倒立摆的神经网络和强化学习控制

刘平

2019 年 8 月 2 日

目录

1 倒立摆的动力学模型推导

倒立摆是比较经典的控制问题, 可以分为一级倒立摆和多级倒立摆, 本文讨论的是一级倒立摆, 因此下文中所说的倒立摆都是指一级倒立摆。倒立摆的模型可以表述为一个滑块上面装有一根铰接的杆, 如图 1。

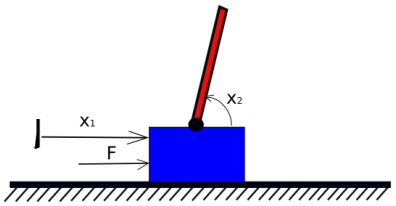

图 1: 一级倒立摆示意图

下面采用拉格朗日方程推导倒立摆的动力学方程。首先设滑块的位移为 x_1, 质量为 m_1, 杆的角位移为 x_2, 质量为 m_2, 杆长为 l。现有一个控制力 F 作用于滑块上。滑块与地面之间的摩擦力不计。系统的势能, 动能和拉格朗日函数分别为为

$$T = \frac{1}{2}(m_1+m_2)\dot{x}_1^2 + \frac{1}{2}m_2 l\dot{x}_1\dot{x}_2 sin(x_2) + \frac{7}{24}l^2\dot{x}_2^2 m_2 \quad (1)$$

$$V = m_2 g sin(x_2) \quad (2)$$

$$L = T - V \quad (3)$$

由拉格朗日方程可得

$$\frac{\mathrm{d}\frac{\partial L}{\partial \dot{x}_1}}{\mathrm{d}t} - \frac{\partial L}{\partial x_1} = (m_1+m_2)\ddot{x}_1 - \frac{1}{2}m_2 l\ddot{x}_2 sin(x_2) - \frac{1}{2}m_2 l\dot{x}_2^2 cos(x_2) = F \quad (4)$$

$$\frac{\mathrm{d}\frac{\partial L}{\partial \dot{x}_2}}{\mathrm{d}t} - \frac{\partial L}{\partial x_2} = -\frac{1}{2}m_2 l\ddot{x}_1 sin(x_2) + \frac{7}{12}l^2\ddot{x}_2 m_2 + \frac{l}{2}m_2 g cos(x_2) = 0 \quad (5)$$

式 (5) 可以化简为

$$-\ddot{x}_1 sin(x_2) + \frac{7}{6}l\ddot{x}_2 + g cos(x_2) = 0 \quad (6)$$

则可得系统的动力学方程为

$$\begin{cases} (m_1+m_2)\ddot{x}_1 - \frac{1}{2}m_2 l\ddot{x}_2 sin(x_2) - \frac{1}{2}m_2 l\dot{x}_2^2 cos(x_2) = F \\ -\ddot{x}_1 sin(x_2) + \frac{7}{6}l\ddot{x}_2 + gcos(x_2) = 0 \end{cases} \tag{7}$$

2 倒立摆的 PID 控制器

选取状态变量 $x_1, x_2, \dot{x}_1, \dot{x}_2$, 则可以把方程组 (7) 写成如下的状态方程形式:

$$\begin{cases} \dot{x}_1 = \dot{x}_1 \\ \dot{x}_2 = \dot{x}_2 \\ \ddot{x}_1 = \frac{7}{6sin(x_2)}l^2\ddot{x}_2 + \frac{gcos(x_2)}{sin(x_2)} \\ \ddot{x}_2 = \frac{Fsin(x_2) - (m_1+m_2)gcos(x_2) + \frac{1}{2}m_2 l\dot{x}_2^2 cos(x_2)sin(x_2)}{\frac{7}{6}l(m_1+m_2) - \frac{1}{2}m_2 lsin^2(x_2)} \end{cases} \tag{8}$$

F 为控制对象, 控制目标是使杆与滑块的夹角为 90 度, 由 PID 控制器原理设计控制器如下

$$F = P(x_t - x_2) + Ix_I + D\dot{x}_2 \tag{9}$$

其中 $x_t = \frac{\pi}{2}$,x_I 为角度的积分误差,P, I, D 为对应的 PID 参数。仿真结果如图 2

3 基于 PID 控制器倒立摆的神经网络控制器

对于 PID 控制器可以看成一个输入状态到输出状态的映射, 而神经网络对于输入输出映射具有很好的学习能力。这里采用 PID 的计算结果训练出一个神经网络。这里的选择输入参数为滑块的位移 x_1, 滑块的速度 $\dot{x}_1$, 杆的角度 x_2, 杆的角速度 $\dot{x}_2$, 以及杆角度误差的积分 x_I, 输出为力 F。定义神经网络的结构为 $5 \times 10 \times 1$ 的全连接网络, 其中 5 为输入单元个数,10 为隐藏层单元个数,1 为输出层单元个数。激活单元为 relu, 即 $f(x) = max(x, 0)$。神经网络结构如图 3。

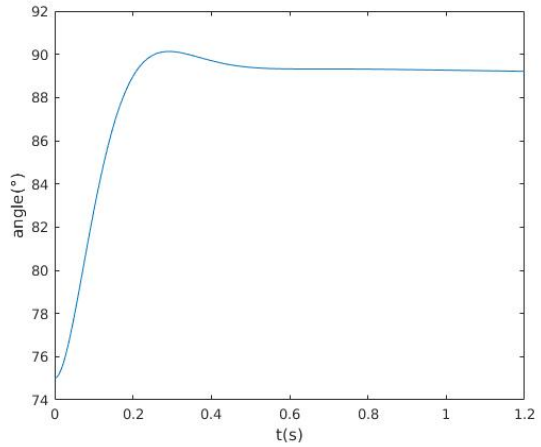

图 2: PID 控制器仿真结果

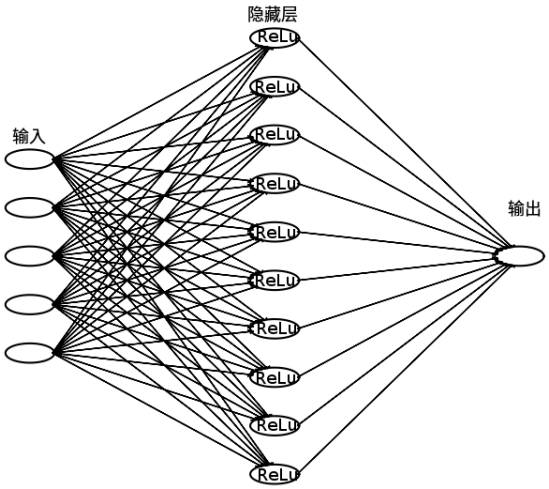

图 3: 神经网络结构

4 基于 Q-learning 的控制器

强化学习算法的基本概念是基于 Agent 与环境之间交互而来的一个概念，如图 4。在 t 时刻 Agent 内部基于 $t-1$ 时刻的状态的描述 S^a_{t-1}，上标

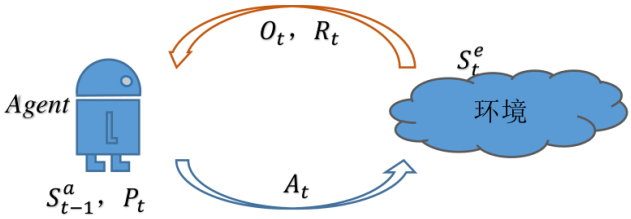

图 4: Agent 与环境交互示意图

a 代表 Agent 和基于行为选择策略 P_t 产生行为 A_t，在执行完动作 A_t 后环境的状态为 S^e_t，Agent 获得回报 R_t 并观察到 O_t，然后基于 O_t 更新 t 时刻的 S^a。通常为了简化问题，把 S^a_t 等价于 O_t，同时不考虑 S^e_t，而把 S^a_t 记为 S_t。在强化学习模型中 P_t 是强化学习的最终想获得的东西，即获得一个应对于环境比较好的一个行为策略。对于 P_t 可以把看成一个关于状态 S 的一个函数，即 $P_t = \pi(S_{t-1}, S_{t-2}, \cdots)$，根据惯例这里用 π 来表示策略函数。在强化学习中有两个关键的函数一是 $V(s)$，二是 $Q(s,a)$。$V(s)$ 是对状态的好坏的一个估计，而 $Q(s,a)$ 是对在某个状态下采取某个动作好坏的估计。基于 $V(s)$ 再加上某些基于次的行为策略即可构建构建出一个 $\pi(s)$，基于 $Q(s,a)$ 再加上一些选择策略也可以构建出 $\pi(s)$。在学习开始之前对于状态 s 的 $V(s)$ 和 s,a 对应的 $Q(s,a)$ 并不确定。在不断的试错的过程来不断更新 $V(s)$ 和 $Q(s,a)$。有多种不同的方法来更新 $V(s)$ 和 $Q(s,a)$。这里主要讨论的是 Q-learning 算法。

Q-Learning 算法细节如图 5。从更新公式

$$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)] \tag{10}$$

中可以看出更新 Q 值的方法是基于采取 a 后获得的回报和下一个状态下的最大 Q 值之间的差值来更新的，因而这种方法也是时间差分算法 $(TD(0))$ 的一种。

对于倒立摆问题，首先是选择合适的状态 S 和行为 A 的合理的表示方

Q-learning: An off-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

图 5: Q-Learning 算法

法,对于行为 A 基于 PID 的控制输出这里设置 A 的取值为 $[-120, -100, -80, -60, -40, -20, -10, -6, -1, 0, 1, 6, 10, 20, 40, 60, 80, 100, 120]$, 对于 s 开始只是选取倒立摆的偏离角度 θ 为状态变量，得到的结果如图 6，图中横坐标为训练周期数，纵坐标为倒立摆在有效角度持续的时间，单位为 $dt = 1/30s$，图中蓝色的曲线为一次训练的结果，可以看出震荡剧烈，黄色的线为平均 20 次训练后的结果，可以看出虽然后面持续时间交前面略有上升，但是训练的效果还不是特别明显。只是如果只对基于角度的 S 进行强化学习训练，对应于 PID 控制中的 P 控制器，效果并不会很好。基于此，在状态变量中增加了角度变化量，即 $S = (\theta, \dot{\theta})$。最后的结果如图 7。由图中可以看出迭代次数明显减少，在 600 次便可以找到最优的控制策略，在训练过程中限制了最长运行步数，运行 100 个 dt 便视为达到稳定状态，然后停止。

5 基于 DQN 的控制器

对于倒立摆如果采用角度 θ 和 $\dot{\theta}$ 来描述，则如上所述采用数量较小的离散化来构建有限的状态空间 $\mathcal{S}$ 便可以取得比较好的效果。如果对于复杂控制问题或者初始输入的状态空间较大情形下，上面的基于表格的 Q-Learning 方法就显得力不从心。例如，考虑 Atair 中的打砖块游戏，如果以图像来构建状态空间 $\mathcal{S}$，假设游戏的分辨率为 48x32，基于 RGB 三色显示，则状态空间的大小为 $2^{48\times32\times3} = 2^{4608}$，大约是宇宙原子总数的 10^{1300} 倍。因而就需要更加有效的方法来解决状态空间较大的问题。神经网络已经在图像处理、自然语言处理等领域取得了突破性的进展，其对于数据维数增大带来的问题具有很强的鲁棒性。DQN 基于此而出现。DQN 的算法如下

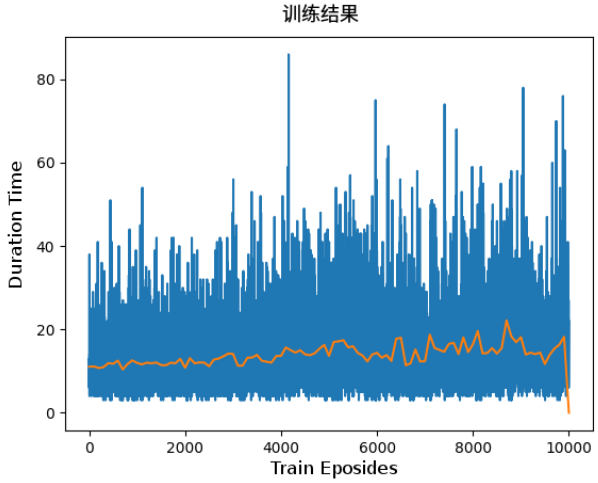

图 6: S 基于角度 θ 的情况下的训练结果

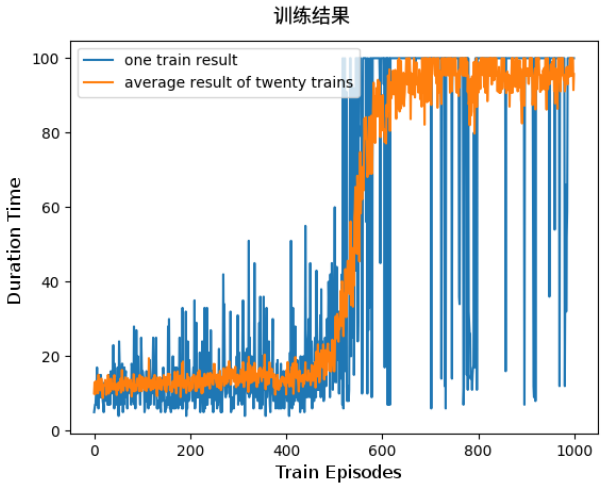

图 7: S 基于角度 θ 和角速度 $\dot{\theta}$ 的情况下的训练结果