基于强化学习的倒立摆控制策略研究

摘要

当控制系统是复杂非线性系统时，设计一类优化控制器是非常复杂的。强化学习是从与控制对象的交互中学习优化策略。本文采取强化学习方法，在未知倒立摆数学模型情况下，通过输入输出数据，实现对倒立摆的控制。

1. 引言

强化学习是一门决策学科，理解最佳的方式来制定决策。在工程控制当中有一门课程叫最优控制，与强化学习使用的方法有很大的类似之处，这种基于强化学习的方法不需要建模，也不需要设计控制器，只需要构建一个强化学习算法。当 RL 应用于系统时，智能体通过与系统交互学会采取行动，以便最大化一些累积奖励。学习可以基于不同形式的奖励反馈。与监督学习相比，强化学习的期望输出是不知道的。通过强化学习智能体与环境的交互得到一些列的输出，这些输出的好坏用来评判智能体学习的好坏。RL 算法关注在线学习性能，涉及到在探索（未知领域）和开发（当前知识）之间的平衡。为了获得最大的奖励，智能体必须利用它已经知道的知识，但是它也必须探索，以便将来做出更好的行动选择。

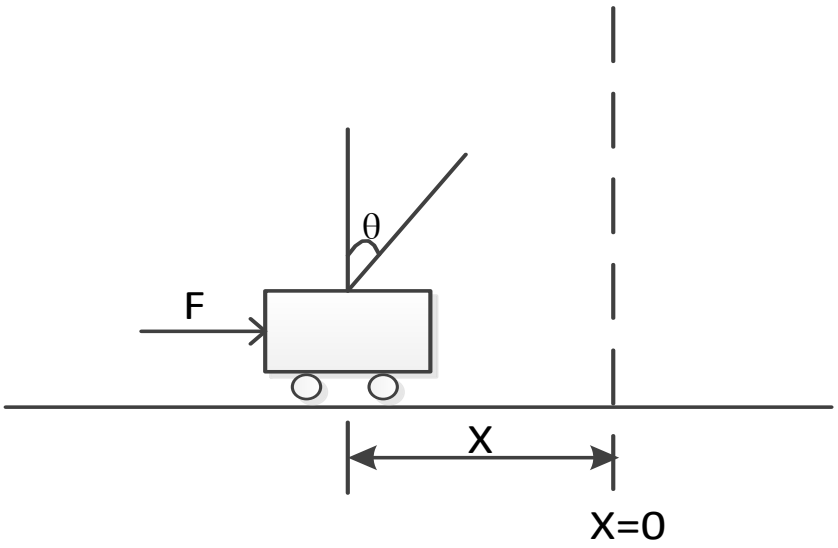
倒立摆问题是控制系统中一类经典的问题。它是一个固有的不稳定和欠驱动的机械系统。这个系统的动力学是用来更好理解平衡维护的任务，如火箭推进器的控制和自平衡的机械系统。 已有大量的文章研究了倒立摆的摆起与稳定控制的若干设计技术。像 PID 控制、线性二次型调节器（LQR）和模糊逻辑控制器。

系统复杂性的增加需要复杂的控制器，特别是在系统存在非线性、不确定性和时变时。由于其固有的本质，RL 使用来自环境的交互数据，生成一个最优控制器，而不需要环境本身的数学模型知识。此外，这种控制器具有适应环境发生扰动的能力。

下文结构组织如下：倒立摆问题在第二节被讨论了，第三节讨论了强化学习算法。第四节介绍了实验以及实验结果。

2. 倒立摆问题

倒立摆控制系统是一个复杂的、不稳定的、非线性系统。是进行控制算法验证的理想实验平台。能有效的反映控制中的许多典型问题：如非线性问题、鲁棒性问题、镇定问题、随动问题以及跟踪问题等。通过对倒立摆的控制，可以较好检验新的控制方法是否有较强的处理非线性和不稳定性问题的能力。



图一　倒立摆示意图

本文所采用的模拟倒立摆如图一所示，摆杆被铰链固定在车体的正中心，可左右灵活摆动，输入控制小车的力，根据其动力学方程可得到小车的位置，速度，角度，角加速度。

倒立摆的动力学方程。

$$\ddot{\theta}=\frac{g\sin\theta_t+\cos\theta_t[\frac{-F_t-ml\theta^2\sin\theta_t-\mu_c\,\mathrm{sgn}(x)}{m_c+m}]-\frac{\mu_p\theta}{ml}}{l[\frac{4}{3}-\frac{m\cos^2\theta_t}{m_c+m}]} \tag{1}$$

$$\ddot{x}=\frac{F_t+ml[\theta_t^2\sin\theta-\ddot{\theta}_t\cos\theta_t]-\mu_c\,\mathrm{sgn}(x_t)}{m_c+m} \tag{2}$$

设定方程中 $g=9.8m/s^2$（重力加速度），$m_c=0.711kg$（小车的重量），$m=0.209kg$（摆杆的质量），$l=0.326\mathrm{m}$（摆杆的质心与旋转中心的距离）

$\mu_c = 0$（小车和地面的摩擦系数）$\mu_p = 0$（摆杆和小车间的摩擦系数）。假定仿真时间间隔$\tau = 0.02s$。

3. 强化学习的理论基础

强化学习是智能体在环境给予的奖励的刺激下，逐步形成对刺激的预期，产生能获得最大利益的习惯性行为。以控制对象的动力学方程建立物理引擎，作为其环境交互对象，并定义其奖励，使智能体获取的奖励最大化，达到控制目的。

目前，解决强化学习问题最好的框架为马尔科夫决策过程。一个离散时间有限范围的折扣马尔科夫决策过程可表示为$M = (S, A, P, r, \rho_0, \gamma, T)$其中$S$为状态集，$A$为动作集，是转移概率，$r: S \times A \to [-R_{\max}, R_{\max}]$为立即回报函数，$\rho_0: S \to R$是初始$P: S \times A \times S \to R$状态分布$\gamma \in [0,1]$为折扣因子，$T$为水平范围$\tau$为一个轨迹序列，即$\tau = (s_0, a_0, s_1, a_1, \cdots)$，累计回报$R = \sum_{t=0}^{T} \gamma^t r_t$，若已知策略$\pi$则其状态值函数$\upsilon_\pi(s) = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s]$，状态行为值函数$q_\pi = E[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a]$。强化学习的目标是找到最优策略$\pi$，也就是最优的值函数，使得该策略下的累计回报期望最大，即$\max_\pi \int R(\tau) p_\pi(\tau) d\tau$。

在马尔科夫决策中采用利用贝尔曼最优性原理得到值函数的贝尔曼最优化方程：

$$\upsilon^*(s) = \max_a R_s^a + \gamma \sum_{s' \in s} P_{ss'}^a \upsilon^*(s') \tag{3}$$

$$q^*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} q^*(s', a') \tag{4}$$

$\upsilon^*(s)$为其状态值函数最优贝尔曼方程，$q^*(s,a)$为其状态-动作值函数的最优贝尔曼方程。

在基于模型的强化学习中可以利用动态规划的思想来解决。其核心为找到找到最优值函数。因为模型已知可知方程（3）是关于值函数的线性方程组，其未

知数的个数为状态的总数，用|s|表示。使用高斯-赛德尔迭代算法进行求解。即：

$$\upsilon_{k+1}(s)=\sum_{a\in A}\pi(a\mid s)\{R_s^a+\gamma\sum_{s'\in s}P_{ss'}^a\upsilon(s')\} \tag{5}$$

在无模型的强化学习中，采用蒙特卡罗的方法计算其状态值函数中返回值的期望，利用随机样本来估计期望。在计算值函数时，利用经验平均代替随机变量的期望。蒙特卡罗方法利用经验平均对策略值函数进行估计。当值函数被估计出来后，对于每个状态 s ，通过最大化动作值函数，来进行策略的改进。结合蒙特卡罗的采样法和动态规划的方法利用一步预测方法计算当前状态值函数，利用经验平均得出后续状态，从而得出无模型 TD 算法的值函数更新公式

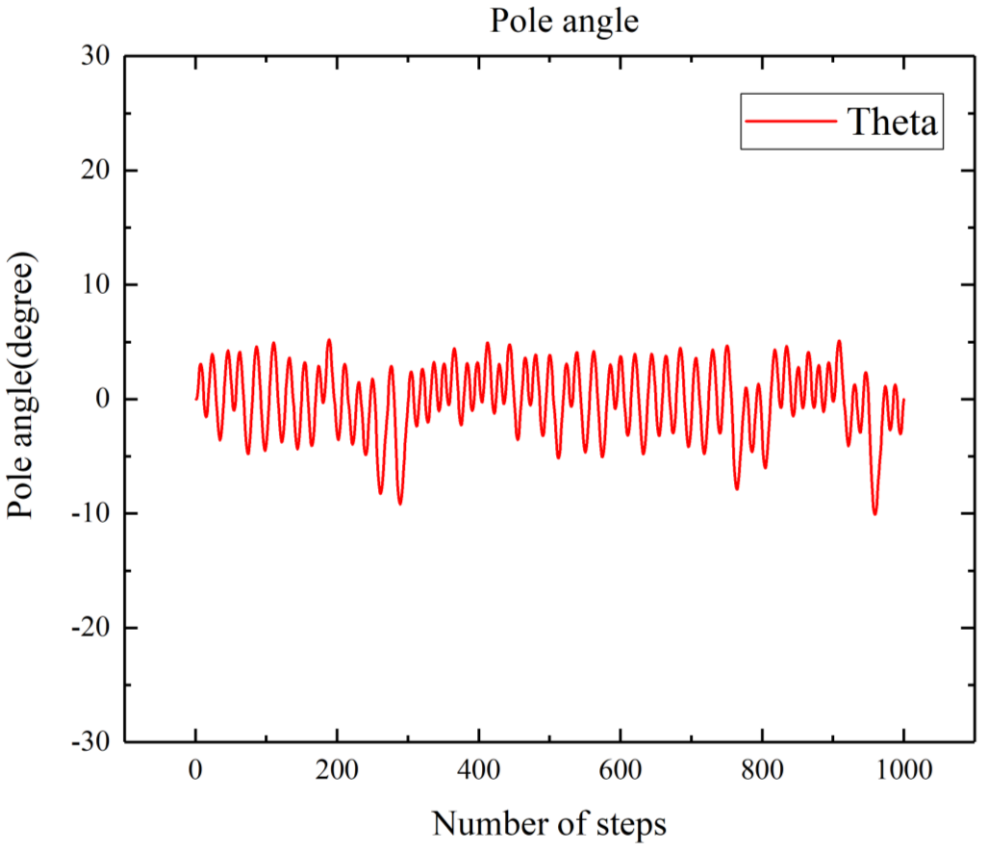
$$V(S_t)=V(S_t)+\alpha[R_{t+1}+\gamma V(S_{t+1})-V(S_t)] \tag{6}$$

$R_{t+1}+\gamma V(S_{t+1})$ 即为 TD 更新目标，$V(S_t)$ 为其值函数。与动态规划相比，不需知道模型，与蒙特卡洛相比，无需每次实验结束后更新，使其学习速度和学习效率大幅提高。Q-learning 采用异策略的 TD 算法。即行动策略采用 ε 贪婪策略，而目标策略采用最大贪婪策略。Q-learning 的算法更新公式如下：

$$Q(S,A)=Q(S,A)+\alpha[R+\gamma\max_{a}Q(S',a)-Q(S,A)] \tag{7}$$

4. 实验

输入接收物理引擎返回的状态 s ,获取的奖励为 r_t ，以及下一个状态定义为 s_{t+1} 。定义学习率为 α ，初始时贪婪度为 ε 。初始化物理引擎得到第一个状态 s_t ,，利用概率 ε 选一个随机动作。若果 ε 概率事件没发生，则用贪婪策略选择当前逼近值函数网络最大的那个动作 a_t 。在仿真器中执行动作 a_t ，观测回报 r_t 以及下一个状态 s_{t+1} 。判断是否是一个事件的终止状态，若是终止状态，则 TD 目标为 r_j 否则计算 TD 目标。进而循环迭代收敛 Q 表。取前 1000 次循环步数，得出如下结果示意图：



图二 倒立摆角度控制结果图

图三 倒立摆位置控制结果图

参考文献：

[1] Zadeh L A. Outline of a new approach to the analysis of complex systems and decision processes[J]. IEEE Transactions on systems, Man, and Cybernetics, 1973 (1): 28-44.

[2] Lin W S. Optimality and convergence of adaptive optimal control by reinforcement synthesis[J]. Automatica, 2011, 47(5): 1047-1052.

[3] Lin W S, Sheu J W. Optimization of train regulation and energy usage of metro lines using an adaptive-optimal-control algorithm[J]. IEEE Transactions on Automation Science and Engineering, 2011, 8(4): 855-864.

[4] Modares H, Lewis F L, Naghibi-Sistani M B. Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems[J]. Automatica, 2014, 50(1): 193-202.

[5] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with Deep Reinforcement Learning[J]. Computer Science, 2013.

[6] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540):529.

[7] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484-489.

[8] dos Santos Mignon A, da Rocha R L A. An Adaptive Implementation of ε-Greedy in Reinforcement Learning[J]. Procedia Computer Science, 2017, 109: 1146-1151.

[9] Lin L J. Reinforcement learning for robots using neural networks[R]. Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[10] Van Hasselt H, Guez A, Silver D. Deep Reinforcement Learning with Double Q-Learning[C]//AAAI. 2016: 2094-2100.

[11] Riedmiller M. Neural fitted Q iteration-first experiences with a data efficient neural reinforcement learning method[C]//ECML. 2005, 3720: 317-328.

[12] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. Cambridge: MIT press, 1998.

[13] Kharola A, Patil P, Raiwani S, et al. A comparison study for control and stabilisation of inverted pendulum on inclined surface (IPIS) using PID and fuzzy controllers[J]. Perspectives in Science, 2016, 8: 187-190.

[14] Dong Z, Song L, Chen H. The modeling and simulation of first-order Inverted pendulum control system[J]. Advances in Electronic Commerce, Web Application and Communication, 2012: 221-225.

代码附录：

cart_pole2.m

```
function [thetaNext,thetaDotNext,thetaacc,xNext,xDotNext] = cart_pole2(force,theta,thetaDot,x,xDot)
GRAVITY = 9.8;
MASSCART = 1;
MASSPOLE = 0.1;
TOTAL_MASS = (MASSPOLE + MASSCART);
LENGTH = 0.5;
POLEMASS_LENGTH = (MASSPOLE*LENGTH);
TAU = 0.02;
FOURTHIRDS = 1.3333333333333;
temp = (force + POLEMASS_LENGTH*thetaDot*thetaDot*sin(theta))/TOTAL_MASS;
thetaacc = (GRAVITY*sin(theta) - cos(theta)*temp)/(LENGTH*(FOURTHIRDS - MASSPOLE*cos(theta)*cos(theta)/TOTAL_MASS));
xacc = temp - POLEMASS_LENGTH*thetaacc*cos(theta)/TOTAL_MASS;
xNext = x + TAU*xDot;
xDotNext = xDot + TAU*xacc;
thetaNext = theta + TAU*thetaDot;
thetaDotNext = thetaDot + TAU*thetaacc;
return;
```

getBox4.m

```
function box = getBox4(theta,thetaDot,x,xDot)
theta = rad2deg(theta);
thetaDot = rad2deg(thetaDot);
if (x < -2.4 || x > 2.4  || theta < -12 || theta > 12)
    box = 163;
else

```

```
if (theta<-6&&theta>=-12)
    thetaBucket = 1;
elseif (theta<-1&&theta>=-6)
    thetaBucket = 2;
elseif (theta<0&&theta>=-1)
    thetaBucket = 3;
elseif (theta<1&&theta>=0)
    thetaBucket = 4;
elseif (theta<6&&theta>=1)
    thetaBucket = 5;
elseif (theta<=12&&theta>=6)
    thetaBucket = 6;
end

if (x<-0.8&&x>=-2.4)
    xBucket = 1;
elseif (x<=0.8&&x>=-0.8)
    xBucket = 2;
elseif (x<=2.4&&x>0.8)
    xBucket = 3;
end

if (xDot<-0.5)
    xDotBucket = 1;
elseif (xDot>=-0.5&&xDot<=0.5)
    xDotBucket = 2;
else
    xDotBucket = 3;
end

```

```
38. if (thetaDot<-50)
39.     thetaDotBucket = 1;
40. elseif (thetaDot>=-50&&thetaDot<=50)
41.     thetaDotBucket = 2;
42. else
43.     thetaDotBucket = 3;
44. end
45.
46. box = sub2ind([6,3,3,3],thetaBucket, thetaDotBucket,xBucket,xDotBucket);
47. end
48. return;
```

QLearningCartPole.m

```
1.  clc;
2.  clear all;
3.  close all;
4.  NUM_BOXES = 163;
5.  ALPHA = 0.5;
6.  GAMMA = 0.999;
7.  Q = zeros(NUM_BOXES,2);
8.  action = [10 -10];
9.  MAX_FAILURES = 1000;
10. MAX_STEPS = 150000;
11. epsilon = 0;
12. steps = 0;
13. failures = 0;
14. thetaPlot = 0;
15. xPlot = 0;
16. theta = 0;
17. thetaDot = 0;
```

```
x = 0;
xDot = 0;
box = getBox4(theta,thetaDot,x,xDot);

while(steps<=MAX_STEPS && failures<+MAX_FAILURES)
    steps = steps + 1;

    if(rand>epsilon)
        [~,actionMax] = max(Q(box,:));
        currentAction = action(actionMax);
    else
        currentAction = datasample(action,1);
    end

    actionIndex = find(action == currentAction);
    [thetaNext,thetaDotNext,thetaacc,xNext,xDotNext] = cart_pole2(currentAction,theta,thetaDot,x,x
Dot);

    thetaPlot(end + 1) = thetaNext*180/pi;
    xPlot(end + 1) = xNext;
    newBox = getBox4(thetaNext,thetaDotNext,xNext,xDotNext);
    theta = thetaNext;
    thetaDot = thetaDotNext;
    x = xNext;
    xDot = xDotNext;
    if(newBox==163)
        r = -1;
        Q(newBox,:) = 0;
        figure(2);
        plot((1:length(thetaPlot)),thetaPlot,'-b');
```

```
47.         figure(3);
48.         plot((1:length(xPlot)),xPlot,'-b');
49.
50.         thetaPlot = 0;
51.         xPlot = 0;
52.         theta = 0;
53.         thetaDot = 0;
54.         x = 0;
55.         xDot = 0;
56.         newBox = getBox4(theta,thetaDot,x,xDot);
57.         failures = failures + 1;
58.         fprintf('Trial %d was %d steps. \n',failures,steps);
59.         figure(1);
60.         plot(failures,steps,'-b');
61.         hold on;
62.         steps = 0;
63.     else
64.         r = 0;
65.     end
66.     Q(box,actionIndex) = Q(box,actionIndex) + ALPHA*(r + GAMMA*max(Q(newBox,:)) - Q(box,actio
    nIndex));
67.     box = newBox;
68.  end
69.  if(failures == MAX_FAILURES)
70.     fprintf('Pole not balanced. Stopping after %d failures.',failures);
71.  else
72.     fprintf('Pole balanced successfully for at least %d steps\n', steps);
73.     figure(1);
74.     plot(failures+1,steps,'-b');
75.     title('failures+1','FontSize',16);
```

```
    hold on;
    figure(2);
    chouxi = 1:length(thetaPlot);
    plot((1:length(thetaPlot)),thetaPlot,'-b');
    title('thetaPlot','FontSize',16);
    figure(3);
    plot((1:length(xPlot)),xPlot,'-b');
    title('xPlot','FontSize',16);
    figure(4);
    plot((1:301),thetaPlot(1:301),'-b');
    title('thetaPlot-301','FontSize',16);
    hold on;
    figure(5);
    plot((1:301),xPlot(1:301),'-b');
    title('xPlot-301','FontSize',16);
    hold on;
end
```