人工智能基础
第三次综合作业

姓名： 程晔安
学号: 2017011627
班级: 自 72

2020 年 1 月 15 日

目录

1 任务描述

本次任务是如下场景: 一个动力不足的小车需要爬上一维的山到达目的地, 目标在右面的山上, 左边的山可以用来获取势能或者加速.

1.1 环境解析

首先要了解 OpenAI 给我们提供的环境. 作为和智能体, 与环境的交互为状态和动作的交互. 我阅读了环境源码, 对两个任务的环境分析如下:

1.1.1 环境和观察

首先, 两个任务的共同点是, 智能体小车从山谷出发, 由于自身的油门不够, 需要通过左右荡来积累自己的动能和势能, 在能量足够的时候就可以到达终点. 通过源码阅读, 我发现山的曲线其实是一个余弦曲线. 速度与位置的计算已经被后端实现完成. 第一题来说, 除了到达的奖励为 100 以外, 其他所有位置的奖励都为 -1. 第二题的奖励为到达右侧山丘目标的 100 减去从开始到目标的动作平方和. 如果不能很快到达目标, 那么智能体可能会选择不移动的策略. 在官方给出的要求中, 需要最终奖励大于 90 才算任务最终完成.

1.1.2 状态和行动

小车的状态由位置和速度组成, 位置的大小为 $[-1.2, 0.6]$ 之间的任意数. 对第二题而言, 其速度为 $[-0.07, 0.07]$ 之间的任意数. 小车的随机出生点为 $[-0.6, -0.4]$. 值得说明的是, 这个出生的范围其实相比整个范围来说是比较大的范围, 因此初值很可能影响算法最后是否成功的结果. 这也是为什么后面的算法很少有可以到达稳定的 100% 成功率的原因.

两个任务的不同点是对于小车的控制是否是连续的. 在第一题中, 对于小车的控制是离散的, 即策略只可以选择小车的如下三种状态: 向左, 向右和不动, 速度为固定值.

在第二题中, 对于小车的控制是连续的, 不但可以选择方向, 也可以选择大小. 因此, 第一题只是第二题的一种特殊情况, 在第二题的情况下, 有可能存在比第一题更优的策略.

1.2 预处理

首先我们要根据题目建立 Q 表. 题目中的状态空间由于所给定的范围是连续的, 其实上有无穷多个状态. 我们首先需要确定一个比较合适的步长, 将无限的状态空间分为有限的状态.

在任务一中, 我将状态分为 20 个离散的取值, 即确定了 Q 表的结构为 $[20 \times 20 \times 3]$ 其中前两个 20 代表位置和速度, 最后一个 3 代表三种行动. 而任务二中, 最后的"3" 会变为人工离散化控制区间的取值结果.

2 任务一

我利用 QLearning,SARSA,E-SARSA 三种时序差分对任务进行了实现, 尝试了 DQN 算法, 并更改相关参数进行了对比. 下面进行分别展示.

2.1 基础模型

2.1.1 Q-Learning

首先我利用 Q-Learning 的方式进行训练, 在每一个 episode 中的每一个步骤中, 选择 $\epsilon-$ 贪心策略作为行为策略, 利用贪心算法作为目标策略. 具体来讲, 假设 S_t 作为 t 时刻的状态, 在本题中,$S_t = (Postion_t, Velocity_t)$ 分别代表小车的位置和速度, A_t 作为所采取的的策略在本题中, 只有 $-1, 0, 1$ 三种取值方式.$\pi(a|s)$ 作为策略.

则在每个 episode 中的每一步中, 首先采取 $\epsilon-$ 贪心策略作为行为策略, 即:

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{m} & if\ a = argmax_{a \in A} Q(s,a) \\ \frac{\epsilon}{m} & otherwise \end{cases}$$

其中 m 为总的策略数.

其中 *epsilon* 随着迭代步数的变大而逐渐变大, 直到最后迭代步数的一半之后,ϵ 的值变为 0, 变成完全的贪心策略. 这样做的目的是在迭代训练的前半部分勇敢探索, 在后半部分小心翼翼.

在采取行为策略之后, 智能体向环境请求回报, 收到自己的下一步的状态 s_{t+1} 和回报 R_{t+1}

之后根据目标策略从 S_{t+1} 产生 A_{t+1} 并计算其行动价值并进行更新. 具体来说, 行动价值采取贪心策略

$$\pi'(S_{t+1}) = argmax_{a \in A} Q(S_{t+1}, a)$$

进行更新:

$$Q(S_{t+1}, A_{t+1}) = Q(S_{t+1}, argmax_{a \in A} Q(S_{t+1}, a)) = argmax_{a \in A} Q(S_{t+1}, a)$$

$$Q(S_t, A_t) = Q(S_t, A_t) + Learning\_rate(R_{t+1} + \gamma max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t))$$

在实际应用中我选择学习率为 0.54, 后面价值的折扣 $\gamma = 0.95$ 总代数为 10000 代,Q 表的大小为 $[20 \times 20 \times 3]$, ϵ 初值为 1, 这些参数的选择对最终的结果都有很大的影响. 参数选择的对比实验在后面进行.

最终的实验结果是, 在第 4000-4500 代的时候出现了成功的结果, 在 8000 代以后, 算法的成功率在 95% 以上, 平均 Reward 在-155 左右.

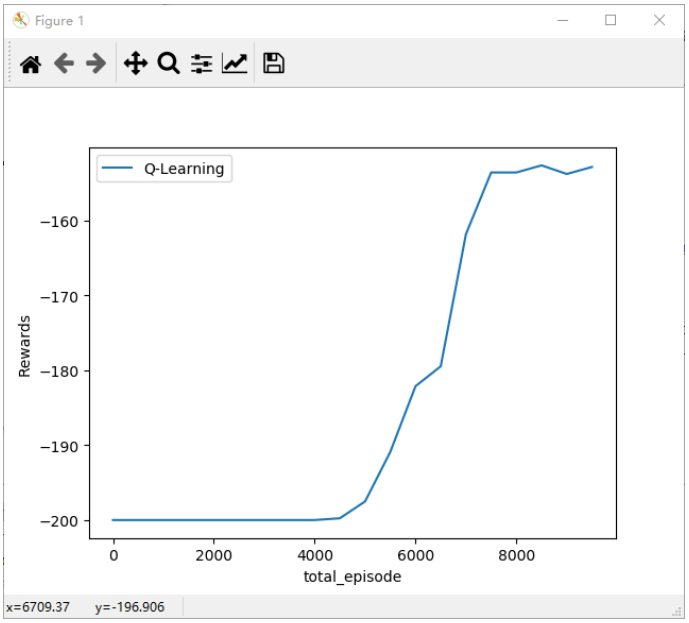

图 1: Q-Learning 结果

2.1.2 SARSA

我还使用了 SARSA 对任务进行完成,SARSA 方法中的不同点是其行为策略和目标策略均为 $\epsilon-$ 贪心策略, 与 QLearning 不同每一步决策之后还会更新自己的行动.

$$Q(S_t, A_t) = Q(S_t, A_t) + Learning\_rate(R_{t+1} + \gamma a \in AQ(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

最终结果为, 在第 3500-4000 代的时候第一次成功, 在第 7000 代的时候就成功完成了训练, 成功率在 96% 以上 Reward 在-156 附近. 如图所示.

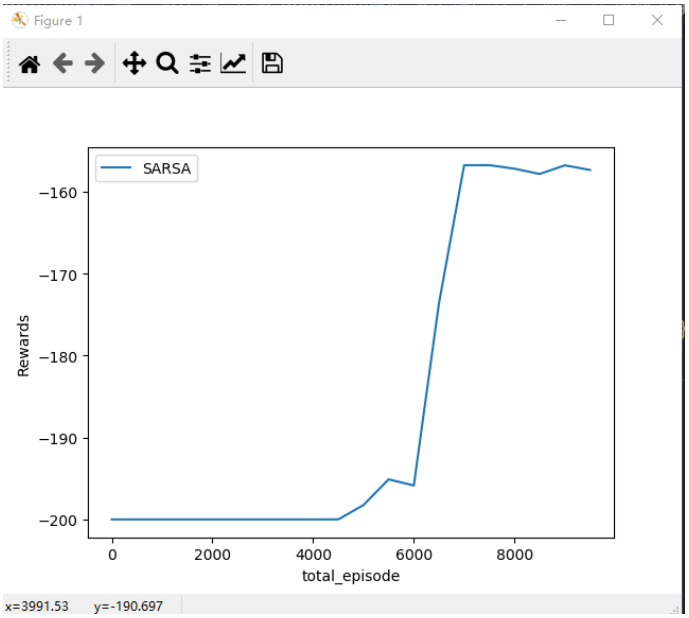

图 2: SARSA 结果

其效果明显优于 Q-Learning.

2.1.3 E-SARSA

我还使用了 E-SARSA 策略进行实验. 其特点是目标策略采用行动期望进行实现, 即:

$$Q(S_t, A_t) = Q(S_t, A_t) + Learning\_rate(R_{t+1} + \gamma \Sigma_{a \in A} \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t))$$

最终结果如下所示, 在六千代左右的时候算法就达到了 90% 以上的正确率. 在后面的正确率也逼近百分之百,Reward 在-151 左右.

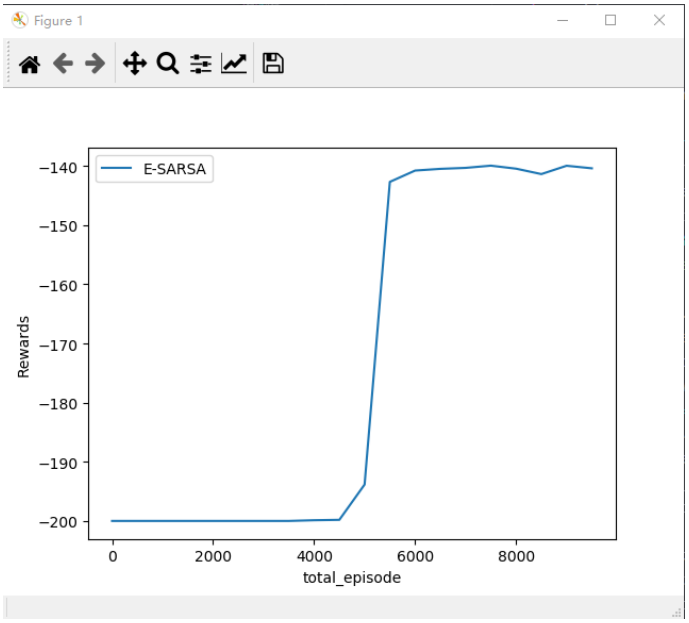

图 3: E-SARSA 结果

2.1.4 综合对比

将三种方法的图放在一起对比来看, 得到如下所示的两次结果.

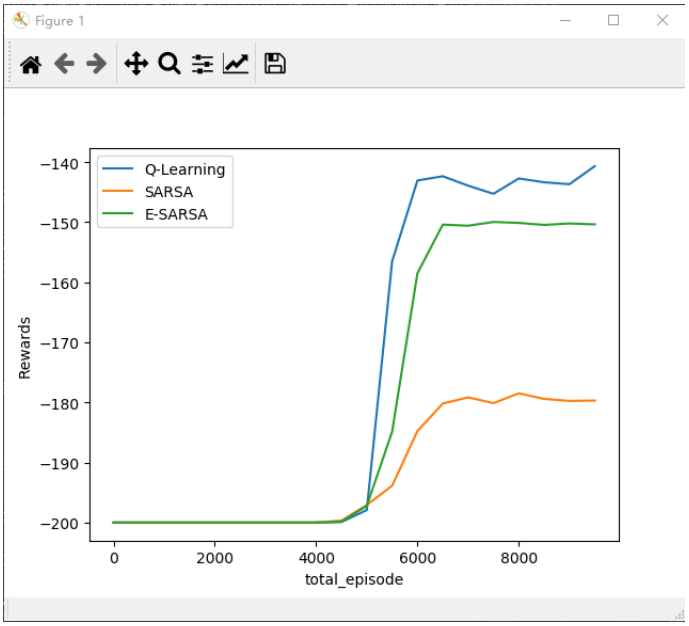

图 4: 综合对比 1

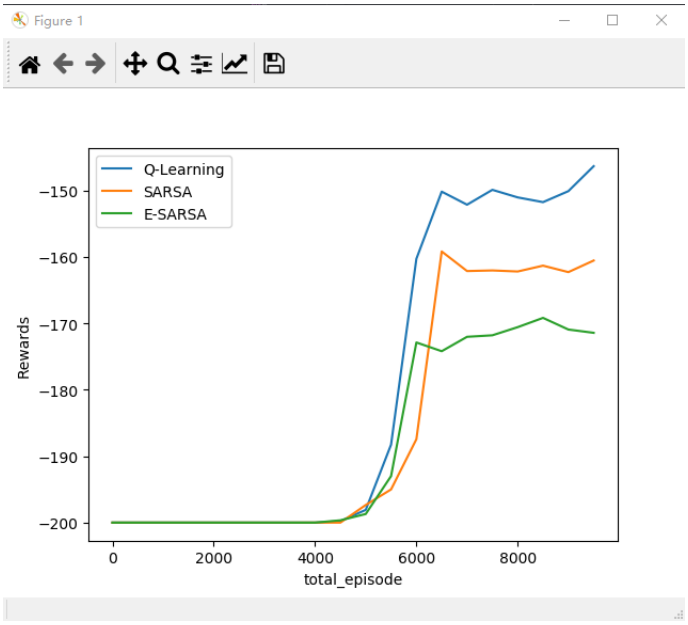

图 5: 综合对比 2

可以看到, 三种方法最后都可以完成任务, 但是 Q-Learning 在本任务中明显表现比较好. 值得说明的是, 本对比的各个方法的初值均为随机, 而本问题的初值可能对结果产生比较大的影响, 因此本对比有可能出现不同方法之间位次不同的情况.

2.2 深度学习 Deep-QLearning

受第二次大作业的启发, 我尝试采用深度学习的方法 Deep-QLearning 进行解决. 利用值函数近似, 我们使用值函数近似的方式实现, 通过与深度学习结合, 我们将问题转换成一个监督学习的问题, 利用标签或者已知的数据来修正神经网络 (比如 TD-target), 如下图所示. 神经网络实际上起到了 Q 表的作用.

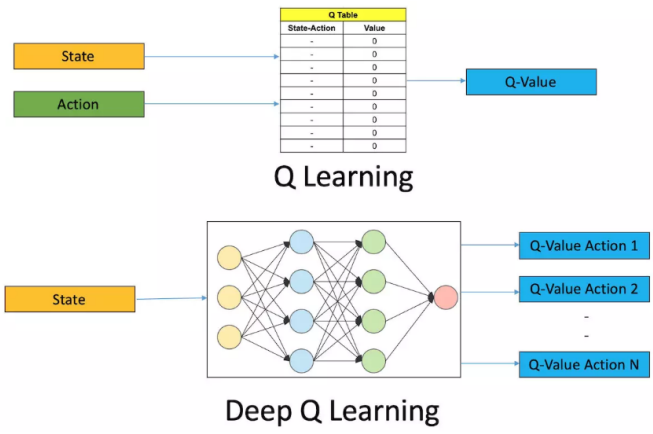

图 6: Deep QLearning

但是, 由于我的电脑的算力限制和我对 tensorflow 框架的掌握较差, 我选择了网络上的实例代码进行参考和对比, 实际的代码并不是我自己完成的. 但是利用这种方法进行对比和参考, 仍然有比较重要的参考价值. 最终, 我得到的结果如下所示.

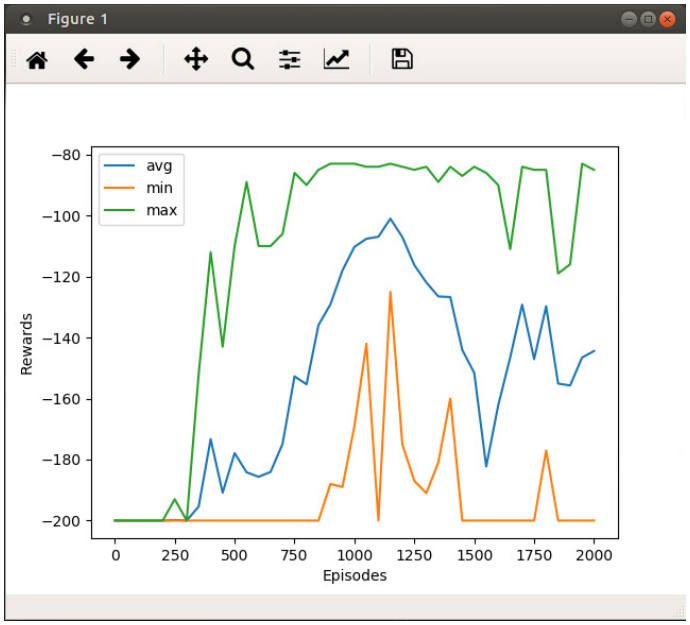

图 7: Deep QLearning 结果

可以看到, 在训练了代数较少的时候,Deep-QLearning 就可以比较好地完成任务, 但是由于本机算力较差, 只完成了 2000 代的学习过程, 最终的结果还在波动状态. 值得说明的是, 利用 Deep-QLearning 我认为在这个问题中并不适合. 本问题的复杂程度可以直接用一般方法进行解决, 效果也并不比神经网络的方法查, 同时节省了算力和时间, 是比较合适的做法.

2.3 参数调整

训练中的超参数十分影响训练的结果. 在本部分, 我们使用 QLearning 作为 baseline 进行不同参数的对比实验. 主要要对比的超参数有: 学习率, 折旧率 γ,Q 表的大小 q_table_size 贪心策略的 ϵ 的初始大小.

2.3.1 学习率

学习率的更新主要代表网络训练的过程中对新的数据的信任程度, 如果比较信任就可以以比较大的学习率进行学习. 这里主要对比 0.5, 0.55, 0.60 个学习率下的结果. 结果如下所示.

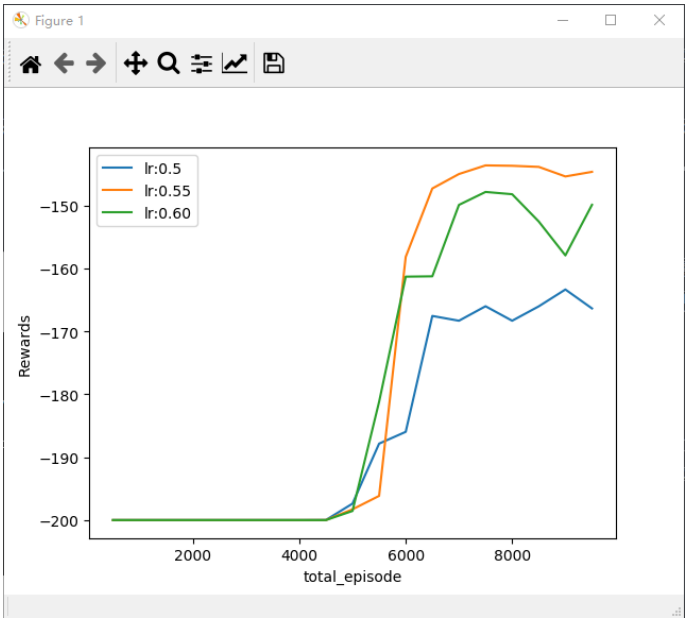

图 8: 学习率结果对比

可以看到在学习率为 0.55 左右的时候, 智能体到达最终目的地时候的 Reward 更高, 因此更为优秀. 值得说明的是, 三种学习率最后的 Q 表都能够达到很稳定能够完成任务的结果, 成功率在 97% 以上. 这说明学习率的结果并不影响算法的能力, 只是影响算法最后的回报大小, 即路径不同.

2.3.2 折旧率

折旧率 γ 表示后续状态对前面状态的影响程度. 如果影响程度大的话, 那么后续的状态价值应该对于当前的状态价值有更大的影响. 在本题中, 由于整个问题在一个一维环境下进行, 状态的前后影响一定很大, 因此选择比较大的折旧会产生比较好的结果. 这里对比了 0.9 0.95 1 三个折旧下的学习率的结果如下所示.

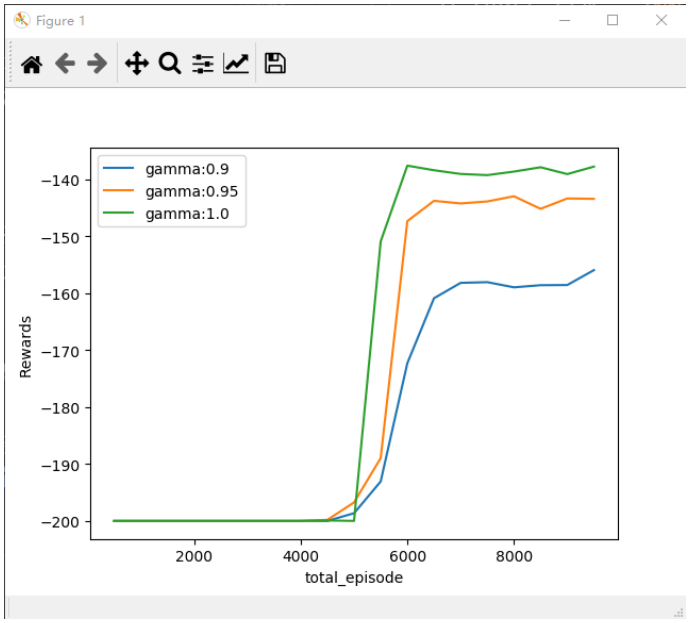

图 9: 折旧率结果对比

可以看到, 更大的折旧率会带来更大的回报, 因此回报率比较大是比较好的策略.

2.3.3 Q 表的大小

Q 表的大小也影响着训练的效果好坏. 直观来看, 如果 Q 表的取值较大, 那么整个状态空间就较大, 整个搜索过程更接近实际的爬坡过程, 有更多的位置点可以进行选择. 因此效果可能比较好, 体现在 Reward 的值可能会比较低; 而较小的 Q 表则把问题进一步离散化, 较小的状态空间意味着更小的计算成本, 表现在更好的成功时间和次数上. 但是小的状态空间代表的随机性更大, 因此这里我采用 10,20,100 三个值进行比较测试. 进行了两次测试, 结果如下:

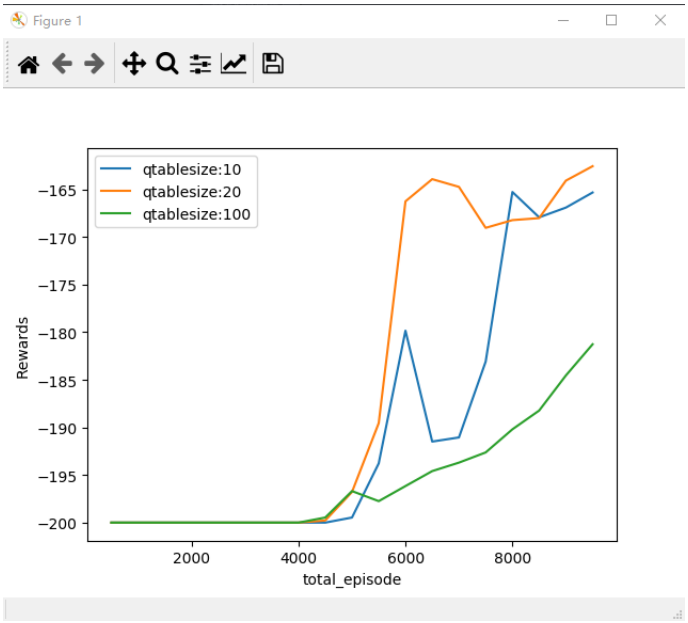

图 10: Q 表大小结果对比

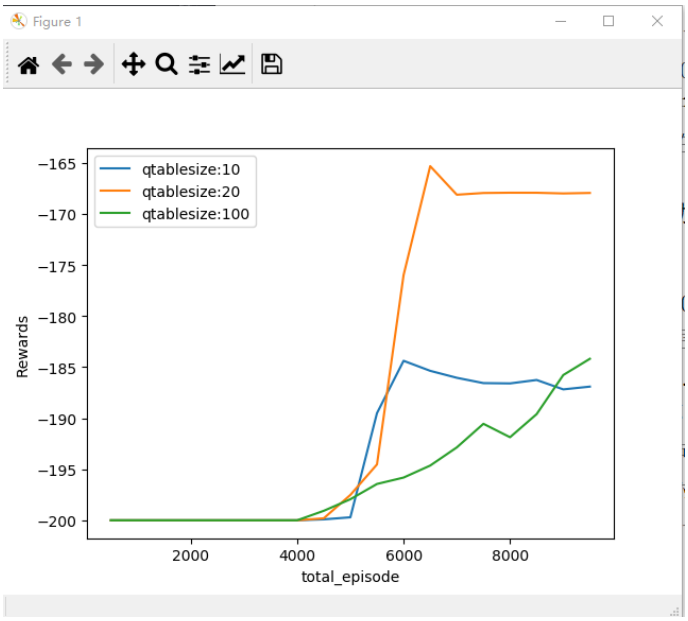

图 11: Q 表大小结果对比

可以看到, 当状态空间过小的时候, 训练的结果呈现出很强的不确定性, 而过大时, 网络可能需要更长的时间来学习 Q 表, 因此表现较差.

2.3.4 贪心策略初值的大小

ϵ 的值在训练过半之后会到达零, 因此它对最终结果的收敛与否可能不会产生过多的影响. 但是在学习刚开始的时候,ϵ 的值代表着网络勇于探索的概率是多少. 而在本问题中,

其实并不存在过多的陷阱和误区, 因此, 较小的 ϵ 初值可能会使训练过程变得更短. 我比较了 ϵ 为 0,0.5,1 三种情况的结果, 如下所示.

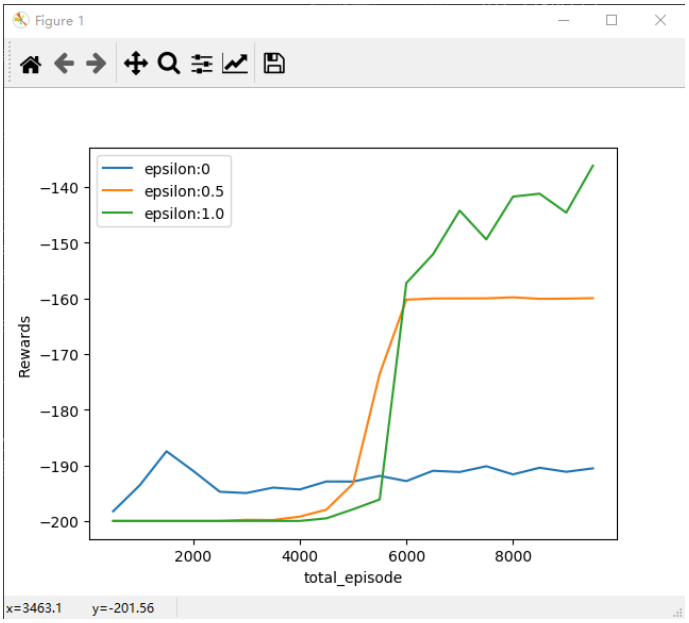

图 12: 贪心策略初值结果比较

可以看到, 过小的 ϵ 值虽然很快就找到了成功策略, 但是很有可能在一开始没有找到好的策略以后就没有办法探索其他策略, 因此只能得到比较一般的策略. 在本次试验中,$\epsilon = 0$ 下的智能体最后只有约 70% 的可能性可以走到终点. 而剩下两种情况都有 98% 以上的几率可以到达山顶, 但是勇于探索的策略 ($\epsilon = 1$) 明显找到了更加优秀的策略.

需要指出的是, 由于本问题对初值的敏感性和随机性, 有时候小 ϵ 的策略也有可能很快地找的非常好的结果数据, 但是十分不稳定. 因此在实际应用中还是应当选择牺牲掉一部分的搜索时间, 以较大的 ϵ 寻找可能的最佳策略.

3 任务二

在任务二中, 控制的连续性被放开了, 因此控制的精细程度也达到了更加大的程度. 针对这样的要求, 我们可以采用原来的方法, 对控制的速度大小也进行离散化, 再进行学习. 还需要指出的是, 智能体到达的奖励是 100 减去其动作的多少, 即

$$reward = reward - action^2 \times 0.1$$

因此本题有激励智能体尽快到达 (在尽量少步数中到达) 目的地的激励.

3.1 传统方法

我们采用原来的无信息的学习方式对第二题进行学习, 与第一题相比, 第二题的行动选择更多了, 但是同时由于奖励的改变, 出现了势井. 即使智能体在某个策略中能够到达目

标点, 由于方法不够优秀, 使用的步数太多, 获得的奖励仍然可能是负的. 在这种情况下, 智能体可能会选择出原地不动的策略. 因为没有把握使用上山策略获得更大的奖励, 还不如直接待在原地获得的回报更大.

我利用第一题中的方法进行实验, 得到了如下所示的结果.

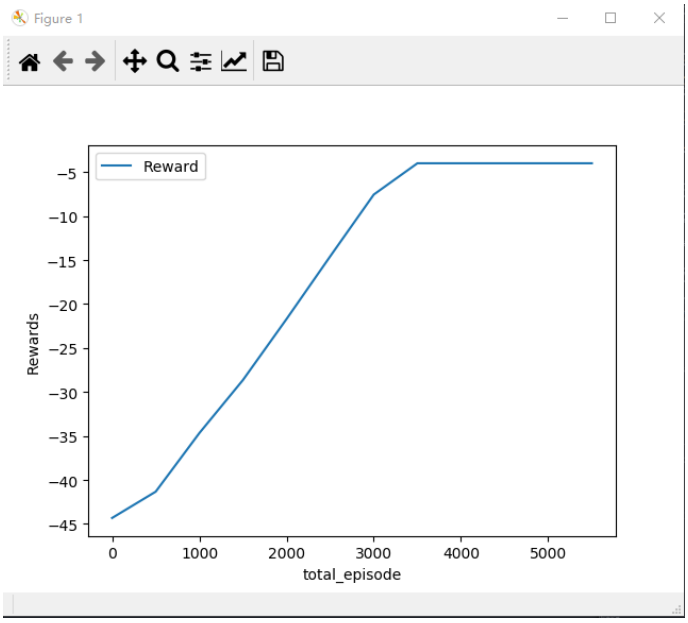

图 13: 无信息策略 Reward

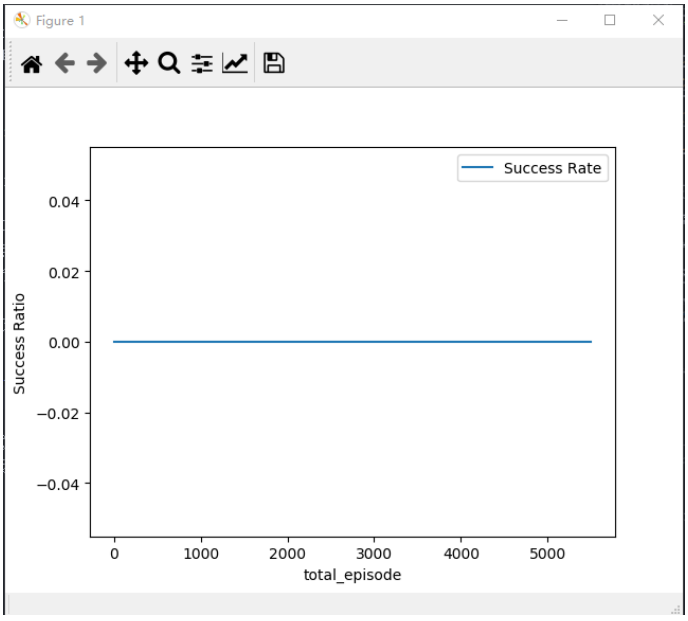

图 14: 无信息策略成功率

可以看到, 虽然智能体的平均回报增加了, 但是没有一次成功到达目标点. 这说明智能体最后总选择了在原地不动的策略. 第一问中的方法失效.

3.2 更改回报

3.2.1 Q-Learning 实现

直观上来看, 我们希望智能体在攀爬右边的山的时候获得更多的回报, 让其更优激励去努力向最后的终点进发. 如果我们不修改 Reward, 由于决策空间的变大, 智能体到达最终目的地的可能性就更小, 同时, 随着 Q 表的变大, 时序差分中的梯度反传所用的步数就更多, 因此在有限步数中学出最终结果的可能性就更小.

因此, 我们利用我们的直观思路来考虑, 当智能体的速度为正 (向右) 并且向右爬坡的时, 这是比较有可能到达终点的情况, 因此在此时我们应该增加激励; 如果智能体的速度为正 (向右) 并且在向左爬坡时, 其实是智能体在来回荡的情况. 这是我们不想看到的情况, 因为智能体有足够的能力不发生几次振动就走到终点. 因此, 我们将环境给与智能体的激励做如下改动:

$$reward = reward + \alpha \times position$$

其中 position 自带符号, 这样的改变即满足了上方描述的直观思路考虑. 式中,α 为改变激励的参数. 具体参数的影响在后面部分进行分析. 这里直观地取 $\alpha = 0.5$, 得到结果如下所示:

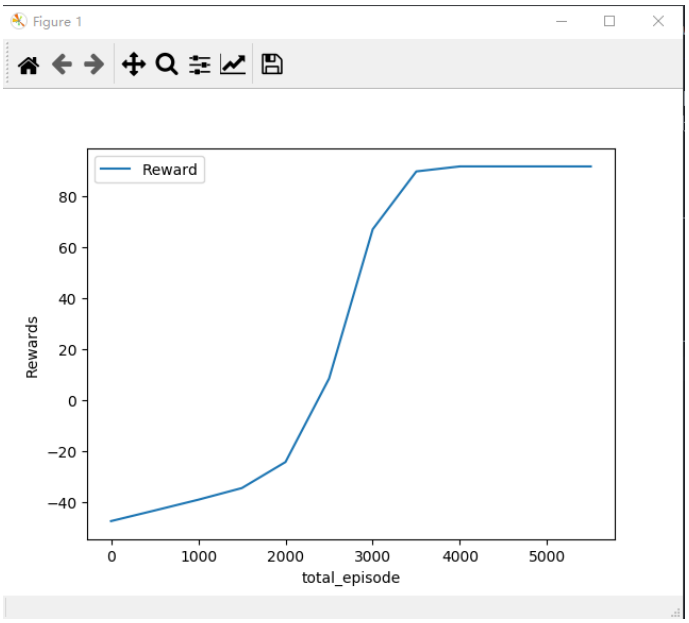

图 15: 更改回报 Reward

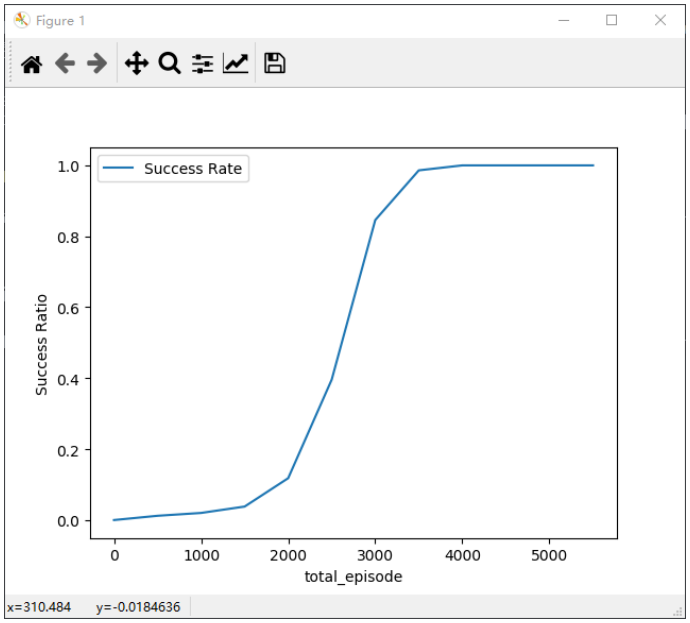

图 16: 更改回报成功率

可以看到, 这个很小的策略改进立刻让智能体很快地学到了到达终点的方法, 找到了走出势井的方法. 最后测试过程中小车的 Reward 为 91, 这也达到了 gym 官方给出的任务成功的标准.

在本题目中, 小车的动力其实比第一题中的动力要好很多, 因此小车只要在左边的山上稍微助力, 爬上四分之一左右的高度, 即可一下冲到终点处. 我们还可以根据自己的直觉创造更多的先验条件, 比如说根据小车的位置和速度条件判断出此时是否应该加速, 还是应该任由势能加速.

3.3 SARSA 实现

与题目一中类似, 我使用 SARSA 对实验进行了实现. 最终的实验结果如下所示.

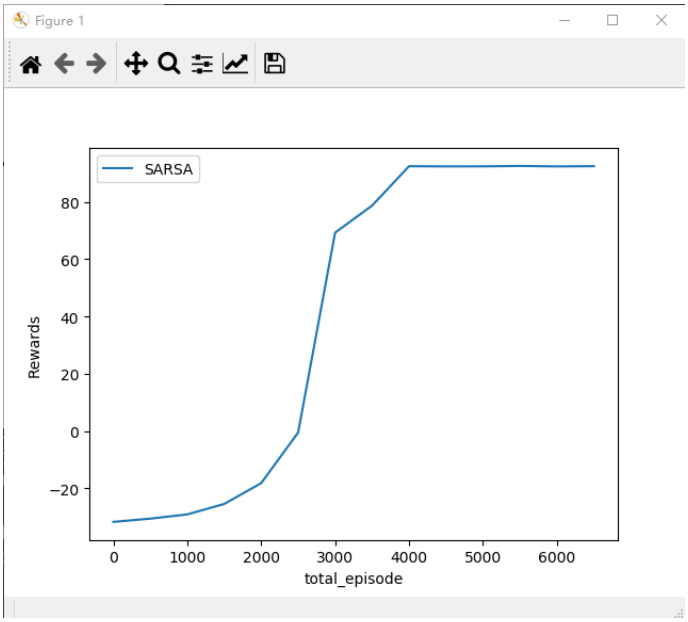

图 17: SARSA 的 Reward

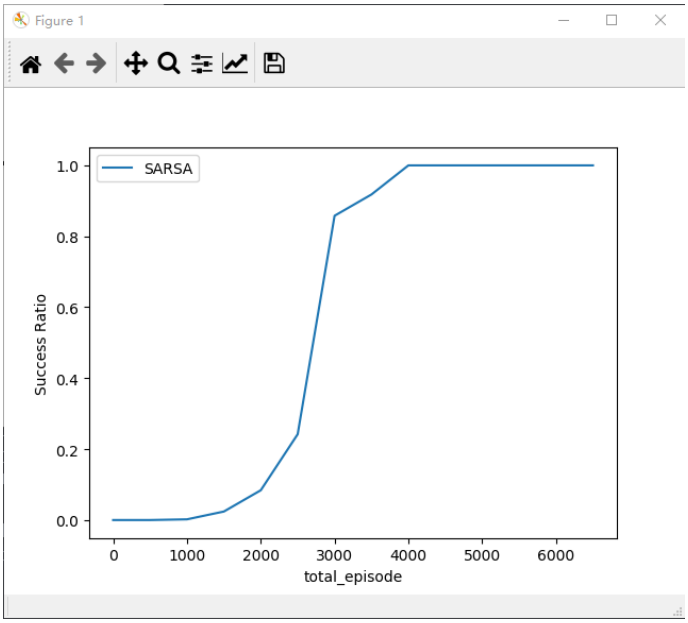

图 18: SARSA 的成功率

最终的得分为 92.36. 方法比较稳定.

3.4 E-SARSA 实现

与题目一中类似, 我使用 E-SARSA 对实验进行了实现. 最终的实验结果如下所示.

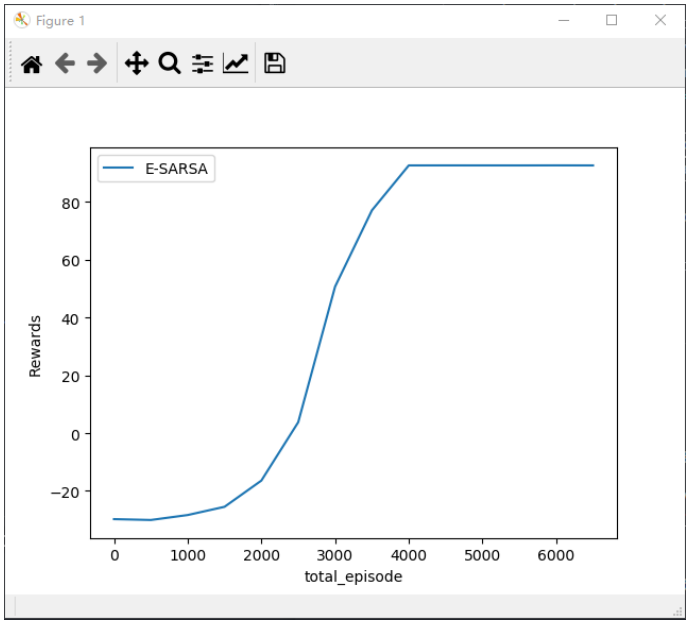

图 19: E-SARSA 的 Reward

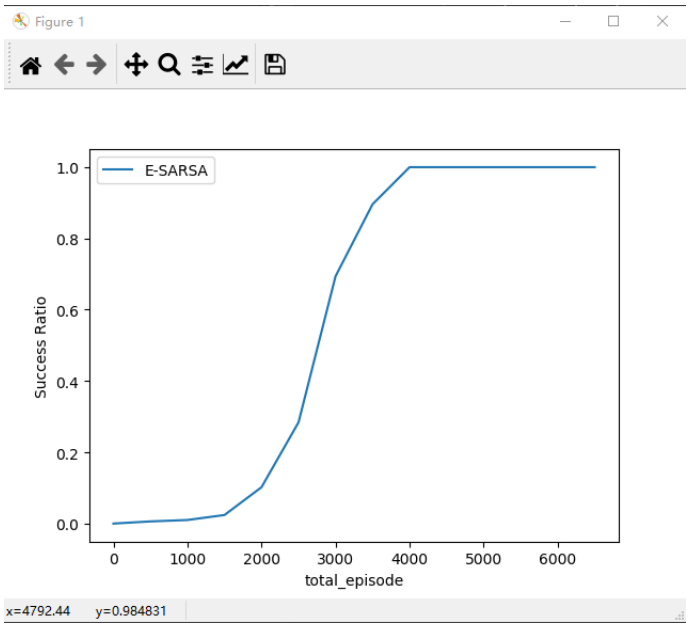

图 20: E-SARSA 的成功率

其最终得分为 92.7124. 但是由于 E-SARSA 的计算过程中需要求期望, 如果行动空间取值较大, 那么其训练过程将远远慢于其他两种方法. 在实际应用中需谨慎选择.

3.5 参数调整

在第二个任务中, 又出现了很多新的超参数可供调整, 同样, 这些新的参数对最终的训练结果也有着很重要的影响. 下面分别就初始化方式, 行动空间的大小, 更改回报后的比率

三个方面分别描述参数的影响.

3.5.1 初始化方式

首先, 随着 Q 表的变大, 全零的初始化或许并不是比较理想的办法. 相反, 随机初始化可能会带来比较好的结果. 在这里我使用全零初始化和两次 np.empty 随机初始化进行对比, 得到如下所示的结果.

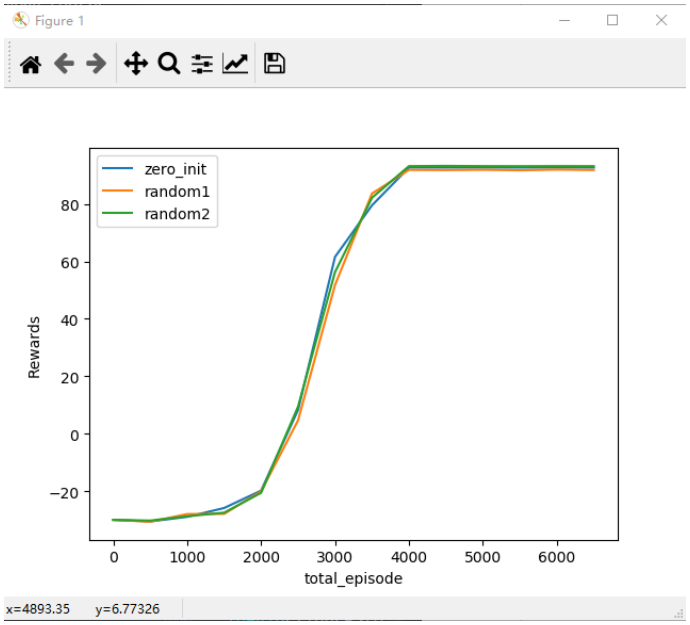

图 21: 初始化方式 Reward 对比

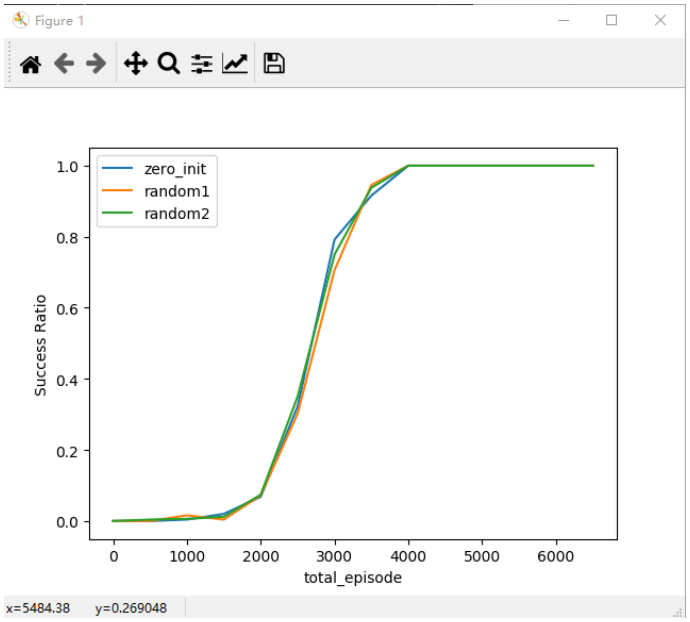

图 22: 初始化方式对比

可以看到, 在初始化的不同对整个训练过程的影响不大, 但是随机初始化的情况下智能体开始学到成功路径的和时间较快. 在实际应用中, 适当选择一定范围内的初始化, 可能会得到比较好的结果.

3.5.2 行动空间大小

正如前文所说, 行动空间的大小在本题中可以自己定义, 当行动空间过大的时候, 训练的速度会大大增强, 但是多余的操作可能并没有对最后的结果产生比较大的影响; 而行动空间过少的时候, 整个行动可能会退化得比较简单, 甚至最后退化到第一题的情况. 这样的退化显然会使问题最后的得分变少.

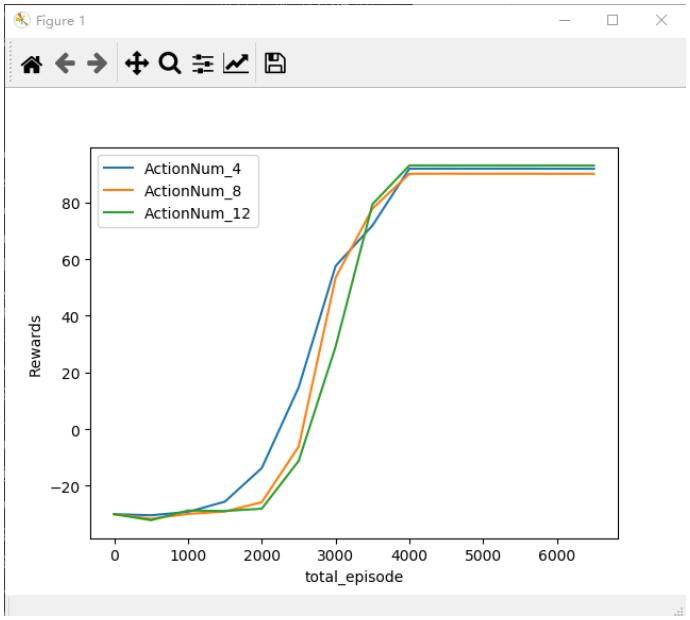

图 23: 行动空间大小 Reward 对比

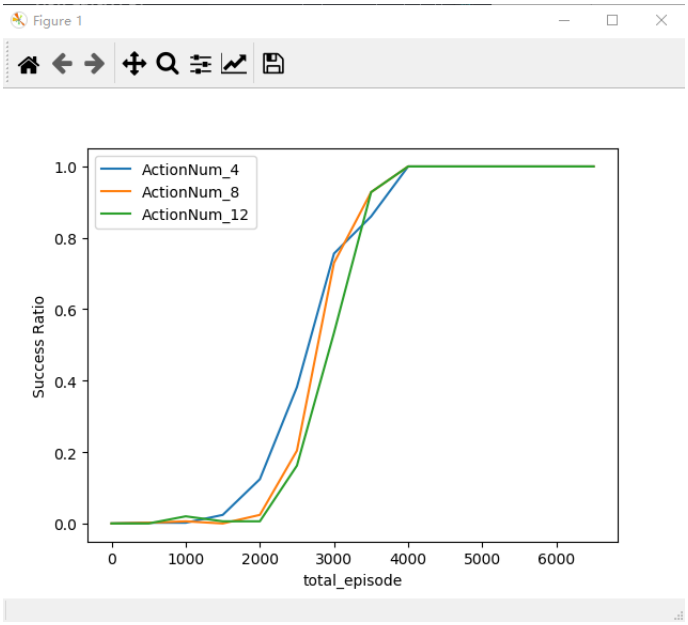

图 24: 行动空间大小成功率对比

可以看到, 比较小的行动空间由于 Q 表的大小也更小, 更容易学习, 智能体更早地开始提升自己的 Reward; 但是到最后收敛之后, 行动空间较大的智能体可能会学到得分更高的路径. 这来自于更加精细的控制.

3.5.3 更改回报后比率

更改给予智能体的回报, 是本方法能够工作的根本原因, 这里重写更改回报比率 α 的表达式:

$$reward = reward + \alpha \times position$$

当 α 值过小的时候, 有可能智能体受到的激励不够, 因此还是不能摆脱势井, 最终会得到在底部不动的策略; 而 α 过大的时候, 可能违背原来的环境信息太多, 导致不稳定. 最终我对比了 $\alpha = 0.2, 0.5, 0.8$ 三个比率, 得到如下所示的结果.

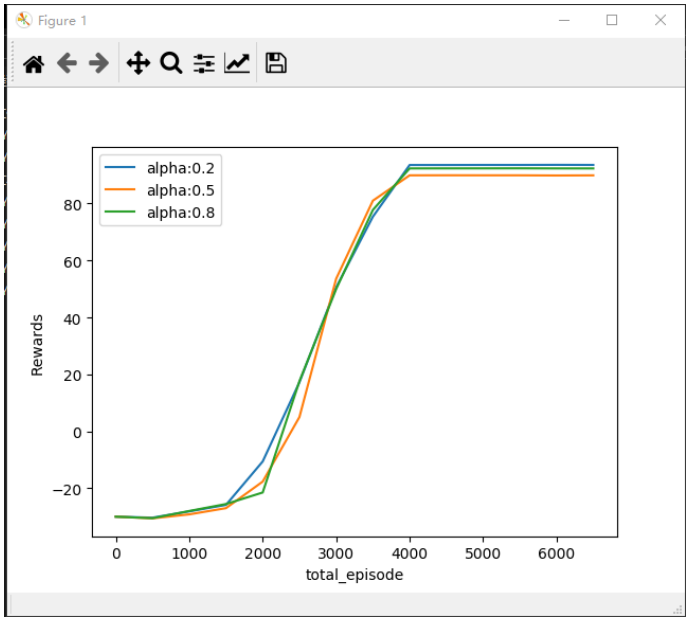

图 25: 回报比率大小 Reward 对比

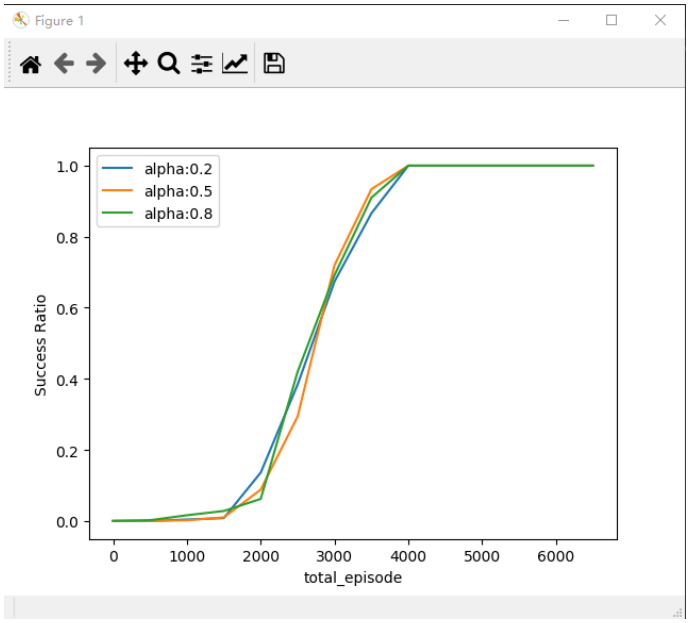

图 26: 回报比率大小成功率对比

可以看到, 即使更改环境的回报比率很小, 最终智能体也可以跳出势井. 较小的 α 值由于和环境比较贴近, 智能体最终的优化目标和状态目标比较接近, 因此可以得到最好的 Reward; 但是如果加大 α, 智能体就可以更快地跳出势井, 走向最优解.

4 总结

本次大作业考察了强化学习的相关知识. 具体来说, 我使用 OpenAI 的 gym 环境中的 MountainCar 题目作为例子, 亲手实现了几种老师上课讲的常用的强化学习的算法, 比如 QLearning,SARSA 等等. 在进一步加深对这几个算法的理解的同时, 我也更深地体会到了几种算法之前的区别和联系. 最后通过参数的调整, 我看到了各种各样超参数对于训练过程和最后结果的深刻影响, 并蕴藏在这些参数背后的物理意义. 我在本次大作业中收获了很多.

通过一学期的学习, 我通过上课听课, 小作业和大作业的方式中系统地了解并学习了人工智能基础的各个领域的发展和重要算法和思路, 我能在这门课上有这么大的收获, 离不开老师和助教的辛勤教导和耐心指导. 感谢老师和助教在这一学期的付出!