第一章 求解贝尔曼方程

1.1 设计思想

1. 网格环境定义

网格结构：使用一维向量 grid 定义网格世界，包含不同的状态和对应的奖励。状态可以是普通状态、禁区、目标或边界。

奖励设置：

OTHERSTEP：普通状态的奖励（0）。

FORBIDDEN：禁区的惩罚（-1）。

TARGET：目标状态的奖励（1）。

BORDER：越界惩罚（-1）。

设置 gamma。

2. 动作定义

定义五种可能的动作：RIGHT、DOWN、UP、LEFT、STAY。每个动作对应一个整数值。

3. 策略初始化

使用一维向量 policy 表示给定策略。

4. 初始化奖励向量 R 和状态转移矩阵 P

P：根据当前策略，生成一个状态转移矩阵，记录从每个状态转移到下一个状态的概率，初始化为 0。

R：为每个状态计算对应的奖励值。

5. 状态转移与奖励计算

循环策略中每一个状态，根据当前状态和所选择的动作，计算下一个状态，设置目标状态概率为 1，得到 P；并根据越界或禁区情况更新奖励，如果动作导致越界，则返回当前状态并加上越界惩罚，得到 R。

6. 贝尔曼方程求解

(1) 封闭式求解：利用 C++矩阵库方法，求得矩阵的逆，得到$v_\pi = (I - \gamma P_\pi)^{-1} r_\pi$，其中$v_\pi$为一维向量，根据网格世界表示形式输出二维形式。

(2) 迭代式求解：利用上述求到的 R、P，计算$v_{k+1} = r_\pi + \gamma P_\pi v_k$，若$v_{k+1} - v_k > \alpha$，则将$v_{k+1}$带入上述过程继续迭代，直到满足终止条件，求得$v_\pi$。

1.2 伪码描述

初始化：网格世界 grid、策略 policy 已知，设置γ

目标：求解贝尔曼方程v_π

对每个状态 s，do

P(s’|s) = 1，当 s 在策略中到达 s’

r(s) = grid(s’)，s’为策略 policy 下 s 的下一个状态

使用数值方法求得$v_\pi = (I - \gamma P_\pi)^{-1} r_\pi$

初始化：网格世界 grid、策略 policy 已知，设置γ、α

目标：求解贝尔曼方程v_π

对网格世界中每个状态 s，do

P(s’|s) = 1，当 s 在策略中到达 s’

r(s) = grid(s’)，s’为策略 policy 下 s 的下一个状态

初始化v_0为 0 矩阵

while $||v_{k+1} - v_k|| > \alpha$，do

$v_{k+1} = r_\pi + \gamma P_\pi v_k$

1.3 时间复杂度分析

设初始网格世界为$n \times m$，令$N = n \times m$

封闭式求解：

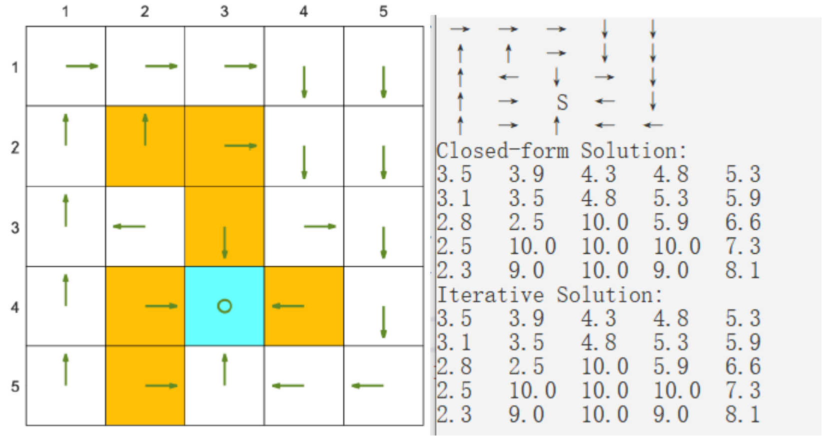
计算 P、R 为$O(N)$，矩阵乘法时间复杂度$O(N^2)$，矩阵求逆时间复杂度$O(N^3)$，综合时间复杂度$O(N^3)$。

迭代法求解：

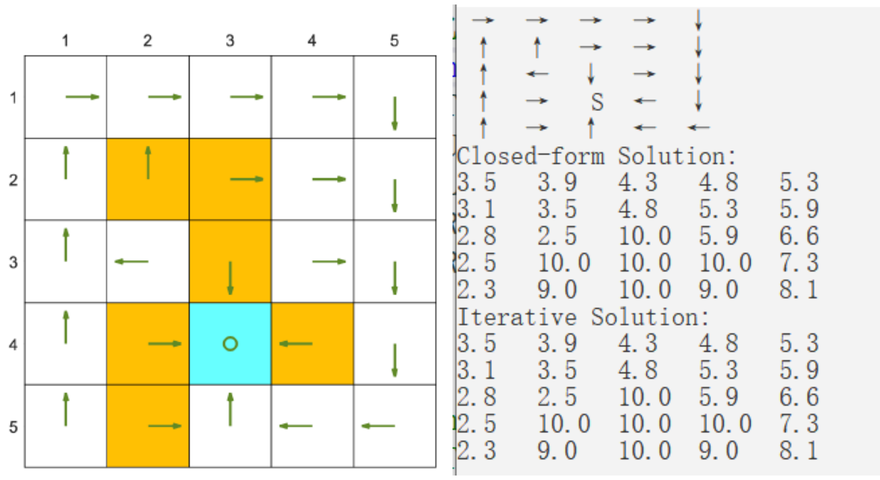
计算 P、R 为$O(N)$，矩阵乘法时间复杂度$O(N^2)$，迭代时两向量相减时间复杂度$O(N)$，设迭代次数为 K，综合时间复杂度$O(K * N^2)$。

1.4 代码运行结果

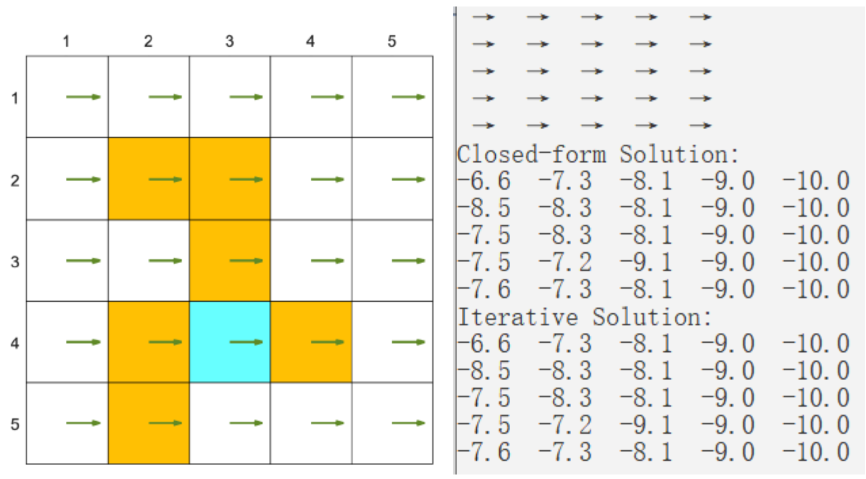
策略 1 封闭式求解与迭代式求解结果：



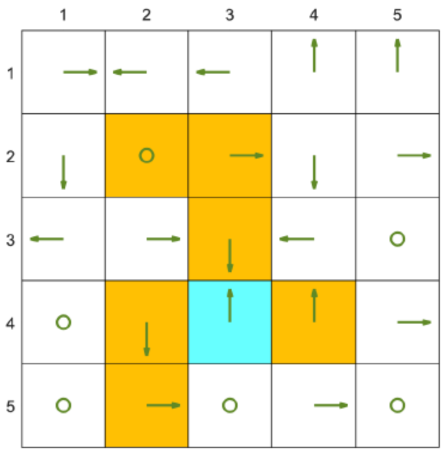
策略 2 封闭式求解与迭代式求解结果：



策略 3 封闭式求解与迭代式求解结果：



策略 4 封闭式求解与迭代式求解结果：




```
  →   ←   ←   ↑   ↑
  ↓   S   →   ↓   →
  ←   →   ↓   ←   S
  S   ↓   ↑   ↑   →
  S   →   S   →   S
Closed-form Solution:
0.0   0.0   0.0   -10.0 -10.0
-9.0  -10.0 -0.4  -0.5  -10.0
-10.0 -0.5  0.5   -0.5  0.0
0.0   -1.0  -0.5  -0.5  -10.0
0.0   0.0   0.0   0.0   0.0
Iterative Solution:
0.0   0.0   0.0   -10.0 -10.0
-9.0  -10.0 -0.4  -0.5  -10.0
-10.0 -0.5  0.5   -0.5  0.0
0.0   -1.0  -0.5  -0.5  -10.0
0.0   0.0   0.0   0.0   0.0
```

第二章 贝尔曼最优方程

2.1 设计思想

1. 网格环境定义

网格结构：使用一维向量 grid 定义网格世界，包含不同的状态和对应的奖励。状态可以是普通状态、禁区、目标或边界。

奖励设置：

OTHERSTEP：普通状态的奖励（0）。

FORBIDDEN：禁区的惩罚（-1）。

TARGET：目标状态的奖励（1）。

BORDER：越界惩罚（-1）。

设置 gamma。

2. 动作定义

定义五种可能的动作：RIGHT、DOWN、UP、LEFT、STAY。每个动作对应一个整数值。

3. 初始化奖励向量 R、状态转移矩阵 P 和状态价值 V

V：状态值向量，长度与状态个数相同，初始化为$v_i = 0$。

P：根据当前策略，生成一个状态转移矩阵，记录从每个状态转移到下一个状态的概率，初始化为 0。

R：与 V 相同大小向量，为每个状态计算对应的奖励值。

4. 状态转移与奖励计算

循环每一个状态与动作，根据当前状态和所选择的动作，计算下一个状态，设置目标状态概率为 1，得到 P；并根据越界或禁区情况更新奖励，如果动作导致越界，则返回当前状态并加上越界惩罚，得到 R。此时 P 和 R 包含所有状态与全部动作情况的结果，方便步骤 5 直接使用。

5. 贝尔曼最优方程求解

循环每一个状态 s，求 s 在每种动作 a 下的动作值$Q(s,a)=R+\gamma PV_k$。取 Q 最大的动作，更新$V_{k+1}=\max Q_k(s,a)$，若$||V_{k+1}-V_k||<\alpha$，则取得结果，并保留最优策略，若不满足，则更新V_k，继续迭代。

2.2 伪码描述

初始化：所有(s,a)概率模型已知，即 P,R。初始猜测V_0

目标：求解贝尔曼最优方程，搜索最优状态值和最优策略

对于第 k+1 次迭代，while $\left||V_{k+1}-V_k|\right|>\alpha$，do

对于每个状态 s, do

对于每个动作 a, do

计算 q 值$Q_k(s,a)=\sum p(r|s,a)r+\gamma\sum p(s'|s,a)v_k(s')$

取最大 q 值对应动作$a_k^*(s)=\arg\max_{\mathrm{a}} Q_k(s,a)$

策略更新：如果$a=a_k^*(s)$，则$\pi_{k+1}(a|s)=1$，否则$\pi_{k+1}(a|s)=0$

值更新：$\pi_{k+1}(s)=\max Q_k(a,s)$

2.3 时间复杂度分析

设初始网格世界为$n\times m$，令$N=n\times m$，每个状态动作值个数 A

求解 P、R 复杂度$O(N*A)$，迭代部分：求 Q 时间复杂度$O(N*A*N)$，其中 N 为外层循环，A 为内层循环，N 为矩阵相乘，更新 V 时间复杂度$O(N)$，迭代次数 K，相加有$O(N*A)+O(K*A*N^2)+O(K*N)$，综合时间复杂度$O(K*A*N^2)$。

2.4 结果分析

动作方向以箭头表示，保持原地不动表示为“S”

1. 使用三网格世界验证

目标+1，边界-1，其他 0，gamma=0.9，动作左、右、保持不动。计算结果如下：

图 2. 1×3 网格世界

```
Optimal State Values (V*):
10.0  10.0  10.0
Optimal Policy (Actions):
 →   S  ←
```

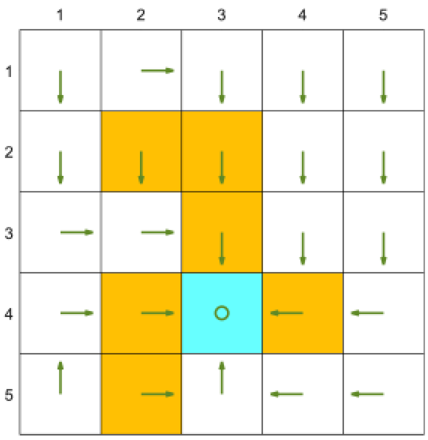
结果正确。

第三章 分析最优策略受到折扣率从γ 以及奖励设计计 r 的影响

使用 5x5 网格世界探究γ和奖励设计 r 的影响

(a) $r_{\text{boundary}} = r_{\text{forbidden}} = -1,\ r_{\text{target}} = 1,\ r_{\text{otherstep}} = 0,\ \gamma = 0.9$;

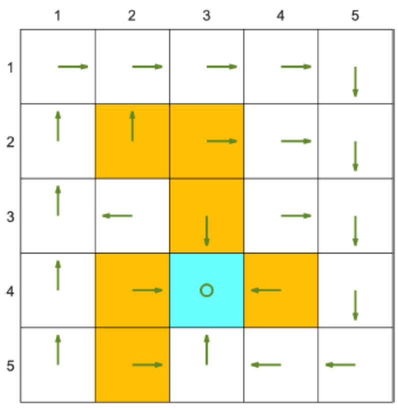
```
Optimal State Values (V*):
5.8   5.6   6.2   6.5   5.8
6.5   7.2   8.0   7.2   6.5
7.2   8.0   10.0  8.0   7.2
8.0   10.0  10.0  10.0  8.0
7.2   9.0   10.0  9.0   8.1
Optimal Policy (Actions):
 ↓    →    ↓    ↓    ↓
 ↓    ↓    ↓    ↓    ↓
 →    →    ↓    ↓    ↓
 →    →    S    ←    ←
 ↑    →    ↑    ←    ←
```



结果如图，此时最优策略勇于闯入禁区，实现高的状态值。

(b) 修改折扣率$\gamma = 0.5$，其他与(a)一致

```
Optimal State Values (V*):
0.0   0.0   0.0   0.0   0.0
0.0   0.0   0.0   0.0   0.1
0.0   0.0   2.0   0.1   0.1
0.0   2.0   2.0   2.0   0.2
0.0   1.0   2.0   1.0   0.5
Optimal Policy (Actions):
 →    →    →    →    ↓
 ↑    ↑    →    →    ↓
 ↑    ←    ↓    →    ↓
 ↑    →    S    ←    ↓
 ↑    →    ↑    ←    ←
```



此时最优策略变得短视，更注重当前回报，从而完全避开禁区，整体状态值也相比(a)更低。

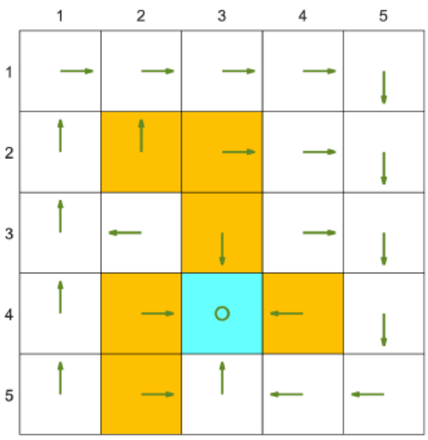
(c) 修改$\gamma = 0$，其他与(a)一致

```
Optimal State Values (V*):
0.0   0.0   0.0   0.0   0.0
0.0   0.0   0.0   0.0   0.0
0.0   0.0   1.0   0.0   0.0
0.0   1.0   1.0   1.0   0.0
0.0   0.0   1.0   0.0   0.0
Optimal Policy (Actions):
 →  →  →  →  ↓
 ↓  ↓  →  →  ↓
 →  ←  ↓  →  ↓
 ↓  →  S  ←  ↓
 ↑  →  ↑  →  ↑
```

策略变得极为短视，绝不进入禁区，仅注重即使奖励，且动作变化较大，即下一步收益相同任何动作均可以为迭代结果，有的达不到目标（如右下状态），状态值极低。

(d) $r_{forbidden} = -10$，其他与(a)一致

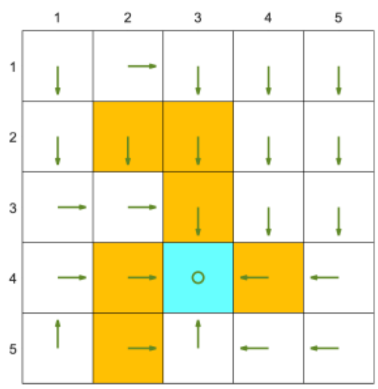
```
Optimal State Values (V*):
3.5   3.9   4.3   4.8   5.3
3.1   3.5   4.8   5.3   5.9
2.8   2.5   10.0  5.9   6.6
2.5   10.0  10.0  10.0  7.3
2.3   9.0   10.0  9.0   8.1
Optimal Policy (Actions):
 →  →  →  →  ↓
 ↑  ↑  →  →  ↓
 ↑  ←  ↓  →  ↓
 ↑  →  S  ←  ↓
 ↑  →  ↑  ←  ←
```



此时由于禁区惩罚加重，最优策略会避开禁区。

(e) $r_{\text{boundary}} = r_{\text{forbidden}} = 0$, $r_{\text{target}} = 2$, $r_{\text{otherstep}} = 1$，其他同(a)

```
Optimal State Values (V*):
15.8  15.6  16.2  16.5  15.8
16.5  17.2  18.0  17.2  16.5
17.2  18.0  20.0  18.0  17.2
18.0  20.0  20.0  20.0  18.0
17.2  19.0  20.0  19.0  18.1
Optimal Policy (Actions):
 ↓  →  ↓  ↓  ↓
 ↓  ↓  ↓  ↓  ↓
 →  →  ↓  ↓  ↓
 →  →  S  ←  ←
 ↑  →  ↑  ←  ←
```



最优策略与(a)中最优策略相同，其他设置满足(a)中条件仿射变换$r_e = ar_a + b$，

其中 a=1, b=1，因此对应最优值仿射变换有：

$$v' = av^* + \frac{b}{1-\gamma}\mathbf{1},$$

其中$\gamma = 0.9$，$v' = v^* + 10$，图中结果瞒住条件。