强化学习作业 01

赵天钧

学号：23210180134

2025 年 2 月 19 日

我使用策略更新方法解决本问题，采用贪心算法更新每次策略，因为数据规模不大所以收敛速度很快，在 4 次内得到了目标价值函数。下面是每次迭代后的价值函数分布。

$$\begin{pmatrix} 3.30899634 & 8.78929186 & 4.42761918 & 5.32236759 & 1.49217876 \\ 1.52158807 & 2.99231786 & 2.25013995 & 1.9075717 & 0.54740271 \\ 0.05082249 & 0.73817059 & 0.67311326 & 0.35818621 & -0.40314114 \\ -0.9735923 & -0.43549543 & -0.35488227 & -0.58560509 & -1.18307508 \\ -1.85770055 & -1.34523126 & -1.22926726 & -1.42291815 & -1.97517905 \end{pmatrix}$$

$$\begin{pmatrix} 21.97748529 & 24.4194281 & 21.97748529 & 18.4501845 & 16.60516605 \\ 19.77973676 & 21.97748529 & 19.77973676 & 16.60516605 & 14.94464945 \\ 17.80176308 & 19.77973676 & 17.80176308 & 14.94464945 & 13.4501845 \\ 16.02158677 & 17.80176308 & 16.02158677 & 13.4501845 & 12.10516605 \\ 14.4194281 & 16.02158677 & 14.4194281 & 12.10516605 & 10.89464945 \end{pmatrix}$$

$$\begin{pmatrix} 21.97748529 & 24.4194281 & 21.97748529 & 19.4194281 & 17.47748529 \\ 19.77973676 & 21.97748529 & 19.77973676 & 17.80176308 & 15.87566177 \\ 17.80176308 & 19.77973676 & 17.80176308 & 16.02158677 & 14.35376184 \\ 16.02158677 & 17.80176308 & 16.02158677 & 14.4194281 & 12.94793547 \\ 14.4194281 & 16.02158677 & 14.4194281 & 12.97748529 & 11.66643934 \end{pmatrix}$$

其更新的梯度图如下 (仅显示了始末状态，从第二次更新开始梯度关系已经保持不变)

源代码如下

```
import math
import random
import numpy as np
import networkx as nx

gamma = 0.9
reward = np.array([-1, 0, 5, 10])
p = np.zeros((5, 5, 5, 5, 4, 4))
#first two are target location, second two are beginning location, fifth and s
#initial strategy is guessing equally
strategy = 0.25 * np.ones((5, 5, 4))
#we know p fully:
#edge
for i in range(0, 5):
p[0, i, 0, i, 0, 0] = 1
p[4, i, 4, i, 2, 0] = 1
p[i, 0, i, 0, 1, 0] = 1
p[i, 4, i, 4, 3, 0] = 1
#other cases
for i in range(0, 5):
for j in range(0, 5):
if i > 0:
p[i - 1, j, i, j, 0, 1] = 1
if i < 4:
p[i + 1, j, i, j, 2, 1] = 1
if j > 0:
p[i, j - 1, i, j, 1, 1] = 1
if j < 4:
p[i, j + 1, i, j, 3, 1] = 1
#except A and B
p[0, 1, 0, 1, 0, 0] = 0
p[0, 0, 0, 1, 1, 1] = 0
p[1, 1, 0, 1, 2, 1] = 0
p[0, 2, 0, 1, 3, 1] = 0
p[0, 3, 0, 3, 0, 0] = 0
```

```
p[0, 2, 0, 3, 1, 1] = 0
p[1, 3, 0, 3, 2, 1] = 0
p[0, 4, 0, 3, 3, 1] = 0
for k in range(0, 4):
p[4, 1, 0, 1, k, 3] = 1
p[2, 3, 0, 3, k, 2] = 1
re1 = np.einsum("abk,xyabkr->xyabr", strategy, p)
re = np.einsum("xyabr,r,xy->ab", re1, reward, np.ones((5, 5))).flatten()
minus = np.zeros((5, 5, 5, 5))
for i in range(4):
minus += re1[:, :, :, :, i]
minus = np.reshape(minus, (25, 25))
eye = np.eye(25)
epi = np.linalg.solve(eye - gamma * minus.T, re).reshape((5, 5))
value = epi
print(value)
t = 1
while t < 10:
improve = np.zeros((5, 5, 4))
for i in range(4):
improve[:, :, i] += reward[i] * np.ones((5, 5))
improve[:, :, i] += gamma * value
imp1 = np.einsum("xyabkr,xyr->abk", p, improve)
maxi = np.max(imp1, axis=2, keepdims=True)
indmax = np.isclose(imp1, maxi, atol=1e-6)
counts = np.sum(indmax, axis=2, keepdims=True)
result = np.zeros((5, 5, 4))
counts = np.tile(counts, (1, 1, 4))
result[indmax] = 1 / counts[indmax]
strategy = result

re1 = np.einsum("abk,xyabkr->xyabr", strategy, p)
re = np.einsum("xyabr,r,xy->ab", re1, reward, np.ones((5, 5))).flatten()
minus = np.zeros((5, 5, 5, 5))
for i in range(4):
minus += re1[:, :, :, :, i]
minus = np.reshape(minus, (25, 25))
eye = np.eye(25)
```

```
epi = np.linalg.solve(eye - gamma * minus.T, re).reshape((5, 5))

if np.linalg.norm(value-epi) < 1e-8:
break
value = epi
print(value)
t += 1
print(t)
```