《计算智能与控制》实验报告

基于深度强化学习的
倒立摆自动起摆

姓　名：	王昕宇
学　院：	空天科学学院
学　号：	XS18073005
专　业：	航空宇航科学与技术

一、实验目的和要求

经过一个学期的学习，同学对智能控制的各个研究领域有了大致的了解。为了进一步理解和掌握智能控制的原理方法，特设置实验内容。希望通过动手实践，将智能控制方法应用到实践当中。

二、实验内容和原理

深度强化学习(DRL)作为一种新型的基于机器学习的智能决策算法，在围棋、机器人控制等领域大放异彩，而在倒立摆自动起摆的控制中，由于系统强非线性的特征，目前的控制策略主要以摆杆能量建模控制为主，针对不同倒立摆系统需要重复建模。本文将深度强化学习应用到倒立摆的自动起摆的控制当中，实现了免模型控制，即该算法设计过程中无需了解摆杆、小车的物理参数。同时，通过修改算法部分参数，可以实现起摆过程中不同的控制要求，具有较强的适应能力和自主控制能力。

2.1 倒立摆稳定及起摆

倒立摆系统的稳摆控制是通过控制小车的往复运动，使摆杆稳定于正上方的倒立点位置，如图 2-1 所示。倒立摆系统的自动起摆控制是通过小车往复运动，使摆杆从静止下垂位置运动至垂直倒立位置，并稳定于垂直倒立位置。

倒立摆系统的稳摆控制可以在倒立点对倒立摆模型进行线性化处理，进而可以采用各种线性控制理论，研究的方法多样且有很大的进展。而倒立摆系统的自动起摆控制由于摆杆大范围运动，非线性很强，不能在全局范围内进行线性化处理，使倒立摆系统在自动起摆过程中呈现出很强的非线性和混沌运动，控制起来相当复杂。

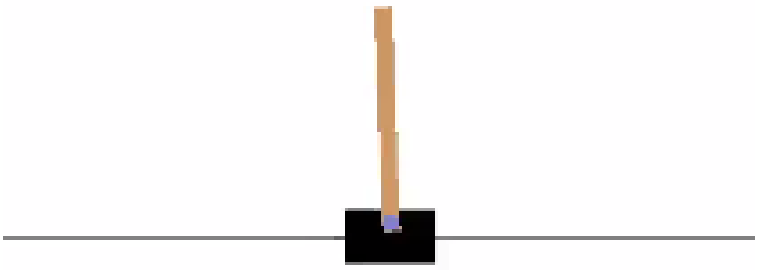

图 2-1 倒立摆系统

2.2 深度强化学习原理简介

首先介绍强化学习基本原理。如图 2-2 所示，智能体在完成某项任务时，首先通过动作 A 与周围环境进行交互，在动作 A 和环境的作用下，智能体会产生新的状态，同时环境会给出一个立即回报。如此循环下去，智能体与环境进行不断地交互从而产生很多数据。强化学习算法利用产生的数据修改自身的动作策略，再与环境交互，产生新的数据，并利用新的数据进一步改善自身的行为，经过数次迭代学习后，智能体能最终地学到完成相应任务的最优动作

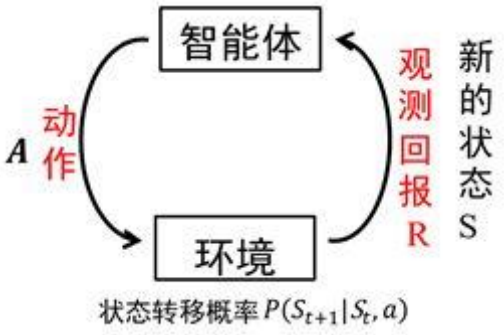

图 2-2 强化学习基本原理

让强化学习真正进入大众视野的是 DeepMind 团队将深度学习和强化学习结合形成深度强化学习算法，其中又以 DQN 算法最具代表性，本实验也采用了该算法，下面将对 DQN 算法原理做简要介绍。

DQN 算法以 Q-learning 算法为基础，而 Q-learning 算法的核心思想分为两部分即 off-policy（异策略）与 TD（Temporal-Difference 时间差分）。所谓异策略，是指行动策略（产生数据的策略）和要评估的策略不是一个策略。在图 2-3 Q-learning 伪代码中，行动策略是第 5 行的策略，而要评估和改进的策略是第 6 行的贪婪策略（每个状态取值函数最大的那个动作）。所谓时间差分方法，是指利用时间差分目标来更新当前行为值函数。在图 2-3 Q-learning 伪代码中，时间差分目标为$r_t + \gamma \max_{a} Q(s_{t+1}, a)$。

1. 初始化 $Q(s,a),\forall s\in S,\ a\in A(s)$,给定参数$\alpha$，$\gamma$

2. Repeat:

3. 给定起始状态 s，并根据 ε贪婪策略在状态 s 选择动作 a

4. Repeat(对于一幕的每一步)　　行动策略为 ε 贪婪策略

5. (a) 根据 ε 贪婪策略在状态 S_t 选择动作 a_t，得到回报r_t和下一个状态 S_{t+1}

6. (b) $Q(s_t,a_t)\leftarrow Q(s_t,a_t)+\alpha\left[r_t+\gamma\max_a Q(s_{t+1},a)-Q(s_t,a_t)\right]$　　目标策略为贪婪策略

7. (c) s=s'，a=a'

8. Until s 是终止状态

9. Until 所有的 $Q(s,a)$ 收敛

10. 输出最终策略：$\pi(s)=\operatorname{argmax}_a Q(s,a)$

图 2-3 Q-learning 伪代码

相比 Q-learning 算法，DQN 算法在以下三个方面做了改进：

（1）利用深度卷积神经网络逼近值函数

（2）利用了经验回放对强化学习的学习过程进行训练

（3）独立设置了目标网络来单独处理时间差分算法中的 TD 偏差。

图 2-4 给出 DQN 的伪代码。

[1] Initialize replay memory D to capacity N

[2] Initialize action-value function Q with random weights θ

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

[4] **For** episode = 1, M **do**

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

[6] **For** t = 1,T **do**

[7] With probability ε select a random action a_t

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

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

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

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

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

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

[14] Perform a gradient descent step on $\left(y_j-Q(\phi_j,a_j;\theta)\right)^2$ with respect to the

[15] network parameters θ

[16] Every C steps reset $\hat{Q}=Q$

[17] **End For**

[18] **End For**

图 2-4 DQN 伪代码

三、实验方法与步骤

3.1 实验基础环境

为了充分利用网络资源，减少重复的工作，本实验选用了常见的强化学习实验环境 Gym，使用 Gym 中的倒立摆模型来训练自己的控制算法。由于 Gym 只能运行在 Linux 系统上，所以安装了 Ubuntu16.04 系统。DQN 算法的核心是训练神经网络，本实验选用了常见的 TensorFlow 深度学习框架。图 3-1 展示了整个实验基础环境。

图 3-1 实验基础环境

3.2 修改 Gym 实现自起摆模型

开源的 Gym 环境中存在倒立摆的稳摆模型，本实验在该模型基础上修改得到自动起摆的训练模型。以下为该修改过程的主要步骤：

（1）图像引擎修改。修改摆杆的角度约束，使得摆杆能够在 0-360 度自由旋转，同时为了使整个摆杆都显示出来，需要修改画布大小。

（2）物理引擎修改。

由于稳摆控制中模型的输入只有两个-10N 和 10N 的推力，自起摆中修改为 -60N，-10N，0N，10N，60N。

建立倒立摆起摆全过程的非线性模型，设置摆杆在最低点时角度为 0，顺时针摆动为正。

（3）设置一次实验的起始状态和终止条件。将摆杆垂直向下设为角度为 0 的起始点，将摆杆大于 175 度或小于-175 度作为实验成功的终止条件。

（4）将新修改后的模型在 Gym 库中注册。以方便训练过程中的直接调用。

3.3 自起摆控制算法实现及训练

在 TensorFlow 中实现 DQN 算法即两个完全同结构的全连接网络，如图 3-2 所示。

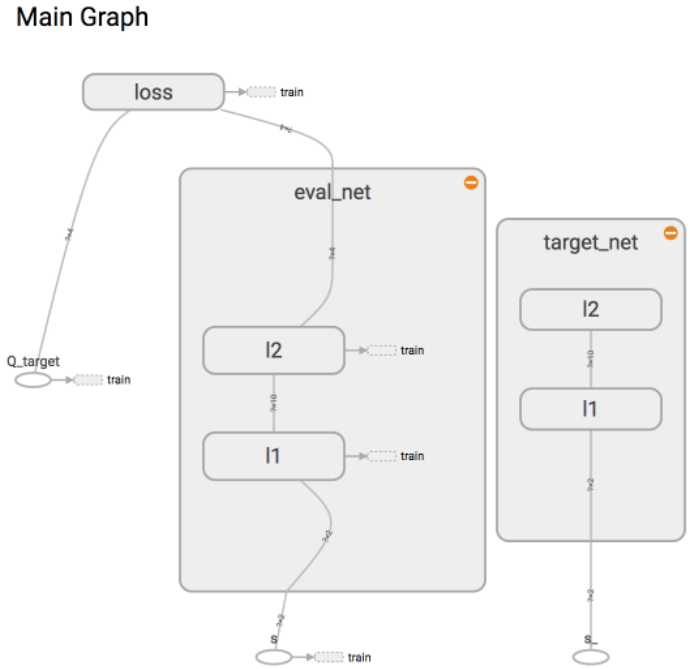

图 3-2 神经网络结构

根据 DQN 算法，实际产生控制指令的是 eval_net，即 eval_net 会接受当前状态（位置，速度，角度，角速度），输出控制指令（力的大小）。而 target_net 是在训练过程中为 eval_net 提供训练目标所用，详见 DQN 原理部分。

图 3-3 展示了 DQN 算法的训练过程，首先神经网络作为控制器直接接受当前状态产生动作，对倒立摆系统进行控制。在此过程中，将一个控制周期内的【状态，动作，奖励，状态】作为数据片存储在数据库中。当数据库中有足够的数据片后，从数据库中随机抽取一批数据片，对神经网络进行修改训练。修改后的神经网络会继续对倒立摆进行控制，从而产生新的数据片，而数据库的容量有限，新数据会顶替旧数据，实现数据更新。当前数据库中的数据会再次被抽取来训练神经网络。如此反复，直到网络参数不再发生变化，视为训练完成。

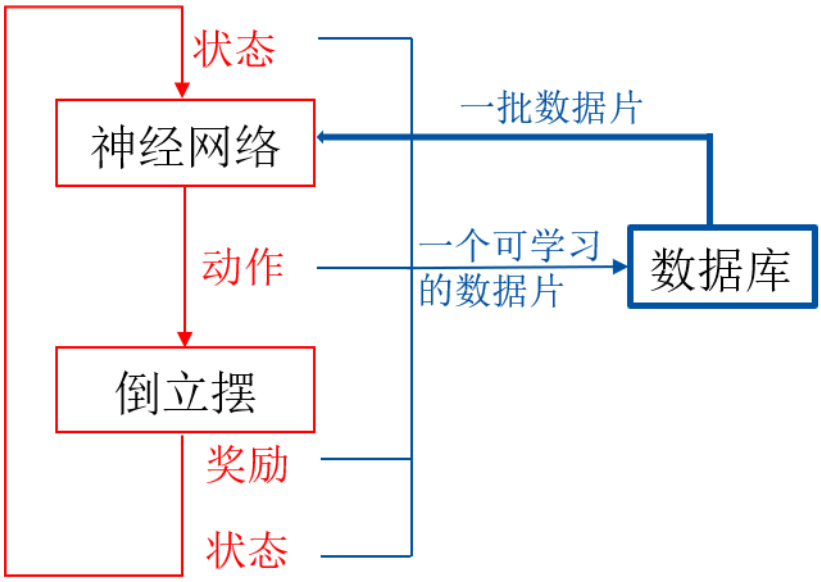

图 3-3 DQN 训练过程

四、实验结果与分析

强化学习的训练效果与设计的奖励函数直接相关，在本实验倒立摆自起摆中，将奖励函数设置为位置约束与摆角约束之和。下面将分别展示不同位置约束下的训练结果。

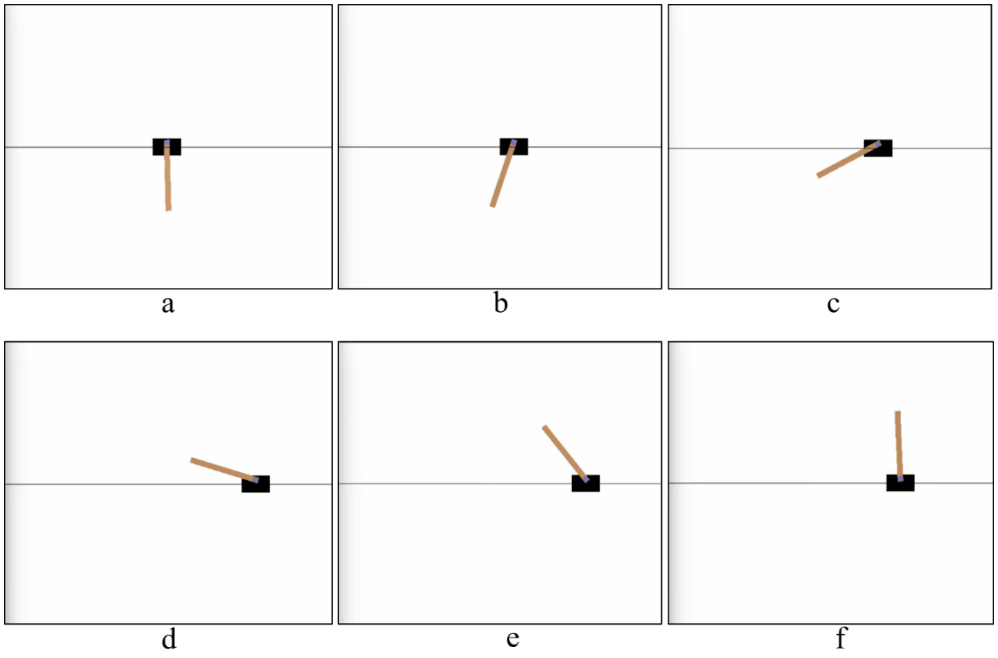

图 4-1 对位置约束小时摆动过程

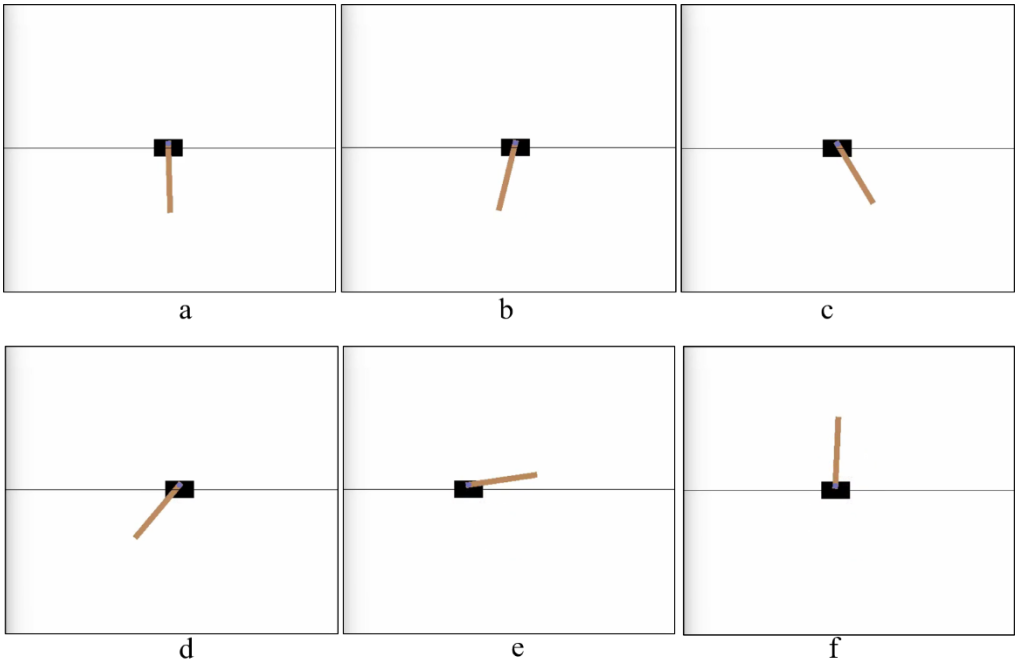

图 4-2 对位置约束大时摆动过程

图 4-1，由于对小车位置约束小，起摆过程从 a-f 一气呵成，直接起摆成功。而图 4-2，起摆过程从 a-f 小车在中间位置不断震荡，给摆杆积蓄能量，达到一定能量后起摆成功。

上述两种不同奖励函数的设置，直接导致了最后完全不同的控制策略，相比于传统算法有了更多了灵活性和自适应能力。同时实验也验证了强化学习在序列决策过程中的优越性，即能够放弃当前奖励而追求未来的更大奖励。

四、实验总结

整个实验过程前前后后加起来做了半个多月，而这半个月让我从强化学习的小白，成长为能够简单应用强化学习算法的入门者。在实验的过程中，我也被强化学习本身的强大能力所震惊，进而对该领域产生了极大的兴趣，希望以后能够深入钻研算法背后原理，为该领域贡献一些自己的力量。最后，感谢老师一学期的辛苦授课，不仅仅让我了解了智能控制的各种算法，更是让我知道了这些算法背后数学原理以及算法提出者的思维过程，这些将是我以后科研道路的宝贵经验。