程序报告

学号：　　姓名：

一、问题重述

（简单描述对问题的理解，从问题中抓住主干，必填）

==========

本问题目的是让小鼠找到迷宫的终点，主要分成两个部分：其一是实现深度优先搜索，其二是强化学习。在本题中，使用的方法是 DQN 算法，即 Q-Learning 与深度学习的结合，为了表示出无限的状态和动作引入深度学习模型来近似。

二、设计思想

（所采用的方法，有无对方法加以改进，该方法有哪些优化方向（参数调整，框架调整，或者指出方法的局限性和常见问题），伪代码，理论结果验证等… **思考题，非必填**）

==========

根据 DQN 算法来进行强化学习，笔者一开始进行调参，之后发现小鼠依旧局限在一个角落，笔者主要考虑到的原因是 reward 设置不合理、epsilon 探索度不够高以及 Memory 过少，为此笔者扩大 Memory 并引入预先探索模块，即首先进行随机探索来获得一些 Memory，然后在实际学习过程中，如果没有探索到终点，就调整 epsilon 重新进入探索状态。笔者了解到 DQN 算法的优化方向有 DDQN 等，但是由于时间关系，没有办法都尝试，未来希望能够继续尝试。

三、代码内容

（能体现解题思路的主要代码，有多个文件或模块可用多个"===="隔开，必填）

==========

笔者主要使用了 MinDQNRobot 和 QNetwork 的 baseline 代码，但经过对算法的理解，发现 baseline 可能存在一点逻辑问题，根据给出的 DQN 算法，每个内存循环都需要对 Loss 函数进行梯度下降并反向传播，同时在较低频率下对 Target_Model 进行参数更新，但是 baseline 中是每 N 轮同时进行梯度下降和参数更新，使得梯度下降并不即时，并且 Eva_Model 和 Target_Model 参数一直保持相同，这不符合 DQN 算法的思路。

同时，在没有开金手指的情况下，Memory 没有足够的经验，导致每次抽取较大 minibatch 是不行的，于是笔者修改了 batch 的大小，即 Memory 较小时全部抽取，等到 Memory 大于 Batchsize 之后就使用传入的 Batchsize。

于是笔者修改了相关的函数：

```python
def _learn(self, batch: int = 16):
    if len(self.memory) < batch:
        batch = len(self.memory)
    state, action_index, reward, next_state, is_terminal = self.memory.random_sample(batch)

    """ convert the data to tensor type"""
    state = torch.from_numpy(state).float().to(self.device)
    action_index = torch.from_numpy(action_index).long().to(self.device)
    reward = torch.from_numpy(reward).float().to(self.device)
    next_state = torch.from_numpy(next_state).float().to(self.device)
    is_terminal = torch.from_numpy(is_terminal).int().to(self.device)

    self.eval_model.train()
    self.target_model.eval()

    """Get max predicted Q values (for next states) from target model"""
    Q_targets_next = self.target_model(next_state).detach().min(1)[0].unsqueeze(1)

    """Compute Q targets for current states"""
    Q_targets = reward + self.gamma * Q_targets_next * (torch.ones_like(is_terminal) - is_terminal)

    """Get expected Q values from local model"""
    self.optimizer.zero_grad()
    Q_expected = self.eval_model(state).gather(dim=1, index=action_index)

    """Compute loss"""
    loss = F.mse_loss(Q_expected, Q_targets)
    loss_item = loss.item()

    """ Minimize the loss"""
    loss.backward()
    self.optimizer.step()

    return loss_item
```

上面_learn 函数中去掉最后的 target_model 被参数更新的部分，并且 batch 修改为传入 batch 和 len(Memory)中较小的数。

```python
def train_update(self):
    """
    以训练状态选择动作并更新Deep Q network的相关参数
    :return :action, reward 如: "u", -1
    """
    # action, reward = "u", -1.0

    # ————————————————请实现你的算法代码————————————————————————————
    state = self.sense_state()
    action = self._choose_action(state)
    reward = self.maze.move_robot(action)
    next_state = self.sense_state()
    is_terminal = 1 if next_state == self.maze.destination or next_state == state else 0

    self.memory.add(state, self.valid_action.index(action), reward, next_state, is_terminal)

    """--间隔一段时间更新target network权重--"""
    self._learn(batch=32)
    if self.step % self.EveryUpdate == 0:
        """copy the weights of eval_model to the target_model"""
        self.target_replace_op()

    """---update the step and epsilon---"""
    self.step += 1
    self.epsilon = max(0.01, self.epsilon * 0.9)
    # ————————————————————————————————————————————————————————————

    return action, reward
```

上图 train_update 中设置为隔一段时间才更新 Target_Model 的权重。

==========

主要修改一些参数，提交结果时的参数如下：

```
''' QLearning parameters'''
epsilon0 = 0.7  # 初始贪心算法探索概率
gamma = 0.5  # 公式中的 γ

EveryUpdate = 10  # the interval of target model's updating

"""some parameters of neural network"""
target_model = None
eval_model = None
batch_size = 32
learning_rate = 1e-2
TAU = 1e-3
step = 1  # 记录训练的步数
```

==========

另外增加了预探索模块，机器人创建的时候会预先探索，代码如下：

```
    """try to explore the world"""
    self.explore()

def explore(self):
    for i in range(self.maze_size ** 5):
        state = self.sense_state()
        action = random.choice(self.valid_action)
        reward = self.maze.move_robot(action)
        next_state = self.sense_state()
        is_terminal = 1 if next_state == self.maze.destination or next_state == state else 0
        self.memory.add(state, self.valid_action.index(action), reward, next_state, is_terminal)
        if next_state == self.maze.destination:
            break
    self.reset()
    return
```

==========

增加了每轮 train 的 learn 的次数：

```
for i in range(10):
    self._learn(batch=32)
```

==========

调整了探索策略，self.flag 表示机器人是否到过出口，当一直没有去过出口时就提高 epsilon，重新开始衰减：

```
self.epsilon = max(0.01, self.epsilon * 0.995)
if self.flag == 0 and self.epsilon < 0.3:
    self.epsilon = 0.7
```

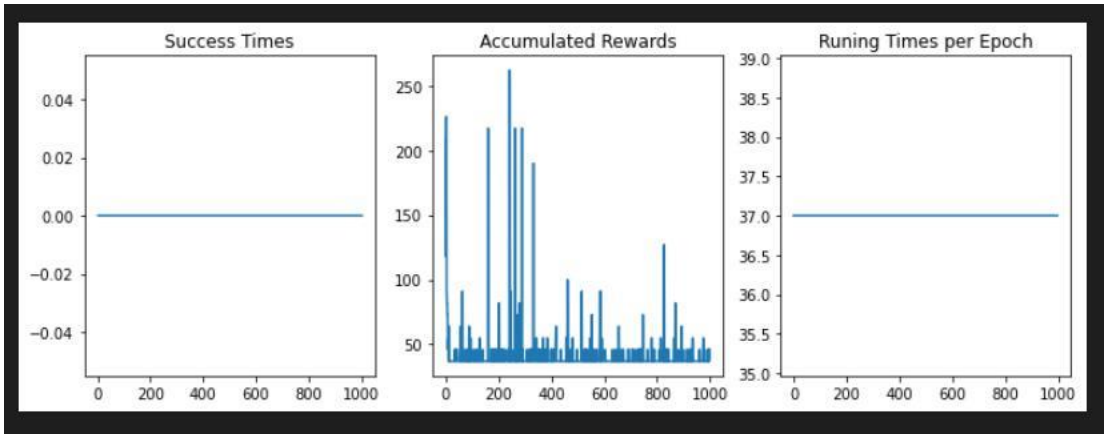
四、实验结果

（实验结果，必填）

==========

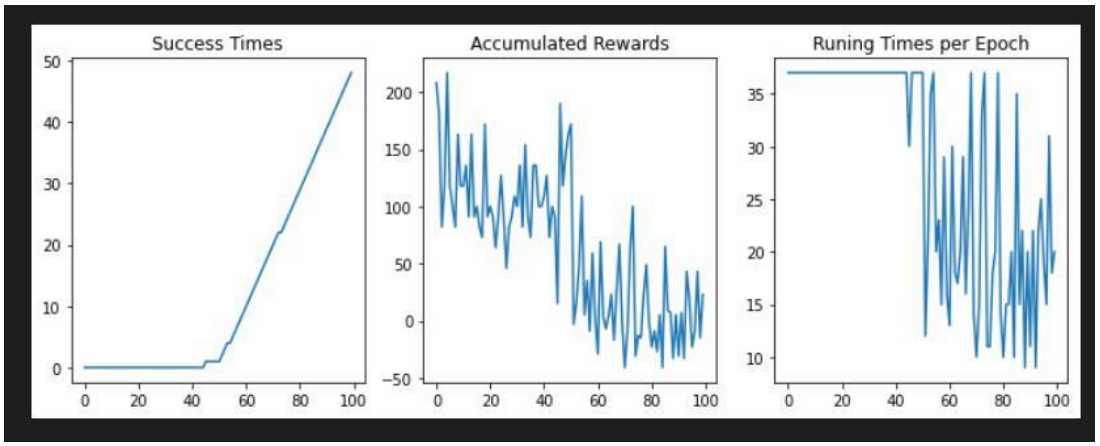
首先声明：**笔者没有使用金手指**。笔者首先直接尝试跑 1000 次查看结果（下图），发现：

1000 次中一次都没有走到终点，并且随着 epsilon 的逐渐衰减，最后探索性不断下降，会一直走原来并没有走到终点的老路，导致最后收敛到一个正数。由此可以发现，目前的探索性非常低。



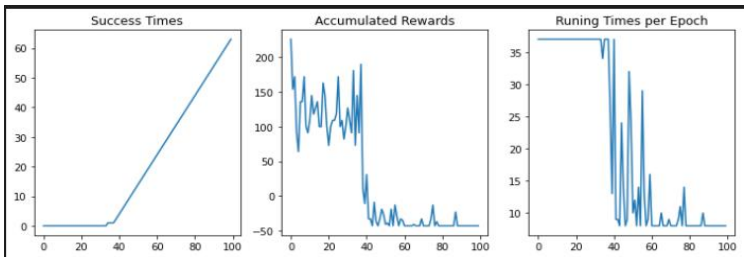
==========

接下来笔者直接去除原来 0.995 的 epsilon 衰减度，让 Agent 保持 0.5 的高探索度，最终发现效果有很大的提升（下图）。Agent 在 40 次左右终于找到了终点，于是在此基础上才能继续走原来正确的路。由此笔者想到：epsilon 的衰减不能太快。



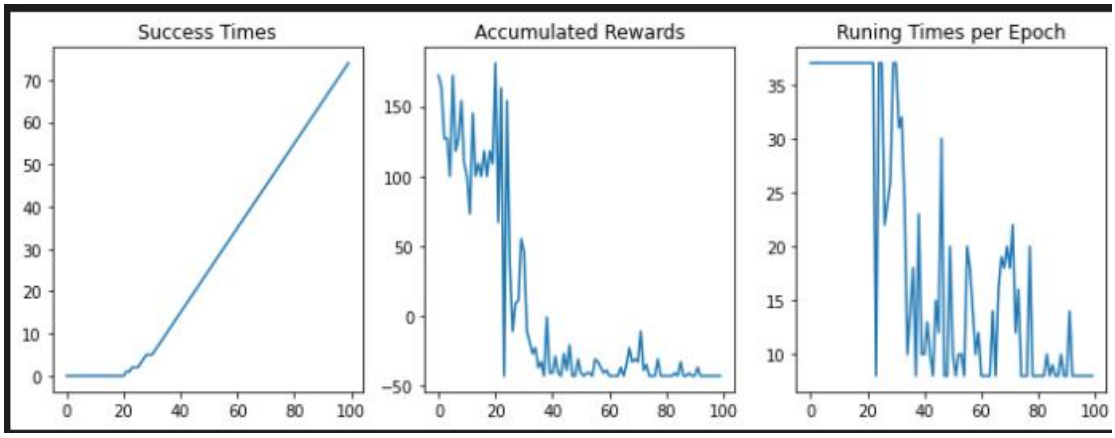
==========

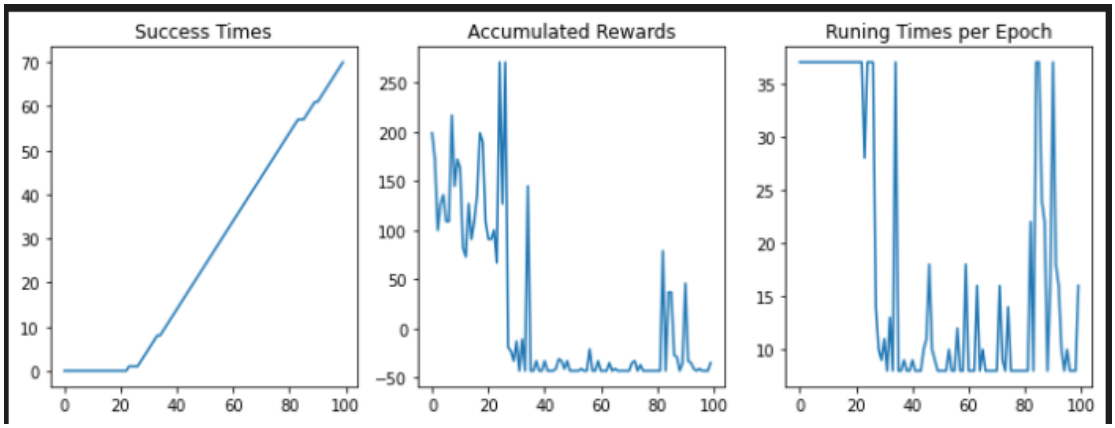
于是笔者修改了代码，设置一个 flag，之后找到最优路径之后才开始以 0.995 速率衰减。最后结果有了更优的效果。

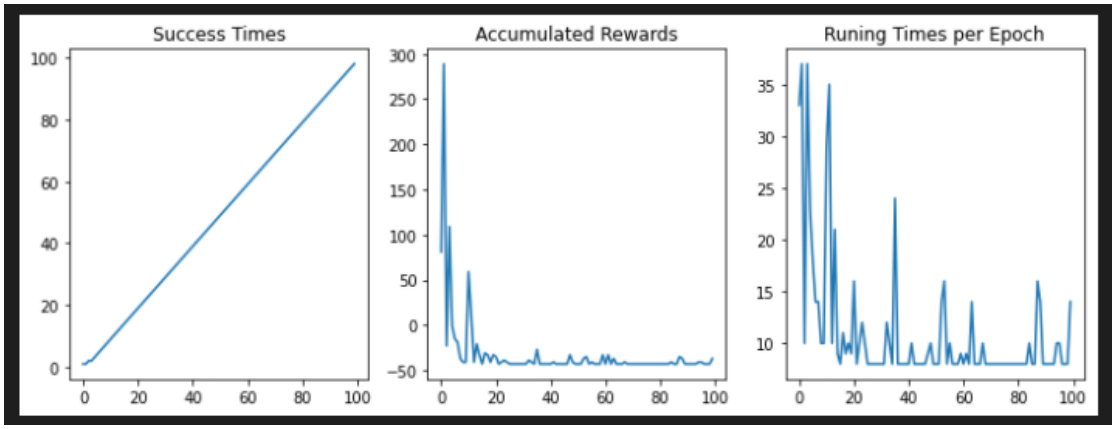


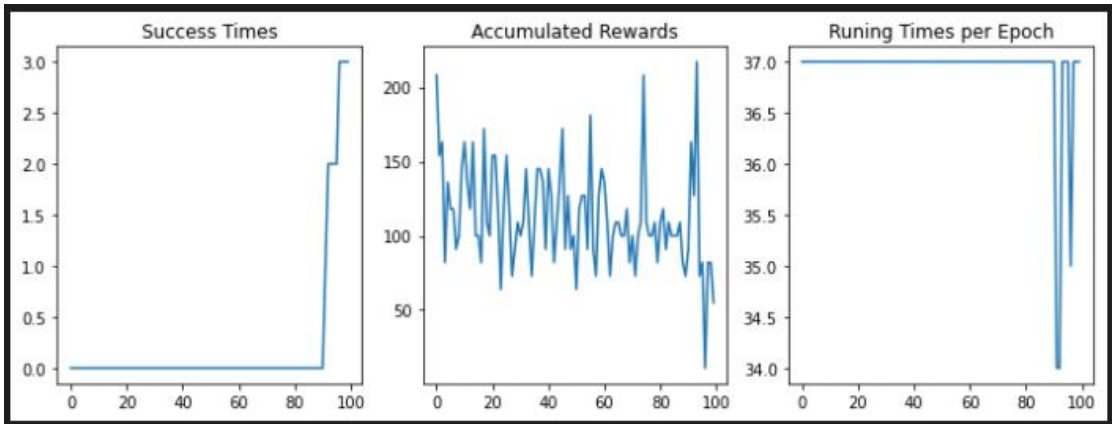
==========

笔者继续考虑探索问题，初始的 epsilon 值为 0.5 似乎较低，笔者需要重新调整 epsilon 和衰减率。首先调整 epsilon 为 0.6/0.7/0.8/0.9，发现前三者效果都比较好，特别是 0.8，所以笔者选择使用 0.8。


Success Times
Accumulated Rewards
Runing Times per Epoch


Success Times
Accumulated Rewards
Runing Times per Epoch


Success Times
Accumulated Rewards
Runing Times per Epoch


Success Times
Accumulated Rewards
Runing Times per Epoch

==========

接下来笔者遇到了巨大的困难，目前的算法情况主要是靠一开始随机走到最后目的地，之后就比较容易走到结果，但是机器人很难靠随机方法走到结果来获得一个正向反馈。笔者在这个阶段尝试过的方法有：

①上面所述设置一个 flag，只有当机器人走到最后结果之后才开始衰减 epsilon，即使用几乎完全的随机来走

②调参：笔者几乎调整过所有的参数，包括 batchsize、EveryUpdate、gamma、epsilon0、learning_rate 以及 reward 的分数。可以从我 66 次的测试结果中看出。

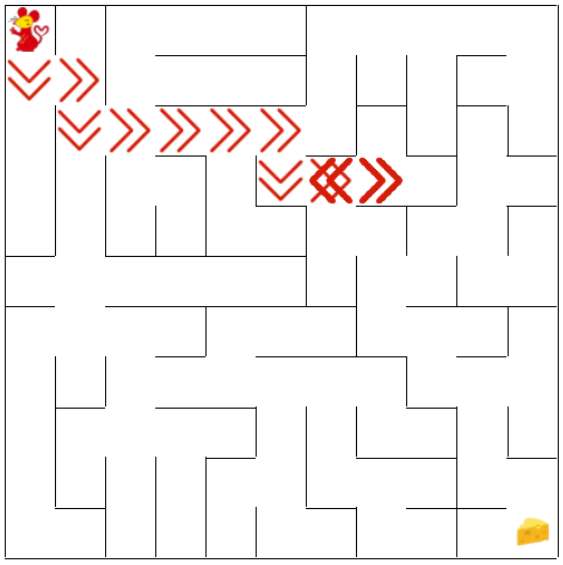
③增加学习次数：笔者怀疑测试程序中训练量太低，于是尝试多次_learn 以提高训练量，但是效果也并没有很好。

==========

笔者主要认为是缺少 Memory 的问题，于是笔者首先增大 Memory 到 1e5，然后让机器人先尝试走获得一些经验，然后再开始 DQN，并且每走一步都多次学习，另外笔者还发现当 gamma 比较大的时候机器人收敛反而会慢一些，于是我调整了 gamma 到 0.5。此时发现小鼠能够通过初级和中级，并且在高级中也一般能不局限于一个位置。

测试详情 隐藏迷宫

强化学习level11 (Defeat)



4 / 4

确定

==========

接下来笔者再考虑 epsilon 的衰减，笔者设计 epsilon 初始值为 0.7，当 epsilon<0.3 并且小鼠还没有去过终点，就重新更新 epsilon=0.7，以此来让小鼠继续探索。但最后还是没有能过高级。

测试详情 展示迷宫

测试点	状态	时长	结果
测试基础搜索算法	✓	0s	恭喜, 完成了迷宫
测试强化学习算法(初级)	✓	2s	恭喜, 完成了迷宫
测试强化学习算法(中级)	✓	24s	恭喜, 完成了迷宫
测试强化学习算法(高级)	✓	650s	很遗憾, 未能走完迷宫

确定

五、总结

（自评分析（是否达到目标预期，可能改进的方向，实现过程中遇到的困难，从哪些方面可以提升性能，模型的超参数和框架搜索是否合理等），**思考题，非必填**）

==========

本次实验离自己的预期还是有点距离，实验中还是遇到了很多的困难，主要就是小鼠一直局限于一角，尝试了大量的调参和策略调整效果还是比较差。笔者未来希望能够优化目前的 DQN 算法，如使用 DDQN 算法、调整 QNetwork、重新调节探索策略等。