

大作业三 - 强化学习

李姜帆 自64 2016011819

大作业三 - 强化学习

游戏/环境

玩法

环境详细设定

钓竿/捕捉条

鱼

reward

算法

基于表格

Q-learning

SARSA

深度强化学习

加大难度

人类经验

小结

参考资料

程序文件

附件

游戏/环境

选择了 **星露谷物语** 的 **钓鱼游戏** 作为题目。写了一个环境 `fish.py` 模拟该游戏，因为源码也不知道是怎么样的，所以就是架构和一些游玩的经验推测的。

玩法

使用 `play.py` 即可体验该游戏。蓝色的条为水深，红色的小鱼会在其中随机移动，可以移动的绿色条由玩家控制，按空格键使其上升，松开则会下降。玩家需要使鱼保持在捕捉条中一定时间来获取胜利，鱼离开捕捉条会使捕捉进度下降，右侧的浮标表示捕捉进度。捕捉进度耗光则游戏失败，捕捉进度满则游戏成功。

环境详细设定

钓竿/捕捉条

捕捉条大致按照物理模型设置，另外设有速度上限。

在上下终止处模拟实际物理世界，即高速落下时会反弹，上浮到水面时速度降为零。

鱼

鱼的加速度随机生成，另有速度上限控制。

原游戏wiki中给出了五种鱼的运动方式 ¹

- **混合型**：这些鱼是基本的运动方式。
- **平滑型**：这些鱼的运动比较固定。
- **下坠型**：这些鱼会迅速加速下滑。
- **漂浮型**：这些鱼会迅速加速上浮；
- **猛冲型**：这些鱼可能会向上向下多移动 $[2 \times \text{难度系数}] \%$ ，钓鱼条会上下不规则移动。

可以用不同的加速度分布来模拟。

reward

reward就以是否在捕鱼条内进行奖惩。

算法

基于表格

```
class Table
```

作为离散的算法，针对本游戏连续的环境，可以离散化状态值。 `Table.ob2state(ob)` 实现将状态离散化，并存储到状态表中。

Q-learning和SARSA的算法区别主要在学习更新一步，故以两个 `class Table` 的子类来实现。

状态空间过大所以训练的效果并不好，2000ep也没有成功过一次，所以还是看下一节吧。

Q-learning

```
Initialize  $Q(s, a)$  arbitrarily
Repeat (for each episode):
  Initialize  $s$ 
  Repeat (for each step of episode):
    Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
    Take action  $a$ , observe  $r, s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'$ ;
  until  $s$  is terminal
```

SARSA

```
Initialize  $Q(s, a)$  arbitrarily
Repeat (for each episode):
  Initialize  $s$ 
  Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
  Repeat (for each step of episode):
    Take action  $a$ , observe  $r, s'$ 
    Choose  $a'$  from  $s'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
     $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'; a \leftarrow a'$ ;
  until  $s$  is terminal
```

深度强化学习

将状态作为神经网络的输入，输出此状态下各个动作的Q值，就可以利用神经网络来存储Q表格，避免了大量的空间去存储Q表。

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$

For episode = 1, M **do**

 Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

For $t = 1, T$ **do**

 With probability ε select a random action a_t

 otherwise select $a_t = \operatorname{argmax}_a Q(\phi(s_t), a; \theta)$

 Execute action a_t in emulator and observe reward r_t and image x_{t+1}

 Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

 Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

 Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

 Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$

 Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ

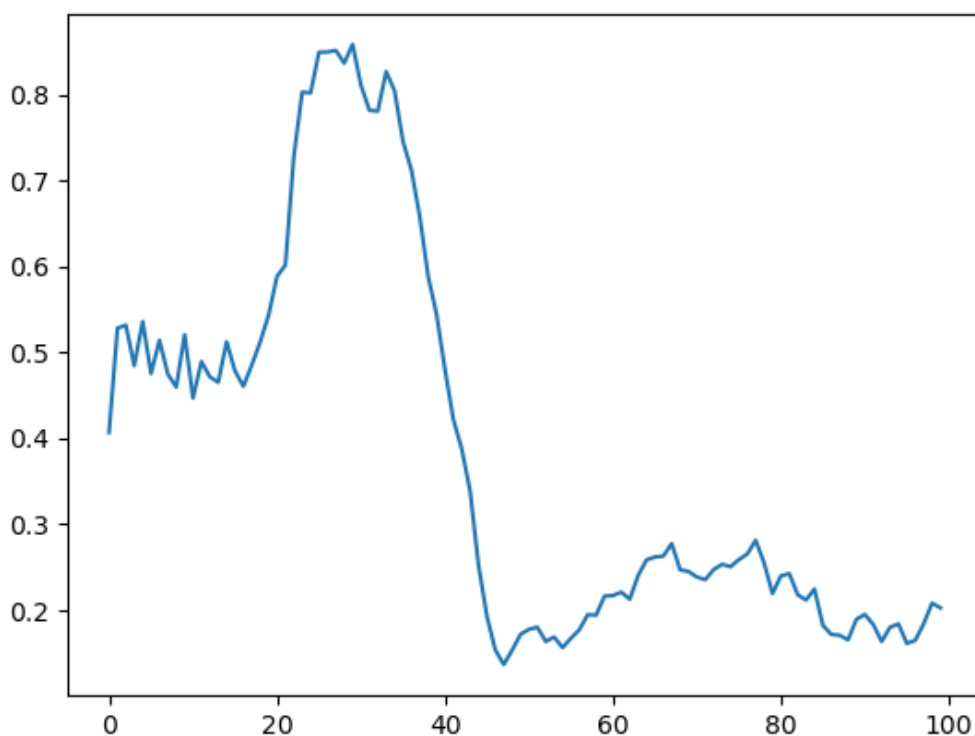
 Every C steps reset $\hat{Q} = Q$

End For

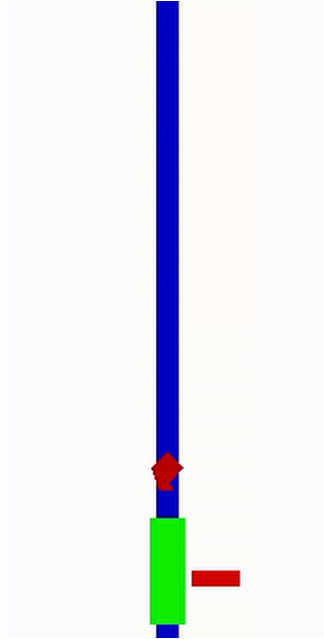
End For

使用了一层8个单元的隐藏层。主要是考虑以人类来操作的话，大致就是根据位置差和速度差进行判断，不会是很复杂的手段。

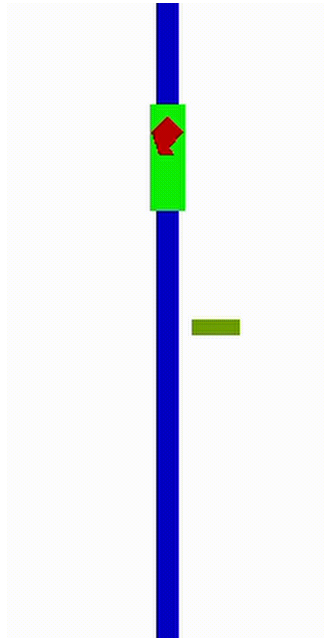
100ep训练loss变化：



ep1表现：（GIF在html或Markdown/Typora版本报告中查看）



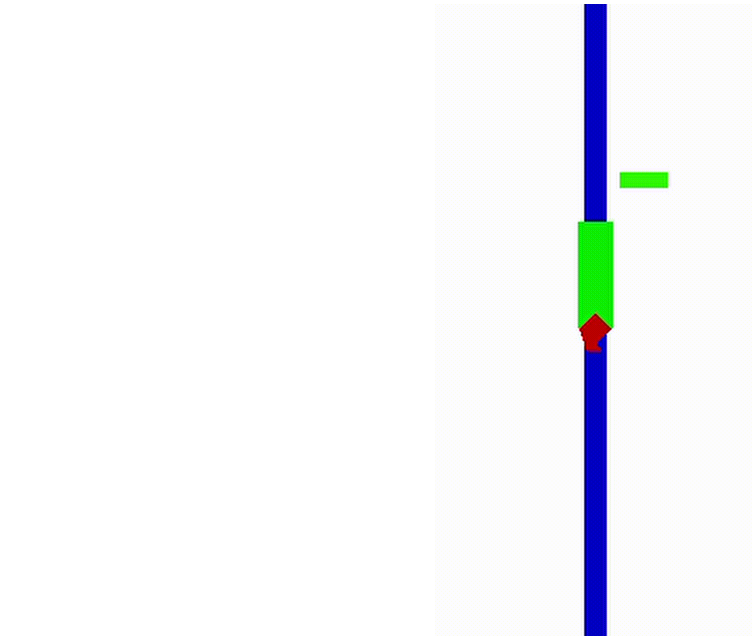
ep100表现：



可以看到它不仅学习到了紧跟鱼移动，保持钓鱼条的位置，而且在下落时有一定的速度控制，减少触底反弹。

加大难度

加大鱼随机运动的最大加速度，使鱼更难被捕捉。使用原先训练的模型，效果如下（鱼最大加速度*2）：



人类经验

主要是在测试小车爬山的时候，小车自己很难爬上去，就没有成功的经验。虽说改reward可以让效果比较好，但是有一点人为地降低了难度——“小车学会爬上山顶”变成了“小车学会加速”。所以可以提供一些人类操作的记录，避免很难达到终点而无法学习到知识。

不过对于本游戏来说，并不是很需要，由于成功的reward很容易获得。

小结

通过本次大作业，对强化学习有了更深刻的了解，同时在作业的驱使下了解了一些当前强化学习各种类型手段。通过神经网络来存储Q表真是精湛的想法。

之后可以考虑加一些图像处理，就能在星露谷钓鱼装外挂啦！（妈妈再也不用担心我钓不到鱼啦

参考资料

强化学习 Reinforcement Learning 教程系列 | 莫烦Python

程序文件

- `rl.py` 强化学习主程序
- `fish.py` 钓鱼游戏环境（基于gym）

附件

- `play.py` ♥点击♥就玩🎣钓鱼🎣小游戏
- `memo.npy` 8000次人类经验
 - 存储格式 `[s, [action, reward], s']`，与网络定义一致
 - 在 `play.py` 中最后一行将 `save` 设为 `True` 即可进入保存操作模式
- `eval.pt`，`tar.pt` 500ep训练之后的参数

1. <https://zh.stardewvalleywiki.com/%E9%B1%BC> ↩