

RL 小作业

211098325 毛丁

2023 年 10 月 28 日

题目 1. 实现倒立摆 (CartPole) 小游戏, 游戏里面有一个小车, 上有竖着一根杆子, 每次重置后的初始状态会有所不同。小车需要左右移动来保持杆子竖直, 为了保证游戏继续进行需要满足以下两个条件:

1. 杆子倾斜的角度 不能大于 15°
2. 小车移动的位置 x 需保持在一定范围 (中间到两边各 2.4 个单位长度)

解答. 本次作业使用 openAI gym 包提供的倒立摆仿真环境, state, action, value, reward 设计过程如下:

- state 状态:

state 采用一个含有 4 个元素的数组表示, 分别为 x : 小车在轨道上的位置, θ : 杆子与竖直方向的夹角, \dot{x} : 小车速度 $\dot{\theta}$: 角度变化率

- action 动作空间:

在 gym 包的设计中, 动作空间为 $\{0,1\}$ 分别表示对小车左移或右移。

- value 价值:

本次作业我对倒立摆模型的学习采用的是 DQN 算法, 将小车的某一状态 S (一个 1×4 的张量) 作为输入放入 `target_net` (下面在解释 DQN

算法时会解释) 中可以得到 output (一个 1×2 的张量), 比如输出为 (1,2), 则采取动作 0 和动作 1 得到的 value 分别为 1,2 此时下一步应该采取动作 1.

- reward 奖励:

如果当前状态 S 下采用动作 a , 游戏能够继续, 则 reward 为 1, 否则为 0。但是实际中的训练结果小车更加专注于让杆子直立而忽略了小车左右运动范围的限制。这可能是由于即使小车即将达到运动范围边界, reward 值仍然不变引起的, 或许可以改进为越接近边界 reward 越小。

采用的强化学习算法为基于 Q-learning 的 DQN 算法。DQN 算法采用了 2 个神经网络, 分别是 evaluate network (Q 值网络) 和 target network (目标网络), 两个网络结构完全相同。其核心是值函数迭代过程, 最终目的为: 使 $Q(s,t)$ 预测出的行动 a_{pred} 尽量接近于实际应该采取的行动 a_{fact}

学习开始之前, evaluate network 被初始化为一个拥有任意权重的网络 (在本次作业中为 linear+relu+linear 层架构的网络)。一个 episode 即一次游戏的训练过程如下: 在每个时间 t , 小车出入状态 S_t , Agent 选择一个动作 a_t (这里动作的选择策略下面会解释), 得到一个奖励 R , 进入一个新的状态 S_{t+1} 然后将四元组 (S_t, a_t, R, S_{t+1}) 存入记忆池 memory pool, 之所以设立 memory pool, 是因为我们实际上不知道采取一个行动 a 后将会得到的下一个状态和获得的奖励是多少, 因此通过 Experience replay 即经验回放的策略, 我们可以通过过去的决策得到的经验来”模拟”我们采取某个动作可能得到的结果。

当经验池中的条目数大于一定数值 (在作业中设置为 1000) 时, 认为已经积累了足够的经验了, 因此可以从 memory pool 中随机取出 batch 大小的条目进行训练。以 batch 为 1 为例: 将 S_t 和输入到 evaluate network 中, 从结果中选取执行动作 a_t 得到的 Q 值 q_{eval} , 然后将 S_{t+1} 输入到 target

network 中, 从结果中选取最大的 Q 值 $q_next.max$, 设置 $q_target = R + \gamma * q_next.max$ 然后使用 $MSE_loss(q_eval, q_target)$, 然后进行反向传播和权重更新, 这里更新的是 evaluate work 的权重, 训练了一定的步数后将 evaluate work 中的权重赋值给 target work 的权重。

动作选择策略: 采用一个递减的初始值为 1 的参数 ϵ , 从 0 和 1 中随机选择一个值 p 若 p 小于 ϵ 则随机选择一个 action, 否则使用 target net 预测的 action。

Algorithm 1: DQN 算法

Data: α 是学习率, γ 为折扣因子 S 为初始环境

```

1 for each episode do
2   Initialize  $S$ ;
3   for each step of episode do
4     Choose  $A$  from  $S$  using policy derived from  $Q$ 
      (e.g.,  $\epsilon$ -greedy);
5     Take action  $A$ , observe  $R, S'$ ;
6     push  $(S, A, R, S')$  into memory pool;
7     if step satisfy some condition then
8       Learn;
9       if step satisfy some condition then
10        target network's weights  $\leftarrow$  evaluate network's
          weights
11      end
12    end
13  end
14  until  $S$  is terminal
15 end

```

tensorboard 可视化训练过程:

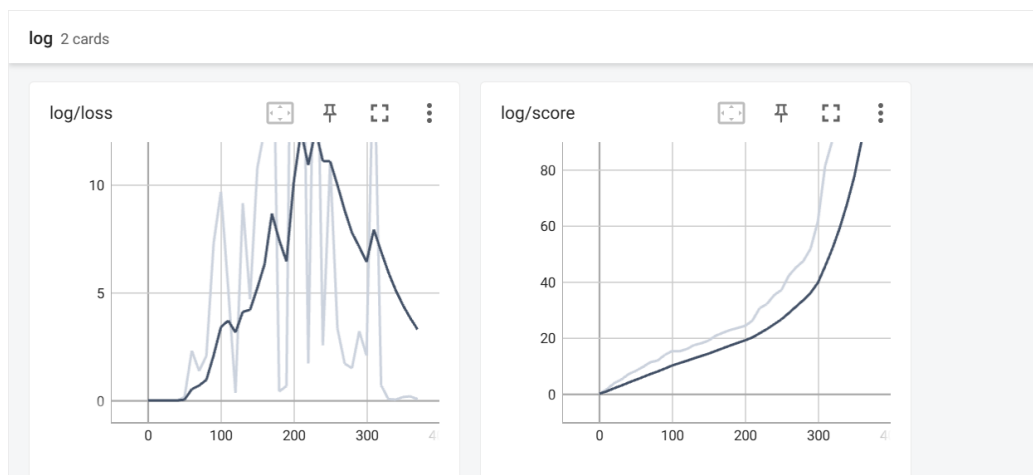


图 1: 训练过程

代码和保存的游戏运行 gif 见文件夹其余文件