



实验二RL报告

**江辉--171250506**

**2020-3-31**

**南京大学**

**软件学院**

目录

[一、 实验目的 1](#_Toc36549859)

[二、 内容描述 1](#_Toc36549860)

[1、 项目目录描述 1](#_Toc36549861)

[三、 运行用例和截图 1](#_Toc36549862)

[1、 输出的q表和学习过程 1](#_Toc36549863)

[四、 Q-Learning伪代码描述 2](#_Toc36549864)

[五、 过程分析描述 2](#_Toc36549865)

[1、 超参数选择 2](#_Toc36549866)

[2、 Q table设计 3](#_Toc36549867)

[3、 epsilon-贪婪策略 3](#_Toc36549868)

[4、 回报R值定义 4](#_Toc36549869)

[5、 Q值更新过程 4](#_Toc36549870)

[6、 Q表分析 5](#_Toc36549871)

[六、 实验中出现的问题和相应的解决办法 5](#_Toc36549872)

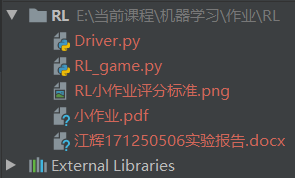
# 实验目的

本次实验使用python构建一个Tabular Q-learning算法完成完成寻宝游戏，了解Q-learning过程

# 内容描述

本程序用python编写，手写Q-learning算法的实现。预先定义好游戏规则和各个超参数，Q table来记录并预测行动和回报，使用ε-greedy策略来选择动作。

## 项目目录描述

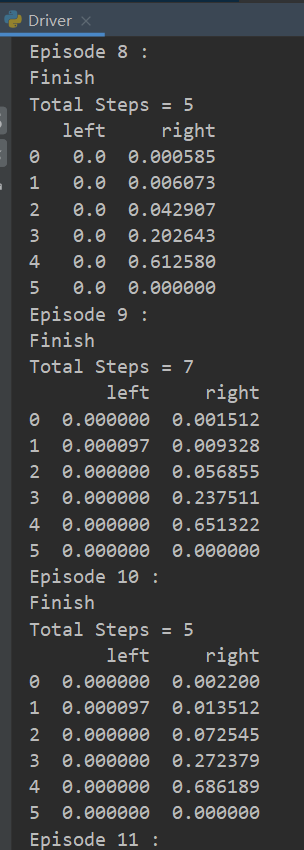
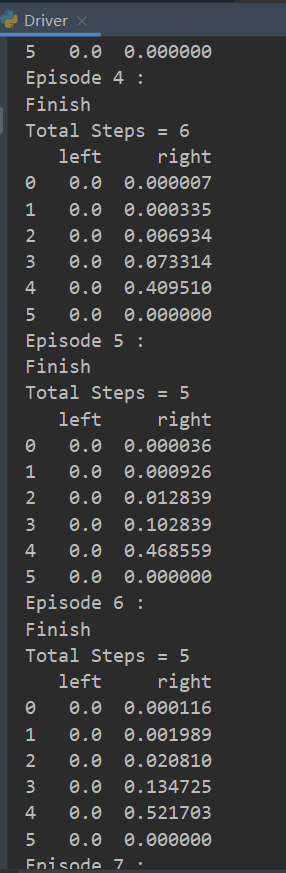
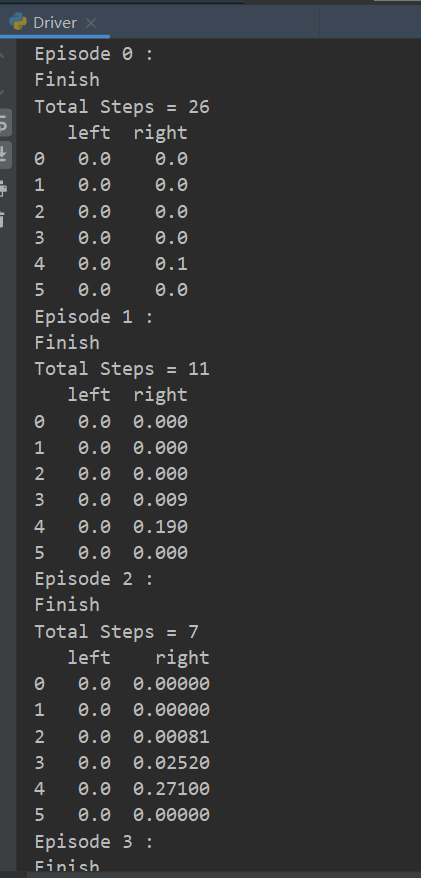
本项目推荐使用pycharm打开

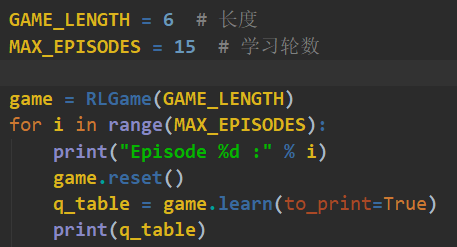
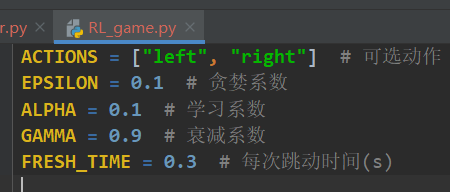
**Driver.py**里写了main函数和几个对照试验的参数，我把对应的对照试验都封装成函数了，可以直接在main函数里调用运行

**RL\_game.py**里写了RL model的类，所有模型相关的函数和参数都以注释的形式写在这个文件里

# 运行用例和截图

## 输出的q表和学习过程

右图为游戏地图长度为6，贪婪系数ε=0.1，α=0.1，γ=0.9的结果。可以看到，基本上在第5次学习之后基本就得到了目标的Q table，已经可以确定是往右走了，此后除了随机选择到往左走外，都会选择往右走。可以看到每一轮后Q表都有更新，且其中采取right动作的预估值远高于采取left动作的预估值(如果ε-贪婪策略没有随机到采取往左走就不会增加，而由于90%的概率选取最高估值动作，所以一旦学习到right，就很少选择往左走)



# Q-Learning伪代码描述

**Initialize Q(s,a) arbitrarily：即初始化Q表(state-action的估值表)**

**Repeat ( for each episode)：每一回合从起点开始，到终点结束**

**Initialize stete s**

**Repeat ( for each step of episode)：回合中每一步都要进行学习**

**// 在当前状态和环境下先通过ε-贪婪策略从Q table中选取一个动作**

**Choose action a from s using policy derived from Q (e.g. ε-greedy)**

**// 预先计算采取动作后的回报和下一状态**

**Take action a, observe reward r, next state s’**

**// 下式右边的Q(s,a)是当前Q table中当前状态下采取动作a的预估价值**

**// maxQ(s’,a’)是采取动作a之后的新状态下的最优动作的价值，即预判一步**

**// γ是代表预判的价值是不确定的，需要打折扣**

**// r是采取动作a获得的实际价值**

**// r + maxQ(s’,a’)得到采取动作a的实际价值，减去Q(s,a)获得矫正值**

**Q( s, a ) ← Q( s, a ) + α[ r + γ maxa’Q(s’,a’) – Q(s,a)]**

**//上面以α的速度更新了Q( s, a )后，下面就是切换状态并准备进行下一步**

**s ← s’**

**until s is terminal state**

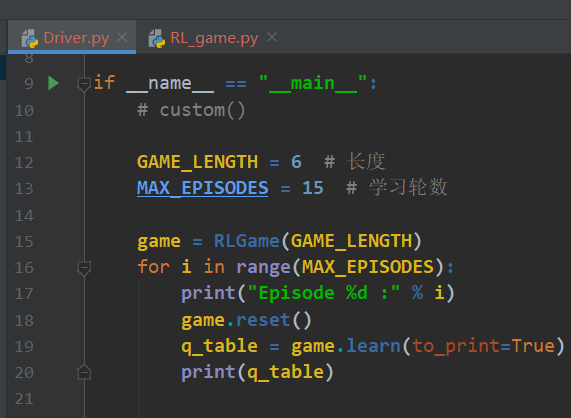
# 过程分析描述

## 超参数选择

本次实验中画布的长度=6，最大学习轮数=15，

贪婪系数ε=0.1，学习速率α=0.1，衰减系数γ=0.9，每一步状态切换用时0.3s

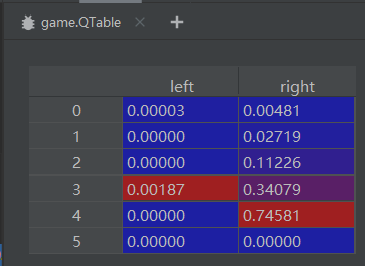
这些超参数都可以在代码中手动设置切换



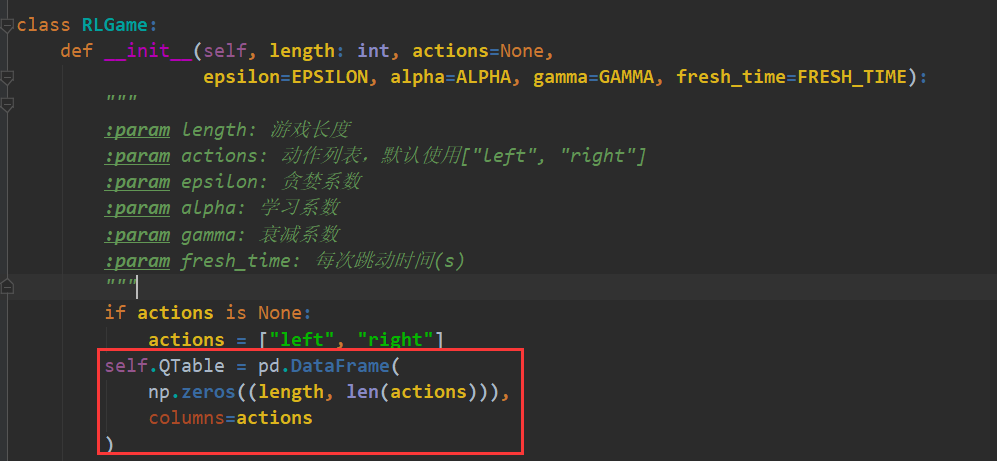


## Q table设计

本次实验有6个状态，每个状态都有left和right两个动作可选，因此Q table为如下的形式



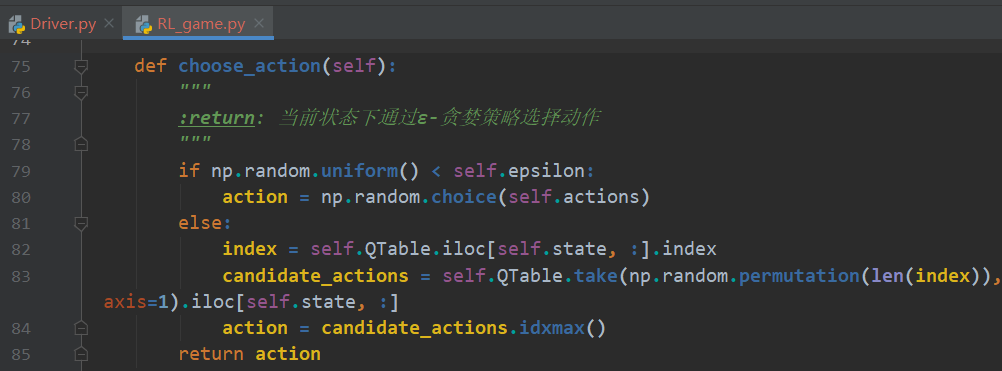
相应的初始化语句如下：



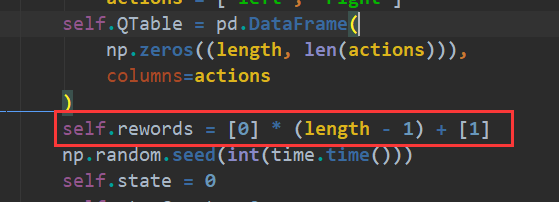
## epsilon-贪婪策略

下面ε=0.1，首先生成随机数，判断是否小于ε，小于时，采取随机动作

大于时，选择最高预估值的动作进行选择，为了避免在多个最高价值动作时只选择前面的，我打乱了index来随机选取最大值标签



## 回报R值定义

终点的回报值为1，其余每一个位置的回报值均设置为0，每次预估观测时同时返回对应回报

## Q值更新过程

每次迭代的每一步都需要进行Q值学习

首先是根据当前状态和环境选择一个动作action，

再从Q表中获取当前状态下采取action的预估价值q\_predict

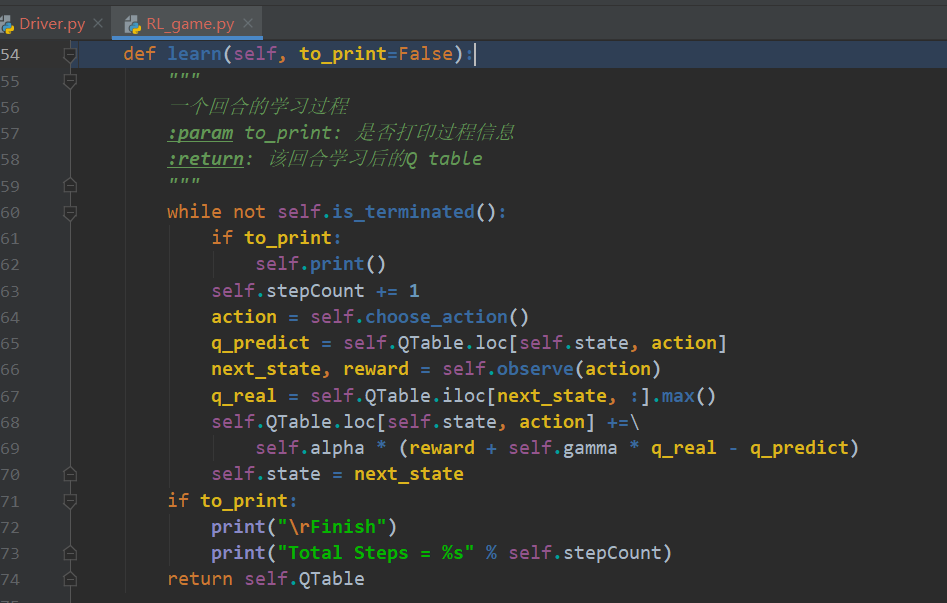
预判采取动作action后的下一个状态next\_state和获得的回报值reward。

从Q表中获取next\_state状态下的能获得的最大预估价值q\_real

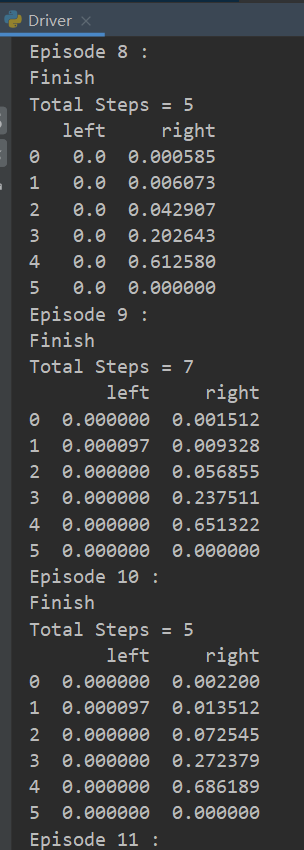
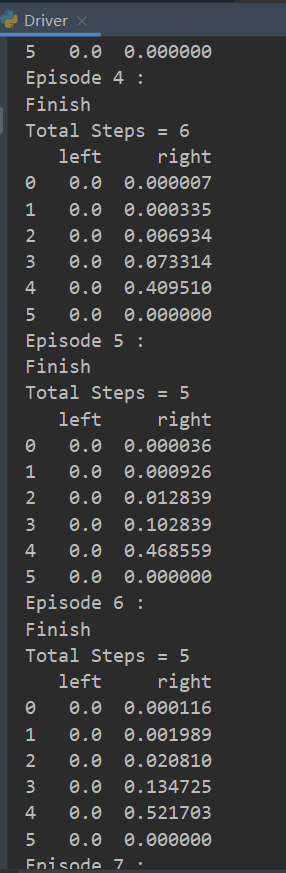
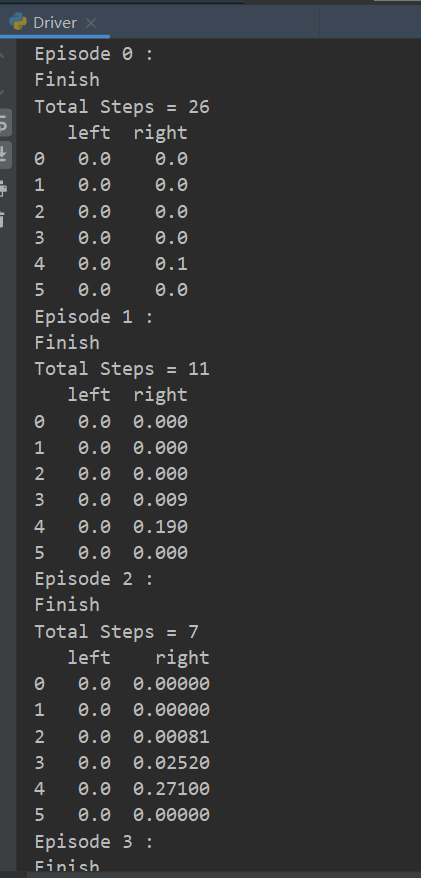
再计算reward + γ \* q\_real作为在当前状态下采取action获得的实际价值

用实际价值-预估价值作为学习反馈，更新在当前状态下采取动作action的q\_predict

最后更新状态并进行下一步



## Q表分析

右图是每一轮Q表的更新过程

从中可以看到，在第一轮结束时只有right的最后一个值有更新(第5是终点，不做动作预估处理)，这是因为初始化时全部为0，随机移动后只在终点位置才会获得一个reward=1，因此第一轮更新只会更新终点前的最后一步(4)的q值。

同理第二轮的时候只在倒数第二步(3)时采取了right动作才会获得预估回报，由于衰减系数为0.9，故(3)更新为α\*0.9\*0.1，而alpha=0.1，所以最后得到的是0.009，而此时(4)因为已经有了一个q值，因此在学习时计算式是q = q + 0.1\*(1+0.9\*0 – 0.1) = 0.19

以此类推，基本上在第5次学习之后基本就得到了只往右走的Q table，此后除了随机选择到往左走外，都会选择往右走。可以看到每一轮后Q表都有更新，且其中采取right动作的预估值远高于采取left动作的预估值，left中一些值为0是因为如果ε-贪婪策略没有随机到采取往左走就不会增加，而由于90%的概率选取最高估值动作，所以一旦学习到right，就很少选择往左走，也就很少更新left的值。这点从上图的Episode9可以看到，步数为7，意味着在状态(1)随机往左走了1次，导致多了2步(还有一步往右)，此时更新了状态(1)处left的q值。

# 实验中出现的问题和相应的解决办法

1. 对pandas不熟悉，所以在编写的时候一直出bug，然后去查函数的用法