

强化学习基础

目录

1. 策略迭代
2. 值迭代
3. 实验任务及报告提交要求

1.策略迭代

单智能体RL形式化定义：

由六元组构成的马尔可夫决策过程定义

具体定义如下：

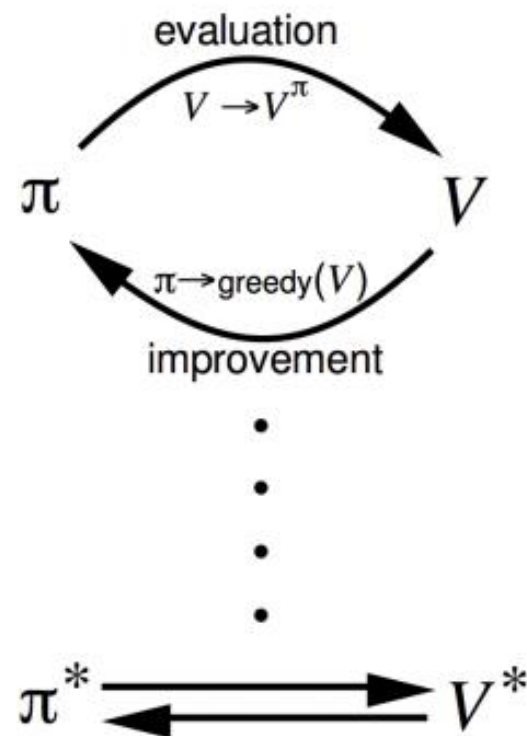
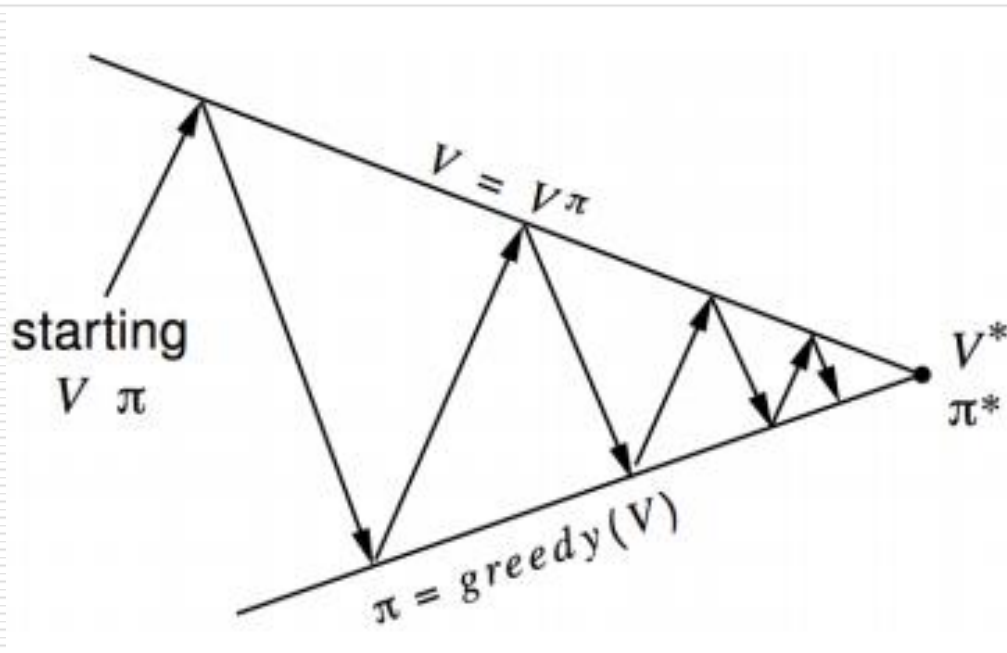
Markov Decision Process(MDP) $(S, A, R, T, P_0, \gamma)$

- S denotes the state space
- A is the action space
- $R = R(s, a)$ is the reward function
- $T: S \times A \times S \rightarrow [0,1]$ is the state transition function
- P_0 is the distribution of the initial state
- γ is a discount factor
- Goal: find the optimal policy that maximizes expected reward



1.策略迭代

从一个初始化的策略出发，先进行策略评估（policy evaluation），然后改进策略（policy improvement），评估改进的策略，再进一步改进策略，经过不断迭代更新，直达策略收敛，这种算法被称为“策略迭代”



1.策略迭代

策略迭代算法

- 策略迭代算法

- ① 先给定初始策略 π ，求解线性方程，计算出 π 相应的值函数 V

$$V(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s, a, s') V(s').$$

- ② 基于计算出的值函数 V ，按下式更新策略

$$\pi(s) \leftarrow \operatorname{argmax}_a \left\{ R(s, a) + \gamma \sum_{s'} T(s, a, s') V(s') \right\}$$

- ③ 不断重复步骤 1, 2，直到收敛

- 策略迭代算法最终收敛到最优值

- 收敛到最优值所需的迭代不大于 $|A|^{|S|}$ （确定性策略的总数目）

1.策略迭代

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 $\leftarrow \pi(s)$

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

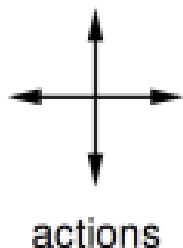
If *old-action* $\neq \pi(s)$, then *policy-stable* \leftarrow false

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

https://blog.csdn.net/qq_30615903

1.策略迭代

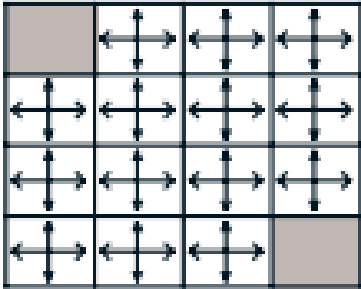
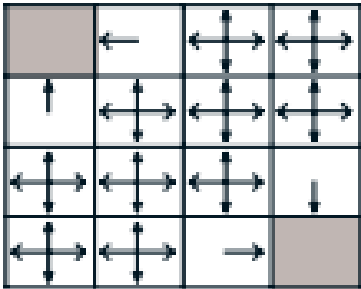
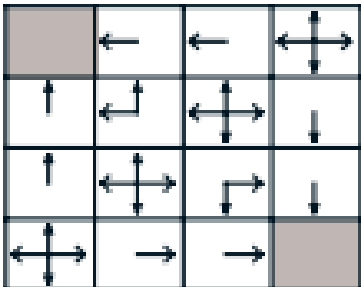
实例：一个 4×4 的小网格世界，左上角和右下角是目的地，每个格子行动方向为上下左右，每走一步reward-1，求一个在每个状态都能以最少步数到达目的地的最优行动策略。解决思路：我们从最开始的随机（1/4）策略开始，对其进行policy evaluation, 然后进行policy iteration by acting greedy



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

$\gamma = 1$
 $r = -1$
on all transitions

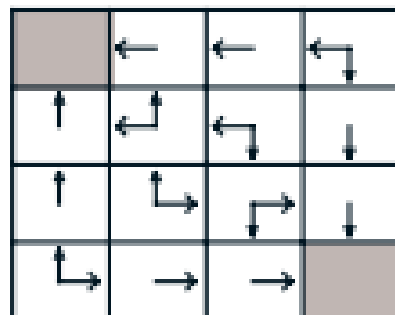
1.策略迭代

	v_k for the Random Policy	Greedy Policy w.r.t. v_k																	
$k = 0$	<table> <tr><td>0.0</td><td>0.0</td><td>0.0</td><td>0.0</td></tr> <tr><td>0.0</td><td>0.0</td><td>0.0</td><td>0.0</td></tr> <tr><td>0.0</td><td>0.0</td><td>0.0</td><td>0.0</td></tr> <tr><td>0.0</td><td>0.0</td><td>0.0</td><td>0.0</td></tr> </table>	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
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																
$k = 1$	<table> <tr><td>0.0</td><td>-1.0</td><td>-1.0</td><td>-1.0</td></tr> <tr><td>-1.0</td><td>-1.0</td><td>-1.0</td><td>-1.0</td></tr> <tr><td>-1.0</td><td>-1.0</td><td>-1.0</td><td>-1.0</td></tr> <tr><td>-1.0</td><td>-1.0</td><td>-1.0</td><td>0.0</td></tr> </table>	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		
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																
$k = 2$	<table> <tr><td>0.0</td><td>-1.7</td><td>-2.0</td><td>-2.0</td></tr> <tr><td>-1.7</td><td>-2.0</td><td>-2.0</td><td>-2.0</td></tr> <tr><td>-2.0</td><td>-2.0</td><td>-2.0</td><td>-1.7</td></tr> <tr><td>-2.0</td><td>-2.0</td><td>-1.7</td><td>0.0</td></tr> </table>	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		
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																

1.策略迭代

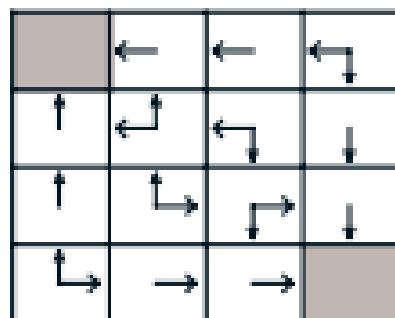
$k = 3$

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



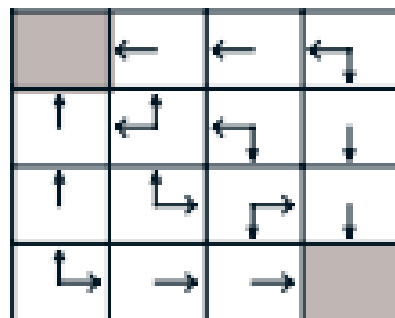
$k = 10$

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

2. 值迭代

对每一个当前状态 s ,对每个可能的动作 a 都计算一下采取这个动作后到达的下一个状态的期望价值。看看哪个动作可以到达的状态的期望价值函数最大, 就将这个最大的期望价值函数作为当前状态的价值函数 $V(s)$, 循环执行这个步骤, 直到价值函数收敛。

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

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

https://blog.csdn.net/qq_30615903

2. 值迭代

值迭代算法

- 值迭代 (value iteration) 算法求 V_t^* 序列, 借助于辅助 $Q_t^a(s)$, 其直观含义为: 在 s 执行 a , 然后执行 $t-1$ 步的最优策略所产生的预期回报

对所有 $s \in S$, $V_0(s) := 0$; $t := 0$;

loop

$t := t + 1$;

loop 对所有 $s \in S$

loop 对所有 $a \in A$

$$Q_t^a(s) := R(s, a) + \gamma \sum_{s'} T(s, a, s') V_{t-1}(s')$$

end loop

$$V_t(s) := \max_a Q_t^a(s) \text{ (Bellman equation)}$$

end loop

until $|V_t(s) - V_{t-1}(s)| < \epsilon$ 对所有 $s \in S$ 成立

3 实验任务及报告提交要求

实验任务

- 实现策略迭代和值迭代算法，并输出算法收敛后的路径。

报告提交要求

- 提交一个压缩包。压缩包命名为：“学号_姓名_作业编号”，例如：
20220525_张三_实验9。
- 压缩包包含三部分：code文件夹和实验报告pdf文件
 - Code文件夹：存放实验代码
 - Pdf文件格式参考发的模板
- 如果需要提交新版本，则在压缩包后面加_v1等。如“学号_姓名_作业编号_v1.zip”，以此类推。
- 截止日期：当天完成，即6月2日24点
- 提交邮箱：zhangyc8@mail2.sysu.edu.cn

附录

策略迭代和值迭代均需给出最优路径，可视化图如右图。

```
(Right)
SFFFFFFF
FFFFFFFF
FFFHFFFF
FFFFFHFF
FFFFFFFF
FHFFFFHF
FHFFHFHF
FFFHFFFG
(Right)
SFFFFFFF
FFFFFFFF
FFFFFFFF
FFFFFHFF
FFFFFHFF
FHFFFFHF
FHFFHFHF
FFFHFFFG
(Down)
SFFFFFFF
FFFFFFFF
FFFFFFFF
FFFFFHFF
FFFFFHFF
FHFFFFHF
FHFFHFHF
FFFHFFFG
(Down)
SFFFFFFF
FFFFFFFF
FFFFFFFF
FFFFFHFF
FFFFFHFF
FHFFHFHF
FHFFHFHF
FFFHFFFG
(Right)
SFFFFFFF
FFFFFFFF
FFFFFFFF
FFFFFHFF
FFFFFHFF
FHFFFFHF
FHFFHFHF
FFFFFHFF
Episode reward: 1.000000
```