强化学习原理：
——动态规划

杨博渊

yby@nankai.edu.cn

2023.03.03

提纲

- 动态规划概述
- 策略评估
- 策略迭代
- 值迭代
- 动态规划延申
- 压缩映射

强化学习的形式化表示

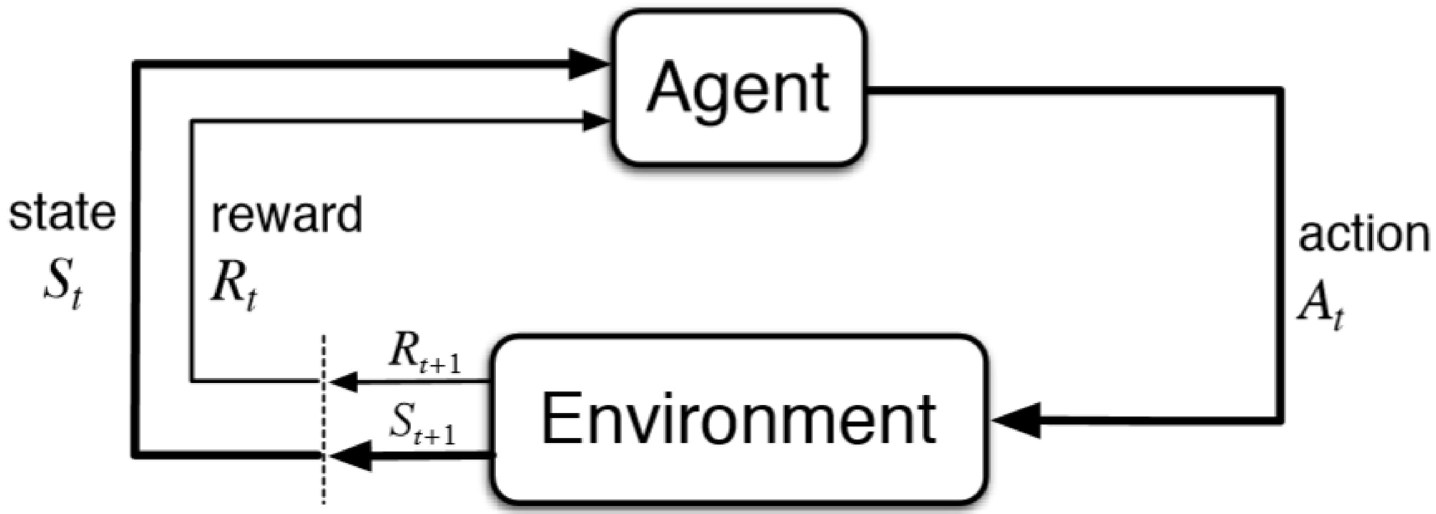
Agent: 智能体

Environment: 环境

智能体与环境进行序贯交互：

trajectory(轨迹)

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \cdots$$



智能体与环境交互

如果将状态 S_t 和回报 R_t 视为随机变量，那么该轨迹可以用一个随机过程来描述

马尔科夫决策过程：元组<S,A,P,R,γ>

提纲

- 动态规划概述
- 策略评估
- 策略迭代
- 值迭代
- 动态规划延申
- 压缩映射

动态规划

动态 问题包含序贯或时间成分
规划 优化一个“程序”

- 一种解决复杂问题的方法
- 把复杂问题分解为若干子问题
  - 求解子问题
  - 联合子问题的解

动态规划求解问题特性：

- 最优子结构
  - 最优化原理适用
  - 最优解可以分解为子问题
- 子问题重复性
  - 子问题多次重复出现
  - 解可以被储存并重复利用

动态规划

最短路径问题：从 q 到 r 的最短路径

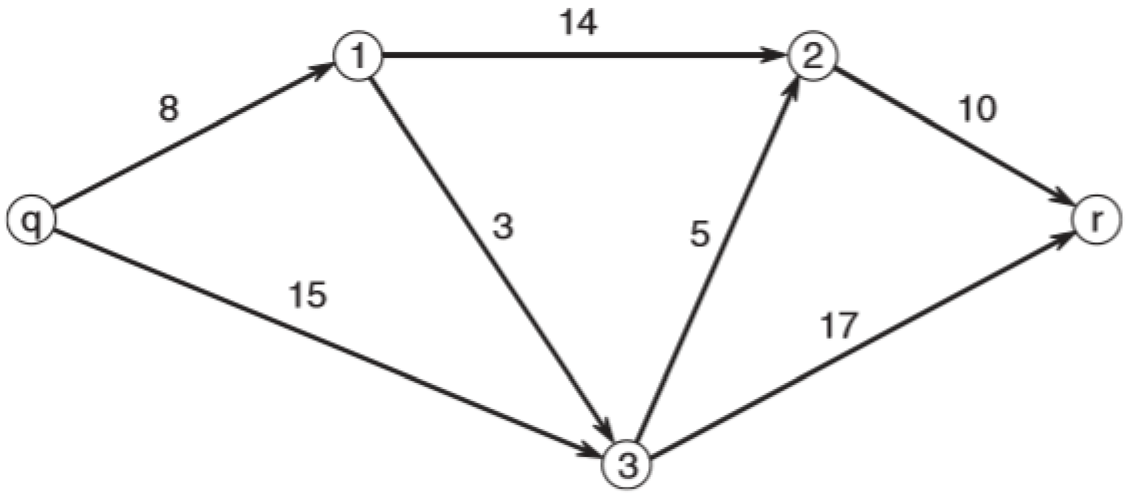


Table 1.1 Path cost from each node to node *r* after each node has been visited

	Cost from Node				
Iteration	q	1	2	3	r
	100	100	100	100	0
1	100	100	10	15	0
2	30	18	10	15	0
3	26	18	10	15	0
4	26	18	10	15	0

$$J^*[x(j), j] = \min_{\substack{u(j)\in U \\ x(j+1)\in X}} \left\{ L(x(j), u(j), j) + J^*[x(j+1), j+1] \right\}$$

值函数的更新公式为：

$$\upsilon_i \leftarrow \min\left\{\upsilon_i, \min_{j\in \mathrm{I}^+}\{c_{ij} + \upsilon_j\}\right\}$$

Warren B. Powell,
Approximate Dynamic Programming: Solving the Curses of Dimensionality

动态规划

动态规划的本质是：将多阶段决策问题通过贝尔曼方程转化为多个单阶段的决策问题

离散贝尔曼方程：

$$J^*[x(j),j]=\min_{u(j)\in U}\min_{\{u(j+1),\cdots,u(N-1)\}\in U}\{L[x(j),u(j),j]+\sum_{k=j+1}^{N-1}L[x[k],u[k],k]\}$$

$$=\min_{\substack{u(j)\in U\\ x(j+1)\in X}}\left\{L(x(j),u(j),j)+J^*[x(j+1),j+1]\right\}$$

$$=\min_{\substack{u(j)\in U\\ x(j+1)\in X}}\left\{L(x(j),u(j),j)+J^*[\,f[x(j),u(j),j],j+1]\right\}$$

求出值函数后，通过贪婪策略重构出最优策略

动态规划

动态规划的本质是：将多阶段决策问题通过贝尔曼方程转化为多个单阶段的决策问题

连续贝尔曼方程：

$$J^*[x(t),t]=\min_{u[t,t+\Delta t]}\left\{\min_{u[t+\Delta t,t_f]}\left[\int_t^{t+\Delta t}L(x(\tau),u(\tau),\tau)\right]d\tau+\int_{t+\Delta t}^{t_f}L(x(\tau),u(\tau),\tau)d\tau+\varphi[x(t_f),t_f]\right\}$$

$$=\min_{\substack{u(\tau)\in U\\ t\le\tau\le t+dt}}\left\{\int_t^{t+dt}L[x(\tau),u(\tau),\tau]d\tau+J^*[x(t)+dx(t),t+dt]\right\} \quad （1）$$

将 $J^*[x(t)+dx(t),t+dt]$ 进行泰勒展开有：

$$J^*[x(t)+dx(t),t+dt]=J^*[x(t),t]+\frac{\partial J^*[x(t),t]}{\partial x^T(t)}dx(t)+\frac{\partial J^*[x(t),t]}{\partial t}dt+\varepsilon[dx(t),dt] \quad （2）$$

胡寿松等， 最优控制理论与系统，科学出版社，2005

将（2）带入（1），并令 $dt\to 0$ Hamilton-Jacobi-Bellman方程

$$-\frac{\partial J^*[x(t),t]}{\partial t}=\min_{u(t)\in U}\left\{L[x(t),u(t),t]+\frac{\partial J^*[x(t),t]}{\partial x^T(t)}f[x(t),u(t),t]\right\} \xRightarrow{\text{重构}} \min_{u(t)\in U}\left\{L[x(t),u(t),t]+\frac{\partial J^*[x(t),t]}{\partial x^T(t)}f[x(t),u(t),t]\right\}$$

动态规划

马尔科夫决策过程符合动态规划的两个性质

- 贝尔曼方程分解为递归求解子问题
- 值函数储存了子问题的解

预测：

Input： MDP <S,A,P,R,γ> 和 策略π

Output： 值函数v_π

控制：

Input： MDP <S,A,P,R,γ>

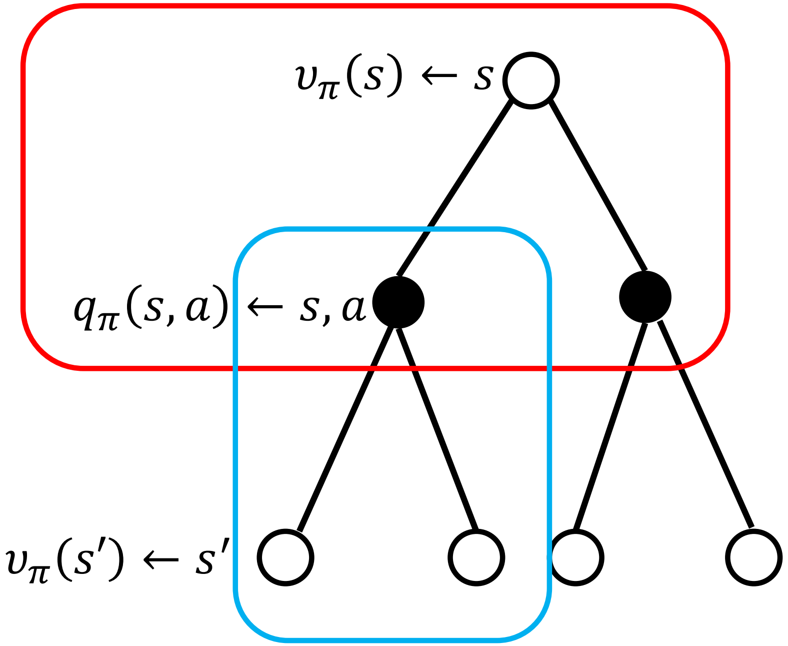
Output： 最优值函数v_*和最优策略π_*

提纲

- 动态规划概述
- 策略评估
- 策略迭代
- 值迭代
- 动态规划延申
- 压缩映射

策略评估

给定策略 π 构造值函数：



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

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

$$\Downarrow$$

$$\boxed{v_\pi(s) = \sum_{a \in A} \pi(a|s)\left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')\right)}$$

模型已知，方程组中只有值函数是未知数，方程组是线性方程组。未知数的数目等于状态的数目。

$$\Downarrow$$

采用值迭代算法

策略评估

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

⇩

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

策略评估算法

[1] 输入：需要评估的策略 π 状态转移概率 $P_{ss'}^a$ 回报函数 R_s^a ，折扣因子 γ

[2] 初始化值函数：$V(s) = 0$

一次状态扫描

[3] Repeat k=0,1,…

[4] for every s do

[5] $$v_{k+1}(s) = \sum_{a\in A}\pi(a|s)\left(R_s^a + \gamma\sum_{s'\in S}P_{ss'}^a v_k(s')\right)$$

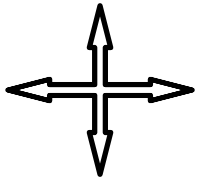
[6] end for

[7] Until $v_{k+1} = v_k$

[8] 输出： $v(s)$

策略评估

MDP



动作

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

状态空间：S={1,2..14}

动作空间:{东，南，西，北}

回报函数：−1，直到终止状态

均匀随机策略：

$\pi(东|\cdot)$=0.25， $\pi(南|\cdot)$=0.25， $\pi(西|\cdot)$=0.25， $\pi(北|\cdot)$=0.25

策略评估算法

输入：需要评估的策略 π 状态转移概率 $P_{ss'}^{a}$, 回报函数 R_s^a ，折扣因子 γ

初始化值函数：$V(s)=0$

一次状态扫描

Repeat k=0,1,…

for every s do

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

end for

Until $v_{k+1}=v_k$

输出： $v(s)$

策略评估

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=0

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=1

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

K=2

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

$K = \infty$

迭代策略评估，用于估算$V \approx v_\pi$

输入将要被评估的策略 π

算法参数：小阈值 $\theta > 0$ 确定估计的准确性

初始化一个数组 $V(s) = 0$，所有的 $s \in \mathcal{S}^+$，除了 $V(\text{终点}) = 0$

循环

$\Delta \leftarrow 0$

对于每个 $s \in \mathcal{S}$:

$v \leftarrow V(s)$

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

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

直到 $\Delta < \theta$ （一个小的正数）

提纲

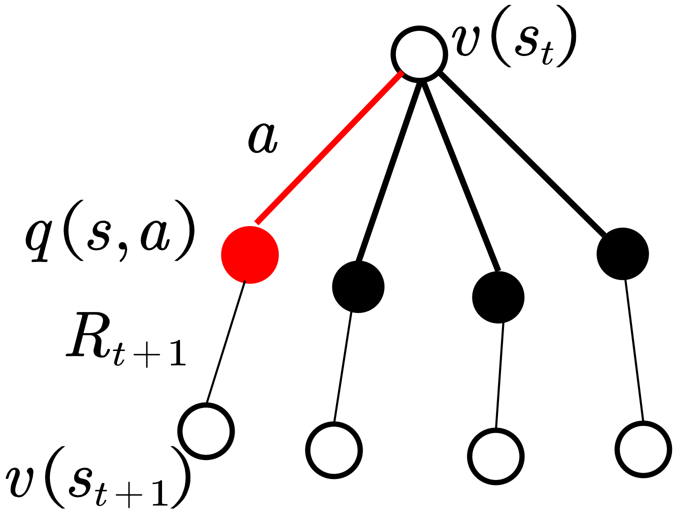
- 动态规划概述
- 策略评估
- 策略迭代
- 值迭代
- 动态规划延申
- 压缩映射

策略迭代：策略改进

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

π_0 均匀策略：

? $\Rightarrow$ 更好的策略 π_1



$$q_\pi(s,a) = \mathbb{E}[R_{t+1} + \gamma v_\pi(s_{t+1}) | s_t = s, a_t = a]$$

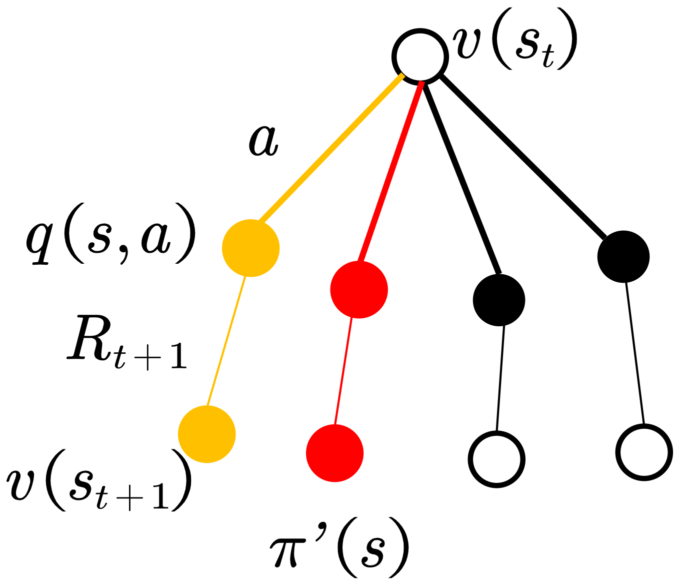
策略改善理论：

如果： $q_\pi(s,\pi'(s)) \geqslant v_\pi(s)$

那么：π' 不比 π 差，甚至比之还要好

策略迭代：策略改进

$$
\begin{aligned}
& v_\pi(s) \leqslant q_\pi(s,\pi'(s)) \\
& = \mathbb{E}[R_{t+1} + \gamma v_\pi(s_{t+1}) | s_t = s,\ a_t = \pi'(s)] \\
& = \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(s_{t+1}) | s_t = s] \\
& \leqslant \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(s_{t+1},\pi'(s_{t+1})) | s_t = s] \\
& = \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(s_{t+1}) | s_{t+1}, a_{t+1} = \pi'(s_{t+1})] | s_t = s] \\
& = \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(s_{t+2}) | s_t = s] \\
& \leqslant \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(s_{t+2}) | s_t = s] \\
& \vdots \\
& \leqslant \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots | s_t = s] \\
& = v_{\pi'}(s)
\end{aligned}
$$



最直观的一个改进策略是什么？ 贪婪策略！

策略迭代：策略改进

计算策略值的目的是为了帮助找到更好的策略，在每个状态采用贪婪策略。

$$\pi_{l+1}(s) \in \arg\max_{a} q^{\pi_l}(s,a)$$

π_0 均匀策略：✥

K=10

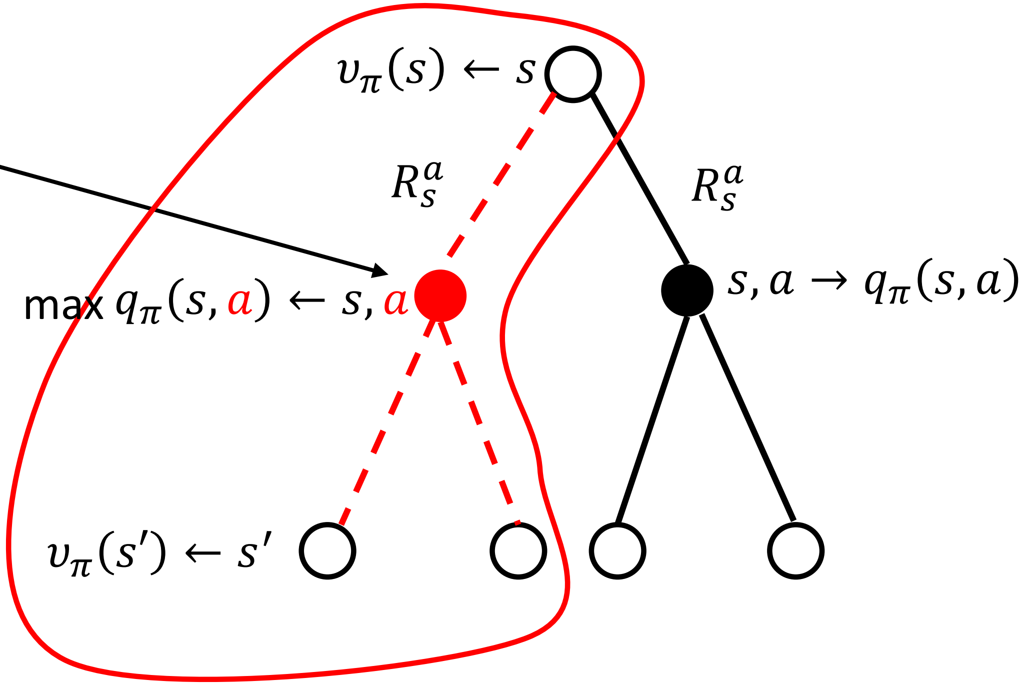
0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

π_1 贪婪策略：

0.0	←	←	←↓
↑	←↑	←↓	↓
↑	↑→	→↓	↓
↑→	→	→	0.0

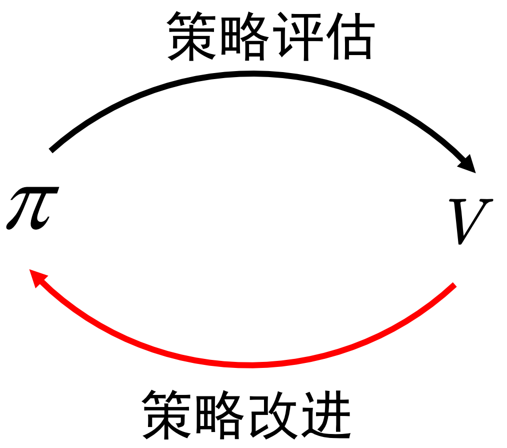
$K=\infty$

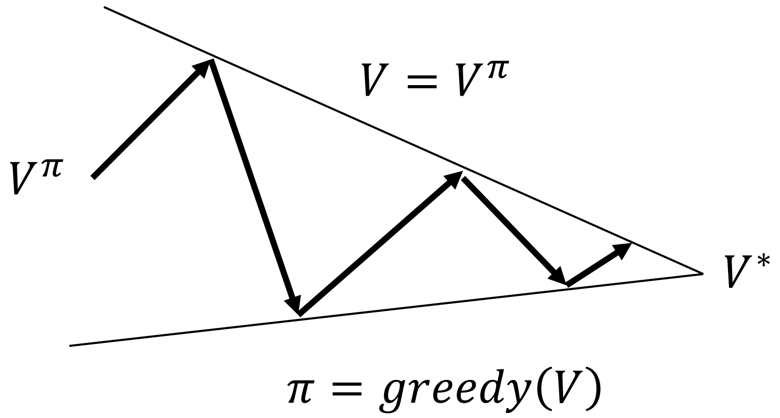
0.0	-14	-20	-22
-14	-18	-20	-20
-20	-20	-18	-14
-22	-20	-14	0.0



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

策略迭代





算法1：策略迭代算法

[1] 输入：状态转移概率 $P_{ss'}^{a}$,回报函数R_s^a ，折扣因子γ

初始化值函数：$V(s) = 0$ 初始化策略 π_0

[2] Repeat l=0,1,…

[3] find V^{π_l} Policy evaluation

[4] $\pi_{l+1}(s) \in \arg\max_a q^{\pi_l}(s,a)$ Policy improvement

[5] Until $\pi_{l+1} = \pi_l$

[6] 输出：$\pi^* = \pi_l$

提纲

- 动态规划概述
- 策略评估
- 策略迭代
- **值迭代**
- 动态规划延申
- 压缩映射

值迭代

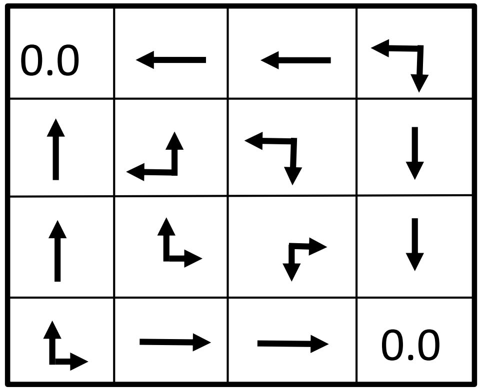
策略改进一定要等到值函数收敛吗？

π_0 均匀策略：

π_1 贪婪策略：

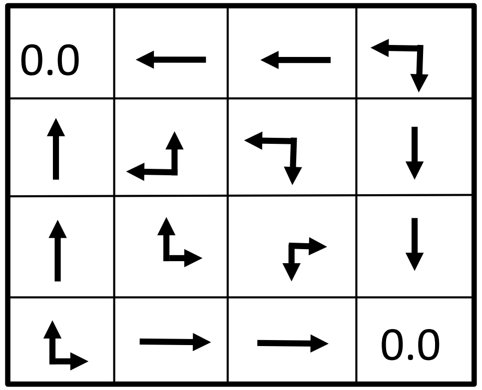
K=10

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



$K=\infty$

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



当K=1时便进行策略改进，得到值函数迭代算法

$$v^*(s) = \max_a R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s')$$

[1] 输入：状态转移概率 $P_{ss'}^a$,回报函数R_s^a ，折扣因子γ

初始化值函数:$v(s)=0$ 初始化策略 π_0

[2] Repeat l=0,1,…

[3] for every s do

[4] $$v_{l+1}(s) = \max_a R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_l(s')$$

[5] Until $v_{l+1} = v_l$

[6] 输出：$\pi(s) = \operatorname{argmax}_a R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_l(s')$

同步动态规划总结

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

问题	贝尔曼方程	算法
预测	贝尔曼期望方程	迭代策略评估
控制	贝尔曼期望方程+ 贪婪策略改进	策略迭代
控制	贝尔曼最优方程	值迭代

- 所有算法都基于状态值函数$V_\pi(s)$或$V_*(s)$
- 计算复杂度$O(mn^2)$
- 计算复杂度$O(m^2n^2)$

提纲

- 动态规划概述
- 策略评估
- 策略迭代
- 值迭代
- **动态规划延申**
- 压缩映射

异步动态规划

- 在位动态规划：直接原地更新下一个状态的v值，而不像同步迭代那样需要额外存储新的v值。

$$v(s) \leftarrow \max_{a\in\mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v(s')\right)$$

$$v_{new}(s) \leftarrow \max_{a\in\mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_{old}(s')\right)$$

$$v_{old} \leftarrow v_{new}$$

- 重要状态优先更新：Bellman error

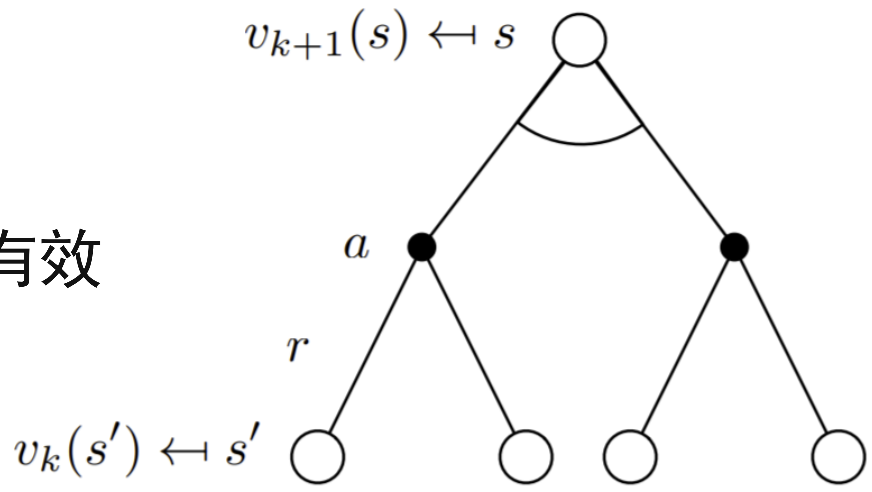
$$\left|\max_{a\in\mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v(s')\right) - v(s)\right|$$

- 实时动态规划：更新那些仅与个体关系密切的状态

$$v(S_t) \leftarrow \max_{a\in\mathcal{A}} \left(\mathcal{R}_{S_t}^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{S_t s'}^a v(s')\right)$$

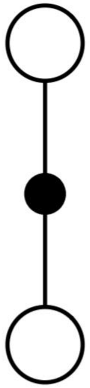
采样更新

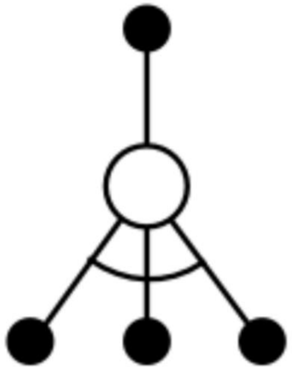
- 动态规划使用full-width backups
- 对于每一次状态更新
  - 考虑到其所有后继状态及所有可能的行为
  - 使用MDP中的状态转移矩阵、奖励函数（信息）
- 对于中等规模（百万级别的状态数）的问题较为有效
- 对于大规模DP问题，会带来维数灾难
  - 状态数导致状态变量指数增加



采样更新

- 采样更新 sample backups
- 利用样本奖励和样本转移
  <S,A,R,S’>
  代替奖励函数和状态转移矩阵





- 优点：
  - 不基于模型：无需MDP的更多信息
  - 通过采用打破维数灾难
  - 更新成本时一个关于状态数的常数

近似动态规划

- 使用其他技术手段（例如神经网络）建立一个参数较少，消耗计算资源较少、同时虽然不完全精确但却够用的近似价值函数：

$$\tilde{v}_k(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \hat{v}(s', \mathbf{w_k}) \right)$$

提纲

- 动态规划概述
- 策略评估
- 策略迭代
- 值迭代
- 动态规划延申
- 压缩映射

问题

- 为什么迭代策略评估收敛到v_π？
- 为什么策略迭代收敛到v_*？
- 为什么值迭代收敛到v_*？
- 解是唯一的吗？
- 上述算法的收敛速度？

压缩映射理论

压缩映射理论

对于任意的度量空间V，如果在算子T(v)下是完备的，
其中T是γ压缩的，那么：

- T收敛到一个唯一的固定点
- 收敛速度与γ线性相关

第三次作业

1.阅读《Reinforcement Learning： An Introduction》第四章，并做读书笔记

2. 阅读策略迭代和值迭代解决鸳鸯找朋友的问题的代码

3. 利用策略迭代和值迭代解决gym中离散的问题