Chapter 4

Dynamic Programming

4.1 GridWorld

4.1.1 问题描述

和 example3.8 类似，这也是一题 GridWorld 题目，规则如下：

- 每个 cell 的初始值为 0，不存在特殊点。
- 每走一步获取的 reward 为-1.
- 若尝试往边界外面走的话，会被弹回原处，并且 reward 依然为-1.
- 左上点 (0,0) 和右下点 (3,3) 类似迷宫的出口，迭代的时候忽略这两个点。

4.1.2 问题分析

这题和上面的题目其实是一样的，只是初始化的区别不同而已。由 γ 折扣积累奖励迭代策略公式，有:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{4.1}$$

在这题中，由于在 s 下执行 a 行动的状态是确定的 (只可能选择东南西北，且状态和 $\sum_{s',r} p(s', r|s, a) = 1$)，$\gamma$ 折扣为 1，故上式可以简化为如下公式：

$$V_\pi(s) = \sum_a \pi(a|s)[R + V_\pi(s')] \tag{4.2}$$

其中 $V_\pi(s)$ 指的是需要更新的状态值函数矩阵，R 是奖励值矩阵，$V_\pi(s')$ 指的是状态值函数进行状态改变后的状态值函数矩阵。

4.1.3 实验结果

取 $\theta = 10^{-6}$, 策略迭代 302 次后，收敛得到如下矩阵：

```
after running:  302
[[  0. -14. -20. -22.]
 [-14. -18. -20. -20.]
 [-20. -20. -18. -14.]
 [-22. -20. -14.   0.]]
```

图 4.1: 实验截图

通过上图的矩阵，当机器进行决策时，它贪心地会选择当前 cell 东南西北四个相邻 cell 中的最大状态值，并且向选定的 cell 移动，若存在相同大小的状态值，则会随机地选择一个，即书中图 4.1 右边的箭头图。

代码已经上传到github。

4.3 Gambler's Problem

4.3.1 问题描述

赌徒问题的规则可以被描述如下：

- 赌徒下注猜测硬币的正反面。
- 如果硬币是正面，那么他获得他下注的金钱数量作为奖励；如果硬币是反面，那么他将失去他下注的金钱。

- 游戏停止的条件是：赌徒赢得 100 块钱或者是输光所有的金钱。
- 除了赌徒获得 100 块时 reward=1，其余时刻 reward 均为 0.
- 硬币正面向上的概率 $p_h = 0.4$。
- 求赌徒在手中金钱和最优策略之间的关系。

4.3.2 问题分析

这道题是一个多维的 gridworld 问题，假设 $v(s)$ 表示赌徒在 s 状态下的状态值，那么若该赌徒当前下注的钱数为 a，那么新的状态值为：

$$v(s) = p_h \times v(s+a) + (1-p_h) \times v(s-a) \tag{4.3}$$

其中 $v(s+a)$ 表示赌徒获胜后转移到的状态值，$v(s-a)$ 表示赌徒失败后转到的状态值，一个典型的 MDP 问题。

同理，我们设置 $\theta = 10^{-6}$ 作为停止参数，但我们得到最优状态值时，我们就可以知道是执行哪个 action 而获得的，从而解出 optimal policy。

4.3.3 实验结果

取 $\theta = 10^{-6}$，策略迭代 14 次后获得 $V(s)$ 最优策略迭代，同时可以得到最终策略。如下图所示

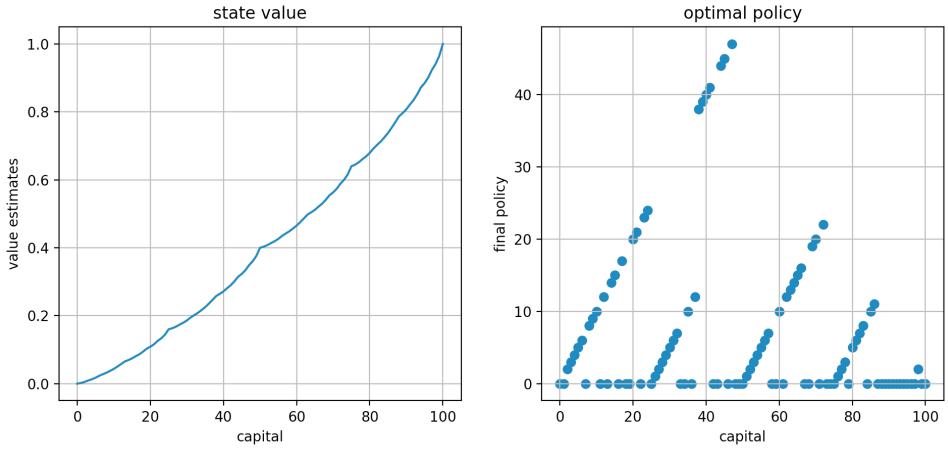

图 4.2: 实验截图

图中左边图是 reward 图，与书上一致。右图是最终策略图，不知道为什么和书上有出入，但是从理论上来说算法是没有问题的 (就是取获得最大 reward 的 action)。

代码已经上传到github。