牧羊人

任务描述

在`farm.xml`中定义了环境：三只羊，一个大型羊圈，一个小型羊圈

主要任务：

Agent在大型羊圈内部出生，需要从出生点学习如何使用稻草吸引这三只羊进入指定的小型羊圈

学习目标：

- 学习从物品栏中拿出稻草（出生时默认不持有稻草）
- 学习如何靠近三只小羊
- 学习如何带领三只小羊进入小型羊圈

相关定义：

环境：21x21

动作：7种，

```
actionMap = {0: 'movenorth 1', 1: 'movesouth 1',
             2: 'moveeast 1', 3: 'movewest 1', 4: 'hotbar.2 1', 5: 'hotbar.1 1',
6: 'tp 0.5 4 0.5'}
```

胜负：

胜利	失败
本次游戏总步数小于100，且总reward值大于200	本次游戏总步数大于100，且总reward值小于0
本次游戏总步数大于100，但总reward值大于0	

reward值：

- 若预测动作不是“前后左右移动 / 拿出稻草”（即`actionMap`中的0，1，2，3，4），reward = -500
- 若预测动作是“拿出稻草”：
    - 若已经拿着稻草，则reward = -200
    - 否则，reward = 200
- 若预测动作是“前后左右移动”，但人物坐标并未变化（如到达护栏边，原地不动），则reward=-200
- 若人物坐标与羊相距较近（如4个单位），则reward = 500
- 若人物坐标与小型羊圈大门坐标距离小于50，则reward = 100
- reward -= 人物与羊的距离（鼓励人物与羊变得更近）
- reward -= 人物与小型羊圈大门的距离（鼓励人物走向小型羊圈）

训练方法

总方法：DQN，经验池大小1000

动作选择采用$\epsilon - greedy$策略

采用DQN双网络结构，全连接层采用(0,0.1)的正态分布初始化

```
p1 = nn.MaxPool2d(3)
f1 = nn.Linear(49, 16)
f2 = nn.Linear(16, num_actions)
```

- optimizer：Adam, lr = 0.005
- loss_fn：MSELoss

训练与测试

实验过程

在同一张地图中，共有100局游戏进行训练。

动作采样根据$\epsilon - greedy$策略，起始$\epsilon = 1$，之后随训练过程逐渐降到0.1（以ϵ的概率选择随机动作，以$1 - \epsilon$的概率选择最优动作）

一开始训练极易漏掉位于左上角的一只羊，导致人物只会反复吸引位于下方的两只羊

解决方法：一开始人工设置，使人物先移动到羊圈左上角，再开始自主训练。

在每一轮迭代时，DQN进行学习时再在内部嵌套10个epoch来更多训练网络

采用eval, target两个不同网络：

利用单网络的DQN结果与双网络相差不大，训练到第50轮时仍然出现average reward = -598.48, max reward = 119.8（LR = 0.005， MSEloss）

采用eval和target两个网络的DQN训练较慢，迭代到第50轮时效果依然不是很理想，到达羊身边后仍然有较大概率不能及时切换稻草来吸引羊。训练到第50轮在cpu至少需要训练1小时。且50轮-100轮之间，依然会出现很多average reward小于-800的较差表现，效果表现不稳定。

trial 1

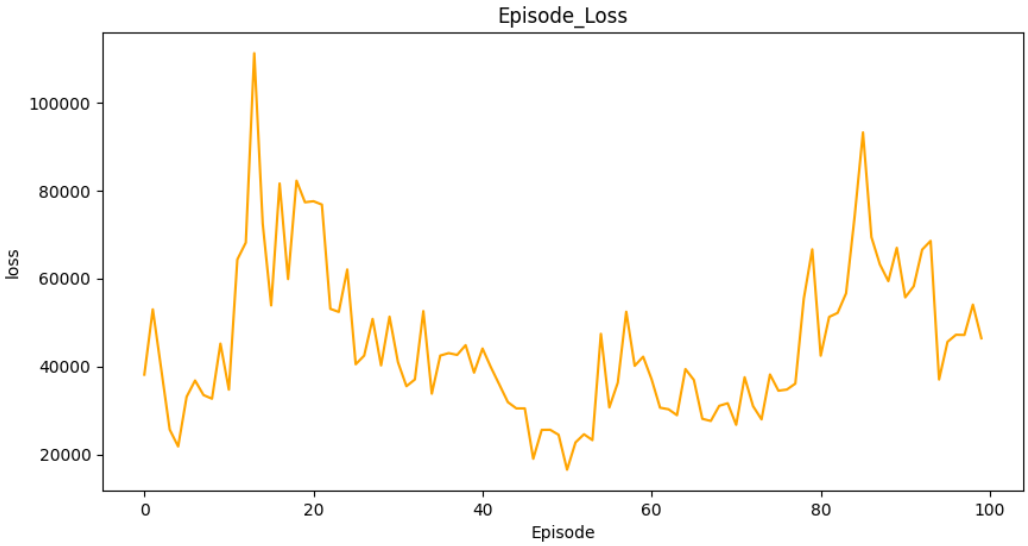
```
class Net(nn.Module):
    def __init__(self, num_actions):
        super(Net, self).__init__()
        self.c1 = nn.Conv2d(1, 1, 3, 3)
        nn.init.kaiming_normal_(self.c1.weight)
        self.f1 = nn.Linear(49, 16)
        self.f1.weight.data.normal_(0, 0.1)
        self.f2 = nn.Linear(16, num_actions)
        self.f2.weight.data.normal_(0, 0.1)

    def forward(self, x):
        x = self.c1(x)
        x = F.relu(x)
        x = x.view(x.size(0), -1)
        x = self.f1(x)
        x = F.relu(x)
        action = self.f2(x)
        return action
```

使用卷积层时，训练十分不稳定，难以收敛，100次训练后仍然出现人物原地打转，不能及时拿出稻草，自己进入小型羊圈后才拿出稻草等情况。在150轮训练后，极易出现原地打转现象，效果明显不如maxpooling层

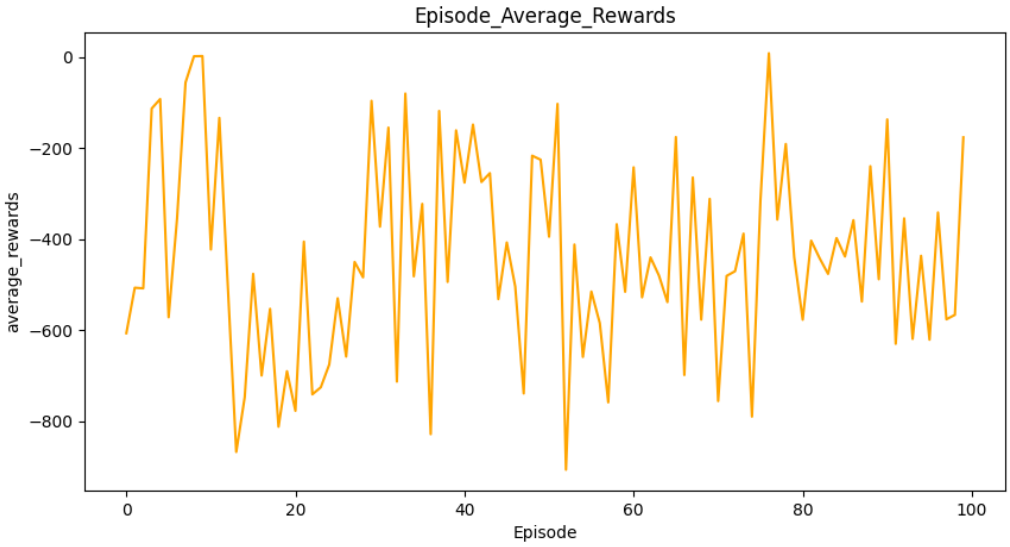
trial 2

```
class Net(nn.Module):
    def __init__(self, num_actions):
        super(Net, self).__init__()
        self.p1 = nn.MaxPool2d(3)
        self.f1 = nn.Linear(49, 16)
        self.f1.weight.data.normal_(0, 0.1)
        self.f2 = nn.Linear(16, num_actions)
        self.f2.weight.data.normal_(0, 0.1)

    def forward(self, x):
        x = self.p1(x)
        x = x.view(x.size(0), -1)
        x = self.f1(x)
        x = F.relu(x)
        action = self.f2(x)
        return action

optimizer = torch.optim.Adam(self.eval_model.parameters(), lr=0.005)
loss_fn = torch.nn.MSELoss()
```

采用maxpooling层，结合人物动作选择的一些trick，整体训练效果较好，能够带领三只小羊进入羊圈。



在100个episode的训练中，loss在刚开始降低非常明显，但在之后的波动程度仍然较大，同时因为该任务中动作选择更随机，地图更大，loss也会偏高。



经过100个episode训练，每轮的平均reward值逐渐上升。

实验中，我们希望每轮agent的动作数越少越好，实验证明我们的模型有效，人物的确每轮的动作逐渐下降，并且能顺利完成任务。

调参

经过不断的实验，如何找到与目标任务相匹配的网络结构和参数是一个非常困难的问题。对于简单任务（例如牧羊人中21x21的环境，3只羊作为交互对象），使用较为简单的全连接层效果可能比卷积层更好，且训练更快收敛。经实验，一些典型的RL任务调参问题体现如下：

- 对于DQN网络，如何调整到合适的replay-buffer大小，更新target network参数的频率非常困难，并且这些参数会和NN网络的参数相互影响。
- RL的学习曲线抖动很大，难以从早期训练（例如40个epoch）中及时发现问题并调参。
- 难以找到有效的方法去设计与任务较好匹配的DQN网络结构。
- 如何有效定义reward十分重要，对任务训练的表现影响极大。（在牧羊人实验中，定义的reward值可直接影响agent找到羊、拿出稻草、带领羊进入羊圈的效率；反之，若定义的不合适，则会出现agent不能找到全部的羊、不能及时拿出稻草、自己先单独进入羊圈等情况）

对于不同的任务，我们总结出了以下的一些训练和调参经验：

- 如果任务是求agent在环境下的最佳表现（一般情况下已知何种路线或方法为“最佳”），那么对于环境交互的定义、reward值的设置，比网络结构和其他参数更加重要

- 如果任务不是让agent在给定环境下求最佳表现（例如允许agent在环境中自由探索），那么人物在环境中的大量探索和轨迹采样更为重要。
- 对于一般的RL任务，DQN网络不宜设计复杂，只要网络结构能够学习target function即可。

思考与展望

在“牧羊人”任务中，我们可以清楚地看到DQN网络在面对较简单的任务时所表现出的过拟合现象，例如：有一头羊在羊圈左上角出现，但agent一旦学会如何引领地图中间的两只羊进入羊圈后，将几乎不可能再去寻找位于左上角的第3只羊。

与此同时，也存在着另外一个令人头疼的问题：在agent移动过程中，一旦 “偶尔” 选择了 “收起稻草” 的动作，那么该次行动中agent之后很难再选择动作 “拿出稻草” ，再去移动引领小羊进入羊圈，这会直接导致本次任务的失败（这种情况下agent会直接自己单独进入羊圈）。 但若规定对动作“收起稻草”进行较严厉的惩罚，那么在一开始的训练过程中因为agent会经常选择此动作，导致reward值很小，网络较难收敛。

针对问题一，我们采取先人为的让agent移动到左上角，再去自主选择动作。

针对问题二，目前还没有找到较好的定义reward值，去平衡agent针对“收起稻草”动作选择的合理性和网络训练的有效性。虽然若人为规定agent需要先“拿出稻草”，之后只能进行“前后左右移动”，会有效避免此问题，但这样就会导致引入了一个相当强的人为定义的“规则”，使得AI可学习性大大降低（面对“牧羊人”中固定的地图环境，完全可以人为设定动作和移动路线，agent进行操作即可）。

如何定义适合的网络参数和环境交互的reward值，仍是我们在未来的工作中需要努力解决的问题。同时面对特定的任务，我们也需要添加适当的人为制定的规则，帮助ai有更好的选择和表现。