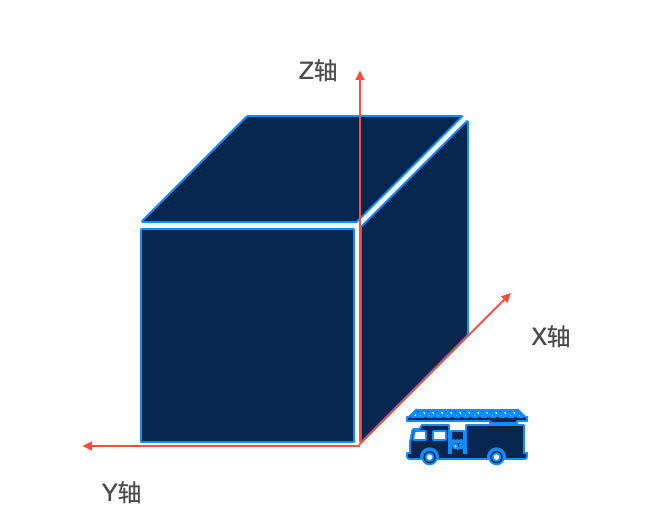
# 履带起重机吊装路径规划方法和控制装置

## 问题抽象

    首先将问题抽象出吊机作业区域，吊机动作，起始结束点三个方面。 为更加清楚叙述问题，以吊机所在位置为原点，我们建立一个三维坐标系。z轴为垂直地面向上为正，y轴为挖掘机面向方向为正，x轴已与y轴垂直，挖掘机右手方向为正。

* 作业区域

     挖掘区域是一个具有长度，宽度，高度的立方区域，我们使用一个三维矩阵来表示作业区域。 第一维是z轴信息，即高度信息。 第二维是y轴信息，即挖掘机面向方向信息。 第三维是x轴信息，即挖掘机右手方向，与x轴垂直。



作业区域

* 挖掘动作

     动作设计使用**三次插值法**来确定挖掘轨迹，使挖掘轨迹更加平滑，确定轨迹内的矩阵元素，将轨迹所含元素值在矩阵内全部更改为*0*。 目前使用三种经典动作来进行挖掘。

## 解决方法

     基于优化吊机路径规划及降低劳动成本方面考虑，我们使用深度强化学习算法**Deep Q Learning**来实现履带起重机无人自动化作业。 **Deep Q Learning**算法基于强化学习算法**Q-Learning**。 **Q-Learning**是强化学习算法中**value-based**的算法，**Q**即为**Q(s,a)**就是在某一时刻的 **s** 状态下， 采取动作**a** 动作能够获得收益的期望，环境会根据**agent**的动作反馈相应的回报**reward r**，所以算法的主要思想就是将**State**与Action构建成一张**Q-table**来存储**Q**值，然后根据**Q**值来选取能够获得最大的收益的动作。

|  |  |  |
| --- | --- | --- |
| **Q-Table** | **a1** | **a2** |
| s1 | q(s1,a1) | q(s1,a2) |
| s2 | q(s2,a1) | q(s2,a2) |
| s3 | q(s3,a1) | q(s3,a2) |

* 模型使用

     当我们使用**已训练好**的智能体时，**Q-table**已经固定，当智能体处在某环境s时，通过查询**Q-table**比较得到回报最高的动作。

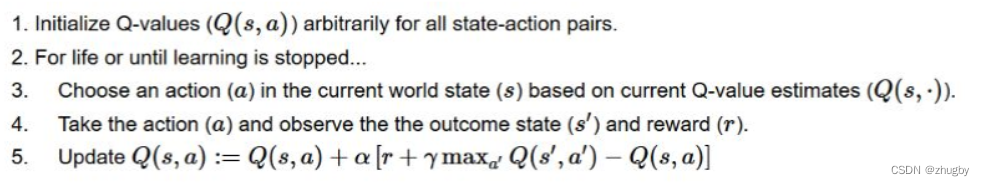
其中,是我们的策略,而我们最终目的是当我们完成目标时所得到的累积奖赏值达到最大化。

* 模型训练

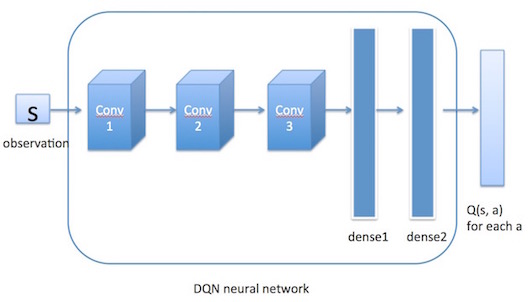
     在**Q-learing**训练中，最重要的问题是**Q-table**里环境与动作所对应单元格值更改的公式，**Q-table**单元格里的值的大小决定我们在当前环境里使用的动作。这里我们使用的公式是

这个公式指的就是智能体在当前状态s下，选择动作*a*时所对应的*Q(s,a)*等于当前时刻所取得的奖励 加上之后所能取得的最大奖励。 其中是我们设置的衰减值，代表下一步的回报。

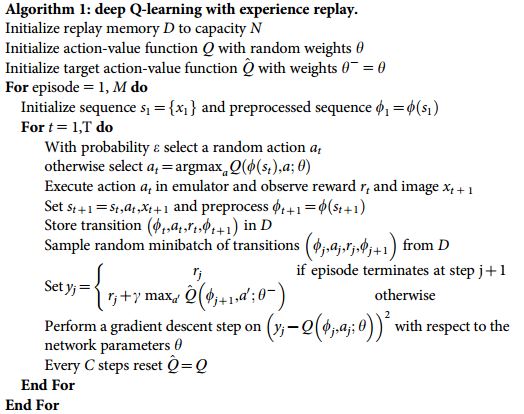
* 学习流程

1. 创建并初始化一个action-space\*state space大小的Q表，一般初始化设置所有值为0；
2. 进入循环，直到达到迭代条件：
3. 检索Q表，在当前状态 s下根据Q的估计值和Policy选择一个action a；
4. 执行action a，检索Q表，转移到的状态对应的Q最大值加上该动作得到的实时奖励reward是状态 s价值的真实值；
5. 根据策略方程更新Q表。 

由于状态空间与动作空间过大，使用一张表来实现，表的占用内存过大及效率问题，我门选择使用神经网络近似模拟值函数**Q(s,a)**。

其中w代表卷积神经网络。       通过神经网络计算出值函数后，DQN使用策略来输出action。值函数网络与策略之间的联系是这样的：首先环境会给出一个obs，智能体根据值函数网络得到关于这个obs的所有Q(s,a)，然后利用ϵ−greedy选择action并做出决策，环境接收到此action后会给出一个奖励及下一个obs。这是一个step。此时我们根据Rew去更新值函数网络的参数。接着进入下一个step。如此循环下去，直到我们训练出了一个好的值函数网络。

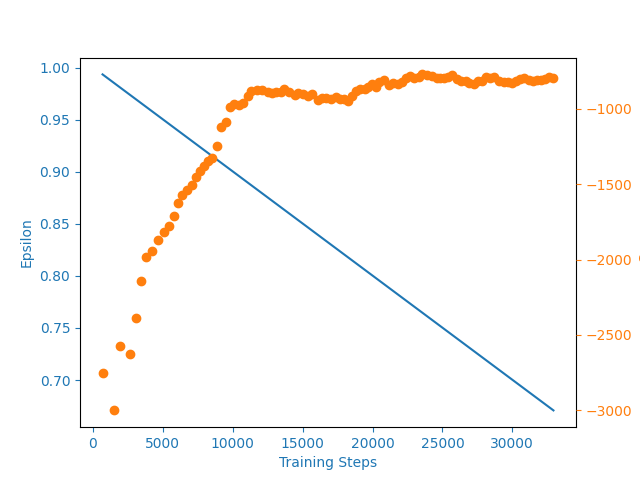
* DQN算法整体流程

1. 首先初始化Memory D，它的容量为N;
2. 初始化Q网络，随机生成权重ωω;
3. 初始化target Q网络，权重为ω−=ωω−=ω;
4. 循环遍历episode =1, 2, …, M:
5. 初始化initial state S1S1;
6. 循环遍历step =1,2,…, T:
   * 用ϵ−greedyϵ−greedy策略生成action atat：以ϵϵ概率选择一个随机的action，或选择at=maxaQ(St,a;1. ω)at=maxaQ(St,a;ω);
   * 执行action atat，接收reward rtrt及新的state St+1St+1;
   * 将transition样本 (St,at,rt,St+1)(St,at,rt,St+1)存入D中；
   * 从D中随机抽取一个minibatch的transitions (Sj,aj,rj,Sj+1)(Sj,aj,rj,Sj+1)；
   * 令yj=rjyj=rj，如果 j+1j+1步是terminal的话，否则，令 yj=rj+γmaxa′Q(St+1,a′;ω−)yj=rj+γmaxa′Q(St+1,a′;ω−)；
   * 对(yj−Q(St,aj;ω))2(yj−Q(St,aj;ω))2关于ωω使用梯度下降法进行更新；
   * 每隔C steps更新target Q网络，ω−=ωω−=ω。
7. End For;
8. End For; 算法流程图 

\*实验设置

     在本次实验中，我们首先采取2d环境进行模拟实验，去掉宽度这一限制，默认宽度为1。 我们智能体参数设置

|  |  |  |
| --- | --- | --- |
| **参数名** | **值** | **说明** |
| lr | 0.0001 | Learning rate for optimizer |
| eps | 1.0 | Starting value for epsilon in epsilon-greedy action selection |
| eps\_dec | 1e-6 | Linear factor for decreasing epsilon |
| eps\_min | 0.1 | Minimum value for epsilon in epsilon-greedy action selection |
| max\_mem | 50000 | Maximum size for memory replay buffer |
| bs | 32 | Batch size for replay memory sampling |
| replace | 1000 | Batch size for replay memory sampling |
| seed | 1 | random seed |

使用DQN算法对作业环境进行强化学习，得出收敛结果。 

     对于将挖掘区域全部挖掘的好坏进行评价，我们采用了两个评价标准。 一个是作业的速度，与人工作业进行对比。 一个是作业的效率，即总共所使用的步数。