强化学习2024
第2节

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

马尔可夫决策过程

张伟楠 – 上海交通大学

2024年上海交通大学ACM班强化学习课程大纲

强化学习基础部分

（中文课件）

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

强化学习前沿部分

（英文课件）

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. 多智能体强化学习基础
13. 多智能体强化学习前沿
14. 基于扩散模型的强化学习
15. AI Agent与决策大模型
16. 技术交流与回顾

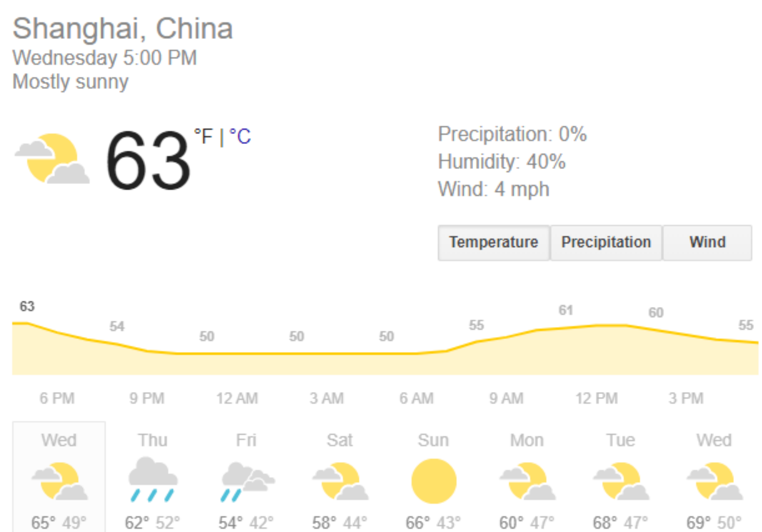
随机过程

- 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



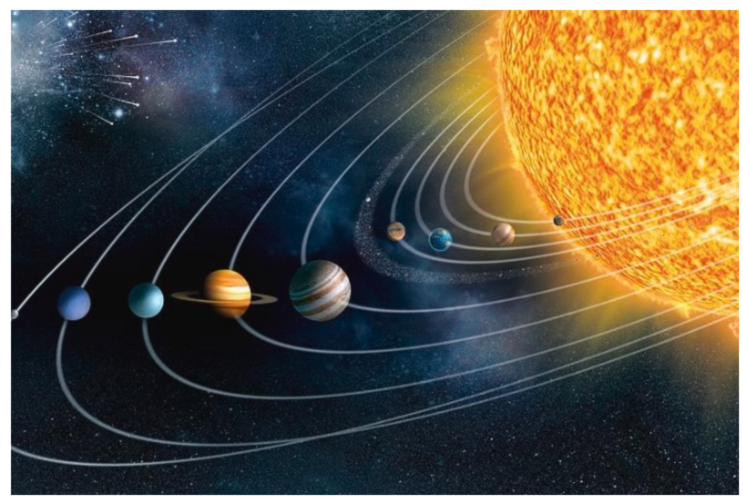
天气变化

随机过程

足球比赛

城市交通

生态系统



星系

马尔可夫过程

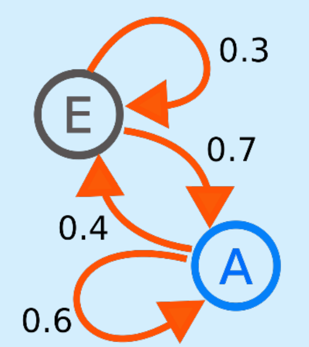
- 马尔可夫过程（Markov Process）是具有马尔可夫性质的随机过程

  "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



一个具有两个转换状态的马尔可夫链

马尔可夫奖励过程

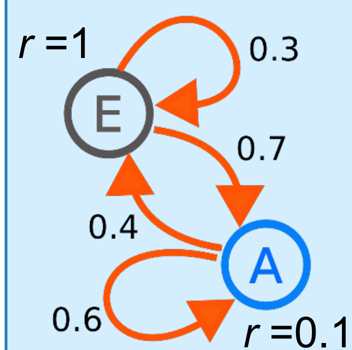
- 在马尔可夫过程的基础上加入奖励函数和折扣因子，就可以得到马尔可夫奖励过程（Markov reward process）

- 主要要素：
  - 状态转移S_t是马尔可夫的

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

  - 基于每个时间步的状态s，环境产生相应的奖励的$r(s)$，其随机变量记为R_t
  - 基于状态序列及其对应的奖励，可以得到序列的回报

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



一个具有两个转换状态的马尔可夫链

马尔可夫奖励过程 – 序列回报的形式

- 问题：为何序列回报需要是如下形式？

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

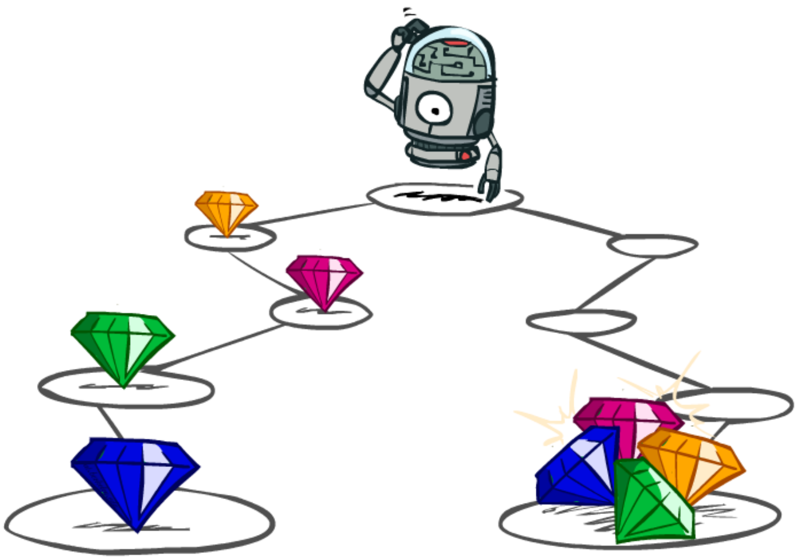
- 需求：我们需要构建序列之间的全序，也即是对任意两个序列，需要有孰好孰坏之分。

- 多还是少？将奖励加和

  [1, 2, 2] $\prec$ [2, 3, 4]

- 近期还是远期？做时间衰减

  [1, 0, 0] $\prec$ [0, 0, 1]



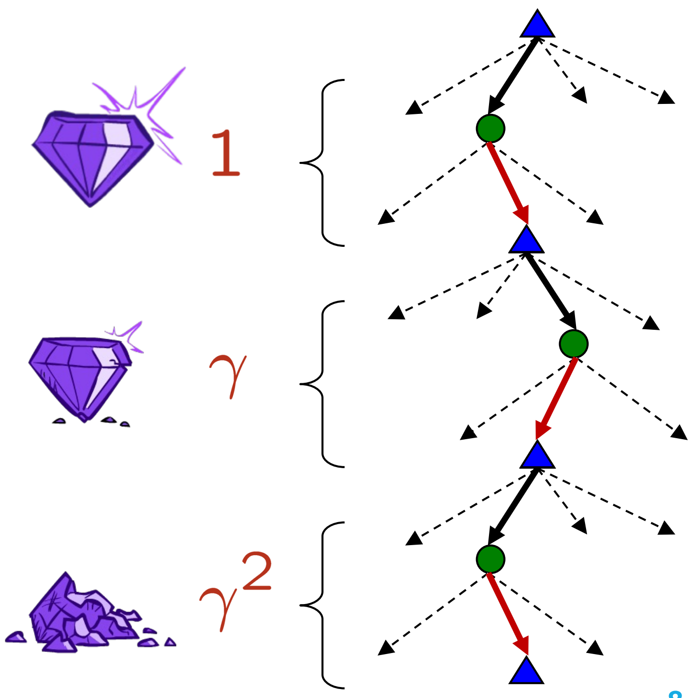
马尔可夫奖励过程 – 序列回报的形式

- 问题：为何序列回报需要是如下形式？

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

- 如何做时间衰减？
  - 在每个时间步，奖励乘上一个衰减因子

    $\gamma \in [0,1]$
  - 可以考虑为每一个时间步都有$1-\gamma$的概率会直接结束该序列，因此未来的奖励需要打折
  - 该衰减因子也帮助算法收敛



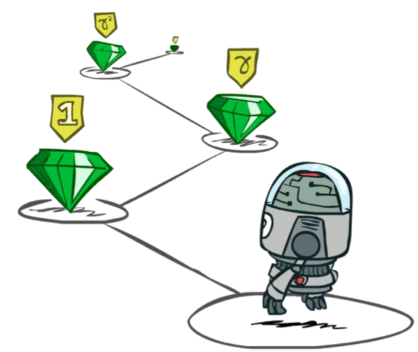
- 以$\gamma = 0.5$为例
  - $G([1,2,3]) = 1\times1 + 0.5\times2 + 0.25\times3$
  - $G([1,2,3]) < G([3,2,1])$

马尔可夫奖励过程 – 序列回报的形式

- 问题：为何序列回报需要是如下形式？

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

- 定理：如果希望对于序列有全序，并满足一下性质



$$[a_1, a_2, \cdots] \prec [b_1, b_2, \cdots] \Leftrightarrow [r, a_1, a_2, \cdots] \prec [r, b_1, b_2, \cdots]$$

那么序列回报形式只会是上述γ-衰减求和形式

- 证明思路：

如果有 $a_0 + \gamma_1 a_1 > b_0 + \gamma_1 b_1 \Leftrightarrow r + \gamma_1 a_0 + \gamma_2 a_1 > r + \gamma_1 b_0 + \gamma_2 b_1$

那么 $(a_0 - b_0) + \gamma_1(a_1 - b_1) > 0 \Leftrightarrow \gamma_1(a_0 - b_0) + \gamma_2(a_1 - b_1) > 0$

因此 $\gamma_2 = \gamma_1^2$ 后续时间步的γ_t可以以此类推

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

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

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

- 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（有时也直接写成$P(\cdot\,|s,a)$）
    - 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
  - 智能体的总回报为

$$r(s_0, a_0) + \gamma r(s_1, a_1) + \gamma^2 r(s_2, a_2) + \cdots$$

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 ⟵

固定数据

强化学习

Agent ⟷

动态环境

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）

Indicator function $\mathbb{I}(z)=1$ 表示事件z发生，否则取0

$$\rho^{\pi}(s,a)=\mathbb{E}\left[\sum_{t=0}^{T}\gamma^{t}\,\mathbb{I}(S_t=s,A_t=a)\,|\,\pi\right],\forall s\in S,a\in A$$

$$=\sum_{t=0}^{T}\gamma^{t}\,P(S_t=s,A_t=a|s_0,\pi)$$

占用度量和策略

- 占用度量（Occupancy Measure）

$$\rho^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t\, \mathbb{I}(S_t = s, A_t = a) \mid \pi\right], \forall s \in S, a \in A$$

- 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）

$$\rho^\pi(s,a) = \sum_{t=0}^{T} \gamma^t P(S_t = s, A_t = a \mid s_0, \pi)$$

- 状态占用度量

$$\rho^\pi(s) = \sum_{t=0}^{T} \gamma^t P(S_t = s \mid s_0, \pi)$$

$$= \sum_{t=0}^{T} \gamma^t P(S_t = s \mid s_0, \pi) \sum_{a'} p(A_t = a' | S_t = s \mid \pi)$$

$$= \sum_{a'} \sum_{t=0}^{T} \gamma^t P(S_t = s, A_t = a | s_0, \pi)$$

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

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

占用度量和累计奖励

- 占用度量（Occupancy Measure）

$$\rho^{\pi}(s,a) = \sum_{t=0}^{T} \gamma^{t} P(S_t = s, A_t = a \mid s_0, \pi)$$

- 策略的累积奖励为

$$V(\pi) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots \mid s_0, \pi]$$

$$= \sum_{s,a} \sum_{t=0}^{T} \gamma^{t} P(S_t = s, A_t = a \mid s_0, \pi)\, r(s,a)$$

$$= \sum_{s,a} \rho^{\pi}(s,a)\, r(s,a) \;=\; \mathbb{E}_{\pi}[r(s,a)]$$

强化学习中的简写

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

MDP中策略的目标

- 策略学习的目标：选择能够最大化累积奖励期望的动作

$$\max_{\pi} \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots \mid s_0, \pi]$$

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

- 如何达到以上学习目标？
  - 策略π和其占用度量ρ^{π}的对应关系是黑盒的，因此以上优化目标并没有直接对π更新方向的指导
  - 在每一个状态s下，策略改变了动作的选择后，策略整体是否变得更优秀了？

“思想总是走在行动的前面，就好像闪电总是走在雷鸣之前。”

德国诗人海涅

策略评估与策略提升

讲师：张伟楠 – 上海交通大学

策略值函数估计（Policy Evaluation）

- 给定环境MDP和策略π，策略值函数估计如下

状态价值

$$V^{\pi}(s) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, \pi]$$

$$= \mathbb{E}_{a \sim \pi(s)}\left[r(s, a) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')\right]$$

$$= \mathbb{E}_{a \sim \pi(s)}[Q^{\pi}(s, a)]$$

动作价值

$$Q^{\pi}(s, a) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, A_0 = a, \pi]$$

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

策略提升（Policy Improvement）

- 对于两个策略π，π'，如果满足如下性质，π'是π的策略提升：
  - 对于任何状态s，有

$$Q^{\pi}\big(s,\pi'(s)\big) \geq V^{\pi}(s)$$

以π来记回报

- 一种特例：给定环境MDP和两个策略π，π'，如果满足如下性质：
  1. 在某个状态s下，两策略的输出不同，并且有

$$\pi'(s) \neq \pi(s) \qquad Q^{\pi}\big(s,\pi'(s)\big) > Q^{\pi}\big(s,\pi(s)\big) = V^{\pi}(s)$$

  2. 在其他所有状态s'下，两策略输出相同，即

$$\pi'(s') = \pi(s') \qquad Q^{\pi}\big(s,\pi'(s)\big) = Q^{\pi}\big(s,\pi(s)\big) = V^{\pi}(s)$$

那么π'是π的一种策略提升

策略提升定理（Policy Improvement Theorem）

- 对于两个策略π，π'，如果满足如下性质，π'是π的策略提升：
  - 对于任何状态s，有

$$Q^{\pi}\big(s,\pi'(s)\big) \geq V^{\pi}(s)$$

以π来记回报

- 进而，π和π'满足：对任何状态s，有

$$V^{\pi'}(s) \geq V^{\pi}(s)$$

以π'来记回报

也即是 π'的策略价值（期望回报）超过π，π'比π更加优秀。

策略提升定理（Policy Improvement Theorem）

- 对于两个策略π，π'，如果满足如下性质，π'是π的策略提升：
  - 对于任何状态s，有$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s)$，因此有$V^{\pi'}(s) \geq V^{\pi}(s)$
- 证明：

$$
\begin{aligned}
V^{\pi}(s) &\leq Q^{\pi}(s, \pi'(s)) \\
&= \mathbb{E}_{\text{Env}}[R_t + \gamma V^{\pi}(S_{t+1}) | S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_t + \gamma V^{\pi}(S_{t+1}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_t + \gamma Q^{\pi}(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_t + \gamma \mathbb{E}_{\text{Env}}[R_{t+1} + \gamma V^{\pi}(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_t + \gamma R_{t+1} + \gamma^2 V^{\pi}(S_{t+2}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 V^{\pi}(S_{t+3}) | S_t = s] \\
&\cdots \\
&\leq \mathbb{E}_{\pi'}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots | S_t = s] \\
&= V^{\pi'}(s)
\end{aligned}
$$

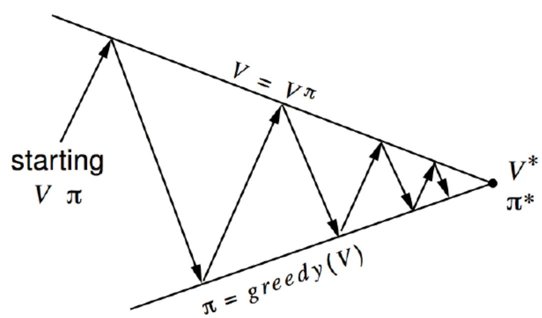
策略提升定理（Policy Improvement Theorem）

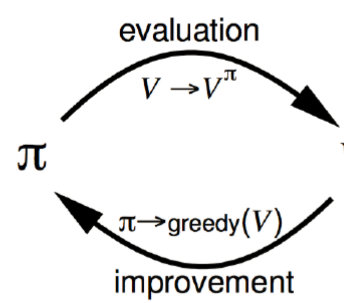
- 对于两个策略π，π'，如果满足如下性质，π'是π的策略提升：
  - 对于任何状态s，有$Q^{\pi}\big(s,\pi'(s)\big) \geq V^{\pi}(s)$
  - 因此有$V^{\pi'}(s) \geq V^{\pi}(s)$

- 策略提升定理带给我们的启示

[找到(s,a)使得 $Q^{\pi}(s,a) \geq V^{\pi}(s)$]

策略 π → 评估策略价值 $V^{\pi}(\cdot)$ → 寻找到新策略π'使得 $Q^{\pi}\big(s,\pi'(s)\big) \geq V^{\pi}(s)$





价值评估指导
策略提升

ε-Greedy 策略提升定理

- 对于m个动作的ε-Greedy策略π

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

- 如果另一个ε-Greedy策略π'是基于Q^π的提升，那么有$V^{\pi'}(s) \geq V^\pi(s)$

$$\begin{aligned} V^{\pi'}(s) \geq \overbrace{Q^\pi(s,\pi'(s))}^{\text{如前面多步推导得出}} &= \sum_{a\in A} \pi'(a|s) Q^\pi(s,a) \\ \text{m actions} \quad &= \frac{\epsilon}{m}\sum_{a\in A} Q^\pi(s,a) + (1-\epsilon)\max_{a\in A} Q^\pi(s,a) \\ &\geq \frac{\epsilon}{m}\sum_{a\in A} Q^\pi(s,a) + (1-\epsilon)\sum_{a\in A}\frac{\pi(a|s)-\epsilon/m}{1-\epsilon} Q^\pi(s,a) \\ &= \sum_{a\in A}\pi(a|s)Q^\pi(s,a) = V^\pi(s) \end{aligned}$$

注意：
1. ε不能大
2. π并不是ε-Greedy(Q^π)，π'才是

- 延升思考：对于随机策略，将策略的动作选择分布移向价值更高的动作

基于动态规划的强化学习

讲师：张伟楠 – 上海交通大学

策略值函数估计（Policy Evaluation）

- 给定环境MDP和策略π，策略值函数估计如下

状态价值 $$V^{\pi}(s) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, \pi]$$

$$= \mathbb{E}_{a \sim \pi(s)}\left[r(s, a) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')\right]$$ Bellman等式

立即奖励

状态转移

时间折扣

下一个状态的价值

策略值函数估计（Policy Evaluation）

- 给定环境MDP和策略π，策略值函数估计如下

状态价值 $$V^\pi(s) = \mathbb{E}[r(S_0,A_0)+\gamma r(S_1,A_1)+\gamma^2 r(S_2,A_2)+\cdots|S_0=s,\pi]$$

$$= \mathbb{E}_{a\sim\pi(s)}\left[r(s,a)+\gamma\sum_{s'\in S}P_{s\pi(s)}(s')V^\pi(s')\right]$$ Bellman等式

$$= \mathbb{E}_{a\sim\pi(s)}[Q^\pi(s,a)]$$

动作价值 $$Q^\pi(s,a) = \mathbb{E}[r(S_0,A_0)+\gamma r(S_1,A_1)+\gamma^2 r(S_2,A_2)+\cdots|S_0=s,A_0=a,\pi]$$

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

$$= r(s,a)+\gamma\sum_{s'\in S}P_{s\pi(s)}(s')\sum_{a'\in A}\pi(a'|s')\,Q^\pi(s',a')$$ Bellman等式

MDP中策略的目标

- 策略的目标：选择能够最大化累积奖励期望的动作

$$\max_{\pi} \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots \mid s_0, \pi]$$

  - $\gamma \in [0,1]$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）
  - $r(s)$和$r(s,a)$的设定是类似的，只需设$r(s) = r(s,a)$

- 给定一个确定性策略 $\pi(\cdot): S \to A$
  - 即，在状态 s 下采取动作 $a = \pi(s)$

- 给策略π定义价值函数

$$V^{\pi}(s) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, \pi]$$

$$Q^{\pi}(s,a) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, A_0 = a, \pi]$$

寻找优化策略的方法：策略迭代和价值迭代

- 价值函数和策略相关

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

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

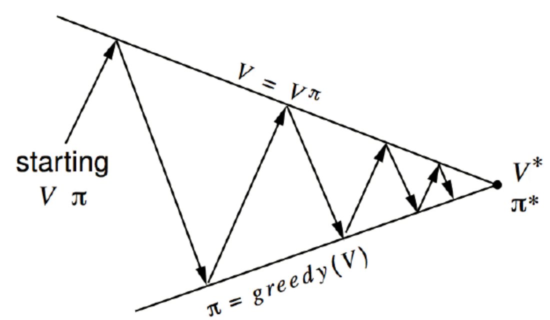
- 可以对最优价值函数和最优策略执行迭代更新
  - 策略迭代
  - 价值迭代

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（基于V价值函数）



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

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

}

更新价值函数会很耗时

策略迭代

- 对于一个动作空间和状态空间有限的MDP

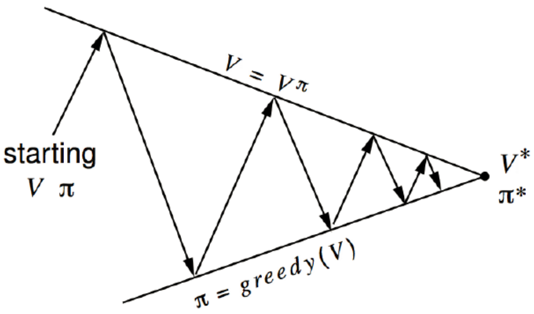
$$|S| < \infty, |A| < \infty$$

- 策略迭代过程（基于Q价值函数）

  1. 随机初始化策略 π
  2. 重复以下过程直到收敛{
     a) 计算 $Q \coloneqq Q^{\pi}$
     b) 对每个状态，更新

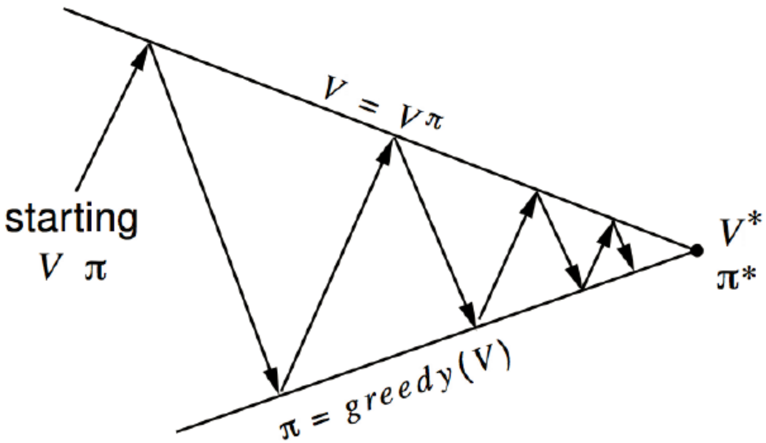
$$\pi(s) = \arg\max_{a \in A} Q(s, a)$$

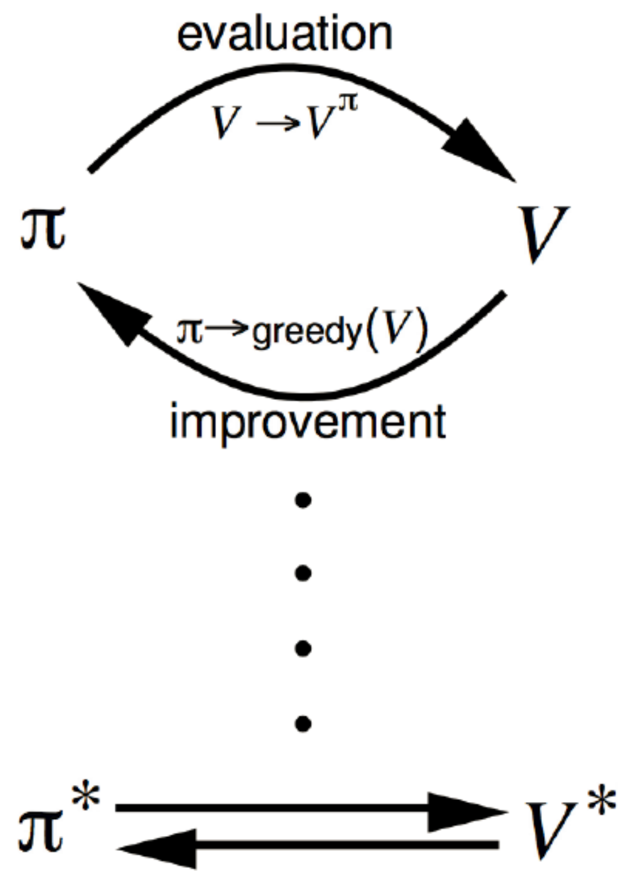
}



更新价值函数会很耗时

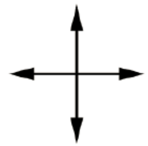
策略迭代



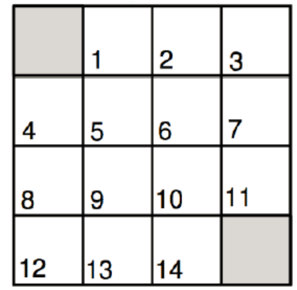


- 策略评估
  - 估计V^π
  - 迭代的评估策略
- 策略改进
  - 生成 $\pi' \geq \pi$
  - 贪心策略改进（后续进一步讨论）

举例：策略评估



动作



- 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

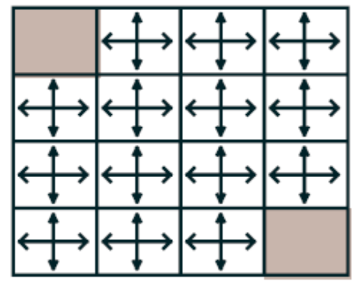
$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

举例：策略评估

随机策略的 V_k | V_k对应的贪心策略

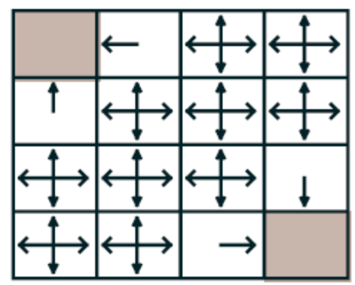
K=0

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



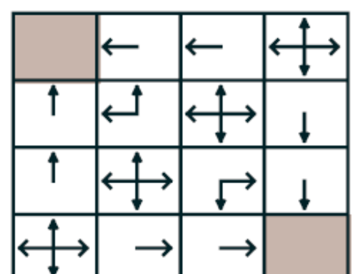
K=1

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

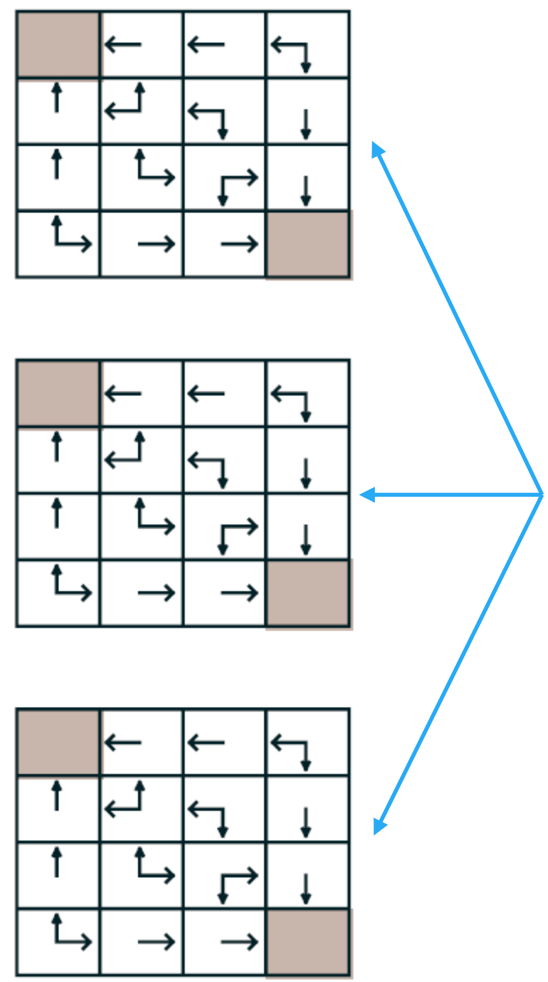
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

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

K=∞

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

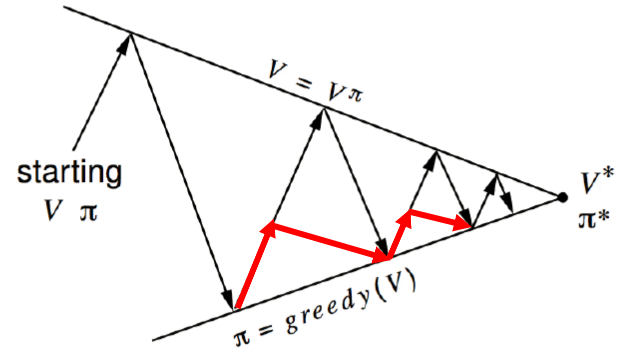


$V := V^{\pi}$
最优策略

思考：如何加速策略迭代方法？

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

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

}

更新价值函数会很耗时，但前面的例子中，迭代计算 V 价值函数为收敛时，其导出的策略已经是最优

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

同步 vs. 异步价值迭代

- 同步的价值迭代会储存两份价值函数的拷贝
  1. 对S中的所有状态s
  $$V_{new}(s) \leftarrow \max_{a \in A}\left(r(s) + \gamma \sum_{s' \in S} P_{sa}(s') V_{old}(s')\right)$$
  2. 更新 $V_{old}(s) \leftarrow V_{new}(s)$

- 异步价值迭代只储存一份价值函数
  1. 对S中的所有状态s
  $$V(s) \leftarrow \max_{a \in A}\left(r(s) + \gamma \sum_{s' \in S} P_{sa}(s') V(s')\right)$$

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

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（Bellman optimality equation）

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

- 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 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) 计算价值 $V := V^{\pi}$
   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
   - 相当于策略评估中进行一轮价值更新，然后直接根据更新后的价值进行策略提升
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

基于模型的强化学习

讲师：张伟楠 – 上海交通大学

学习一个MDP模型

- 目前我们关注在给出一个已知MDP模型后（也就是说，状态转移$P_{sa}(s')$和奖励函数$r(s)$明确给定后）
  - 计算最优价值函数
  - 学习最优策略
- 在实际问题中，状态转移和奖励函数一般不是明确给出的
  - 比如，我们只看到了一些episodes

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

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

学习一个MDP模型

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

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

⋮ ⋮

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

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

  - 学习奖励函数$r(s)$，也就是立即奖赏期望

$$r(s) = \text{average}\{r(s)^{(i)}\}$$

学习模型&优化策略

- 算法

1. 随机初始化策略π

2. 重复以下过程直到收敛 {
   a) 在MDP中执行π，收集经验数据
   b) 使用MDP中的累积经验更新对P_{sa}和R的估计
   c) 利用对P_{sa}和r的估计执行价值迭代，得到新的估计价值函数V
   d) 根据V更新策略π为贪心策略

}

学习一个MDP模型

- 在实际问题中，状态转移和奖励函数一般不是明确给出的
  - 比如，我们只看到了一些episodes

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

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

- 另一种解决方式是不学习MDP，从经验中直接学习价值函数和策略
  - 也就是无模型的强化学习（Model-free Reinforcement Learning）

马尔可夫决策过程总结

- MDP由一个五元组构成 $(S, A, \{P_{sa}\}, \gamma, r)$，其中状态转移$P$和奖励函数$r$构成了动态系统

- 动态系统和策略交互的占用度量

$$\rho^{\pi}(s,a) = \sum_{t=0}^{T} \gamma^{t} P(S_t = s, A_t = a \mid s_0, \pi)$$

- 一个白盒环境给定的情况下，可用动态规划的方法求解最优策略
  - 值迭代和策略迭代
  - 策略迭代源自策略提升定理

- 如果环境是黑盒的，可以根据统计信息来拟合出动态环境P和r，然后做动态规划求解最优策略，但是这样的情况更多由无模型强化学习来解决

THANK YOU