WDS

DataScience@Web

第二部分

动态规划

汇报人：蔡新宇

2021.7.8

2021WDS暑期讨论班

目录

- 概述
- 策略评估
- 策略迭代
- 价值迭代
- 总结

动态规划

动态规划是运筹学的一个分支，是求解决策过程最优化的过程。

使用动态规划解决的问题包含以下性质：

- 最优子结构
- 重叠子问题

MDP满足以上性质：

- 贝尔曼方程将问题递归分解
- 价值函数保存和重用问题的解

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

两种规划问题

- 预测：

  □□输入： MDP ⟨S, A, P, R, γ⟩和策略 π，

  输出：V_π□□

□□□□□□

- 控制：

  输入： MDP ⟨S, A, P, R, γ⟩

  输出：V_*(最优价值函数)和π_*(最优策略)

目录

- 概述
- 策略评估
- 策略迭代
- 价值迭代
- 总结

策略评估

输入：MDP ⟨S, A, P, R, γ⟩和策略 π，

输出：V_π　　$v_\pi = (1 - \gamma P^\pi)^{-1} R^\pi$

算法：从任意一个状态价值函数V开始，依据给定的策略π，结合贝尔曼期望方程、状态转移概率、和奖励同步迭代更新V，直至V收敛。

$$V_1 \to V_2 \to \cdots \to V_\pi$$

贝尔曼期望方程：$v_{k+1}(s) \doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s]$

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

使用上一个迭代周期k内的后续状态价值来计算当前当前迭代周期k+1内某状态s的价值。

网格世界的策略评估

MDP〈 S, A, P, R, γ〉 ：

状态空间S：s_1-s_{14}为非终止状态、灰色为终止状态

行为空间A：向东西南北移动四个行为，进入终止状态终止

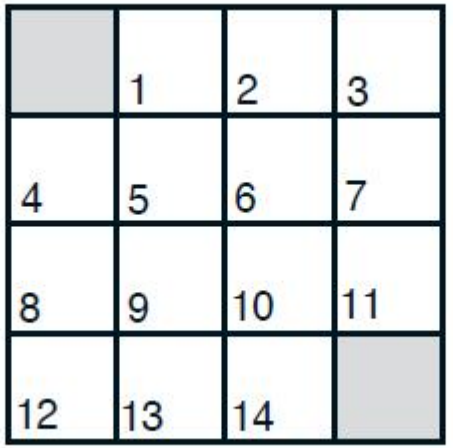
转移概率P：任何试图离开方格世界的动作将100% 停留在原状态，其余条件下将100%地转移到动作指向的状态；

即时奖励R：均为-1

衰减系数γ：1

策略π：在任何一个非终止状态下有均等的几率采取任一移动方向这个行为

actions



$R_t = -1$
on all transitions

网格世界的策略评估

$$V_1^{(21)} = \frac{1}{4}[(-1+0)*4] = -1$$

$$V_1^{(22)} = \frac{1}{4}[(-1+0)*4] = -1$$

$$V_2^{(21)} = \frac{1}{4}[(-1-1)*3+(-1+0)] = -1$$

$$V_2^{(22)} = \frac{1}{4}[(-1-1)*4] = -2$$

$$V_3^{(21)} = \frac{1}{4}[(-1-2.0)*2+(-1-1.7)+(-1+0)] = -2.425$$

$$V_3^{(22)} = \frac{1}{4}[(-1-1.7)*2+(-1-2)*2] = -2.85$$

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

(a) 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

(b) 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

(c) k=2

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

(d) k=3

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

(e) k=10

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

(f) k=∞

算法

Iterative policy evaluation

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

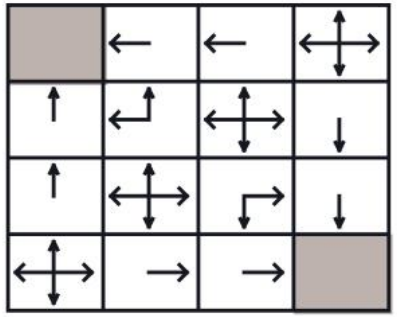
目录

- 概述
- 策略评估
- 策略迭代
- 价值迭代
- 总结

策略迭代

$k = 2$

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



给定一个策略π

策略评估

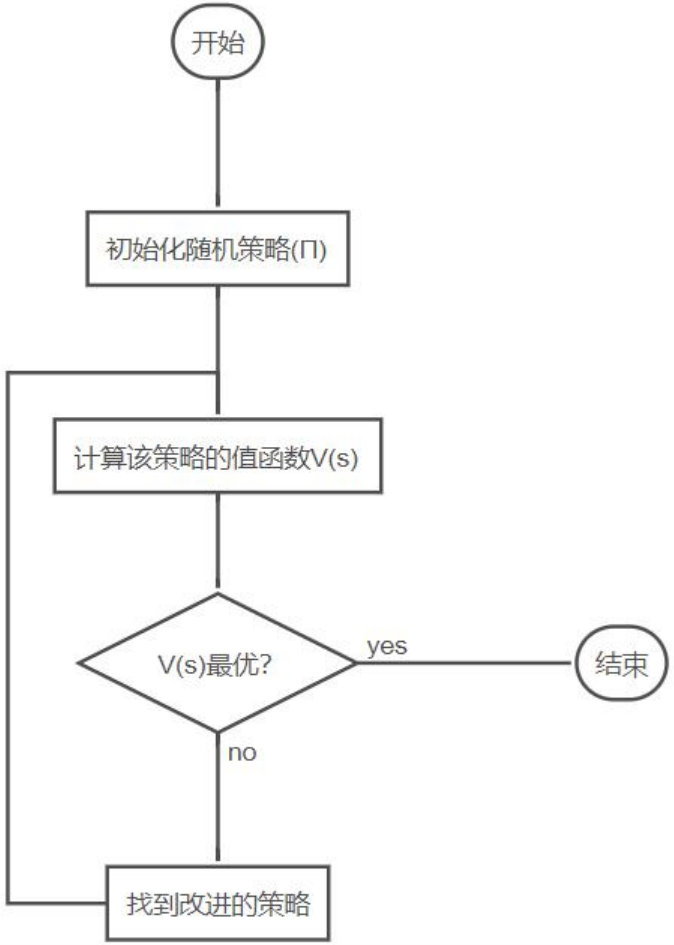
$$v_\pi(s) = \mathrm{E}\left[R_{t+1} + \gamma R_{t+2} + \ldots | S_t = s\right]$$

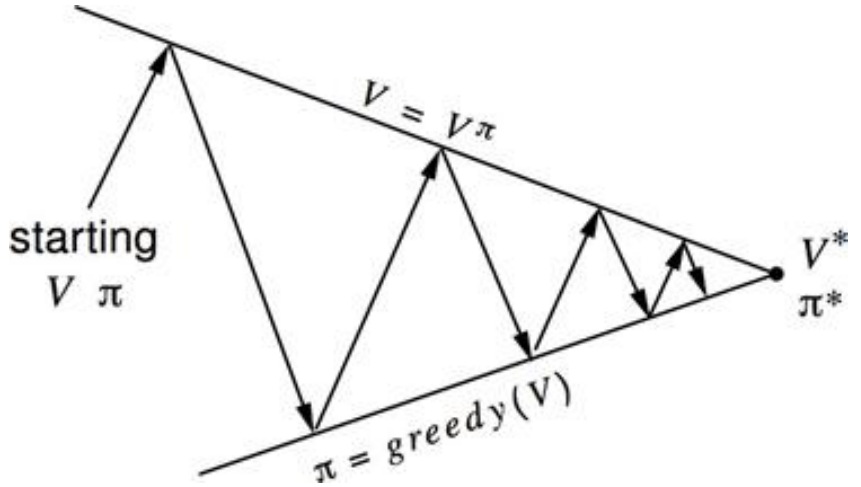
策略改进 基于v_π通过贪心算法找到新的策略

$$\pi^t = \mathrm{greedy}(v_\pi)$$

在格子问题中， $\pi = \pi^t$

策略迭代





$$\pi_0 \overset{E}{\Rightarrow} V_{\pi_0} \overset{I}{\Rightarrow} \pi_1 \overset{E}{\Rightarrow} V_{\pi_1} \overset{I}{\Rightarrow} \pi_2 \overset{E}{\Rightarrow} \ldots \overset{I}{\Rightarrow} \pi_* \overset{E}{\Rightarrow} V_*$$

策略改进证明

- 给定一个确定的策略π：$a = \pi(s)$
- 贪心策略在同样的状态s下得到新的行为：$a' = \pi'(s)\ \ \pi'(s) = \underset{a \in A}{argmax}\ q_\pi(s,a)$
- 假如仅在下一步采取该贪心策略产生的行为，而后续步骤仍采用原策略的行为，那么

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

- 如果对后续每个状态s均使用贪心策略产生的行为，那么有：$v_{\pi'} \geq v_\pi(s)$，推导如下：

$$\begin{aligned} v_\pi(s) &\leq q_\pi(s,\pi'(s)) = \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right] \\ &\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma q_\pi(S_{t+1},\pi'(S_{t+1})) \mid S_t = s\right] \\ &\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 q_\pi(S_{t+2},\pi'(S_{t+2})) \mid S_t = s\right] \\ &\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \ldots \mid S_t = s\right] = v_{\pi'}(s) \end{aligned}$$

策略改进证明

- 如果一直迭代到状态价值函数不再改善，即：

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

- 此时，满足了bellman最优方程：$v_*(s) = \max_a q_*(s,a)$

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

- 因此，对于所有的状态s，满足：$v_\pi(s) = v_*(s)$
- 故此时的策略π为最优策略

算法

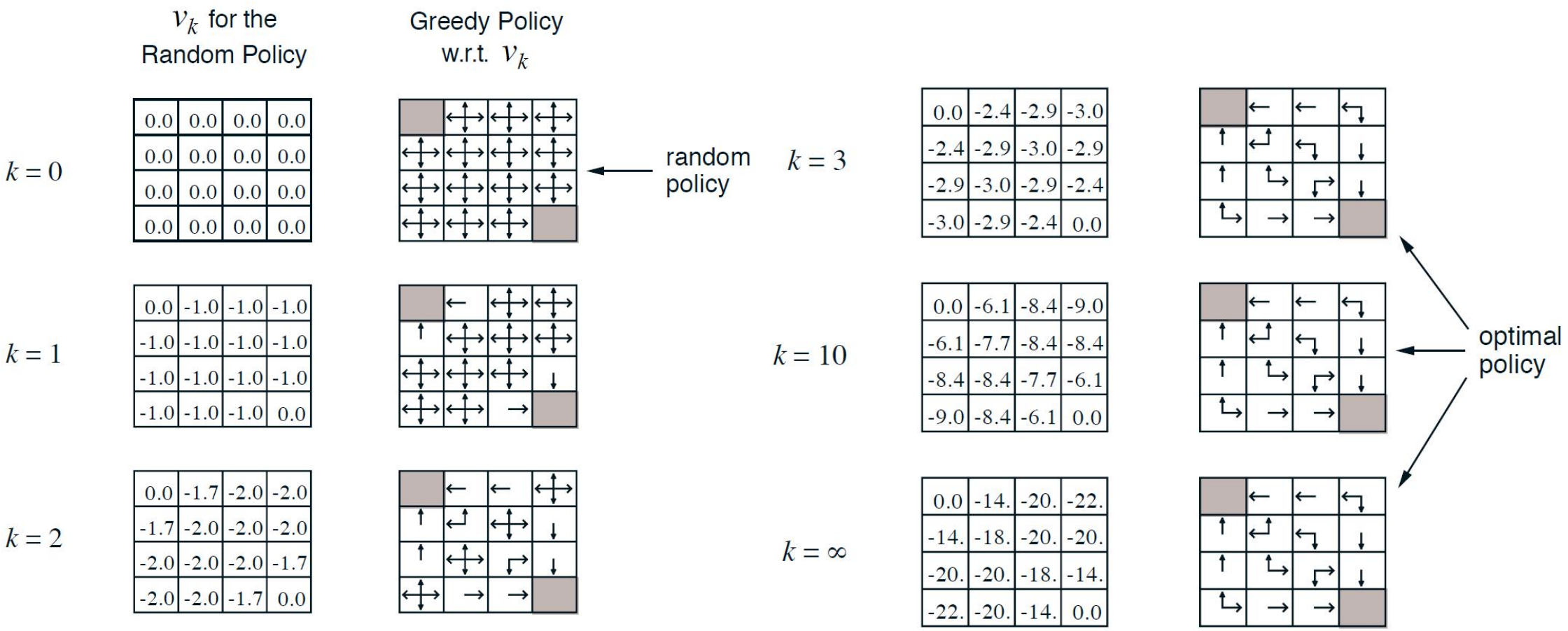
Policy iteration (using iterative policy evaluation)

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

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\left[r + \gamma V(s')\right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
   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

网格世界


v_k for the Random Policy
Greedy Policy w.r.t. v_k
$k = 0$
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
random policy
$k = 1$
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
$k = 3$
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
$k = \infty$
0.0 -14. -20. -22.
-14. -18. -20. -20.
-20. -20. -18. -14.
-22. -20. -14. 0.0
optimal policy

策略迭代的改进

- 设置一些条件提前终止迭代

  例如，设置ε，当价值函数的更新小于ε时就停止迭代。

- 直接设置迭代次数

  例如，在网格世界中，设置迭代次数k=3

- 每迭代一次便更新一次策略

  即价值迭代

目录

- 概述
- 策略评估
- 策略迭代
- 价值迭代
- 总结

价值迭代

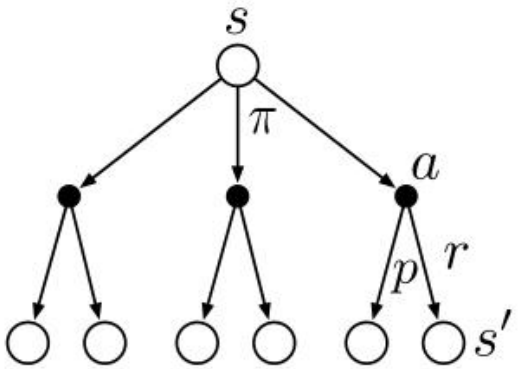
问题：寻找最优策略π

解决方案：从初始状态开始同步迭代计算到最终收敛

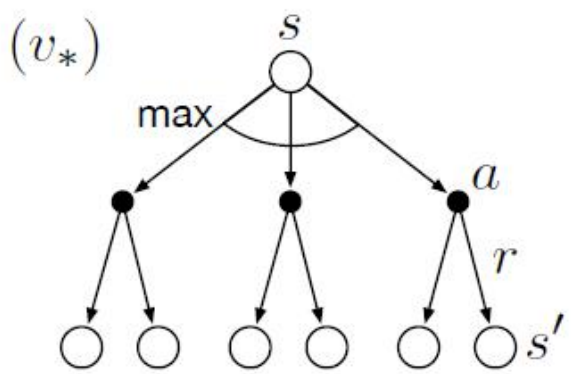
贝尔曼最优方程：

$$v_{k+1}(s) \doteq \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a]$$
$$= \max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_k(s')\Big],$$

Policy Evaluation



Value Iteration



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

算法

Value iteration

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$

小结

$$V_1 \to V_2 \to \cdots \to V_\pi$$

$$\pi_0 \overset{E}{\Rightarrow} V_{\pi_0} \overset{I}{\Rightarrow} \pi_1 \overset{E}{\Rightarrow} V_{\pi_1} \overset{I}{\Rightarrow} \ldots \overset{I}{\Rightarrow} \pi_* \overset{E}{\Rightarrow} V_*$$

$$V_1 \to V_2 \to \cdots \to V_*$$

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

目录

- 概述
- 策略评估
- 策略迭代
- 价值迭代
- 总结

总结

- 策略评估是给定策略下值函数的迭代计算，解决预测问题。
- 策略迭代和值迭代是将策略评估和策略提升结合的算法，解决控制问题。
- 在给定MDP的所有先验知识的情况下，策略迭代和价值迭代均能求解出最优策略和价值函数。
- 收敛性的证明：压缩映射定理[1]
- 局限性：一般用于所有情况都能预先知道的问题、问题规模大会发生维度灾难。

https://zhuanlan.zhihu.com/p/39279611 [1]

谢 谢