清华大学机械工程系

机器人技术与应用

2018-2019 秋季学期

强化学习算法介绍

机器人技术与应用技术报告

机械 51 王芮 2015010445

2019 年 1 月 5 日

目　录

1 强化学习与机器人的关系

2 理论基础

2.1 强化学习文献综述

在强化学习的著作 *Reinforcement Learning An Introduction*[?] 中，Sutton 对强化学习的理论进行了详尽的讨论。在这里，我们对强化学习的发展做一个精炼的综述，并从契合课题的框架进行一些讨论。另外，我们还将详细讨论强化学习的一些最新进展和它们在本课题中的应用。

2.1.1 问题描述

Sutton 在 1988 年首次提出强化学习问题的一般描述[?]。诸如学习走路、学习下棋的一般性问题都可以归为强化学习的问题。强化学习问题的四个基本元素是：一个策略（policy），一个奖励函数（reward），一个价值函数（value function）和一个环境模型（model）。例如，自动驾驶问题中，智能体面对它观察到的一个周边车流的**环境**，会给出一个输出，例如以某一加速度加速。在采取这一行动之后，智能体会得到一个**奖励**（可能有延迟），例如它到达了目的地，我们给它价值 100 的奖励。而如何决定加速度，就是我们想要智能体去学习的**策略**。在制定策略的过程中，智能体会去试图衡量目前所处状态的**价值**（价值函数），例如周围车辆非常拥堵，这个状态的价值就比较低，是我们希望避免进入的状态；而车前非常空旷则可能是我们赋予价值较高，希望进入的状态。

在这里，我们将用一些研究者所默认的记号对我们的问题进行数学抽象。假设下标 t 表示任一时刻，而 S_t 代表某一时刻的状态，A_t 表示智能体采取的行动，π 表示智能体的策略，R_t 表示智能体某一步获得的奖励。我们的目标是让智能体通过学习，获得尽可能大的总奖励，记为 G。有时我们可以定义

$$G = \sum_t R_t$$

但是多数时候我们会考虑加入一个折扣 $\gamma(0 < \gamma < 1)$，认为一个行为直接获得的奖励对它更有意义，而很久以后获得的奖励与这个行为的关联越来越小，记为

$$G_{t_0} = \sum_{t=t0}^{t_n} \gamma^{t-t_0} R_t$$

其中 t_n 要么是一个过程因为某种原因终止的时刻，要么是 ∞。

强化学习解决的是一个马尔可夫决策过程（MDP）[?]。在 MDP 中，S_{t+1} 只取决于 S_t 和 A_t，而不受到之前历史过程的影响。也就是说，S_t 状态本身就包含了它所有的历史对后续过程的影响。在这个假设下，我们可以这样表达强化学习的目标[?]：记 $v^{\pi}(s)$ 为状态 s 的价值函数。我们有

$$v^{\pi}(s) = E[G_{t_0}|S_{t_0} = s, \pi(s)]$$

$$J(\pi) = \max_{\pi} E(v^{\pi}(s))$$

$$\pi^*(s) = \operatorname*{argmax}_{\pi} J(\pi)$$

强化学习的终极目标就是找到这个 $\pi^*(s)$。Bellman 在 *Dynamic Programming* 一书中提出了著名的 Bellman 最优性等式[?]

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \tag{1}$$

同时，我们用 $q(s, a)$ 表示在状态 s 下采取行动 a 可以获得的价值，那么 Bellman 最优性等式又可以表示为

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')] \tag{2}$$

这两个表达式在后续的推导中会非常有用。

2.1.2 典型算法综述

强化学习包含许多的算法，也有很多分类方法。其中，有一大类是列表方法，可以理解为是把 $v(s)$ 保存起来的方法；另一大类是近似方法，也就是用一个函数去近似 $v(s)$[?]。其中，第二类方法在近年来使用最多，神经网络使得函数可以近似非常复杂的映射关系，也让许多复杂问题通过强化学习得到了解决。

列表式的强化学习方法包括动态规划[?]，蒙特卡罗方法，时序差分学习（Temporal-difference learning，简称 TD 方法，包括 Sarsa[?], Q-learning[?][?]，多步自助法（n-step bootstrapping）[?]，Dyna 规划方法[?] 等。

TD 方法是列表法强化学习的核心。这一族的算法有许多衍生版本，但其核心是利用经验不断更新价值函数 $V(s)$，直至其收敛。更新的基本模式为

$$V(s) \leftarrow V(s) + \alpha[r + V(s') - V(s)]$$

其中，α 表示函数更新的步幅。而当我们用 $Q(s,a)$ 去替代 $V(s)$ 的时候，算法就成为了 Sarsa（state, action, reward, state, action[?]）：

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$$

注意这里，a' 是依据当前策略 π 采取的行动。而当我们把 $Q(s',a')$ 替换成 $\max_{a'} Q(s',a')$ 的时候，算法就成为了 Q-学习法（Q-learning）：

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{3}$$

在这里，我们发现 Sarsa 和 Q-学习的重要区别：Sarsa 是同策略学习的（on-policy），估算 Q 值时采用的策略就是当前策略，而 Q-学习是采用异策略（off-policy）学习的，估算 Q 值时采用的策略不是当前策略，而是最优策略。同策略与异策略也作为强化学习中的两种并行方法，本身并不改变算法架构，但是会影响收敛性等特性。许多算法都可以在这个维度上进行变异。

注意到在以上的算法中，我们每一步都对 Q 或者 V 进行更新。在多步骤自助法（n-step bootstrapping）中，我们则将更新进行延迟。例如，某一行动结束，n 步过后，我们才观察这一行为造成的影响，并且对它的值进行更新，其更新方式如下

$$V(S_t) \leftarrow V(S_t) + \alpha(G_{t:t+n} - V(S_t))$$

其中，$G_{t:t+n} = \sum_{i=1}^{n} \gamma^{i-1} R_{t+i} + \gamma^n V(S_{t+n})$

Sutton 等人指出，多步骤自助法强化学习在训练时的表现往往比蒙特卡罗和 TD 方法都要更优[?]。

归功于深度强化学习的提出[?][?]，近似函数的方法在近年来得到了极大的发展。近似函数有两种基本思路。第一种，是用某一类型的函数去近似状态的价值函数 $v(s)$，从而帮助智能体根据不同的 v 采取不同的策略。第二种思路是跳过 v 的计算，而直接使用函数去近似一个策略 π。在第二种思路中，$\pi(s)$ 的输出将直接是智能体应该采取的行为（或者行为的一个概率分布）。这种方法被称为 ‘‘策略优化’’，围绕策略优化问题衍生出一系列的策略梯度求解方法，成为近年来强化学习的热点研究方向之一。进一步的，在策略优化问题中，如果我们同时对价值函数 $v(s)$ 进行近似和更新，那么可以得到一类框架，我们称之为 actor-critic 架构[?]。在后续的篇目中，我们会对此进行具体介绍。

在深度强化学习中，比较著名的算法有 DDPG[?]，A2C[?]，TRPO[?]，ACKTR[?]，ACER[?]，PPO[?] 等。这些算法全部忽略了环境的模型，或者环境模型是学习得到的（TRPO）。还有一类算法，其中的环境模型是提前设计建模得到的而非习得，其中有代表性的就是 AlphaZero[?]。这类算法也是我们想探究的方向——是否通过提前建模，能够让自动驾驶车辆在环岛这类复杂环境中有更好的表现? 在接下来的章节中，我们会详细介绍其中有代表性，并且在我们的场景下易于使用的算法。

另外，在对价值函数进行优化时（梯度下降），不同的算法可能带来不同的结果。A3C 采用了 RMSProp 算法[?]，我们往往还可以选择随机梯度下降法（SGD），带动量的随机梯度下降法[?] 等方法，在算法设计时需要进行尝试和选择。Sutton 在书中对此进行了讨论[?]。此外，Ruder 在 2016 的一篇综述文中对各种梯度下降算法进行了详尽的探讨[?]。

2.1.3 无模型的深度强化学习

Mnih 等人在 2013 年首度提出深度 Q-网络（deep Q-network，简称 DQN)[?]，2015 年，他们的另一篇论文同样描述了 DQN[?]，激发了一轮深度神经网络与强化学习结合的热潮[?]。DQN 的基本思路是创造一个很一般的端到端的网络 $Q(s,a|\theta)$，其中 θ 是这个网络的参数。例如对于一个游戏，网络的输入直接就是未处理的图像，输出直接就是在某个阶段，游戏中的动作对应的价值。在传统 Q-learning 的框架下，DQN 在每一步更新 $q(s,a)$ 值的时候，不是直接使用 (3)，而是利用梯度下降法去更新网络 Q 的参数 θ，使得 Q 估值总损失最小

$$\nabla_{\theta_i} Loss(\theta_i) = E_{s,a,s'}[(r + \gamma \max_{a'} Q(s', a'|\theta_{i-1}) - Q(s,a|\theta_i)) \nabla_{\theta_i} Q(s,a|\theta_i)]$$

这里 DQN 的网络结构参考了 Hinton 等人提出的 AlexNet[?]，也就是典型的卷积神经网络（CNN）结构。DQN 的最大贡献在于，由于使用了经验回放的方法，DQN 可以收敛，从而使得深度神经网络在强化学习中的应用成为了可能。由于神经网络的高度非线性，DQN 得以模拟足够复杂的场景。DQN 在 Atari 系列游戏中取得了突破性的进展，甚至在一些游戏中超越了人类专家的水平。

不过由于我们想探究的问题更多地涉及到低维度的传感器信息输入，而不是高维度的图像直接输入，因此 DQN 一类的方法对我们的帮助并不是很大。而策略梯度方法则对我们有很大的借鉴意义，因此我们也将集中讨论这一类的方法。

前面我们提到，在策略优化问题中，有一类架构被称为 actor-critic 架构。Mnih 等人在 2016 年提出的 A3C（asynchronous advantage actor-critic）算法[?] 成为应用 actor-critic 架构的经典案例。A3C 算法同时维护一个策略函数 $\pi(a|s;\theta)$ 和一个价值函数的估计 $V(s;\theta_v)$，这两个函数均为卷积神经网络 CNN，并且两个网络除了输出层以外的层均是共享的。其中，$\pi(a|s;\theta)$ 称为 actor，而 $V(s;\theta_v)$ 称为 critic。actor-critic 会计算一个 ‘‘优势’’，定义为 $\delta = R + \gamma V(S';\theta_v) - V(S;\theta_v)$[?]。A3C 借鉴了多步骤自助法的思路（设步数为 k），在计算优势时，变异为 $\delta_t = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(S_{t+k};\theta_v) - V(S;\theta_v)$。每一步，我们依然对 θ 和 θ_v 进行更新，梯度的计算方法如下（注意计算梯度以后，还需要选择一个梯度下降方法对 θ 和 θ_v 进行更新）

$$d\theta = \delta_t \bigtriangledown_\theta log_\pi(A_t|S_t;\theta) \tag{4}$$

$$d\theta_v = \delta_t \bigtriangledown_{\theta_v} \delta_t^2 \tag{5}$$

如果我们采用多个智能体并行训练，把它们各自的梯度叠加在一起（也就是累加每个智能体各自求得的 (4)(5) 式），最后统一更新 θ 和 θ_v，就称之为异步训练方法，这种算法也因此被称为 asynchronous advantage actor-critic。如果条件受限没有异步，而只是用一个智能体进行训练，这个算法就称为 A2C（advantage actor-critic）。actor-critic 架构被认为是能够有效加速训练的，因此越来越多的算法都基于这一架构进行设计。

2015 年，Schulamn 等人提出了经典的（Trust Region Policy Optimization, TRPO）算法[?]。在文章中，他们首先从理论上证明了一种策略迭代更新的 MM（minorization-maximization）算法，可以确保策略对应的总价值（$\eta(\pi) = E_{s_0}[\sum \gamma^t r(s_t);\pi_\theta]$）单调不下降。依据这一理论基础，他们进行了多步近似，将 MM 算法中的优化（求 argmax）的问题转化为了一个更简单的加限制的优化问题（信任空间优化，trust-region optimization）。在进行进一步近似后，优化式中的期望形式可以用蒙特卡罗方法进行估算。最后，优化式本身则利用一种近似的方法进行优化。在实验中，他们采用了神经网络作为策略 π 的表达。TRPO 的更新方法是：

$$\begin{aligned} \theta &= \underset{\theta}{\text{argmax}}\, E\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} Q_{\theta_{old}}(s,a)\right] \\ &\text{subject to } E[D_{KL}(\pi_{\theta_{old}}(\cdot|s)||\pi_\theta(\cdot|s))] \le \delta \end{aligned} \tag{6}$$

其中，δ 是一个认为选定的小值。注意，在论文的附录中，作者提出采用一阶

近似对上式进行求解，优化问题就可以就变为一个典型的拉格朗日共轭问题[?]。不过，由于 TRPO 只是保证了每一步更新，策略都变得更好，却没有考虑探索性，因此，TRPO 的结果很容易困在局部最优值。在 Schulman 之后，Wu 等人于 2017 年又提出了一个改进算法 ACKTR[?]（Actor Critic using Kronecker-Factored Trust Region），采用 actor-critic 架构，并利用 Kronecker 因子去近似梯度下降。同年，Schulman 等人又提出了 PPO 算法[?]，在实现难度和算法表现方面取得了平衡。PPO 算法的基本思路有三点：

- 去除 TRPO 的优化限制条件，在 TRPO 的基础上，采用一个 clip 函数，从而阻止 θ 每一次更新太大，确保收敛
- 在优化的函数里，通过引入系数的形式，同时优化 actor 和 critic，整个优化过程只有一个优化目标函数
- 通过在优化目标中加入一个熵奖励，增加算法的探索性

PPO 把强化学习问题转化为一个形式简单的单一函数优化问题，在许多方面取得了比以往算法（A2C, ACER, TRPO, etc.）更优的表现。

在 2015 年，Silver 首度提出了针对连续动作空间的确定性算法 DPG（deterministic policy gradient)[?]。与确定性算法对应的是随机算法，也就是策略是行动的概率分布。在高维的动作空间内（例如动作是连续的），DPG 比与之对应的随机算法获得了更优的结果。DPG 采用 actor-critic 架构，并且是异策略（off-policy）训练的。注意这里，DPG 的 actor-critic 架构与 Sutton 书中提到的 actor-critic 架构有所区别。它是不以价值函数作为基准的，而正常的 actor-critic 架构应该以价值函数作为基准，计算优势，再进行求导[?]，参见前面提到的 A3C 算法[?]。同时，DPG 算法采用的是 Q-learning 结构，不是用 $V(s)$ 对价值进行估计，而是用 $Q(s, \pi(s))$ 对价值进行估计。由于算法本身是确定性的，DPG 与随机异策略 actor-critic 架构[?] 不同，不需要针对策略进行重要性采样（importance sampling，在[?] 中有详细讨论）。DPG 本身没有选择神经网络，而是可以采用一般的函数，尤其是线性函数。

DPG 算法对 $\pi(s)$ 和 $\mu(s)$ 的更新方法如下：

$$
\begin{aligned}
\delta_t &= r_t + \gamma Q(S_{t+1}, A_{t+1}; \theta_Q) - Q(S_t, A_t; \theta_Q) \\
\theta_Q &\leftarrow \theta_Q + \alpha_{\theta_Q} \delta_t \, \nabla_{\theta_Q} \, Q(s, a; \theta_Q) \\
\theta_\mu &\leftarrow \theta_\mu + \alpha_{\theta_\mu} \, \nabla_{\theta_\mu} \, \mu(s; \theta_\mu) \, \nabla_a \, Q(s, a; \theta_Q)|_{a=\mu(s;\theta_\mu)}
\end{aligned}
\tag{7}
$$

基于 DPG，Lillicrap 等人提出的改进算法，DDPG 算法[?]，则全面采用了深

度神经网络。为了能够让训练收敛，DDPG 使用了一个目标网络（target network）的概念。目标网络的定义是缓慢地跟踪依据梯度下降优化得到的网络参数 θ_Q 和 θ_μ，θ_Q 是状态行动对的价值估计网络 Q 的参数，而 θ_μ 是 actor-critic 架构中 actor 采用的策略的参数。目标网络参数 θ'_Q 和 θ'_μ 的更新方法如下（在每一步迭代过后进行这个更新）：

$$\theta' = \tau\theta + (1-\tau)\theta', \tau \ll 1$$

这样，相当于强迫目标网络缓慢地变化，从而增强了训练的收敛性。目标网络主要用于带入 (7) 中 δ_t 的计算。同时，DDPG 采用了经验回放的方法，在每一步迭代中都采用之前历史中一批随机挑选的过往经验进行批量计算，而非一步的结果，这样可以确保网络训练时不带有时序偏差。另外，DDPG 没有像 DPG 一样采用异策略学习，而是在策略基础上加上一个噪声分布以确保 ‘‘探索’’ 性。

2.1.4 有模型的深度强化学习

有模型的深度强化学习问题可以分为模型是 ‘‘习得’’ 的和模型是给定的两种。在这里，我们首先关注一下给定模型的强化学习的应用。

在模型给定的算法中，最著名的例子就是 2016 年提出的 AlphaGo[?]。Deepmind 在 2017 年进一步提出了 AlphaZero[?]，其算法和 AlphaGo 相比有几大特点：

1. AlphaZero 完全采用逐步迭代的方式，而 AlphaGo 在训练中会挑选之前出现过的最好策略
2. AlphaZero 拥有的先验知识只有棋盘的格局和下棋的规则，并且这些知识只用来进行模拟
3. AlphaZero 对参数的变动更加鲁棒，不同棋类游戏用的都是同一套参数
4. AlphaZero 在训练中没有对输入进行任何数据加强的操作，而 AlphaGo 则对棋盘进行了镜像操作

AlphaZero 采用蒙特卡罗树状搜索（Monte Carlo Tree Search, MCTS[?]）的方法，每次优先访问训练历史中访问次数低，但访问概率 $\pi(a|s)$ 应该较高，且 $Q(s,a)$ 的状态行为对 (s,a)。同时，在此基础上加上一个噪声（在 AlphaZero 的例子中，一个狄利赫里分布的噪声），确保探索性。AlphaZero 采用深度卷积神经网络作为策略 $\pi(s)$ 的拟合函数，其输出为 $\pi(s) = (\mathbf{p}, v)$，其中，$\mathbf{p}$ 为在某个状态下选择各个行为的概率，v 为根据 s 推测的状态估值,$v = \mathbf{E}[z]$（胜负平局，z

的值分别为 $+1, -1, 0$）。针对该网络定义的损失函数为

$$L = (z - v)^2 - \pi^T log(p) + \lambda||\theta||^2 \tag{8}$$

可以看出，这个损失函数试图使 v 的预测和棋局结果 z 更加接近；π 代表了训练中各个动作的搜索概率，损失函数试图让神经网络的输出 (p) 与之尽可能接近。

这里所谓“给定模型”，其实就是环境是已知的，不需要真的和环境交互，而是在仿真里直接进行下棋这个活动即可。然而，这也意味着这个问题本身与直接和未知环境交互并没有区别。尽管模型给定，算法并没有从模型本身得出什么规律，因此这个模型本质上就是一个环境。

在强化学习问题中，从与环境的交互历史中逐渐学习到模型，则是另一个有趣的思路。在[?] 中，Dyna 架构就是这样一种思路。在强化学习中，利用模型去近似求解 $v(s)$ 是一个 ‘‘规划’’（planning）问题。传统的列表式方法可以根据训练中得到的环境反馈不断更新模型。2015 年，Oh 等人在[?] 一文中提出了一种利用神经网络学习高维度环境的方法，使得复杂环境的模拟也成为了可能。

基于[?] 的结果，Racanière 等人在论文[?] 中提出了一种利用模型去增强无模型强化学习的方案，简称为 I2C 算法。这种方法本质上是细化了的 A3C 算法。注意到 A3C 本身是一个无模型的方法，它仅仅依赖于 $\pi(a|s)$ 和 $v(s)$ 的估计（回顾一下 (4)(5) 两个更新式）。但是，I2C 算法修改了 θ 的组织形式，构建了一个较为复杂的网络结构，因而引入了环境模型。其基本架构如下：

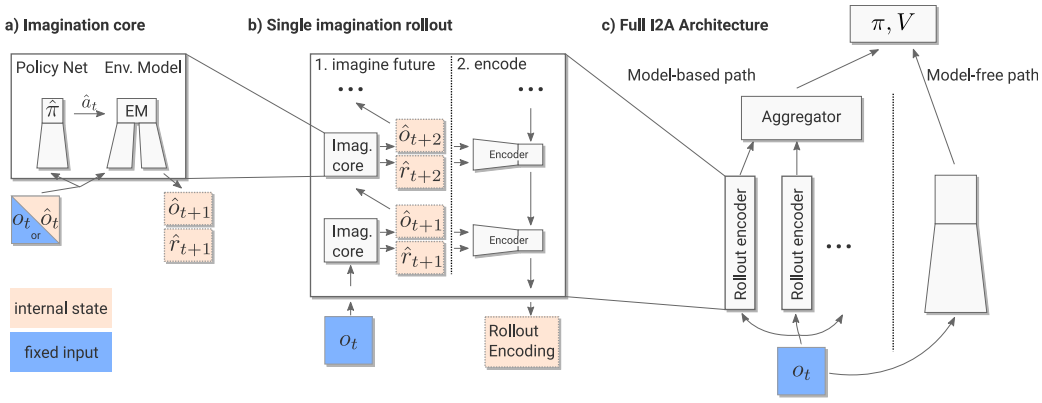

图 1　I2A 的基本结构：a) 一个预先训练好的神经网络，可以预测在策略 $\hat{\pi}$ 下，状态 s 的一系列时序变化；b) 一个子网络，用于 ‘‘理解”a) 的输出结果；c) 整体架构

图1c) 展示了网络整体架构，可以看到整体网络的输入为 o_t，也就是对状态

S_t 的观测，输出为策略 π 和价值函数估值 V（用于 actor-critic 架构）。因此，这个结构满足 A3C 的基本框架。在框架内部，网络结构采用了含有预测模型网络处理的输入，也采用了无模型网络处理的输入，分别作为两个分支，最后通过一个子网络汇合。具体的结构参数可以参见论文[?] 的附录。通过引入这个特殊的网络结构，I2A 算法将习得的模型结合进了原有的无模型学习框架中。同时，实验结果表明，这里的模型可以有错误和不精确的地方，学习结果依然是鲁棒的。

2.2 部分可观测的马尔可夫决策过程

我们要解决的问题是典型的部分可观测马尔科夫决策过程。当在马尔可夫过程中，状态 s 不能被完全观测时（例如汽车只能看到自己视野范围内的几辆车），我们称这个问题为部分可观测马尔科夫决策过程（partially observable Markov decision process, POMDP）。POMDP 在强化学习社区中被当做一种特殊的问题看待，也围绕这类问题衍生出许多解决方案。看论文 DRQ-POMDP

2.3 强化学习在自动驾驶领域的应用

看论文 Survey 的 Autonomous Vehicles 的那一节，看看有没有可抄的。

3 课题内容及开展情况

3.1 第一阶段

3.2 第二阶段

4 进度计划