强化学习基础篇（十二）策略评估算法在FrozenLake中的实现

本节将主要基于gym环境中的FrozenLake-v0进行策略评估算法的实现。

1. 迭代策略评估算法的伪代码

迭代策略评估算法，用于估计$V = v_\pi$

输入待评估的策略π

算法参数：小阈值$\theta > 0$，用于确定估计量的精度

对于任意$s \in S^+$，任意初始化$V(s)$,其中$V(\text{终止状态}) = 0$

循环：

$\Delta \leftarrow 0$

对每一个$s \in S$循环：

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|$

直到$\Delta < \theta$

2. FrozenLake-v0环境

FrozenLake环境是一个GridWorld环境，名字是指在一块冰面上有四种state：

S: initial stat 起点

F: frozen lake 冰湖

H: hole 窟窿

G: the goal 目的地

智能体要学会从起点走到目的地，并且不要掉进窟窿。

首先我们调用 FrozenLake-v0环境:

```
# 导入库信息
import numpy as np
import gym
# 调用环境
env=gym.make("FrozenLake-v0")
```

环境可视化

```
# 查看当前状态
env.render()
```

运行结果为：

```
SFFF
FHFH
FFFH
HFFG
```

查看环境的观测空间：

```
# 查看观测空间
print(env.observation_space,env.nS)
```

运行结果为：

```
Discrete(16) 16
```

查看环境的动作空间：

```
# 查看动作空间
print(env.action_space,env.nA)

```

运行结果为：

```
Discrete(4) 4
```

动作的定义为：

```
LEFT = 0
DOWN = 1
RIGHT = 2
UP = 3
```

转移概率

使用动态规划算法需要直到环境的所有信息，即转移概率，可以通过env.P查看环境的所有转移概率：

P[][]本质上是一个“二维数组”，状态和动作分别由数字0-15和0-3表示。$P[state][action]$存储的是，在状态s下采取动作a获得的一系列数据，即(转移概率，下一步状态，奖励，完成标志)这样的元组。

```
# 查看环境转移矩阵
print(env.P)
```

运行结果为：

```
{
    0: {
        0: [(0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 4, 0.0, False)],
        1: [(0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 4, 0.0, False), (0.3333333333333333, 1, 0.0, False)],
        2: [(0.3333333333333333, 4, 0.0, False), (0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 0, 0.0, False)],
        3: [(0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 0, 0.0, False)]
    },
    1: {
        0: [(0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 5, 0.0, True)],
        1: [(0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 5, 0.0, True), (0.3333333333333333, 2, 0.0, False)],
        2: [(0.3333333333333333, 5, 0.0, True), (0.3333333333333333, 2, 0.0, False), (0.3333333333333333, 1, 0.0, False)],
        3: [(0.3333333333333333, 2, 0.0, False), (0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 0, 0.0, False)]
    },
    2: {
        0: [(0.3333333333333333, 2, 0.0, False), (0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 6, 0.0, False)],
        1: [(0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 6, 0.0, False), (0.3333333333333333, 3, 0.0, False)],
        2: [(0.3333333333333333, 6, 0.0, False), (0.3333333333333333, 3, 0.0, False), (0.3333333333333333, 2, 0.0, False)],
        3: [(0.3333333333333333, 3, 0.0, False), (0.3333333333333333, 2, 0.0, False), (0.3333333333333333, 1, 0.0, False)]
    },
    3: {
        0: [(0.3333333333333333, 3, 0.0, False), (0.3333333333333333, 2, 0.0, False), (0.3333333333333333, 7, 0.0, True)],
        1: [(0.3333333333333333, 2, 0.0, False), (0.3333333333333333, 7, 0.0, True), (0.3333333333333333, 3, 0.0, False)],
        2: [(0.3333333333333333, 7, 0.0, True), (0.3333333333333333, 3, 0.0, False), (0.3333333333333333, 3, 0.0, False)],
        3: [(0.3333333333333333, 3, 0.0, False), (0.3333333333333333, 3, 0.0, False), (0.3333333333333333, 2, 0.0, False)]
    },
    4: {
        0: [(0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 4, 0.0, False), (0.3333333333333333, 8, 0.0, False)],
```

```
        1: [(0.3333333333333333, 4, 0.0, False), (0.3333333333333333, 8,
0.0, False), (0.3333333333333333, 5, 0.0, True)],
        2: [(0.3333333333333333, 8, 0.0, False), (0.3333333333333333, 5,
0.0, True), (0.3333333333333333, 0, 0.0, False)],
        3: [(0.3333333333333333, 5, 0.0, True), (0.3333333333333333, 0, 0.0,
False), (0.3333333333333333, 4, 0.0, False)]
    },
    5: {
        0: [(1.0, 5, 0, True)],
        1: [(1.0, 5, 0, True)],
        2: [(1.0, 5, 0, True)],
        3: [(1.0, 5, 0, True)]
    },
    6: {
        0: [(0.3333333333333333, 2, 0.0, False), (0.3333333333333333, 5,
0.0, True), (0.3333333333333333, 10, 0.0, False)],
        1: [(0.3333333333333333, 5, 0.0, True), (0.3333333333333333, 10,
0.0, False), (0.3333333333333333, 7, 0.0, True)],
        2: [(0.3333333333333333, 10, 0.0, False), (0.3333333333333333, 7,
0.0, True), (0.3333333333333333, 2, 0.0, False)],
        3: [(0.3333333333333333, 7, 0.0, True), (0.3333333333333333, 2, 0.0,
False), (0.3333333333333333, 5, 0.0, True)]
    },
    7: {
        0: [(1.0, 7, 0, True)],
        1: [(1.0, 7, 0, True)],
        2: [(1.0, 7, 0, True)],
        3: [(1.0, 7, 0, True)]
    },
    8: {
        0: [(0.3333333333333333, 4, 0.0, False), (0.3333333333333333, 8,
0.0, False), (0.3333333333333333, 12, 0.0, True)],
        1: [(0.3333333333333333, 8, 0.0, False), (0.3333333333333333, 12,
0.0, True), (0.3333333333333333, 9, 0.0, False)],
        2: [(0.3333333333333333, 12, 0.0, True), (0.3333333333333333, 9,
0.0, False), (0.3333333333333333, 4, 0.0, False)],
        3: [(0.3333333333333333, 9, 0.0, False), (0.3333333333333333, 4,
0.0, False), (0.3333333333333333, 8, 0.0, False)]
    },
    9: {
        0: [(0.3333333333333333, 5, 0.0, True), (0.3333333333333333, 8, 0.0,
False), (0.3333333333333333, 13, 0.0, False)],
        1: [(0.3333333333333333, 8, 0.0, False), (0.3333333333333333, 13,
0.0, False), (0.3333333333333333, 10, 0.0, False)],
        2: [(0.3333333333333333, 13, 0.0, False), (0.3333333333333333, 10,
0.0, False), (0.3333333333333333, 5, 0.0, True)],
        3: [(0.3333333333333333, 10, 0.0, False), (0.3333333333333333, 5,
0.0, True), (0.3333333333333333, 8, 0.0, False)]
    },
    10: {
        0: [(0.3333333333333333, 6, 0.0, False), (0.3333333333333333, 9,
0.0, False), (0.3333333333333333, 14, 0.0, False)],
        1: [(0.3333333333333333, 9, 0.0, False), (0.3333333333333333, 14,
0.0, False), (0.3333333333333333, 11, 0.0, True)],
        2: [(0.3333333333333333, 14, 0.0, False), (0.3333333333333333, 11,
0.0, True), (0.3333333333333333, 6, 0.0, False)],
        3: [(0.3333333333333333, 11, 0.0, True), (0.3333333333333333, 6,
0.0, False), (0.3333333333333333, 9, 0.0, False)]
```

```
    },
    11: {
        0: [(1.0, 11, 0, True)],
        1: [(1.0, 11, 0, True)],
        2: [(1.0, 11, 0, True)],
        3: [(1.0, 11, 0, True)]
    },
    12: {
        0: [(1.0, 12, 0, True)],
        1: [(1.0, 12, 0, True)],
        2: [(1.0, 12, 0, True)],
        3: [(1.0, 12, 0, True)]
    },
    13: {
        0: [(0.3333333333333333, 9, 0.0, False), (0.3333333333333333, 12, 0.0, True), (0.3333333333333333, 13, 0.0, False)],
        1: [(0.3333333333333333, 12, 0.0, True), (0.3333333333333333, 13, 0.0, False), (0.3333333333333333, 14, 0.0, False)],
        2: [(0.3333333333333333, 13, 0.0, False), (0.3333333333333333, 14, 0.0, False), (0.3333333333333333, 9, 0.0, False)],
        3: [(0.3333333333333333, 14, 0.0, False), (0.3333333333333333, 9, 0.0, False), (0.3333333333333333, 12, 0.0, True)]
    },
    14: {
        0: [(0.3333333333333333, 10, 0.0, False), (0.3333333333333333, 13, 0.0, False), (0.3333333333333333, 14, 0.0, False)],
        1: [(0.3333333333333333, 13, 0.0, False), (0.3333333333333333, 14, 0.0, False), (0.3333333333333333, 15, 1.0, True)],
        2: [(0.3333333333333333, 14, 0.0, False), (0.3333333333333333, 15, 1.0, True), (0.3333333333333333, 10, 0.0, False)],
        3: [(0.3333333333333333, 15, 1.0, True), (0.3333333333333333, 10, 0.0, False), (0.3333333333333333, 13, 0.0, False)]
    },
    15: {
        0: [(1.0, 15, 0, True)],
        1: [(1.0, 15, 0, True)],
        2: [(1.0, 15, 0, True)],
        3: [(1.0, 15, 0, True)]
    }
}
```

3.策略评估源代码

```python
import numpy as np
import gym

def policy_eval(enviroment,policy,discount_factor=1.0,theta=0.1):
    # 引用环境
     env = enviroment

    # 初始化值函数
     V = np.zeros(env.nS)

    # 开始迭代
     for _ in range(500):
```

```
        delta = 0
        # 扫描所有状态
        for s in range(env.nS):
            v=0
            # 扫描动作空间
            for a,action_prob in enumerate(policy[s]):
                # 扫描下一状态
                for prob,next_state,reward,done in env.P[s][a]:
                    # 更新值函数
                    v += action_prob * prob * ( reward + discount_factor * V[next_state])
            # 更新最大的误差值
            delta=max(delta,np.abs(v-V[s]))
            V[s] =v

        if delta < theta:
            break
    return np.array(V)

# 定义策略生成函数
def generate_policy(env,input_policy):
    policy=np.zeros([env.nS,env.nA])
    for _ , x in enumerate(input_policy):
        policy[_][x] = 1
    return policy


if __name__=="__main__":
    # 创建环境
    env=gym.make("FrozenLake-v0")
    # 定义动作策略
    input_policy=[2,1,2,3,2,0,2,0,1,2,2,0,0,1,1,0] # 定义了在每个状态采取的动作，LEFT = 0、DOWN = 1、RIGHT = 2、UP = 3
    # 生成策略
    policy=generate_policy(env,input_policy)
    Value=policy_eval(env,policy)
    print("This is the final value:\n")
    print(Value.reshape([4,4]))
```

运行结果为：

```
This is the final value:

[[0.         0.         0.         0.        ]
 [0.         0.         0.03703704 0.        ]
 [0.         0.07407407 0.17283951 0.        ]
 [0.         0.19753086 0.55967078 0.        ]]
```

4. 代码解析

首先我们会定义策略生成函数

```python
# 定义策略生成函数
def generate_policy(env,input_policy):
    policy=np.zeros([env.nS,env.nA])
    for _ , x in enumerate(input_policy):
        policy[_][x] = 1
    return policy
```

该函数会生成一个[env.nS,env.nA]大小的数组，然后根据输入的每个状态的策略生成一个矩阵，将该状态的某状态置为1。

例如这里我们要评估策略：

```python
input_policy=[2,1,2,3,2,0,2,0,1,2,2,0,0,1,1,0] # 定义了在每个状态采取的动作，LEFT = 0、DOWN = 1、RIGHT = 2、UP = 3
```

生成的策略矩阵如下所示：

```
array([[0., 0., 1., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 1., 0.],
       [1., 0., 0., 0.],
       [0., 0., 1., 0.],
       [1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 1., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [1., 0., 0., 0.]])
```

在迭代过程中完全按照公式$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r+\gamma V(s')]$进行。

历史文章链接：

- 强化学习基础篇（十一）迷宫环境搭建
- 强化学习基础篇（十）OpenAI Gym环境汇总
- 强化学习基础篇（九）OpenAI Gym基础介绍
- 强化学习基础篇（八）动态规划扩展
- 强化学习基础篇（七）动态规划之价值迭代
- 强化学习基础篇（六）动态规划之策略迭代（2）
- 强化学习基础篇（五）动态规划之策略迭代（1）
- 强化学习基础篇（四）动态规划之迭代策略评估
- 强化学习基础篇（三）动态规划之基础介绍
- 强化学习基础篇（二）马尔科夫决策过程（MDP）
- 强化学习基础篇（一）强化学习入门
- 9.进一步讨论Policy Gradients方法
- 8. DRL中的Q-Function
- 7. 值函数方法（Value Function Methods）
- 6. Actor-Critic算法

- 5. 策略梯度（Policy Gradients）
- 4.强化学习简介
- 3.TensorFlow示例
- 2.模仿学习（Imitation Learning）
- 1.深度强化学习简介
- A survey on value-based deep reinforcement learning
- Chinese Stock Prediction Using Deep Neural Network
- Differential Dynamics of the Maternal Immune System阅读笔记
- bib如何生成author-year格式的bbl的问题
- 如何使用GPU运行TensorFlow（Win10）
- 通过frp实现内网穿透
- 关于t-SNE降维方法
- 系统评价与Meta分析基础
- Meta分析入门工具介绍
- 使用ruptures检测变量关系
- 如何读取rda格式数据