2022《人工智能》本科生课程

强化学习

教学课程组

2022年

- 参考教材：吴飞，《人工智能导论：模型与算法》，高等教育出版社
- 在线课程(MOOC)：https://www.icourse163.org/course/ZJU-1003377027
- 在线实训平台（智海-Mo）：https://mo.zju.edu.cn/classroom/class/zju_ai_2022
- 系列科普读物《走进人工智能》 https://www.ximalaya.com/album/56494803

提纲

一、强化学习问题定义

二、基于价值的强化学习

三、基于策略的强化学习

四、深度强化学习的应用

强化学习中的概念

智能体（agent）：智能体是强化学习算法的主体，它能够根据经验做出主观判断并执行动作，是整个智能系统的核心。

环境（environment）：智能体以外的一切统称为环境，环境在与智能体的交互中，能被智能体所采取的动作影响，同时环境也能向智能体反馈状态和奖励。

状态（state）：状态可以理解为智能体对环境的一种理解和编码，通常包含了对智能体所采取决策产生影响的信息。

动作（action）：动作是智能体对环境产生影响的方式。

策略（policy）：策略是智能体在所处状态下去执行某个动作的依据，即给定一个状态，智能体可根据一个策略来选择应该采取的动作。

奖励（reward）：奖励是智能体序贯式采取一系列动作后从环境获得的收益。

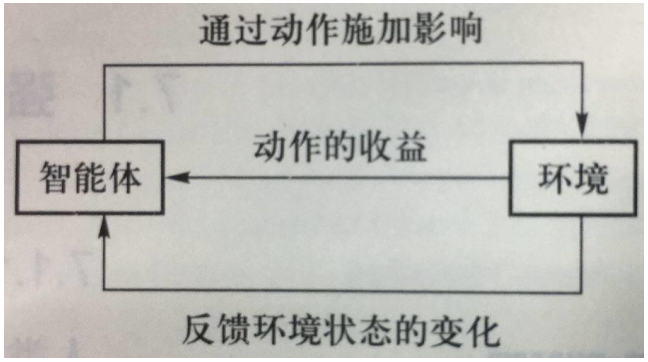


图7.1 强化学习的过程

强化学习的特点

表7.1 三种学习方式特点对比

	有监督学习	无监督学习	强化学习
学习依据	基于监督信息	基于对数据结构的假设	基于评估
数据来源	一次给定	一次给定	在时序交互中产生
决策过程	单步决策 （如分类和识别等）	无	序贯决策 （如棋类博弈）
学习目标	样本到语义标签的映射	数据的分布模式	选择能够获取最大收益的状态到动作的映射

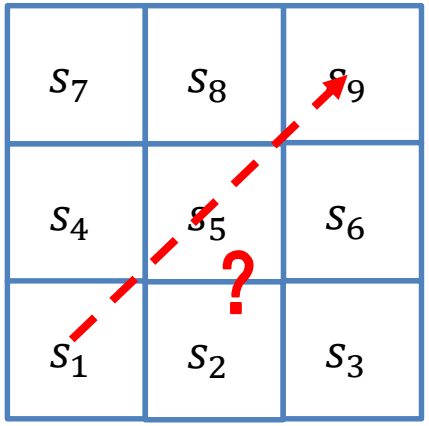
- **基于评估**：强化学习利用环境评估当前策略，以此为依据进行优化
- **交互性**：强化学习的数据在与环境的交互中产生
- **序列决策过程**：智能主体在与环境的交互中需要作出一系列的决策，这些决策往往是前后关联的

注：现实中强化学习问题往往还具有奖励滞后，基于采样的评估等特点

马尔可夫决策过程

（序列优化）问题：

- 在下图网格中，假设有一个机器人位于s_1，其每一步只能向上或向右移动一格，跃出方格会被惩罚（且游戏停止）
- 如何使用强化学习找到一种策略，使机器人从s_1到达s_9？



刻画解该问题的因素

智能主体	迷宫机器人
环境	3×3方格
状态	机器人当前时刻所处方格
动作	每次移动一个方格
奖励	到达s_9时给予奖励；越界时给予惩罚

离散马尔可夫过程（Discrete Markov Process）

- 一个随机过程实际上是一列随时间变化的随机变量，其中当时间是离散量时，一个随机过程可以表示为$\{X_t\}_{t=0,1,2,\cdots}$，其中每个X_t都是一个随机变量，这被称为离散随机过程
- 马尔可夫链（Markov Chain）：满足马尔可夫性（Markov Property）的离散随机过程，也被称为离散马尔科夫过程。

$$Pr(X_{t+1} = x_{t+1} | X_0 = x_0, X_1 = x_1, \cdots, X_t = x_t) = Pr(X_{t+1} = x_{t+1} | X_t = x_t)$$

$t+1$时刻状态仅与t时刻状态相关

马尔可夫奖励过程（Markov Reward Process）：引入奖励

为了在序列决策中对目标进行优化，在马尔可夫随机过程框架中加入了奖励机制：

- 奖励函数$\boldsymbol{R}: \boldsymbol{S} \times \boldsymbol{S} \mapsto \mathbb{R}$，其中$\boldsymbol{R}(\boldsymbol{S_t}, \boldsymbol{S_{t+1}})$描述了从第$\boldsymbol{t}$步状态转移到第$\boldsymbol{t+1}$步状态所获得奖励
- 在一个序列决策过程中，不同状态之间的转移产生了一系列的奖励$(\boldsymbol{R_1}, \boldsymbol{R_2}, \cdots)$，其中$\boldsymbol{R_{t+1}}$为$\boldsymbol{R}(\boldsymbol{S_t}, \boldsymbol{S_{t+1}})$的简便记法。
- 引入奖励机制，这样可以衡量任意序列的优劣，即对序列决策进行评价。

问题：给定两个因为状态转移而产生的奖励序列$(\boldsymbol{1}, \boldsymbol{1}, \boldsymbol{0}, \boldsymbol{0})$和$(\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{1}, \boldsymbol{1})$，哪个序列决策更好?

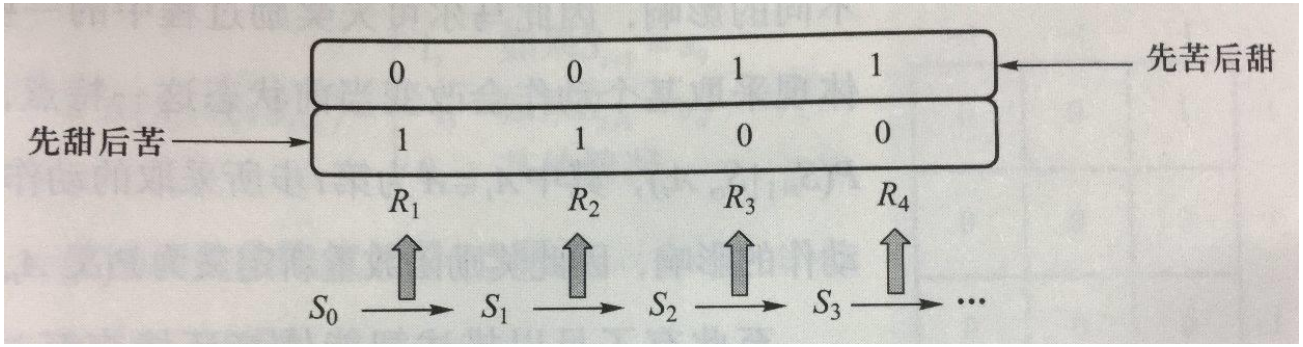


图7.3 两个奖励序列的比较

马尔可夫奖励过程（Markov Reward Process）

问题：给定两个因为状态转移而产生的奖励序列$(1,1,0,0)$和$(0,0,1,1)$，哪个奖励序列更好?

为了比较不同的奖励序列，定义反馈（return），用来反映累加奖励：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

其中折扣系数（discount factor）$\gamma \in [0,1]$

反馈值反映了某个时刻后所得到累加奖励，当衰退系数小于1时，越是遥远的未来对累加反馈的贡献越少

假设 $\gamma = 0.99$

(1,1,0,0): $G_0 = 1 + 0.99 \times 1 + 0.99^2 \times 0 + 0.99^3 \times 0 = 1.99$

(0,0,1,1): $G_0 = 0 + 0.99 \times 0 + 0.99^2 \times 1 + 0.99^3 \times 1 = 1.9504$

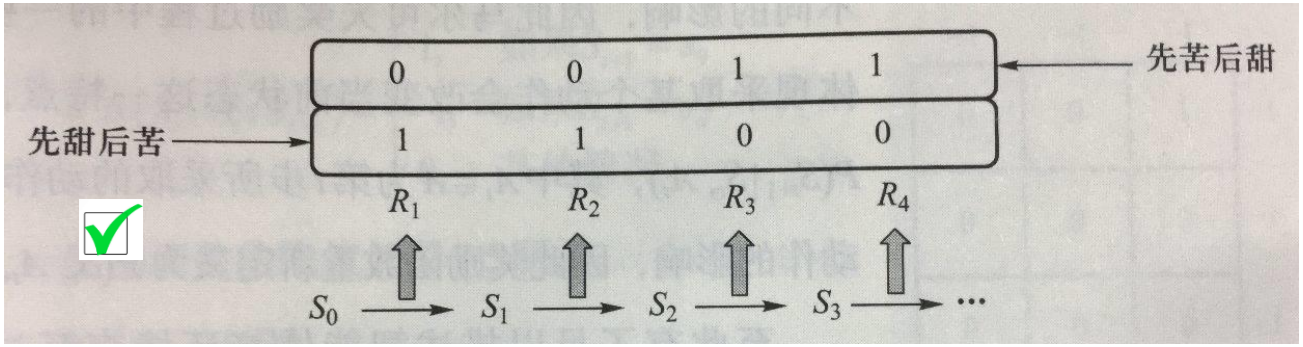


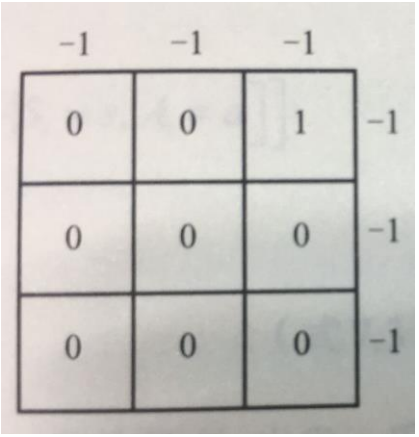
图7.3 两个奖励序列的比较

马尔可夫奖励过程（Markov Reward Process）

使用离散马尔可夫过程描述机器人移动问题

- 随机变量序列$\{S_t\}_{t=0,1,2,\cdots}$：S_t表示机器人第t步的位置，每个随机变量S_t的取值范围为$S=\{s_1,s_2,\cdots,s_9,s_d\}$
- 状态转移概率：$Pr(S_{t+1}|S_t)$满足马尔可夫性
- 定义奖励函数$R(S_t,S_{t+1})$：从S_t到S_{t+1}所获得奖励，其取值如图中所示
- 定义衰退系数：$\gamma\in[0,1]$

图7.5 机器人寻路问题中的奖惩函数



综合以上信息，可用$MRP=\{S,Pr,R,\gamma\}$来刻画马尔科夫奖励过程

这个模型不能体现机器人能动性，仍然缺乏与环境进行进行交互的手段

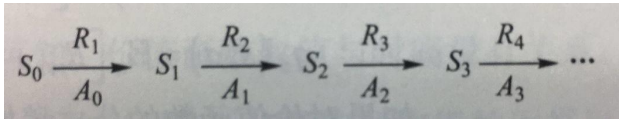
马尔可夫决策过程（Markov Decision Process）：引入动作

在强化学习问题中，智能主体与环境交互过程中可自主决定所采取的动作，不同**动作**会对环境产生不同影响，为此：

- 定义智能主体能够采取的动作集合为A
- 由于不同的动作对环境造成的影响不同，因此状态转移概率定义为$Pr(S_{t+1}|S_t, a_t)$，其中$a_t \in A$为第t步采取的动作
- 奖励可能受动作的影响，因此修改奖励函数为$R(S_t, a_t, S_{t+1})$

- 动作集合A可以是有限的，也可以是无限的
- 状态转移可是确定（deterministic）的，也可以是随机概率性（stochastic）的。
- 确定状态转移相当于发生从S_t到S_{t+1}的转移概率为1

图7.6 智能体与环境的交互过程

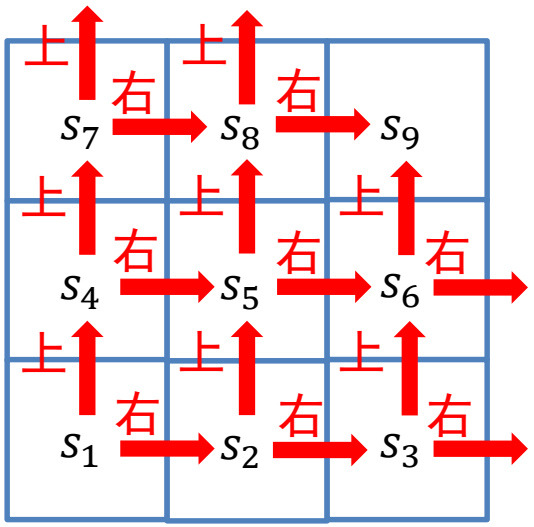


马尔可夫决策过程（Markov Decision Process）

使用离散马尔可夫过程描述机器人移动问题

- 随机变量序列$\{S_t\}_{t=0,1,2,\cdots}$：S_t表示机器人第t步所在位置（即状态），每个随机变量S_t的取值范围为$S = \{s_1, s_2, \cdots, s_9, s_d\}$
- 动作集合：$A = \{$上，右$\}$
- 状态转移概率$Pr(S_{t+1}|S_t, a_t)$：满足马尔可夫性，其中$a_t \in A$。状态转移如图所示。
- 奖励函数：$R(S_t, a_t, S_{t+1})$
- 衰退系数：$\gamma \in [0, 1]$

综合以上信息，可通过$MDP = \{S, A, Pr, R, \gamma\}$来刻画马尔科夫决策过程



马尔可夫决策过程（Markov Decision Process）

- 马尔可夫决策过程$MDP = \{S, A, Pr, R, \gamma\}$是刻画强化学习中环境的标准形式
- 马尔可夫决策过程可用如下序列来表示：

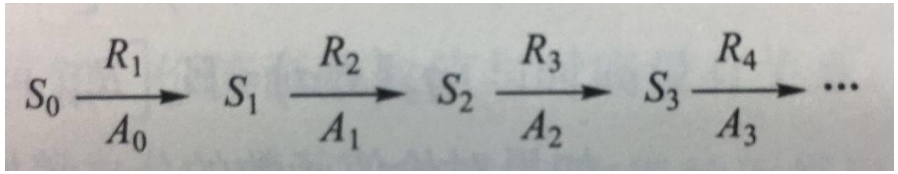


图7.6 智能体与环境的交互过程

马尔科夫过程中产生的状态序列称为轨迹(trajectory)，可如下表示

$$(S_0, a_0, R_1, S_1, a_1, R_2, \cdots, S_T)$$

- 轨迹长度可以是无限的，也可以有终止状态S_T。有终止状态的问题叫做分段的（即存在回合的）（episodic），否则叫做持续的（continuing）

分段问题中，一个从初始状态到终止状态的完整轨迹称为一个片段或回合（episode）。如围棋对弈中一个胜败对局为一个回合。

马尔可夫决策过程（Markov Decision Process）

在机器人移动问题中：状态、行为、衰退系数、起始/终止状态、反馈、状态转移概率矩阵的定义如下

$S = \{s_1, s_2, \cdots, s_9, s_d\}$　起始状态：$\boldsymbol{S_0 = s_1}$

$A = \{$上，右$\}$　　终止状态:$\boldsymbol{S_T \in \{s_9, s_d\}}$

$\gamma = 0.99$

$$R(S_t, a_t, S_{t+1}) = \begin{cases} 1, \text{如果} S_{t+1} = s_9 \\ -1, \text{如果} S_{t+1} = s_d \\ 0, \text{其他情况} \end{cases}$$

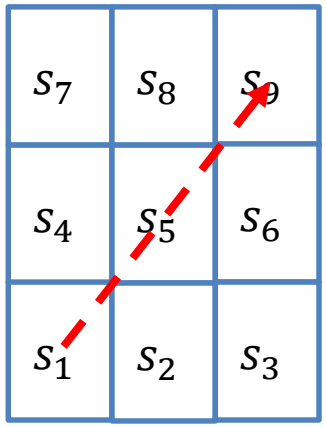
图7.4 机器人寻路问题的状态转移函数

$P(S_{t+1}|S_t, a_t=$右$)$

S_t \ S_{t+1}	S_1	S_2	S_3	…	S_9	S_d
S_1	0	1	0	…	0	0
S_2	0	0	1	…	0	0
S_3	0	0	0	…	0	1
⋮						
S_8	0	0	0	…	1	0
S_9	0	0	0	…	0	1

$P(S_{t+1}|S_t, a_t=$上$)$

S_t \ S_{t+1}	S_1	S_4	S_7	…	S_9	S_d
S_1	0	1	0	…	0	0
S_4	0	0	1	…	0	0
S_7	0	0	0	…	0	1
⋮						
S_6	0	0	0	…	1	0
S_9	0	0	0	…	0	1



如何从起始状态到终止状态？

强化学习中的策略学习

马尔可夫决策过程$MDP = \{S, A, Pr, R, \gamma\}$对环境进行了描述，那么

智能主体如何与环境交互而完成任务？**需要进行策略学习**

策略函数：

- 策略函数$\pi: S \times A \mapsto [0,1]$，其中$\pi(s,a)$的值表示在状态$s$下采取动作$a$的概率。
- 策略函数的输出可以是确定的，即给定s情况下，只有一个动作a使得概率$\pi(s,a)$取值为1。对于确定的策略，记为$a = \pi(s)$。

强化学习问题定义

如何进行策略学习：一个好的策略是在当前状态下采取了一个行动后，该行动能够在未来收到最大化的反馈：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

由马尔可夫性，未来的状态和奖励只与当前状态相关，与t无关。因此t取任意值该等式均成立，如“**逢山开路，遇水搭桥**”。

为了对策略函数π进行评估，定义

- **价值函数（Value Function）**$V: S \mapsto \mathbb{R}$，其中$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$，即在第$t$步状态为$s$时，按照策略$\pi$行动后在未来所获得反馈值的期望
- **动作-价值函数(Action-Value Function)** $q: S \times A \mapsto \mathbb{R}$，其中$q_\pi(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$表示在第$t$步状态为$s$时，按照策略$\pi$采取动作$a$后，在未来所获得反馈值的期望

这样，策略学习转换为如下优化问题：

寻找一个最优策略π^*，对任意$s \in S$使得$V_{\pi^*}(s)$值最大

强化学习问题定义

智能体选择动作的模型：

策略函数$\pi: S \times A \mapsto [0,1]$，$\pi(s,a)$表示智能体在状态$s$下采取动作$a$的概率。

最大化每一时刻的回报值：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

一个好的策略函数应该能够使得智能体在采取了一系列行动后可获得最佳奖励

- **价值函数（value function）**：$V: S \mapsto \mathbb{R}$，其中$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$，表示智能体在时刻$t$处于状态$s$时，按照策略$\pi$采取行动时所获得回报的期望。价值函数衡量了某个状态的好坏程度，反映了智能体从当前状态转移到该状态时能够为目标完成带来多大“好处”。
- **动作-价值函数(action-value function)**：$q: S \times A \mapsto \mathbb{R}$，其中$q_\pi(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$，表示智能体在时刻$t$处于状态$s$时，选择了动作$a$后，在$t$时刻后根据策略$\pi$采取行动所获得回报的期望。

强化学习问题定义

定义： 给定一个马尔可夫决策过程$MDP = (S, A, P, R, \gamma)$，学习一个最优策略$\pi^*$，对任意$s \in S$使得$V_{\pi^*}(s)$值最大。

- 价值函数和动作-价值函数反映了智能体在某一策略下所对应状态序列获得回报的期望，它比回报本身更加准确地刻画了智能体的目标。

- 价值函数和动作-价值函数的定义之所以能够成立，离不开决策过程所具有的马尔可夫性，即当位于当前状态s时，无论当前时刻t的取值是多少，一个策略回报值的期望是一定的（当前状态只与前一状态有关，与时间无关）。

一个好的策略函数应该能够使得智能体在采取了一系列行动后可得到最佳奖励

贝尔曼方程 (Bellman Equation)

贝尔曼方程（Bellman Equation）也被称作动态规划方程（Dynamic Programming Equation），由理查德·贝尔曼（Richard Bellman）提出。

- **价值函数（Value Function）** $V_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle| S_t = s\right]$
- **动作-价值函数(Action-Value Function)**

$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle| S_t = s, A_t = a\right]$$

$$V_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle| S_t = s\right]$$
$$= \mathbb{E}_{a\sim\pi(s,\cdot)}\left[\mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle| S_t = s, A_t = a\right]\right]$$
$$= \sum_{a\in A} \underbrace{\pi(s,a)}_{\text{采取动作}a\text{的概率}} \times \underbrace{q_\pi(s,a)}_{\text{采取动作}a\text{后带来的回报期望}}$$
$$= \sum_{a\in A} \pi(s,a) q_\pi(s,a)$$

贝尔曼方程 (Bellman Equation)

贝尔曼方程（Bellman Equation）也被称作动态规划方程（Dynamic Programming Equation），由理查德·贝尔曼（Richard Bellman）提出。

- **价值函数（Value Function）** $V_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle| S_t = s\right]$
- **动作-价值函数(Action-Value Function)**

$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle| S_t = s, A_t = a\right]$$

$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle| S_t = s, A_t = a\right]$$
$$= \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[R(s,a,s') + \gamma \mathbb{E}_\pi[R_{t+2} + \gamma R_{t+3} + \cdots | S_{t+1} = s']\right]$$
$$= \sum_{s' \in \boldsymbol{S}} \underbrace{P(s'|s,a)}_{\text{在状态s采取行动}a\text{进入状态}s'\text{的概率}} \times [\underbrace{R(s,a,s')}_{\text{在s采取}a\text{进入}s'\text{得到的回报}} + \gamma \times \underbrace{V_\pi(s')}_{\text{在}s'\text{获得的回报期望}}]$$
$$= \sum_{s' \in \boldsymbol{S}} P(s'|s,a)\,[R(s,a,s') + \gamma V_\pi(s')]$$

贝尔曼方程 (Bellman Equation)

贝尔曼方程（Bellman Equation）也被称作动态规划方程（Dynamic Programming Equation），由理查德·贝尔曼（Richard Bellman）提出。

- **价值函数（Value Function）** $V_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle| S_t = s\right]$
- **动作-价值函数(Action-Value Function)**

$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle| S_t = s, A_t = a\right]$$

$$V_\pi(s) = \sum_{a\in A} \pi(s,a) \sum_{s'\in \boldsymbol{S}} P(s'|s,a)\left[R(s,a,s') + \gamma V_\pi(s')\right]$$
$$= \mathbb{E}_{a\sim\pi(s,\cdot)} \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[R(s,a,s') + \gamma V_\pi(s')\right]$$

价值函数取值与时间没有关系，只与策略π、在策略π下从某个状态转移到其后续状态所取得的回报以及在后续所得回报有关。

贝尔曼方程 (Bellman Equation)

贝尔曼方程（Bellman Equation）也被称作动态规划方程（Dynamic Programming Equation），由理查德·贝尔曼（Richard Bellman）提出。

- **价值函数（Value Function）** $V_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle| S_t = s\right]$
- **动作-价值函数(Action-Value Function)**

$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \middle| S_t = s, A_t = a\right]$$

$$q_\pi(s,a) = \sum_{s' \in S} P(s'|s,a)\left[R(s,a,s') + \gamma \sum_{a' \in A} \pi(s',a') q_\pi(s',a')\right]$$

$$= \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[R(s,a,s') + \gamma \mathbb{E}_{a' \sim \pi(s',\cdot)}[q_\pi(s',a')]\right]$$

动作-价值函数取值同样与时间没有关系，而是与瞬时奖励和下一步的状态和动作有关。

价值函数与动作-价值函数的关系：动作-价值函数的贝尔曼方程

价值函数的贝尔曼方程

$$V_\pi(s) = \mathbb{E}_{a\sim\pi(s,\cdot)}\ \mathbb{E}_{s'\sim P(\cdot|s,a)}[R(s,a,s') + \gamma V_\pi(s')]$$

动作-价值函数的贝尔曼方程

$$q_\pi(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}[R(s,a,s') + \gamma \mathbb{E}_{a'\sim\pi(s',\cdot)}[q_\pi(s',a')]]$$

贝尔曼方程描述了价值函数或动作-价值函数的递推关系，是研究强化学习问题的重要手段。其中价值函数的贝尔曼方程描述了当前状态价值函数和其后续状态价值函数之间的关系，即当前状态价值函数等于瞬时奖励的期望加上后续状态的（折扣）价值函数的期望。而动作-价值函数的贝尔曼方程描述了当前动作-价值函数和其后续动作-价值函数之间的关系，即当前状态下的动作-价值函数等于瞬时奖励的期望加上后续状态的（折扣）动作-价值函数的期望。

价值函数与动作-价值函数的关系：动作-价值函数的贝尔曼方程

价值函数的贝尔曼方程

$$V_\pi(s) = \mathbb{E}_{a\sim\pi(s,\cdot)}\ \mathbb{E}_{s'\sim P(\cdot|s,a)}[R(s,a,s') + \gamma V_\pi(s')]$$

动作-价值函数的贝尔曼方程

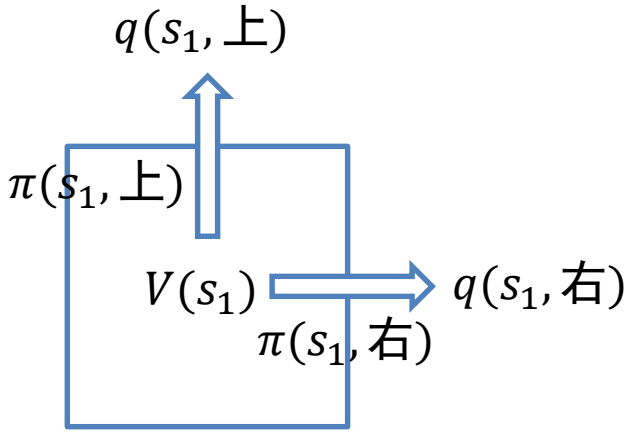
$$q_\pi(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}[R(s,a,s') + \gamma \mathbb{E}_{a'\sim\pi(s',\cdot)}[q_\pi(s',a')]]$$

在实际中，需要计算得到最优策略以指导智能体在当前状态如何选择一个可获得最大回报的动作。求解最优策略的一种方法就是去求解最优的价值函数或最优的动作-价值函数（即基于价值方法，**value-based approach**）。一旦找到了最优的价值函数或动作-价值函数，自然而然也就是找到最优策略。当然，在强化学习中还有基于策略（**policy-based**）和基于模型（**model-based**）等不同方法。

价值函数与动作-价值函数的关系：以状态s_1的计算为例

$$V_\pi(s) = \sum_{a \in A} \pi(s,a) q_\pi(s,a)$$

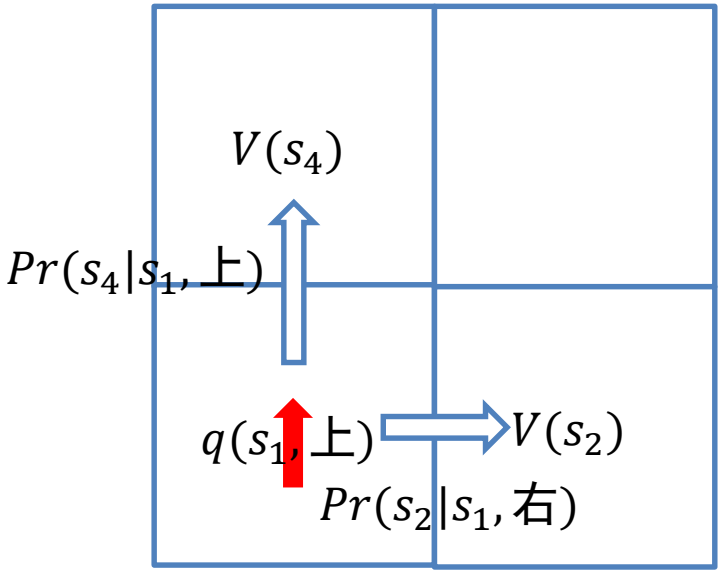
$$V_\pi(s_1) = \pi(s_1, 上) q_\pi(s_1, 上) + \pi(s_1, 右) q_\pi(s_1, 右)$$



不同动作下的反馈累加

$$q_\pi(s,a) = \sum_{s' \in S} Pr(s'|s,a)\,[R(s,a,s') + \gamma V_\pi(s')]$$

$$q_\pi(s_1, 上) = Pr(s_4|s_1, 上)[R(s_1, 上, s_4) + \gamma V_\pi(s_4)]$$



动作确定时状态转移后的反馈结果

提纲

一、强化学习问题定义

二、基于价值的强化学习

三、基于策略的强化学习

四、深度强化学习的应用

策略迭代的基本模式

回顾强化学习问题的定义:

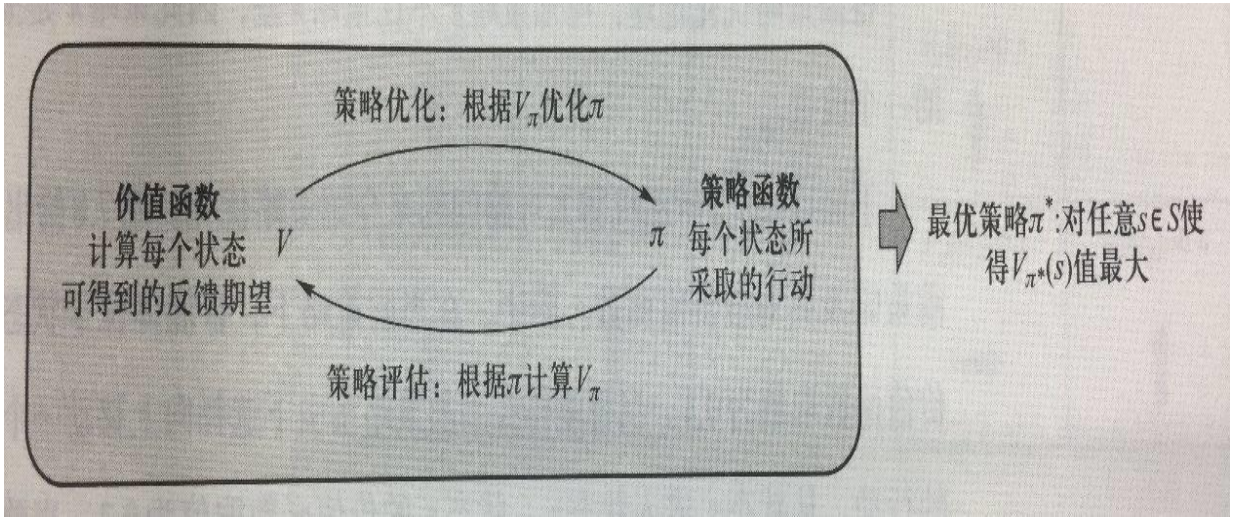
给定一个马尔可夫决策过程$MDP = (S, A, P, R, \gamma)$,

强化学习会寻找一个最优策略π^*，在策略π^*作用下使得任意状态$s \in S$

对应的价值函数$V_{\pi^*}(s)$取值最大。

策略迭代的基本模式

图7.7 基于通用策略迭代来学习最优策略的模式



为了求解最优策略π^*，图7.7展示了一种思路：从一个任意的策略开始，首先计算该策略下价值函数（或动作-价值函数），然后根据价值函数调整改进策略使其更优，不断迭代这个过程直到策略收敛。通过策略计算价值函数的过程叫做策略评估（policy evaluation），通过价值函数优化策略的过程叫做策略优化（policy improvement），策略评估和策略优化交替进行的强化学习求解方法叫做通用策略迭代（Generalized Policy Iteration，GPI）。

强化学习中的策略优化

策略优化定理：

对于确定的策略π和π'，如果对于任意状态$s \in S$

$$q_\pi\big(s,\pi'(s)\big) \geq q_\pi(s,\pi(s))$$

那么对于任意状态$s \in S$，有

$$V_{\pi'}(s) \geq V_\pi(s)$$

即策略π'不比π差

注意，不等式左侧的含义是只在当前这一步将动作修改为$\pi'(s)$，未来的动作仍然按照π的指导进行

在讨论如何优化策略之前，首先需要明确什么是“更好”的策略。分别给出$\boldsymbol{\pi}$和$\boldsymbol{\pi'}$两个策略，如果对于任意状态$\boldsymbol{s \in S}$，有$\boldsymbol{V_\pi(s) \leq V_{\pi'}(s)}$，那么可以认为策略$\boldsymbol{\pi'}$不比策略$\boldsymbol{\pi}$差，可见“更优”策略是一个偏序关系。

强化学习中的策略优化

给定当前策略π、价值函数V_π和行动-价值函数q_π时，可如下构造新的策略π'，只要π'满足如下条件：

$$\pi'(s) = \mathrm{argmax}_a q_\pi(s,a) \quad （对于任意 s \in S）$$

π'便是对π的一个改进。于是对于任意$s \in S$，有

$$q_\pi\big(s,\pi'(s)\big) = q_\pi\big(s,\mathrm{argmax}_a\, q_\pi(s,a)\big)$$

$$= \max_a q_\pi(s,a)$$

$$\geq q_\pi(s,\pi(s))$$

强化学习中的策略优化：机器人寻路问题为例子

图7.8 对状态s_1所采取的行动进行策略优化的示意图

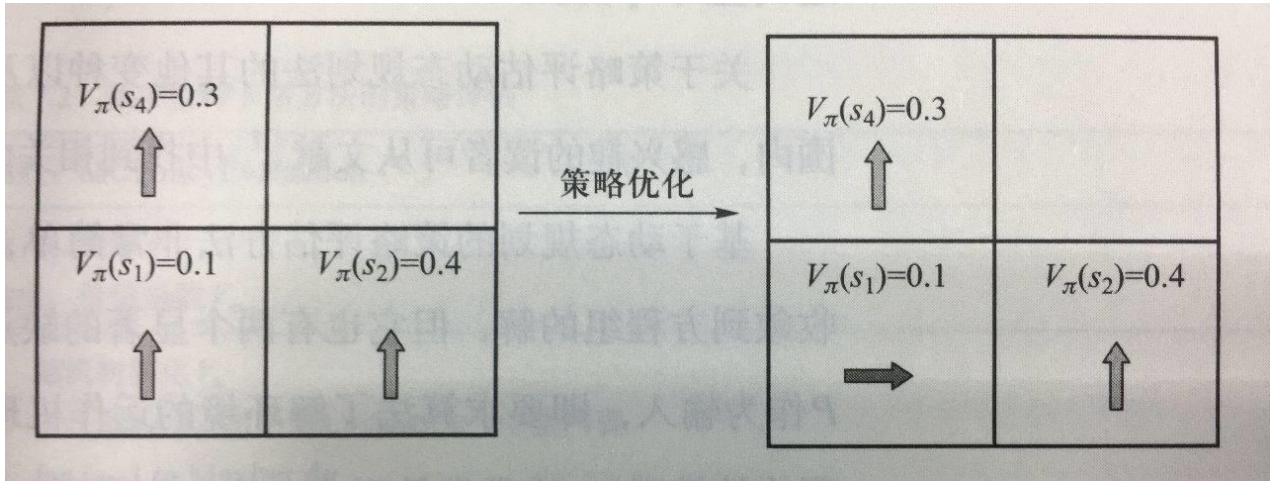
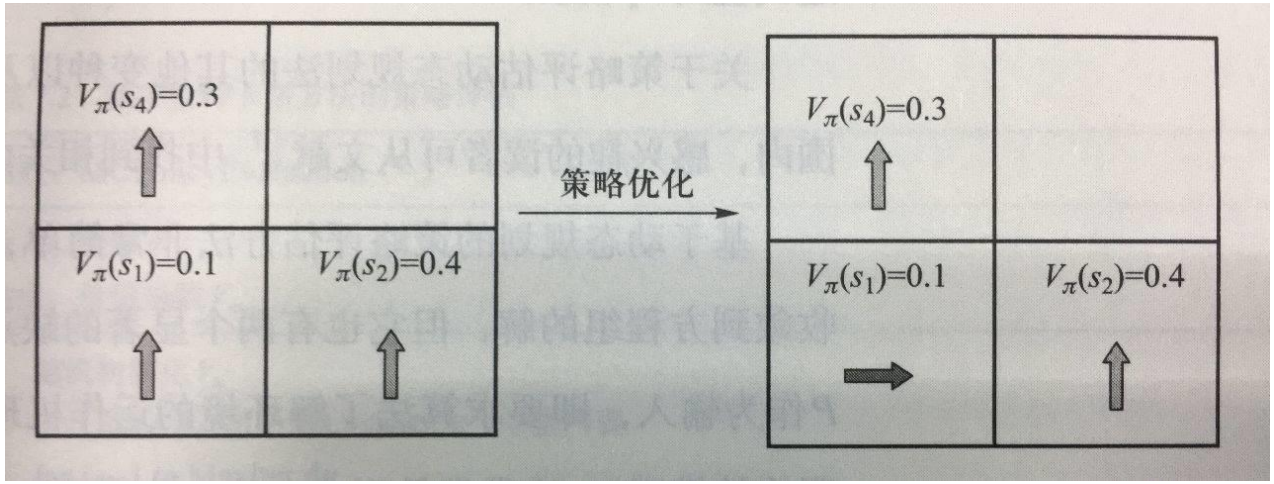


图7.8给出了机器人寻路问题的原有策略及其对应价值函数。在图中，根据原有策略，智能体位于状态s_1的价值函数取值为0.1。当智能体位于s_1状态时，智能体在原有策略指引下将选择向上移动一个方格的行动，从状态s_1进入状态s_4，状态s_4的价值函数取值为0.3。原有策略所给出的其他信息，可从图中周知。

强化学习中的策略优化：机器人寻路问题为例子

图7.8 对状态s_1所采取的行动进行策略优化的示意图

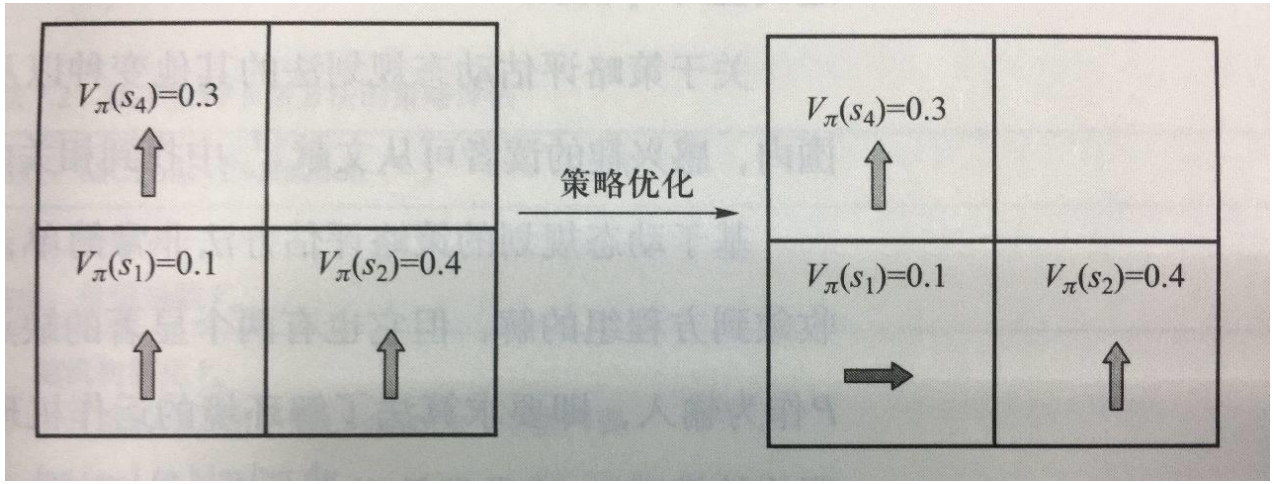


$$q_\pi(s_1, \text{上}) = \sum_{s' \in S} \mathrm{P}(s' | s_1, \text{上}) \left[R(s_1, \text{上}, s') + \gamma V_\pi(s') \right]$$
$$= 1 \times (0 + 0.99 \times 0.3) + 0 \times \cdots = 0.297$$

下面来看智能体如何通过策略优化来改变在状态s_1所采取的行动。由于智能体在状态s_1能够采取“向上移动一个方格”或“向右移动一个方格”两个行动中的一个。首先计算状态s_1选择“向上移动一个方格”后所得动作-价值函数取值。

强化学习中的策略优化：机器人寻路问题为例子

图7.8 对状态s_1所采取的行动进行策略优化的示意图

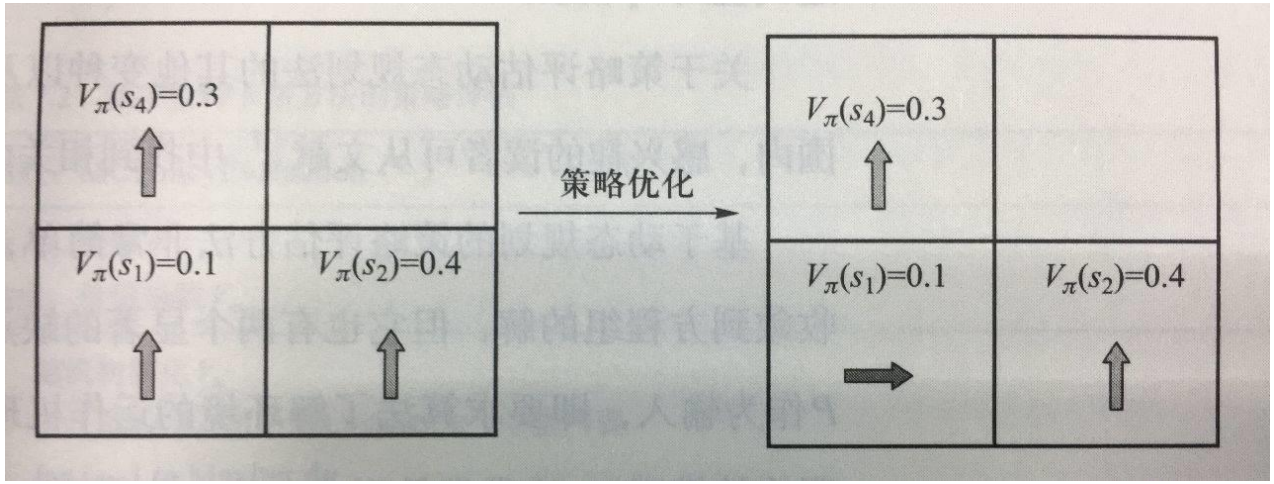


$$q_\pi(s_1, \text{右}) = \sum_{s' \in S} \mathrm{P}(s'|s_1, \text{右}) \left[R(s_1, \text{右}, s') + \gamma V_\pi(s') \right]$$
$$= 1 \times (0 + 0.99 \times 0.4) + 0 \times \cdots = 0.396$$

接着计算状态s_1选择“向右移动一个方格”后所得动作-价值函数取值。

强化学习中的策略优化：机器人寻路问题为例子

图7.8 对状态s_1所采取的行动进行策略优化的示意图



$$q_\pi(s_1, 右) == 0.396 > q_\pi(s_1, 上) = 0.297$$

可见，智能体在状态s_1选择“向上移动一个方格”行动所得回报$q_\pi(s_1, 上)$值为**0.297**、选择“向右移动一个方格”行动所得回报$q_\pi(s_1, 右)$值为**0.396**。显然，智能体在状态s_1应该选择“向右移动一个方格”行动，这样能够获得更大的回报。于是，经过策略优化后，状态s_1处的新策略为$\pi'(s_1) = \text{argmax}_a q_\pi(s, a) = 右$，则将$s_1$处的策略从“上”更新为“右”。其他状态的情况可用类似方法计算得到。

强化学习中的策略评估方法

- 动态规划
- 蒙特卡洛采样
- 时序差分（Temporal Difference）

假定当前策略为π，策略评估指的是根据策略π来计算相应的价值函数V_π或动作-价值函数q_π。这里将介绍在状态集合有限前提下三种常见的策略评估方法，它们分别是基于动态规划的方法、基于蒙特卡洛采样的方法和时序差分（**temporal difference**）法。

强化学习中的策略评估：动态规划

基于动态规划的价值函数更新：使用迭代的方法求解贝尔曼方程组

初始化V_π函数
循环
　　枚举$s \in S$

$$V_\pi(s) \leftarrow \sum_{a \in A} \pi(s,a) \underbrace{\sum_{s' \in S} Pr(s'|s,a)\left[R(s,a,s') + \gamma V_\pi(s')\right]}_{q_\pi(s,a)}$$

直到V_π收敛

更新$V_\pi(s_1)$的值：

$$q_\pi(s_1, 上) = 1 \times (0 + 0.99 \times 0.3) + 0 \times (0 + 0.99 \times 0.4)$$

$$+\ldots = 0.297$$

$$V_\pi(s_1) = 1 \times q_\pi(s_1, 上) + 0 \times q_\pi(s_1, 右) = 0.297$$

- **动态规划法的缺点：**

1) 智能主体需要事先知道状态转移概率;

2) 无法处理状态集合大小无限的情况

强化学习中的策略评估：动态规划

更新$V_\pi(s_1)$的值：

$$q_\pi(s_1, 上) = 1 \times (0 + 0.99 \times 0.3) + 0 \times (0 + 0.99 \times 0.4)$$

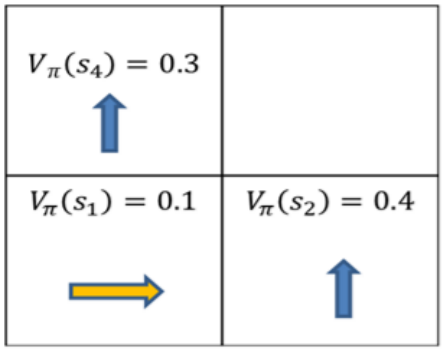
$$+ \ldots = 0.297$$

$$V_\pi(s_1) = 1 \times q_\pi(s_1, 上) + 0 \times q_\pi(s_1, 右) = 0.297$$

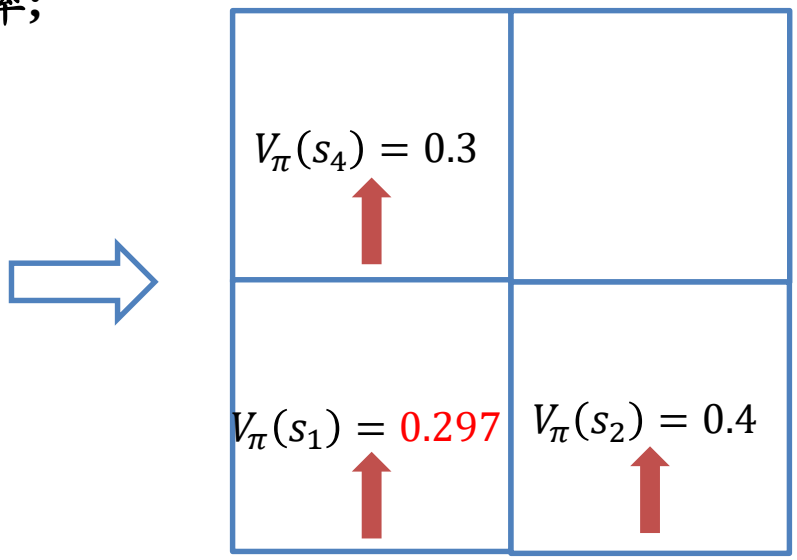
动态规划法的缺点：

1) 智能主体需要事先知道状态转移概率；

2) 无法处理状态集合大小无限的情况



优化后策略



强化学习中的策略评估：动态规划

基于蒙特卡洛采样的价值函数更新

选择不同的起始状态，按照当前策略 π 采样若干轨迹，记它们的集合为 D

枚举 $s \in S$

计算 D 中 s 每次出现时对应的反馈 $G_1, G_2, \cdots, G_k$

$$V_\pi(s) \leftarrow \frac{1}{k}\sum_{i=1}^{k} G_i$$

假设按照当前策略可样得到以下两条轨迹

$$(s_1, s_4, s_7, s_8, s_9)$$

$$(s_1, s_2, s_3, s_d)$$

s_1 对应的反馈值分别为

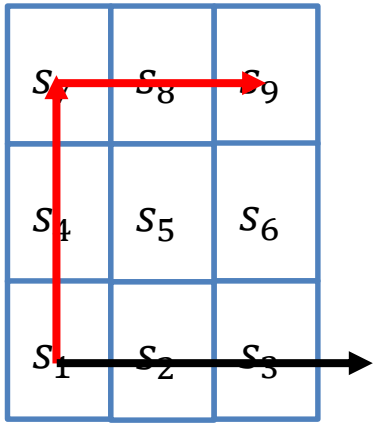
$$0 + \gamma \times 0 + \cdots + \gamma^3 \times 1 = 0.970$$

$$0 + \gamma \times 0 + \gamma^2 \times (-1) = -0.980$$

因此估计

$$V(s_1) = \frac{1}{2}(0.970 - 0.980) = -0.005$$

如果是确定的策略，每个起点只会产生一种轨迹



根据数理统计的知识，期望可以通过样本均值来估计的，这正是蒙特卡洛方法（Monte-Carlo method）的核心思想。即大数定理指出：对于独立同分布（independent and identically distributed, i.i.d)的样本数据，当样本足够大的时候，样本平均值向期望值收敛。

强化学习中的策略评估：动态规划

基于蒙特卡洛采样的价值函数更新

选择不同的起始状态，按照当前策略π采样若干轨迹，记它们的集合为D

枚举$s \in S$

计算D中s每次出现时对应的反馈$G_1, G_2, \cdots, G_k$

$V_\pi(s) \leftarrow \frac{1}{k}\sum_{i=1}^{k} G_i$

按照这样的思路可使用蒙特卡洛方法来进行策略评估：给定状态s，从该状态出发不断采样后续状态，得到不同的采样序列。通过这些采样序列来分别计算状态s的回报值，对这些回报值取均值，作为对状态s价值函数的估计，从而避免对状态转移概率的依赖。

强化学习中的策略评估：动态规划

基于蒙特卡洛采样的价值函数更新

选择不同的起始状态，按照当前策略π采样若干轨迹，记它们的集合为D

枚举$s \in S$

计算D中s每次出现时对应的反馈$G_1, G_2, \cdots, G_k$

$V_\pi(s) \leftarrow \frac{1}{k}\sum_{i=1}^{k} G_i$

蒙特卡洛采样法的优点

- 智能主体不必知道状态转移概率
- 容易扩展到无限状态集合的问题中

蒙特卡洛采样法的缺点

- 状态集合比较大时，一个状态在轨迹可能非常稀疏，不利于估计期望
- 在实际问题中，最终反馈需要在终止状态才能知晓，导致反馈周期较长

强化学习中的策略评估：动态规划

基于时序差分（Temporal Difference）的价值函数更新

初始化V_π函数
循环
　　初始化s为初始状态
　　循环
　　　　$a \sim \pi(s,\cdot)$
　　　　执行动作a，观察奖励R和下一个状态s'
　　　　更新$V_\pi(s) \leftarrow V_\pi(s) + \alpha[R(s,a,s') + \gamma V_\pi(s') - V_\pi(s)]$
　　　　$s \leftarrow s'$
　　直到s是终止状态
直到V_π收敛

时序差分法可以看作蒙特卡罗方法和动态规划方法的有机结合。时序差分算法与蒙特卡洛方法相似之处在于，时序差分方法从实际经验中获取信息，无需提前获知环境模型的全部信息。时序差分算法与动态规划方法的相似之处在于，时序差分方法能够利用前序已知信息来进行在线实时学习，无需等到整个片段结束（终止状态抵达）再进行价值函数的更新。

强化学习中的策略评估：动态规划

基于时序差分（Temporal Difference）的价值函数更新

初始化V_π函数
循环
　　初始化s为初始状态
　　循环
　　　　$a \sim \pi(s,\cdot)$
　　　　执行动作a，观察奖励R和下一个状态s'
　　　　更新$V_\pi(s) \leftarrow V_\pi(s) + \alpha[R(s,a,s') + \gamma V_\pi(s') - V_\pi(s)]$
　　　　$s \leftarrow s'$
　　直到s是终止状态
直到V_π收敛

- 更新$V_\pi(s)$的值：$V_\pi(s) \leftarrow (1-\alpha)V_\pi(s) + \alpha[R(s,a,s') + \gamma V_\pi(s')]$

过去的价值函数值　　学习得到的价值函数值

- 动态规划法根据贝尔曼方程迭代更新价值函数，要求算法事先知道状态之间的转移概率，这往往是不现实的。为了解决这个问题，时序差分法借鉴蒙特卡洛法思想，通过采样a和s'来估计计算$V_\pi(s)$
- 由于通过采样进行计算，所得结果可能不准确，因此时序差分法并没有将这个估计值照单全收，而是以α作为权重来接受新的估计值，即把价值函数更新为$(1-\alpha)V_\pi(s) + \alpha[R + \gamma V_\pi(s')]$，对这个式子稍加整理就能得到算法7.2.3中第7行形式：$V_\pi(s) \leftarrow V_\pi(s) + \alpha[R + \gamma V_\pi(s') - V_\pi(s)]$。这里$R + \gamma V_\pi(s')$为时序差分目标，$R + \gamma V_\pi(s') - V_\pi(s)$为时序差分偏差。

强化学习中的策略评估：动态规划

基于时序差分（Temporal Difference）的价值函数更新

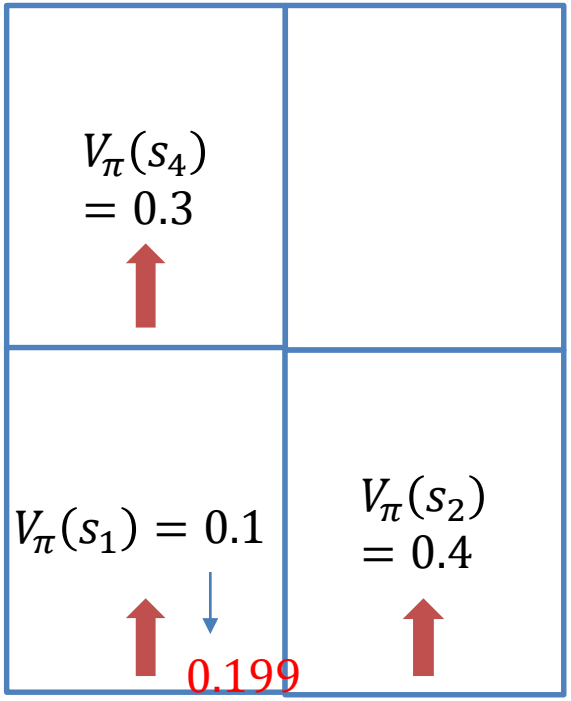
初始化V_π函数
循环
　　初始化s为初始状态
　　循环
　　$a \sim \pi(s,\cdot)$
　　执行动作a，观察奖励R和下一个状态s'
　　更新$V_\pi(s) \leftarrow V_\pi(s) + \alpha[R(s,a,s') + \gamma V_\pi(s') - V_\pi(s)]$
　　$s \leftarrow s'$
　　直到s是终止状态
直到V_π收敛

时序差分法和蒙特卡洛法都是通过采样若干个片段来进行价值函数更新的，但是时序差分法并非使用一个片段中的终止状态所提供的实际回报值来估计价值函数，而是根据下一个状态的价值函数来估计，这样就克服了采样轨迹的稀疏性可能带来样本方差较大的不足问题，同时也缩短了反馈周期。

强化学习中的策略评估：动态规划

基于时序差分（Temporal Difference）的价值函数更新

初始化V_π函数
循环
　　初始化s为初始状态
　　循环
　　$a \sim \pi(s,\cdot)$
　　执行动作a，观察奖励R和下一个状态s'
　　更新$V_\pi(s) \leftarrow V_\pi(s) + \alpha[R(s,a,s') + \gamma V_\pi(s') - V_\pi(s)]$
　　$s \leftarrow s'$
　　直到s是终止状态
直到V_π收敛

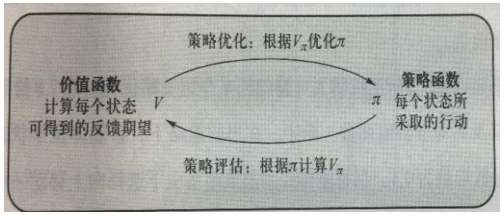


假设$\alpha = 0.5$，更新$V_\pi(s_1)$的值：
从$\pi(s_1,\cdot)$中采样得到动作$a =$ 上
从$Pr(\cdot|s_1,$上$)$中采样得到下一步状态$s' = s_4$

$$V_\pi(s_1) \leftarrow V_\pi(s_1) + \alpha[R(s_1,\text{上},s_4) + \gamma V_\pi(s_4) - V_\pi(s_1)]$$
$$= 0.1 + 0.5 \times [0 + 0.99 \times 0.3 - 0.1] = 0.199$$

在对片段进行采样的同时，不断以上述方法更新当前状态的价值函数，不断迭代直到价值函数收敛为止。

强化学习中的策略评估：Q-learning



初始化q_π函数
循环
　　初始化s为初始状态
　　循环
　　~~$a \sim \pi(s,)$~~ ➡ $a = \text{argmax}_{a'} q_\pi(s, a')$
　　执行动作a，观察奖励R和下一个状态s'
　　~~更新$V_\pi(s) \leftarrow V_\pi(s) + \alpha[R + \gamma V_\pi(s') - V_\pi(s)]$~~
　　➡ 更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s,a)\right]$
　　$s \leftarrow s'$
　　直到s是终止状态
直到q_π收敛

先前的策略优化：$\pi'(s) = \text{argmax}_a q_\pi(s,a)$

$$q_\pi(s,a) \leftarrow (1-\alpha) q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s', a')\right]$$

图7.7 基于通用策略迭代来学习最优策略的模式

Q学习中直接记录和更新动作-价值函数q_π而不是价值函数V_π，这是因为策略优化要求已知动作-价值函数q_π，如果算法仍然记录价值函数V_π，在不知道状态转移概率的情况下将无法求出q_π。于是，Q学习中，只有动作-价值函数（即q函数）参与计算。

强化学习中的策略评估：Q-learning

初始化q_π函数
循环
　　初始化s为初始状态
　　循环
　　$a = \text{argmax}_{a'} q_\pi(s, a')$
　　执行动作a，观察奖励R和下一个状态s'
　　更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s,a)\right]$
　　$s \leftarrow s'$
　　直到s是终止状态
直到q_π收敛

s_d		
s_7	s_8	s_9
s_4	s_5	s_6
s_1	s_2	s_3

初始化q_π函数

在右图中，$^a/_b$表示$q_\pi(s, 上) = a, q_\pi(s, 右) = b$

所有终止状态的q函数值设为$^0/_0$，其余状态可随机初始化，此处设$^{0.2}/_0$

初始化s，s的值在右图中用黑框框出

$^0/_0$	$^{0.2}/_0$	$^0/_0$
$^{0.2}/_0$	$^{0.2}/_0$	$^{0.2}/_0$
$^{0.2}/_0$	$^{0.2}/_0$	$^{0.2}/_0$

强化学习中的策略评估：Q-learning

初始化q_π函数
循环
　　初始化s为初始状态
　　循环
　　$a = \text{argmax}_{a'} q_\pi(s, a')$
　　执行动作a，观察奖励R和下一个状态s'
　　更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s,a)\right]$
　　$s \leftarrow s'$
　　直到s是终止状态
直到q_π收敛

s_d			
	s_7	s_8	s_9
	s_4	s_5	s_6
	s_1	s_2	s_3

$a = \text{argmax}_{a'} q_\pi(s_1, a') = \text{上}$

$R = 0,\ s' = s_4$

$q_\pi(s_1, \text{上}) \leftarrow 0.2 + 0.5 \times [0 + 0.99 \times \max\{0, 0.2\} - 0.2]$

$= 0.199$

$s \leftarrow s_4$

	$0.2/0$	$0.2/0$	$0/0$
$0/0$	$0.2/0$	$0.2/0$	$0.2/0$
$0.199/0$		$0.2/0$	$0.2/0$

强化学习中的策略评估：Q-learning

初始化q_π函数
循环
　　初始化s为初始状态
　　循环
　　$a = \text{argmax}_{a'} q_\pi(s, a')$
　　执行动作a，观察奖励R和下一个状态s'
　　更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s,a)\right]$
　　$s \leftarrow s'$
　　直到s是终止状态
直到q_π收敛

s_d			
	s_7	s_8	s_9
	s_4	s_5	s_6
	s_1	s_2	s_3

$a = \text{argmax}_{a'} q_\pi(s_4, a') =$ 上

$R = 0$，$s' = s_7$

$q_\pi(s_4, \text{上}) \leftarrow 0.2 + 0.5 \times [0 + 0.99 \times \max\{0, 0.2\} - 0.2] = 0.199$

$s \leftarrow s_7$

$0/0$	$0.2/0$	$0.2/0$	$0/0$
$0.199/0$		$0.2/0$	$0.2/0$
$0.199/0$	$0.2/0$	$0.2/0$	

强化学习中的策略评估：Q-learning

初始化q_π函数
循环
　　初始化s为初始状态
　　循环
　　$a = \text{argmax}_{a'} q_\pi(s, a')$
　　执行动作a，观察奖励R和下一个状态s'
　　更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s,a)\right]$
　　$s \leftarrow s'$
　　直到s是终止状态
直到q_π收敛

s_d	s_7	s_8	s_9
	s_4	s_5	s_6
	s_1	s_2	s_3

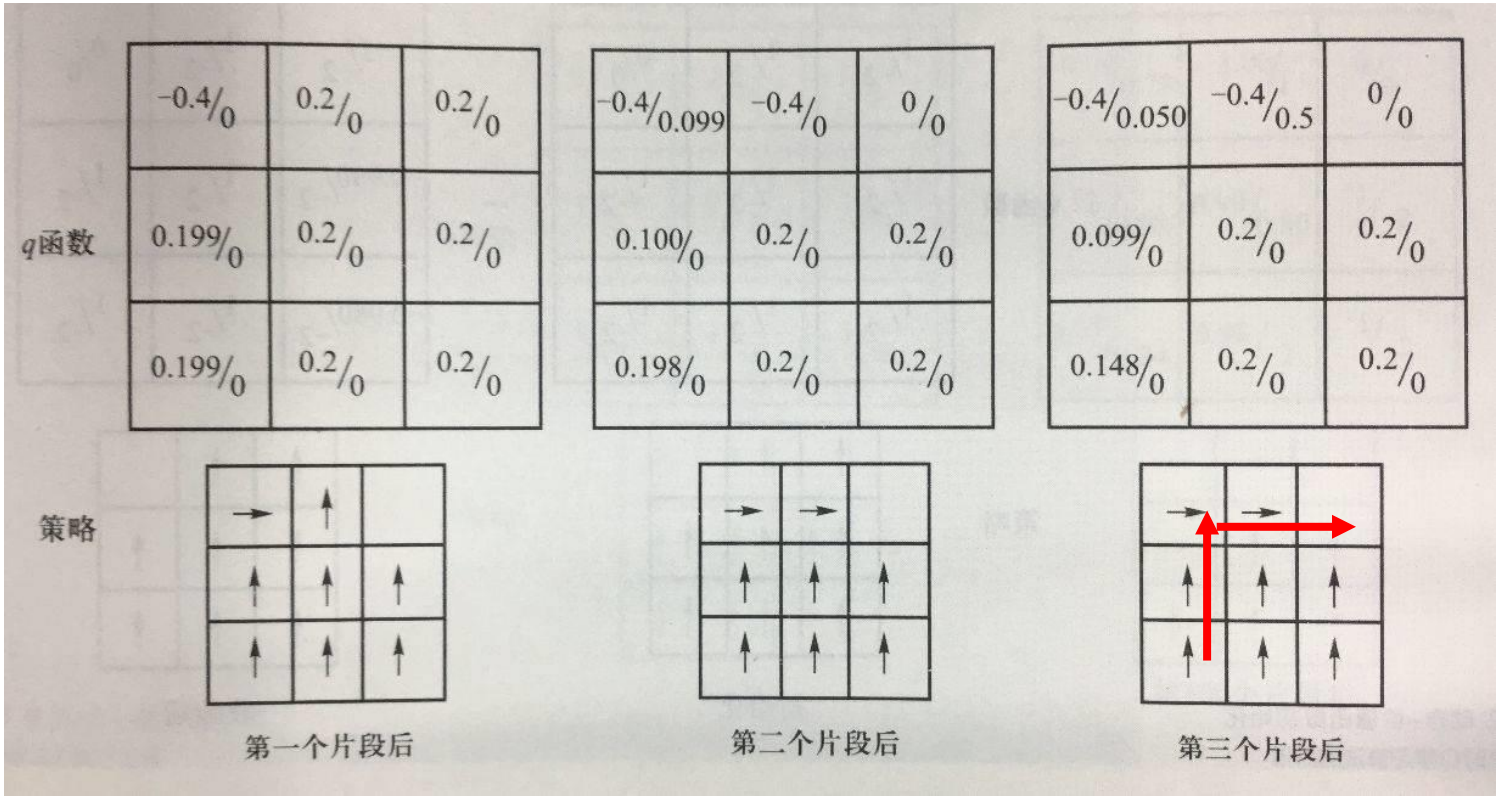
$a = \text{argmax}_{a'} q_\pi(s_7, a') = $ 上
$R = -1$，$s' = s_d$
$q_\pi(s_7, \text{上}) \leftarrow 0.2 + 0.5 \times [-1 + 0.99 \times \max\{0,0\} - 0.2] = -0.4$
$s \leftarrow s_d$

因为s_d是终止状态，因此一个片段（episode）结束

$0/0$	$-0.4/0$	$0.2/0$	$0/0$
	$0.199/0$	$0.2/0$	$0.2/0$
	$0.199/0$	$0.2/0$	$0.2/0$

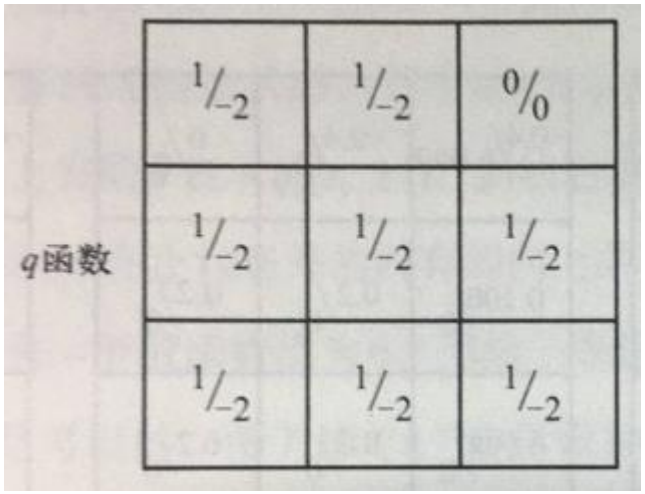
强化学习中的策略评估：Q-learning

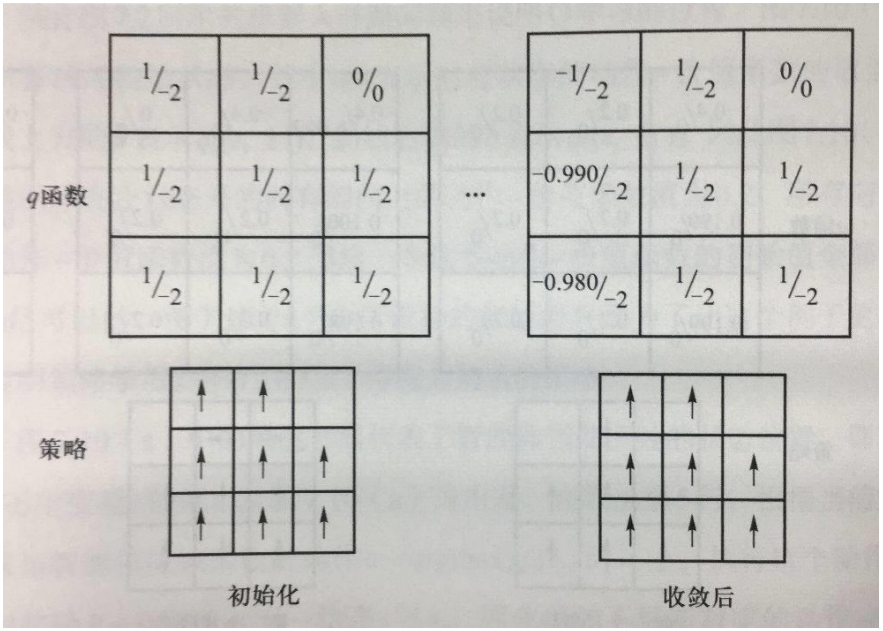
图7.11 Q学习执行三个片段的过程



强化学习中的策略评估：Q-learning

图7.12 动作-价值函数初始化为1/-2时Q学习算法的流程

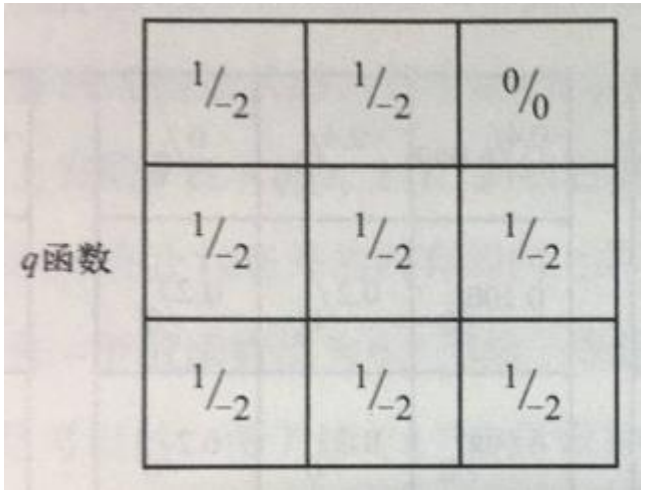


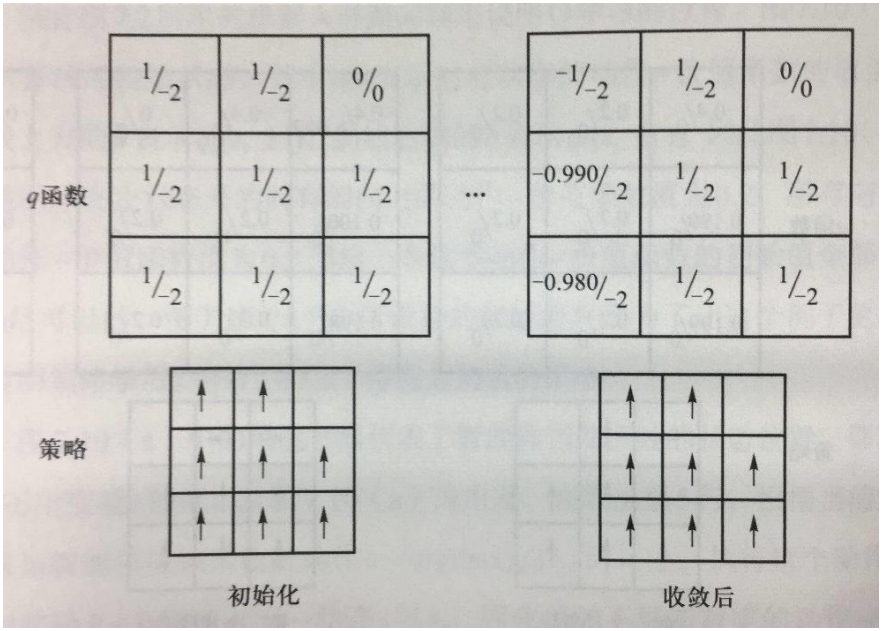


- 如果q函数的初始化为$^{1}/_{-2}$，在模型收敛后，策略仍无法使得智能主体找到目标状态
- 原因在于除了终止状态，对于每个状态，由于向上一个方格可得到1回报、而向右一个方格仅得到-2回报，因此智能体更倾向于向上移动一个方格而非向右一个方格。在这样初始策略下，即使在Q学习算法收敛以后，所得到的策略仍然不能使得机器人抵达目标状态s_9，而是沿着轨迹(s_1, s_4, s_7, s_d)导致机器人越界而被损坏。

强化学习中的策略评估：Q-learning

图7.12 动作-价值函数初始化为1/-2时Q学习算法的流程





这一问题出现的原因概括来说就是$q_\pi(\cdot, 右)$的初始值太小，导致机器人被损坏产生的-1奖励不足以推动智能体来改变策略。既然外部刺激不足以使机器人尝试新的策略，那么不妨从内部入手为智能体改变固有策略来添加一个探索的动力。

策略学习中探索（exploration）与利用（exploitation）的平衡

ϵ贪心（ϵ-greedy）策略

$$\epsilon\text{-}greedy_\pi(s) = \begin{cases} \text{argmax}_a q_\pi(s,a), & \text{以} 1-\epsilon \text{的概率} \\ \text{随机的} a \in A, & \text{以} \epsilon \text{的概率} \end{cases}$$

初始化q_π函数
循环
　初始化s
　循环

用ϵ贪心（ϵ-greedy）策略
代替$a = \text{argmax}_{a'} q_\pi(s,a')$

$$a = \text{argmax}_{a'} q_\pi(s,a')$$

执行动作a，观察奖励R和下一个状态s'

$$\text{更新} q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s',a') - q_\pi(s,a)\right]$$

$s \leftarrow s'$

　直到s是终止状态
直到q_π收敛

为此，在Q学习中引入探索（**exploration**）与利用（**exploitation**）机制。这一机制用用ϵ贪心（ϵ-**greedy**）策略来代替$a = \text{argmax}_{a'} q_\pi(s,a')$。用$\epsilon$贪心（$\epsilon$-**greedy**）策略定义如下：在状态$s$, 以$1-\epsilon$的概率来选择带来最大回报的动作，或者以$\epsilon$的概率来随机选择一个动作。

策略学习中探索（exploration）与利用（exploitation）的平衡

ϵ贪心（ϵ-greedy）策略

$$\epsilon\text{-}greedy_\pi(s) = \begin{cases} \text{argmax}_a q_\pi(s,a), & \text{以} 1-\epsilon \text{的概率} \\ \text{随机的} a \in A, & \text{以} \epsilon \text{的概率} \end{cases}$$

初始化q_π函数
循环
　初始化s
　循环

$$a = \text{argmax}_{a'} q_\pi(s,a')$$

用ϵ贪心（ϵ-greedy）策略
代替$a = \text{argmax}_{a'} q_\pi(s,a')$

执行动作a，观察奖励R和下一个状态s'

更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s',a') - q_\pi(s,a)\right]$

$s \leftarrow s'$

　直到s是终止状态
直到q_π收敛

$-1/-2$	$1/-2$	$0/0$
$-0.990/-$	$1/-2$	$1/-2$
$-0.980/-$	$1/-2$	$1/-2$

ϵ贪心策略的解释：大体上遵循最优策略的决定，偶尔（以ϵ的小概率）进行探索。如右图所示，如果能够偶尔在某些状态随机选择“向右移动一个方格”的动作，则可克服机器人无法走到终点s_9这一不足。

强化学习中的策略评估：使用ϵ贪心策略的Q学习

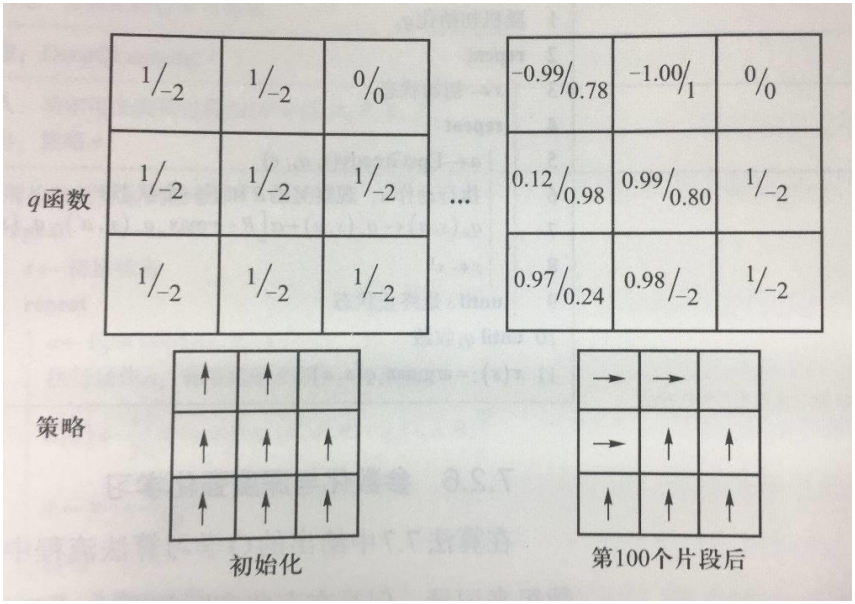
初始化q_π函数
循环
　　初始化s为初始状态
　　循环
　　采样$a \sim \epsilon - greedy_\pi(s)$
　　执行动作a，观察奖励R和下一个状态s'
　　更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s',a') - q_\pi(s,a)\right]$
　　$s \leftarrow s'$
　　直到s是终止状态
直到q_π收敛

采样策略与更新策略不同

- 将动作采样从“确定地选取最优动作”改为“按照ϵ贪心策略选取动作”
- 更新时仍保持用max操作选取最佳策略。像这样更新时的目标策略与采样策略不同的方法，叫做离策略（off-policy）方法

强化学习中的策略评估：使用ϵ贪心策略的Q学习

图7.13 使用ε贪心策略进行探索的Q学习的执行过程



令$\epsilon = 0.1$，采用探索策略的Q学习，算法在执行了100个片段后，得到了经过轨迹$(s_1, s_4, s_5, s_8, s_9)$到达目标状态的策略。由于$\epsilon$贪心策略具有概率性，因此图中的结果并不是确定的，有可能经过100个片段仍不能得到一个合适的策略，且在图中也能发现存在还没有被访问过的状态s_3。但随着迭代次数的增加，算法探索到最优策略的可能性也会显著增加，通过适当增大参数ϵ的值也能够促进算法乐于探索。

用神经网络拟合（行动）价值函数：Deep Q-learning

使用ϵ贪心策略的Q学习

初始化q_π函数
循环
　　初始化s为初始状态
　　循环
　　采样$a \sim \epsilon\text{-}greedy_\pi(s)$
　　执行动作a，观察奖励R和下一个状态s'
　　更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s',a') - q_\pi(s,a)\right]$
　　$s \leftarrow s'$
　　直到s是终止状态
直到q_π收敛

- 状态数量太多时，有些状态可能始终无法采样到，因此对这些状态的q函数进行估计是很困难的
- 状态数量无限时，不可能用一张表（数组）来记录q函数的值

思路：将q函数参数化（**parametrize**），用一个非线性回归模型来拟合q函数，例如（深度）神经网络

- 能够用有限的参数刻画无限的状态
- 由于回归函数的连续性，没有探索过的状态也可通过周围的状态来估计

用神经网络拟合（行动）价值函数：Deep Q-learning

用深度神经网络拟合q函数

初始化$\boldsymbol{q_\pi}$函数的参数$\boldsymbol{\theta}$
循环
　　初始化s为初始状态
　　循环
　　采样$\boldsymbol{a \sim \epsilon\text{-}greedy_\pi(s;\theta)}$
　　执行动作$\boldsymbol{a}$，观察奖励$\boldsymbol{R}$和下一个状态$\boldsymbol{s'}$
　　损失函数$\boldsymbol{L(\theta) = \frac{1}{2}\left[R + \gamma \max_{a'} q_\pi(s', a'; \theta) - q_\pi(s, a; \theta)\right]^2}$
　　根据梯度$\boldsymbol{\partial L(\theta)/\partial\theta}$更新参数$\boldsymbol{\theta}$
　　$\boldsymbol{s \leftarrow s'}$
　　直到$\boldsymbol{s}$是终止状态
直到$\boldsymbol{q_\pi}$收敛

- 损失函数刻画了q的估计值$R + \gamma \max_{a'} q_\pi(s', a'; \theta)$与当前值的平方误差
- 利用梯度下降法优化参数θ
- 如果用深度神经网络来拟合q函数，则算法称为深度Q学习或者深度强化学习

提纲

一、强化学习问题定义

二、基于价值的强化学习

三、基于策略的强化学习

四、深度强化学习的应用

基于策略的强化学习

基于价值的强化学习

- 以对价值函数或动作-价值函数的建模为核心。

基于策略的强化学习

- 直接参数化策略函数，求解参数化的策略函数的梯度。

策略函数的参数化可以表示为$\pi_\theta(s,a)$，其中θ为一组参数，函数取值表示在状态s下选择动作a的概率。和Q学习的ϵ贪心策略相比，这种参数化的一个显著好处是：选择一个动作的概率是随着参数的改变而光滑变化的，实际上这种光滑性对算法收敛有更好的保证。

策略梯度定理

最大化目标：$MAX\, J(\boldsymbol{\theta}) \coloneqq V_{\pi_{\boldsymbol{\theta}}}(s_0)$

策略梯度定理（**Policy Gradient Theorem**）：

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \sum_s \mu_{\pi_{\boldsymbol{\theta}}}(s) \sum_a q_{\pi_{\boldsymbol{\theta}}}(s,a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(s,a) \quad (7.11)$$

- 其中$\mu_{\pi_{\boldsymbol{\theta}}}(s)$称为策略$\pi_{\boldsymbol{\theta}}$对于状态$s$的策略分布，假设$\gamma = 1$（即前面定义的折扣系数）。在持续问题中，$\mu_{\pi_{\boldsymbol{\theta}}}(s)$为算法从$s_0$出发经过无限多步后位于状态$s$的概率；在分段问题中，$\mu_{\pi_{\boldsymbol{\theta}}}(s)$为归一化后的算法从$s_0$出发访问$s$的次数的期望。当$\gamma \in (0,1)$时，则需要给每个状态的$\mu_{\pi_{\boldsymbol{\theta}}}(s)$值加上一个权重。在保证算法通用性的前提下，为了简化说明，下文在进行公式推导时始终假设$\gamma = 1$，但是在算法流程中会体现折扣系数γ的影响。
- **策略梯度法的求解思路：如果能够计算或估计出公式(7.11)中的梯度，智能体就能直接对策略函数进行优化。**

基于蒙特卡洛采样的策略梯度法

根据策略梯度公式计算策略梯度并非一个简单的问题，其中对μ_{π_θ}和q_{π_θ}的准确估计本来就是难题，更不用说进一步计算$\nabla_\theta J(\theta)$了。好在蒙特卡洛法能被用来估计这类问题的取值，为此首先要对策略梯度公式进行如下的适当变形：

$$\nabla_\theta J(\boldsymbol{\theta}) \propto \sum_s \mu_{\pi_\theta}(s) \sum_a q_{\pi_\theta}(s,a) \nabla_\theta \pi_\theta(s,a)$$

$$= \sum_s \mu_{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \left[q_{\pi_\theta}(s,a) \frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} \right]$$

$$= \mathbb{E}_{s,a\sim\pi} \left[q_{\pi_\theta}(s,a) \frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} \right]$$

$$= \mathbb{E}_{s,a\sim\pi} \left[q_{\pi_\theta}(s,a) \nabla_\theta \ln \pi_\theta(s,a) \right] \quad （这里利用了 \nabla_\theta \ln Z = \frac{1}{Z} \nabla_\theta Z）$$

由于公式中存在q_{π_θ}，算法无法直接使用蒙特卡洛法来求取其中的期望。为此，进一步将公式变形

$$\nabla_\theta J(\boldsymbol{\theta}) \propto \mathbb{E}_{s,a\sim\pi} \left[\mathbb{E}_{\mathcal{T}(s,a)\sim\pi} [G_t | s,a] \nabla_\theta \ln \pi_\theta(s,a) \right]$$

$$= \mathbb{E}_{s,a,\mathcal{T}(s,a)\sim\pi} [G_t \nabla_\theta \ln \pi_\theta(s,a)]$$

其中$\mathcal{T}(s,a)$表示从状态s开始执行动作a得到的一条轨迹（不包括s和a），G_t为从状态s开始沿着轨迹$\mathcal{T}(s,a)$运动所得回报。可以使用蒙特卡洛采样法来求解公式(7.13)，即算法只需根据策略来采样一个状态s、一个动作a和将来的轨迹，就能构造公式(7.13)中求取期望所对应的一个样本。

基于蒙特卡洛采样的策略梯度法

算法 7.9 REINFORCE 算法

函数：REINFORCE

输入： 马尔可夫决策过程$MDP = (S, A, P, R, \gamma)$

输出： 策略π

随机初始化 $\boldsymbol{\theta}$
repeat
  根据策略 $\pi_{\boldsymbol{\theta}}$ 采样一个片段 $s_0, a_0, R_1, s_1, \cdots, s_{T-1}, a_{T-1}, R_T$
  for $t \leftarrow 0$ **to** $T-1$ **do**
    $G \leftarrow \sum_{k=1}^{T-t} \gamma^{k-1} R_{t+k}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \gamma^t G \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s_t, a_t)$
  end
until $\boldsymbol{\theta}$ 收敛

问题：假设某个状态s_0下有两种可选动作a_0和a_1，满足$q_{\pi_\theta}(s_0, a_0) > q_{\pi_\theta}(s_0, a_1) > 0$，那么理想状态下学习得到的策略应满足$\pi_\theta(s_0, a_0) > \pi_\theta(s_0, a_1)$。但如果样本片段中只出现了$a_1$而没有出现$a_0$，由于$q_{\pi_\theta}(s_0, a_1) > 0$，所以改变$\theta$的值使得$\pi_\theta(s_0, a_1)$增大（相应地$\pi_\theta(s_0, a_0)$减小）有助于提高样本的加权对数似然概率，但这显然与期望的结果不相符。在实际中，这个问题确实会影响REINFORCE方法的稳定性和收敛速度。

基于蒙特卡洛采样的策略梯度法

首先观测策略梯度公式，可以发现如下性质：

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \mathbb{E}_{s,a\sim\pi}\left[q_{\pi_{\boldsymbol{\theta}}}(s,a)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s,a)\right]$$

$$= \mathbb{E}_{s,a\sim\pi}\left[\left(q_{\pi_{\boldsymbol{\theta}}}(s,a) - b(s) + b(s)\right)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s,a)\right]$$

$$= \mathbb{E}_{s,a\sim\pi}\left[\left(q_{\pi_{\boldsymbol{\theta}}}(s,a) - b(s)\right)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s,a)\right] + \mathbb{E}_{s,a\sim\pi}[b(s)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s,a)]$$

其中

$$\mathbb{E}_{s,a\sim\pi}[b(s)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s,a)] = \mathbb{E}_{s\sim\pi}\left[b(s)\sum_a \pi_{\boldsymbol{\theta}}(s,a)\frac{\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(s,a)}{\pi_{\boldsymbol{\theta}}(s,a)}\right]$$

$$= \mathbb{E}_{s\sim\pi}\left[b(s)\nabla_{\boldsymbol{\theta}}\sum_a \pi_{\boldsymbol{\theta}}(s,a)\right]$$

$$= \mathbb{E}_{s\sim\pi}[b(s)\nabla_{\boldsymbol{\theta}} 1]$$

$$= 0$$

因此

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \mathbb{E}_{s,a\sim\pi}\left[\left(q_{\pi_{\boldsymbol{\theta}}}(s,a) - b(s)\right)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s,a)\right]$$

其中$b(s)$是一个关于状态的函数。其含义是，在动作-价值函数上任意减去一个关于状态的函数不会影响策略梯度。这回答了第一个问题，即通过设计一个任意函数$b(s)$就可以达到调整动作-价值函数取值而不影响优化目标的目的。

基于蒙特卡洛采样的策略梯度法

算法 7.10 使用基准函数的 REINFORCE 方法

函数：BaselineREINFORCE

输入： 马尔可夫决策过程$MDP = (S, A, P, R, \gamma)$

输出： 策略π

随机初始化 $\boldsymbol{\theta}, \boldsymbol{w}$
repeat
  根据策略 π_θ 采样一个片段 $s_0, a_0, R_1, s_1, \cdots, s_{T-1}, a_{T-1}, R_T$
  for $t \leftarrow 0$ **to** $T-1$ **do**
    $G \leftarrow \sum_{k=1}^{T-t} \gamma^{k-1} R_{t+k}$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta_{\boldsymbol{w}} \gamma^t \frac{1}{2} \nabla_{\boldsymbol{w}} [G - V_{\boldsymbol{w}}(s_t)]^2$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \gamma^t [G - V_{\boldsymbol{w}}(s_t)] \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s_t, a_t)$
  end
until θ 收敛

基于时序差分的策略梯度法：Actor-Critic

基于时序差分的策略梯度法的流程，其名称为Actor-Critic算法

算法 7.11 Actor-Critic 算法

函数：ActorCritic

输入： 马尔可夫决策过程$MDP=(S,A,P,R,\gamma)$

输出： 策略π

随机初始化 $\boldsymbol{\theta},\boldsymbol{w}$
repeat
  $s\leftarrow$ 初始状态
  $t\leftarrow 0$
  repeat
    $a\sim\pi_{\boldsymbol{\theta}}(s,\cdot)$
    执行动作 a，观察奖励 R 和下一时刻状态 s'
    $\boldsymbol{w}\leftarrow\boldsymbol{w}+\eta_{\boldsymbol{w}}\gamma^t[R+\gamma V_{\boldsymbol{w}}(s')-V_{\boldsymbol{w}}(s)]\nabla_{\boldsymbol{w}}V_{\boldsymbol{w}}(s)$
    $\boldsymbol{\theta}\leftarrow\boldsymbol{\theta}+\eta_{\boldsymbol{\theta}}\gamma^t[R+\gamma V_{\boldsymbol{w}}(s')]\nabla_{\boldsymbol{\theta}}\ln\pi_{\boldsymbol{\theta}}(s,a)$
    $t\leftarrow t+1$
  until s 是终止状态
until θ 收敛

Actor-Critic

算法 7.12 优势 Actor-Critic 算法

函数：A2C

输入： 马尔可夫决策过程$MDP = (S, A, P, R, \gamma)$

输出： 策略π

随机初始化 $\boldsymbol{\theta}, \boldsymbol{w}$
repeat
  $s \leftarrow$ 初始状态
  $t \leftarrow 0$
  repeat
    $a \sim \pi_{\boldsymbol{\theta}}(s, \cdot)$
    执行动作 a，观察奖励 R 和下一时刻状态 s'
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta_{\boldsymbol{w}} \gamma^t [R + \gamma V_{\boldsymbol{w}}(s') - V_{\boldsymbol{w}}(s)] \nabla_{\boldsymbol{w}} V_{\boldsymbol{w}}(s)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \gamma^t [R + \gamma V_{\boldsymbol{w}}(s') - V_{\boldsymbol{w}}(s)] \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s, a)$
    $t \leftarrow t + 1$
  until s 是终止状态
until θ 收敛

提纲

一、强化学习问题定义

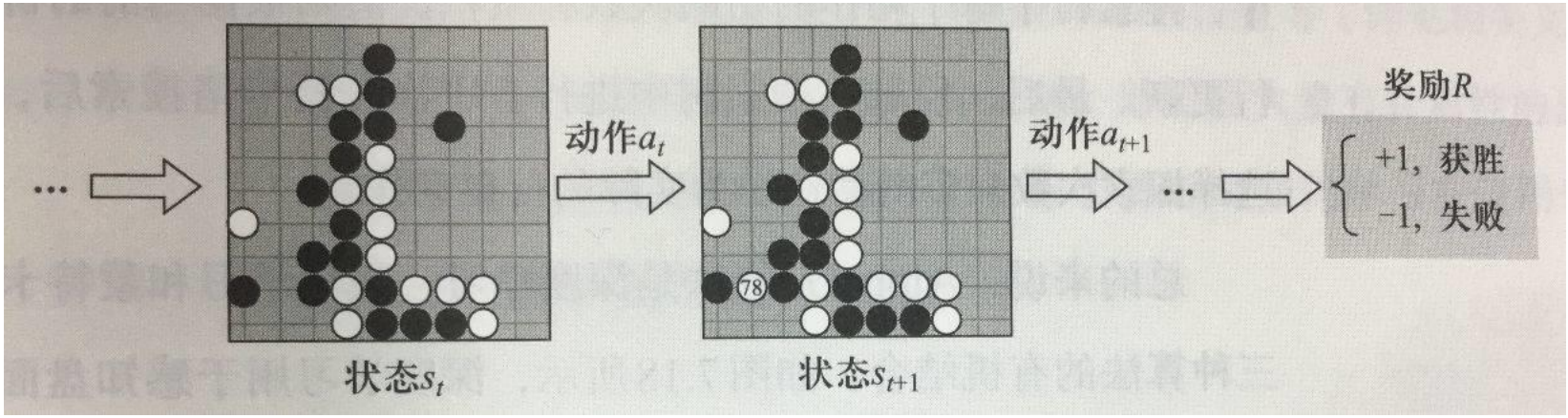
二、基于价值的强化学习

三、基于策略的强化学习

四、深度强化学习的应用

深度Q学习的应用实例: 围棋博弈

图7.17 围棋游戏一个片段的轨迹



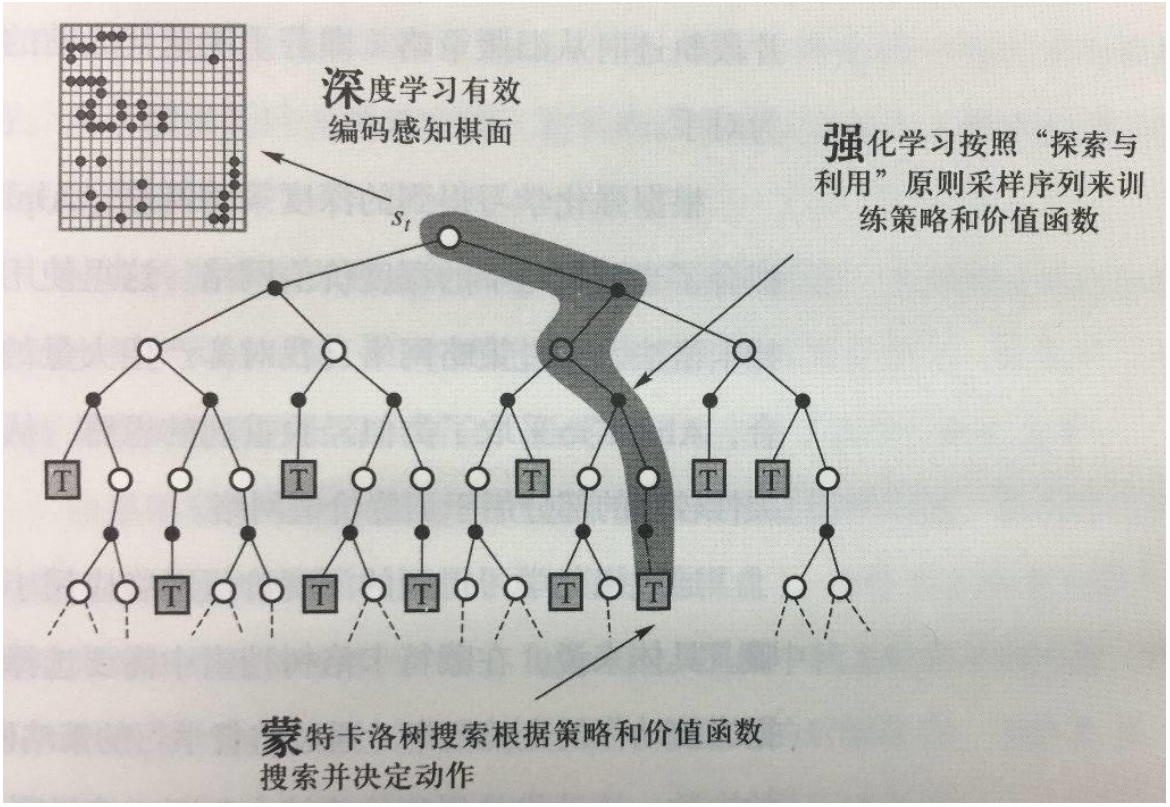
深度Q学习的应用实例: 围棋博弈

深度学习
有效编码黑白相间围棋的棋面

强化学习
助力做出长时间序贯决策

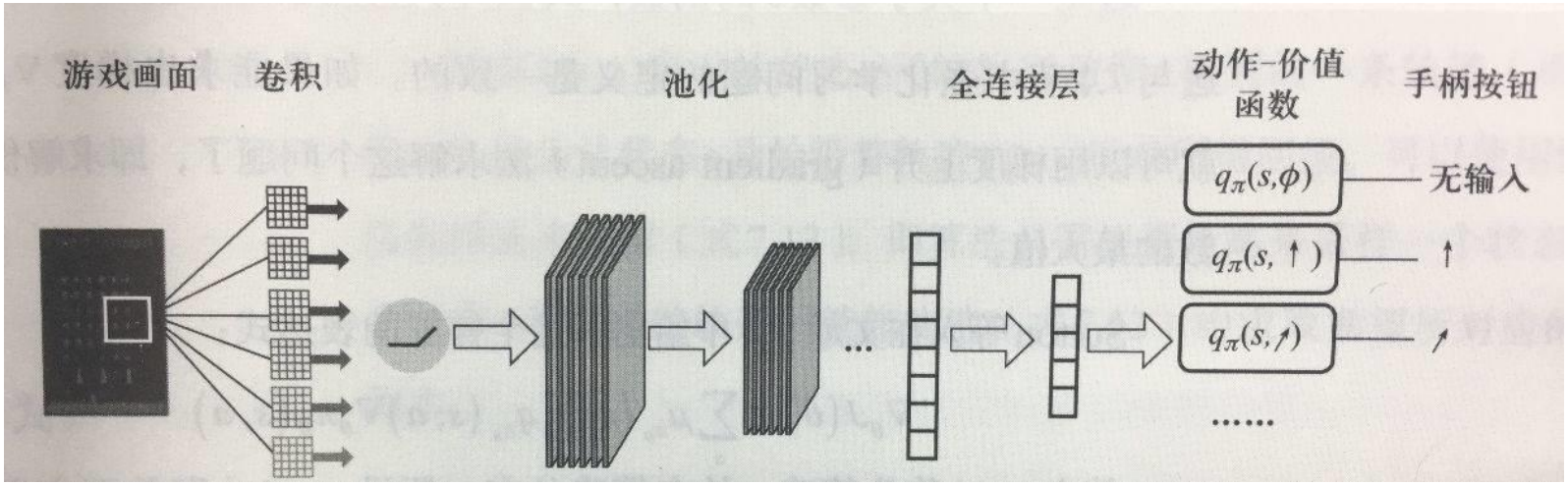
蒙特卡洛树搜索
支持从浩渺样本空间中采样

图7.18 AlphaGo算法的三个重要组成部分



深度Q学习的应用实例: 雅达利游戏

图7.16 用于游戏的DQN动作-价值函数模型

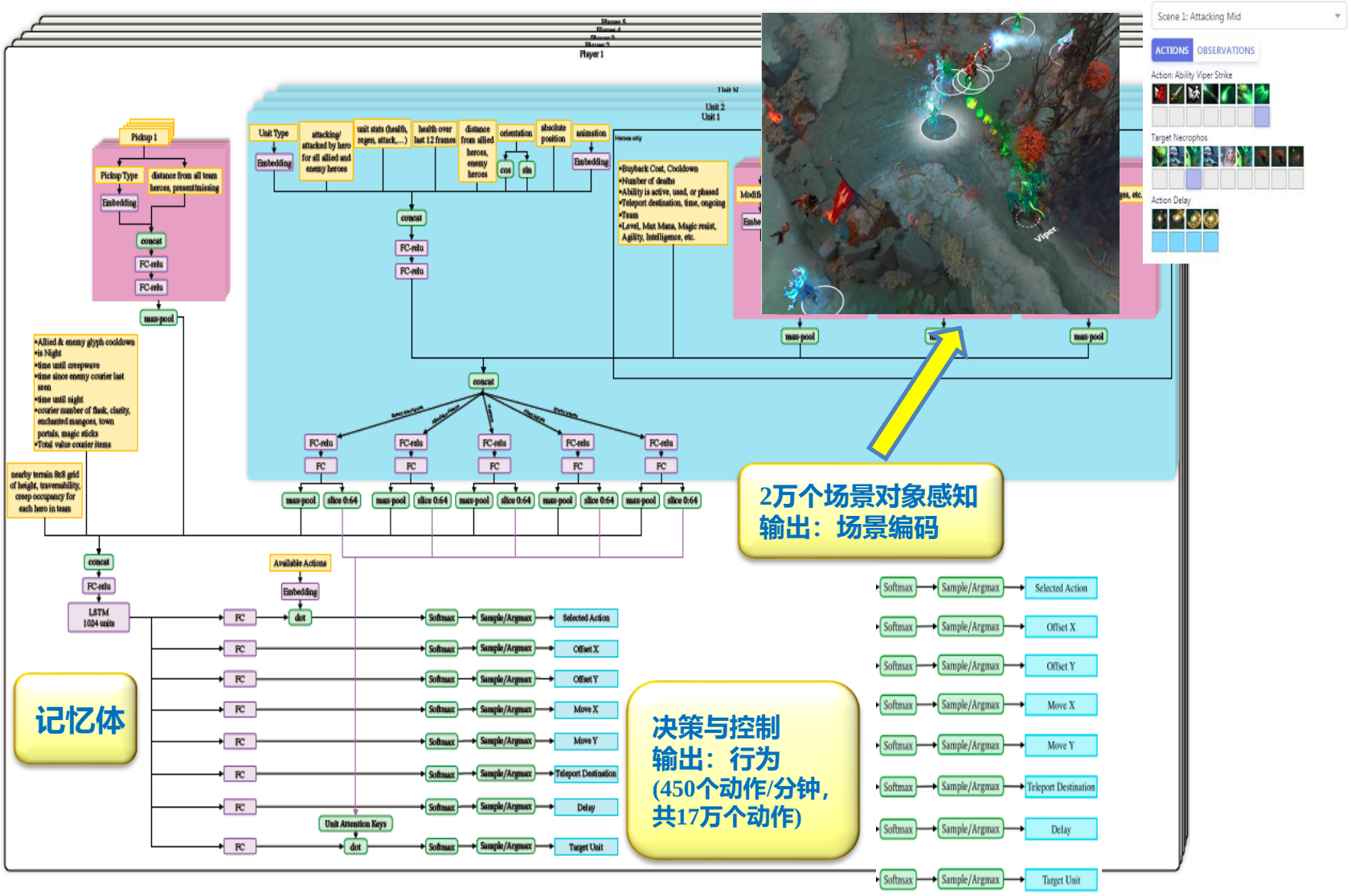


q函数的学习模型

Mnih, Volodymyr, et al, Human-level control through deep reinforcement learning, Nature 518.7540 (2015)

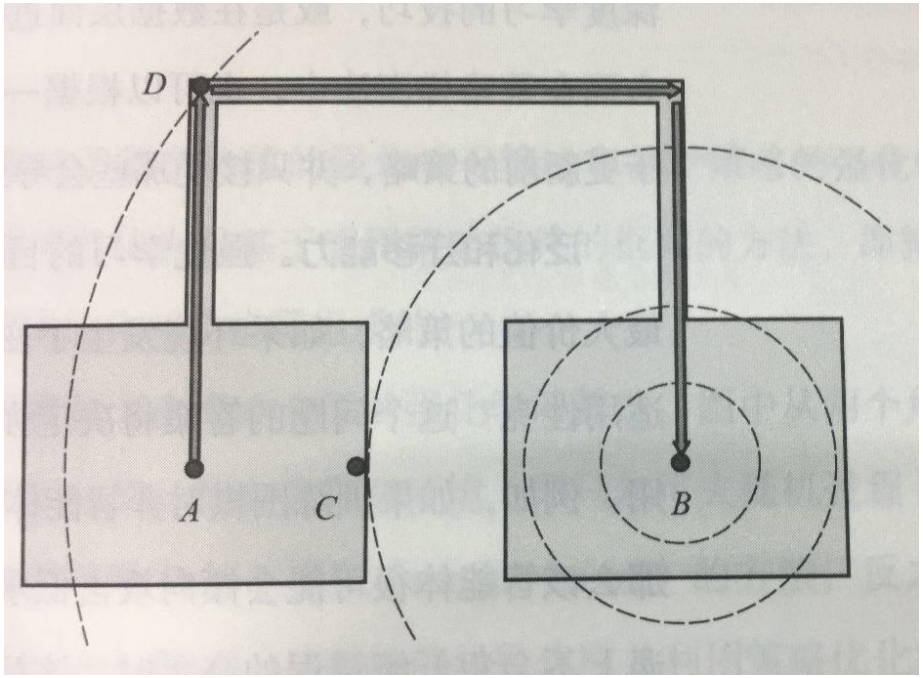
博弈对抗的算法例子

DOTA2（状态空间极其庞大下完全信息下博弈）



深度Q学习的应用实例: 难以探索的例子

图7.19 一个难以探索的例子

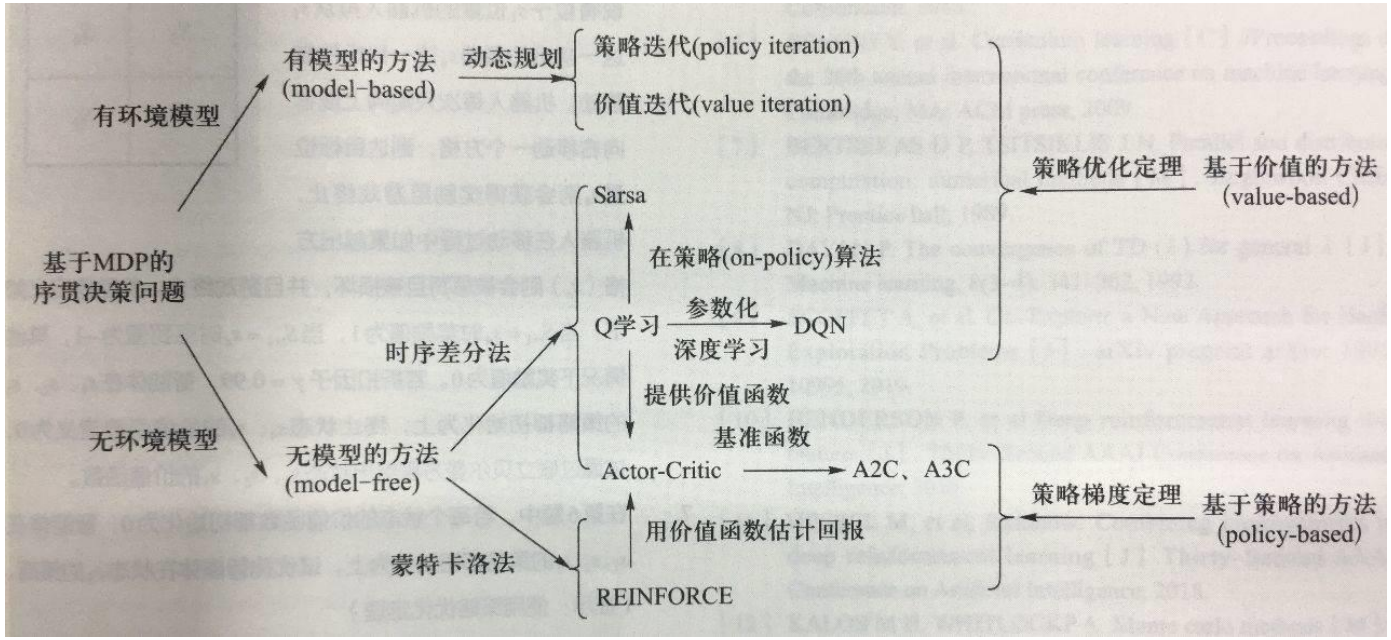


假设智能体位于A点，想要学习一个到达B点的策略，一种直观的思路是采用当前位置到目标的直线距离的倒数来初始化价值函数（图中虚线表示到B点的等距线）。如果此时只利用而不探索，则智能体会陷入移动到C点的局部最优解。如果此时采用ϵ-贪心的探索策略，为了探索找到箭头所示的最优解，智能体必须在D点之前的每一次行动都违反当前价值函数的指导，而这种概率极小。

强化学习的分类

欲乘者务时　欲治者因势

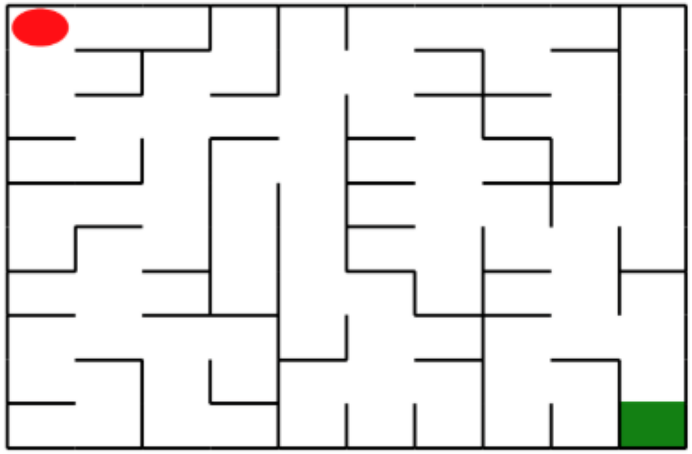
图7.20 本章所介绍方法之间相互关系示意图



图中从两个角度对强化学习算法做了分类，其中依靠对环境（即马尔可夫随机过程）的先验知识或建模的算法称为基于模型（Model-based）的方法，反之称为无模型的方法（Model-free）；只对价值函数建模并利用策略优化定理求解的方法称为基于价值（Value-based）的方法，对策略函数建模并利用策略梯度定理求解的方法称为基于策略（Policy-based）的方法。

作业：深度Q函数学习

在本实验中，要求分别使用基础搜索算法和 Deep QLearning 算法，完成机器人自动走迷宫。



如上图所示，左上角的红色椭圆既是起点也是机器人的初始位置，右下角的绿色方块是出口。

游戏规则为:从起点开始，通过错综复杂的迷宫，到达目标点(出口)。在任一位置可执行动作包括：向上走 'u'、向右走 'r'、向下走 'd'、向左走 'l'。

- 执行不同的动作后，根据不同的情况会获得不同的奖励，具体而言，有以下几种情况。
- 撞墙；走到出口；其余情况
- 你需要实现基于基础搜索算法和 Deep QLearning 算法的机器人，使机器人自动走到迷宫的出口。
- 使用 Python 语言；使用基础搜索算法完成机器人走迷宫；使用 Deep QLearning 算法完成机器人走迷宫。
- 算法部分需要自己实现，不能使用现成的包、工具或者接口。

可以使用 Python 实现基础算法的实现， 使用 Keras、PyTorch等框架实现 Deep QLearning 算法。

布置日期：2022年5月23日；提交作业：2022年6月10日晚上12点前