实验3. 强化学习实践

MF1733086，周小多， xiaoduo_zhou@163.com

2017 年 12 月 29 日

综述

本次实验为强化学习的实践，不同于传统的机器学习方式，强化学习无监督的模式，通过与环境的交互来更新状态动作函数，再利用状态动作函数依照当前的状态选择一个动作与环境交互，通过不断的交互最终得到一个在当前环境下比较好的策略。

实验二.

离散环境下的强化学习实现，分为两步，一是环境的离散化，二是Q-learning的实现：

环境离散化解决方案：

方法1 ：

CarPole 的observation 一共由四个值组，随机取一个的observation值观测 [1.19244749 1.6276515 0.17622697 0.3683256] ，我们可将环境值乘以 [100,100,1000,1000] 后将小数部分截取掉，这样将会得到一个离散的状态空间。但是此方法会导致状态空间不均匀，效果不好，故弃用。

方法2 ：

方法1扩大倍数再截掉小数简单粗暴了些，我们这里可以考虑一种均匀的分配状态空间的办法，利用环境的最大值减去最小值再将每个环境变量均分为N份，将环境值映射到0 100之间,这种方法十分公平，对于环境变量中每一个参数都能均匀划分，而且普遍适用于所有环境变量，至于N份到底取多少要看情况，在 CartPole-v0环境中小车的对动作的敏感度要求高状态空间应该大一些，本实验尝试了分为20,50,100,200份几种情况，最后发现分为100份成本较低效果最好，200份状态空间太大。MountainCar中也是100份比较好，而Acrobot分为100份时，Q表非常大训练了很久都不能收敛，故只分了20份。

Q-learning算法实现：

1、给定起始状态S,按ε概率随机选择状态S下的动作A，1-ε概率选择s状态下Q值最大的动作) 在状态S选择动作A。

2、在环境中执行动作A,得到奖励R和下一状态S'

3、 $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma max_a Q(S',A) - A(S,A)]$

4、 $S \leftarrow S'$

5、循环1、2、3、4直到S是terminal

超参设置及reward 均值和标准差：

CartPole-v0：

learning-rate = 0.01 、exploration-rate=0.1 、gamma = 0.98 、episodes = 7000、状态离散为100

测试20次，平均reward =427.4363654342102　标准差 =264.573229155853624

MountainCar：

learning-rate = 0.01 、exploration-rate=0.1 、gamma = 0.98 、episodes = 7000、状态离散为100

测试20次，平均reward = -131.25　标准差 = 27.291554506264774

Acrobot：

learning-rate = 0.01 、exploration-rate=0.1 、gamma = 0.95 、episodes = 3000、状态离散为20

测试20次，平均reward = -842.37　标准差 = 247.80479847661471

实验三.

连续环境下强化学习实现连续环境下的利用DQN算法实现如下：

1、初始化记忆D的容量为N

2、随机初始化Q网络的参数为Θ

3、获得初始状态s_1

4、按照ε方法将状态传入得到行动a_t,将a_t代入环境得到新的rewardr_t和下一个状态s_{t+1}

5、将(s_t, a_t, r_t, s_{t+1})存入记忆D中

6、从D中随机从记忆D中取mini-batch个记忆Non

7、$y_i = \begin{cases} r_j & Terminals_{t+1} \\ r_j + \gamma max_{a'} Q(s_{t+1}, a'; \theta))^2 & NotTerminals_{t+1} \end{cases}$

8、通过loss功能$(y_i - Q(s_j, a_j; \theta))^2$利用梯度下降更新$\theta$

三个任务超参设置及reward 均值和标准差：

网络结构为3层全连接网络，层与层之间激活函数为relu,隐层节点数为20，loss='mse'，optimizer=Adam(lr=learning-rate)

MountainCar参数及训练过程：

learning-rate = 0.001 、exploration-rate=0.1 、gamma = 0.98 、episodes = 1000

测试20次，平均reward = -131.25　标准差 = 27.291554506264774

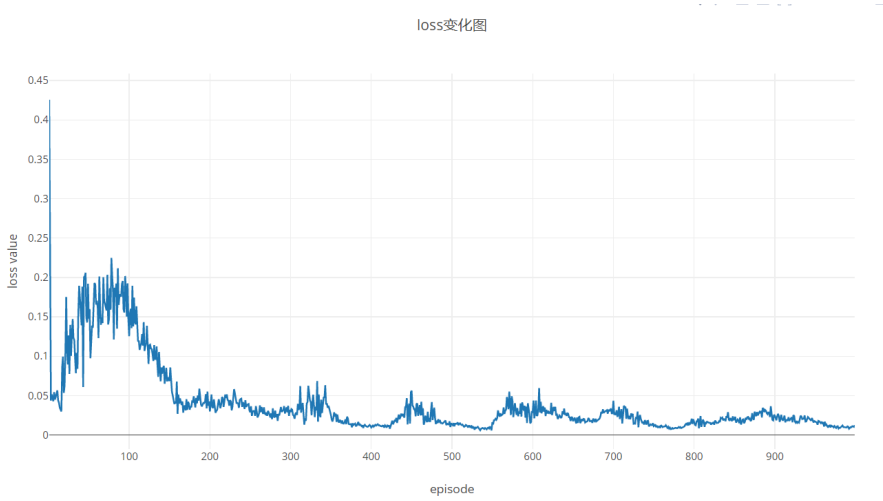

图 1: loss随训练轮数变化图

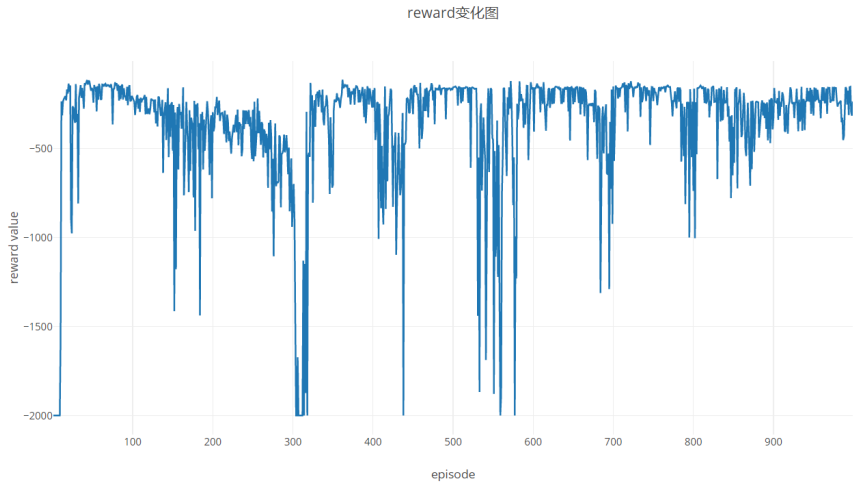

图 2: reward随训练轮数变化图

Acrobot参数及训练过程：

learning-rate = 0.001 、exploration-rate=0.1 、gamma = 0.9 、episodes = 1000

测试20次，平均reward =-627.04999999999995　标准差 = 301.39752993784066

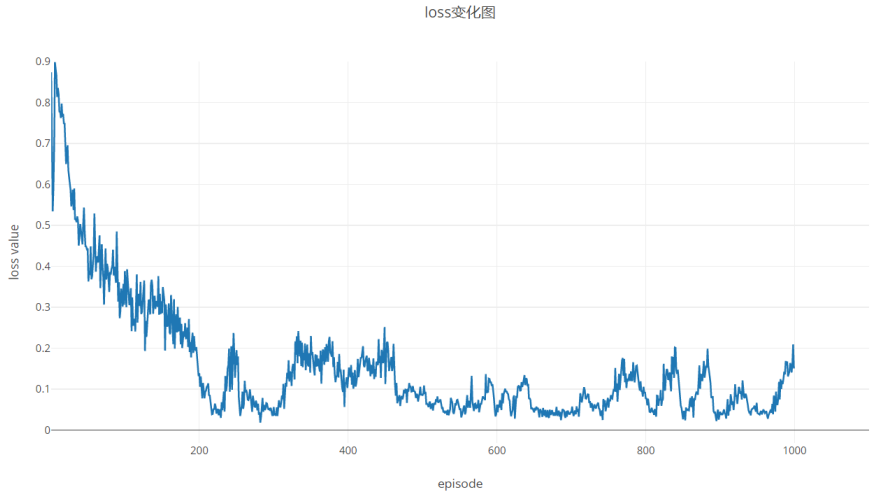

图 3: loss随训练轮数变化图

图 4: reward随训练轮数变化图

CartPole-v0 参数及训练过程：

learning-rate = 0.001 、exploration-rate=0.1 、gamma = 0.98 、episodes = 7000

测试20次，平均reward =147.2363194742106　标准差 =229.92547

注：CartPole使用DQN调过好多参数但是效果依然不是很理想

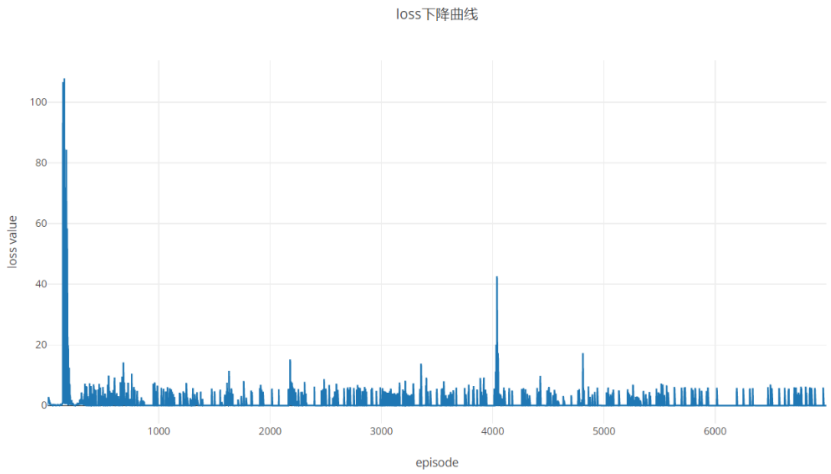

图 5: loss随训练轮数变化图

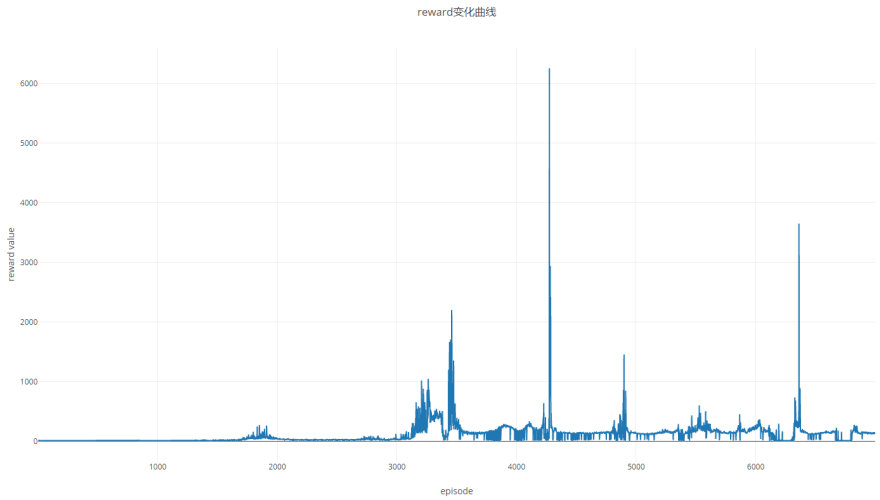

图 6: reward随训练轮数变化图

实验四.

改进版的DQN改进点主要在于每次不立即更新Q网络的θ值，而是几步过后才更新Q网络。

ImproveDQN算法实现如下：

1、初始化记忆D的容量为N

2、随机初始化Q网络的参数为θ

3、初始化$\hat{Q}$网络的参数为$\theta^{-}=\theta$

4、获得初始状态s_1

5、按照ε方法将状态传入得到行动a_t,将a_t代入环境得到新的rewardr_t和下一个状态s_{t+1}

6、将(s_t, a_t, r_t, s_{t+1})存入记忆D中

7、从D中随机从记忆D中取mini-batch个记忆Non

8、$y_i = \begin{cases} r_j & Terminals_{t+1} \\ r_j + \gamma max_{a'} \hat{Q}(s_{t+1}, a'; \theta))^2 & NotTerminals_{t+1} \end{cases}$

9、通过loss功能$(y_i - Q(s_j, a_j; \theta))^2$利用梯度下降更新$\theta$

10、每C步就利用Q网络更新$\hat{Q}$

超参设置同实验三，训练过程及reward 均值和标准差如下：

MountainCar测试及训练过程：

测试20次，平均reward = -213.09999999999999　　标准差 = 150.50752735153708

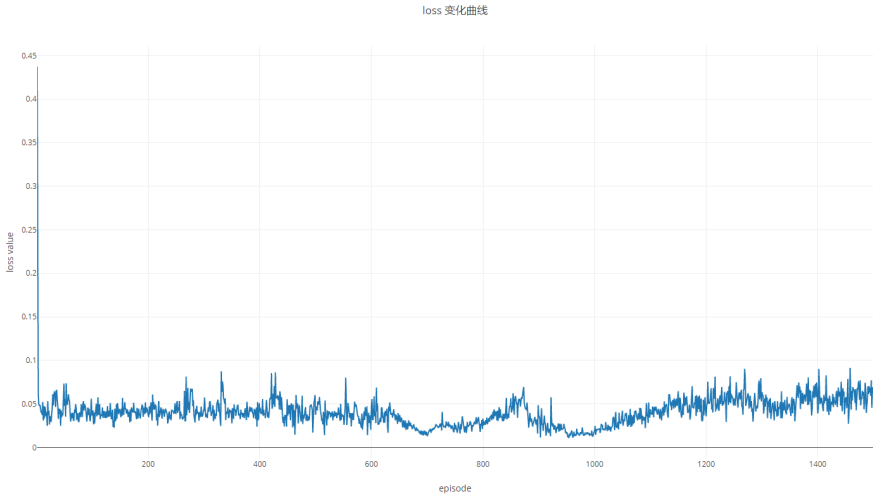

图 7: loss随训练轮数变化图

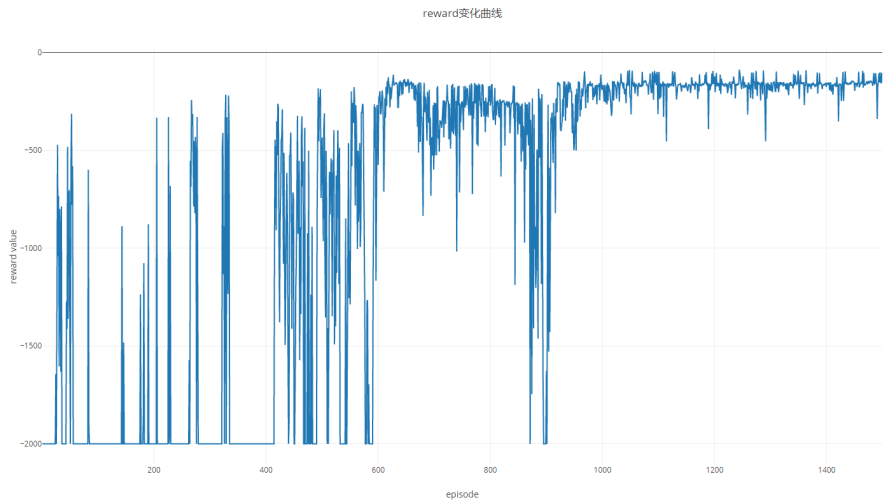

图 8: reward随训练轮数变化图

Acrobot测试及训练过程：

测试20次，平均reward =-543.95000000000005　标准差 = 190.80479829561827

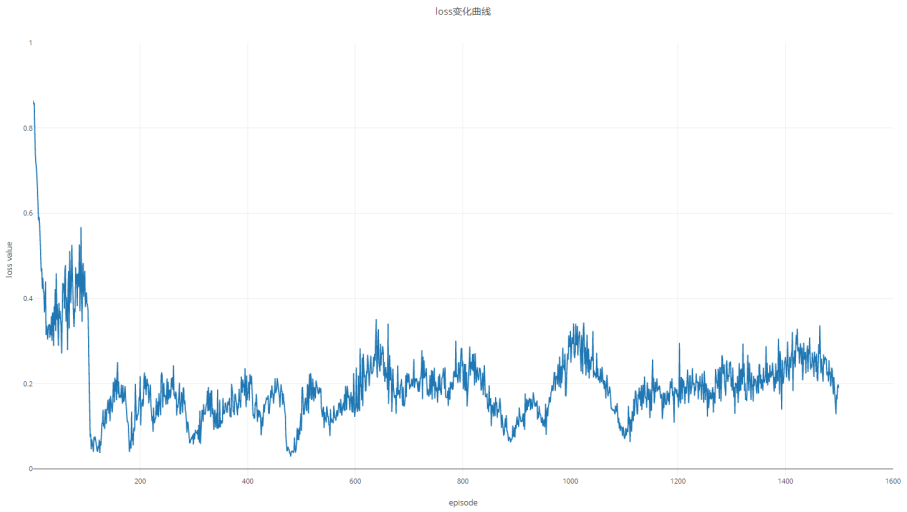

图 9: loss随训练轮数变化图

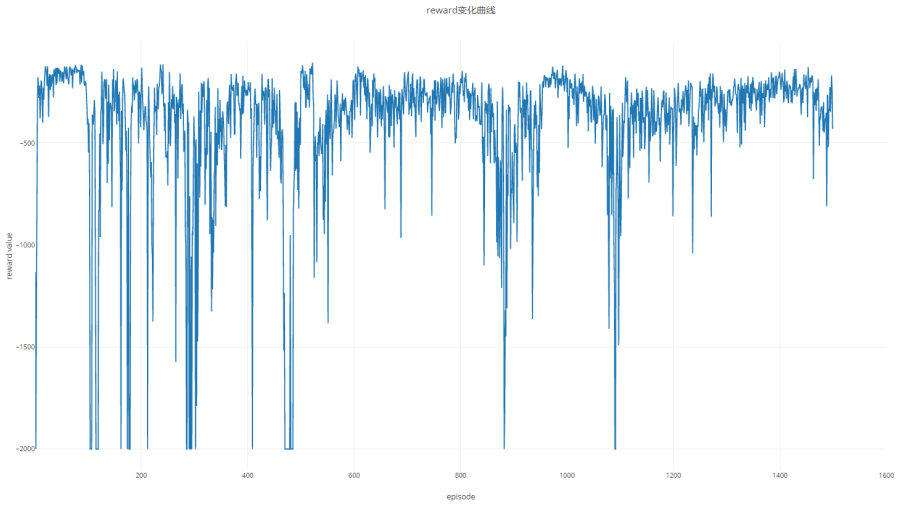

图 10: reward随训练轮数变化图

CartPole-v0 测试及训练过程：
测试20次，平均reward =20000　标准差 =0

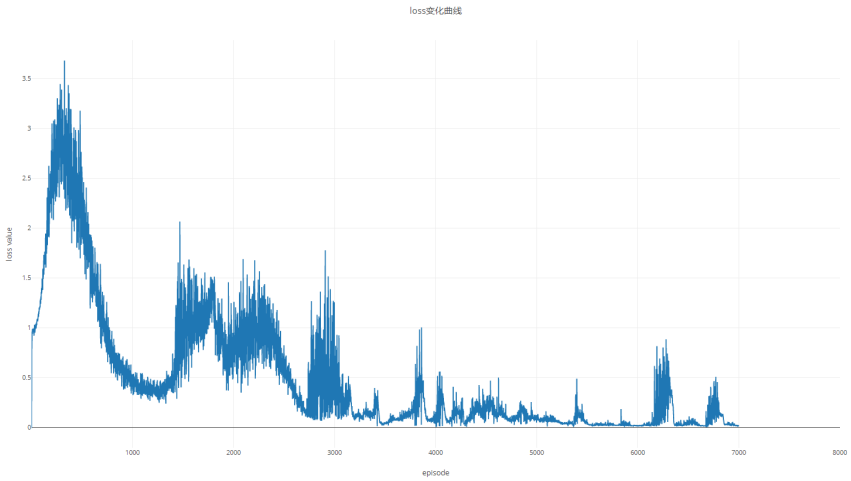

图 11: loss随训练轮数变化图

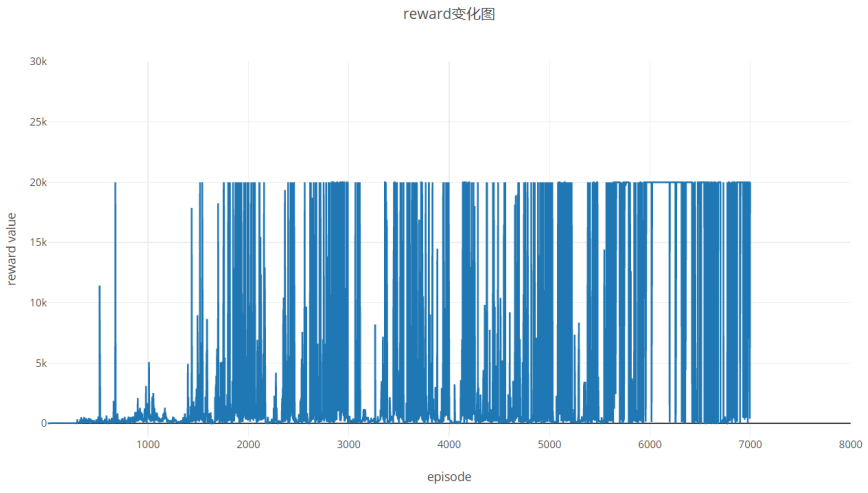

图 12: reward随训练轮数变化图

本方法对DQN做了一个小小的改动，把原先的Q网络又复制了一份$\hat{Q}$，利用旧的网络的$\hat{Q}$的值进行loss计算，多步之后更新$\hat{Q}$。这样做的好处就是减少了目标计算和当前值的相关性，因为DQN的目标Q网络变化比较动态，这样对计算目标Q不利，所以延迟更新是个好办法。

以MountainCar为例,下列两个reward图一个是DQN的，一个是ImproveDQN的，我们可以看到DQN后期波动还是很大，因为Q网络实时更新，马上就影响到了性能，而Imporve DQN后期越来越稳定得益于延迟更新。

图 13: DQN reward

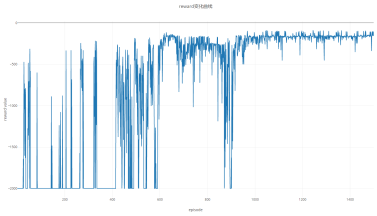

图 14: Improve DQN reward