UC Berkeley CS285 第 4 讲: 强化学习介绍

翻译/总结: 钱秋辰

版本：1

日期：2022 年 5 月 2 日

1 定义

术语与符号如图1所示:

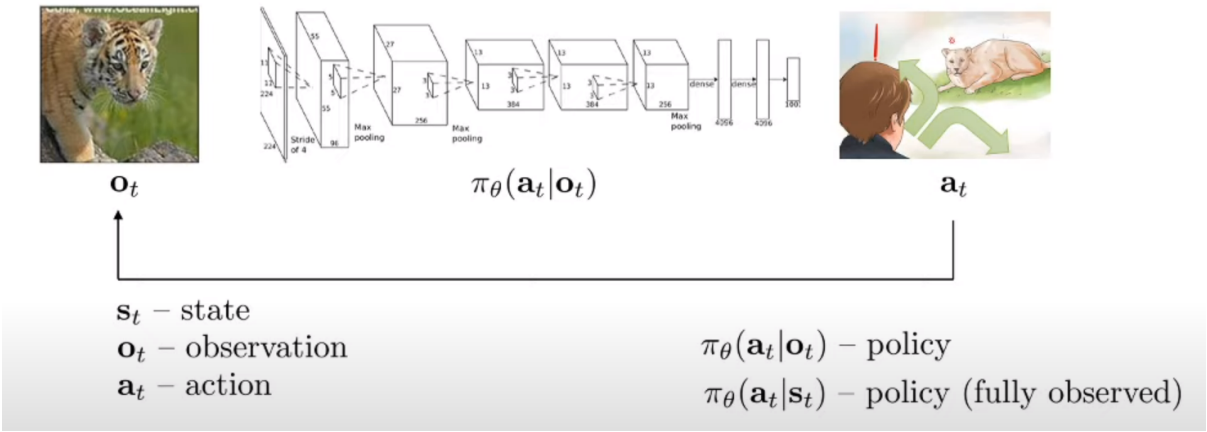

图 1: 术语与符号定义

其中 θ 表示 policy 的参数. 给定时刻 t 的观察 o_t, 根据策略 π_θ 进行采样, 可以得到动作 a_t. 而不同的状态之间可以相互转移, 如图2所示:

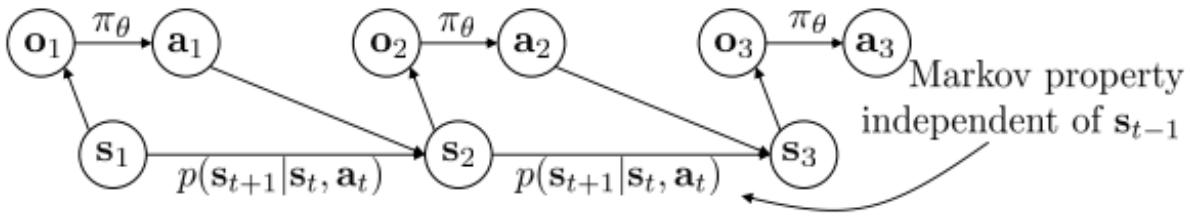

图 2: 状态间的转移

这里需要额外介绍一下 **马尔可夫性质 (Markov Property)**, 指的是 s_{t+1} 仅与 s_t 有关, 与 s_{t-1} 无关. 这一性质可以用来区分状态与观察, 状态需要满足马尔可夫性质, 而观察不用. 因为观察是状态的随机函数, 其可能包含 (也可能不包含) 所有用来推断完全状态的

信息. 因此, 我们可以划分完全观察的强化学习算法为状态可知; 而部分可观察的强化学习算法为仅观察可知.

2 奖励函数引入

在没有更多的专家数据的情况下, 引入了奖励函数以判断动作的好坏, 因为我们追求的是未来的高奖励而非一时的高奖励. 通过在高速公路上开汽车的例子, 我们就可以很好的理解这一点. 如果我们在短时间内加速, 可以得到这段时间的最高奖励, 但这同时也会提高发生车祸的可能性, 所以整体获得的奖励不一定是最优的. 而对于强化学习来说, 核心就在于如何选择当前的动作以在未来获得高的奖励.

此外, 根据状态 s, 动作 a, 奖励 $r(s,a)$ 和转移概率 $p(s'|s,a)$, 我们可以定义 **马尔可夫决策过程 (Markov Decision Process，MDP)**. 但在介绍马尔可夫决策过程之前, 我们需要先了解 **马尔可夫链 (Markov Chain)**.

3 马尔可夫链

Andrey Markov 创建了 **马尔可夫链**的理论基础:

$$\mathcal{M} = \{\mathcal{S}, \mathcal{T}\}$$

其中 $\mathcal{S}$ 为状态空间, $\mathcal{T}$ 为转移算子. 如图3所示:

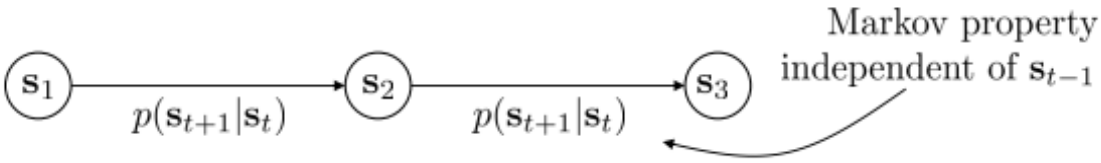

图 3: 马尔可夫链

当然, 马尔可夫链满足马尔可夫性质, 也就是在给定 s_t 时, 我们有 $p(s_{t+1}|s_t)$ 的概率转移到 s_{t+1}, 这一概率与 s_{t-1} 无关. 在离散状态系统中, 我们可以定义 $\mu_{t,i} = p(s_t = i)$, 则 $\vec{\mu}_t$ 为一个概率向量, 例如状态的可能为 $\{1,2,3,4\}$, 那么 $\vec{\mu}_t = \{p(s_t=1), p(s_t=2), p(s_t=3), p(s_t=4)\}$. 此时若我们定义从状态 j 转移到状态 i 的条件概率为:

$$\mathcal{T}_{i,j} = p(s_{t+1} = i | s_t = j)$$

则我们可以得到:

$$\vec{\mu}_{t+1} = \mathcal{T}\vec{\mu}_t$$

这也是为什么 $\mathcal{T}$ 被称为算子的原因.

4 马尔可夫决策过程

Richard Bellman 等人提出了 **马尔可夫决策过程**, 马尔可夫决策过程是将马尔可夫链放在一个决策环境的拓展产物:

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$$

其中 $\mathcal{A}$ 为动作空间. 如图4所示:

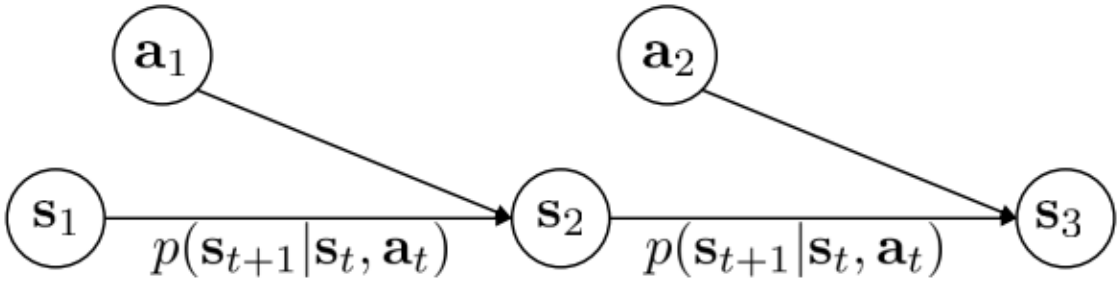

图 4: 马尔可夫决策过程

需要注意的是, 因为添加了一个动作因素, 现在 $\mathcal{T}$ 是一个张量了. 令:

$$\mu_{t,j} = p(s_t = j) \tag{1}$$

$$\xi_{t,i} = p(a_t = i) \tag{2}$$

$$\mathcal{T}_{i,j,k} = p(s_{t+1} = i | s_t = j, a_t = k) \tag{3}$$

那么我们有

$$\mu_{t+1,i} = \sum_{j,k} \mathcal{T}_{i,j,k}\, \mu_{t,j}\, \xi_{t,k}$$

同样地, 下一时刻的状态仅与当前时刻的状态和动作有关.

5 部分可观察马尔可夫决策过程

实际中, 我们往往不能直接获取状态, 而是通过观察值来选择动作, 因此更广义的 MDP 也被称为 **部分可观察马尔可夫决策过程 (Partially Observable Markov Decision Process, POMDP)**:

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \epsilon, r\}$$

其中 $\mathcal{O}$ 为观察空间; ϵ 为排放概率 (emission probability), 决定了给定 s_t 时, o_t 的概率分布 $p(o_t|s_t)$. 以自动驾驶为例, $\mathcal{S}$ 表示汽车的位置和运动信息, $\mathcal{A}$ 表示踩油门和刹车, 而 $\mathcal{O}$ 表示摄像头采集到的图像. POMDP 如图5所示:

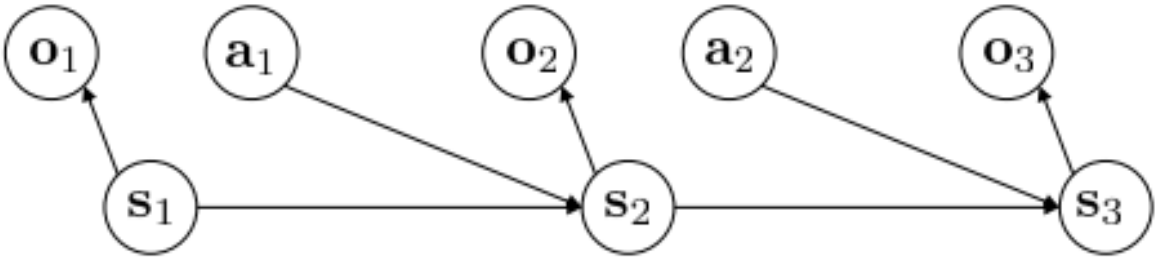

图 5: 部分可观察马尔可夫决策过程

6 强化学习的目标

一般地, 强化学习的过程如图6

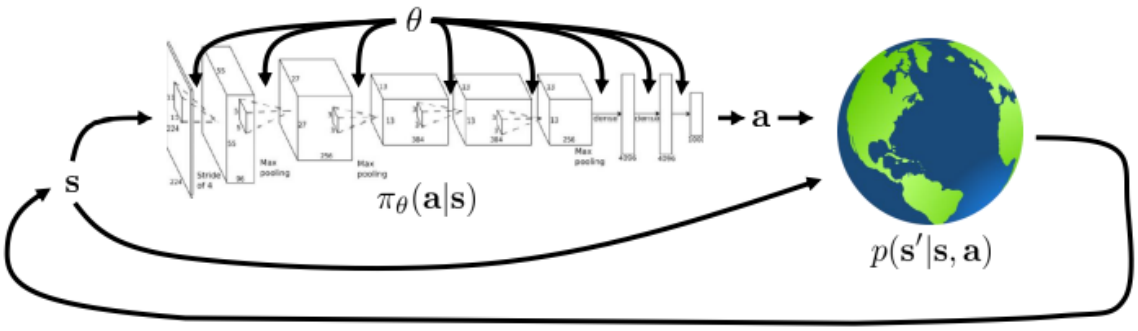

图 6: 强化学习过程

根据状态 s 和策略 $\pi_\theta(a|s)$ 可以决定动作 a. 而状态 s, 动作 a 和转移概率 $p(s'|s,a)$ 来确定下一个状态 s', 如此反复. 这里我们定义一个有限长度的 **动作轨迹** (或序列) 为:

$$\tau = \{s_1, a_1, ..., s_T, a_T\}$$

对于这样的轨迹, 我们易知其发生的概率为:

$$p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

因此在这些条件下, 我们可以写出 **强化学习的目标函数**:

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right]$$

其中 $\sum_t r(s_t, a_t)$ 表示在给定策略 π_θ 下, 某条轨迹 τ 的所有动作奖励之和. 那么, 强化学习的目标为找到能够定义策略 π 的参数 θ, 使得该策略下轨迹的所有动作奖励之和的期望值最大. 换句话说, 我们想要找到一个策略, 其产生的轨迹拥有期望中可能的最高奖励. 当然, 这里的 期望考虑了策略, 转移概率和初始状态分布的随机性.

时序动作轨迹 $\tau = \{s_1, a_1, ..., s_T, a_T\}$ 也可以看成一个在空间 (s, a) 的马尔可夫链, 如下图7所示:

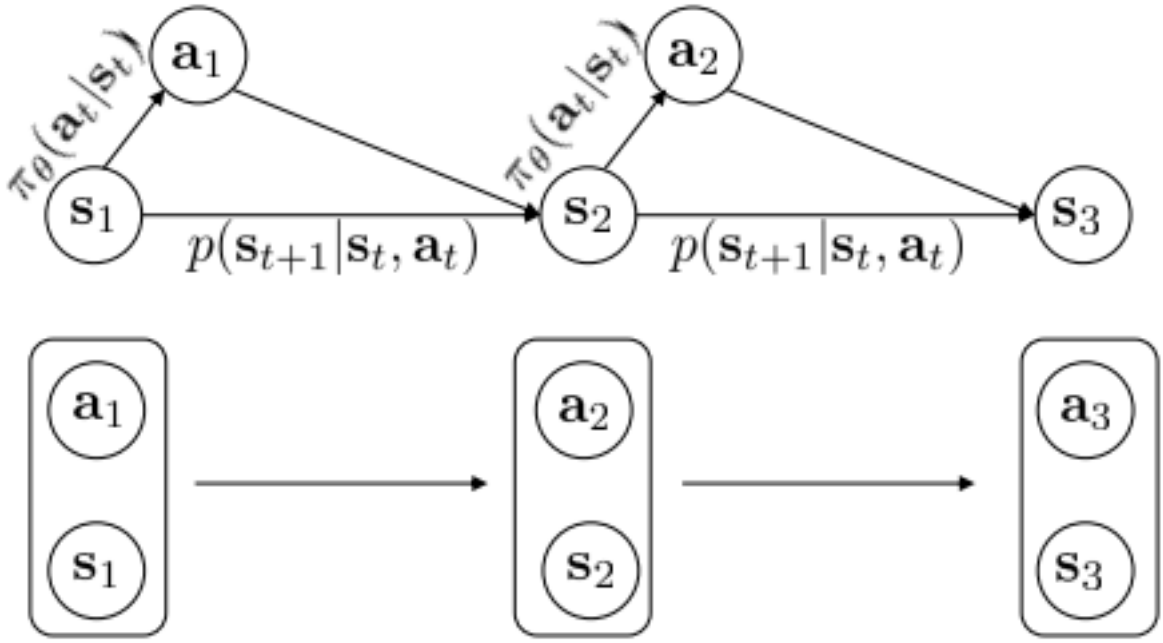

图 7: 转换轨迹为马尔可夫链

其中, 转移概率可以表示为:

$$p((s_{t+1}, a_{t+1})|(s_t, a_t)) = p(s_{t+1}|s_t, a_t)\pi_\theta(a_{t+1}|s_{t+1})$$

6.1 有限长度轨迹

我们可以重写目标函数为:

$$\theta^\star = \arg\max_\theta E_{(s_t,a_t)\sim p_\theta(s_t,a_t)}\Big[r(s_t, a_t)\Big]$$

其中 $p_\theta(s_t, a_t)$ 为 **状态-动作边际 (state-action marginal)**, 则我们只需要关注马尔可夫链在时间 T 上的边际分布即可. 但如果 $T = \infty$, 是什么情况呢?

6.2 无限长度轨迹

如果我们简单地根据 T 来叠加, 很明显是不能实现的. 但是可以通过对目标函数进行平均, 因为 T 是一个常数, 也就是:

$$\theta^\star = \arg\max_\theta \frac{1}{T}\sum_{t=1}^{T} E_{(s_t,a_t)\sim p_\theta(s_t,a_t)}\Big[r(s_t, a_t)\Big] \to E_{(s,a)\sim p_\theta(s,a)}\Big[r(s, a)\Big]$$

由于我们之前已经把 (s, a) 马尔可夫链化, 因此有:

$$\begin{bmatrix} s_{t+1} \\ a_{t+1} \end{bmatrix} = \mathcal{T}\begin{bmatrix} s_t \\ a_t \end{bmatrix}$$

其中 $\mathcal{T}$ 为状态-动作转移算子. 则我们可以推导出第 k 步时:

$$\begin{bmatrix} s_{t+k} \\ a_{t+k} \end{bmatrix} = \mathcal{T}^k \begin{bmatrix} s_t \\ a_t \end{bmatrix}$$

但这也会引起一个问题: 当 $k \to \infty$ 时, $p(s_t, a_t)$ 能收敛到一个 **平稳分布 (stationary distribution)** 吗? 令 $\mu = p_\theta(s, a)$, 则平稳分布指的是:

$$\mu = \mathcal{T}\mu$$

也就是转移前后分布不变. 变换一下等式, 我们可以得到:

$$(\mathcal{T} - I)\mu = 0$$

不难发现, μ 是 $\mathcal{T}$ 的特征向量, 而特征值为 1. 后面通过一些假设 (遍历性 Ergodicity 和马尔可夫链为非周期的 aperiodic), 我们可以证明平稳分布存在. 其中, 遍历性指的是每个状态都可以从非零概率的其他状态到达. 这一假设很有用, 因为它防止了从 MDP 的一部分开始, 可能永远到达不了另一部分的情况. 如此, 则我们从哪部分开始就变得非常重要, 从而平稳分布也就不存在了.

需要注意的是, 本节开始处的平均奖励策略也是满足平稳分布的.

7 期望和随机系统

在强化学习中, 我们几乎总是在关心期望 (expectations), 而不是个别奖励值, 这给予了我们很好的数学性质. 比如图8中, 当一辆汽车行驶在盘山公路上:

图 8: 行驶在盘山公路上的汽车

若保持在公路上的奖励为 $+1$, 冲下悬崖的奖励为 -1, 那么我们有了一个二元离散奖励函数 $r(x)$. 若此时我们的策略:

$$\pi_\theta(a = \text{fall}) = p'$$

其中 p' 为某个概率, 则平稳分布下的收益期望 $E_{\pi_\theta}[r(x)]$ 仍然是连续的, 光滑的. 因为我们有 p' 的概率跌落 (奖励为 -1) 和 $1 - p'$ 的概率保持行驶, 则最终, 奖励函数的期望值为 $(+1) \cdot (1 - p') + (-1) \cdot p'$, 是可微分的.

8 强化学习结构

通常强化学习的算法分为三个步骤进行循环迭代, 如图9所示:

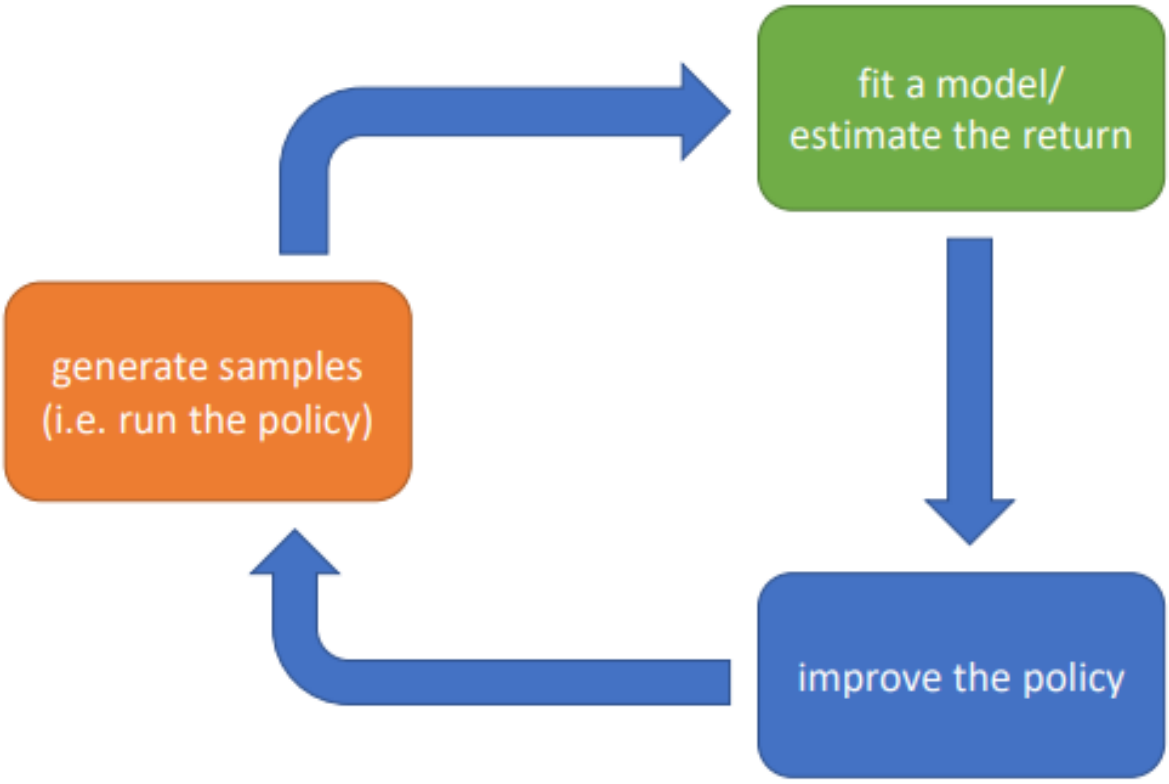

图 9: 强化学习框架

首先是运行策略, 获得数据; 其次是评估奖励; 最后是改进策略; 如此往复.

8.1 简单的例子 1

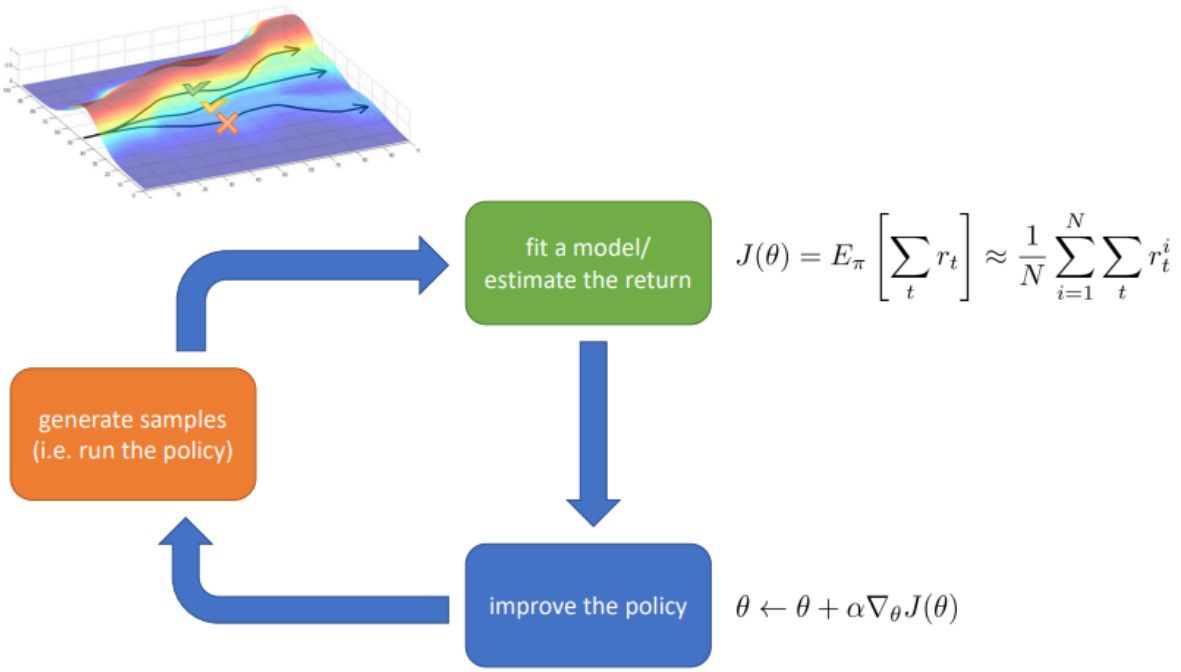

图 10: 一个简单的例子

首先橙色方块中, 我们生成三条轨迹, 接着绿色方块中构造 $J(\theta)$ 来评估模型, 然后蓝色方块中使用梯度下降的方法来改进策略.

8.2 稍复杂的例子 2 - 通过后向传播

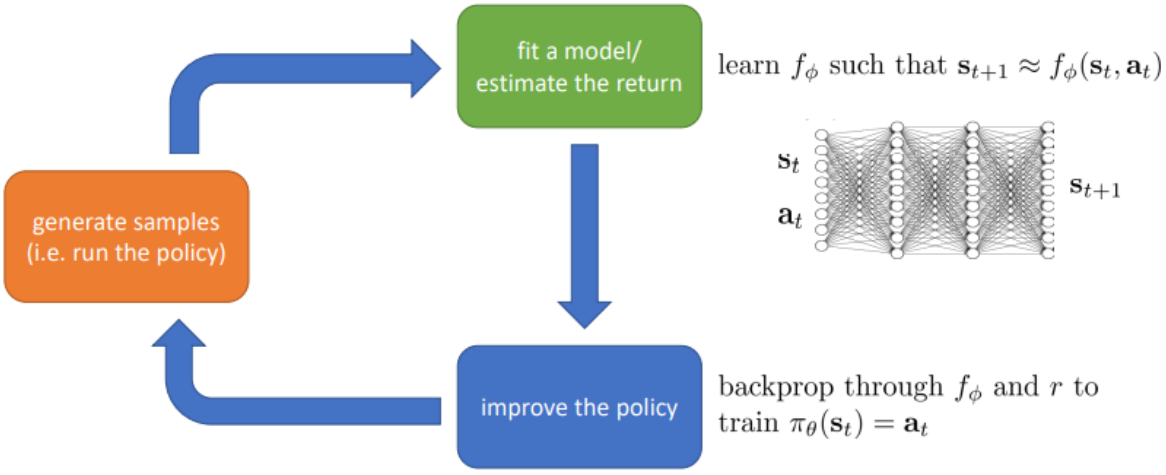

图 11: 后向传播例子

绿色方块中也可以通过构造函数 f_ϕ 来近似下一个状态, 通常这种方法被称作基于模型的 (model-based). 在蓝色方块中, 我们改进的是 f_ϕ, 使其更精确, 然后通过 f_ϕ 的后向传播

来评估并改进策略.

8.3 哪部分计算代价比较高?

- 对于橙色方块的步骤来说, 如果是真实世界的机器人/汽车/电网等等, 那么计算时间会比较长, 因为我们需要基于真实时间进行采样 (但在仿真中, 比如 MuJoCo, 可以达到最快于 10000 倍的真实时间的模拟时间)
- 对于绿色方块来说, 如果我们构造 $J(\theta)$, 那么计算代价会比较小, 毕竟只是奖励总和的平均, 但学习一个函数 f_ϕ 的代价会比较高
- 对于蓝色方块来说, 使用梯度方法的成本较低, 而使用基于模型的方法成本较高

9 值函数 Value function 与 Q 函数

现在让我们考虑如何处理这些期望值. 考虑我们目标函数的期望项:

$$E_{\tau\sim p_\theta(\tau)}[r(s_t,a_t)]$$

展开可得一长长长长串:

$$E_{s_1\sim p(s_1)}\left[E_{a_1\sim\pi(a_1|s_1)}\left[r(s_1,a_1)+E_{s_2\sim p(s_2|s_1,a_1)}\left[E_{a_2\sim\pi(a_2|s_2)}\left[r(s_2,a_2)+\dots\,|s_2\right]|s_1,a_1\right]\right]|s_1\right]$$

这里我们令:

$$Q(s_1,a_1)=r(s_1,a_1)+E_{s_2\sim p(s_2|s_1,a_1)}\left[E_{a_2\sim\pi(a_2|s_2)}\left[r(s_2,a_2)+\dots\,|s_2\right]|s_1,a_1\right]$$

可得:

$$E_{\tau\sim p_\theta(\tau)}[r(s_t,a_t)]=E_{s_1\sim p(s_1)}\left[E_{a_1\sim\pi(a_1|s_1)}\left[Q(s_1,a_1)\,|s_1\right]\right]$$

这样如果我们已知 $Q(s_1,a_1)$, 则优化第一步的策略会变得非常容易, 因为我们只需要选择会使得期望值最大的 $\pi_\theta(a_1|s_1)$ 即可. 举个例子, 若 $a_1=\arg\max_{a_1}Q(s_1,a_1)$, 那么 $\pi(a_1|s_1)=1$. 因此, 我们可以得到 **Q 函数**的定义 - 在状态 s_t 下采取动作 a_t 后, 根据给定策略 π_θ, 未来总奖励值的期望:

$$Q^\pi(s_t,a_t)=\sum_{t'=t}^{T}E_{\pi_\theta}[r(s_{t'},a_{t'})\,|s_t,a_t]$$

同时, 我们也可以定义 **值函数** - 在状态 s_t 后, 根据给定策略 π_θ, 未来总奖励值的期望:

$$V^\pi(s_t)=\sum_{t'=t}^{T}E_{\pi_\theta}[r(s_{t'},a_{t'})\,|s_t]$$

因为 Q 函数同时考虑状态和动作, 而值函数只考虑状态, 两者间也存在这样的关系:

$$V^\pi(s_t)=E_{a_t\sim\pi(a_t|s_t)}[Q^\pi(s_t,a_t)]$$

不难发现, 我们的目标函数被简化成了 $E_{s_1\sim p(s_1)}[V^\pi(s_1)]$.

10 利用值函数和 Q 函数来求解最优策略

10.1 想法一

如果我们已知策略 π 和 $Q^\pi(s,a)$, 那么我们可以这么提升 π: 若满足 $a = \arg\max_a Q(s,a)$, 则我们令 $\pi(a|s) = 1$, 这一新策略至少和原先的 π 一样好 (甚至可能更好, 因为新新策略会最大化未来的奖励). 而且我们甚至可以对所有 t 执行这一操作, 无论原先的 π 是什么, 我们总是可以这样提升.

10.2 想法二

我们可以用来提高 '好的' 动作发生的概率. 我们可以发现 $V^\pi(s) = E_{a\sim\pi(a|s)}[Q^\pi(s,a)]$ 代表了在策略 $\pi(a|s)$ 下, 进行动作的平均表现, 那么如果 $Q^\pi(s,a) > V^\pi(s)$, 就能说明动作 a 是高于平均表现的动作. 因此, 我们可以通过改动 $\pi(a|s)$ 来提高这个 a 发生的概率. 这也是策略梯度方法的主要更新规则.

11 强化学习算法类型

在目标为最大化期望 $\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(s_t,a_t)\right]$ 的背景下, 强化学习通常有以下方法:

- 策略梯度方法 (Policy Gradient): 直接计算目标函数关于参数的梯度, 由于轨迹数量很多, 所以一般通过采样的形式来计算梯度. 得到梯度之后, 执行如梯度下降之类的过程. 代表有 REINFORCE; 自然策略梯度 Natural Policy Gradient; 置信域策略优化 Trust Region Policy Optimization 等
- 基于值函数的方法 (Value-based): 尝试估计最优策略的值函数/Q 函数, 然后使用它们来提升策略. 其中, 值函数/Q 函数通常是个神经网络. 代表有 Q-Learning, DQN; 时间差分学习 Temporal-Difference Learning; 拟合值迭代 Fitted Value Iteration 等
- 演员-评论家方法 (Actor-Critic): 是介于策略梯度法和值函数方法中间的一种方法, 其估计当前策略的值函数/Q 函数, 然后使用估计的函数来提升策略, 代表有异步优势演员-评论家 (Asynchronous Advantage Actor-Critic), soft Actor-Critic 等
- 基于模型的方法 (Model-based): 估计转移模型 (具体来说可能是建立描述物理现象或其他系统动态的模型), 用于动作的规划, 改进策略等. 代表有 Dyna; 引导策略搜索 Guided Policy Search 等. 与上述三种方法都不同, 在蓝色方块改进策略的步骤, 该方法可以有多个选项:
  1. 抛开策略, 仅使用模型来规划动作. 对于连续问题, 可以使用轨迹优化/最优控制, 本质上是通过后向传播在动作上进行优化, 比如学习机器人连续环境的物理性质, 然后通过习得的物理模型来进行最优控制或轨迹优化过程; 对于离散

问题, 可以使用如蒙特卡洛树搜索 (Monte Carlo Tree Search) 的方法来对离散的动作空间进行规划, 比如下围棋

2. 将梯度后向传播回策略之中, 但往往需要一些技巧才能很好奏效, 通常是为了考虑数值稳定性, 比如在后向传播策略中, 2 阶方法通常比 1 阶好得多
3. 使用模型来学习值函数, 在离散环境下可以使用动态规划; 又或者是生成仿真经验给无模型学习器 (如 Dyna)

不同强化学习算法的类型如图13所示:

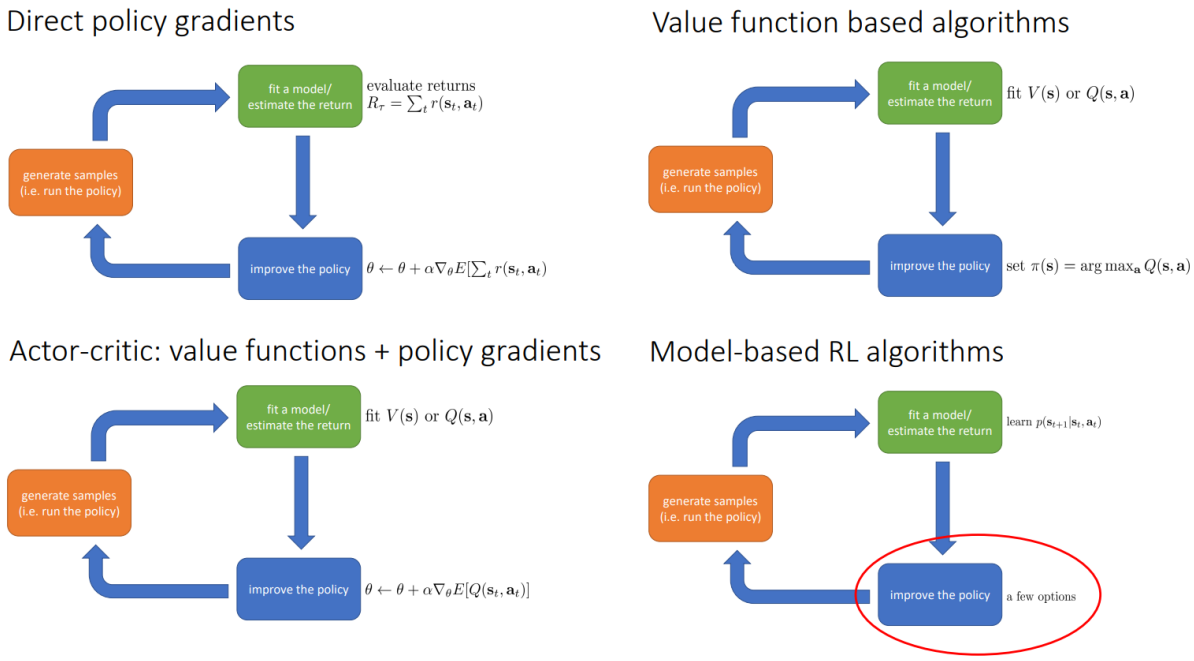

图 12: 不同强化学习算法的类型

12 怎么选择算法

强化学习中, 每个算法都有自身的特色与适用范围. 因此在如何选择上, 我们需要权衡多种因素:

- 样本效率, 指的是需要多少数据才能得到一个较好的策略
- 稳定性与易用性, 主要体现在算法收敛性上
- 随机/确定, 有些算法假设系统是动态的, 策略是随机的, 而有些算法假设这些是确定的
- 离散/连续
- 有限域/无限域, T
- 表征的难易, 有些问题中, 表征一个策略是比较容易的, 而有些问题中表征一个模型是容易的

12.1 样本效率

在样本效率中, 我们最关注的问题是这个算法是 off-policy 还是 on-policy. off-policy 指的是我们无需从策略生成新的样本即可改进策略, 换句话说, 我们可以使用历史数据来改进策略. 而 on-policy 的定义与之相反, 即每次策略发生改变, 即使只改变了一点点, 也需要重新生成新的样本.

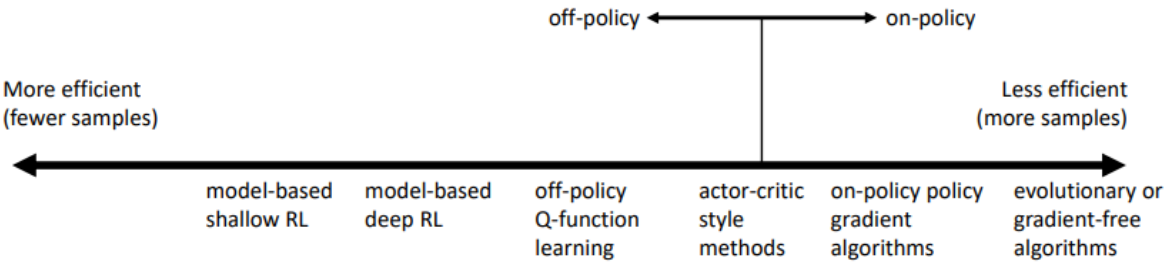

图 13: On-policy 和 off-policy 算法

从样本效率的角度出发, 右边的算法是低效的, 左边的算法是高效的. 但是样本效率低并不代表这个算法就不好了, 因为我们有其他的指标要去衡量. 比如下围棋, 采样的计算代价很低, 其大部分的计算时间都集中在更新值函数, 模型和策略上了, 那么在这个例子中我们并不怎么关心样本效率.

12.2 稳定性与易用性

在稳定性与易用性中, 我们需要考虑:

- 算法会收敛吗? 或者说当我们运行足够长的时间, 可以保证其最终收敛至一个固定解吗?
- 如果收敛, 收敛到哪里 (比如目标函数的局部最优值)?
- 每次运行都会收敛吗?

在监督学习中训练巨大的模型, 由于采用了梯度下降法, 所以往往都是收敛的. 但在强化学习中, 我们很多时候并不使用梯度下降法, 也就不能保证其收敛性. 很多强化学习算法实际上是固定点迭代算法, 其只在很简化的离散状态假设下才能保证收敛, 这在实际中往往是不可行的. 相比之下:

- Q-Learning 是固定点迭代, 仅在某些条件下收敛
- 值函数拟合也是固定点迭代, 并且在最好情况下, 它在最小化拟合误差 (Bellman error), 即是否值函数在准确地预测值, 但这与能找到高奖励的策略并不等价. 最坏情况下, 值函数拟合甚至没有在最小化拟合误差, 最终会导致发散的结果
- 基于模型的方法并不是优化目标函数, 而是为了得到更精确的模型而优化, 模型训练本身是收敛的, 但是得到更好的模型并不等价于得到更好的策略

- 策略梯度法会使用梯度下降, 因此是收敛的, 但同时也是最低效的

12.3 前提假设

- 常规假设 1: 完全观察, 也就是我们可以获取状态而不仅仅是观察值, 或者说我们的观察值满足马尔科夫特性
  - 一般被值函数拟合法作为假设前提
  - 可以通过添加循环或已意来缓解, 但一般难度较高
- 情景学习 (episodic learning), 比如一个机器人进行一次尝试, 然后重置
  - 经常被纯策略梯度法和一些基于模型的方法作为假设前提
- 连续性和光滑性
  - 被一些连续值函数方法作为假设前提
  - 经常被一些从最优控制推导而来的基于模型的方法作为假设前提

13 参考

原课程链接: https://rail.eecs.berkeley.edu/deeprlcourse/
文字版本翻译: https://zhuanlan.zhihu.com/p/44575779