实验 3. 强化学习实践

MF1733034，李青坪，lqp19940918@163.com

2018 年 1 月 3 日

综述

与传统的监督、无监督学习不同，强化学习的过程需要与学习环境进行交互，利用环境反馈信息进行学习。因此在本次实验中，我们将脱离从训练数据集中获取学习模型的学习框架，转向与动态的学习环境打交道。本次实验，我们将从学习环境的安装、常用强化学习算法的实现，强化学习算法改进这些方面完整的体会一次强化学习研究的过程。

本次实验采用的强化学习环境为 OpenAI 开源的 Gym。Gym 中集成了很多强化学习的实验环境，并为测试强化学习提供了很多学习任务，如 CartPole 和 MountainCar 等。在 DQN 算法实现中会用到深度学习工具，本次实验采用 PyTorch，PyTorch 提供了神经网络的搭建模型，方便我们实现 DQN 算法。

实验二.

Gym 为我们提供了一些基础学习任务，但是在这些任务里面，状态空间都是连续的，而 Q-learning 算法的状态空间是离散的。所以本次实验的首要任务就是对状态空间离散化，通过 Gym 文档中提供的状态-动作空间的查询命令，我们知道 CartPole、MountainCar 和 Acrobot 三个任务的状态-动作空间分别为 (4, 2), (2, 3), (6, 3)，我们需要对这些空间进行离散化。本次实验采用的方法是将连续的空间分成等间隔的 n 份（n 为每个维度上划分的块数）。具体算法如下：

设 observation[i] 表示第 i 维状态观测值，$STATE\_THRESHOLD[i]$ 表示第 i 维状态的下界，$bound\_width$ 表示第 i 维状态的观测值的数据宽度，$STATE\_NUM[i]$ 表示自定义的第 i 维状态的离散状态数目。则对于每一个观测值，求出其在离散化的状态空间中对应的下标即可。

$$state\_index =$$
$$(observation[i] - STATE\_THRESHOLD[i])/bound\_width * STATE\_NUM[i]$$

通过状态空间离散化，即将连续的观测空间状态转换到离散的空间上来，我们即可使用 Q-learning 算法进行强化学习 [1]。

Q-learning 算法实现的思路如下：

- 首先初始化一张 Q(s, a) 表，里面的每一行表示在状态 s，执行动作 a 所拥有的价值。

- 对每一个片段执行以下操作：
- 初始化状态 s
- 对每一个片段里的每一个步骤执行以下操作：
- 在状态 s 使用 ϵ 贪心策略选出动作 a
- 执行动作 a，得到奖励 r 和下一个状态 $s^{'}$
- 更新 Q 值：$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a^{'}} Q(s^{'}, a^{'}) - Q(s,a)]$
- $s \leftarrow s^{'}$

算法实现思路比较简单，而且代码也不是特别复杂，想要实验可以成功的完成任务，最重要的还是参数的调整。在本次实验中，探索率 ϵ 初始设置为 1.0，最小值设置为 0.01，按照指数级别递减；学习率 α 初始设置为 0.5，最小值设置为 0.1，按照指数级别递减；衰减值 γ 设置为 0.99 恒定不变。

针对 CartPole 任务，最大观测长度设置为 20000，片段数设置为 1000（考虑到接近收敛的时候，基本都能保证杆子不倒下来，故设置在连续 200 次成功解决 CartPole 问题后就认为已经收敛，则终止片段循环），成功解决 CartPole 问题的步骤数设置为 199。每个状态维度的可操作状态数设置为 (1, 1, 6, 5)，表示只考虑杆子的角度和角速度，忽略小车的位置和速度。虽然损失两个状态，但对解决 CartPole 问题不会产生影响。

针对 MountainCar 任务，最大观测长度设置为 2000，片段数设置为 1000（考虑到接近收敛的时候，基本都能保证小车到达旗杆处，故设置在连续 200 次成功解决 MountainCar 问题后就认为已经收敛，则终止片段循环）。每个状态维度的可操作状态数设置为 (8, 8)，表示小车的位置和速度各有 8 个可操作状态。

针对 Acrobot 任务，最大观测长度设置为 2000，片段数设置为 1000（考虑到接近收敛的时候，基本都能保证关节达到系统设定的高点，故设置在连续 200 次成功解决 Acrobot 问题后就认为已经收敛，则终止片段循环）。每个状态维度的可操作状态数设置为 (1, 1, 1, 1, 10, 10)，表示只考虑两个杆子的角速度，忽略它们的角度。虽然损失两个状态，但对解决 Acrobot 问题不会产生影响。

对三个强化学习任务分别进行 5 次实验，得到多条轨迹上 reward 的均值和标准差，见表 1.

实验三.

由于 Q-learning 需要穷举状态-动作对，所以面对连续状态-动作空间，Q-learning 无法适用。但是值函数近似 Q-learning 方法能够解决此类问题。若将值函数近似模型选取为一个深度网络，即为 Deep Q-learning (DQN) 算法。DQN 算法基于 Q-learning，算法框架和 Q-learning 完全相同，不同之处在于 Q 函数的表示和更新方式上。DQN 中 Q 函数用一个深度网络表示，记为 Q(s, a; θ)，其中 s 表示状态，a 表示执行的动作，θ 表示网络参数。DQN 的目的是得到一个 Q 函数网络，对于一组状态动作对 (s, a)，使得 Q 函数网络的输

表 1: 均值标准差

-	1	2	3	4	5
均值	7111.80	7376.67	7636.14	7219.83	7426.34
标准差	6009.24	5946.63	6190.02	6025.62	6020.15

-	1	2	3	4	5
均值	-163.78	-178.46	-185.27	-161.381	-212.69
标准差	22.57	129.53	7.98	276.15	32.82

-	1	2	3	4	5
均值	-144.01	-103.94	-105.65	-88.99	-261.28
标准差	29.49	193.17	122.38	63.22	271.97

出 Q(s, a; θ) 等于 (s, a) 下真实的 Q 值。具体到深度网络表示的 Q 函数模型上，Q 函数的学习问题转化为了如何学习一组深度网络参数 θ 使网络输出近似于真实的 Q 值。

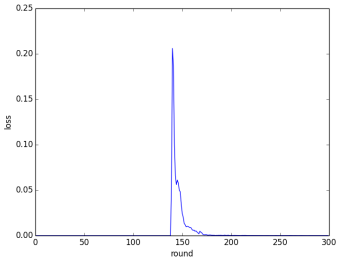
本次实验中，我们采用 PyTorch 提供的神经网络模型建立我们所需的 Q-evaluation 网络,输入层神经元数目为可观测的状态数目;隐藏层有一层,这里设置隐藏层有 $hidden\_num$ 个神经元（$hidden\_num$ 由具体的任务指定，如 CartPole 的 $hidden\_num$ 设为 50）；输出层的神经元个数为动作的数目。新建一个 DQN 类，表示我们将要实现的算法，私有属性 memory 表示大小为 N 的重现内存，这里设置 N=2000，设置损失函数为 PyTorch 提供的 nn.MSELoss() 函数，即平方损失函数，公式如下:

$$loss(x, y) = \frac{1}{N}\sum_{i=1}^{N}|x - y|^2$$

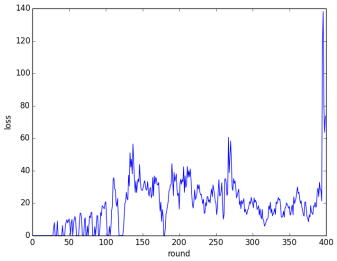
使用 torch.optim.Adam() 作为学习的优化器，它利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率，参数为 Q-evaluation 网络，学习率为 α，不同的任务学习率不同;探索率 ϵ 初始设置为 1，最小值设置为 0.01，随片段 episode 的增加呈指数级别减小；算法中每次从 memory 中抽取的 MINI_BATCH 数目为 32。

DQN 算法实现思路如下:

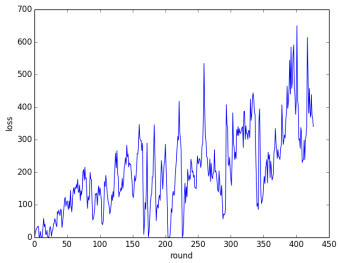
- 初始化大小为 (N x (状态数 +2)) 的 memory 矩阵。
- 新建 Q-target 网络和 Q-evaluation 网络，即完成状态动作函数的初始化。
- 对每一个片段执行以下操作:
- 初始化状态 s
- 对每一个片段里的每一个步骤执行以下操作:
- 在状态 s 使用 ϵ 贪心策略选出动作 a_t
- 执行动作 a_t，得到奖励 r_t 和下一个状态 s_{t+1}
- 将 (s_t, a_t, r_t, s_{t+1}) 存入 memory 中



CartPole



MountainCar



Acrobot

图 1: loss 比较

- 从 memory 中随机抽取 MINI_BATCH 个数据
- 对抽取到的每个数据计算 $y_i = r_j + \gamma max_{a'} Q(s_{t+1}, a'; \theta)$
- 将 y_j 和 $Q(s_j, a; \theta)$ 带入损失函数，修正 θ

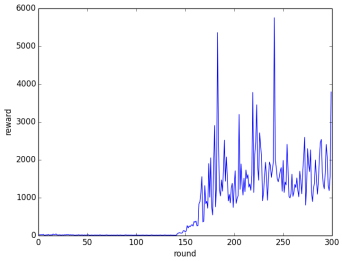
针对 CartPole 任务，最大观测长度设置为 20000，片段数设置为 300。输入层神经元个数为 4，表示 4 个可观测状态；隐藏层神经元个数为 50；输出层神经元个数为 2。同时，需要修改 reward 的值，使状态动作函数尽快收敛，修改思路如下：考虑小车的位置离中点越近 reward 值越高、杆子越保持垂直 reward 值越高。学习率 α 的初始值设为 0.008；衰减值 γ 设置为 0.9 恒定不变；

针对 MountainCar 任务，最大观测长度设置为 2000，片段数设置为 1000（考虑到接近收敛的时候，基本都能保证小车到达旗杆处，故设置在连续 100 次成功解决 MountainCar 问题后就认为已经收敛，则终止片段循环）。输入层神经元个数为 2，表示 2 个可观测状态；隐藏层神经元个数为 30；输出层神经元个数为 3。修改 reward 值：小车离坡道最低点越远 reward 值越高、一旦小车达到最高点则给一个较大的 reward。学习率 α 的初始值设为 0.005；衰减值 γ 设置为 0.85 恒定不变；

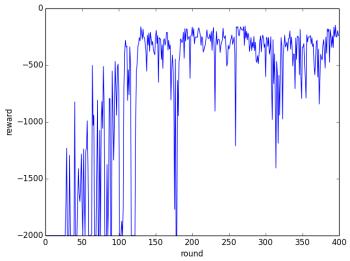
针对 Acrobot 任务，最大观测长度设置为 2000，片段数设置为 300。输入层神经元个数为 6，表示 6 个可观测状态；隐藏层神经元个数为 50；输出层神经元个数为 2。修改 reward 值：只要机械臂达到给定的高度，系统将返回 reward=0，此时我们将 reward 修改为一个较大的值。学习率 α 的初始值设为 0.005；衰减值 γ 设置为 0.75 恒定不变；

网络训练误差随训练轮数的变化关系见图 1.

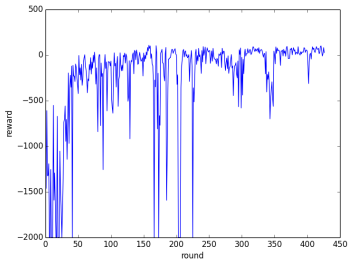
每轮训练 reward 之和随训练轮数的变化关系随训练轮数的变化关系见图 2



CartPole



MountainCar



Acrobot

图 2: reward 比较

对三个强化学习任务分别进行 5 次实验，得到多条轨迹上 reward 的均值和标准差，见表 2.

表 2: 均值标准差

CartPole	1	2	3	4	5
均值	1079.10	20000.0	1505.11	2832.19	1210.64
标准差	95.29	0	144.00	302.37	73.14

MountainCar	1	2	3	4	5
均值	-235.35	-227.41	-244.09	-213.23	-236.48
标准差	73.47	99.30	92.47	64.48	75.38

Acrobot	1	2	3	4	5
均值	-200.87	-152.51	-183.53	-311.80	-346.57
标准差	107.78	40.08	77.10	232.30	159.13

从图中看出，DQN 算法在学习的时候稳定性较差，波动幅度很大；而且从表格中可以看出，在 CartPole 任务中，对于某次训练效果很好的 Q 网络，CartPole 任务可以坚持很久；而对训练效果不好的 Q 网络，CartPole 任务很快便结束。而事实上，DQN 算法的效果甚至还不如 Q-learning 算法的效果（参数可能影响了实验结果）。

实验四.

在 DQN 中，每在任务环境中探索一步均要更新一次网络权重，研究者指出这样的更新方式是不稳定的，对学习 Q 值网络不利。故需要对原来的 DQN 算法进行改进。引入 Q-target 网络，因为在 DQN 算法中目标 Q 网络是随着 Q 网络的更新而变化的，这样会造成目标 Q 值和当前的 Q 值得相关性较大。本次实验中，引入 Q-target 网络，每隔一段时间更新该 Q-target 网络，提升强化学习的稳定性。

本次实验中，我们采用 PyTorch 提供的神经网络模型建立我们所需的 Q-target 网络和 Q-evaluation 网络；参数为 Q-evaluation 网络，学习率 α 的初始值设为 0.008；探索率 ϵ 初始设置为 1，最小值设置为 0.01，随片段 episode 的增加呈指数级别减小；衰减值 γ 设置为 0.99 恒定不变；Q-target 更新频率设置为每学习 100 次更新一次；算法中每次从 memory 中抽取的 MINI_BATCH 数目为 32。

改进的 DQN 算法实现思路与 DQN 算法的实现思路类似，只是加入了一个 Q-target 网络，在修正网络参数 θ 的时候，改为从 Q-target 网络中获取下一个 reward 最大的动作，并设计每隔一段时间更新 Q-target 网络。

针对 CartPole 任务，最大观测长度设置为 20000，片段数设置为 200（实验发现，在执行 200 次片段过后就接近收敛）。输入层神经元个数为 4，表示 4 个可观测状态；隐藏层神经元个数为 50；输出层神经元个数为 2。同时，需要修改 reward 的值，使状态动作函数尽快收敛，修改思路如下：考虑小车的位置离中点越近 reward 值越高、杆子越保持垂直 reward 值越高。

针对 MountainCar 任务，最大观测长度设置为 2000，片段数设置为 400（考虑到接近收敛的时候，基本都能保证小车到达旗杆处，故设置在连续 300 次成功解决 MountainCar 问题后就认为已经收敛，则终止片段循环）。输入层神经元个数为 2，表示 2 个可观测状态；隐藏层神经元个数为 25；输出层神经元个数为 3。修改 reward 值：小车离坡道最低点越远 reward 值越高、一旦小车达到最高点则给一个较大的 reward。

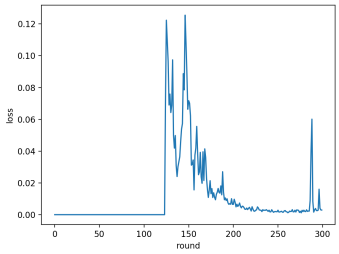
针对 Acrobot 任务，最大观测长度设置为 2000，片段数设置为 300。输入层神经元个数为 6，表示 6 个可观测状态；隐藏层神经元个数为 75；输出层神经元个数为 2。该任务没有修改 reward 值，利用系统给定的 reward 值就能很好的完成强化学习。

网络训练误差随训练轮数的变化关系如图 3 所示

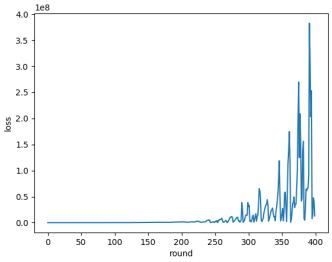
每轮训练 reward 之和随训练轮数的变化关系随训练轮数的变化关系如图 4 所示

对三个强化学习任务分别进行 5 次实验，得到多条轨迹上 reward 的均值和标准差，见表 3.

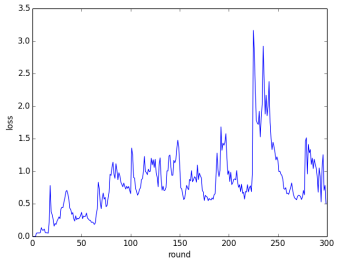
通过实验数据我们可以看出，改进后的 DQN 算法在稳定性和性能上都比原始的 DQN 算法要好，特别是在 CartPole 任务中，改进后的 DQN 基本能保证一条轨迹执行 20000 步后结束，而且其他两个任务完成所花费的步数也比原始的 DQN 算法训练出来的结果要好。改进后的 DQN 之所以能增加 Q 值训练的稳定性，主要是因为改进后的 DQN 算法增加了 Q-target 网络，该网络只在一段时间间隔后更新，而不用像原始的 Q 网络一样需要每次都更新，这样在代入损失函数修正网络参数 θ 的时候，降低了 Q 值与当前数据的相关性。所以改进后的 DQN 相比于原始的 DQN 具有更好的稳定性。



CartPole

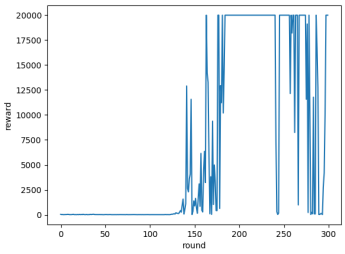


MountainCar

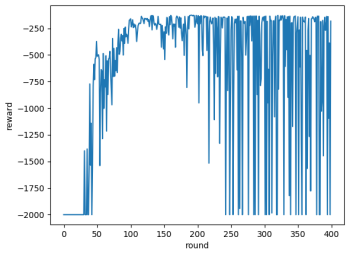


Acrobot

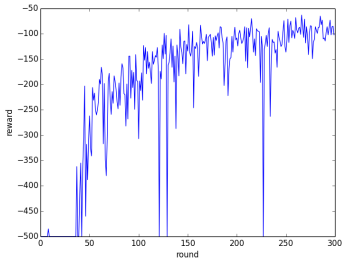
图 3: loss 比较



CartPole



MountainCar



Acrobot

图 4: reward 比较

表 3: 均值标准差

CartPole	1	2	3	4	5
均值	19667.55	19575.45	19489.12	19456.23	19535.27
标准差	1953.69	2357.74	2471.89	2001.16	1959.57

MountainCar	1	2	3	4	5
均值	-135.28	-134.68	-147.42	-133.76	-145.34
标准差	34.46	36.66	26.81	33.58	65.38

Acrobot	1	2	3	4	5
均值	-112.25	-103.86	-103.35	-108.71	-151.89
标准差	42.20	33.14	42.01	49.26	129.11

参考文献

[1] 周志华. 机器学习. 清华大学出版社, 2016.