一.Gym mountaincar 环境学习

Mountaincar-v0：

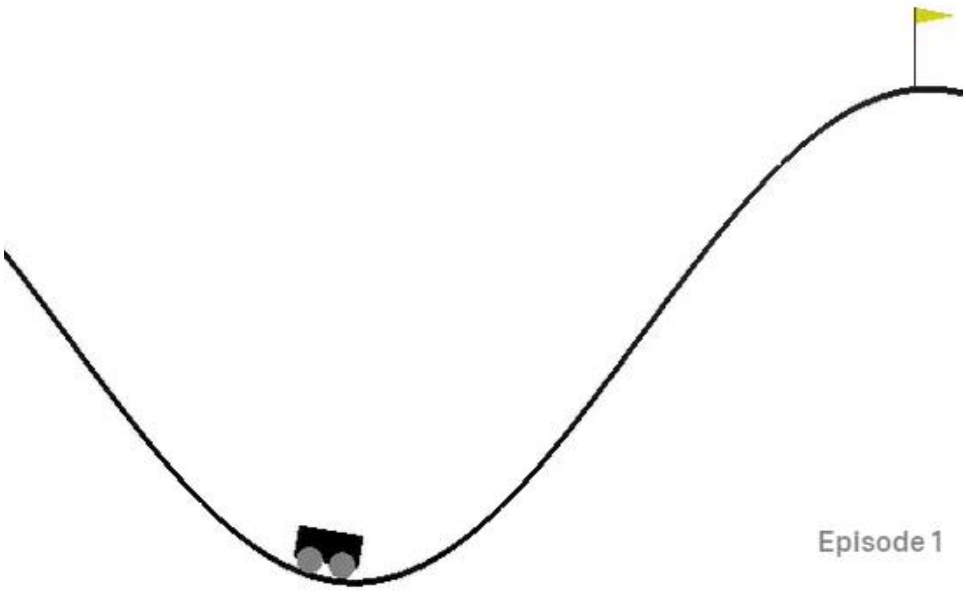
位置在-1.2 到 0.6 之间，路面是 cos 函数，高度是 cos(3*position)所以初始化时位置在-0.6 到-0.4 之间基本就在最下面。动作有 0，1，2 分别表示向左，无向右速度。终止状态是位置大于等于 0.5 。到达目标或超过 200 步则结束一个 episode

Gym 这个包主要所做的就是

1.帮我们写了__init__来规定这些规则

2.提供了 step 函数扮演物理引擎的角色，输入动作，它计算模拟物理引擎给我们下一步的状态，回报，是否终止

3.提供了 render 函数，图像引擎，帮我们画下面的图，这样可以直观地看到学习效果



MountainCarContinuous-v0

在上面的基础上动作空间不再是三个离散的值，而是连续的，步数限制由 200 步增加到 999 步。（虽然用 Q-learning 做的时候还是要给动作空间再离散化的）

二.所选方法讲解

在离散情况下讲解，连续情况稍微改动即可

2.1 Q-learning

Q 表为 20×20×3 的表，意义是[位置，速度，动作]，然后格内的预计累积回报值。

没有直接采用ε-贪心策略，而是让ε值最初是 1，然后随训练轮次下降直至设定的一个小的值。这样子最初鼓励随机尝试，后来就逐渐变为纯粹的ε贪心策略。

训练时每个轮次：

当未到达终点或步数小于 200 的每步：

按当前状态，根据当前 Q 表和策略得到一个动作

动作放到 step 里去模拟，得到下一个状态，回报，是否 done

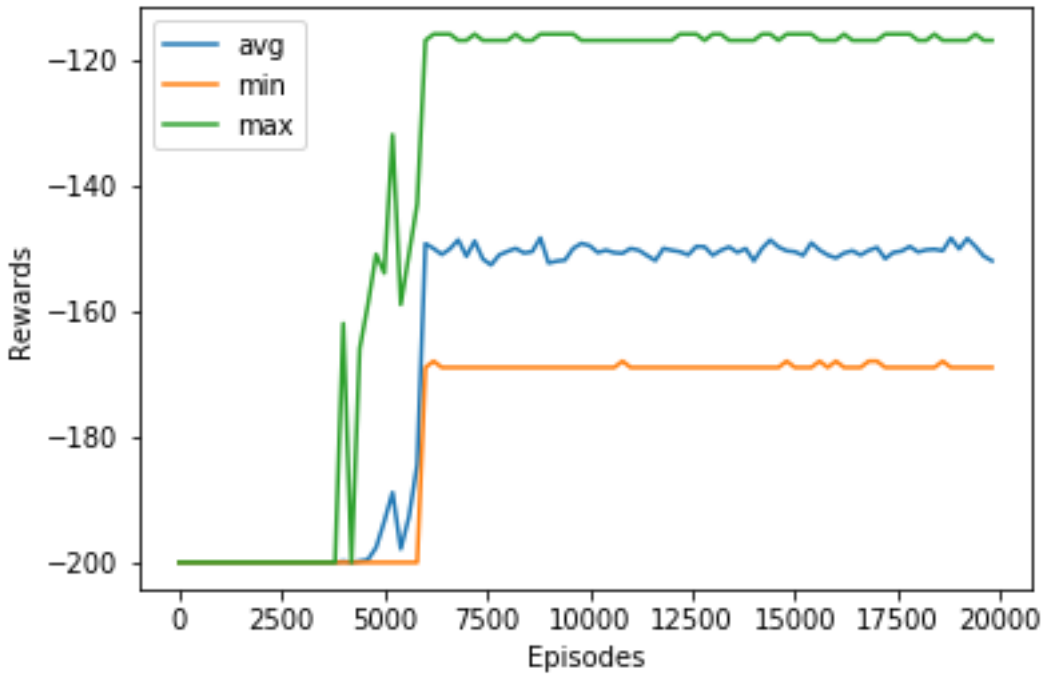
更新 Q 表里当前状态该动作的值

ε减小一点

其中更新 Q 表的方法是，该处的值+=学习率*（回报+下一个状态的预计回报中最大值）

最终这样就学到了一个好的 Q 表。

学习中间过程记录一个轮次的回报值，每 200 个统计一下显示，图如下：



可以看出从 5000 过后就开始学到东西了（这也是完全变为ε-贪心策略的时候）

最初的探索让其可以学到很好的效果，max 很高，结果比较稳定。

用 rendering 显示可以看到小车左右荡 3 下就上去了，花费 149 步。

2.2 DQN

DQN 就是把原本 Q 表的学习交给神经网络来做，神经网络的输入是状态，输出是三个动作的预计累计回报值。

则神经网络可以预测出一个 Q 表。

我们需要两个网络 prediction_model 和 target_model，前者是通过训练得到，后者是隔一段时间的训练就把前者的参数赋给它。

设置一个 memory 用来记录（当前状态，动作，回报，下一个状态，done）的五元组，类型用 deque，设置最大长度 5 万，这样模拟运行的每一步都会记录，超过 5 万条记录时扔掉最早的记录留下最新的。

2.2.1 运行过程：

对每个轮次：

对每一步：

根据当前状态和策略和用 prediction_model 预测当前状态得到的三个动作的预计累计回报，得到一个动作，然后放入 step 模拟得到下一个状态，回报和 done，在 memory 里加入该记录，进行一次训练。

2.2.2 训练过程：

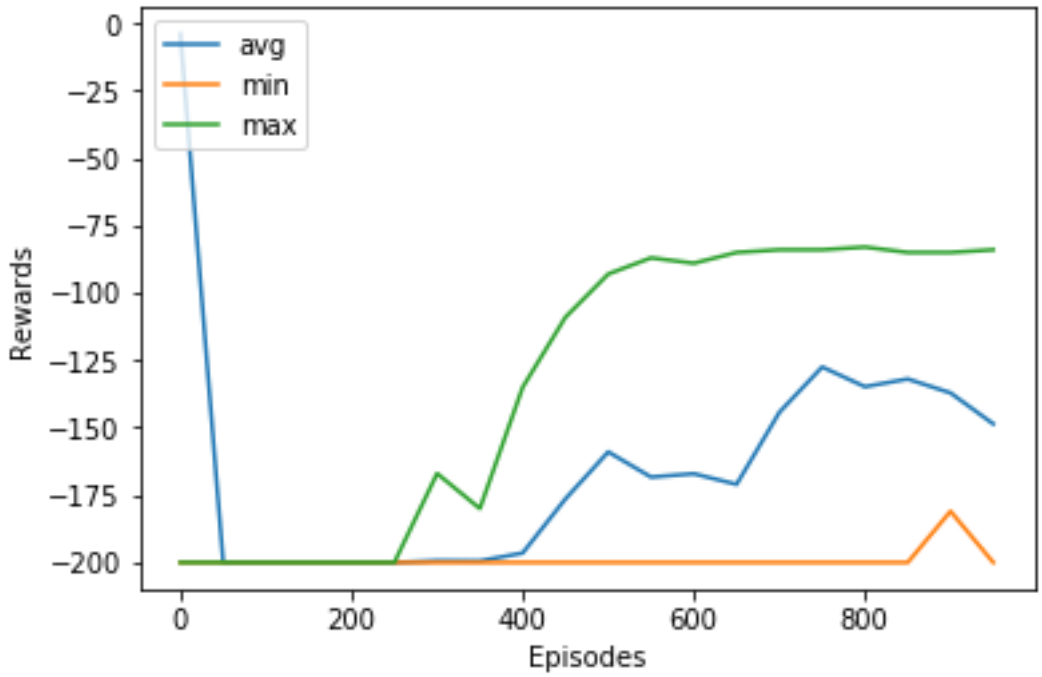
首先判定 memory 里记录数大于设定的下线，否则不训练

按照 batchsize 在 memory 里随机取出一部分记录作学习样本，

首先利用 prediction_model 对这些记录里的当前状态预测得到一个 Q1 表

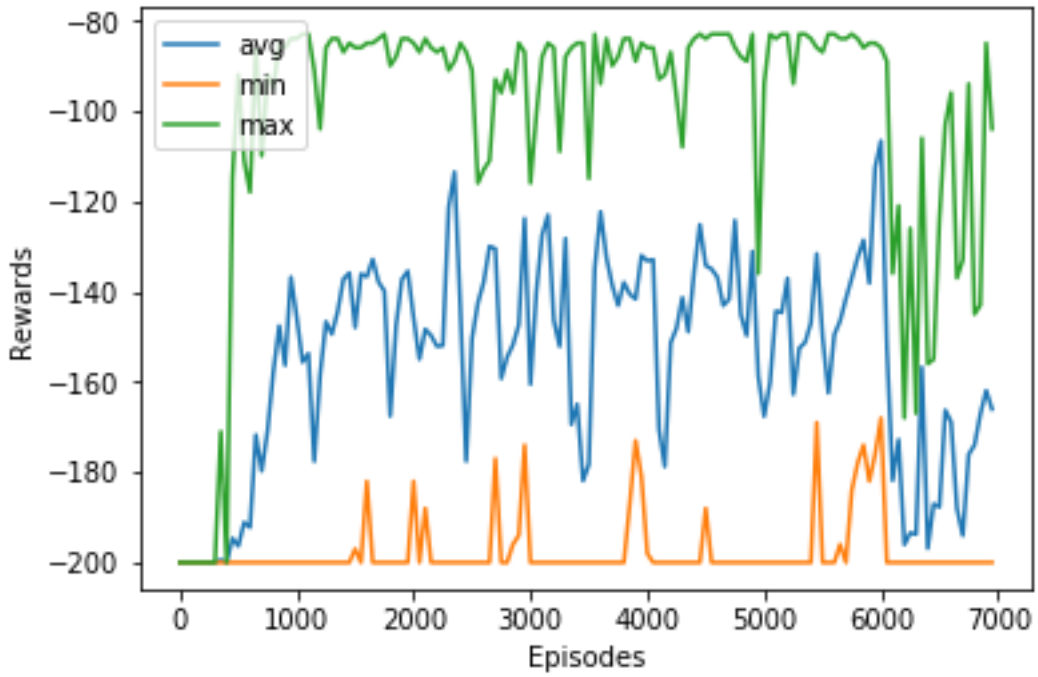
然后用 target_model 对这些记录里的下一个状态预测得到一个 Q2 表，选出每个状态各个动作的预计累计回报中最大的 Qmax，用 new_q = reward+gama*Qmax 来更新 Q1 表，然后把 Q1 表的状态作为 x_train,动作的预计累积回报作为 y_train 放入神经网络训练 prediction_model

如果当前步是 done 为真的步，则计数加一，计数达到设定的阈值时，将 prediction_model 的参数赋值给 target_model，并把计数清零。



最初的为什么那么高是因为我从 50 开始画的，所以最初的就是 0 了，所以从 50 往后看就行，是没问题的。

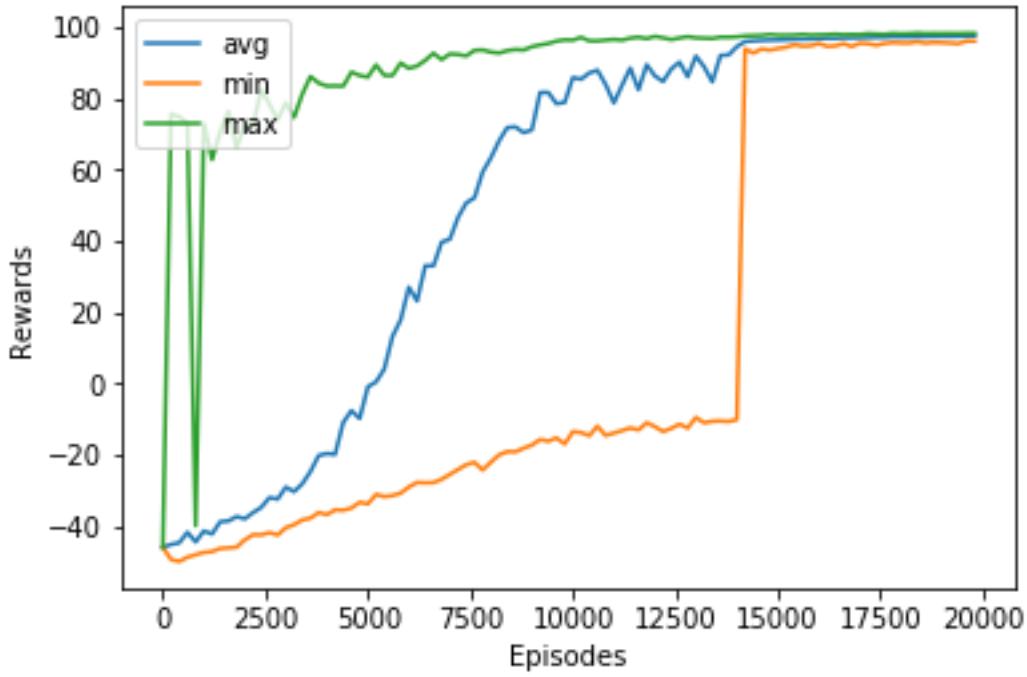
从 400 开始就学到东西了，但是可以看到 min 没有升起来，又尝试了下训练 7000 个轮次。



看到还是没有上升，由于这个跑起来比较慢，故不再尝试更多的轮次了。

用 rendering 看的话花了 149 步跑上去。

2.2.3 对连续的那道题：将动作离散为 6 个动作：-1 -0.6 -0.2 +0.2 +0.6 +1 然后 Q 表变为 20×20×6，其它的相似，然后要注意放到 step 模拟前根据下表 index 去查对应的 6 个动作里的哪个值，而且注意看环境的说明，这时候放的不是一个数，而是一个列表，比如 0.2 就要放[0.2]。还要注意一个问题，之前将状态离散为 20 个，现在显然不能这样，不然连续的速度变化显的没有意义了，因为位置一离散化很多速度的差异就消去了，所以将其设置为 200 个，则 Q 表变为 200×200×6 的。



14000 轮次左右之后可以看出训练的很好，min 值也上来了，非常好!

Rendering 看到 400 多步就上去了.

三.总结

相比于迷宫，这个小车作业免去了图像显示部分的工作，让我们有更多的时间去关注强化学习算法本身。不过一开始确实不知道怎么下手，只好去看各种英文资料，找现有的代码研究明白了再自己写。也算锻炼了自学和查英文资料的能力吧。还有建议课程删去部分前半学期的东西把这个大作业放到学期中，说实话，学期中强化学习的那几种方法基本上完全没学明白，就听着那么回事感觉了下，这次敲完代码思路清晰多了！总算学明白了

四.遇到的问题与解决，可能的改进

1.小车有时候上的去有时候上不去：就是训练次数不够，虽然均值上去了，但是最小值还在最底下彳亍，运气不好就上不去了

2.改为连续的后运行报错：一开始想当然每仔细研究环境变化，改了后的动作要求带【】，这样才能正常跑

3. 改为连续的后 Q 表学习不出来效果贼差：没有考虑到动作连续变化了对状态的离散化应当有更多的状态，否则不足以反映出速度间的差距

可能的改进：取出一些取到 max 时的 Q 表取平均作为最终 Q 表，预计可以让效果更好。