涉及知识点：
马尔可夫决策过程、基于动态规划的强化学习、
基于模型的强化学习

强化学习基础1

田政

https://www.deepmind.com/learning-resources/reinforcement-learning-lecture-series-2021

课程助教

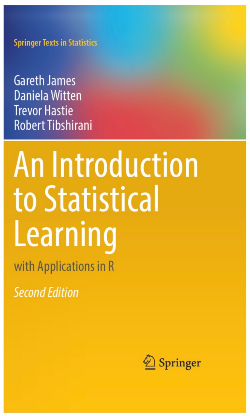
王彦聪
wangyc2023@shanghaitech.edu.cn

群聊：强化学习应用实践

该二维码7天内(2月24日前)有效，重新进入将更新

课程参考材料

- 本课程没有官方教材，以下内容作为参考资料
- 统计学习参考书



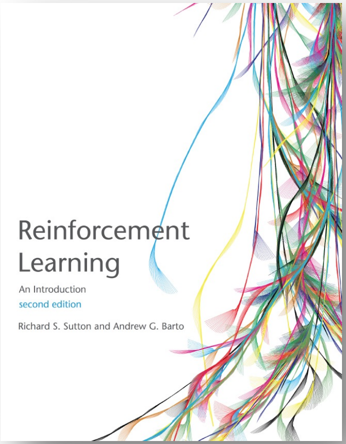
[Gareth James](#) [Daniela Witten](#) [Trevor Hastie](#) [Rob Tibshirani](#)

https://www.statlearning.com/

李航

课程参考材料

- 本课程没有官方教材，以下内容作为参考资料
- 强化学习参考书



Rich Sutton

Andew Barto

http://incompleteideas.net/book/RLbook2020.pdf

- 参考课程
  - UCL David Silver RL Course: https://www.davidsilver.uk/teaching/
  - Berkeley Sergey Levine Deep RL Course: http://rail.eecs.berkeley.edu/deeprlcourse/
  - OpenAI DRL Camp: https://sites.google.com/view/deep-rl-bootcamp/lectures
  - RL China Camp: http://rlchina.org/

课程大纲

1	课程安排与组织，强化学习基础1
2	动态规划实践1
3	深度强化学习基础
4	深度强化学习实践1
5	深度强化学习实践2
6	多智能体强化学习基础1
7	多智能体强化学习实践1
8	多智能体强化学习实践2
9	快思考与慢思考双系统基础1
10	快思考与慢思考双系统实践1
11	快思考与慢思考双系统实践2
12	快思考与慢思考双系统实践3

课程评分机制

- 实践1：20%
- 实践2：20%
- 实践3：20%
- 实践4：35%
- 出勤：5%

学术诚信

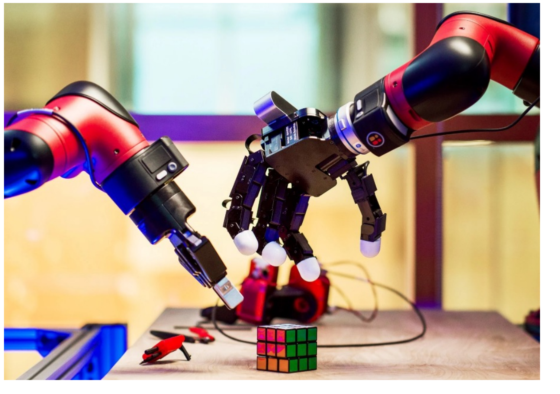
本课程高度重视学术诚信，严禁抄袭、作弊等行为。

“在学习、科研、实习实践等活动中，学生应恪守学术道德，坚守学术诚信，保护知识产权，坚持勇于创新、求真务实的科学精神，努力培养自己严谨求实、诚实自律、真诚协作的科学态度，成为良好学术风气的维护者、严谨治学的力行者、优良学术道德的传承者。”

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

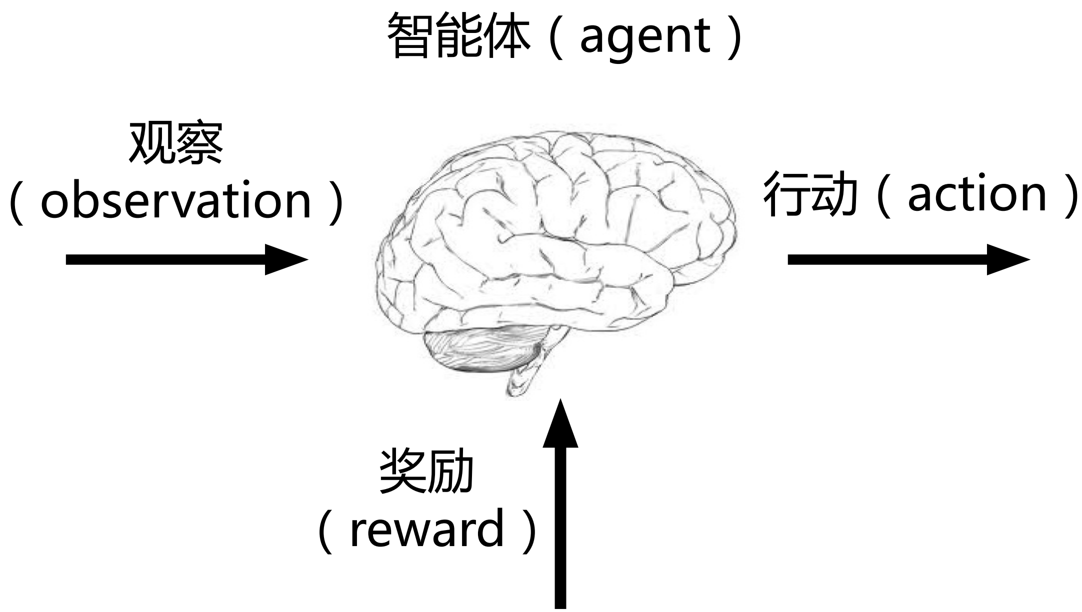


序贯决策（Sequential Decision Making）

只要是序贯决策问题，就可以用强化学习来解

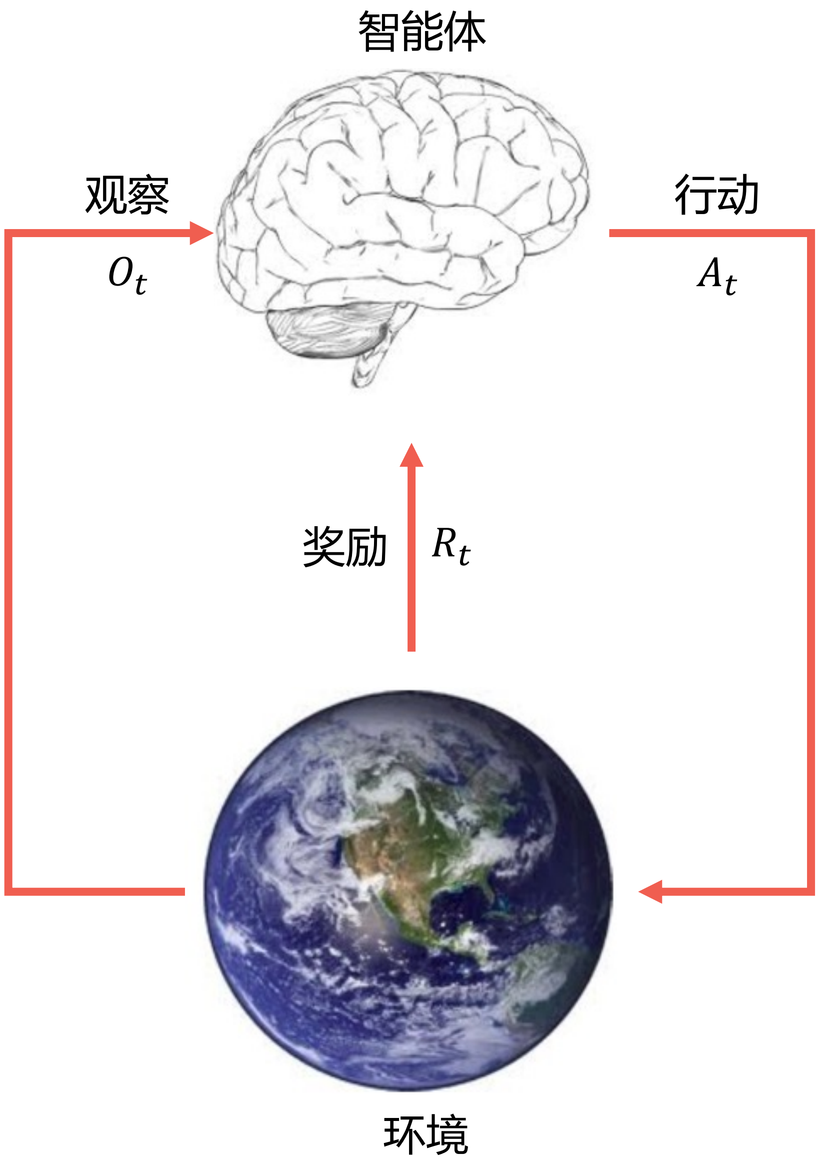
强化学习定义

- 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

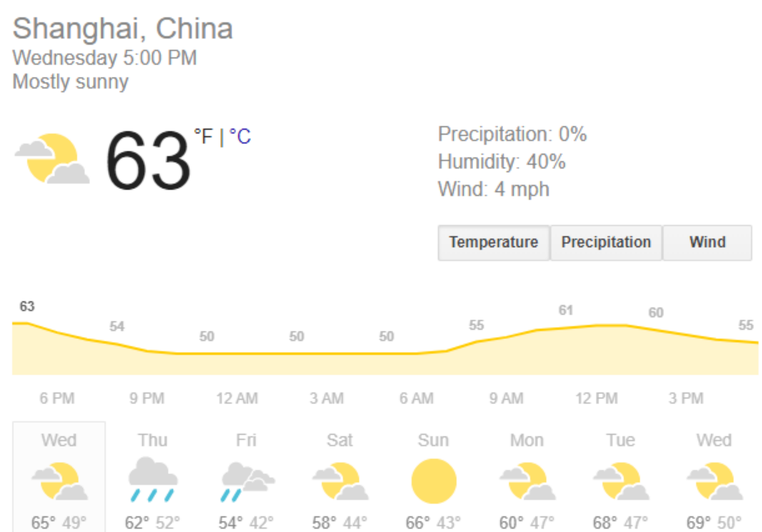
随机过程

- 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



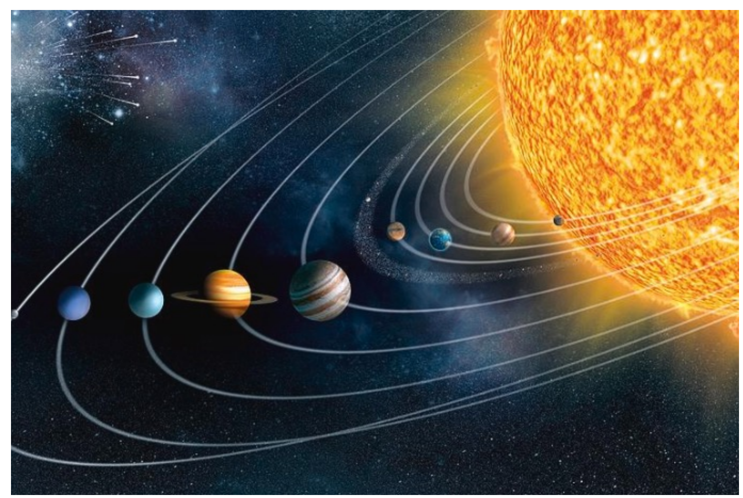
天气变化

随机过程

足球比赛

城市交通

生态系统



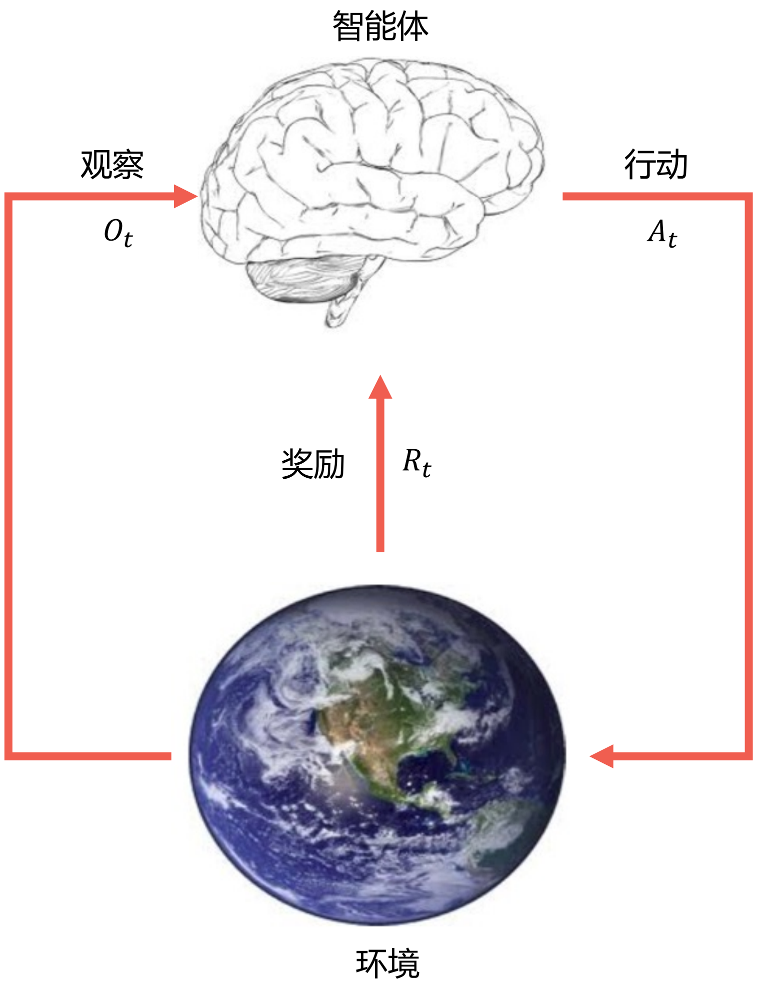
星系

强化学习的基础

1、强化学习的最重要的基础概念：马尔科夫决策过程。

2、一种数学的语言来描述智能体与环境之间的互动关系。

3、这可以方便我们能更清晰的、严谨地讨论在这个互动过程中，我们的目标是什么，以及怎样实现这个目标。



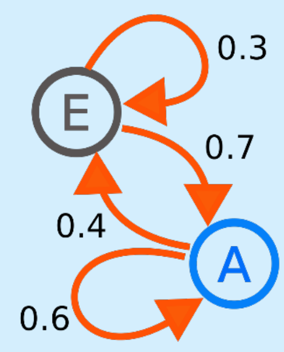
马尔可夫过程

- 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

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



一个具有两个转换状态的马尔可夫链

- 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
  - $R: S \times A \longmapsto \mathbb{R}$ 是奖励函数
    - 有时奖励只和状态相关

MDP的平稳性

一个MDP是平稳的，表示五元组中的任意对象都不会随时间变化而变化

- 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
  - $R: S \times A \longmapsto \mathbb{R}$ 是奖励函数
    - 有时奖励只和状态相关

此处请输出出处和引用

MDP的马尔可夫性

- 马尔可夫决策过程中的所有状态均满足马尔可夫性。

$$p\left(S_{t+1}=s' \mid S_t=s, A_t=a\right)=p\left(S_{t+1}=s' \mid S_1, \ldots, S_{t-1}, A_1, \ldots, A_t, S_t=s\right) \quad (1)$$

$$p\left(S_{t+1}=s' \mid S_t=s, A_t=a\right)=p\left(S_{t+1}=s' \mid S_1, \ldots, S_{t-1}, S_t=s, A_t=a\right) \quad (2)$$

$$p\left(S_{t+1}=s' \mid S_t=s, A_t=a\right)=p\left(S_{t+1}=s' \mid S_1, \ldots, S_{t-1}, S_t=s\right) \quad (3)$$

$$p\left(R_{t+1}=r, S_{t+1}=s' \mid S_t=s\right)=p\left(R_{t+1}=r, S_{t+1}=s' \mid S_1, \ldots, S_{t-1}, S_t=s\right) \quad (4)$$

https://www.deepmind.com/learning-resources/reinforcement-learning-lecture-series-2021

强化学习的目标

- R_t ：及时的，在一个状态s_t下，采取动作a_t，获得的奖励
- G_t：长期的，从一个状态s_t开始，直到当前决策过程结束，获得累积的奖励指标

Undiscounted return (episodic/finite horizon.)

$$G_t = R_{t+1} + R_{t+2} + \ldots + R_T = \sum_{k=0}^{T-t-1} R_{t+k+1}$$

Discounted return (finite or infinite horizon.)

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-t} R_T = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

Average return (continuing, infinite horizon pb.)

$$G_t = \frac{1}{T-t-1}(R_{t+1} + R_{t+2} + \ldots + R_T) = \frac{1}{T-t-1} \sum_{k=0}^{T-t-1} R_{t+k+1}$$

https://www.deepmind.com/learning-resources/reinforcement-learning-lecture-series-2021

折扣的回报

无限期的折扣汇报G_t

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

折扣因子$\gamma \in [0,1]$是将未来的奖励折算成现值的折扣系数：

- 如果当前的步数为1，那在k+1步获得的奖励R折算成当前的现值为γ^kR；
- $\gamma < 1$表示当前的奖励要比未来的奖励更有价值；
- 因此，γ接近于0的时候，表示我们是短视的，只关心当前的奖励；
- γ接近于1的时候，表示我们是远视的，将未来的奖励与当前的奖励视为同等重要

为什么要折扣？

常识：

- 通常来说，当前的奖励比未来的奖励更有价值（彩票兑换）
- 人和动物一般都偏向于当前的奖励

解决MDP：

- 无限期问题如果没有折扣因子无法定义优化目标
- 一些算法的收敛性依赖于折扣因子的存在

误区：奖励决定了我们的目标。

奖励和折扣因子共同决定了我们的目标。

策略Policies

Goal of an RL agent

To find a behaviour policy that maximises the (expected) return G_t

- A policy is a mapping $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$ that, for every state s assigns for each action $a \in \mathcal{A}$ the probability of taking that action in state s. Denoted by $\pi(a|s)$.
- For deterministic policies, we sometimes use the notation $a_t = \pi(s_t)$ to denote the action taken by the policy.

MDP的动态

- 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

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

  - 直到终止状态s_T出现为止，或者无止尽地进行下去

MDP的随机性

通常MDP的随机性来自于两层因素：

- 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

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

  - 直到终止状态s_T出现为止，或者无止尽地进行下去

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

$$\rho^{\pi_1} = \rho^{\pi_2} \ \text{ iff } \ \pi_1 = \pi_2$$

- 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

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

U. Syed, M. Bowling, and R. E. Schapire. Apprenticeship learning using linear programming. ICML 2008.

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

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

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

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

$$= \sum_{a'} \rho^{\pi}(s,a')$$

U. Syed, M. Bowling, and R. E. Schapire. Apprenticeship learning using linear programming. ICML 2008.

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

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

- 累积奖励为

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

Bellman等式

状态值函数

- The value function $v(s)$ gives the long-term value of state s

$$v_\pi(s) = \mathbb{E}\left[G_t \mid S_t = s, \pi\right]$$

- It can be defined recursively:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, \pi\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t \sim \pi(S_t)\right] \\ &= \sum_a \pi(a \mid s) \sum_r \sum_{s'} p(r, s' \mid s, a)\left(r + \gamma v_\pi(s')\right) \end{aligned}$$

- The final step writes out the expectation explicitly

动作值函数

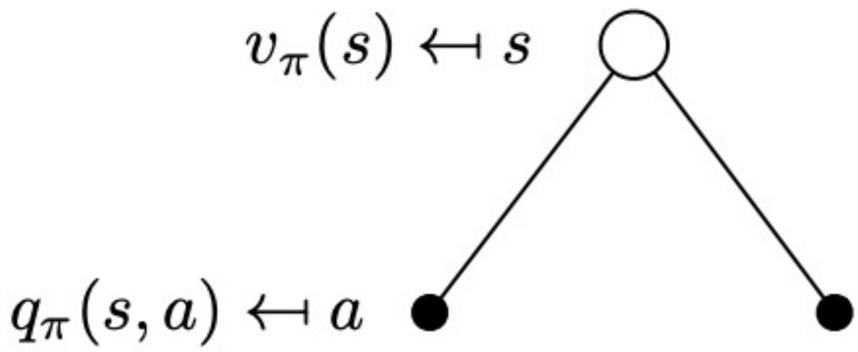
- We can define state-action values

$$q_\pi(s,a) = \mathbb{E}\left[G_t \mid S_t = s, A_t = a, \pi\right]$$

- This implies

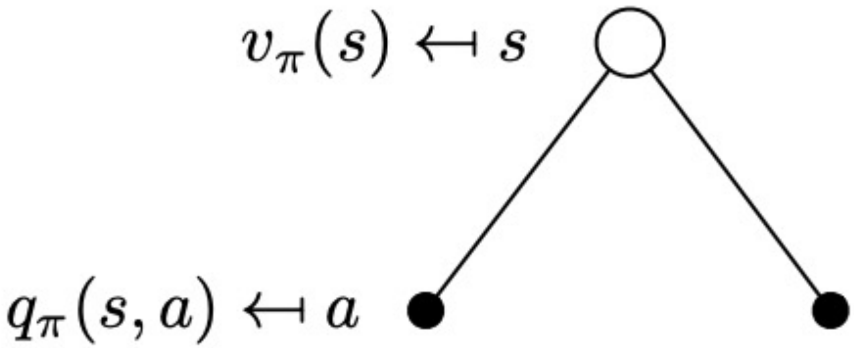
$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right] \\ &= \sum_r \sum_{s'} p(r, s' \mid s, a) \left(r + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s', a') \right) \end{aligned}$$

状态值函数(1)



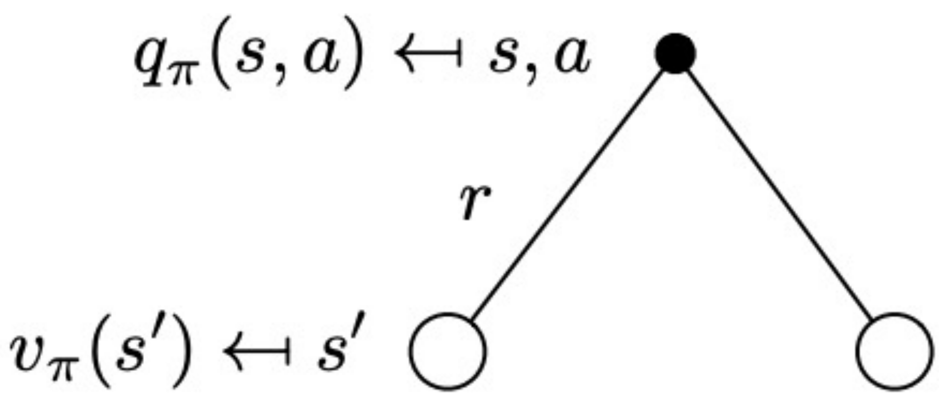
$$v_\pi(s) = \boxed{\quad ? \quad}$$

状态值函数(1)



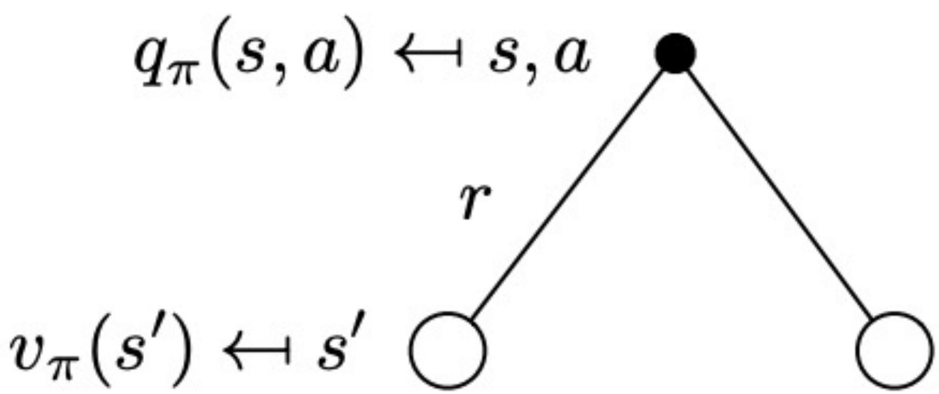
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

动作值函数(1)



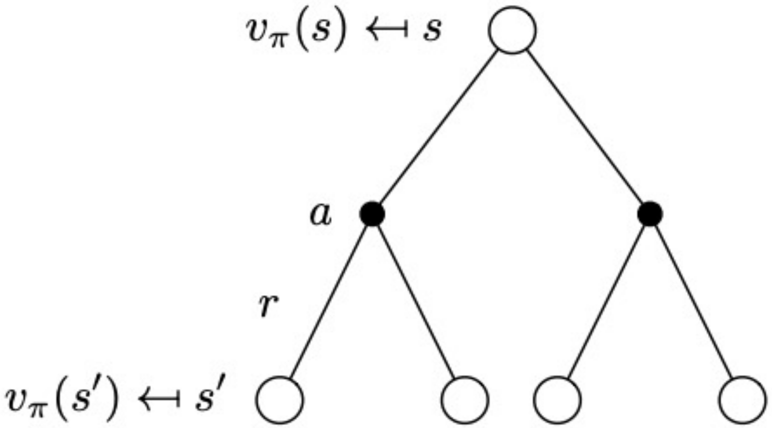
$$q_\pi(s,a) = \boxed{\quad ? \quad}$$

动作值函数(1)



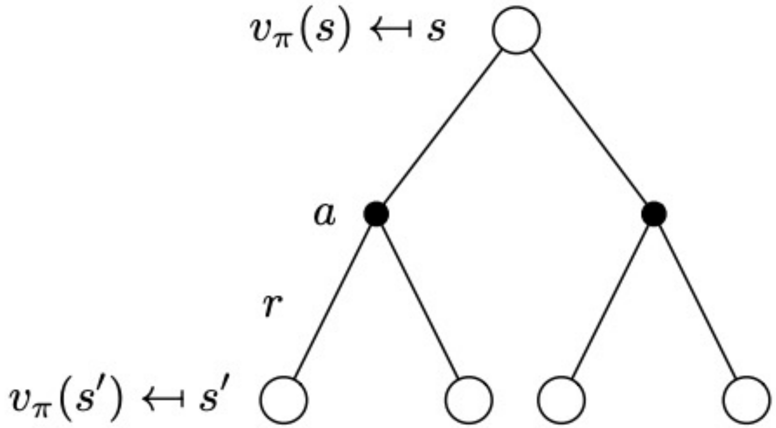
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

状态值函数(2)



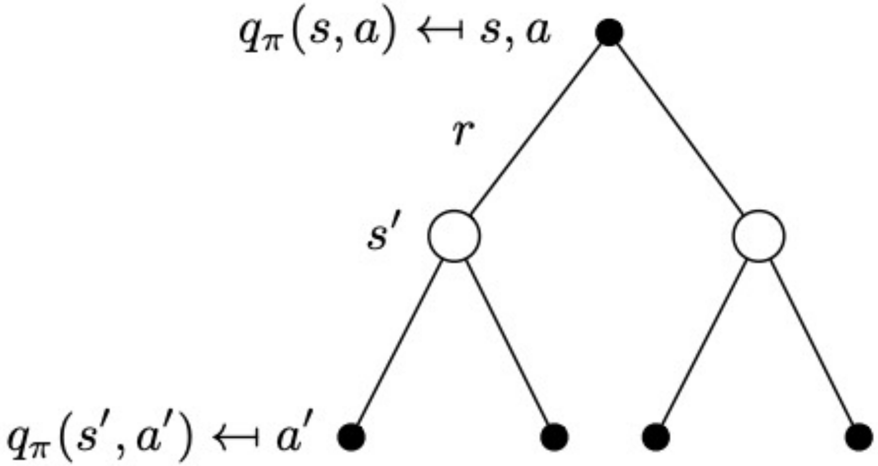
$$v_\pi(s) = \boxed{\quad ? \quad}$$

状态值函数(2)



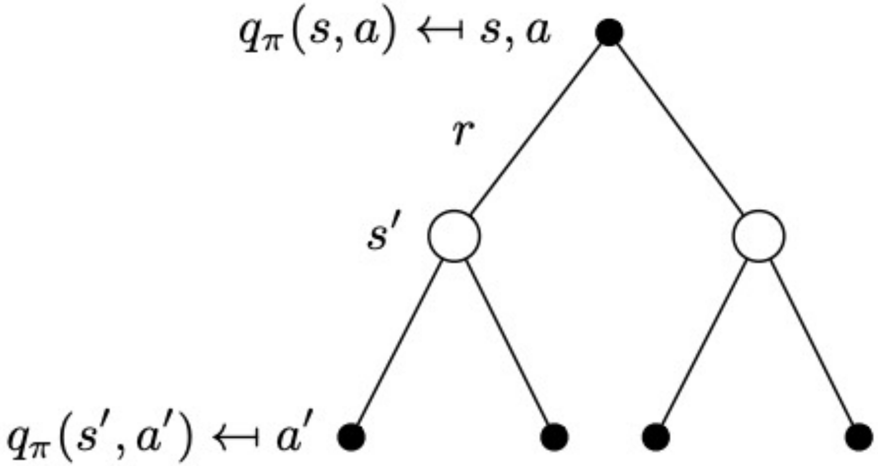
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

动作值函数(2)



$$q_\pi(s,a) = \boxed{\quad ? \quad}$$

动作值函数(2)



$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

Bellman期望公式

Theorem (Bellman Expectation Equations)

Given an MDP, $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, for any policy π, the value functions obey the following expectation equations:

$$v_\pi(s) = \sum_a \pi(s,a) \left[r(s,a) + \gamma \sum_{s'} p(s'|a,s) v_\pi(s') \right] \tag{5}$$

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|a,s) \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a') \tag{6}$$

Bellman最优公式

Theorem (Bellman Optimality Equations)

Given an MDP, $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, the optimal value functions *obey the following expectation equations:*

$$v^*(s) = \max_a \left[r(s,a) + \gamma \sum_{s'} p(s'|a,s) v^*(s') \right] \tag{7}$$

$$q^*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|a,s) \max_{a' \in \mathcal{A}} q^*(s',a') \tag{8}$$

There can be no policy with a higher value than $v_*(s) = \max_\pi v_\pi(s), \forall s$

最优值函数

Definition (Optimal value functions)

The optimal state-value function $v^*(s)$ is the maximum value function over all policies

$$v^*(s) = \max_\pi v_\pi(s)$$

The optimal action-value function $q^*(s, a)$ is the maximum action-value function over all policies

$$q^*(s, a) = \max_\pi q_\pi(s, a)$$

- The optimal value function specifies the best possible performance in the MDP
- An MDP is "solved" when we know the optimal value function

最优策略

Define a partial ordering over policies

$$\pi \geq \pi' \iff v_\pi(s) \geq v_{\pi'}(s) \ , \ \forall s$$

Theorem (Optimal Policies)

For any Markov decision process

- *There exists an optimal policy* π^* *that is better than or equal to all other policies,* $\pi^* \geq \pi, \forall \pi$
  (There can be more than one such optimal policy.)
- *All optimal policies achieve the optimal value function,* $v^{\pi^*}(s) = v^*(s)$
- *All optimal policies achieve the optimal action-value function,* $q^{\pi^*}(s,a) = q^*(s,a)$

从最优值函数到最优策略

An optimal policy can be found by maximising over $q^*(s, a)$,

$$\pi^*(s,a) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ q^*(s,a) \\ 0 & otherwise \end{cases}$$

Observations:

- If we know q*(s, a), we immediately have the optimal policy ?

从最优值函数到最优策略

An optimal policy can be found by maximising over $q^*(s, a)$,

$$\pi^*(s,a) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, q^*(s,a) \\ 0 & otherwise \end{cases}$$

Observations:

- If we know q*(s, a), we immediately have the optimal policy ?
- There is always a deterministic optimal policy for any MDP?

剪刀石头布的游戏中也是这样吗？

从最优值函数到最优策略

An optimal policy can be found by maximising over $q^*(s, a)$,

$$\pi^*(s, a) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\text{argmax}}\ q^*(s, a) \\ 0 & \textit{otherwise} \end{cases}$$

Observations:

- If we know q*(s, a), we immediately have the optimal policy ?
- There is always a deterministic optimal policy for any MDP?
- There can be multiple optimal policies ?
- If multiple actions maximize q*(s,.), we can also just pick any of these(including stochastically) ?

基于动态规划的强化学习

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）

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

MDP目标和策略

- 目标：选择能够最大化累积奖励期望的动作

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

  - $\gamma \in [0,1]$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

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

  - 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

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

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

立即奖励　　状态转移

时间折扣　　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

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) 评估当前策略的价值 $V^{\pi} \quad = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')$

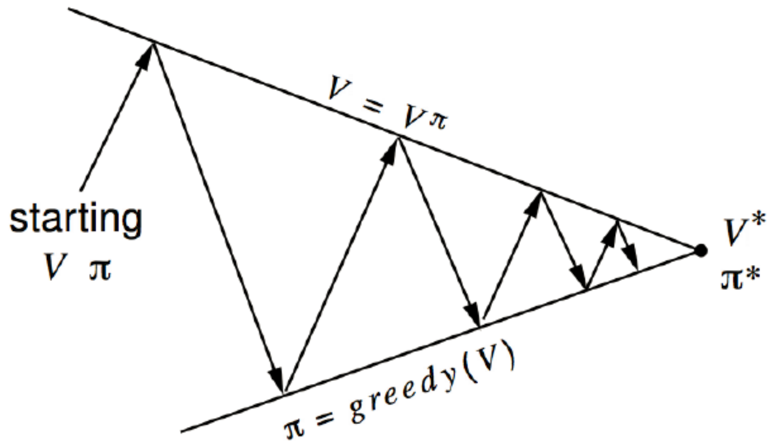
   b) 对每个状态，更新

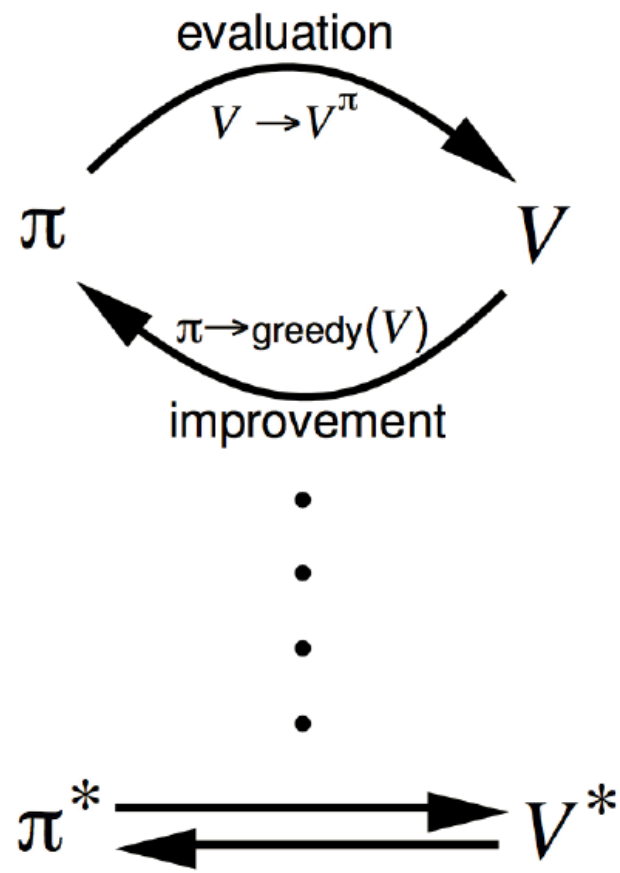
$$\pi(s) = arg \max_{a \in A} \sum_{s' \in S} P_{sa}(s') V(s')$$

}

更新价值函数会很耗时

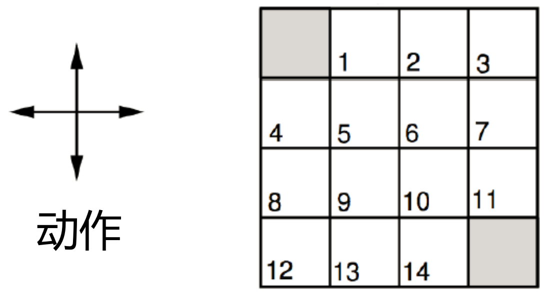
策略迭代





- 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

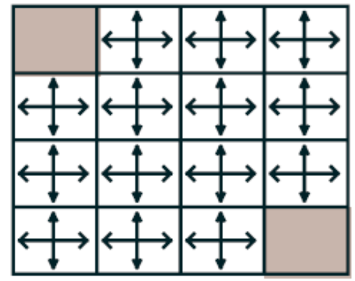
$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

举例：策略评估

随机策略的 V_k | V_k对应的贪心策略

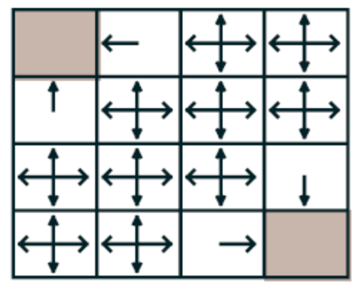
K=0

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	0.0



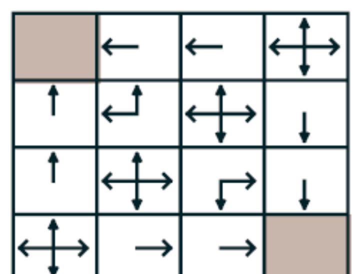
K=1

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



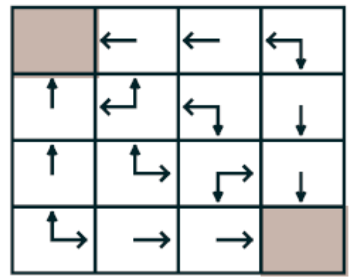
举例：策略评估

随机策略的 V_k

V_k对应的贪心策略

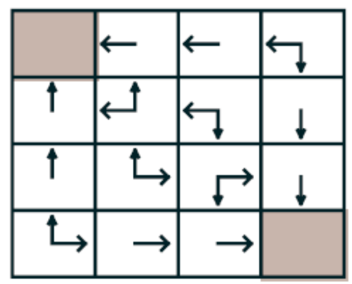
K=3

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



K=10

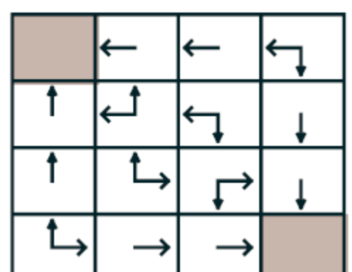
0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0



$V := V^{\pi}$
最优策略

K=∞

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



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

价值迭代收敛性证明：https://towardsdatascience.com/mathematical-analysis-of-reinforcement-learning-bellman-equation-ac9f0954e19f

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

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

THANK YOU