SmartCab项目报告

1. 实现基本的驾驶智能体
   Qusetion：在您的报告中，说明观察到的智能体的行为。它最终到达目标位置了吗？
   Answer：
   在代码中加入

```
self.state = (self.next_waypoint, inputs)
action = random.choice(self.env.valid_actions)
```

使agent接受将next_waypoint, inputs作为当前状态state。
选取的行为从valid_actions随机选取 （即前进、左转、右转）。
经观察由于每次行为是随机的，无法对小车的行为进行预估。但是误打误撞还是能到达重点的。此时没有设置Qtable以及Reward，因此在每一步中也无法进行更新。
————————————————————————————
2. 确定和更新状态
Qusetion：对您选取上述状态组的原因，以及这些状态组对模拟智能体及其环境的方式进行说明。
Answer：上题中已经说明，选择nextwaypoint, inputs作为状态组。因为上述变量可以唯一并且精确地确定一个cab所处的状态。其中包括当前状态（红绿灯，是否有来车等），在一开始的时候曾经考虑将deadline也加入state中，后来觉得如果加入了deadline变数字典索引就太麻烦了，发现在没考虑deadline情况下训练效率不错。因此最后只将next_waypoint, inputs作为状态组。
————————————————————————————
3. 实现 Q 学习

```
self.Qtabel = dict()
self.discount = 0.3
self.gamma = 0.6
self.epsilon = 0.2

self.lastState = 0
self.lasAction = 0
self.lastReward = 0
```

在reset：

```
self.lastState = 0
self.lasAction = 0
self.lastReward = 0
```

只重设这三个变量。Qtable不需要重设。
在agent内定义一个Qtable作为Q学习的矩阵，将state和action生成的tuple作为键值。每次update函数执行时，重复以下过程。

获取当前状态，取得上一个状态的数据
`lastState,lastReward,lastAction`。
按照公式:

```
Q(s,a) = (1-α) * Q(s,a) + α*(R + β* max (Q(s + 1, a')))
```

对Qtable进行更新。由于Q(s+1, a')只有当执行action后才能得知，因此采取的策略是存储上一状态的数据，在下一个update更新。
发现智能体的行为具有了策略性，也就是在Qtable的指导下更加的“智能“。在pygame中可以看出来：在开始的几轮中，由于Qtable为空，小车采取随机的行为。因此开始的trial中是对小车进行训练，小车不一定能达到目的地。随着Qtable的更新，小车行为越来越有目的性。
加入epsilon参数后，小车会在一定的概率下出现随机的行为（而不是按照qtable走）通过这样的方法可以一定程度上解决局部最优值得问题。但是这个值不宜过大，过大的话就影响影响准确率了。后面详述

————————————————————————————

4. 改善驾驶智能体的表现
Question::报告您为了获得智能体的最终版本而对 Q 学习的基础实现所做的更改。它的表现如何？
智能体是否快接近找到最优策略，即在尽可能短的时间内到达目的地，同时未受到任何惩罚？

在对小车参数进行更改时，考虑了以下因素：
alpha:学习速率。由于公式中
Q(s,a) = (1- α) * Q(s,a) + α *(R* + β max (Q(s + 1, a')))
alpha越大代表新状态代表的比值越高，当alpha为1的时候完全忽略之前的状态，alpha为0的时候相当于没有进行学习。
视频上说可以讲alpha设置为1/t，但是经过实践发现效果并不是特别的理想。现在将alpha设置为常数0.3.即新的状态还是占了较大的比重，因为考虑到destination会改变，新的状态要比之前的重要。

选取alpha∈(0.1, 0.2, 0.3)
gamma∈(0.4, 0.5, 0.6)
epsilon∈（0.06, 0.1, 0.14）
每组参数实验五次，最后求平均数作为该参数的准确率，得到以下数据（附件）。现对部分进行分析：

	A	B	C	D	E	F	G
1	(alpha, gamma, epsilon)						Average
2	0.1 0.6 0.06	98	96	97	97	97	97
3	0.1 0.5 0.06	96	96	96	94	96	95.6
4	0.1 0.4 0.06	94	99	93	97	91	94.8
5	0.1 0.7 0.06	94	93	92	97	93	93.8
6	0.3 0.4 0.06	94	93	92	95	93	93.4
7	0.2 0.4 0.1	93	88	94	95	96	93.2
8	0.3 0.4 0.1	95	90	95	93	93	93.2

最优决策为(0.1, 0.6, 0.06)
从之后的趋势看到随着gamma参数的降低准确率也在降低，但是gamma参数最高的0.7却是整组数据的最小值。

alpha:这里选择为0.1，代表旧的状态占10%的比重，而新状态占90%比重。我认为在本项目中是符合预期的。因为本项目的终点处于变化状态。之前的Qtable可能对当前状态不适用。因此新状态占较高的比重是合理的。

epsilon：这里选择为最低的0.06.我认为也是合理的。因为此参数存在的意义是使得决策避免陷入局部最

优值。但是如果此参数过高就使得小车有更高的概率随机行驶，也就没有学习的特性了。

gamma：这个参数也是用于形容当前状态的重要性的。但是并没有总结出规律来。只能说这个参数处于0.5~0.7之间我认为是有意义的。

action选择方案：
由于epsilon参数的加入，action的决策分为两个：一个是按照Qtable最大值进行决策，概率为(1-epsilon)即，智能车在大部分情况下按照学习的经验在行驶。另一个是随机行驶，目的是避免局部最优值。概率为epsilon，同时当Qtable有多个最大值时从这些最大值中随机选取一个进行action、

我认为本项目中智能车的最优策略是安全、快速的到达目的地。其中安全更加重要，因此我觉得如果自己修改reward的话可以将闯红灯的惩罚加重。其次才是高效快速的到达目的地。
经过qlearn训练的小车在参数选择合适的情况下可以达到95%以上的准确率，同时保证reward为正。