实验 3. 强化学习实践

MF1733040，刘兆洋，zyliumy@gmail.com

2017 年 12 月 28 日

综述

最近两年人工智能被大众广泛谈论的例子莫过于 AlphaGO 战胜了世界围棋冠军李世石。这让更多的人认识到了人工智能领域的发展现状。AlphaGO 是 DeepMind 公司研究开发出来的，主要利用了强化学习的知识。

强化学习是个重要的机器学习方法，与传统的监督学习和无监督学习不同的是，强化学习是一个与环境不断进行交互，并从中获得反馈的过程。它没有现成的 label，所有的数据都得靠自己去探索，从获得的奖励与惩罚中学习最优策略。

强化学习的过程通常用马尔科夫过程来描述 [**?**]：机器处于环境 E 中，状态空间为 X，其中每个状态 x∈X 是机器感受的环境描述，机器所能采取的动作构成了动作空间 A，当机器处于 x 状态下采取了某个动作 a∈A, 潜在的转移函数 P 将使得当前环境状态转移到另一个状态，转移的同时，环境会根据潜在的奖赏函数 R，反馈给机器一个奖赏。

本次强化学习的作业是基于 OpenAI 开源出来的 Gym，Gym 中包含很多现成而且有趣的实验环境。我们需要解决的问题包括 CartPole-v0、MountainCar-v0 和 Acrobotv1。首先我先简单的介绍一下这三个环境。CartPole-v0 是一个保持平衡的游戏，一个杆立在小车上，可以通过移动小车，使杆保持平衡，当杆的角度大于 $\pm 12^\circ$ 或小车偏离屏幕中心 2.4 单位距离时，游戏就终止了。MountainCar-v0 是一个小车在类似 U 型的谷底不断来回震荡，通过给小车一个力，让小车可以通过右边谷顶的红旗，此时问题解决。Acrobotv1 是由两个杆连在一起，目的是通过挥动下面的杆，让它晃动到某个高度，此时问题被解决。

下面将详细介绍问题的解决过程。

实验二.

实验二是使用 Q-learning 求解 CartPole-v0、MountainCar-v0 和 Acrobotv1 问题。Gym 提供了 CartPole-v0、MountainCar-v0 和 Acrobotv1 的环境，动作空间是离散的，但状态空间是连续的，因此，如果要使用 Q-learning 求解这些问题，首先我们得将状态空间离散化。至于怎么离散化，我将依照不同的环境，逐一介绍。

1.CartPole-v0

CartPole-v0 的状态空间有四维，每个维度依次为小车的位置 ∈[-2.4,+2.4]、小车的速度 $\in [-\infty, +\infty]$、杆的角度 $\in [-41.8^\circ, +41.8^\circ]$、杆顶端的速度 $\in [-\infty, +\infty]$。按照停止条

件，我将车的位置离散成了 24 段，[-2.4, +2.4] 被均匀的分成 24 段，$[-\infty, -2.4)$ 和区间 $[+\infty, +2.4)$ 共 2 段。由于车的速度并不会直接导致环境终止，故我只简单将 [-1, +1] 均匀分成 5 段，再加上区间之外的 2 段。杆的角度在 [-0.3, +0.3]，区间内分成了 12 段和区间外 2 段。杆顶端速度角度和终止条件没有直接关系，因此我将 [-1.5, +1.5] 分成了 6 段，区间外 2 段。

在具体的实现程中，我的参数设置为：episodes = 500000，alpha = 0.5，gamma = 0.99，exploration_rate = 0.5，exploration_rate_decay = 0.6。各个参数的含义分别是训练的 episodes，alpha 为折扣，gamma 为折扣，exploration_rate 是 ϵ-贪心策略中的探索概率，而 exploration_rate_decay 为衰减速率。

我每训练 2000 个 episodes 后（如果每个 episode 都测试一下，不仅训练慢，并且会导致画出的曲线过于稠密而不能看出有效的信息），就测试一次，已检测学习的效果，如图 1。另外我还画出了训练完成后，测试阶段 reward 可视化结果，如图 2。

我在训练完成后，进行了 20 次测试, reward 的最大值、均值和标准差分别为：max rewards = 20000.0, means = 1133.58, std = 2796.8559247126045 ....

2.MountainCar-v0

MountainCar-v0 的状态只有 2 维，第一个是小车的位置 position$\in [-1.2\ 0.6]$，第二个是小车的速度 velocity$\in [-0.07, 0.07]$。我将小车的位置在 [-1.2，0.6] 区间均匀分成 18 段，区间外共 2 段。小车的速度在 [-0.07, 0.07] 均匀分成 10 段，区间外工 2 段。

在具体的实现程中，我的参数设置为：episodes = 2000，alpha = 0.07，alpha_decay = 0.8，gamma = 0.99，exploration_rate = 0.4，exploration_rate_decay = 0.95。各个参数的含义分别是训练的 episodes，alpha 为学习率，alpha_decay 为 alpha 的衰减速率，gamma 为折扣，exploration_rate 是 ϵ-贪心策略中的探索概率，而 exploration_rate_decay 为衰减速率。

我每训练 10 个 episodes 后（理由同上），就测试一次，来检测学习的效果，如图 3。

另外我还画出了训练完成后，测试阶段 reward 可视化结果，如图 4。我在训练完成后，进行了 100 次测试, reward 的最大值、均值和标准差：max rewards = -149.0，means = -590.97，std = 741.8077035863134

3.Acrobotv1

Acrobotv1 的状态略多,有 6 个,分别是 [cos(theta1),sin(theta1),cos(theta2),sin(theta2), thetaDot1，thetaDot2], theta1 和 theta2 分别是两个杆的角度，thetaDot1、thetaDot2 分别两个杆的角速度。我对前四个状态的离散方法是一样的，在 [-1, 1] 之间均匀分成 10 段，区间外分成 2 段。第五个状态在 [-12.56637061, 12.56637061] 区间内分成 25 段，区间外 2 段。第六个状态 [-28.27433388, 28.27433388] 区间内分成 56 段，区间外 2 段。

在具体的实现程中，我的参数设置为：episodes = 5000，alpha = 0.2，gamma = 0.99，exploration_rate = 0.4，exploration_rate_decay = 0.95。各个参数的含义分别是训练的 episodes，alpha 为折扣，gamma 为折扣，exploration_rate 是 ϵ-贪心策略中的探索概率，而 exploration_rate_decay 为衰减速率。

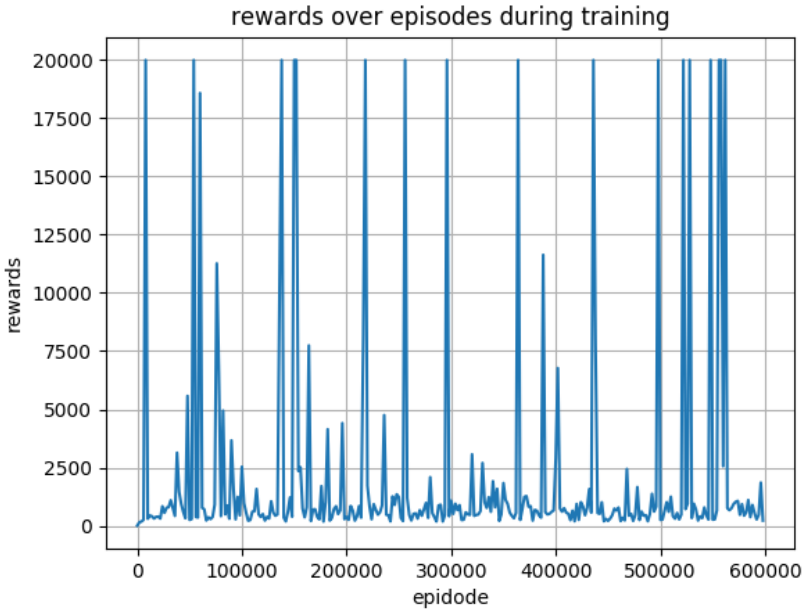

图 1: CartPole-v0 训练期间 rewar 随着 episode 变化曲线

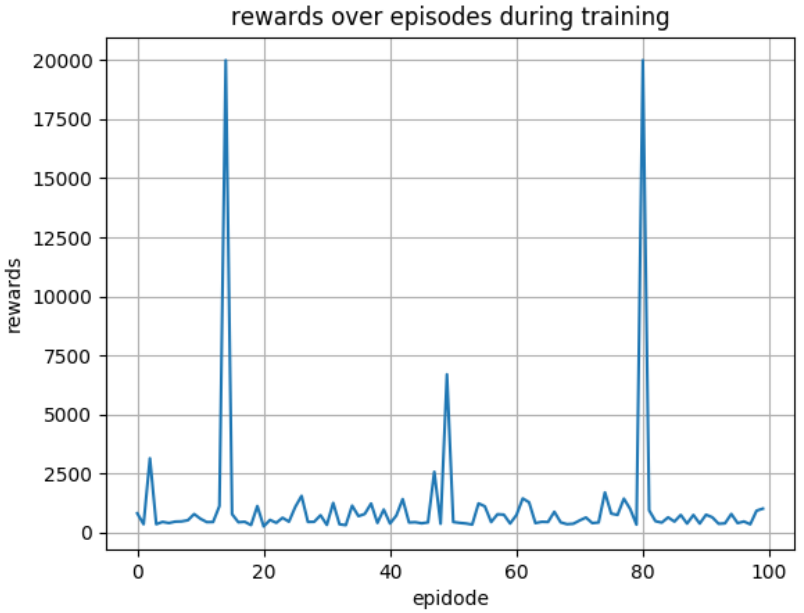

图 2: CartPole-v0 测试期间 rewar 随着 episode 变化曲线

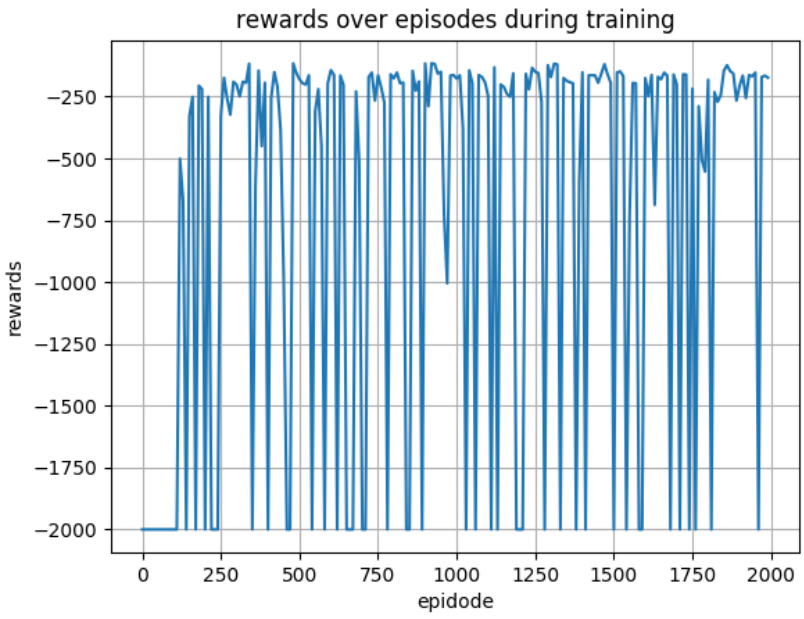

图 3: MountainCar-v0 训练期间 rewar 随着 episode 变化曲线

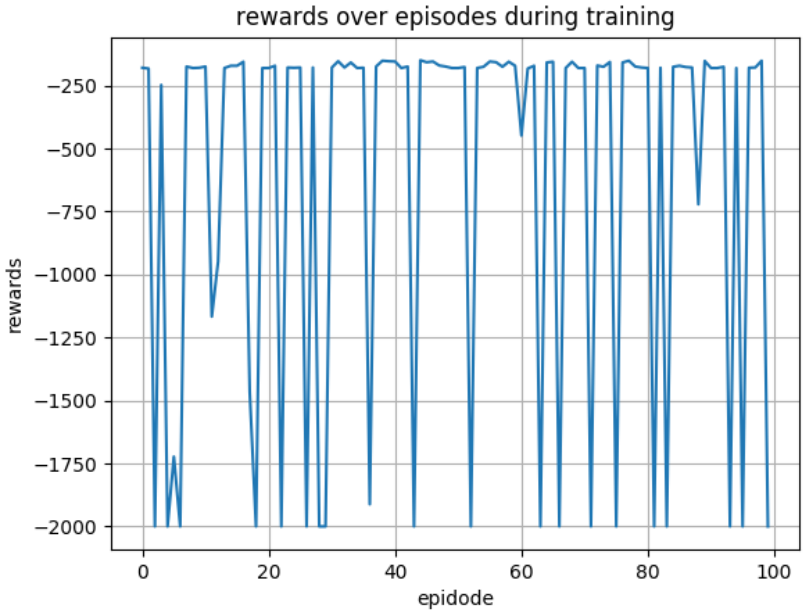

图 4: MountainCar-v0 测试间 rewar 随着 episode 变化曲线

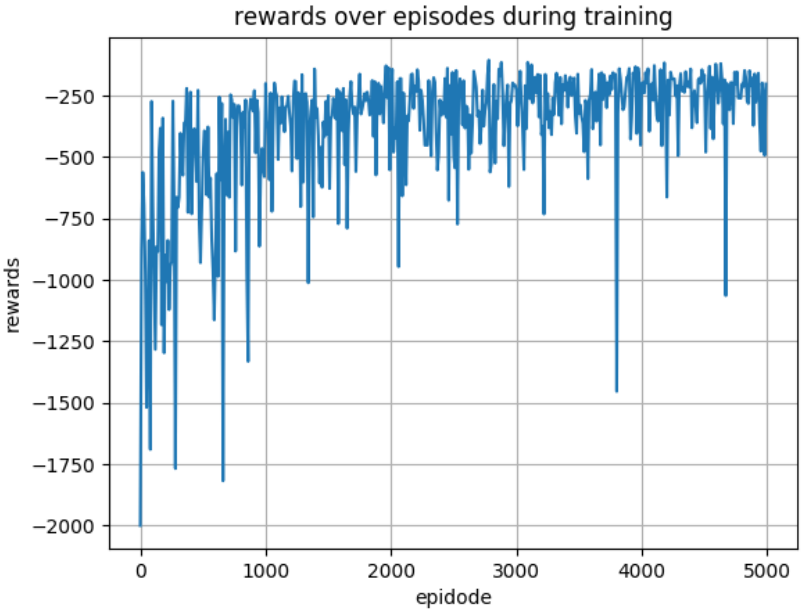

图 5: Acrobotv1 训练期间 rewar 随着 episode 变化曲线

图 6: Acrobotv1 测试间 rewar 随着 episode 变化曲线

可视化结果：

我在训练完成后，进行了 20 次测试, reward 的均值和标准差为：max_rewards = -107.0, means = -250.2, std = 121.52769231743028

Q-learning 算法简介 以上就是我在实验二实现 Q-learning 的过程中，对状态离散的具体方法。下面简单的介绍一下 Q-learning 算法。Q-learning 的训练过程中，机器处在状态 x 下，先采取 ϵ-贪心策略选取一个动作执行，之后跳转到另一个状态 x'，并获得奖励 r，由于 Q-learning 是 off-policy，故选取概率最大的动作 a'，然后采用下面的公式 (1) 更新 Q 函数。

$$Q(x,a) = Q(x,a) + \alpha * (r + \gamma * Q(x',a') - Q(x,a)) \tag{1}$$

进行数次迭代后，动作值函数 Q 会逐渐收敛。具体如何实现 Q-learning，请参考 [?] 书中关于强化学习的内容。参数设置

实验三.

DQN 算法简介 DQN 被称作深度 Q 网络，首先将状态输入到一个深度神经网络，然后用神经网络的输出作为 Q 值，神经网络的参数为 w，

$$Q(s,a,w) \approx Q(s,a) \tag{2}$$

然后在 Q 值中使用均方差 mean-square error 来定义目标函数 objective function，也就是 loss function：

$$L(w) = E[(r + \lambda max_{a'} Q(s',a',w) - Q(s,a,w))^2] \tag{3}$$

接着对 L(w) 求导，就会得到：

$$\frac{L(w)}{w} = E[(r + \lambda max_{a'} Q(s',a',w) - Q(s,a,w)) \frac{\partial Q(s,a,w)}{\partial w}] \tag{4}$$

最后再使用我们熟悉的梯度下降更新参数，不断的迭代，直到收敛，就可以求得最优策略 [?]。

环境的状态并不是图片，所以并没有用到 CNN。我的网络结构很简单，就是 MLP。我的网络有三层，第一层网络是 state_num*24，第二层是 24*48，第三层是 48*action_num，将状态输入进去，输出每个动作的分数，激活函数使用的 ReLU，网络参数的初始化使用了 Xavier 随机初始化方法。在训练的过程，做了一点稍微的改动。按照原始的论文描述的那样，每执行一次动作就将 transition 四元组存储起来，然后就更新网络。但在实验的过程中，我发现这样做不仅会导致 loss 上下波动很大，而且训练很慢，迟迟不能收敛。在后面我将介绍我的改进，并以 cartpole 为例，进行对比。

1.CartPole-v0

参数设置如下：gamma = 0.95，epsilon = 0.6，learning_rate = 0.001，BATCH_SIZE = 64

我首先介绍一下自己改进后 DQN 训练情况。图 7 展示了网络训练误差随训练轮数的变化关系 (x 轴为训练轮数，y 轴为每轮中训练误差的均值) 图 8 展示了随着训练次数的增

加，每隔 10 个 episode，reward 的变化情况。图 9 展示了在测试阶段 reward 随着 episode 的变化曲线，可以看到是一条直线，说明学习到的策略很优秀。我在训练完成后，进行了 20 次测试, reward 的最大值、均值和标准差：max rewards = 20000，means = 20000，std = 0

然后再来看未改进之前的 DQN 具体情况，未改进的 DQN 和改进的 DQN，在参数方面是一致的，尽量减少其他因素的干扰。我们仅仅只需要看图 9 与图 12，就能看到测试阶段，改进的和未改进的 DQN 的差距。改进后的 DQN，在测试阶段一直保持 20000 的 reward，而未改进的起伏比较大，无论是均值，方差都远远不如改进后的 DQN。当然，改进后的 DQN 偶尔也会不稳定，但在大多数情况下都要比未改进的好。

2.MountainCar-v0

MountainCar-v0 和 Acrobotv1 都将使用改进后的 DQN 作为 basenet。参数设置如下：gamma = 0.99，epsilon = 0.5，learning_rate = 0.001，replay_size = 5000 BATCH_SIZE = 64 具体效果请看：图 13，图 14，图 15，含义与上面 cartpole 一样。

我在训练完成后，进行了 20 次测试, 效果并不好，也懒得调参了，reward 的最大值、均值和标准差：max rewards = -2000，means = -2000，std = 0。

3.Acrobotv1

参数设置如下：gamma = 0.99，epsilon = 0.5，learning_rate = 0.01，replay_size = 500 BATCH_SIZE = 64 具体效果请看：图 13，图 14，图 15，含义与上面 cartpole 一样。

我在训练完成后，进行了 20 次测试, 效果并不好，同样没有调参，调来调去的意义不大。reward 的最大值、均值和标准差：max rewards = -2000，means = -2000，std = 0。

实验四.

improved DQN 算法简介 实验三用的 DQN 有个很大的缺点，有时候需要经过长时间的调参，才能使模型正常运行起来，大多数情况下，都是很难收敛的。基于实验三的 DQN，主要做了以下改进，首先用了两个网络，第一个网络参数为 θ_i^-，用来更新目标值 y：

$$y = r + \gamma max_{a'} Q(s', a', \theta_i^-) \tag{5}$$

第二个网络的参数为 θ_i，此时，实验三的 loss 就变成了：

$$L_i(\theta_i) = E_{s,a,r,s'}[(y - Q(s, a, \theta_i))^2] + E_{s,a,r}[V_{s'}[y]] \tag{6}$$

当优化 $L_i(\theta_i)$ 时，会固定住上一次迭代中的 θ_i^- 而在实验三的 DQN 中每次更新完参数 θ_i 后，都会更新 θ_i^-，这会导致学习到的策略上下波动很大。改进后 DQN 每 C 次更新完 θ_i 后才更新 θ_i^-。具体的改进方法请参看 [**?**]。

1.CartPole-v0

参数的设置基本与 DQN 中的一样。

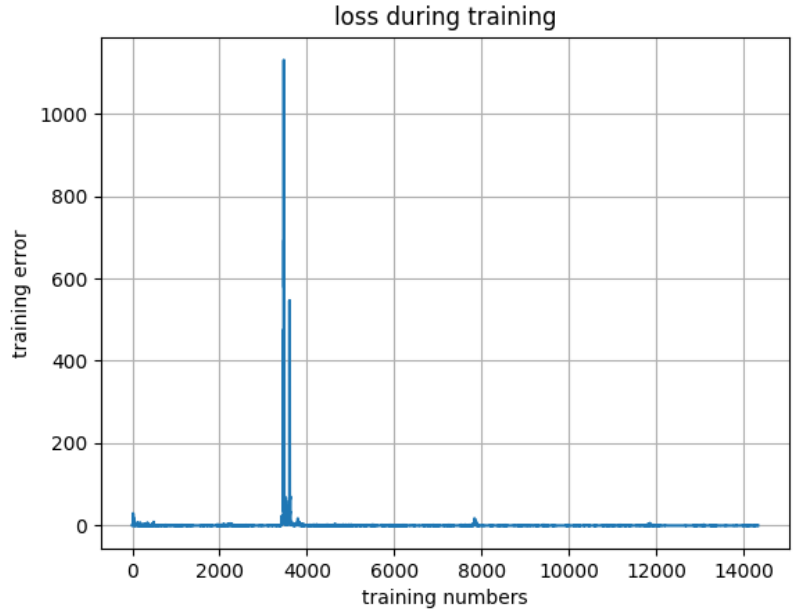

图 7: CartPole-v0 训练期间 loss 随着 episode 变化曲线

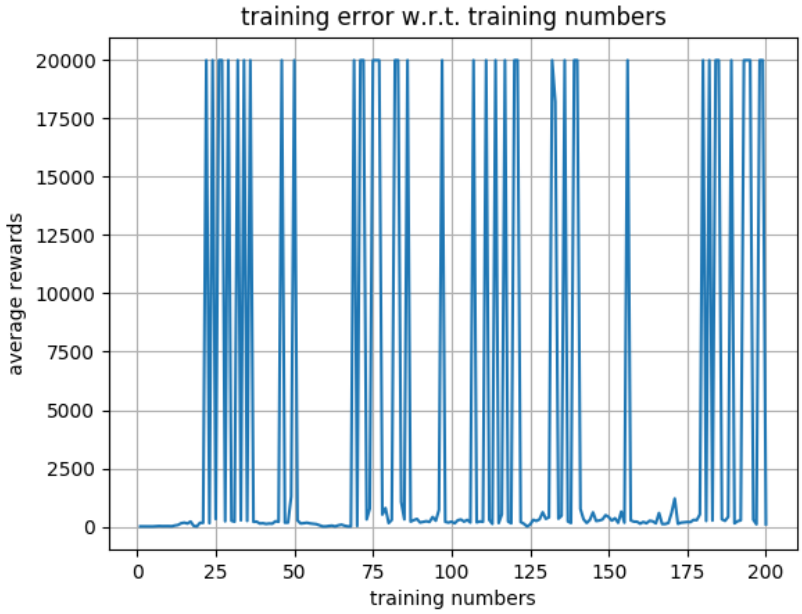

图 8: CartPole-v0 训练期间 reward 随着 episode 变化曲线

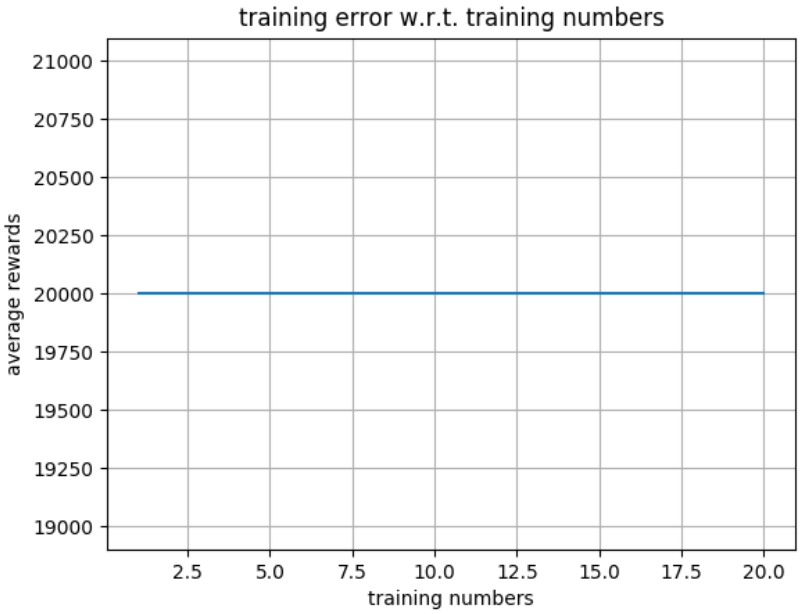

图 9: DQN 中 CartPole-v0 测试间 rewar 随着 episode 变化曲线

我在训练完成后，进行了 20 次测试。reward 的最大值、均值和标准差：max rewards = 1044.0，means = 746.25, std = 185.58660377300941。

可视化请见图 19，图 20，图 21。具体的改进请看与实验三对比。

2.MountainCar-v0

epsilon = 0.5，learning_rate = 0.001，replay_size = 5000，BATCH_SIZE = 64，TRAIN_EPISODE = 2000。

我在训练完成后，进行了 20 次测试。reward 的最大值、均值和标准差：max rewards = -2000.0 means = -2000.0, std = 0.0。

可视化请见图 22，图 23，图 24。具体的改进请看与实验三对比。

3.Acrobotv1

参数设置：gamma = 0.99，epsilon = 0.5，learning_rate = 0.02，replay_size = 500，BATCH_SIZE = 64，TRAIN_EPISODE = 3000。我在训练完成后，进行了 20 次测试。reward 的最大值、均值和标准差：max rewards = -2000.0，means = -2000.0, std = 0.0。

可视化请见图 25，图 26，图 27。具体的改进请看与实验三对比。

总结

完成试验后，虽然很累，但收货颇丰。

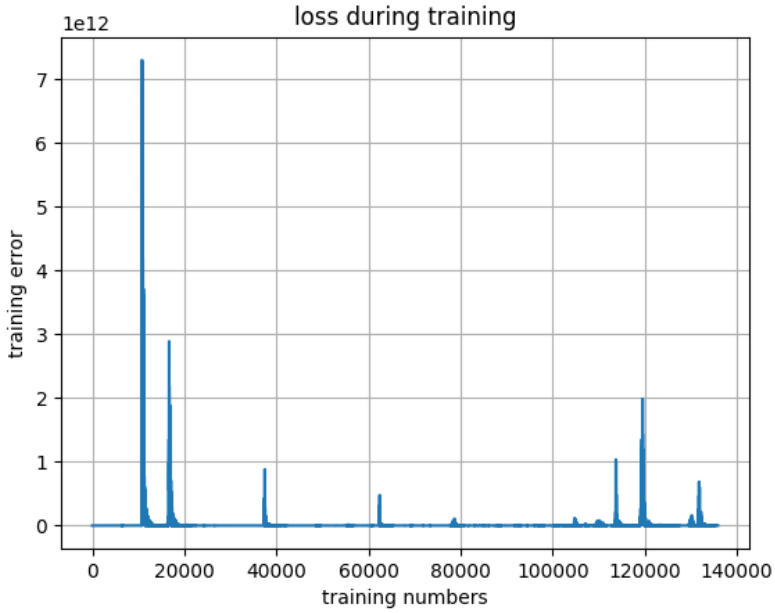

图 10: 未改进的 DQN 中，CartPole-v0 训练期间 loss 随着 episode 变化曲线

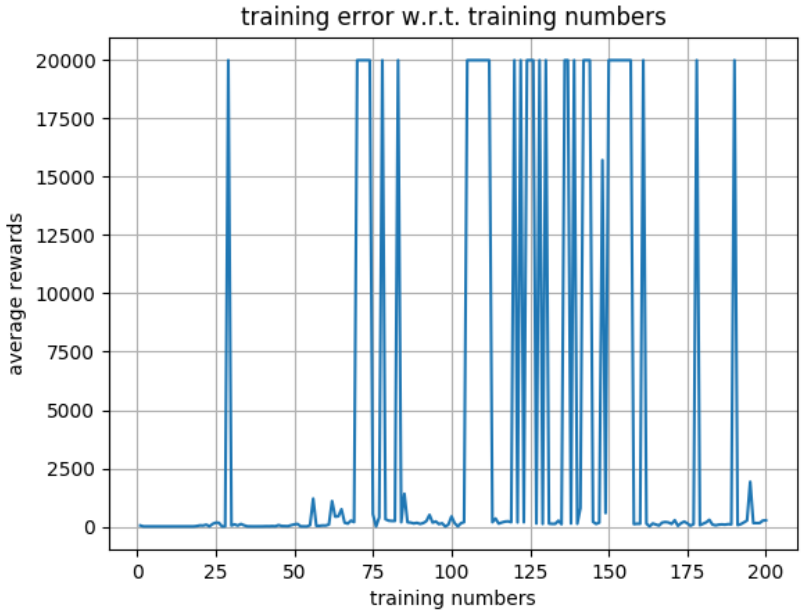

图 11: 未改进的 DQN 中，CartPole-v0 训练期间 reward 随着 episode 变化曲线

图 12: 未改进的 DQN 中，CartPole-v0 测试间 rewar 随着 episode 变化曲线

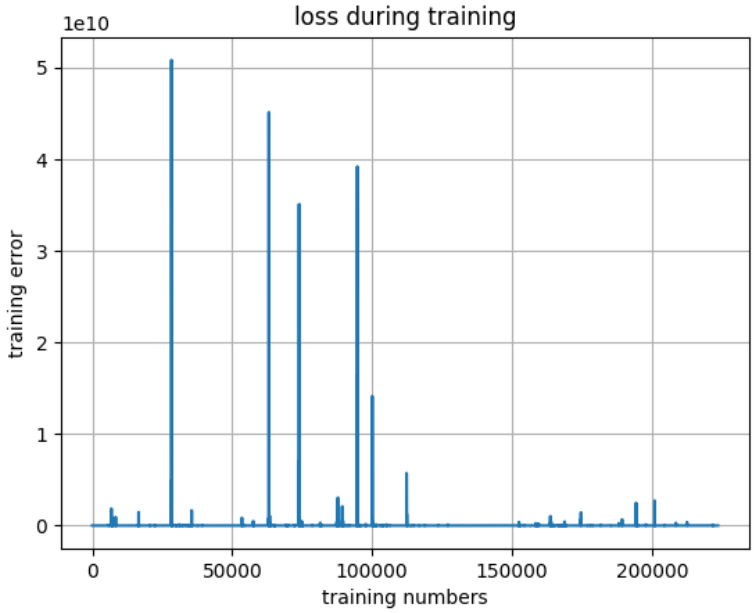

图 13: MountainCar-v0 训练期间 loss 随着 episode 变化曲线

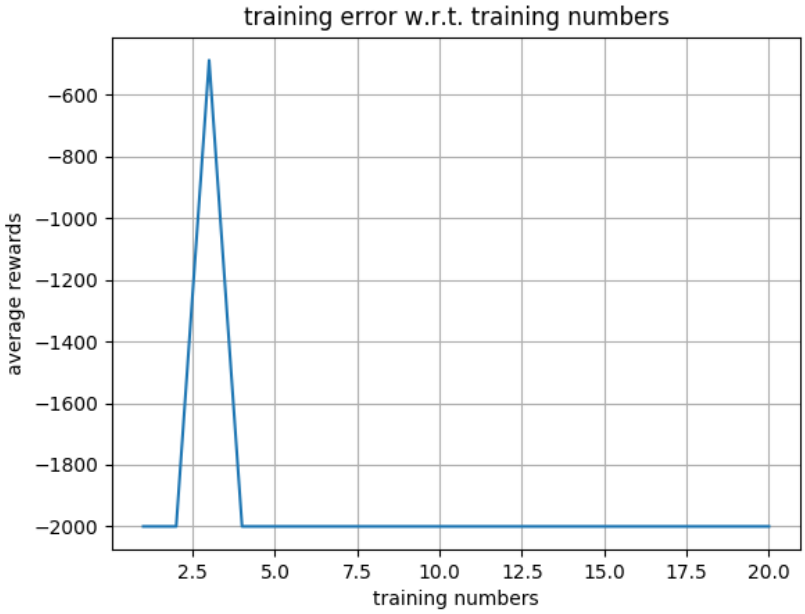

图 14: MountainCar-v0 训练期间 reward 随着 episode 变化曲线

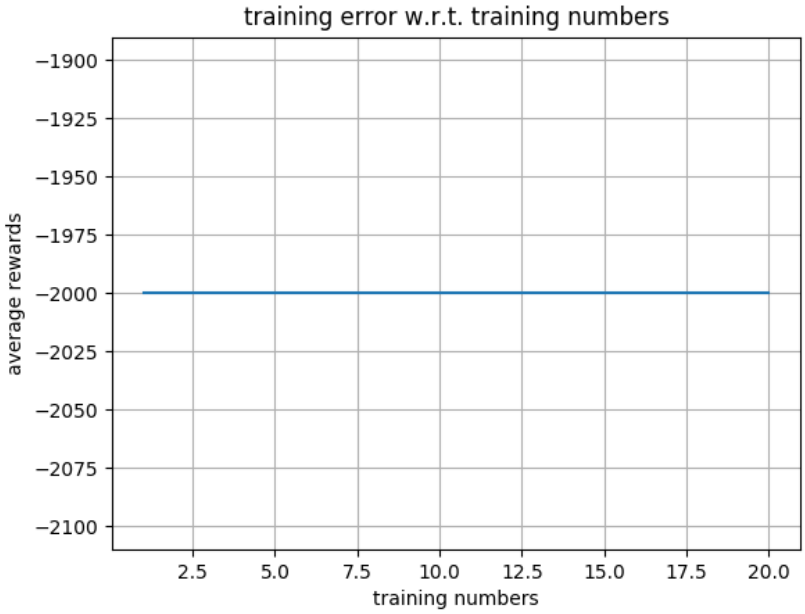

图 15: DQN 中 MountainCar-v0 测试间 rewar 随着 episode 变化曲线

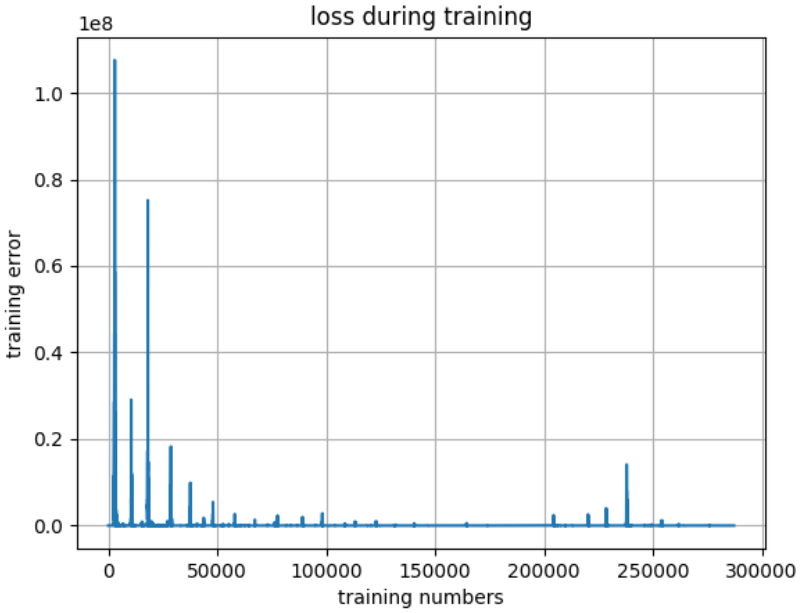

图 16: DQN 中 Acrobotv1 训练期间 loss 随着 episode 变化曲线

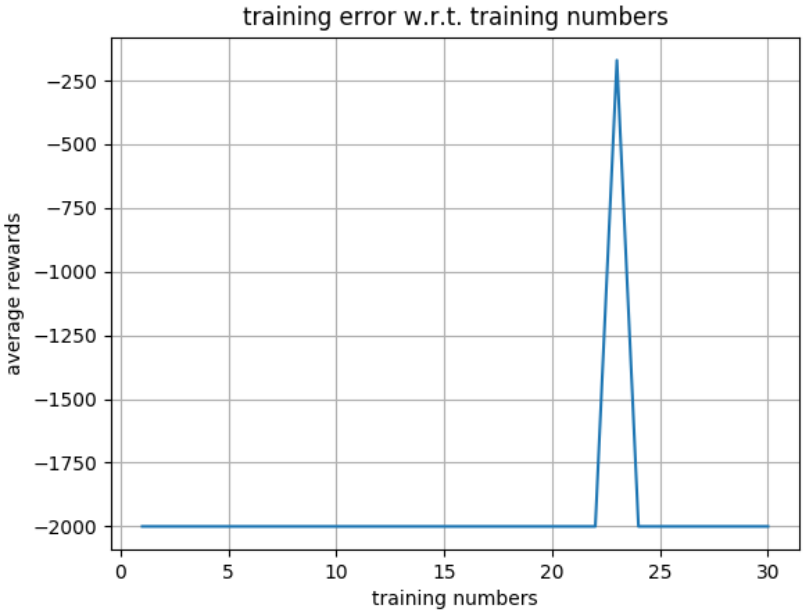

图 17: DQN 中 Acrobotv1 训练期间 reward 随着 episode 变化曲线

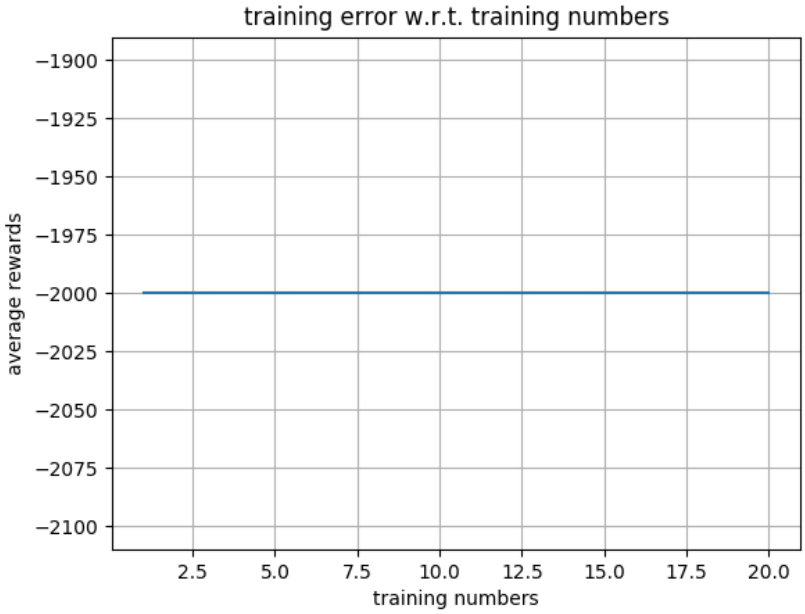

图 18: DQN 中 Acrobotv1 测试间 rewar 随着 episode 变化曲线

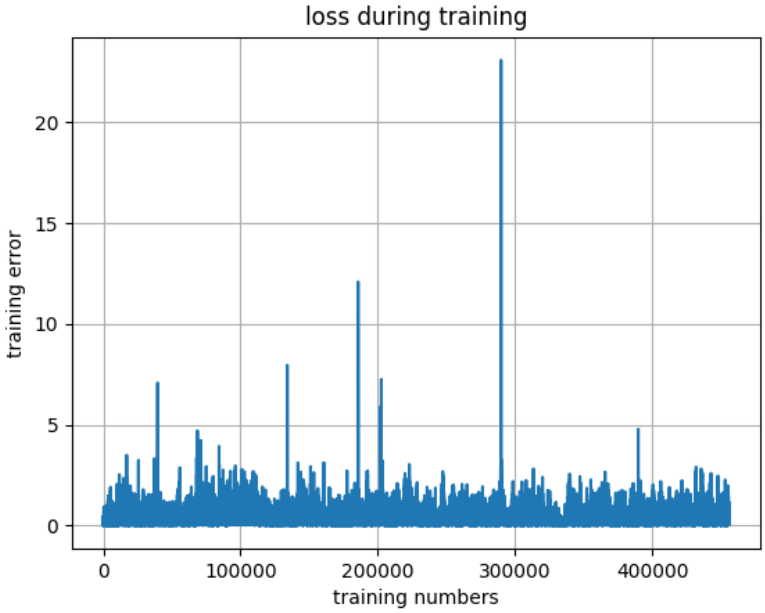

图 19: improved DQN 中，CartPole-v0 训练期间 loss 随着 episode 变化曲线

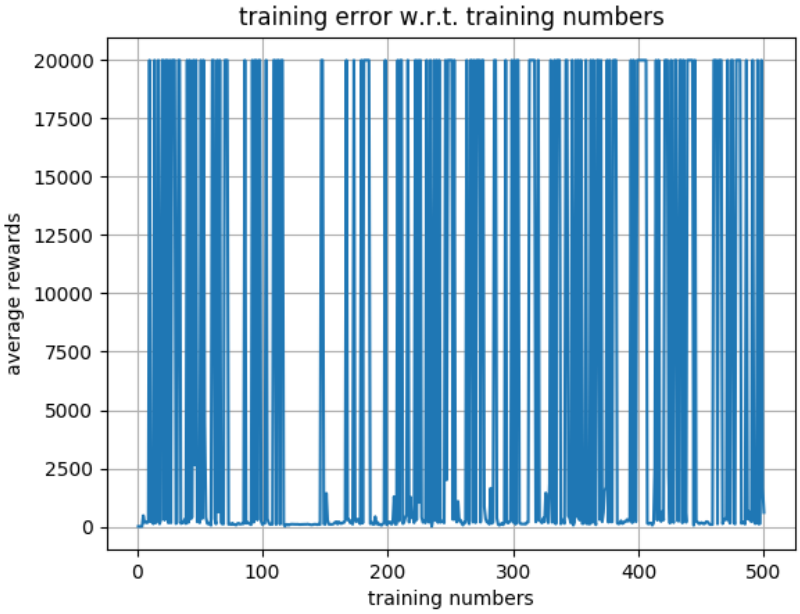

图 20: improved DQN 中，CartPole-v0 训练期间 reward 随着 episode 变化曲线

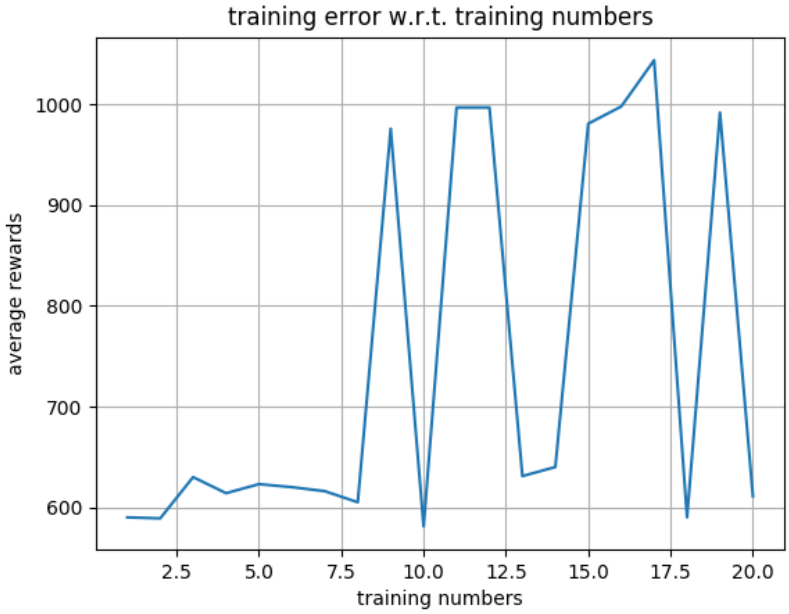

图 21: improved DQN 中，CartPole-v0 测试间 rewar 随着 episode 变化曲线

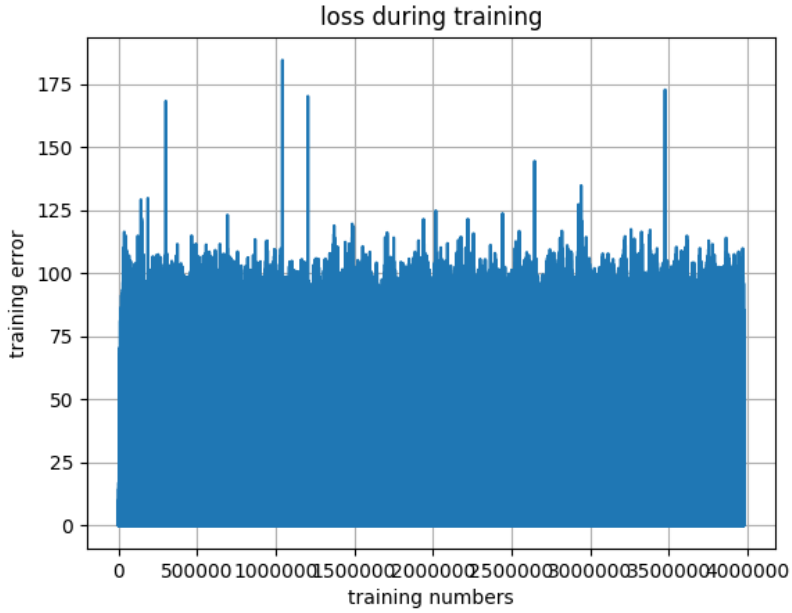

图 22: improved DQN 中，CartPole-v0 训练期间 loss 随着 episode 变化曲线

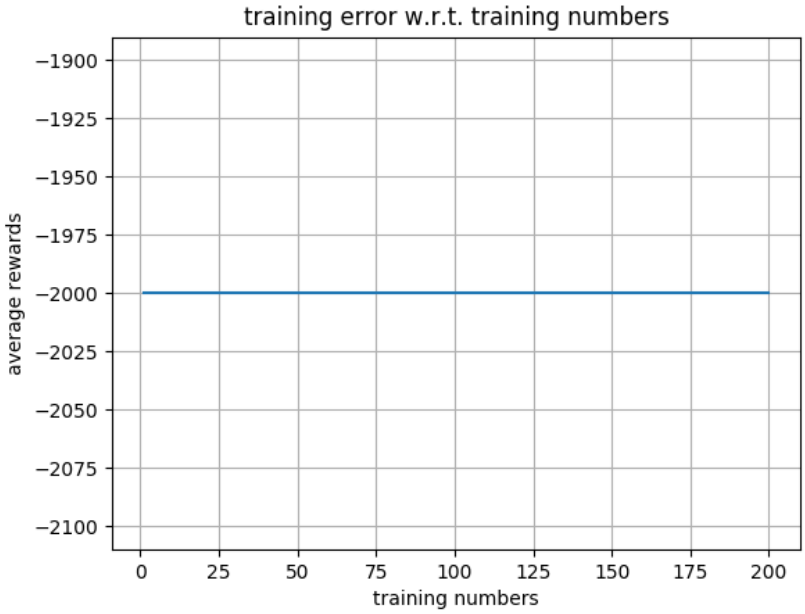

图 23: improved DQN 中，CartPole-v0 训练期间 reward 随着 episode 变化曲线

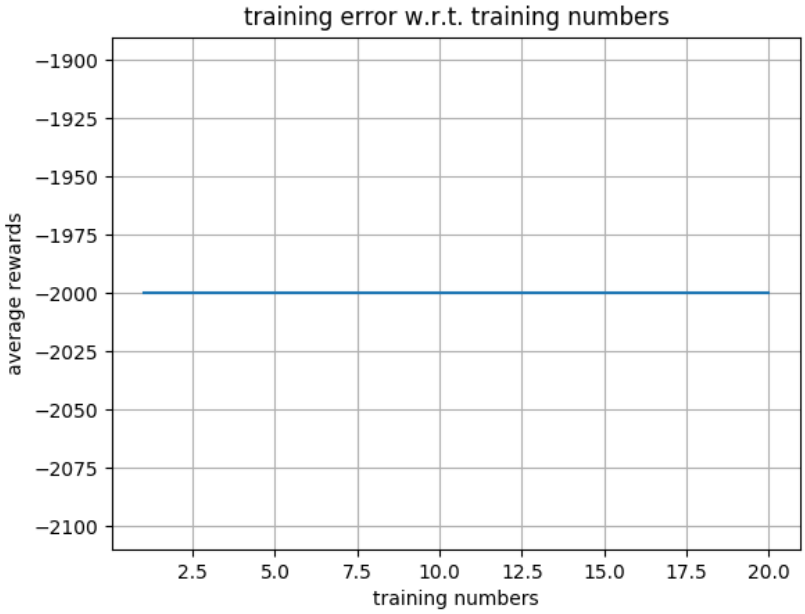

图 24: improved DQN 中，CartPole-v0 测试间 rewar 随着 episode 变化曲线

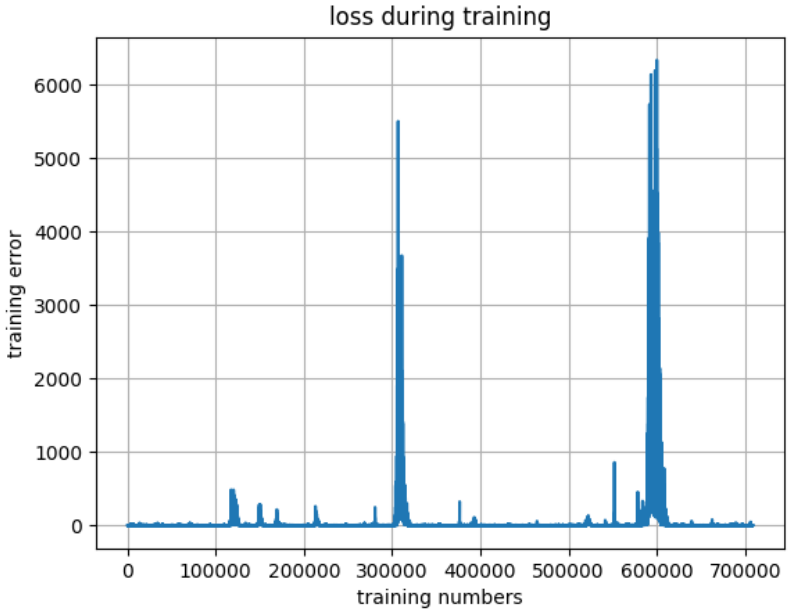

图 25: improved DQN 中，Acrobotv1 训练期间 loss 随着 episode 变化曲线

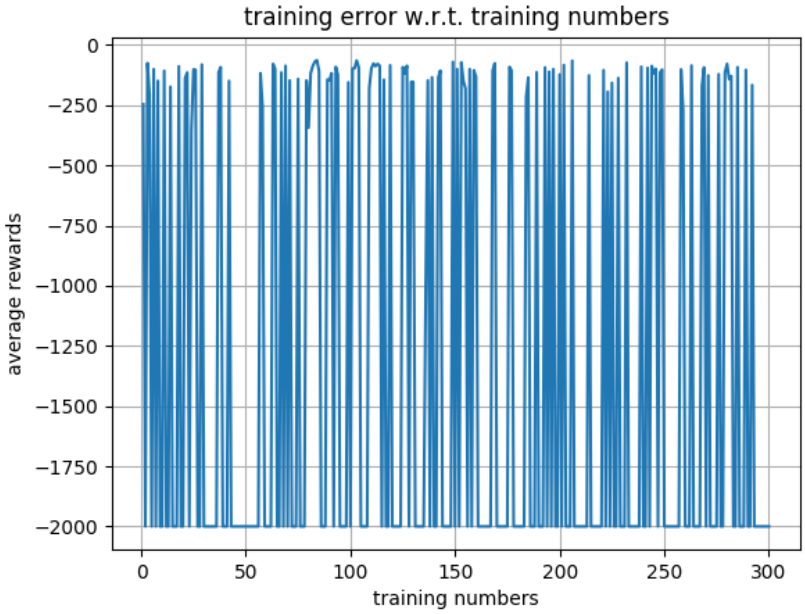

图 26: improved DQN 中，Acrobotv1 训练期间 reward 随着 episode 变化曲线

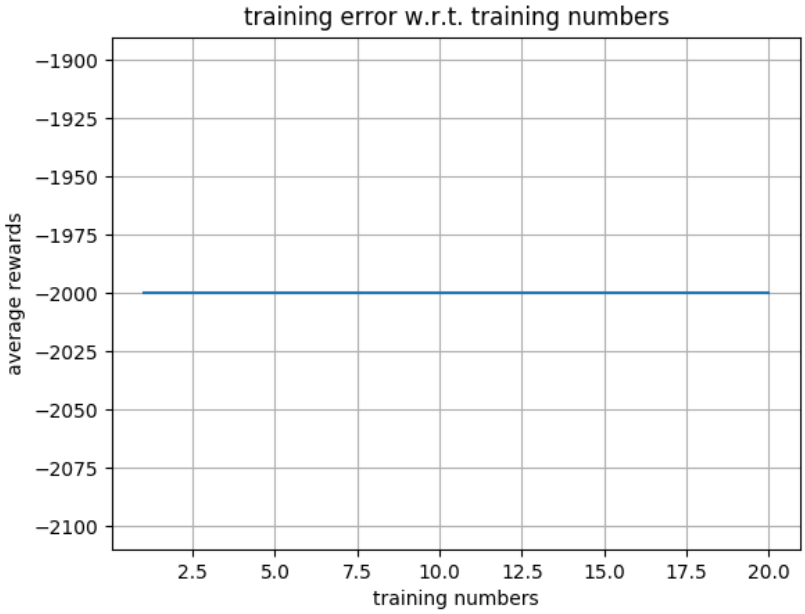

图 27: improved DQN 中，Acrobotv1 测试间 rewar 随着 episode 变化曲线