

# 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  $\pi$

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：

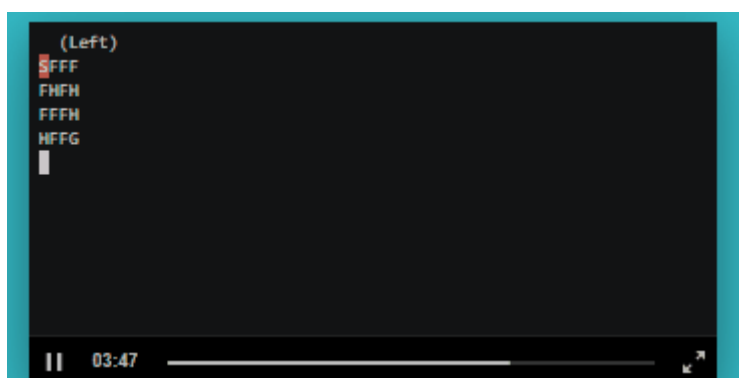
S: initial stat 起点

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：

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

## 环境可视化

```
1 # 查看当前状态
2 env.render()
```

运行结果为：

```
1 SFFF
2 FHFH
3 FFFH
4 HFFG
```

## 查看环境的观测空间：

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

运行结果为：

```
1 Discrete(16) 16
```

## 查看环境的动作空间：

```
1 # 查看动作空间
2 print(env.action_space,env.nA)
3
```

运行结果为：

```
1 Discrete(4) 4
```

动作的定义为：

```
1 LEFT = 0
2 DOWN = 1
3 RIGHT = 2
4 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。

```
1 # 查看环境转移矩阵
2 print(env.P)
```

运行结果为：

```
1 {
2     0: {
3         0: [(0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 0,
4         0.0, False), (0.3333333333333333, 4, 0.0, False)],
5         1: [(0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 4,
6         0.0, False), (0.3333333333333333, 1, 0.0, False)],
7         2: [(0.3333333333333333, 4, 0.0, False), (0.3333333333333333, 1,
8         0.0, False), (0.3333333333333333, 0, 0.0, False)],
9         3: [(0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 0,
10        0.0, False), (0.3333333333333333, 0, 0.0, False)]
11    },
12    1: {
13        0: [(0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 0,
14        0.0, False), (0.3333333333333333, 5, 0.0, True)],
15        1: [(0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 5,
16        0.0, True), (0.3333333333333333, 2, 0.0, False)],
17        2: [(0.3333333333333333, 5, 0.0, True), (0.3333333333333333, 2, 0.0,
18        False), (0.3333333333333333, 1, 0.0, False)],
19        3: [(0.3333333333333333, 2, 0.0, False), (0.3333333333333333, 1,
20        0.0, False), (0.3333333333333333, 0, 0.0, False)]
21    },
22    2: {
23        0: [(0.3333333333333333, 2, 0.0, False), (0.3333333333333333, 1,
24        0.0, False), (0.3333333333333333, 6, 0.0, False)],
25        1: [(0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 6,
26        0.0, False), (0.3333333333333333, 3, 0.0, False)],
27        2: [(0.3333333333333333, 6, 0.0, False), (0.3333333333333333, 3,
28        0.0, False), (0.3333333333333333, 2, 0.0, False)],
29        3: [(0.3333333333333333, 3, 0.0, False), (0.3333333333333333, 2,
30        0.0, False), (0.3333333333333333, 1, 0.0, False)]
31    },
32    3: {
33        0: [(0.3333333333333333, 3, 0.0, False), (0.3333333333333333, 2,
34        0.0, False), (0.3333333333333333, 7, 0.0, True)],
35        1: [(0.3333333333333333, 2, 0.0, False), (0.3333333333333333, 7,
36        0.0, True), (0.3333333333333333, 3, 0.0, False)],
37        2: [(0.3333333333333333, 7, 0.0, True), (0.3333333333333333, 3, 0.0,
38        False), (0.3333333333333333, 3, 0.0, False)],
39        3: [(0.3333333333333333, 3, 0.0, False), (0.3333333333333333, 3,
40        0.0, False), (0.3333333333333333, 2, 0.0, False)]
41    },
42    4: {
43        0: [(0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 4,
44        0.0, False), (0.3333333333333333, 8, 0.0, False)],
```

```
28     1: [(0.3333333333333333, 4, 0.0, False), (0.3333333333333333, 8,
29     0.0, False), (0.3333333333333333, 5, 0.0, True)],
30     2: [(0.3333333333333333, 8, 0.0, False), (0.3333333333333333, 5,
31     0.0, True), (0.3333333333333333, 0, 0.0, False)],
32     3: [(0.3333333333333333, 5, 0.0, True), (0.3333333333333333, 0, 0.0,
33     False), (0.3333333333333333, 4, 0.0, False)]
34 },
35 5: {
36     0: [(1.0, 5, 0, True)],
37     1: [(1.0, 5, 0, True)],
38     2: [(1.0, 5, 0, True)],
39     3: [(1.0, 5, 0, True)]
40 },
41 6: {
42     0: [(0.3333333333333333, 2, 0.0, False), (0.3333333333333333, 5,
43     0.0, True), (0.3333333333333333, 10, 0.0, False)],
44     1: [(0.3333333333333333, 5, 0.0, True), (0.3333333333333333, 10,
45     0.0, False), (0.3333333333333333, 7, 0.0, True)],
46     2: [(0.3333333333333333, 10, 0.0, False), (0.3333333333333333, 7,
47     0.0, True), (0.3333333333333333, 2, 0.0, False)],
48     3: [(0.3333333333333333, 7, 0.0, True), (0.3333333333333333, 2, 0.0,
49     False), (0.3333333333333333, 5, 0.0, True)]
50 },
51 7: {
52     0: [(1.0, 7, 0, True)],
53     1: [(1.0, 7, 0, True)],
54     2: [(1.0, 7, 0, True)],
55     3: [(1.0, 7, 0, True)]
56 },
57 8: {
58     0: [(0.3333333333333333, 4, 0.0, False), (0.3333333333333333, 8,
59     0.0, False), (0.3333333333333333, 12, 0.0, True)],
60     1: [(0.3333333333333333, 8, 0.0, False), (0.3333333333333333, 12,
61     0.0, True), (0.3333333333333333, 9, 0.0, False)],
62     2: [(0.3333333333333333, 12, 0.0, True), (0.3333333333333333, 9,
63     0.0, False), (0.3333333333333333, 4, 0.0, False)],
64     3: [(0.3333333333333333, 9, 0.0, False), (0.3333333333333333, 4,
65     0.0, False), (0.3333333333333333, 8, 0.0, False)]
66 },
67 9: {
68     0: [(0.3333333333333333, 5, 0.0, True), (0.3333333333333333, 8, 0.0,
69     False), (0.3333333333333333, 13, 0.0, False)],
70     1: [(0.3333333333333333, 8, 0.0, False), (0.3333333333333333, 13,
71     0.0, False), (0.3333333333333333, 10, 0.0, False)],
72     2: [(0.3333333333333333, 13, 0.0, False), (0.3333333333333333, 10,
73     0.0, False), (0.3333333333333333, 5, 0.0, True)],
74     3: [(0.3333333333333333, 10, 0.0, False), (0.3333333333333333, 5,
75     0.0, True), (0.3333333333333333, 8, 0.0, False)]
76 },
77 10: {
78     0: [(0.3333333333333333, 6, 0.0, False), (0.3333333333333333, 9,
79     0.0, False), (0.3333333333333333, 14, 0.0, False)],
80     1: [(0.3333333333333333, 9, 0.0, False), (0.3333333333333333, 14,
81     0.0, False), (0.3333333333333333, 11, 0.0, True)],
82     2: [(0.3333333333333333, 14, 0.0, False), (0.3333333333333333, 11,
83     0.0, True), (0.3333333333333333, 6, 0.0, False)],
84     3: [(0.3333333333333333, 11, 0.0, True), (0.3333333333333333, 6,
85     0.0, False), (0.3333333333333333, 9, 0.0, False)]
86 }
```

```

67     },
68     11: {
69         0: [(1.0, 11, 0, True)],
70         1: [(1.0, 11, 0, True)],
71         2: [(1.0, 11, 0, True)],
72         3: [(1.0, 11, 0, True)]
73     },
74     12: {
75         0: [(1.0, 12, 0, True)],
76         1: [(1.0, 12, 0, True)],
77         2: [(1.0, 12, 0, True)],
78         3: [(1.0, 12, 0, True)]
79     },
80     13: {
81         0: [(0.3333333333333333, 9, 0.0, False), (0.3333333333333333, 12,
0.0, True), (0.3333333333333333, 13, 0.0, False)],
82         1: [(0.3333333333333333, 12, 0.0, True), (0.3333333333333333, 13,
0.0, False), (0.3333333333333333, 14, 0.0, False)],
83         2: [(0.3333333333333333, 13, 0.0, False), (0.3333333333333333, 14,
0.0, False), (0.3333333333333333, 9, 0.0, False)],
84         3: [(0.3333333333333333, 14, 0.0, False), (0.3333333333333333, 9,
0.0, False), (0.3333333333333333, 12, 0.0, True)]
85     },
86     14: {
87         0: [(0.3333333333333333, 10, 0.0, False), (0.3333333333333333, 13,
0.0, False), (0.3333333333333333, 14, 0.0, False)],
88         1: [(0.3333333333333333, 13, 0.0, False), (0.3333333333333333, 14,
0.0, False), (0.3333333333333333, 15, 1.0, True)],
89         2: [(0.3333333333333333, 14, 0.0, False), (0.3333333333333333, 15,
1.0, True), (0.3333333333333333, 10, 0.0, False)],
90         3: [(0.3333333333333333, 15, 1.0, True), (0.3333333333333333, 10,
0.0, False), (0.3333333333333333, 13, 0.0, False)]
91     },
92     15: {
93         0: [(1.0, 15, 0, True)],
94         1: [(1.0, 15, 0, True)],
95         2: [(1.0, 15, 0, True)],
96         3: [(1.0, 15, 0, True)]
97     }
98 }

```

### 3.策略评估源代码

```

1  import numpy as np
2  import gym
3
4  def policy_eval(enviroment, policy, discount_factor=1.0, theta=0.1):
5      # 引用环境
6      env = enviroment
7
8      # 初始化值函数
9      v = np.zeros(env.ns)
10
11     # 开始迭代
12     for _ in range(500):

```

```

13     delta = 0
14     # 扫描所有状态
15     for s in range(env.nS):
16         v=0
17         # 扫描动作空间
18         for a,action_prob in enumerate(policy[s]):
19             # 扫描下一状态
20             for prob,next_state,reward,done in env.P[s][a]:
21                 # 更新值函数
22                 v += action_prob * prob * ( reward + discount_factor *
v[next_state])
23                 # 更新最大的误差值
24                 delta=max(delta,np.abs(v-v[s]))
25                 v[s] =v
26
27         if delta < theta:
28             break
29     return np.array(v)
30
31 # 定义策略生成函数
32 def generate_policy(env,input_policy):
33     policy=np.zeros([env.nS,env.nA])
34     for _ , x in enumerate(input_policy):
35         policy[_][x] = 1
36     return policy
37
38
39 if __name__=="__main__":
40     # 创建环境
41     env=gym.make("FrozenLake-v0")
42     # 定义动作策略
43     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
44     # 生成策略
45     policy=generate_policy(env,input_policy)
46     value=policy_eval(env,policy)
47     print("This is the final value:\n")
48     print(value.reshape([4,4]))

```

运行结果为：

```

1 This is the final value:
2
3 [[0.          0.          0.          0.          ]
4  [0.          0.          0.03703704 0.          ]
5  [0.          0.07407407 0.17283951 0.          ]
6  [0.          0.19753086 0.55967078 0.          ]]

```

## 4. 代码解析

首先我们会定义策略生成函数

```

1 # 定义策略生成函数
2 def generate_policy(env,input_policy):
3     policy=np.zeros([env.nS,env.nA])
4     for _ , x in enumerate(input_policy):
5         policy[_][x] = 1
6     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：

```

1 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：

```

1 array([[0., 0., 1., 0.],
2        [0., 1., 0., 0.],
3        [0., 0., 1., 0.],
4        [0., 0., 0., 1.],
5        [0., 0., 1., 0.],
6        [1., 0., 0., 0.],
7        [0., 0., 1., 0.],
8        [1., 0., 0., 0.],
9        [0., 1., 0., 0.],
10       [0., 0., 1., 0.],
11       [0., 0., 1., 0.],
12       [1., 0., 0., 0.],
13       [1., 0., 0., 0.],
14       [0., 1., 0., 0.],
15       [0., 1., 0., 0.],
16       [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](#)