pi}(s, a) - Q_{\theta}(s, a)) \frac{\partial Q_{\theta}(s, a)}{\partial \theta} \end{aligned}$$

线性状态-动作值函数近似

- 用特征向量表示状态-动作对

$$x(s, a) = \begin{bmatrix} x_1(s, a) \\ \vdots \\ x_k(s, a) \end{bmatrix}$$

- 线性情况下，参数化后 Q 函数

$$Q_\theta(s, a) = \theta^T x(s, a)$$

- 利用随机梯度下降更新

$$\begin{aligned} \theta &\leftarrow \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \\ &= \theta + \alpha \left(Q^\pi(s, a) - \theta^T x(s, a) \right) x(s, a) \end{aligned}$$

时序差分状态-动作值函数近似

$$\theta \leftarrow \theta + \alpha(Q^\pi(s, a) - Q_\theta(s, a)) \frac{\partial Q_\theta(s, a)}{\partial \theta}$$

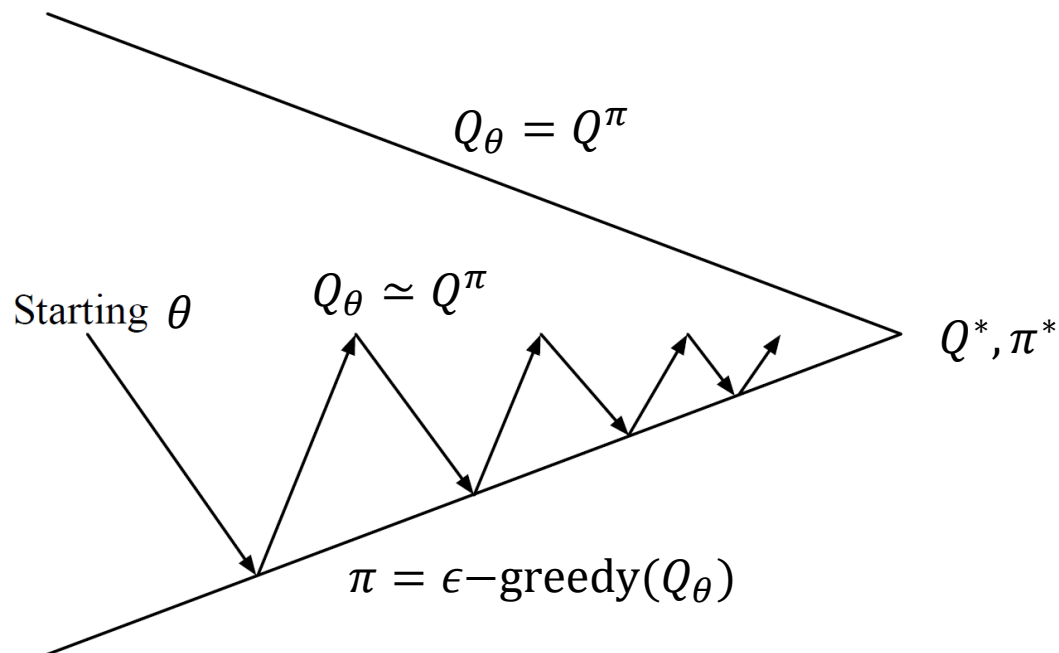
- 对于蒙特卡罗学习，目标是累计奖励 G_t

$$\theta \leftarrow \theta + \alpha(G_t - Q_\theta(s, a)) \frac{\partial Q_\theta(s, a)}{\partial \theta}$$

- 对于时序差分学习，目标是 $r_{t+1} + \gamma Q_\theta(s_{t+1}, a_{t+1})$

$$\theta \leftarrow \theta + \alpha(r_{t+1} + \gamma Q_\theta(s_{t+1}, a_{t+1}) - Q_\theta(s, a)) \frac{\partial Q_\theta(s, a)}{\partial \theta}$$

时序差分状态-动作值函数近似



- 策略评估：近似策略评估 $Q_\theta \simeq Q^\pi$
- 策略改进： ϵ -贪心策略改进

时序差分学习参数更新过程

□ 时序差分学习的目标是

- 状态值函数

$$\begin{aligned}\theta &\leftarrow \theta + \alpha(V^\pi(s_t) - V_\theta(s)) \frac{\partial V_\theta(s_t)}{\partial \theta} \\ &= \theta + \alpha(r_{t+1} + \gamma V_\theta(s_{t+1}) - V_\theta(s)) \frac{\partial V_\theta(s_t)}{\partial \theta}\end{aligned}$$

- 动作-状态值函数

$$\begin{aligned}\theta &\leftarrow \theta + \alpha(Q^\pi(s, a) - Q_\theta(s, a)) \frac{\partial Q_\theta(s, a)}{\partial \theta} \\ &= \theta + \alpha(r_{t+1} + \gamma Q_\theta(s_{t+1}, a_{t+1}) - Q_\theta(s, a)) \frac{\partial Q_\theta(s, a)}{\partial \theta}\end{aligned}$$

- ## □ 虽然 θ 在时序差分学习的目标中出现，但是我们并不需要计算目标函数的梯度。想想这是为什么。

强化学习基础

1. 强化学习
2. MDP
3. 动态规划
4. 值函数估计
5. 无模型控制方法
6. 参数化的值函数
- 7. 参数化的策略**

参数化策略

- 我们能够将策略参数化

$$\pi_{\theta}(a|s)$$

策略可以是确定性的

$$a = \pi_{\theta}(s)$$

也可以是随机的

$$\pi_{\theta}(a|s) = P(a|s; \theta)$$

- θ 是策略的参数
- 将可见的已知状态泛化到未知的状态上
- 在本课程中我们主要讨论的是模型无关的强化学习

基于策略的强化学习

优点

- 具有更好的收敛性质
- 在高维度或连续的动作空间中更有效
 - 最重要的因素：基于值函数的方法，通常需要取最大值
- 能够学习出随机策略

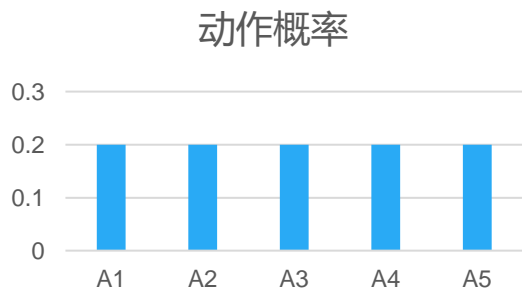
缺点

- 通常会收敛到局部最优而非全局最优
- 评估一个策略通常不够高效并具有较大的方差 (variance)

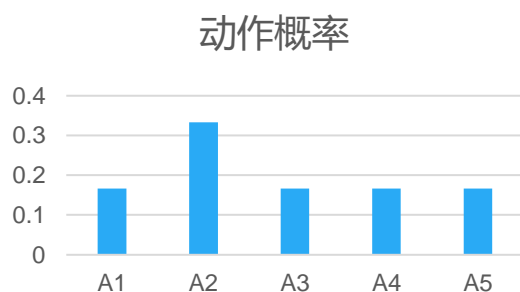
策略梯度

- 对于随机策略 $\pi_{\theta}(a|s) = P(a|s; \theta)$
- 直觉上我们应该
 - 降低带来较低价值/奖励的动作出现的概率
 - 提高带来较高价值/奖励的动作出现的概率
- 一个离散动作空间维度为5的例子

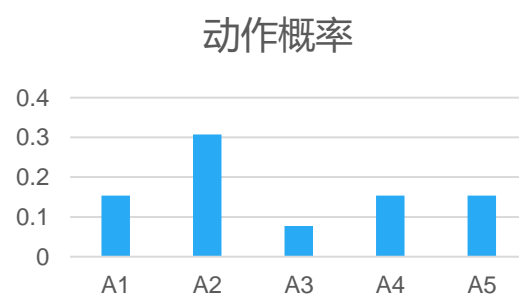
1. 初始化 θ



3. 根据策略梯度更新 θ



5. 根据策略梯度更新 θ



2. 采取动作A2
观察到正的奖励

4. 采取动作A3
观察到负的奖励

单步马尔可夫决策过程中的策略梯度

□ 考虑一个简单的单步马尔可夫决策过程

- 起始状态为 $s \sim d(s)$
- 决策过程在进行一步决策后结束，获得奖励值为 r_{sa}

□ 策略的价值期望

$$J(\theta) = \mathbb{E}_{\pi_{\theta}}[r] = \sum_{s \in S} d(s) \sum_{a \in A} \pi_{\theta}(a|s) r_{sa}$$

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{s \in S} d(s) \sum_{a \in A} \frac{\partial \pi_{\theta}(a|s)}{\partial \theta} r_{sa}$$

似然比 (Likelihood Ratio)

- 似然比利用下列特性

$$\begin{aligned}\frac{\partial \pi_{\theta}(a|s)}{\partial \theta} &= \pi_{\theta}(a|s) \frac{1}{\pi_{\theta}(a|s)} \frac{\partial \pi_{\theta}(a|s)}{\partial \theta} \\ &= \pi_{\theta}(a|s) \frac{\partial \log \pi_{\theta}(a|s)}{\partial \theta}\end{aligned}$$

- 所以策略的价值期望可以写成

$$\begin{aligned}J(\theta) &= \mathbb{E}_{\pi_{\theta}}[r] = \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(a|s) r_{sa} \\ \frac{\partial J(\theta)}{\partial \theta} &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \frac{\partial \pi_{\theta}(a|s)}{\partial \theta} r_{sa} \\ &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(a|s) \frac{\partial \log \pi_{\theta}(a|s)}{\partial \theta} r_{sa} \\ &= \mathbb{E}_{\pi_{\theta}} \left[\frac{\partial \log \pi_{\theta}(a|s)}{\partial \theta} r_{sa} \right]\end{aligned}$$

这一结果可以通过从 $d(s)$ 中采样状态 s 和从 π_{θ} 中采样动作 a 来近似估计

策略梯度定理

- 策略梯度定理把似然比的推导过程泛化到多步马尔可夫决策过程
 - 用长期的价值函数 $Q^{\pi_{\theta}}(s, a)$ 代替前面的瞬时奖励 r_{sa}
- 策略梯度定理涉及
 - 起始状态目标函数 J_1 , 平均奖励目标函数 J_{avR} , 和平均价值目标函数 J_{avV}
- 定理
 - 对任意可微的策略 $\pi_{\theta}(a|s)$, 任意策略的目标函数 $J = J_1, J_{avR}, J_{avV}$, 其策略梯度是

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_{\theta}} \left[\frac{\partial \log \pi_{\theta}(a|s)}{\partial \theta} Q^{\pi_{\theta}}(s, a) \right]$$

详细证明过程请参考 Rich Sutton's Reinforcement Learning: An Introduction (2nd Edition) 第13章

蒙特卡罗策略梯度 (REINFORCE)

- 利用随机梯度上升更新参数
- 利用策略梯度定理
- 利用累计奖励值 G_t 作为 $Q^{\pi_\theta}(s, a)$ 的无偏采样

$$\Delta\theta_t = \alpha \frac{\partial \log \pi_\theta(a_t | s_t)}{\partial \theta} G_t$$

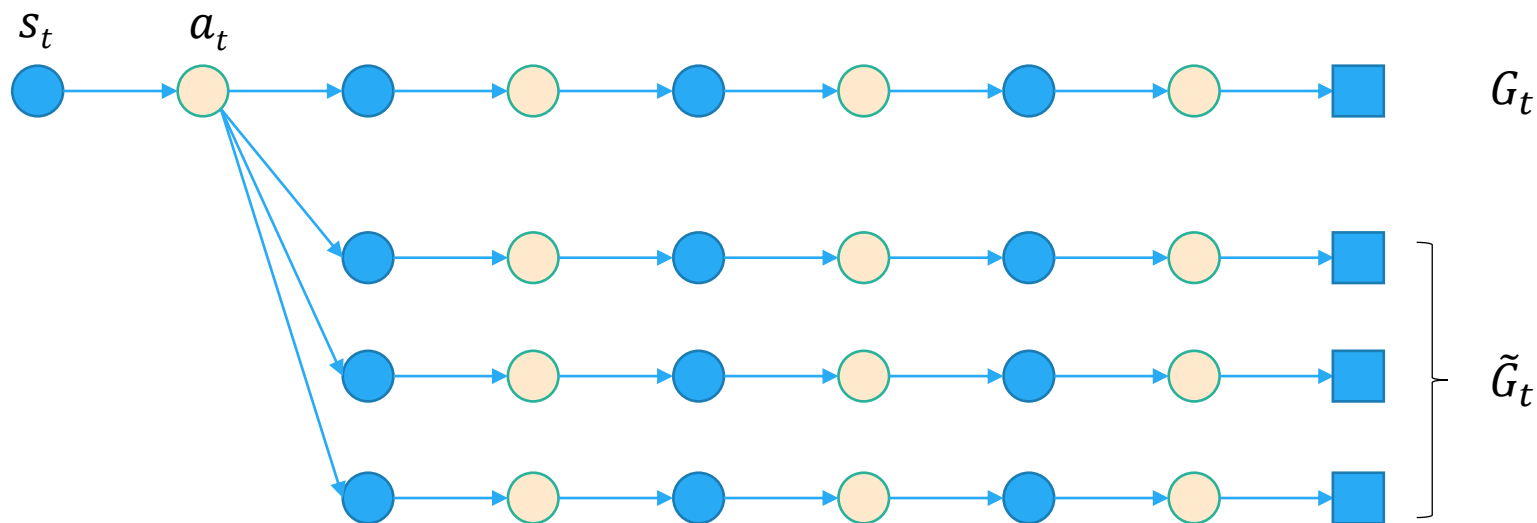
□ REINFORCE算法

```
initialize  $\theta$  arbitrarily
for each episode  $\{s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$  do
    for  $t = 1$  to  $T - 1$  do
         $\theta \leftarrow \theta + \alpha \frac{\partial}{\partial \theta} \log \pi_\theta(a_t | s_t) G_t$ 
    end for
end for
return  $\theta$ 
```

蒙特卡罗策略梯度 (REINFORCE)

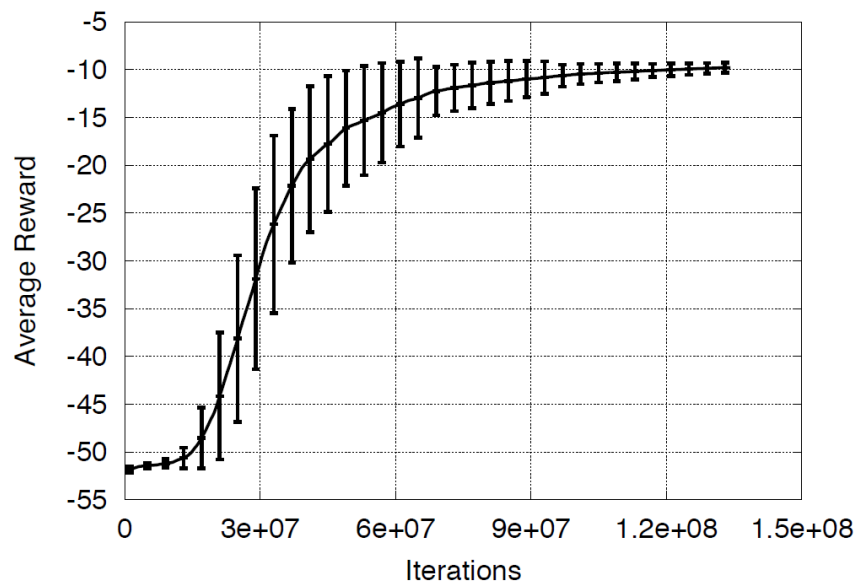
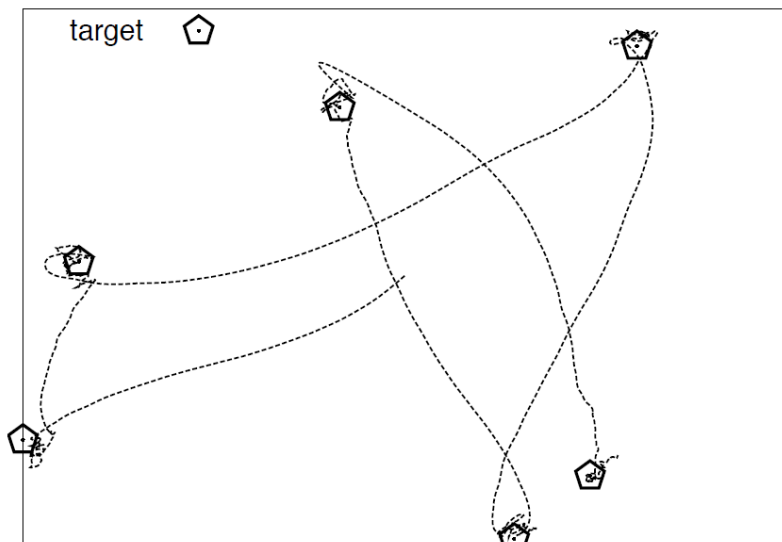
$$\Delta\theta_t = \alpha \frac{\partial \log \pi_{\theta}(a_t | s_t)}{\partial \theta} G_t$$

- 可通过多次roll-out的 G_t 平均值来逼近 $Q(s_t, a_t)$



$$\tilde{G}_t = \frac{1}{N} \sum_{i=1}^n G_t^{(i)}$$

Puck World 冰球世界示例



- 连续的动作对冰球施加较小的力
- 冰球接近目标可以得到奖励
- 目标位置每30秒重置一次
- 使用蒙特卡罗策略梯度方法训练策略

REINFORCE 存在的问题

□ 基于片段式数据的任务

- 通常情况下，任务需要有终止状态，REINFORCE才能直接计算累计折扣奖励

□ 低数据利用效率

- 实际中，REINFORCE需要大量的训练数据

□ 高训练方差（最重要的缺陷）

- 从单个或多个片段中采样到的值函数具有很高的方差

$$\Delta\theta_t = \alpha \frac{\partial \log \pi_{\theta}(a_t | s_t)}{\partial \theta} G_t$$

Actor-Critic

□ Actor-Critic的思想

- REINFORCE策略梯度方法：使用蒙特卡罗采样直接估计 (s_t, a_t) 的值 G_t
- 为什么不建立一个可训练的值函数 Q_Φ 来完成这个估计过程？

□ 演员 (Actor) 和评论家 (Critic)

演员 $\pi_\theta(a|s)$

采取动作使评论家满意的策略



评论家 $Q_\Phi(s, a)$

学会准确估计演员策略所采取动作价值的值函数

Actor-Critic训练

□ 评论家Critic: $Q_{\Phi}(s, a)$

- 学会准确估计当前演员策略 (actor policy) 的动作价值

$$Q_{\Phi}(s, a) \simeq r(s, a) + \gamma \mathbb{E}_{s' \sim p(s'|s, a), a' \sim \pi_{\theta}(a'|s')} [Q_{\Phi}(s', a')]$$

□ 演员Actor: $\pi_{\theta}(a|s)$

- 学会采取使critic满意的动作

$$J(\theta) = \mathbb{E}_{s \sim p, \pi_{\theta}} [\pi_{\theta}(a|s) Q_{\Phi}(s, a)]$$

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_{\theta}} \left[\frac{\partial \log \pi_{\theta}(a|s)}{\partial \theta} Q_{\Phi}(s, a) \right]$$

A2C: Advantage Actor-Critic

□ 思想：通过减去一个基线函数来标准化评论家的打分

- 更多信息指导：降低较差动作概率，提高较优动作概率
- 进一步降低方差

□ 优势函数 (Advantage Function)

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$$



A2C: Advantageous Actor-Critic

□ 状态-动作值和状态值函数

$$\begin{aligned} Q^\pi(s, a) &= r(s, a) + \gamma \mathbb{E}_{s' \sim p(s'|s, a), a' \sim \pi_\theta(a'|s')} [Q_\Phi(s', a')] \\ &= r(s, a) + \gamma \mathbb{E}_{s' \sim p(s'|s, a)} [V^\pi(s')] \end{aligned}$$

□ 因此我们只需要拟合状态值函数来拟合优势函数

$$\begin{aligned} A^\pi(s, a) &= Q^\pi(s, a) - V^\pi(s) \\ &= r(s, a) + \gamma \mathbb{E}_{s' \sim p(s'|s, a)} [V^\pi(s') - V^\pi(s)] \\ &\simeq r(s, a) + \gamma (V^\pi(s') - V^\pi(s)) \end{aligned}$$



采样下一个状态 s'

本课总结：强化学习基础

1. 强化学习：面向序列决策任务的机器学习方法
2. MDP：强化学习环境的数学形式化
3. 动态规划：已知环境模型后直接求最优策略
 - 值迭代、策略迭代
4. 值函数估计：未知环境下估计状态的价值
 - 蒙特卡罗方法、时序差分方法
5. 无模型控制方法：未知环境下估计动作的价值并执行
 - On-policy SARSA、Off-policy Q-learning
6. 参数化的值函数
 - 价值更新 -> 价值函数参数更新
7. 参数化的策略
 - 策略梯度和Actor-Critic
8. 深度强化学习



Network Going Deeper