Requires Changes

**还需满足 5 个要求 变化**

* 总体来说做的不错，但是在细节方面存在一些问题，请参考审阅进行修改。
* 同时，你对参数的调整和分析还可以更充分——这是 Q Learning 算法中非常重要的一个部分。请根据审阅的提示，深入你对参数的理解，并补充你的分析。
* 继续加油，期待看到你的下一次的提交！

**算法理解**

**正确完成 *问题 1：请参照如上的定义，描述出 “机器人走迷宫这个问题” 中强化学习四个组成部分对应的实际对象。***

**【问题1】**

* 对问题1的分析可以更具体：
* 「环境 : 迷宫对机器人来说就是环境」，那么具体来说，迷宫中包括哪些组成要素，会影响到我们的机器人呢？请具体说明。

**【问题2】**

* 「状态 : 状态包括：正常行走、撞到墙壁、走到陷阱、走到终点」对状态(State)的理解有些不到位。请参考如下的审阅，优化你的回答：
  + 实际上状态就是描述机器人当前情况的一个抽象概念，它与机器人能否成功学到一个策略息息相关。
  + **在我们的项目中，状态小车所处的迷宫坐标位置，例如 (0,1)、(1,1) 等。**
  + 如果将其设定为“处于起点”、“处于陷阱”、“处于终点”等情况，那么思考一下：
    - 这样设置与项目中设计的区别在哪里？
    - 这样机器人能够学习到一个成功的策略吗？
* 提示：如果按照「墙壁、终点、陷阱、其它」设置状态，那么当机器人处于这些状态中的一种时，有一个明确的、到达终点的行动吗？如果状态换成「小车所处的迷宫坐标位置」，存在这样的明确行动吗？

**正确完成 *问题 2：根据已知条件求 $q(s\_{t},a)$，在如下模板代码中的空格填入对应的数字即可。***

**【解释】**

* R\_{t+1} 是小车在S1执行动作u之后得到的奖励，不是下S2执行动作R得到的奖励。
* 这里的下标为t+1的含义是，在t的时刻执行动作a，则会在第t+1时刻得到对应的奖励 R\_{t+1}。这里是 小车在 S1 执行了动作u之后，因为没有撞到墙壁，所以获得-0.1的奖励，见报告中，Section0的第一节的第三点。
* 相应地，max\_a q(a,st+1) 代表着，在 t+1 时刻，在状态 s\_{t+1} 下， 对所有的动作而言，小车 q 值能够取得的最大值，也就是我们在S2下的最大的Q值。

**正确完成 *问题 3：在如下的代码块中，实现 epsilon-greedy 算法的逻辑，并运行测试代码。***

**【问题】**

* 在这里，随机选择的功能实现不太正确，if True 这行代码并不会按照 epsilon 的概率执行对应的分支。
* 你可以考虑使用一个均布分布的随机变量来与之比较，如 random.random() < epsilon ，来实现这个随机选择的功能。
* 思考：这么做的原因是为什么呢？

**代码实现**

**正确完成 *问题 4：在如下的代码块中，创建你的迷宫并展示。***

**【优化】**

* 你可以尝试使用 test\_world 目录下的迷宫，如 test\_world/maze\_01.txt，或者修改 Maze 类中 \_\_generate\_maze 函数中的 complexity 及 density 变量，来增加迷宫的难度～

**正确完成 *问题 5：随机移动机器人，并记录下获得的奖励，展示出机器人最后的位置。***

**正确完成 *问题 6：实现 Robot.py 中的8段代码，并运行如下代码检查效果。***

* 请参考代码审阅优化你的代码。

**【解释】**

* 请根据下方的解释，优化一下你对 Q Learning 公式的理解：
* 在如下所示 [Q-learning](https://en.wikipedia.org/wiki/Q-learning) 更新公式中，它考虑了两部分的信息：之前学习到的Q值，以及新学习到Q值。  
  ，
* 在新学习到的Q值中，γ\*maxQ 的一项目就考虑了所谓的「未来奖励」——这是强化学习中的一个巨大亮点。也就是说，我们在计算、衡量一个动作的时候，不仅考虑它当前一步获得的奖励 r，还要考虑它执行这个动作之后带来的累计奖励——这能够帮助我们更好地衡量一个动作的好坏。但是这时候机器人并没有真正地往前走，而是使用Qtable 中原有地 next\_state 的值来估计这个未来奖励。
* 其中 γ 是折扣因子，是一个(0,1)之间的值。一般我们取0.9，能够充分地对外来奖励进行考虑。如果这个值大于1，那么实际上未来奖励会发散开来（因为这是一个不断累加、迭代的过程），导致Qtable不能发散。它能够帮助终点处的正奖励“扩散”到周围，也就是说，这样机器人更能够成功地学习到通往终点的路径。

**正确完成 *问题 7：尝试利用下列代码训练机器人，并进行调参。***

* 请根据如下的解释，进一步调整参数～

**【解释1】**

* 首先给你补充一下对于 epsilon greedy 算法的解释：
* 对于 epsilon-greedy 算法，你可以参考论坛中的 [这个帖子](https://discussions.youdaxue.com/t/topic/33333)：

Q: 如何理解 greed-epsilon 方法／如何设置 epsilon／如何理解 exploration & exploitation 权衡？  
A: (1) 我们的小车一开始接触到的 state 很少，并且如果小车按照已经学到的 qtable 执行，那么小车很有可能出错或者绕圈圈。同时我们希望小车一开始能随机的走一走，接触到更多的 state。(2) 基于上述原因，我们希望小车在一开始的时候不完全按照 Q learning 的结果运行，即以一定的概率 epsilon，随机选择 action，而不是根据 maxQ 来选择 action。然后随着不断的学习，那么我会降低这个随机的概率，使用一个衰减函数来降低 epsilon。(3) 这个就解决了所谓的 exploration and exploitation 的问题，在“探索”和“执行”之间寻找一个权衡。

**【解释2】**

* 再给你补充一下对 alpha 的解释。 alpha 是一个权衡上一次学到结果和这一次学习结果的量，如：Q = (1-alpha)\*Q\_old + alpha\*Q\_current。
* alpha 设置过低会导致机器人只在乎之前的知识，而不能积累新的 reward。一般取 0.5 来均衡以前知识及新的 reward。

**【解释3】**

* gamma 是考虑未来奖励的因子，是一个(0,1)之间的值。一般我们取0.9，能够充分地对未来奖励（也就是执行了这一步动作之后，在对应的下一状态下，执行动作会带来什么奖励）进行考虑。
* 实际上如果你将它调小了，你会发现终点处的正奖励不能够“扩散”到周围，也就是说，机器人很有可能无法学习到一个到达终点的策略。你可以自己尝试一下。

**正确完成 *问题 8：使用 runner.plot\_results() 输出训练结果，根据该结果对你的机器人进行分析。***

* 你做了很好的尝试，对参数进行了初步的分析。

**【问题】**

* 请进一步补充你的分析：
  + 我们希望你在这里能够更详细地说明每个参数（alpha、gamma、epsilon0 和 epsilon下降函数两者的区别和联系、训练次数）的**作用**是什么，它们的变化大概会怎样影响运行结果，然后有目的地对小车进行调参，比较不同参数下的训练结果，并说明你使用这个参数的原因。
  + 总结出这些参数值的变化将如何影响你小车的训练结果。
  + 对比在不同的参数组合下小车的运行结果，并将结果打印出来你（你可以复制 runner.plot\_results() 代码对结果多次打印。请至少对比三组参数组合的结果。
* 这样你的报告会更加严谨～

**【优化】**

* 在训练中，不同的迷宫将会给训练带来很大的影响，如：
  + 迷宫本身较大或者较难，不利于训练，将会造成训练结果很糟糕。
  + 陷阱位置不好：如下图所示，在通往终点的路上，有一个陷阱，也会导致训练结果变差。
* 因此，建议你固定迷宫，再对比不同参数下的结果；同时你也可以把迷宫看作一个可以修改的变量，来看看如何调整参数，才能来对复杂迷宫学到一个成功策略。

**【优化2】**

* 关于炸弹堵死道路的问题，你是不是可以：
  1. 设置机器人，在遇到陷阱的情况下，增大随机探索的机率，能从陷阱跳出来？
  2. 尝试调整 reward 的设置？

**将 robot.py、 robot\_maze.ipynb、robot\_maze.html 的文件打包，提交文件给审阅者。**

 重新提交项目

[**下载项目**](https://review-api.udacity.com/api/v1/submissions/1874651/archive)

[**1**  代码审阅评注](https://review.udacity.com/)

了解 [修改和重新提交项目的最佳做法](https://review.udacity.com/).