

周次	日期	内容	作业	编程	项目	内容
1	9月15日	绪论				绪论。课程内容。Python 简介
2	9月22日	无信息搜索		1		状态空间表示, 宽度优先, 一致代价, 深度优先
3	9月29日	有信息搜索	1		1	算法复杂度分析。A*算法。问题求解项目
4	10月 6日	约束满足		2		启发函数的设计, 约束满足问题, 回溯搜索
5	10月13日	对抗搜索	2			局部搜索, 极小极大搜索, 蒙特卡洛树搜索
6	10月20日	命题逻辑	3			命题逻辑, 演绎定理, 归结原理
7	10月27日	谓词逻辑	4			谓词逻辑
8	11月 3日	线性回归	5			线性回归
9	11月10日	Logistic 回归		3	2	Logistic 回归, Softmax 回归, 机器学习项目
10	11月17日	前馈神经网络	6			前馈神经网络, 深度学习框架
11	11月24日	卷积神经网络		4		卷积神经网络, 深度学习框架
12	12月 1日	马尔可夫决策过程	7			强化学习的数学基础
13	12月 8日	策略迭代与价值迭代		5		状态转移概率已知时的预测与控制
14	12月15日	蒙特卡洛与时序差分	8			状态转移概率未知时的预测与控制
15	12月22日	深度强化学习				价值函数的近似, 深度强化学习。考试解读

问题求解
逻辑推理
机器学习

人工智能基础

Fundamentals of Artificial Intelligence

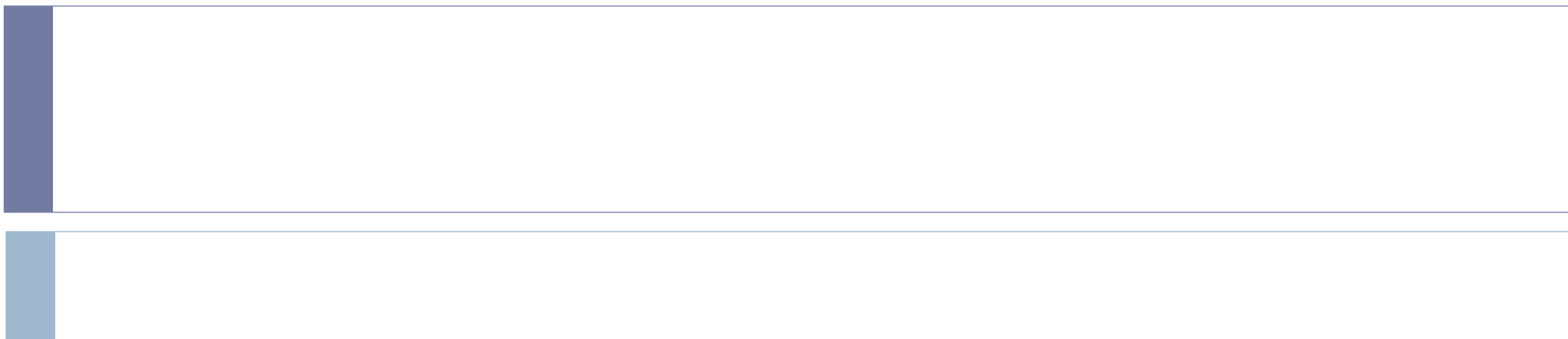


江瑞

自动化系

马尔可夫决策过程

Markov Decision Process



马尔可夫过程

Markov Process

s	1	2	3	4	5	6	7
名字	回归	分类	编程	提交	微信	溜达	睡觉

- 考虑两个相邻时刻，定义**状态转移概率**

$$p_{ss'} = P(S_{t+1} = s' | S_t = s)$$

- 考虑所有状态，有**状态转移矩阵**

$$P = \begin{pmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & p_{ss'} & \vdots \\ p_{n1} & \cdots & p_{nn} \end{pmatrix}$$

从行 \rightarrow 转移到列 \uparrow

$$\sum_{s'=1}^n p_{ss'} = 1$$

每行之和为1

- 满足马尔可夫性的随机变量序列 S_1, S_2, \dots 称为一个马尔可夫过程，由二元组 (S, P) 描述

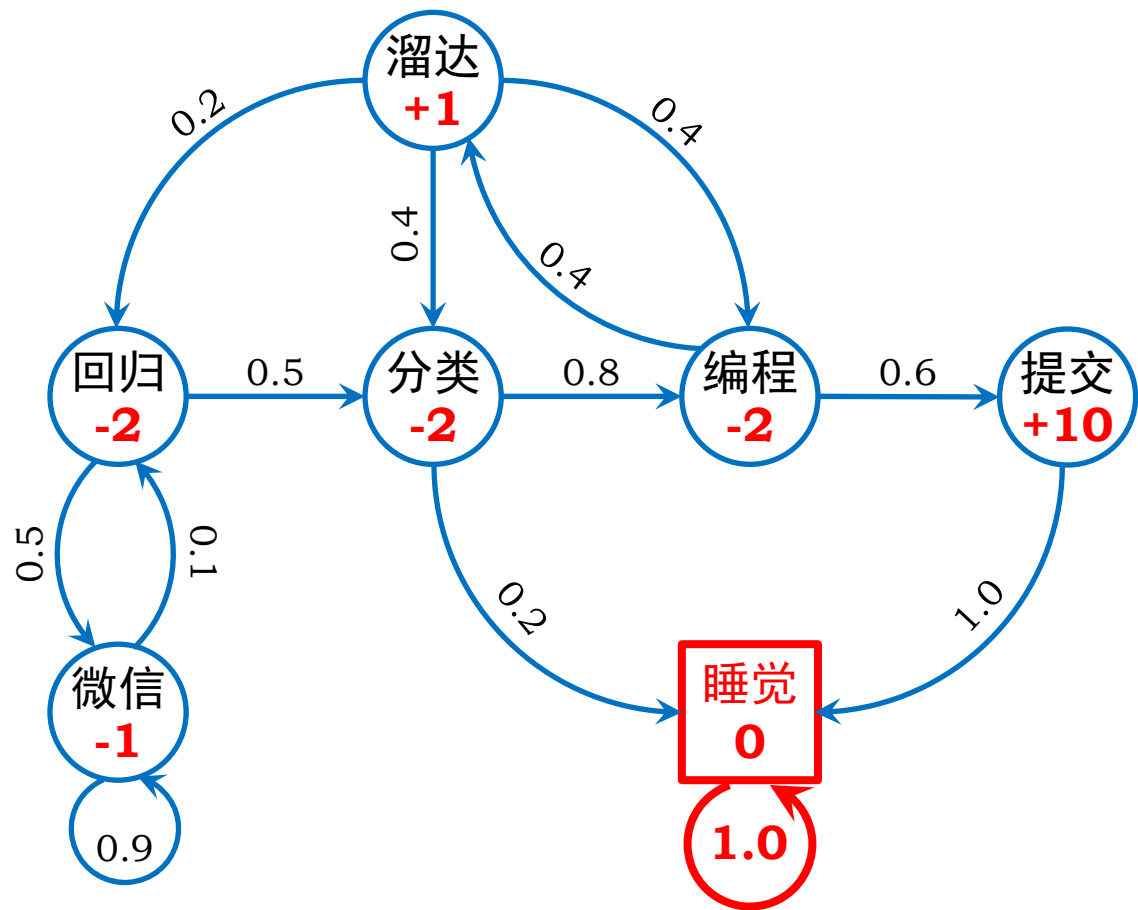
- $S = \{s_1, \dots, s_n\}$ 称为状态空间

- $P = (p_{ss'})_{n \times n}$ 称为状态转移矩阵

$$p_{ss'} = P(S_{t+1} = s' | S_t = s)$$

马尔可夫回报过程

Markov Reward Process



马尔可夫回报过程是对状态（节点）建模

▶ 状态转移 $S_t \rightarrow S_{t+1}$ 产生回报 R_{t+1}

▶ 随机变量序列 $S_1, R_2, S_2, R_3, \dots$ 构成一个马尔可夫回报过程, 由四元组 $(S, P, \mathbf{r}, \gamma)$ 描述

▶ 状态空间: $S = \{s_1, \dots, s_n\}$

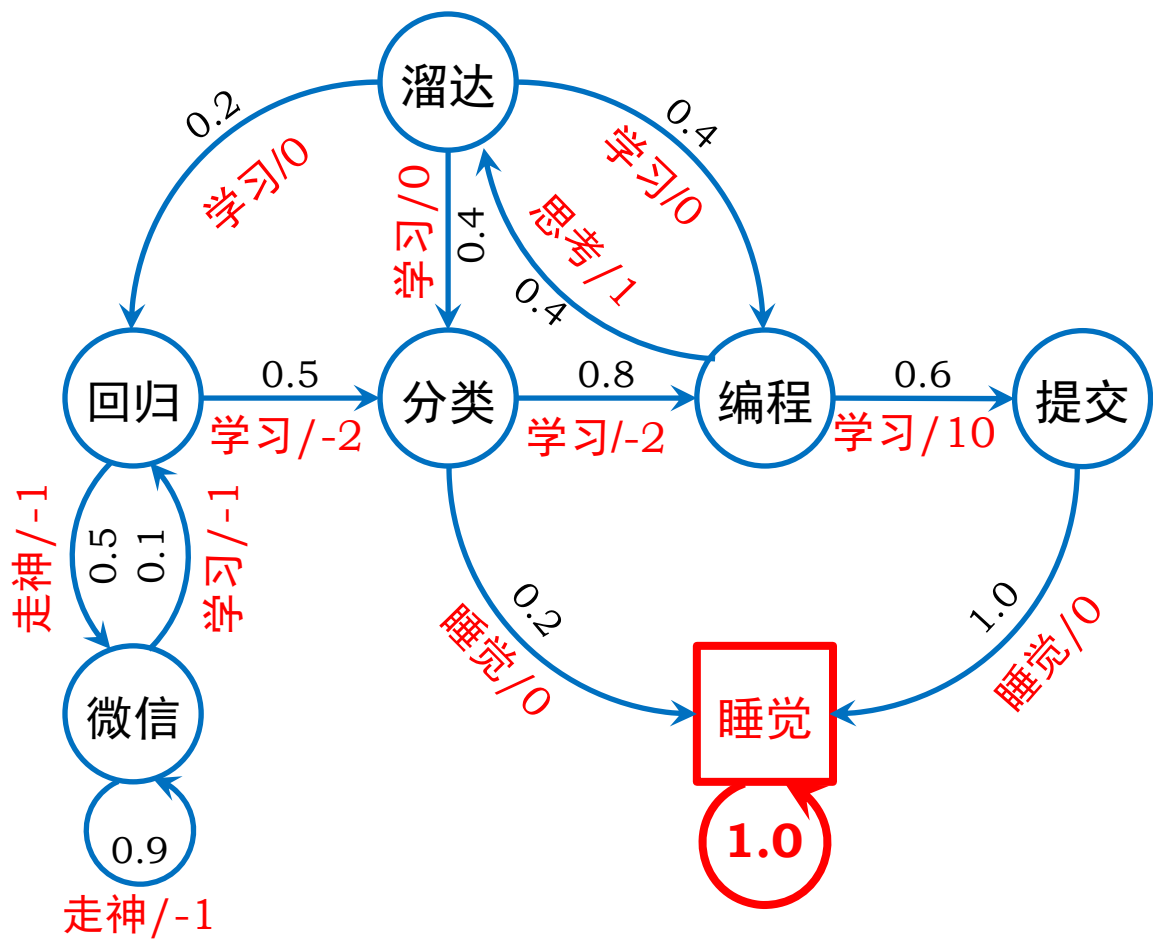
▶ 状态转移矩阵: $P = (p_{ss'})_{n \times n}$
 $p_{ss'} = P(S_{t+1} = s' | S_t = s)$

▶ 状态期望回报: $\mathbf{r} = (r_1, \dots, r_n)$
 $r_s = E[R_{t+1} | S_t = s]$

▶ 折现因子: $\gamma \in [0, 1]$

$$E[R_{t+1} | S_t = s] = \sum_{r \in \mathbf{R}} rp(r | S_t = s)$$

马尔可夫决策过程



马尔可夫决策过程是对行动（边）建模

Markov Decision Process

- ▶ 行动 A_t 导致状态转移 $S_t \rightarrow S_{t+1}$
产生回报 R_{t+1}
- ▶ 随机变量序列 $S_1, A_1, R_2, S_2, A_2, R_3, \dots$
构成一个马尔可夫决策过程,
由五元组 (S, A, P, R, γ) 描述
- ▶ 状态空间: $S = \{s_1, \dots, s_n\}$
- ▶ 行动空间: $A = \{a_1, \dots, a_m\}$
- ▶ 状态转移: $P = (p_{ss'}^a)_{n \times n \times m}$
$$p_{ss'}^a = P(S_{t+1} = s' \mid S_t = s, A_t = a)$$
- ▶ 行动期望回报: $R = (r_s^a)_{n \times m}$
$$r_s^a = E[R_{t+1} \mid S_t = s, A_t = a]$$
- ▶ 折现因子: $\gamma \in [0, 1]$

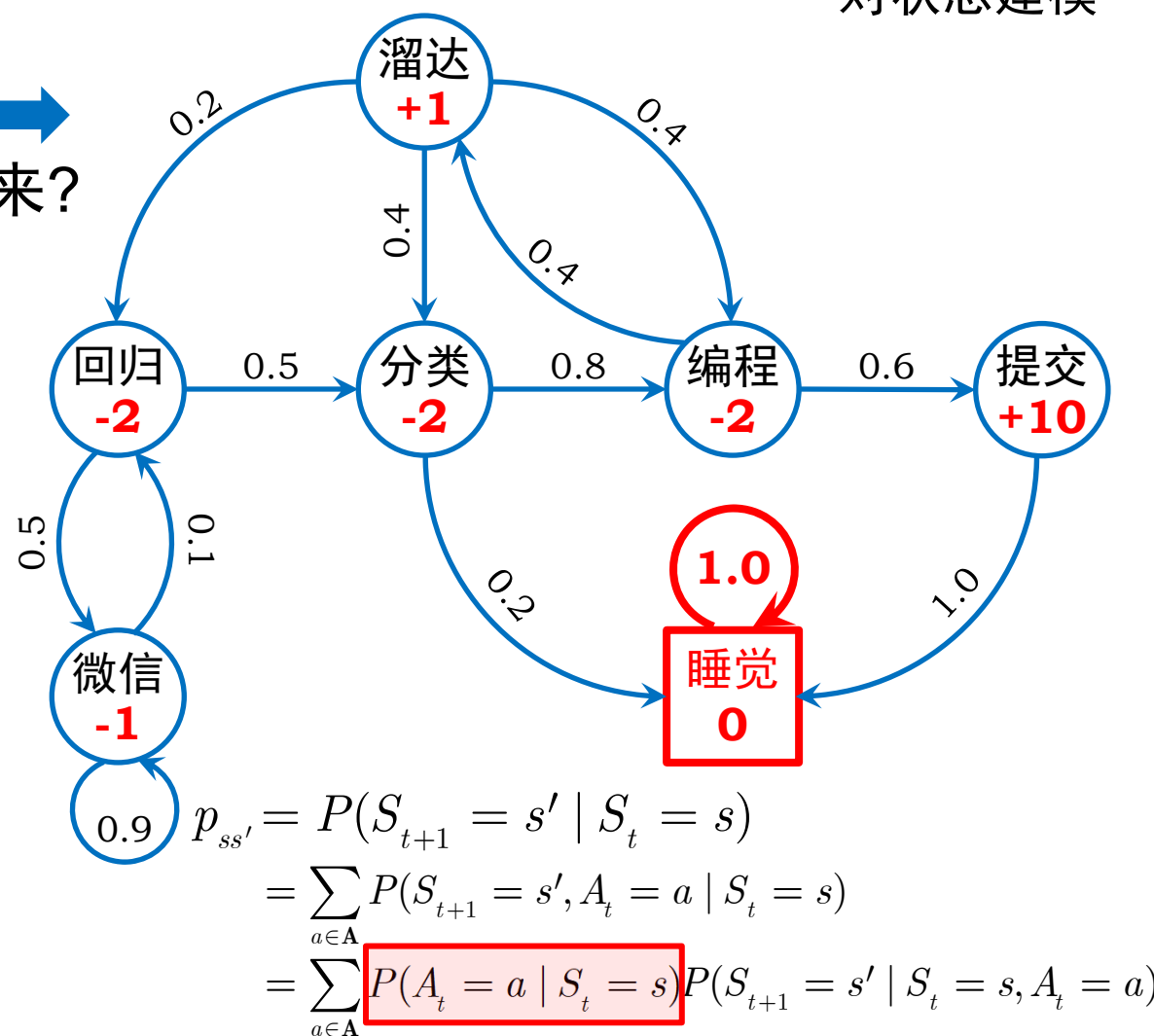
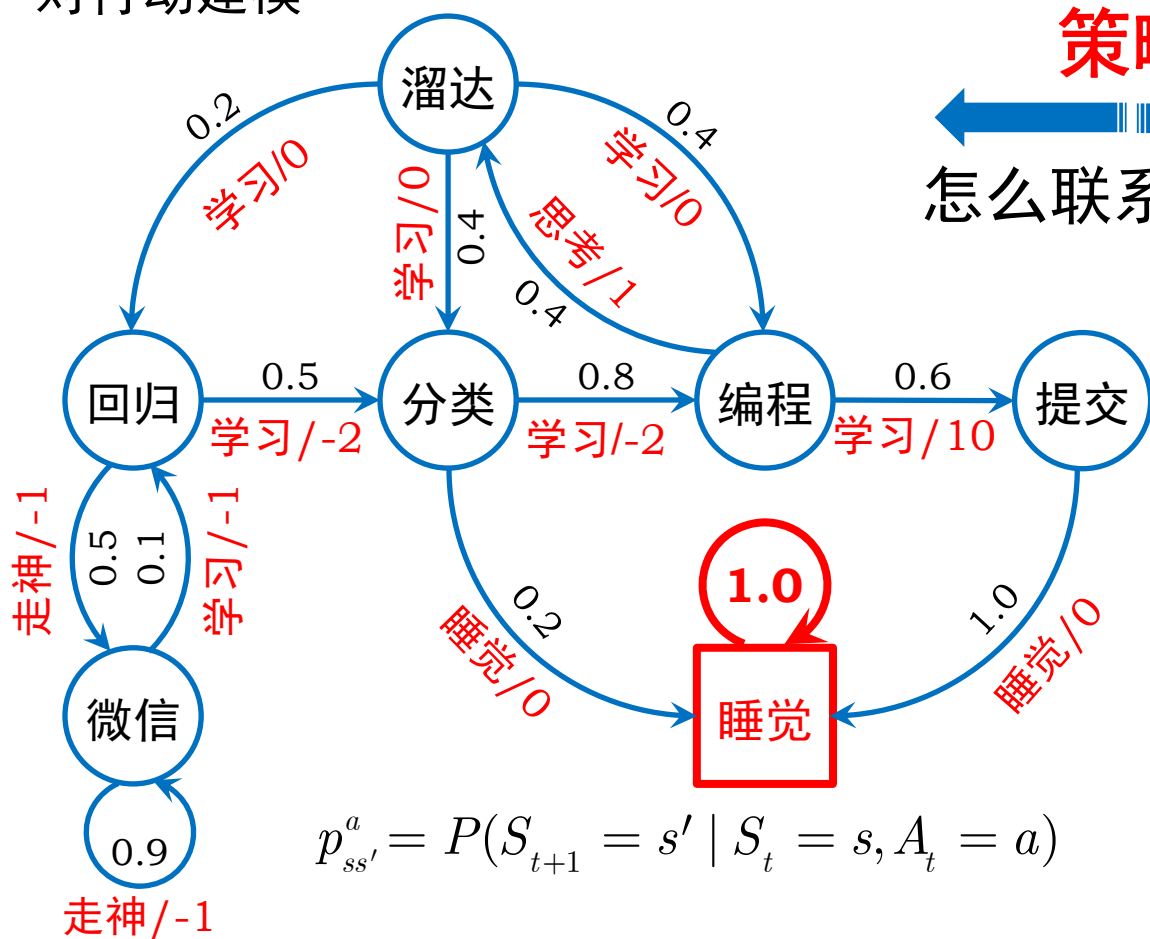
马尔可夫决策过程

对行动建模

$$\pi(a | s) = P(A_t = a | S_t = s)$$

马尔可夫回报过程

对状态建模



状态价值

行动价值

▶ 状态价值函数

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

▶ 状态价值是同时刻行动价值的期望

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

▶ 状态价值贝尔曼期望方程

$$v_{\pi}(s) = \sum_{a \in \mathbf{A}} \pi(a \mid s) \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{\pi}(s') \right)$$

▶ 行动价值函数

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

▶ 行动价值由后续状态价值的期望计算

$$q_{\pi}(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{\pi}(s')$$

▶ 行动价值贝尔曼期望方程

$$q_{\pi}(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a \sum_{a' \in \mathbf{A}} \pi(a' \mid s') q_{\pi}(s', a')$$

如何评价已有策略 — 预测问题

如何找到最优策略 — 控制问题

本课程建模 ... — ... 四元组建模

- ▶ 行动 A_t 导致状态转移 $S_t \rightarrow S_{t+1}$ 产生回报 R_{t+1}
- ▶ 随机变量序列 $S_1, A_1, R_2, S_2, A_2, R_3, \dots$ 构成一个马尔可夫决策过程，由五元组 $(\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{R}, \gamma)$ 描述

- ▶ 状态空间: $\mathbf{S} = \{s_1, \dots, s_n\}$

- ▶ 行动集合: $\mathbf{A} = \{a_1, \dots, a_m\}$

- ▶ 状态转移: $\mathbf{P} = (p_{ss'}^a)_{n \times n \times m}$

$$p_{ss'}^a = P(S_{t+1} = s' \mid S_t = s, A_t = a)$$

- ▶ 行动期望回报: $\mathbf{R} = (r_s^a)_{n \times m}$

$$r_s^a = E[R_{t+1} \mid S_t = s, A_t = a]$$

- ▶ 折现因子: $\gamma \in [0, 1]$

$$p(s', r \mid s, a) = P(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$$

- ▶ 状态转移

$$p_{ss'}^a = P(S_{t+1} = s' \mid S_t = s, A_t = a)$$

$$= \sum_{r \in \mathbf{R}} P(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$$

$$= \sum_{r \in \mathbf{R}} p(s', r \mid s, a)$$

- ▶ 行动期望回报

$$r_s^a = E[R_{t+1} \mid S_t = s, A_t = a]$$

$$= \sum_{r \in \mathbf{R}} r p(R_{t+1} = r \mid S_t = s, A_t = a)$$

$$= \sum_{r \in \mathbf{R}} r \sum_{s' \in \mathbf{R}} p(s', r \mid s, a)$$

- ▶ 行动 A_t 导致状态转移 $S_t \rightarrow S_{t+1}$
产生回报 R_{t+1}

- ▶ 随机变量序列 $S_1, A_1, R_2, S_2, A_2, R_3, \dots$
构成一个马尔可夫决策过程，由五元组
(**S**, **A**, **P**, **R**, γ) 描述

- ▶ 状态空间: $\mathbf{S} = \{s_1, \dots, s_n\}$

- ▶ 行动集合: $\mathbf{A} = \{a_1, \dots, a_m\}$

- ▶ **状态转移: $\mathbf{P} = (p_{ss'}^a)_{n \times n \times m}$**

$$p_{ss'}^a = P(S_{t+1} = s' \mid S_t = s, A_t = a)$$

- ▶ 行动期望回报: $\mathbf{R} = (r_s^a)_{n \times m}$

$$r_s^a = E[R_{t+1} \mid S_t = s, A_t = a]$$

- ▶ 折现因子: $\gamma \in [0, 1]$

$$p(s', r \mid s, a) = P(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$$

- ▶ 状态转移

$$p_{ss'}^a = P(S_{t+1} = s' \mid S_t = s, A_t = a)$$

$$= \sum_{r \in \mathbf{R}} P(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$$

$$= \sum_{r \in \mathbf{R}} p(s', r \mid s, a)$$

- ▶ 行动期望回报

$$r_s^a = E[R_{t+1} \mid S_t = s, A_t = a]$$

$$= \sum_{r \in \mathbf{R}} r p(R_{t+1} = r \mid S_t = s, A_t = a)$$

$$= \sum_{r \in \mathbf{R}} r \sum_{s' \in \mathbf{R}} p(s', r \mid s, a)$$

预测问题

- ▶ 计算价值函数，评价给定策略
- ▶ 基于价值函数
 - ▶ 基于状态转移模型 (Model-based)
 - ▶ 线性方程
 - ▶ 动态规划
 - ▶ 不基于状态转移模型 (Model-free)
 - ▶ 蒙特卡洛预测
 - ▶ 时序差分预测
 - ▶ 近似方法
 - ▶ 线性回归
 - ▶ 深度学习
- ▶ 基于策略模型

控制问题

- ▶ 优化价值函数，获得最优策略
- ▶ 基于价值函数
 - ▶ 基于状态转移模型
 - ▶ 策略迭代
 - ▶ 价值迭代
 - ▶ 不基于状态转移模型
 - ▶ 蒙特卡洛控制
 - ▶ 时序差分控制：SARSA, Q-learning
 - ▶ 近似方法
 - ▶ 线性回归
 - ▶ 深度学习：DQN
- ▶ 基于策略模型

求解贝尔曼期望方程

▶ 状态价值函数

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

▶ 状态价值是同时刻行动价值的期望

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

▶ 状态价值贝尔曼期望方程

$$v_{\pi}(s) = \sum_{a \in \mathbf{A}} \pi(a \mid s) \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{\pi}(s') \right)$$

$$p_{ss'}^{\pi} = \sum_{a \in \mathbf{A}} \pi(a \mid s) p_{ss'}^a$$

$$r_s^{\pi} = \sum_{a \in \mathbf{A}} \pi(a \mid s) r_s^a$$

▶ 行动价值函数

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

▶ 行动价值由后续状态价值的期望计算

$$q_{\pi}(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{\pi}(s')$$

▶ 行动价值贝尔曼期望方程

$$q_{\pi}(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a \sum_{a' \in \mathbf{A}} \pi(a' \mid s') q_{\pi}(s', a')$$



$$v_{\pi}(s) = r_s^{\pi} + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^{\pi} v_{\pi}(s')$$

$$\mathbf{v}_{\pi} = \mathbf{r}^{\pi} + \gamma \mathbf{P}^{\pi} \mathbf{v}_{\pi}$$

$$\mathbf{v}_{\pi} = (\mathbf{I} - \gamma \mathbf{P}^{\pi})^{-1} \mathbf{r}^{\pi}$$

扫地机器人的期望累积回报

$$\mathbf{v}_{\pi} = (\mathbf{I} - \gamma \mathbf{P}^{\pi})^{-1} \mathbf{r}^{\pi}$$

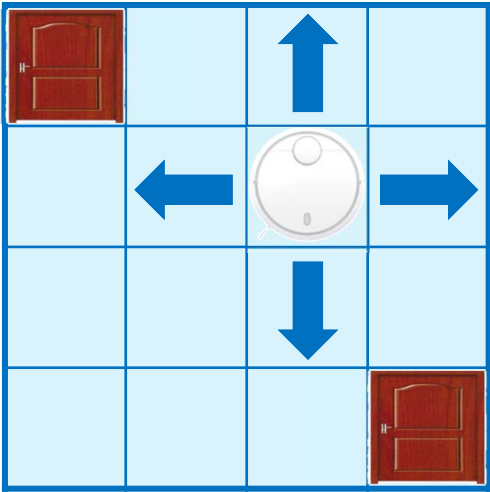
状态空间

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

即时回报

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

$$\pi(a | s) = \frac{1}{4} \text{ for all actions}$$



状态转移矩阵

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1.0															
1	.25	.25	.25			.25										
2		.25	.25	.25			.25									
3			.25	.50				.25								
4	.25				.25	.25			.25							
5		.25			.25		.25			.25						
6			.25		.25		.25		.25		.25					
7				.25		.25	.25					.25				
8					.25		.25	.25		.25			.25			
9						.25		.25	.25		.25			.25		
10							.25		.25	.25		.25			.25	
11								.25		.25	.25					.25
12									.25			.50	.25			
13										.25			.25	.25	.25	
14											.25			.25	.25	.25
15																1.0

$\gamma = 1.0$

0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0

$\gamma = 0.8$

0.0	-3.4	-4.3	-4.5
-3.4	-4.1	-4.4	-4.3
-4.3	-4.4	-4.1	-3.4
-4.5	-4.3	-3.4	0.0

$\gamma = 0.6$

0.0	-2.0	-2.4	-2.5
-2.0	-2.3	-2.4	-2.4
-2.4	-2.4	-2.3	-2.0
-2.5	-2.4	-2.0	0.0

$\gamma = 0.4$

0.0	-1.5	-1.6	-1.7
-1.5	-1.6	-1.7	-1.6
-1.6	-1.7	-1.6	-1.5
-1.7	-1.6	-1.5	0.0

$\gamma = 0.2$

0.0	-1.2	-1.2	-1.2
-1.2	-1.2	-1.2	-1.2
-1.2	-1.2	-1.2	-1.2
-1.2	-1.2	-1.2	0.0

$\gamma = 0.0$

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

扫地机器人的期望累积回报

$$\mathbf{v}_{\pi} = (\mathbf{I} - \gamma \mathbf{P}^{\pi})^{-1} \mathbf{r}^{\pi}$$

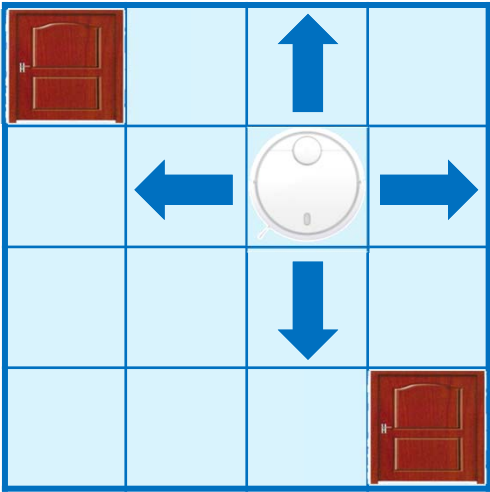
状态空间

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

即时回报

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

$$\pi(a | s) = \begin{cases} 0.4 & \text{up, left} \\ 0.1 & \text{down, right} \end{cases}$$



状态转移矩阵

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1.															
1	.4	.4	.1			.1										
2		.4	.4	.1			.1									
3			.4	.5				.1								
4	.4				.4	.1			.1							
5		.4			.4	.1				.1						
6			.4		.4	.1					.1					
7				.4		.4	.1					.1				
8					.4		.4	.1					.1			
9						.4		.4	.1					.1		
10							.4		.4	.1					.1	
11								.4		.4	.1					.1
12									.4			.5	.1			
13										.4			.4	.1	.1	
14											.4			.4	.1	.1
15																1.

$\gamma = 1.0$

0.0	-3.8	-7.2	-9.8
-3.8	-5.7	-8.1	-10
-7.2	-8.1	-9.4	-9.8
-9.8	-10	-9.8	0.0

$\gamma = 0.8$

0.0	-2.2	-3.4	-4.0
-2.2	-3.0	-3.7	-4.1
-3.4	-3.7	-4.0	-3.9
-4.0	-4.1	-3.9	0.0

$\gamma = 0.6$

0.0	-1.7	-2.2	-2.4
-1.7	-2.1	-2.3	-2.4
-2.2	-2.3	-2.4	-2.3
-2.4	-2.4	-2.3	0.0

$\gamma = 0.4$

0.0	-1.3	-1.6	-1.7
-1.3	-1.6	-1.6	-1.7
-1.6	-1.6	-1.7	-1.6
-1.7	-1.7	-1.6	10

$\gamma = 0.2$

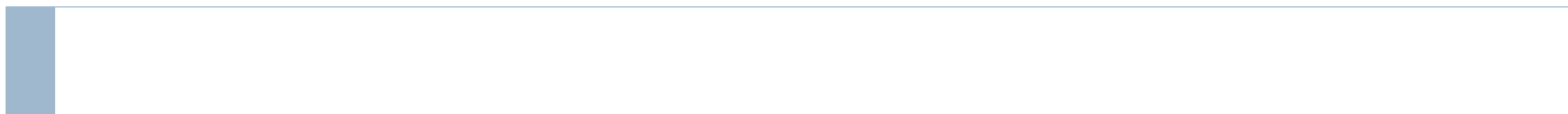
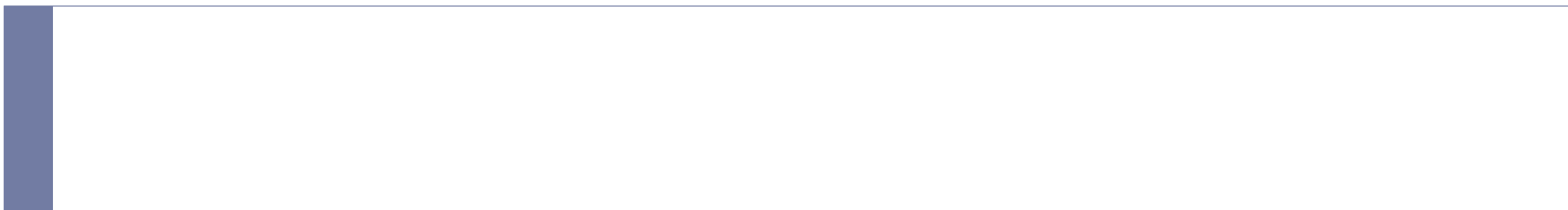
0.0	-1.1	-1.2	-1.2
-1.1	-1.2	-1.2	-1.2
-1.2	-1.2	-1.2	-1.2
-1.2	-1.2	-1.2	10

$\gamma = 0.0$

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

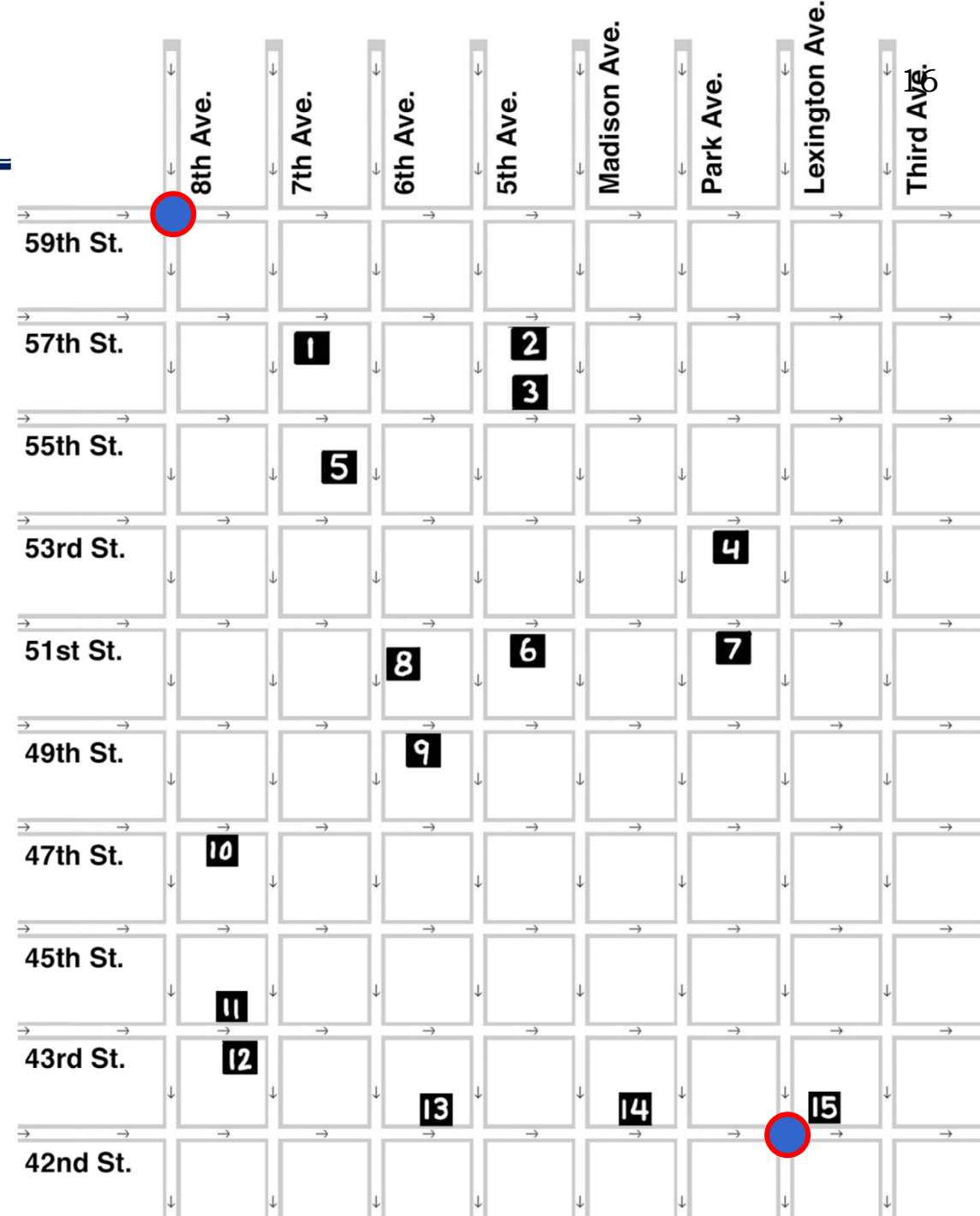
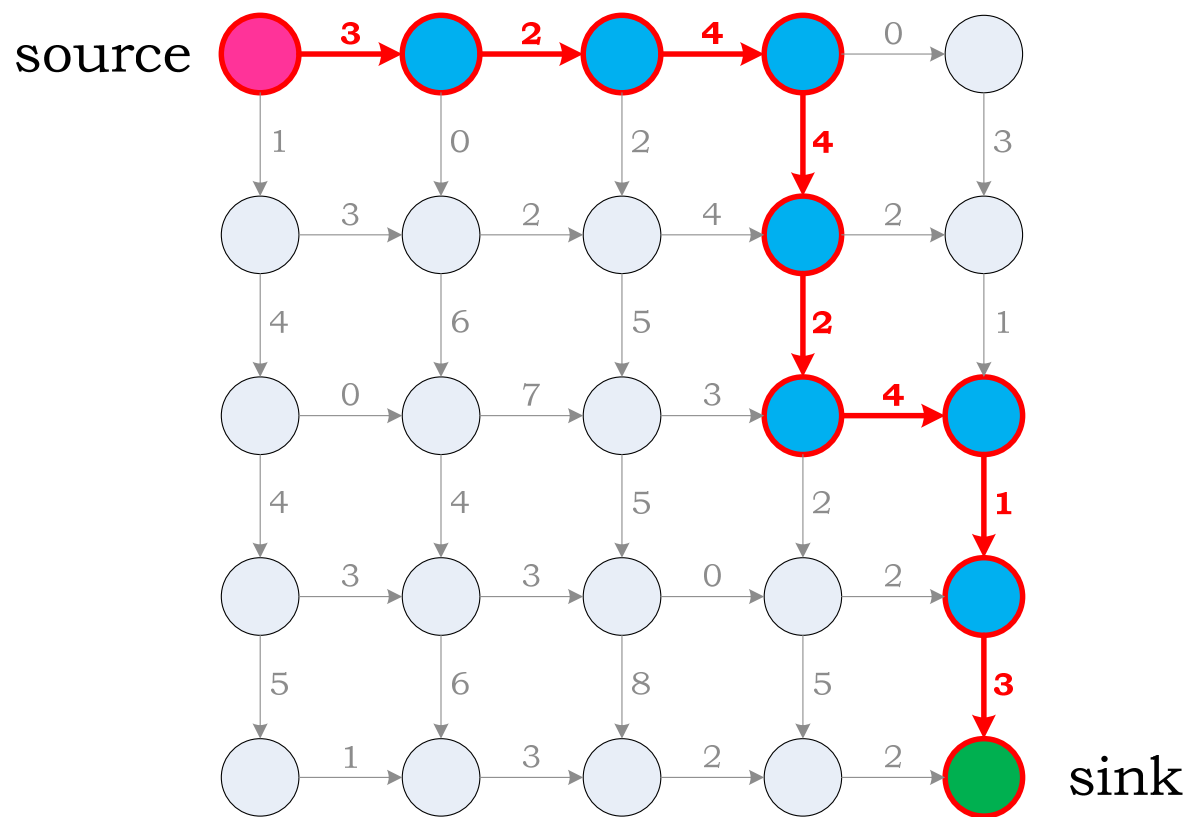
动态规划

Dynamic Programming



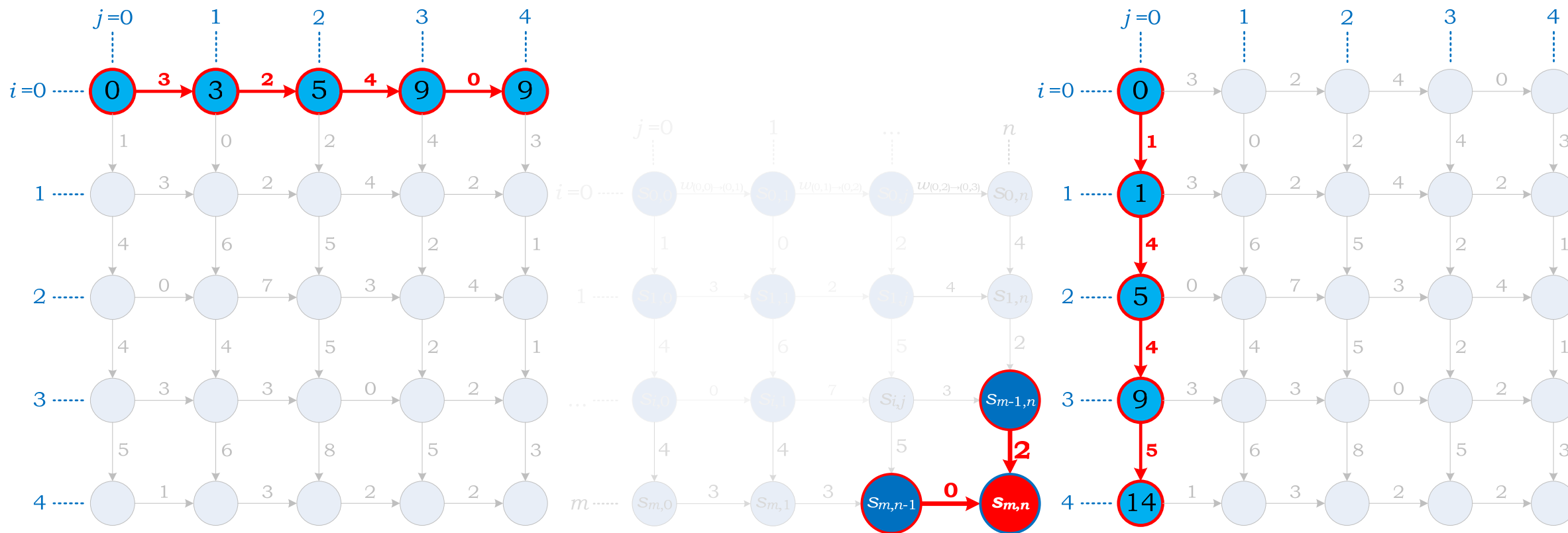
曼哈顿旅游者问题

- ▶ 最多能看多少个景点？
- ▶ 加权图最长路径问题



最长路径

$$LP(m, n) = \max \begin{cases} LP(m-1, n) + w_{(m-1, n) \rightarrow (m, n)} \\ LP(m, n-1) + w_{(m, n-1) \rightarrow (m, n)} \end{cases}$$



$$LP(0, n) = LP(0, n-1) + w_{(0, n-1) \rightarrow (0, n)}, \text{ if } n > 0$$

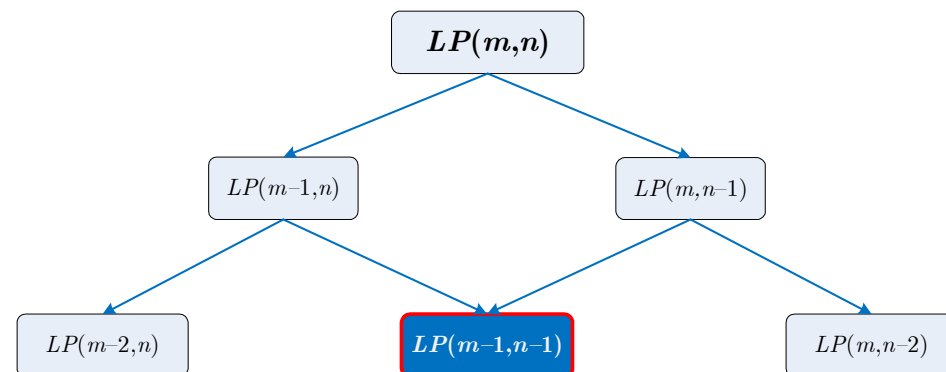
$$LP(m, 0) = LP(m-1, 0) + w_{(m-1, 0) \rightarrow (m, 0)}, \text{ if } m > 0$$

自顶向下递归求解

ALGORITHM *RecursiveLP*(G, m, n)
INPUT A weighted graph G and the sink
OUTPUT The longest path from the source to the sink

IF $m=0$ **AND** $n=0$
 RETURN 0
ELSE IF $m=0$
 RETURN $\text{RecursiveLP}(G, m, n-1) + w_{(m, n-1) \rightarrow (m, n)}$
ELSE IF $n=0$
 RETURN $\text{RecursiveLP}(G, m-1, n) + w_{(m-1, n) \rightarrow (m, n)}$
ELSE
 $L_1 = \text{RecursiveLP}(G, m-1, n) + w_{(m-1, n) \rightarrow (m, n)}$
 $L_2 = \text{RecursiveLP}(G, m, n-1) + w_{(m, n-1) \rightarrow (m, n)}$
 RETURN $\max(L_1, L_2)$

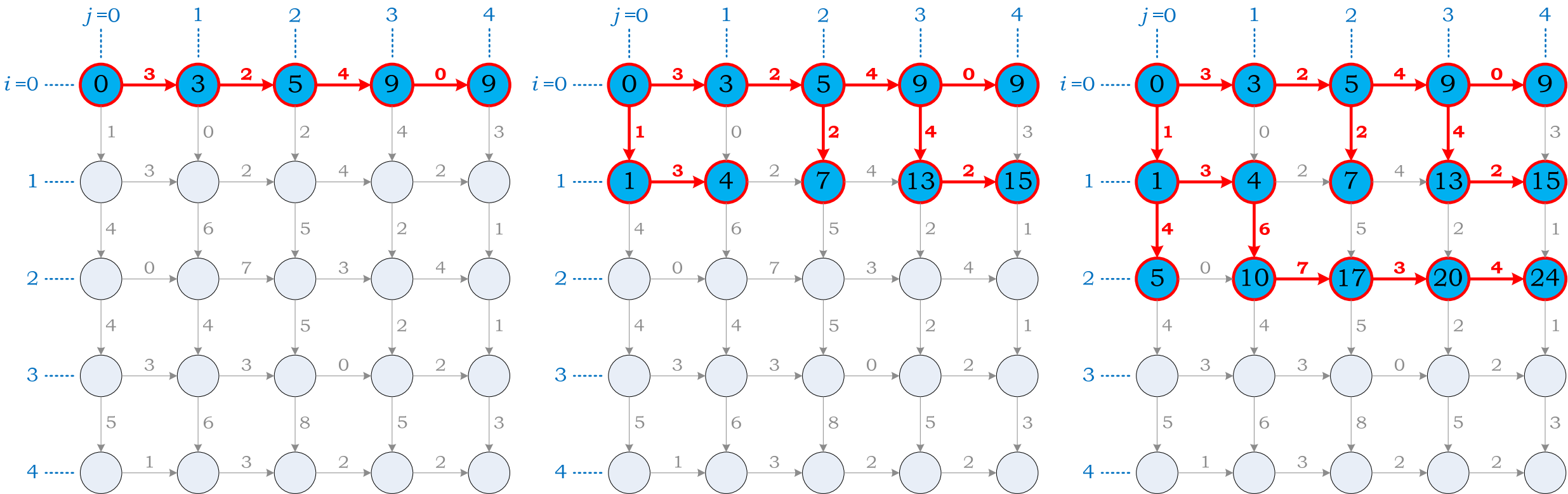
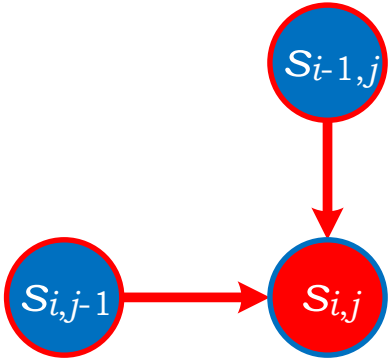
$$O\left(\frac{(m+n)!}{m!n!}\right)$$



重叠子问题的存在使得递归算法的时间复杂度是指数复杂度

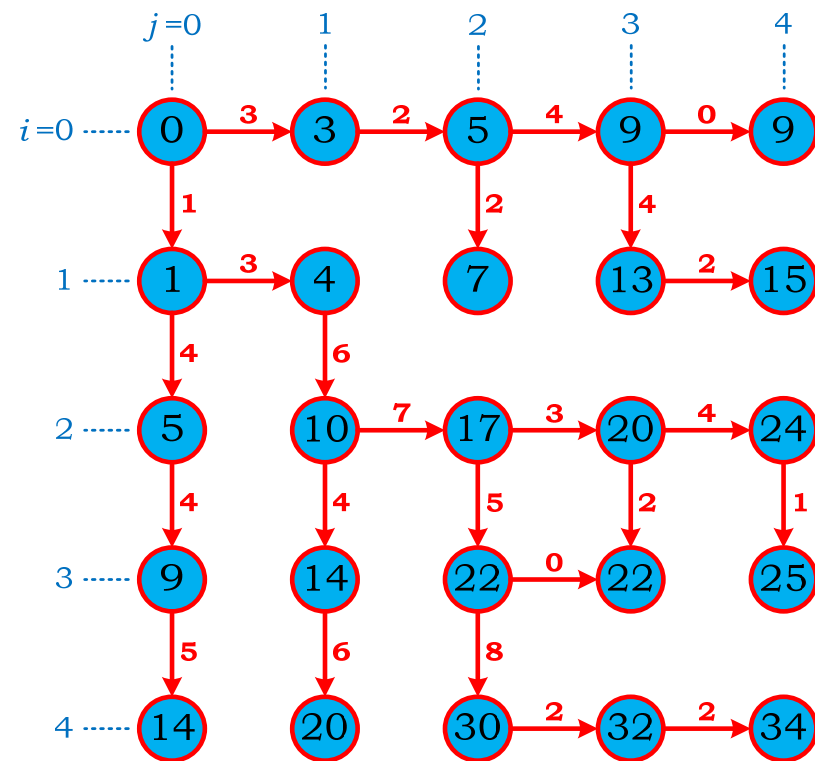
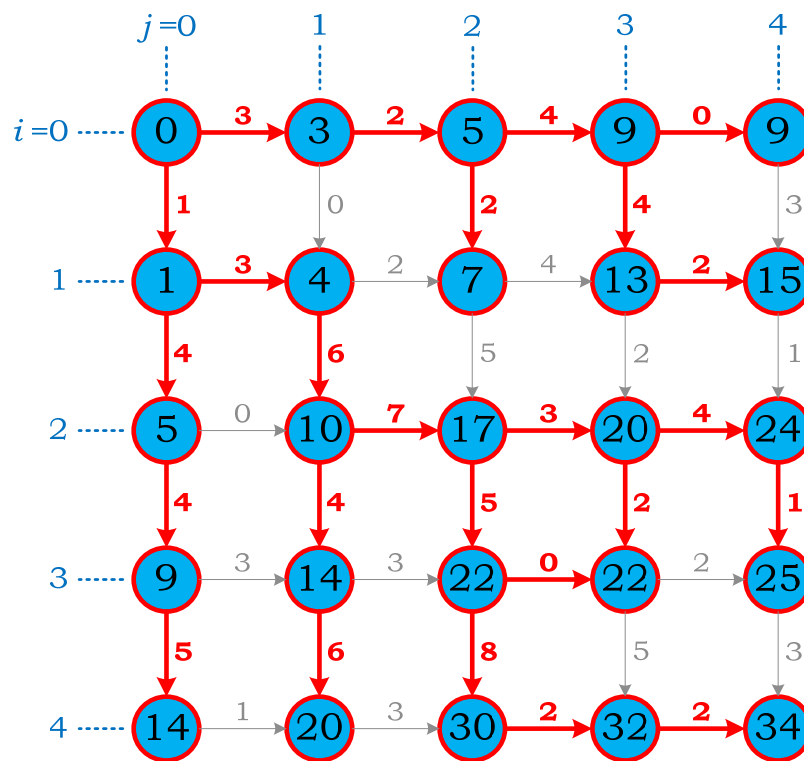
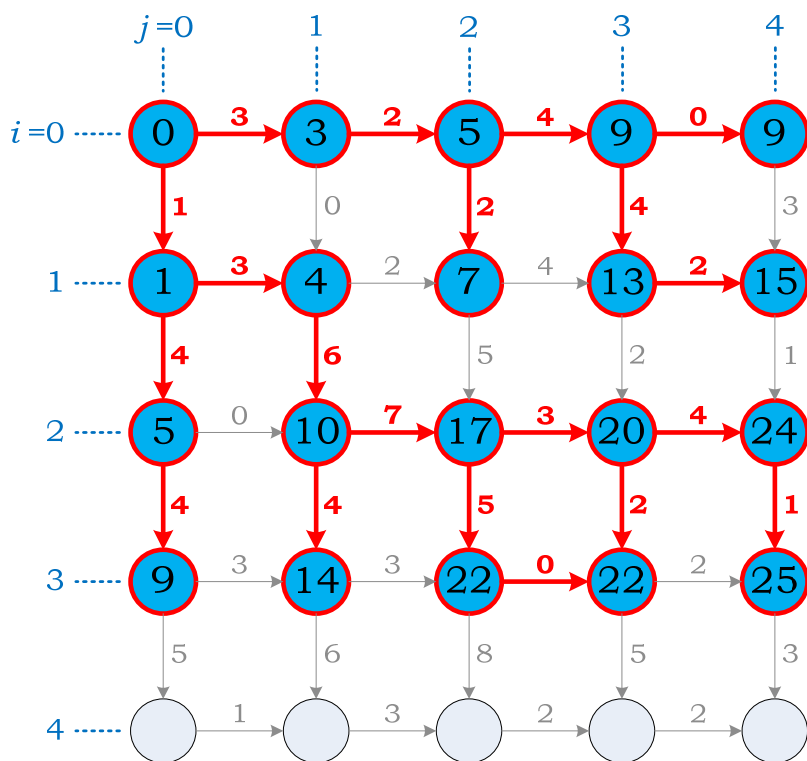
自底向上填表求解

$$s_{i,j} = \max(s_{i-1,j} + w_{(i-1,j) \rightarrow (i,j)}, s_{i,j-1} + w_{(i,j-1) \rightarrow (i,j)})$$



自底向上填表求解

$$s_{i,j} = \max(s_{i-1,j} + w_{(i-1,j) \rightarrow (i,j)}, s_{i,j-1} + w_{(i,j-1) \rightarrow (i,j)})$$



动态规划

Dynamic programming

ALGORITHM *DPLongestPath*(G, m, n)
INPUT A weighted graph G and the sink
OUTPUT The longest path from the source to the sink

```
FOR  $i \leftarrow 0$  TO  $m$  DO
  FOR  $j \leftarrow 0$  TO  $n$  DO
    IF  $i = 0$  AND  $j = 0$ 
       $L[i, j] \leftarrow 0$ 
    ELSE IF  $i = 0$ 
       $L[i, j] \leftarrow L[i, j-1] + w_{(i, j-1) \rightarrow (i, j)}$ 
    ELSE IF  $j = 0$ 
       $L[i, j] \leftarrow L[i-1, j] + w_{(i-1, j) \rightarrow (i, j)}$ 
    ELSE
       $L[i, j] \leftarrow \max($ 
         $L[i-1, j] + w_{(i-1, j) \rightarrow (i, j)},$ 
         $L[i, j-1] + w_{(i, j-1) \rightarrow (i, j)})$   $O(mn)$ 
```

- ▶ 动态规划解决
由递推关系定义
包含重叠子问题
的问题
- ▶ 建立将大问题分解为小问题的
递推关系式
- ▶ 使用表格法求解
 - ▶ 解决小问题，解填入表格
 - ▶ 从表格中抽取原问题的解

Richard Bellman

University of Southern California

1920~1984. He was a Fellow in the American Academy of Arts and Sciences (1975), a member of the National Academy of Engineering (1977), and a member of the National Academy of Sciences (1983).

He was awarded the IEEE Medal of Honor in 1979, "**for contributions to decision processes and control system theory, particularly the creation and application of dynamic programming**".



Richard Bellman

American electrical engineer and businessman who co-founded Qualcomm Inc. and invented the **Viterbi algorithm**. He is currently Presidential Chair Professor of Electrical Engineering at the University of Southern



Andrew Viterbi

California's Viterbi School of Engineering, which was named in his honor in 2004 in recognition of his \$52 million gift.

Member of the American Academy of Arts and Sciences (1995), National Academy of Sciences (2001), National Academy of Engineering (2012), **Chinese Academy of Sciences (2013)**, French Academy of Sciences (2005). He is a professor at the University of Southern California.



Mike Waterman

One of the founders and current leaders in the area of computational biology. His work has contributed to some of the most widely used tools in the field. In particular, the **Smith-Waterman algorithm**.

策略评价

Policy Evaluation

- ▶ 问题：计算在给定策略 π 下各状态的价值
- ▶ 算法：迭代应用状态价值贝尔曼期望方程
- ▶ 计算： $\mathbf{v}^{(k+1)} = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{v}^{(k)}$
- ▶ 过程：生成序列 $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_\pi$

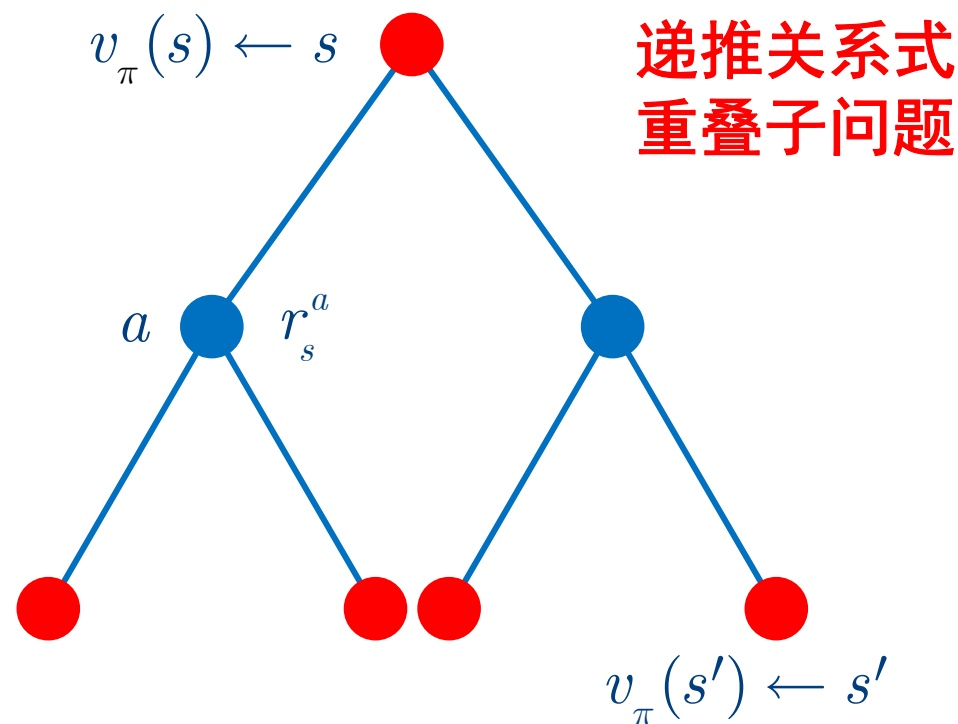
- ▶ 状态价值贝尔曼期望方程

$$v_\pi(s) = \sum_{a \in \mathbf{A}} \pi(a | s) \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_\pi(s') \right)$$

$$v_{k+1}(s) = \sum_{a \in \mathbf{A}} \pi(a | s) \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_k(s') \right)$$

$$v_{k+1}(s) = \underbrace{\sum_{a \in \mathbf{A}} \pi(a | s) r_s^a}_{r_s^\pi} + \gamma \sum_{s' \in \mathbf{S}} \underbrace{\sum_{a \in \mathbf{A}} \pi(a | s) p_{ss'}^a}_{p_{ss'}^\pi} v_k(s')$$

$$v_{k+1}(s) = r_s^\pi + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^\pi v_k(s')$$



策略评价算法

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π , the policy to be evaluated

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation

Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

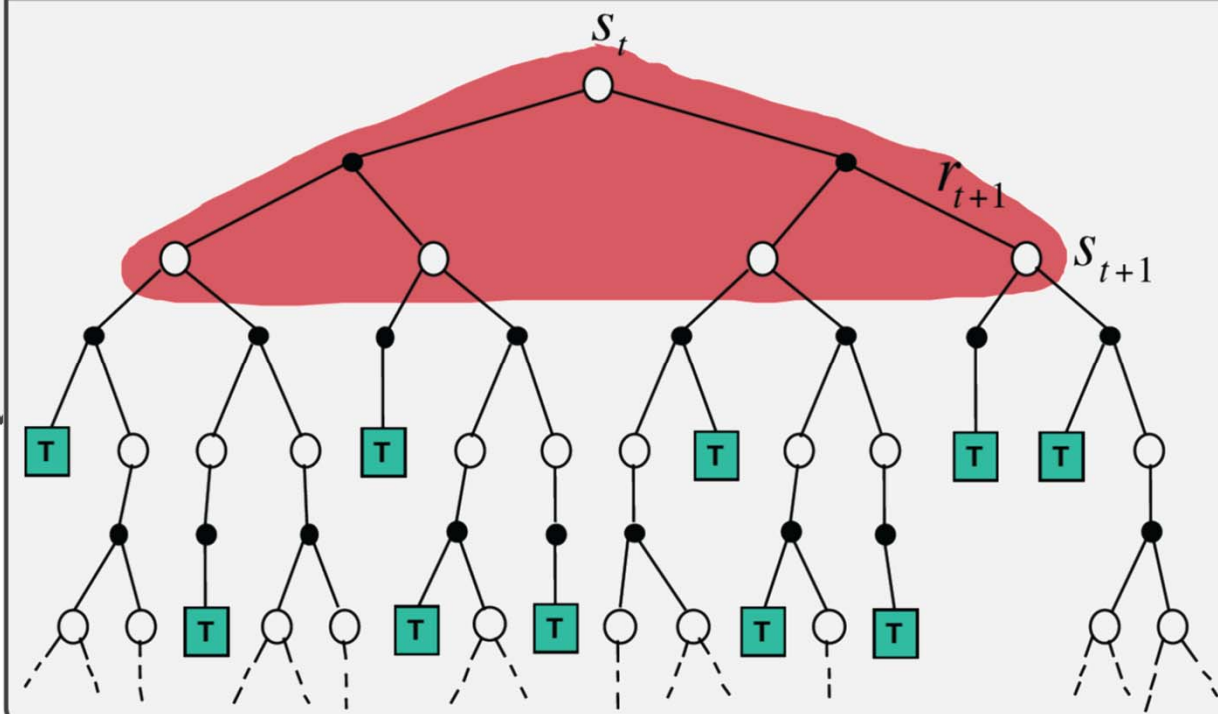
$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$

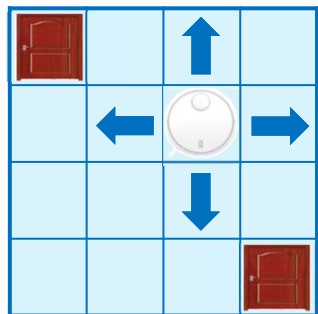
像宽度优先搜索

Iterative Policy Evaluation, for estimating $V \approx v_\pi$



动态规划策略评价

$$\mathbf{v}^{(k+1)} = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{v}^{(k)}$$



$\gamma = 1.0$

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

$k = 0$

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

$k = 1$

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

0	-1.75	-2	-2
-1.75	-2	-2	-2
-2	-2	-2	-1.75
-2	-2	-1.75	0

$k = 2$

0	-2.44	-2.94	-3
-2.44	-2.88	-3	-2.94
-2.94	-3	-2.88	-2.44
-3	-2.94	-2.44	0

$k = 3$

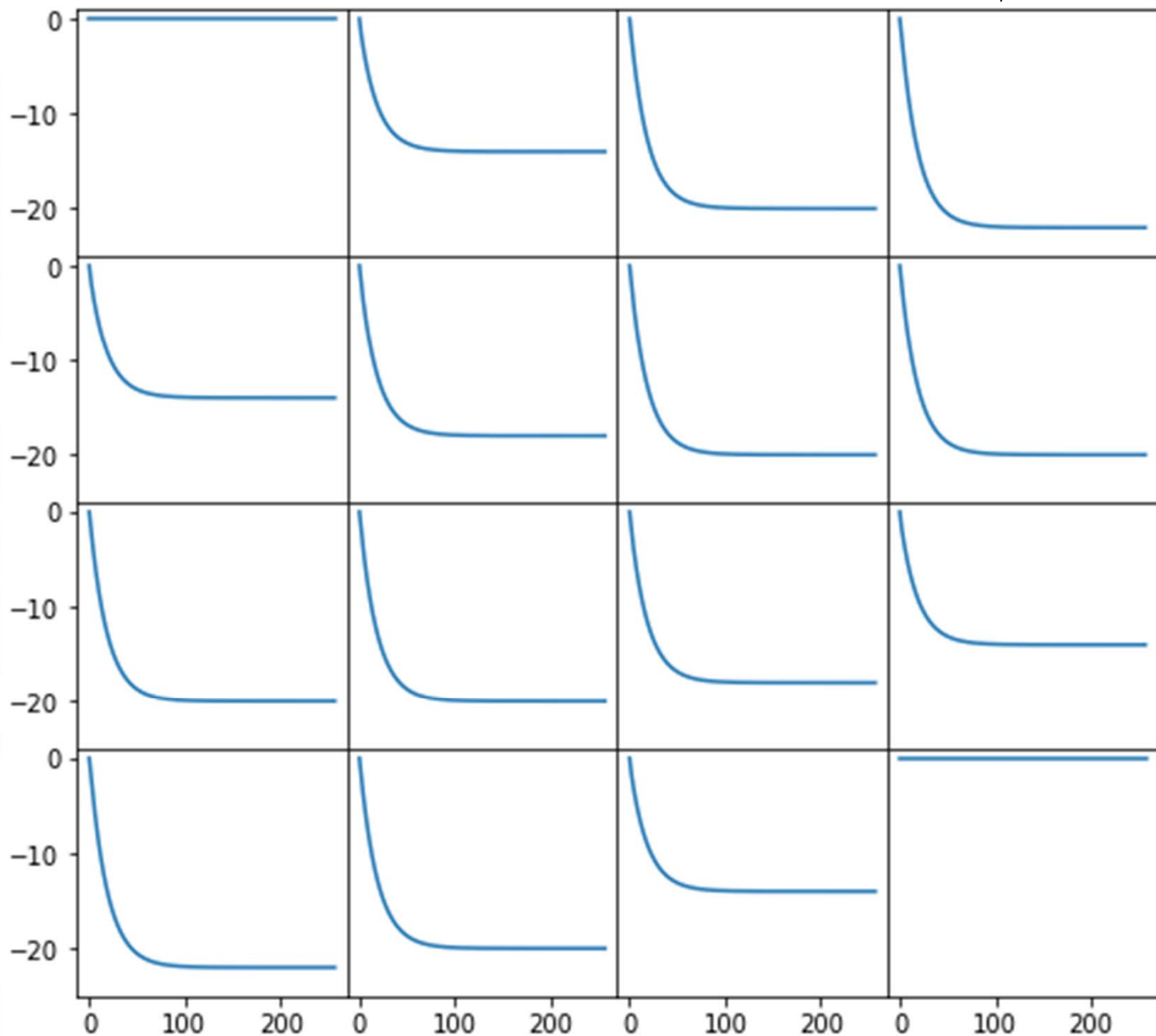
0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

0	-6.14	-8.35	-8.97
-6.14	-7.74	-8.43	-8.35
-8.35	-8.43	-7.74	-6.14
-8.97	-8.35	-6.14	0

$k = 10$

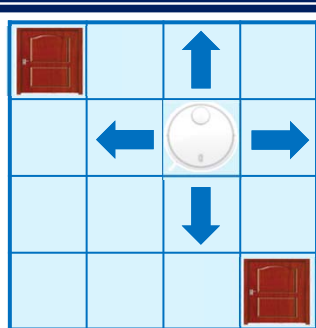
0	-13.9	-19.9	-21.9
-13.9	-17.9	-19.9	-19.9
-19.9	-19.9	-17.9	-13.9
-21.9	-19.9	-13.9	0

$k = 100$



对初值不敏感

$$\mathbf{v}^{(k+1)} = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{v}^{(k)}$$



$\gamma = 1.0$

0	-0.67	0.25	-0.93
0.39	1.53	-1.23	0.32
-1.50	-1.22	1.09	1.12
-1.10	1.06	-0.87	0

$k = 0$

0	-0.39	-1.31	-1.32
-0.9	-1.35	-0.20	-1.18
-1.86	-0.46	-1.55	-0.37
-1.66	-1.53	-0.68	0

$k = 1$

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

0	-1.76	-1.81	-2.28
-2.20	-1.49	-2.35	-1.77
-2.22	-2.57	-1.43	-1.77
-2.68	-2.08	-1.94	0

$k = 2$

0	-2.26	-3.05	-3.04
-2.43	-3.18	-2.62	-3.04
-3.37	-2.80	-3.16	-2.24
-3.41	-3.32	-2.36	0

$k = 5$

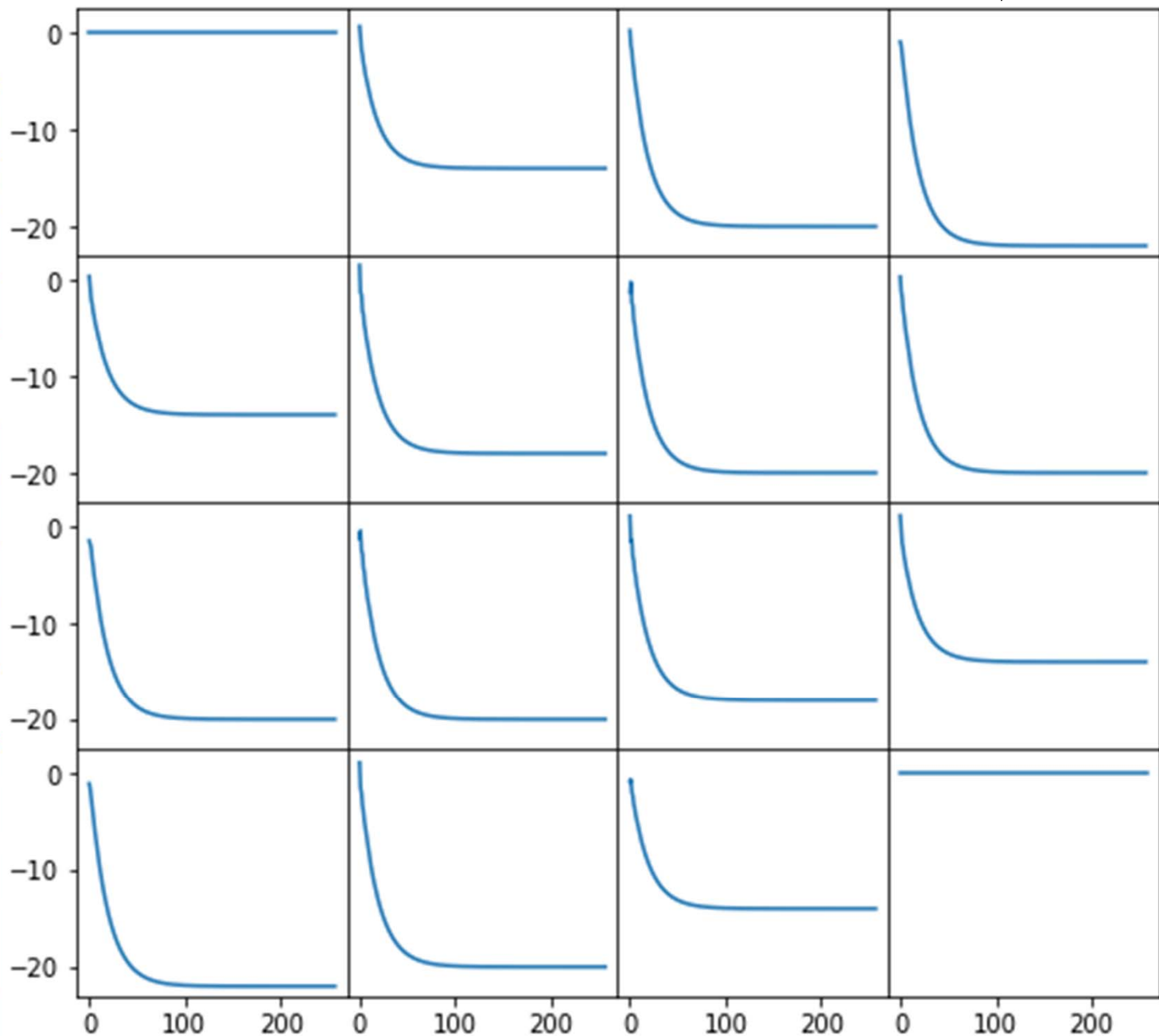
0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

0	-6.17	-8.36	-8.98
-6.21	-7.77	-8.49	-8.36
-8.46	-8.55	-7.77	-6.17
-9.12	-8.46	-6.21	0

$k = 10$

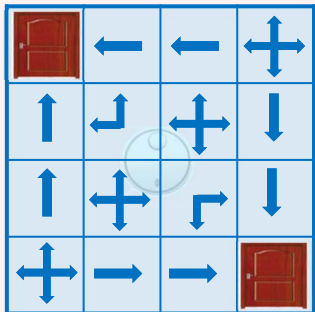
0	-13.9	-19.9	-21.9
-13.9	-17.9	-19.9	-19.9
-19.9	-19.9	-17.9	-13.9
-21.9	-19.9	-13.9	0

$k = 100$



中间策略

$$\mathbf{v}^{(k+1)} = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{v}^{(k)}$$



$\gamma = 1.0$

0	-0.67	0.25	-0.93
0.39	1.53	-1.23	0.32
-1.50	-1.22	1.09	1.12
-1.10	1.06	-0.87	0

$k = 0$

0	-1.00	-0.33	-1.32
-1.00	-0.47	0.20	0.12
-0.61	0.46	-0.88	-1.00
-1.66	-1.87	-1.00	0

$k = 1$

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

0	-1.00	-2.00	-1.71
-1.00	-2.00	-1.39	-2.00
-2.00	-1.96	-2.00	-1.00
-2.45	-2.00	-1.00	0

$k = 2$

0	-1	-2	-2.86
-1	-2	-3	-2
-2	-3	-2	-1
-3.22	-2	-1	0

$k = 3$

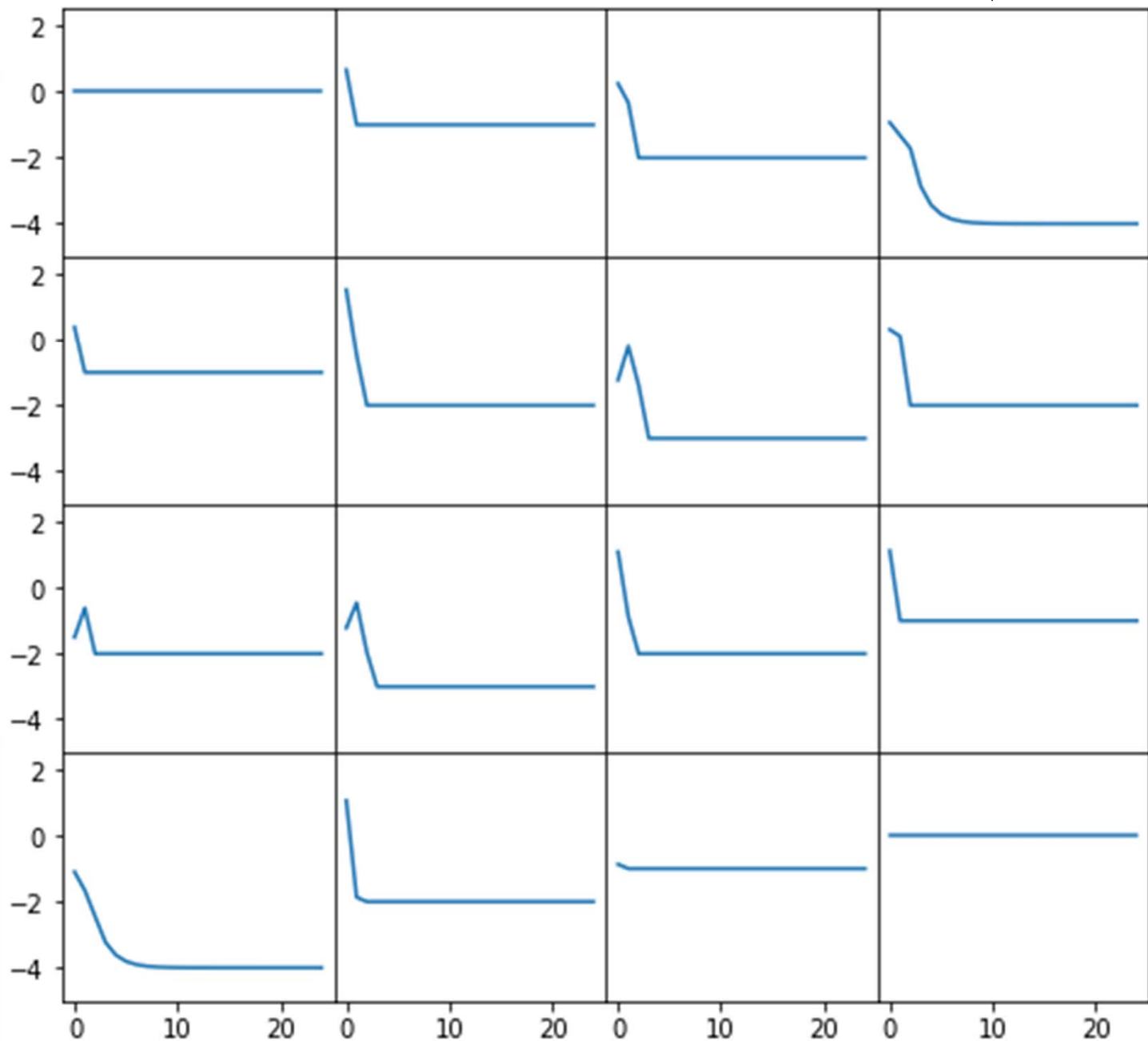
0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

0	-1	-2	-3.99
-1	-2	-3	-2
-2	-3	-2	-1
-4.00	-2	-1	0

$k = 10$

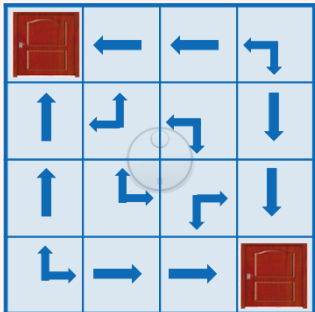
0	-1	-2	-4
-1	-2	-3	-2
-2	-3	-2	-1
-4	-2	-1	0

$k = 24$



最优策略

$$\mathbf{v}^{(k+1)} = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{v}^{(k)}$$



$\gamma = 1.0$

0	-0.67	0.25	-0.93
0.39	1.53	-1.23	0.32
-1.50	-1.22	1.09	1.12
-1.10	1.06	-0.87	0

$k = 0$

0	-1.00	-0.33	-0.72
-1.00	-0.47	0.31	0.12
-0.61	0.31	-0.88	-1.00
-1.22	-1.87	-1.00	0

$k = 1$

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

0	-1.00	-2.00	-1.10
-1.00	-2.00	-1.67	-2.00
-2.00	-1.67	-2.00	-1.00
-2.24	-2.00	-1.00	0

$k = 2$

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

$k = 3$

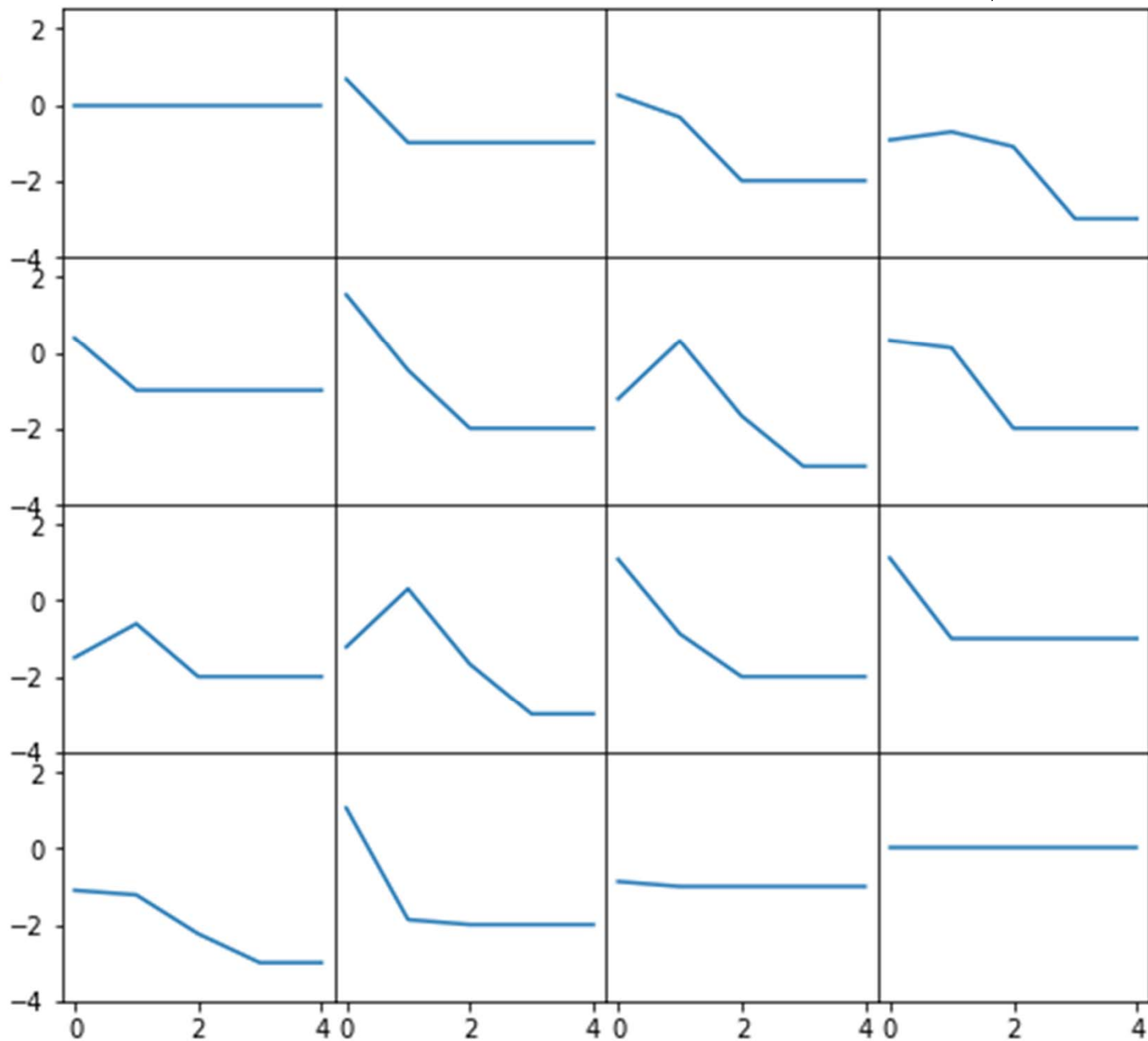
0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

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

$k = 4$

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

$k = \infty$

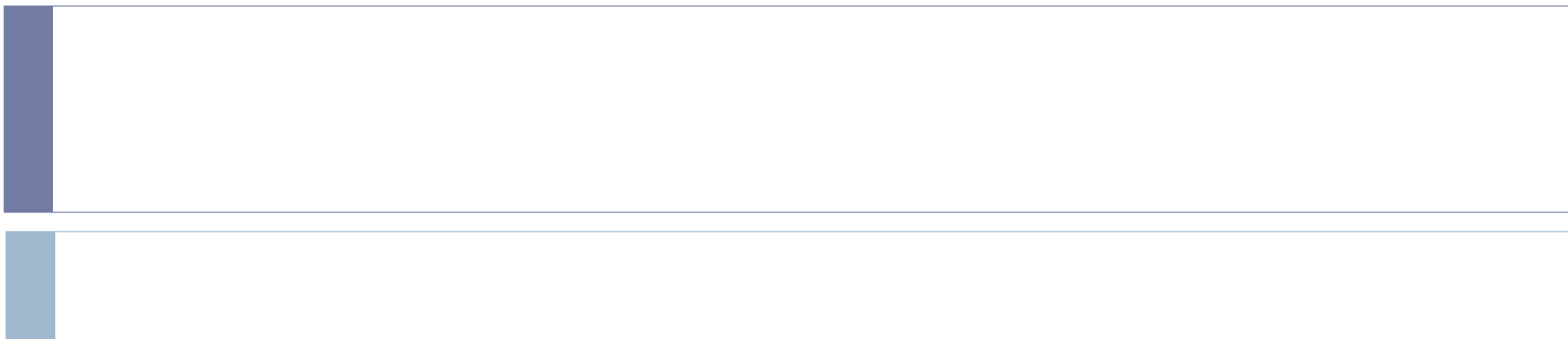


预测问题

- ▶ 计算价值函数，评价给定策略
 - ▶ 基于价值函数
 - ▶ **基于状态转移模型 (Model-based)**
 - ▶ **线性方程**
 - ▶ **动态规划**
 - ▶ 不基于状态转移模型 (Model-free)
 - ▶ 蒙特卡洛预测
 - ▶ 时序差分预测
 - ▶ 近似方法
 - ▶ 线性回归
 - ▶ 深度学习
 - ▶ 基于策略模型
- ▶ 求解线性方程组和动态规划迭代改进是**基于状态转移模型计算价值函数的方法**
 - ▶ 因为策略的评价是基于价值函数进行的计算价值函数的值也就是求解预测问题
 - ▶ 求解线性方程组适用于规模较小的问题动态规划迭代则适用于规模较大的问题

策略迭代

Policy Iteration



最优策略

控制问题

Optimal Policy

- 策略就是在特定状态时采取什么行动

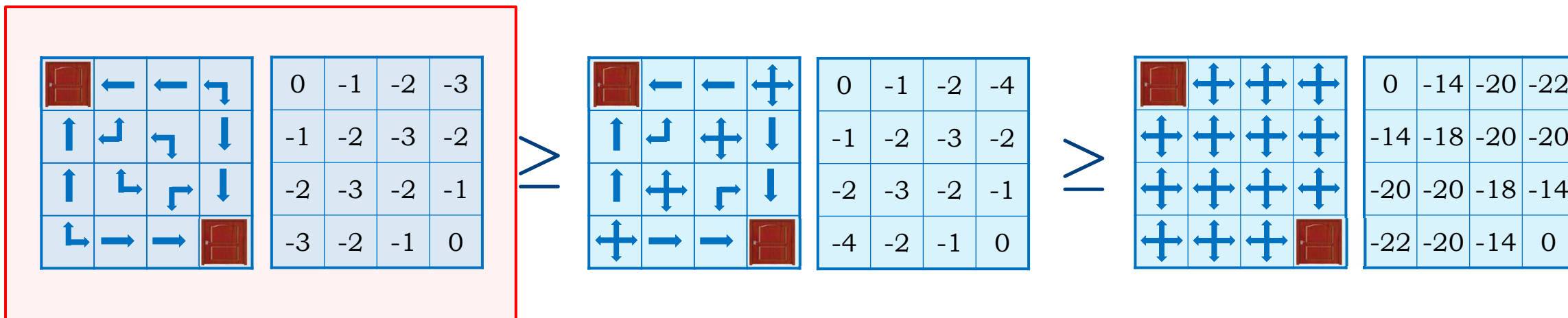
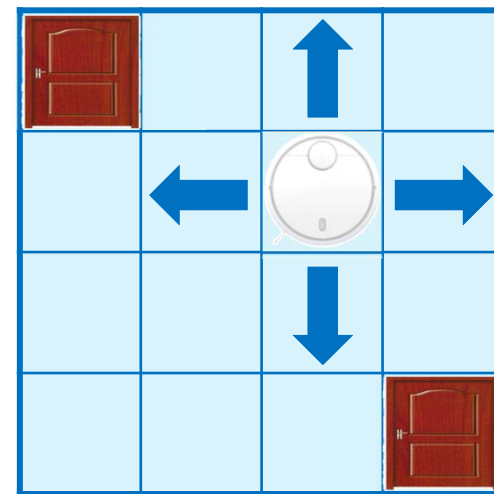
$$\pi(a | s) = p(A_t = a | S_t = s)$$

- 策略的好坏用状态价值来评价（预测状态价值的原因）

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

- 最优策略

$$\pi_* \geq \pi' : v_{\pi_*}(s) \geq v_{\pi'}(s), \forall s, \forall \pi'$$



最优状态价值

Optimal state value

状态价值函数

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

最优状态价值

$$v_*(s) = \max_{\pi} v_{\pi}(s)$$

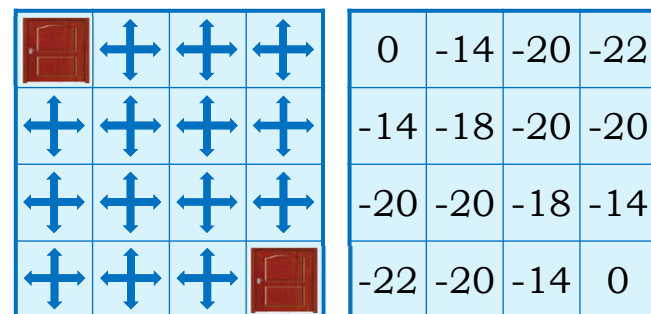
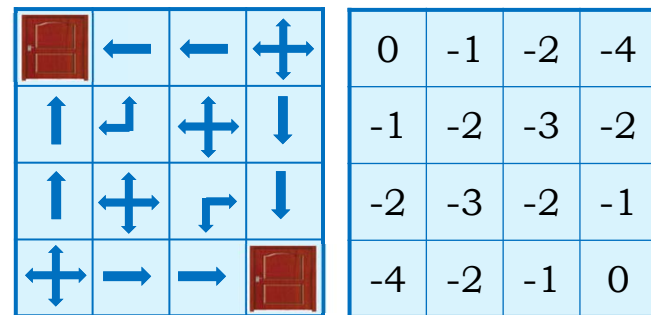
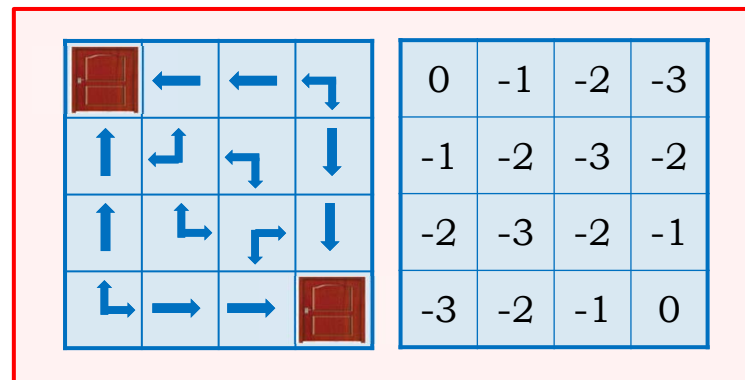
在最优策略下状态价值取得最大值

$$\pi_* \geq \pi' : v_{\pi_*}(s) \geq v_{\pi'}(s), \forall s, \forall \pi'$$

$$v_{\pi_*}(s) = \max_{\pi} v_{\pi}(s)$$

$$v_*(s) = v_{\pi_*}(s)$$

最优状态价值即最优策略下的状态价值



最优行动价值

Optimal action value

行动价值函数

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

最优行动价值

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$



行动价值由后续状态价值的期望计算



$$q_{\pi}(s, a) = r_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{ss'}^a v_{\pi}(s')$$

$$q_{\pi_*}(s, a) = \max_{\pi} q_{\pi}(s, a)$$

$$q_*(s, a) = q_{\pi_*}(s, a)$$

最优行动价值即最优策略下的行动价值

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

	←	←	↙
↑	↖	↙	↓
↑	↖	↘	↓
↙	→	→	

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

↑	-2	-3	-4	-1	-2	-3	-4	-2	-3	-4	-3	-3	-4	-3
→	-3	-4	-4	-3	-4	-3	-3	-4	-3	-2	-2	-3	-2	-1
↓	-3	-4	-3	-3	-4	-3	-2	-4	-3	-2	-1	-4	-3	-2
←	-1	-2	-3	-2	-2	-3	-4	-3	-3	-4	-3	-4	-4	-3
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

最优价值

- ▶ 状态价值函数

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

- ▶ 最优状态价值

$$v_*(s) = \max_{\pi} v_{\pi}(s)$$

- ▶ 在最优策略下，状态价值取得最大值

$$\pi_* \geq \pi' : v_{\pi_*}(s) \geq v_{\pi'}(s), \forall s, \forall \pi'$$

$$v_{\pi_*}(s) = \max_{\pi} v_{\pi}(s)$$

$$v_*(s) = v_{\pi_*}(s)$$

- ▶ 最优状态价值即最优策略下的状态价值

Optimal value functions

- ▶ 行动价值函数

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

- ▶ 最优行动价值

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

- ▶ 行动价值是后续状态价值的期望

$$q_{\pi}(s, a) = r_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{ss'}^a v_{\pi}(s')$$

$$q_{\pi_*}(s, a) = \max_{\pi} q_{\pi}(s, a)$$

$$q_*(s, a) = q_{\pi_*}(s, a)$$

- ▶ 最优行动价值即最优策略下的行动价值

最优策略的性质

▶ 最优状态价值

$$v_*(s) = \max_{\pi} v_{\pi}(s)$$

▶ 状态价值是同时刻行动价值的期望

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

▶ 在最优策略下

$$v_*(s) = \sum_{a \in \mathbf{A}} \pi_*(a | s) q_*(s, a)$$

▶ 求解优化问题

$$\begin{aligned} \max_{\pi_*} \quad & \sum_{a \in \mathbf{A}} \pi_*(a | s) q_*(s, a) \\ \text{s.t.} \quad & \sum_{a \in \mathbf{A}} \pi_*(a | s) = 1 \\ & \pi_*(a | s) \geq 0 \end{aligned}$$

▶ 最优行动价值

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

▶ 线性规划问题，解为确定性贪心策略

$$\pi_*(a | s) = \begin{cases} 1 & \text{if } a = \arg \max_{a \in \mathbf{A}} q_*(s, a) \\ 0 & \text{otherwise} \end{cases}$$

▶ 最优状态价值即同时刻最优行动价值

$$v_*(s) = \max_{a \in \mathbf{A}} q_*(s, a)$$

0	-1	-2	-3	-2	-3	-4	-1	-2	-3	-4	-2	-3	-4	-3	-3	-4	-3
-1	-2	-3	-2	-3	-4	-4	-3	-4	-3	-3	-4	-3	-2	-3	-2	-1	-1
-2	-3	-2	-1	-3	-4	-3	-3	-4	-3	-2	-4	-3	-2	-1	-4	-3	-2
-3	-2	-1	0	-1	-2	-3	-2	-2	-3	-4	-3	-3	-4	-3	-4	-4	-3
1	2	3	4	5	6	7	8	9	10	11	12	13	14				

↑

→

↓

←

策略改进

- ▶ 最优状态价值是同时刻最优行动价值

$$v_*(s) = \max_{a \in \mathbf{A}} q_*(s, a)$$

- ▶ 但是最优策略、最优状态价值、最优行动价值均未知
- ▶ 已知的是某个策略 π ，用动态规划计算出的状态价值 $v_\pi(s)$ ，行动价值 $q_\pi(s, a)$

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

$$q_\pi(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_\pi(s')$$

Policy Improvement

- ▶ 如果对某一状态 s ，仿照最优化方程改进一下策略，而保持其他状态的策略不变

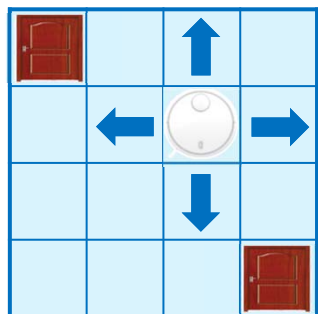
- ▶ 例如，改为贪心策略

$$\pi'(a | s) = \begin{cases} 1 & \text{if } a = \arg \max_{a \in \mathbf{A}} q_\pi(s, a) \\ 0 & \text{otherwise} \end{cases}$$

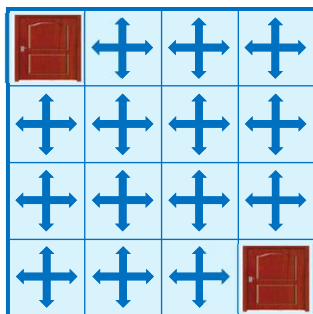
- ▶ 该状态的价值会发生什么变化？

$$v_\pi(s) \begin{matrix} = \\ < \\ \leq \\ > \\ \geq \end{matrix} v_{\pi'}(s)$$

策略改进试验



$\gamma = 1.0$



0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0

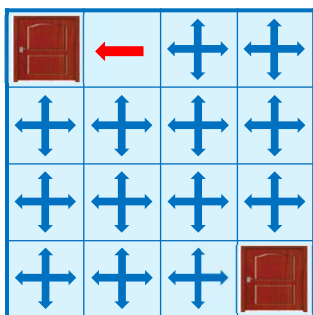
$k = 200$

-15	-21	-23	-1	-15	-21	-23	-15	-19	-21	-21	-21	-21	-19
-21	-23	-23	-19	-21	-21	-21	-21	-19	-15	-15	-21	-15	-1
-19	-21	-21	-21	-21	-19	-15	-23	-21	-15	-1	-23	-21	-15
-1	-15	-21	-15	-15	-19	-21	-21	-21	-21	-19	-23	-23	-21

1 2 3 4 5 6 7 8 9 10 11 12 13 14

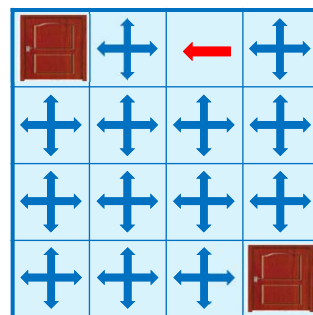


0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15



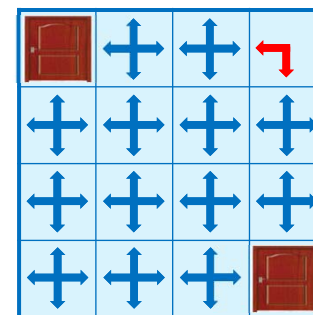
0	-1	-11.0	-14.6
-10.2	-10.9	-13.4	-14.2
-15.7	-15.1	-13.6	-10.6
-17.9	-16.1	-11.2	0

$k = 187$



0	-9.66	-10.7	-14.8
-11.9	-14.3	-14.7	-14.9
-17.5	-17.0	-14.8	-11.2
-19.5	-17.6	-12.1	0

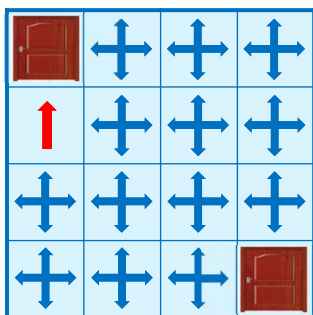
$k = 200$



0	-13.4	-18.7	-19.7
-13.6	-17.4	-19.0	-18.7
-19.6	-19.5	-17.4	-13.4
-21.6	-19.6	-13.6	0

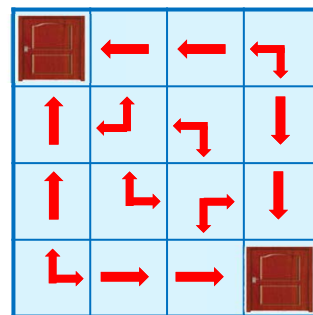
$k = 200$

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



0	-10.2	-15.7	-17.9
-1.00	-10.9	-15.1	-16.1
-11.0	-13.4	-13.6	-11.2
-14.6	-14.2	-10.6	0

$k = 187$



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

$k = 4$

策略改进能够
提升状态价值

策略改进提升状态价值

- 基于原来策略 π , 产生新策略 π'

$$\pi'(a | s) = \begin{cases} 1 & \text{if } a = \arg \max_{a \in \mathbf{A}} q_{\pi}(s, a) \\ 0 & \text{otherwise} \end{cases}$$

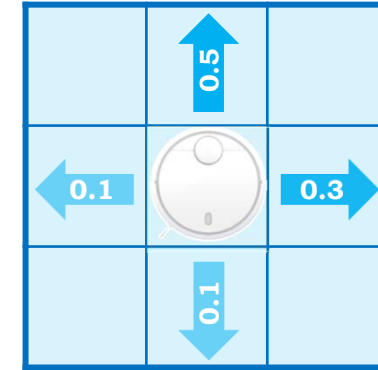
→ $\pi'(s) = \arg \max_{a \in \mathbf{A}} q_{\pi}(s, a)$

→ $q_{\pi}(s, \pi'(s)) = \max_{a \in \mathbf{A}} q_{\pi}(s, a)$

→
$$\begin{aligned} v_{\pi}(s) &= \sum_{a \in \mathbf{A}} \pi(a | s) q_{\pi}(s, a) \\ &\leq \sum_{a \in \mathbf{A}} \pi(a | s) q_{\pi}(s, \pi'(s)) \\ &= q_{\pi}(s, \pi'(s)) \sum_{a \in \mathbf{A}} \pi(a | s) \\ v_{\pi}(s) &\leq q_{\pi}(s, \pi'(s)) \end{aligned}$$

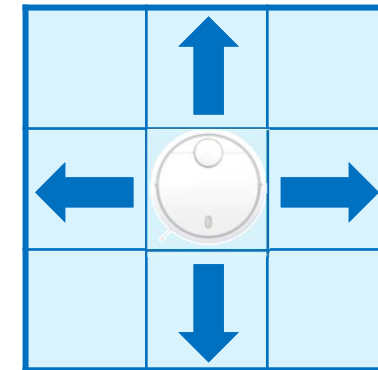
- 随机性策略

$$\pi(a | s)$$



- 确定性策略

$$\pi(s) = a$$



策略改进定理

- ▶ 基于原策略 π , 产生新策略 π'
- ▶ 如果 $v_{\pi}(s) \leq q_{\pi}(s, \pi'(s))$
- ▶ 那么 $v_{\pi}(s) \leq v_{\pi'}(s)$

Chapter 4.2
p76~80

Reinforcement
Learning

An Introduction
second edition

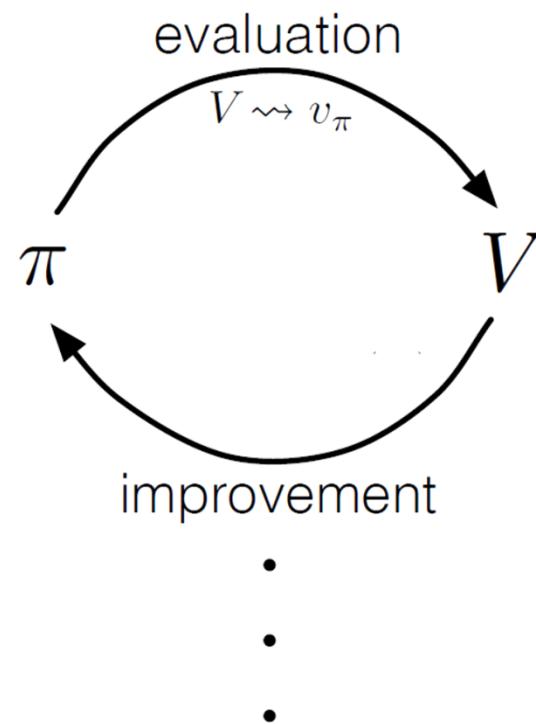
Richard S. Sutton and Andrew G. Barto



http://rl.qiwihiui.com/zh_CN/latest/

策略迭代思想

- ▶ 交替进行策略评价和策略改进



- ▶ 那么状态价值会持续提升

$$v_1(s) \leq v_2(s) \leq \dots \leq v_t(s) \leq \dots$$

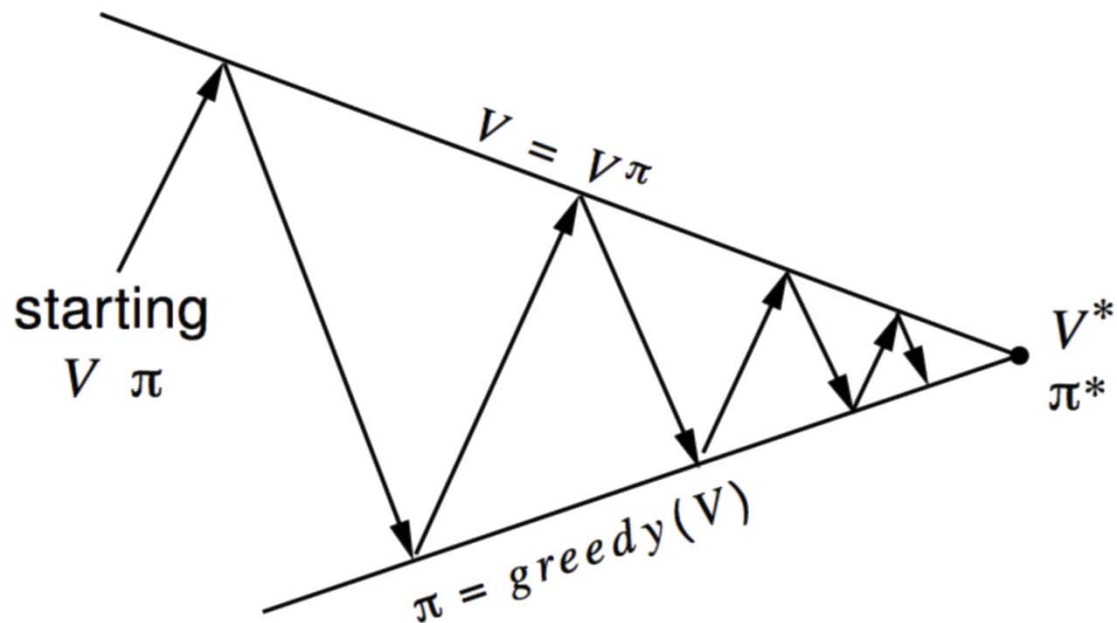
- 改进到最后

$$v_{\pi}(s) = q_{\pi}(s, \pi'(s)) = \max_{a \in \mathbf{A}} q_{\pi}(s, a)$$

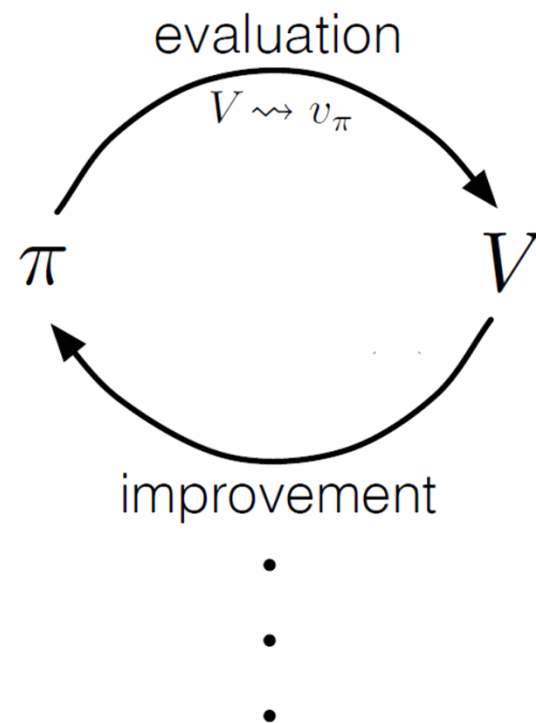
- 满足最优策略的性质

$$v_{*}(s) = \max_{a \in \mathbf{A}} q_{*}(s, a)$$

- 此时，策略 π 为最优策略



- 交替进行策略评价和策略改进



- 那么状态价值会持续提升

$$v_1(s) \leq v_2(s) \leq \dots \leq v_t(s) \leq \dots$$

策略迭代

Policy iteration

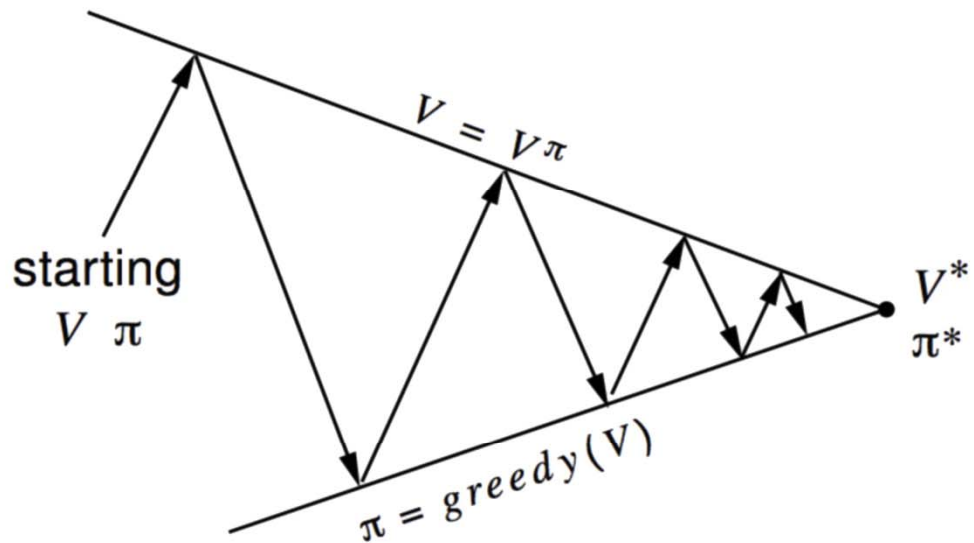
- 交替进行策略评价和策略改进

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$



策略评价 策略改进

- 直至收敛（策略不再变化）



Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s)) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement

policy-stable \leftarrow true

For each $s \in \mathcal{S}$:

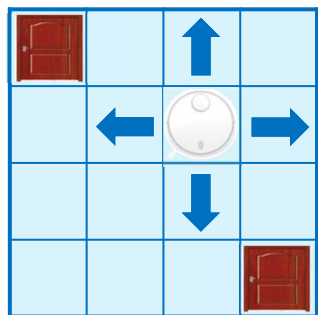
old-action $\leftarrow \pi(s)$

$\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$

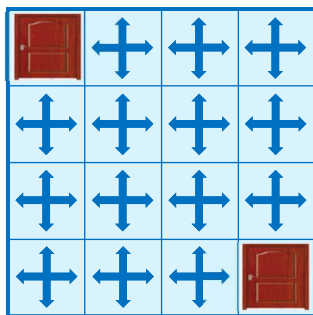
If *old-action* $\neq \pi(s)$, then *policy-stable* \leftarrow false

If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

策略迭代过程



$\gamma = 1.0$



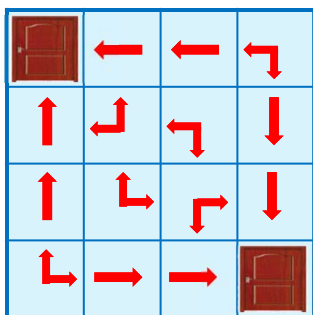
0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0

$k = 200$

-15	-21	-23	-1	-15	-21	-23	-15	-19	-21	-21	-21	-21	-19
-21	-23	-23	-19	-21	-21	-21	-21	-19	-15	-15	-21	-15	-1
-19	-21	-21	-21	-21	-19	-15	-23	-21	-15	-1	-23	-21	-15
-1	-15	-21	-15	-15	-19	-21	-21	-21	-21	-19	-23	-23	-21

1 2 3 4 5 6 7 8 9 10 11 12 13 14

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15



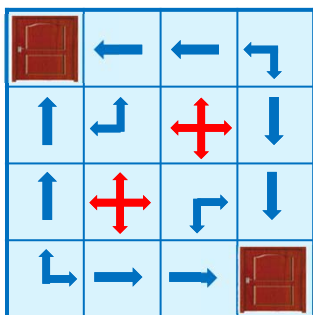
0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0

$k = 4$

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

1 2 3 4 5 6 7 8 9 10 11 12 13 14

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



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

$k = 4$

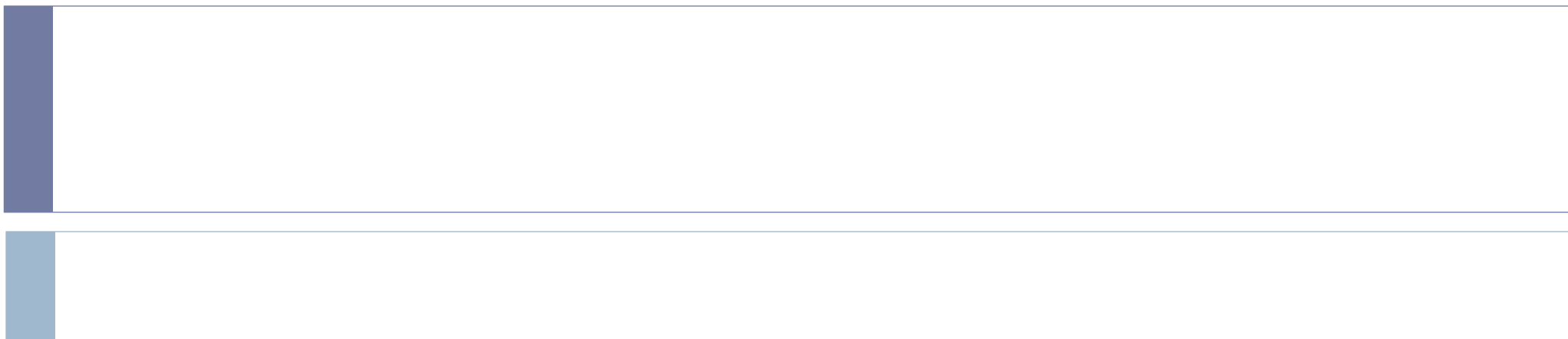
-2	-3	-4	-1	-2	-3	-4	-2	-3	-4	-3	-3	-4	-3
-3	-4	-4	-3	-4	-3	-3	-4	-3	-2	-2	-3	-2	-1
-3	-4	-3	-3	-4	-3	-2	-4	-3	-2	-1	-4	-3	-2
-1	-2	-3	-2	-2	-3	-4	-3	-3	-4	-3	-4	-4	-3

1 2 3 4 5 6 7 8 9 10 11 12 13 14

- ▶ 策略迭代是基于状态转移模型优化价值的方法，也就是获得最优策略、求解控制问题的方法
- ▶ 策略迭代是策略评价和策略改进两个步骤交替进行的过程
- ▶ 策略改进的原理是基于策略评价的结果使用贪心算法获得更好的策略，即状态价值在改进后会提升
- ▶ 优化价值函数，获得最优策略
- ▶ 基于价值函数
 - ▶ 基于状态转移模型
 - ▶ 策略迭代
 - ▶ 价值迭代
 - ▶ 不基于状态转移模型
 - ▶ 蒙特卡洛控制
 - ▶ 时序差分控制：SARSA, Q-learning
 - ▶ 近似方法
 - ▶ 线性回归
 - ▶ 深度学习：DQN
- ▶ 基于策略模型

价值迭代

Value Iteration



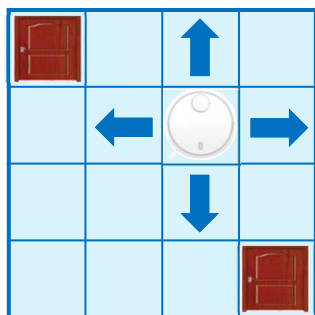
策略迭代

交替进行策略评价和策略改进

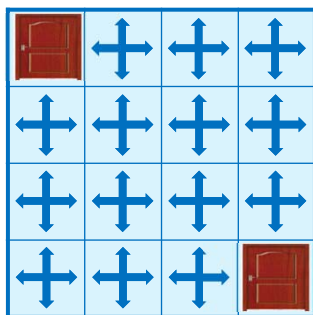
$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

策略改进

策略评价计算量很大



$\gamma = 1.0$



0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0

$k = 200$

-15	-21	-23	-1	-15	-21	-23	-15	-19	-21	-21	-21	-21	-19
-21	-23	-23	-19	-21	-21	-21	-21	-19	-15	-21	-15	-1	
-19	-21	-21	-21	-21	-19	-15	-23	-21	-15	-1	-23	-21	-15
-1	-15	-21	-15	-15	-19	-21	-21	-21	-21	-19	-23	-23	-21
1	2	3	4	5	6	7	8	9	10	11	12	13	14

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s)) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement

$policy_stable \leftarrow true$

For each $s \in \mathcal{S}$:

$old_action \leftarrow \pi(s)$

$\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$

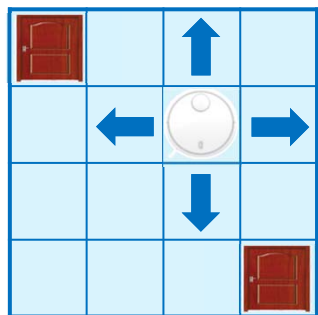
If $old_action \neq \pi(s)$, then $policy_stable \leftarrow false$

If $policy_stable$, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

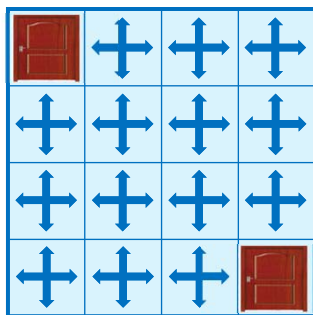
真的有必要迭代到收敛吗？

迭代多少次就停止合适呢？

每次策略评价后立即进行策略改进



$\gamma = 1.0$



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

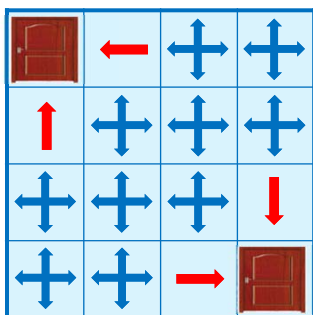
$k = 1$

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

1 2 3 4 5 6 7 8 9 10 11 12 13 14



0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15



0	-1.75	-2.00	-2.00
-1.75	-2.00	-2.00	-2.00
-2.00	-2.00	-2.00	-1.75
-2.00	-2.00	-1.75	0

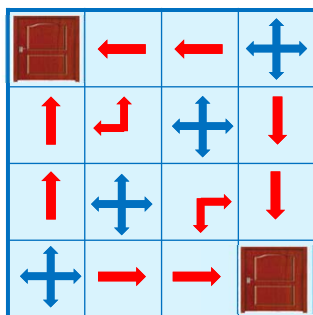
$k = 2$

-2	-2	-2	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-1
-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-1	-2	-2	-2
-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2

1 2 3 4 5 6 7 8 9 10 11 12 13 14



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



0	-2.44	-2.94	-3.00
-2.44	-2.88	-3.00	-2.94
-2.94	-3.00	-2.88	-2.44
-3.00	-2.94	-2.44	0

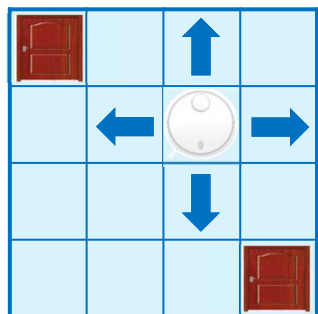
$k = 3$

-2.75	-3.00	-3.00	-1.00	-2.75	-3.00	-3.00	-2.75	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00
-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-2.75	-2.75	-3.00	-2.75	-1.00
-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-2.75	-3.00	-3.00	-2.75	-1.00	-3.00	-3.00	-2.75	-3.00
-1.00	-2.75	-3.00	-2.75	-2.75	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00

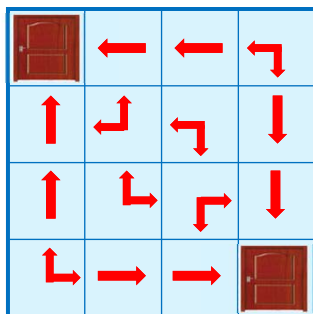
1 2 3 4 5 6 7 8 9 10 11 12 13 14



每次策略评价后立即进行策略改进



$\gamma = 1.0$



0	-3.06	-3.84	-3.97
-3.06	-3.72	-3.91	-3.84
-3.84	-3.91	-3.72	-3.06
-3.97	-3.84	-3.06	0

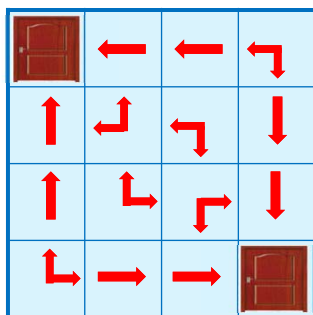
$k = 4$

-3.44	-3.94	-4.00	-1.00	-3.44	-3.94	-4.00	-3.44	-3.88	-4.00	-3.94	-3.94	-4.00	-3.88
-3.94	-4.00	-4.00	-3.88	-4.00	-3.94	-3.94	-4.00	-3.88	-3.44	-3.44	-3.94	-3.44	-1.00
-3.88	-4.00	-3.94	-3.94	-4.00	-3.88	-3.44	-4.00	-3.94	-3.94	-1.00	-4.00	-3.94	-3.44
-1.00	-3.44	-3.94	-3.44	-3.44	-3.88	-4.00	-3.94	-3.94	-4.00	-3.88	-4.00	-4.00	-3.94

1 2 3 4 5 6 7 8 9 10 11 12 13 14



0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15



0	-6.14	-8.35	-8.97
-6.14	-7.74	-8.43	-8.35
-8.35	-8.43	-7.74	-6.14
-8.97	-8.35	-6.14	0

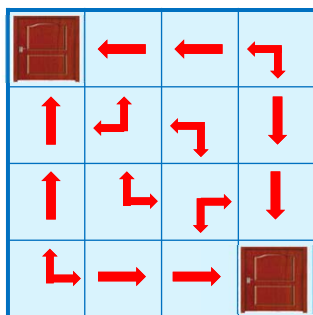
$k = 10$

-6.70	-8.70	-9.24	-1.00	-6.70	-8.70	-9.24	-6.70	-8.16	-8.78	-8.70	-8.70	-8.78	-8.16
-8.70	-9.24	-9.24	-8.16	-8.78	-8.70	-8.70	-8.78	-8.16	-6.70	-6.70	-8.70	-6.70	-1.00
-8.16	-8.78	-8.70	-8.70	-8.78	-8.16	-6.70	-9.24	-8.70	-6.70	-1.00	-9.24	-8.70	-6.70
-1.00	-6.70	-8.70	-6.70	-6.70	-8.16	-8.78	-8.70	-8.70	-8.78	-8.16	-9.24	-9.24	-8.70

1 2 3 4 5 6 7 8 9 10 11 12 13 14



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



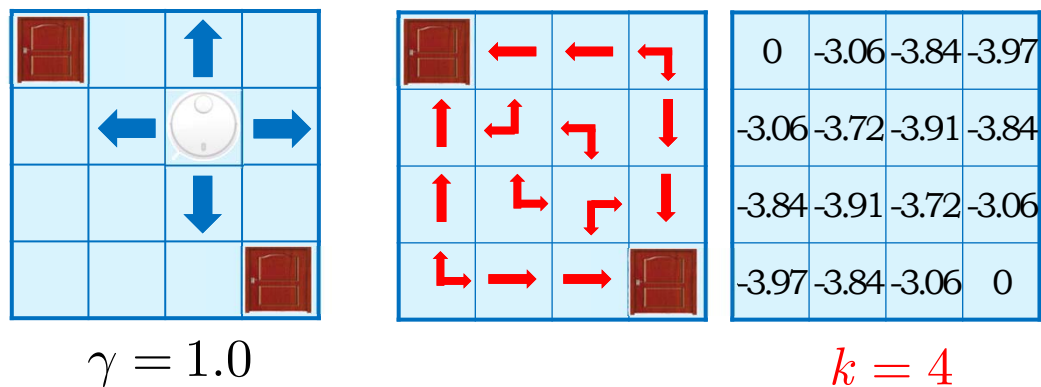
0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0

$k = 200$

-15	-21	-23	-1	-15	-21	-23	-15	-19	-21	-21	-21	-21	-19
-21	-23	-23	-19	-21	-21	-21	-21	-19	-15	-15	-21	-15	-1
-19	-21	-21	-21	-21	-19	-15	-23	-21	-15	-1	-23	-21	-15
-1	-15	-21	-15	-15	-19	-21	-21	-21	-21	-19	-23	-23	-21

1 2 3 4 5 6 7 8 9 10 11 12 13 14



$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$


If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

- ▶ 似乎没有必要迭代到收敛
- ▶ 何不每次迭代都更新策略

每次策略评价后立刻改进策略

策略评价：贝尔曼期望方程

策略改进：贪心策略

$$v_{\pi}(s) \approx v_{k+1}(s) = \sum_{a \in \mathbf{A}} \pi(a | s) \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_k(s') \right) \quad \pi'(a | s) = \begin{cases} 1 & \text{if } a = \arg \max_{a \in \mathbf{A}} q_{\pi}(s, a) \\ 0 & \text{otherwise} \end{cases}$$

$$q_{\pi}(s, a) \approx q_{k+1}(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{k+1}(s') \quad \pi'(a | s) \approx \begin{cases} 1 & \text{if } a = \arg \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{k+1}(s') \right) \\ 0 & \text{otherwise} \end{cases}$$

将贪心策略带入贝尔曼期望方程

$$v_{\pi'}(s) \approx v_{k+2}(s) = \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{k+1}(s') \right) \approx \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{\pi}(s') \right)$$

不断迭代

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

$$v_*(s) = \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_*(s') \right)$$

Bellman optimality equations

- ## ▶ 行动价值函数

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

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

- ▶ 最优状态价值是同时刻最优行动价值

$$v_*(s) = \max_{a \in \mathbf{A}} q_*(s, a)$$

- ### ► 行动价值由后续状态价值的期望计算

代入

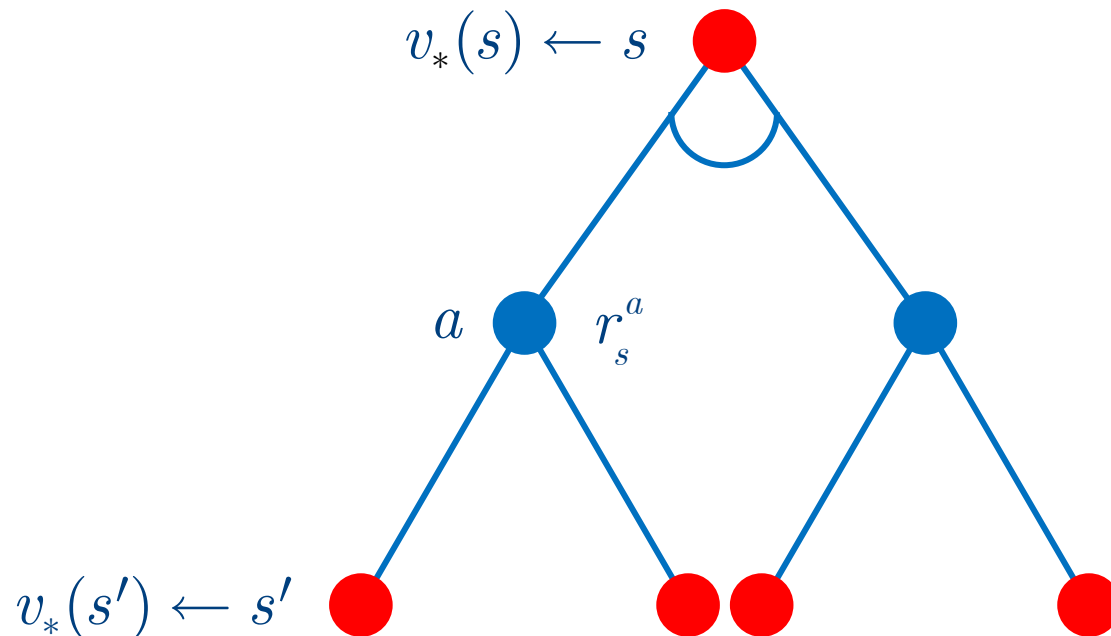
$$q_{\pi}(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{\pi}(s')$$

- ## ► 状态价值贝尔曼最优方程

$$v_*(s) = \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_*(s') \right)$$

- ## 状态价值贝尔曼期望方程

$$v_{\pi}(s) = \sum_{a \in \mathbf{A}} \pi(a \mid s) \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{\pi}(s') \right)$$

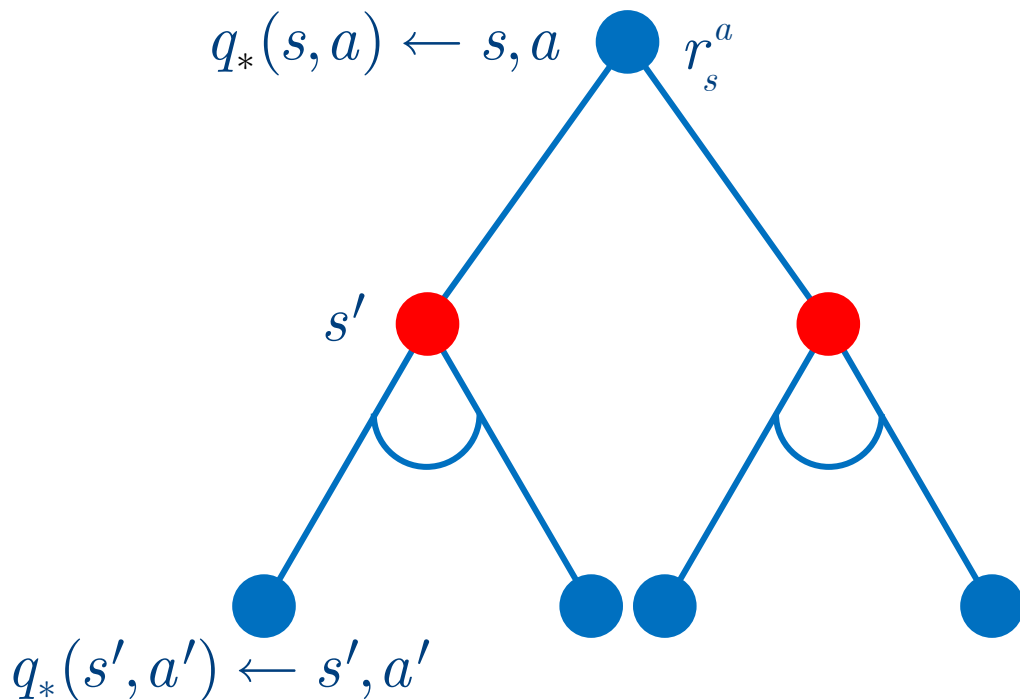


状态价值函数

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

最优状态价值是同时刻最优行动价值

$$v_*(s) = \max_{a \in \mathbf{A}} q_*(s, a)$$



行动价值函数

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

行动价值由后续状态价值的期望计算

代入

$$q_{\pi}(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{\pi}(s')$$

行动价值贝尔曼最优方程

$$q_*(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a \max_{a' \in \mathbf{A}} q_*(s', a')$$

行动价值贝尔曼期望方程

$$q_{\pi}(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a \sum_{a' \in \mathbf{A}} \pi(a' \mid s') q_{\pi}(s', a')$$

贝尔曼最优方程

Bellman Optimality Equations

▶ 状态价值函数

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

▶ 最优状态价值是同时刻最优行动价值

$$v_*(s) = \max_{a \in \mathbf{A}} q_*(s, a)$$

▶ 状态价值贝尔曼最优方程

$$v_*(s) = \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_*(s') \right)$$

▶ 状态价值贝尔曼期望方程

$$v_{\pi}(s) = \sum_{a \in \mathbf{A}} \pi(a \mid s) \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{\pi}(s') \right)$$

▶ 行动价值函数

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

▶ 行动价值由后续状态价值的期望计算

$$q_{\pi}(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{\pi}(s')$$

▶ 行动价值贝尔曼最优方程

$$q_*(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a \max_{a' \in \mathbf{A}} q_*(s', a')$$

▶ 行动价值贝尔曼期望方程

$$q_{\pi}(s, a) = r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a \sum_{a' \in \mathbf{A}} \pi(a' \mid s') q_{\pi}(s', a')$$

价值迭代

Value Iteration

- ▶ 问题：搜索最优策略 π
- ▶ 算法：迭代应用贝尔曼最优方程
- ▶ 计算： $\mathbf{v}^{(k+1)} = \max_{a \in \mathbf{A}} (\mathbf{r}^a + \gamma \mathbf{P}^a \mathbf{v}^{(k)})$
- ▶ 结果：生成序列 $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_*$

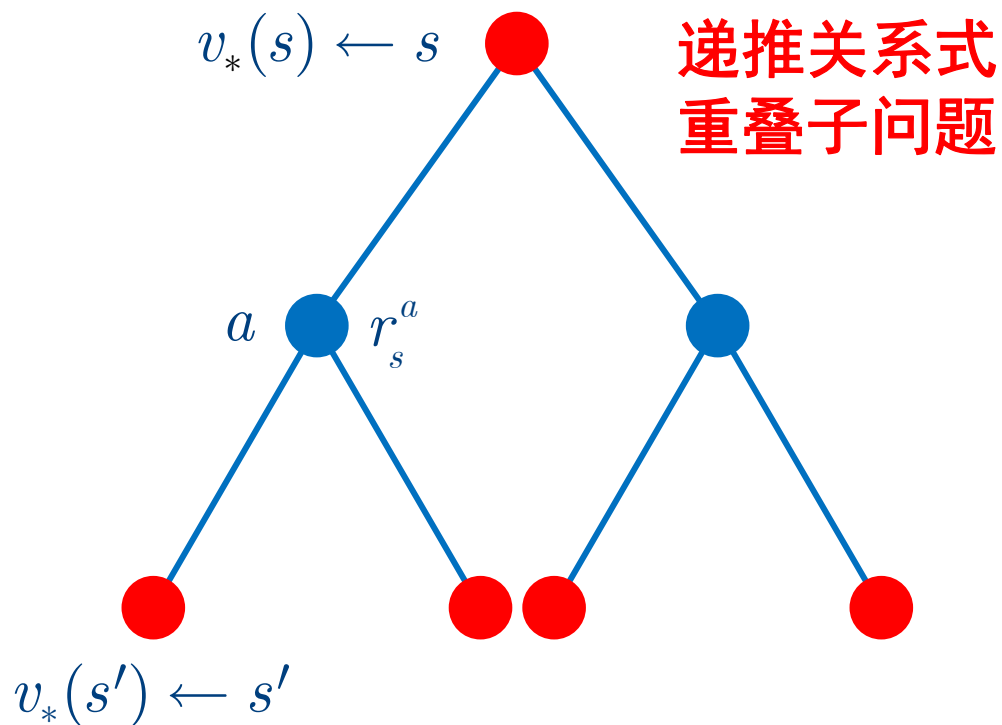
$$v_{k+1}(s) = \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_k(s') \right)$$

- ▶ 策略：

$$\pi_*(s) = \arg \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_*(s') \right)$$

- ▶ 状态价值贝尔曼最优方程

$$v_*(s) = \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_*(s') \right)$$



价值迭代

Value Iteration

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation

Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop:

```
|  $\Delta \leftarrow 0$   
| Loop for each  $s \in \mathcal{S}$ :  
|    $v \leftarrow V(s)$   
|    $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$   
|    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$   
until  $\Delta < \theta$ 
```

Output a deterministic policy, $\pi \approx \pi_*$, such that

$$\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$$

- ▶ 价值迭代
$$v_{k+1}(s) = \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_k(s') \right)$$
- ▶ 策略提取
$$\pi_*(s) = \arg\max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_*(s') \right)$$

同步迭代

- ▶ 算完所有状态后一次更新

$$v_{k+1}(s) = \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_k(s') \right)$$

- ▶ 保存两份状态价值
- ▶ 计算量大收敛较慢

异步迭代

- ▶ 算一个更新一个

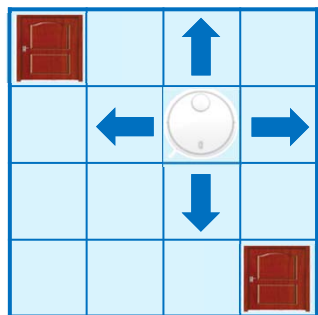
$$v(s) = \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v(s') \right)$$

- ▶ 只有一份状态价值
- ▶ 计算量小收敛较快
- ▶ 更新的顺序没关系

有点像梯度下降与随机梯度下降之间的关系

同步价值迭代示例

$$v_{k+1}(s) = \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_k(s') \right)$$



$\gamma = 1.0$

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

$k = 0$

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

$k = 1$

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

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

$k = 2$

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

$k = 3$

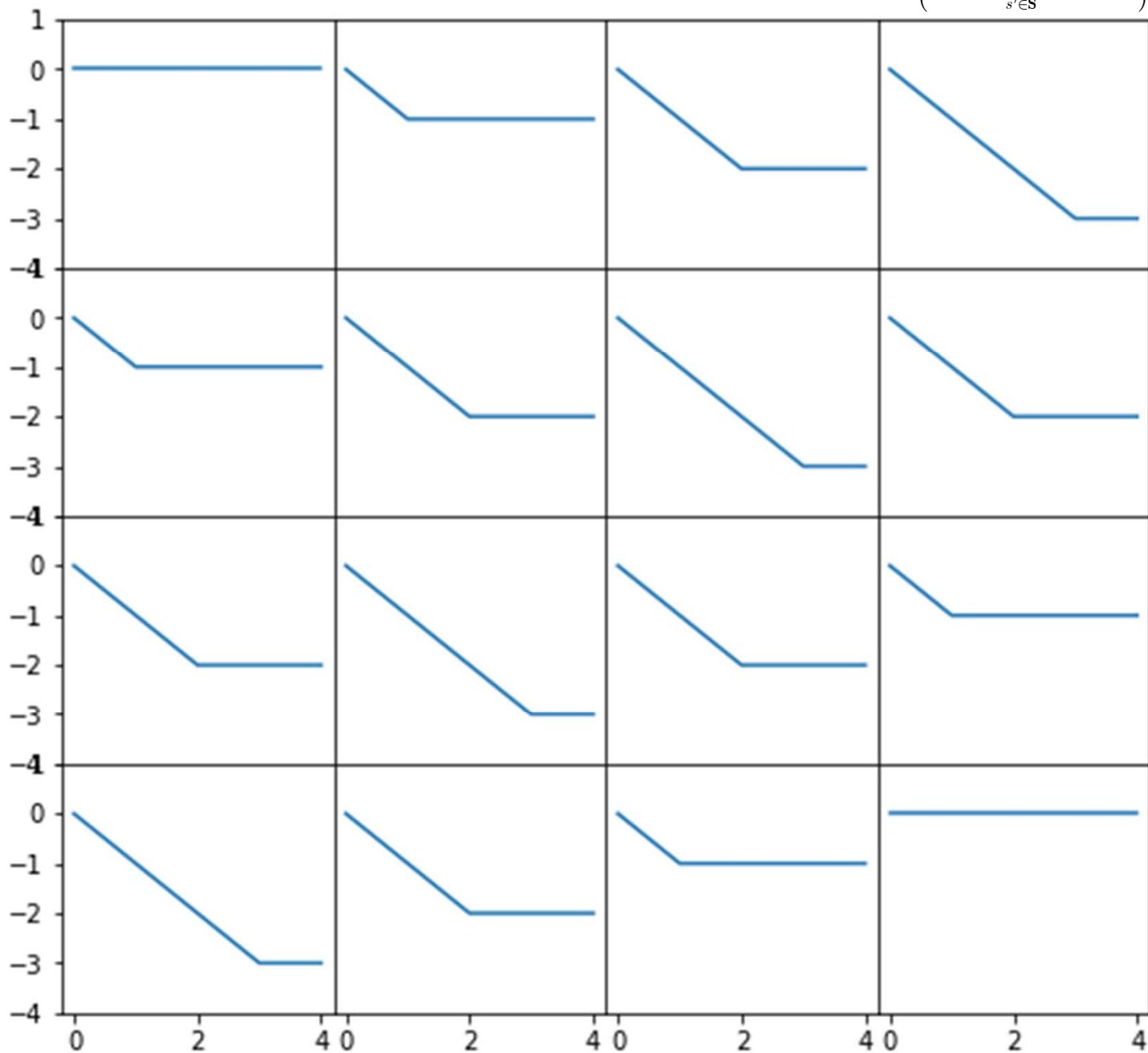
0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

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

$k = 4$

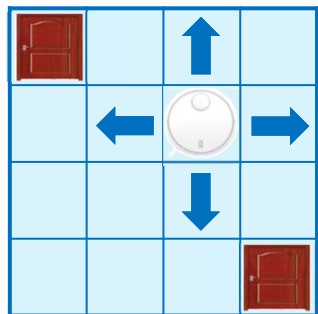
0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0

$k = \infty$



异步价值迭代示例

$$v(s) = \max_{a \in A} \left(r_s^a + \gamma \sum_{s' \in S} p_{ss'}^a v(s') \right)$$



$\gamma = 1.0$

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

$k = 0$

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

$k = 1$

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

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

$k = 2$

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

$k = 3$

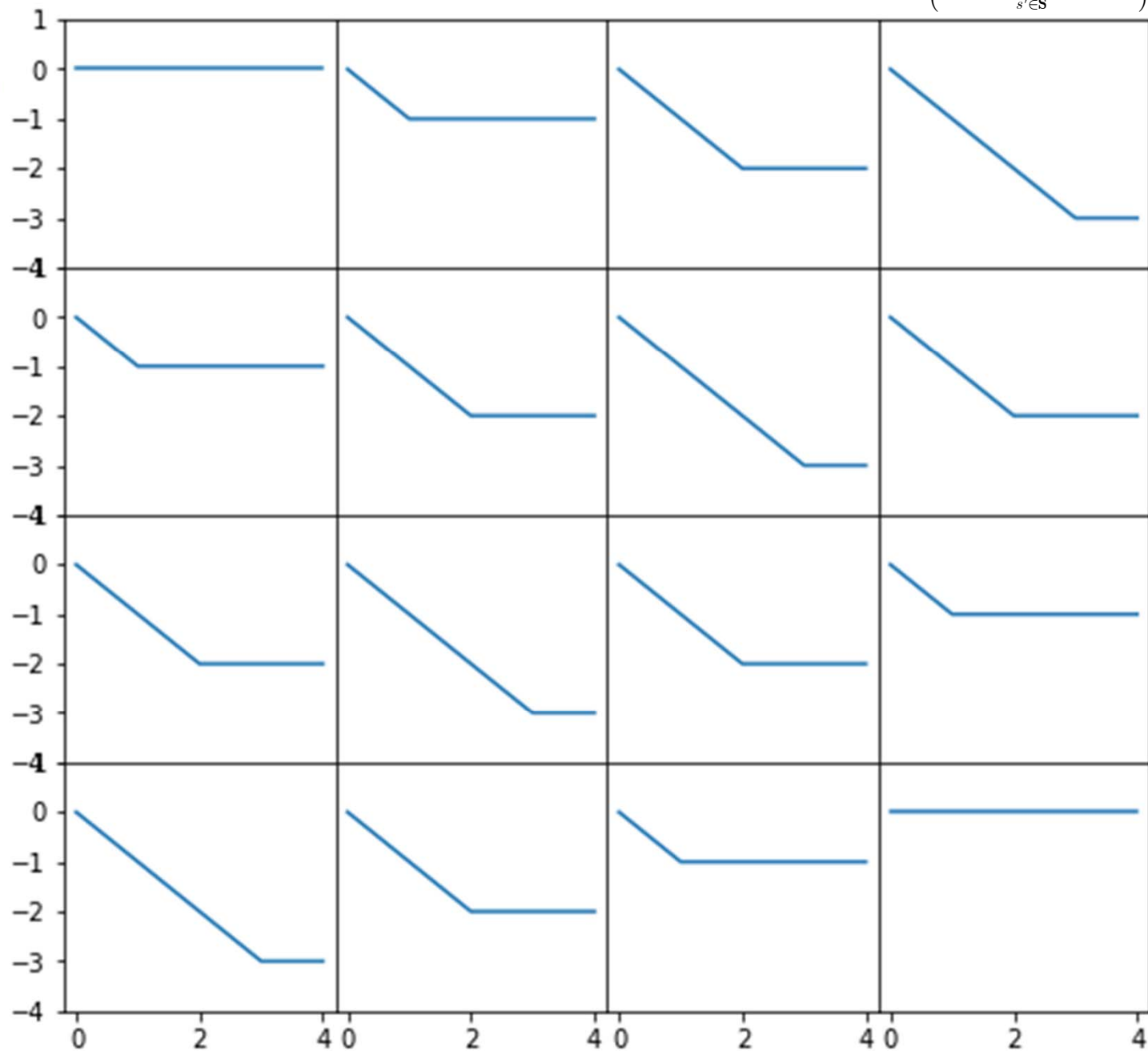
0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

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

$k = 4$

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

$k = \infty$



- ▶ 价值迭代是基于状态转移模型优化价值的方法，也就是获得最优策略、求解控制问题的方法
- ▶ 价值迭代的原理是基于卡尔曼最优方程和动态规划算法获得最优价值，进而提取出最优策略
- ▶ 价值迭代包括同步迭代和异步迭代两种
- ▶ 价值迭代与策略迭代相比，迭代次数少，收敛速度快
- ▶ 优化价值函数，获得最优策略
- ▶ 基于价值函数
 - ▶ 基于状态转移模型
 - ▶ 策略迭代
 - ▶ 价值迭代
 - ▶ 不基于状态转移模型
 - ▶ 蒙特卡洛控制
 - ▶ 时序差分控制：SARSA, Q-learning
 - ▶ 近似方法
 - ▶ 线性回归
 - ▶ 深度学习：DQN
- ▶ 基于策略模型

策略评价

$$v_{k+1}(s) = \sum_{a \in \mathbf{A}} \pi(a | s) \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_k(s') \right)$$

策略改进

$$\pi' = \arg \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_{\pi}(s') \right)$$



状态转移的概率分布

价值更新

$$v_{k+1}(s) = \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_k(s') \right)$$

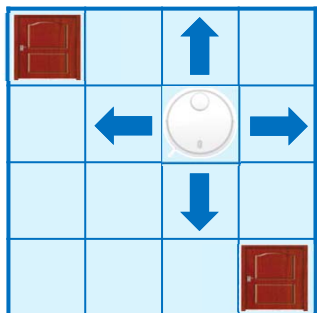
策略获取

$$\pi_*(s) = \arg \max_{a \in \mathbf{A}} \left(r_s^a + \gamma \sum_{s' \in \mathbf{S}} p_{ss'}^a v_*(s') \right)$$



模型

建立模型需要对环境有全局的了解



在动态环境中不太可能获取

有什么问题?

蒙特卡洛

那怎么解决?

时序差分

- ▶ 任选一种强化学习方法编程解决以下老鼠找蛋糕问题



12月23日网络学堂提交

周次	日期	内容	作业	编程	项目	内容
1	9月15日	绪论				绪论。课程内容。Python 简介
2	9月22日	无信息搜索		1		状态空间表示, 宽度优先, 一致代价, 深度优先
3	9月29日	有信息搜索				启发式搜索, 解项目
4	10月 6日	约束满足问题				回溯搜索
5	10月13日	对抗搜索				各树搜索
6	10月20日	命题逻辑				
7	10月27日	谓词逻辑				
8	11月 3日	线性回归				
9	11月10日	Logistic 回归				Logistic 回归, Softmax 回归, 机器学习项目
10	11月17日	前馈神经网络	6			前馈神经网络, 深度学习框架
11	11月24日	卷积神经网络		4		卷积神经网络, 深度学习框架
12	12月 1日	马尔可夫决策过程	7			强化学习的数学基础
13	12月 8日	策略迭代与价值迭代		5		状态转移概率已知时的预测与控制
14	12月15日	蒙特卡洛与时序差分	8			状态转移概率未知时的预测与控制
15	12月22日	深度强化学习				价值函数的近似, 深度强化学习。考试解读

考试时间：2022年1月3日（星期一）
下午 2:30~4:30
考试地点：一教201, 205

问题求解
逻辑推理
机器学习

Thank you very much!