中山大学计算机学院

人工智能

本科生实验报告

（2022 学年春季学期）

课程名称：Artificial Intelligence

教学班级	信息与计算科学班	专业（方向）	信息与计算科学
学号	21311359	姓名	何凯迪

一、 实验题目

基于 Gym 库中的 Frozen Lake 环境在 vi_and_pi.py 中实现策略迭代和值迭代算法，并输出算法收敛后的路径。

实验要求 ：

1. 实现 vi_and_pi.py 中的策略评估、策略提升、策略迭代函数。
2. 实现 vi_and_pi.py 中的值迭代函数。

二、 实验内容

1. 算法原理

策略评估：策略评估的目标是估计一个策略在给定环境下的期望回报或价值函数。这个过程可以用来比较不同策略的优劣，为策略改进提供指导。

策略提升：策略提升是指在策略评估的基础上，通过改进当前策略来达到更好的性能。在强化学习中，策略提升的目标是找到一个更优的策略，使其在给定环境下的期望回报或价值函数更高。

策略迭代：策略迭代算法通过反复进行策略评估和策略改进的迭代过程，逐渐逼近最优策略。这种方法确保在每次策略改进之后都能保证性能的提升，直到找到最优策略为止。

值迭代：值迭代是一种经典的强化学习算法，用于解决马尔可夫决策过程中的最优策略问题。值迭代的目标是找到每个状态的最优值函数或最优动作值函数，从而确定最优策略。核心思想是通过迭代地更新值函数，使其逐渐逼近最优值函数。这个过程包括在每次迭代中更新值函数的估计，以及在最后从最终的值函数估计中提取最优策略。值迭代算法保证在足够的迭代次数后可以找到最优策略。

2. 伪代码

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 $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   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 $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
   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

值迭代：

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 $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

3. 关键代码展示（带注释）

```
 # 策略评价
def policy_evaluation(P, nS, nA, policy, gamma=0.9, tol=1e-3):
    # 初始化状态值函数
    value_function = np.zeros(nS)
    while True:
        delta = 0
        for s in range(nS):
            v = value_function[s]
            action = policy[s]
            # 根据给定策略和状态转移概率计算累计奖励
            new_value = 0
            for prob, next_state, reward, _ in P[s][action]:
            # 更新值函数估计值
            value_function[s] = new_value
            # 计算值函数估计值的最大变化
            delta = max(delta, abs(v - value_function[s]))
        # 判断值函数估计是否收敛
        if delta < tol:
            break
    # 返回值函数估计值
    return value_function
```

```
# 策略改进
def policy_improvement(P, nS, nA, value_from_policy, policy, gamma=0.9):
    # 创建一个新策略数组，用于存储改进后的策略
    new_policy = np.zeros(nS, dtype='int')
    for s in range(nS):
        # 存储每个动作的 Q 值
        q_values = []

        for a in range(nA):
            # 计算状态-动作对的 Q 值
            q_value = sum([p * (r + gamma * value_from_policy[s_]) for p, s_, r, _
                           in P[s][a]])

            # 将 Q 值添加到列表中
            q_values.append(q_value)

        # 根据 Q 值选择最优动作，更新策略
        new_policy[s] = np.argmax(q_values)
    # 返回改进后的策略
    return new_policy
```

```python
# 策略迭代
def policy_iteration(P, nS, nA, gamma=0.9, tol=10e-3):
    # 初始化值函数和策略
    value_function = np.zeros(nS)
    policy = np.zeros(nS, dtype=int)
    while True:
        # 策略评估
        value_function = policy_evaluation(P, nS, nA, policy, gamma, tol)
        # 策略改进
        new_policy = policy_improvement(P, nS, nA, value_function, policy, gamma)
        # 检查策略是否收敛
        if np.array_equal(policy, new_policy):
            break
        # 更新策略
        policy = new_policy
    # 返回最终的值函数和策略
    return value_function, policy
```

```python
# 值迭代
def value_iteration(P, nS, nA, gamma=0.9, tol=1e-3):
    # 初始化值函数和策略
    value_function = np.zeros(nS)
    policy = np.zeros(nS, dtype=int)
    while True:
        delta = 0
        for s in range(nS):
            v = value_function[s]
            q_values = []
            for a in range(nA):
                # 计算状态-动作对的 Q 值
                q_value = sum([p * (r + gamma * value_function[s_]) for p, s_, r, _
                               in P[s][a]])
                q_values.append(q_value)
            # 更新值函数估计值为 Q 值的最大值
            # 计算值函数估计值的最大变化
            delta = max(delta, abs(v - value_function[s]))
        # 检查值函数估计是否收敛
        if delta < tol:
            break
    for s in range(nS):
        for a in range(nA):
            # 计算状态-动作对的 Q 值
            q_value = sum([p * (r + gamma * value_function[s_]) for p, s_, r, _ in P[s][a]])
            q_values.append(q_value)
```

```
        # 选择具有最大 Q 值的动作作为策略
        policy[s] = np.argmax(q_values)
    # 返回最终的值函数和策略
    return value_function, policy
```

三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

策略迭代算法：

```
-------------------------
Beginning Policy Iteration
-------------------------

SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
Episode reward: 1.000000
```

值迭代算法：

```
-------------------------
Beginning Value Iteration
-------------------------

SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
Episode reward: 1.000000
PS D:\python文件\RL_pi_vi>
```

2. 评测指标展示及分析（机器学习实验必须有此项，其它可分析运行时间等）

```
policy time: 0.0019993782043457031
value time: 0.0
```

可以看到运算时间极短，性能较优。

四、 参考资料

价值迭代和策略迭代：

•https://www.bilibili.com/video/BV1bd4y1z7nT/?vd_source=fe8a61bc42546259ff5da93db830dc1a
•https://www.bilibili.com/video/BV1at4y1P7P4/?spm_id_from=333.788&vd_source=fe8a61bc42546259ff5da93db830dc1a
•https://zhuanlan.zhihu.com/p/32311522?refer=mlexplained