
中山大学计算机院本科生实验报告

(2024学年秋季学期)

课程名称：强化学习与博弈论

批改人：

实验	Assignment1	专业（方向）	计算机科学与技术计科一班
学号	21307099	姓名	李英骏
Email	liyj323@mail2.sysu.edu.cn	完成日期	2024 年 11 月 19 日

目录

1	实验目的	2
2	实验过程和核心代码	2
2.1	MDP definition	2
2.2	价值迭代算法	2
2.2.1	价值函数	3
2.2.2	Bellman 最优方程	3
2.2.3	算法步骤	3
3	实验结果	4
4	实验感想	5

1 实验目的

Solve the Maze Problem using Value Iteration. Code has already been uploaded to GitHub

2 实验过程 and 核心代码

2.1 MDP definition

- S : 迷宫中的所有可行走位置。
- A : NESW四个方向。
- P : 动作导致的状态转移是确定性的。
- R : 每一步的奖励为 -1。
- γ : 设为 1.0, 因为我们希望最小化步数 (总惩罚)。

即代码如下部分:

```
21 // MDP
22 std::vector<std::pair<int, int>> states; // S
23 // location / get S
24 int getStateIndex(int x, int y)
25 {
26     for (size_t i = 0; i < states.size(); ++i) {
27         if (states[i].first == x && states[i].second == y) {
28             return i;
29         }
30     }
31     return -1;
32 }
33 // A
34 int actions[4][2] = {
35     {-1, 0}, // N
36     {0, 1},  // E
37     {1, 0},  // S
38     {0, -1} // W
39 };
40 // R
41 double reward = -1.0;
42 // gamma
43 const double GAMMA = 1.0;
```

图 1:

2.2 价值迭代算法

价值迭代算法是一种动态规划算法, 用于求解MDP的最优策略。其核心思想是通过迭代地更新每个状态的价值函数, 直到收敛到最优价值函数, 然后从中导出最优策略。

2.2.1 价值函数

- 状态价值函数 (V): 表示在状态 s 下, 按照最优策略所能获得的期望累积奖励。
- 动作价值函数 (Q): 表示在状态 s 下, 采取动作 a , 然后按照最优策略所能获得的期望累积奖励。

2.2.2 Bellman 最优方程

价值迭代算法基于 Bellman 最优方程:

$$V(s) = \max_{a \in A(s)} \left[R(s, a) + \gamma \sum_{s'} P(s'|s, a) V(s') \right]$$

其中:

- $V(s)$: 状态 s 的价值。
- $R(s, a)$: 在状态 s 下采取动作 a 获得的即时奖励。
- γ : 折扣因子。
- $P(s'|s, a)$: 从状态 s 采取动作 a 转移到状态 s' 的概率。
- $V(s')$: 下一状态 s' 的价值。

2.2.3 算法步骤

Algorithm 1 价值迭代算法

```
1: Initialize  $V(s)$  for all states  $s$ 
2: repeat
3:    $\Delta \leftarrow 0$ 
4:   for each state  $s$  do
5:      $v \leftarrow V(s)$ 
6:      $V(s) \leftarrow \max_{a \in A(s)} [R(s, a) + \gamma \sum_{s'} P(s'|s, a) V(s')]$ 
7:      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ 
8:   end for
9: until  $\Delta < \theta$ 
```

对应代码的如下部分:

```

74 while (!isValueStable) {
75     double delta = 0.0;
76     for (int s = 0; s < numStates; ++s) {
77         int x = states[s].first;
78         int y = states[s].second;
79
80         if (x == endX && y == endY) {
81             continue;
82         }
83
84         double v = V[s];
85         double maxActionValue = -std::numeric_limits<double>::infinity();
86
87         for (int a = 0; a < numActions; ++a) {
88             int newX = x + actions[a][0];
89             int newY = y + actions[a][1];
90
91             if (!isValid(newX, newY)) {
92                 newX = x;
93                 newY = y;
94             }
95
96             int sPrime = getStateIndex(newX, newY);
97             double reward = -1.0;
98             double actionValue = reward + GAMMA * V[sPrime];
99
100             if (actionValue > maxActionValue) {
101                 maxActionValue = actionValue;
102                 policy[s] = a;
103             }
104         }
105         V[s] = maxActionValue;
106         delta = std::max(delta, std::abs(v - V[s]));
107     }
108
109     if (delta < THETA) {
110         isValueStable = true;
111     }
112 }
113

```

图 2:

3 实验结果

```

Optimal Policy:
* * * * * * * *
* E E E E E S *
E N * * N * S *
* N W * * S W *
* * N W * S * *
* S * N * E S *
* E E N W * E E
* * * * * * * *

```

图 3: 输出

```

Optimal Policy:
* * * * * * * *
* E E E E E S *
E N * * N * S *
* N W * * S W *
* * N W * S * *
* S * N * E S *
* E E N W * E E
* * * * * * * *

```

图 4: 规划路线

4 实验感想

在本次实验中，我深入了解了强化学习中的价值迭代算法及其在迷宫问题中的应用。通过将理论知识与实践相结合，我不仅掌握了如何定义马尔可夫决策过程(MDP)，还实践了价值迭代算法的实现。这种算法通过不断更新状态价值函数，最终收敛到最优策略，反映了动态规划在强化学习中的重要性。