

UC Berkeley CS285 第 5 讲: 策略梯度法

翻译/总结: 钱秋辰

版本: 1

日期: 2022 年 5 月 2 日

1 强化学习的目标

上一讲我们了解到强化学习的过程如图1所示:

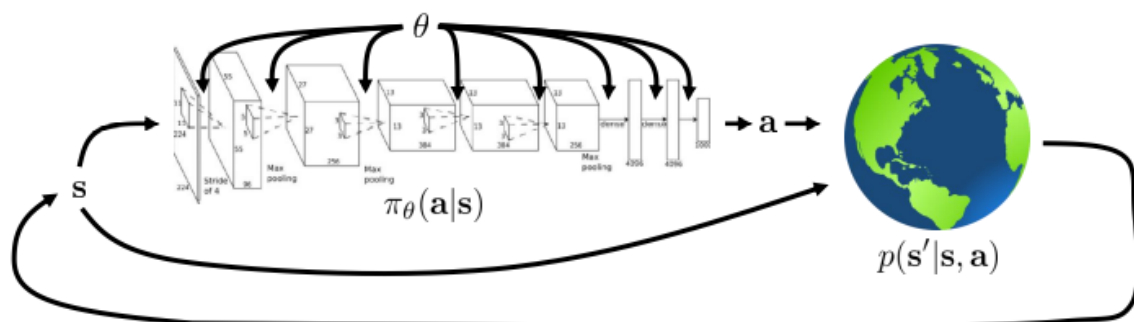


图 1: 强化学习的过程

我们可知轨迹分布, 也就是一个状态和动作序列上的概率分布:

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

因此, 强化学习的目标函数为:

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

接下来, 我们将着重考虑有限轨迹长度的情况, 也就是时间 T 内:

$$\theta^* = \arg \max_{\theta} \sum_{t=1}^T E_{(s_t, a_t) \sim p_{\theta}(s_t, a_t)} \left[\sum_t r(s_t, a_t) \right]$$

我们的目的是 **最大化轨迹分布的整体奖励函数的期望**, 可以通过图2来理解:

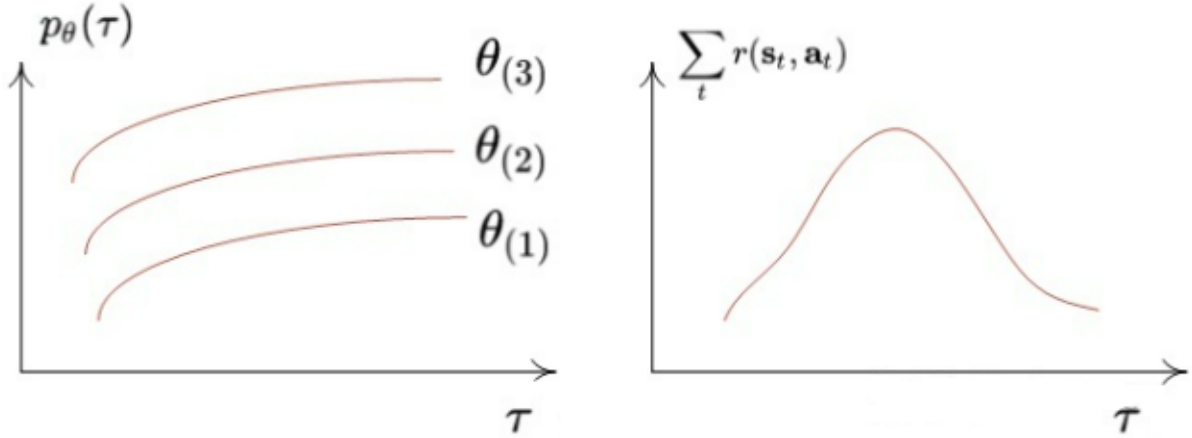


图 2: 对目标函数最大化的理解

左图展示当参数 θ 确定之后, 每条轨迹都对应着概率分布 $p_\theta(\tau)$. 而右图展示了不同轨迹对应着不同的奖励函数之和.

2 评估目标函数

那么我们如何评估目标函数呢? 这就需要我们近似出目标函数的值. 这里, 我们令:

$$J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t})$$

需要注意的是, 这里对 i 进行求和, 指的是对来自于策略 π_θ 的样本进行求和. 相应地, t 也就是表示第 i 次采样的时间步. 为了方便推导, 我们再次令:

$$r(\tau) = \sum_{t=1}^T r(s_t, a_t)$$

则 $J(\theta)$ 可以写成:

$$J(\theta) = E_{\tau \sim p_\theta(\tau)} [r(\tau)] = \int p_\theta(\tau) r(\tau) d\tau$$

这里需要用到对 \log 求导的一个小技巧:

$$p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) = p_\theta(\tau) \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} = \nabla_\theta p_\theta(\tau)$$

代入 $\nabla_\theta J(\theta)$ 中, 可得:

$$\begin{aligned} \nabla_\theta J(\theta) &= \int \nabla_\theta p_\theta(\tau) r(\tau) d\tau \\ &= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) r(\tau) d\tau \\ &= E_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log p_\theta(\tau) r(\tau) \right] \end{aligned}$$

那么这又带来一个问题, 我们怎么处理括号内的 \log 项呢? 回忆:

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

我们可以对左右两边同时添加 \log , 可得:

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

再次代入 $\nabla_\theta J(\theta)$ 中, 可得:

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \left(\cancel{\log p(s_1)} + \sum_{t=1}^T \log \pi_\theta(a_t|s_t) + \cancel{\log p(s_{t+1}|s_t, a_t)} \right) r(\tau) \right]$$

因为与 θ 无关, 在求导时我们可以消去两项, 最后可以得到:

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left(\sum_{t=1}^T r(s_t, a_t) \right) \right]$$

这个形态的 $\nabla_\theta J(\theta)$ 的最大优点是只与策略和奖励有关, 不需要知道转移概率 $p(s'|s, a)$, 从而我们可以发现 **策略梯度算法是无模型 (model-free) 算法**, 也就是不需要知道环境是怎么样子的。

3 REINFORCE 算法

回忆有限域的 $J(\theta)$:

$$J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t})$$

这里我们同样可以对 $\nabla_\theta J(\theta)$ 这么操作:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left[\left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right) \right]$$

得到梯度之后, 我们就可以去更新参数了:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

这也就是 REINFORCE 算法的主要步骤:

1. 从策略 $\pi_\theta(a|s)$ 中采样 $\{\tau^i\}$ (例如在真实世界中运行策略)
2. 估计梯度 $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left[\left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right) \right]$
3. 使用梯度更新参数 $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

4 理解策略梯度

到目前为止, 我们已经得到了 $\nabla_\theta J(\theta)$, 那么 $\log \pi_\theta(a_t|s_t)$ 究竟是什么? 想象自动驾驶的场景, 如果我们的策略是一个从驾驶图像到离散动作的映射, 那么 $\log \pi_\theta$ 则代表着该策略分配到 ‘向左’ 或 ‘向右’ 动作之一的 \log 概率. 那么 $\nabla \log \pi_\theta$ 也就是这个 \log 概率的梯度. 如此, 我们可以将 策略梯度与模仿学习中的 最大似然 (Maximum Likelihood) 估计进行比较. 在模仿学习中, 我们需要收集一些专家做选择的数据, 然后对这些数据应用监督学习, 这

将产生一个策略 $\pi_\theta(a_t|s_t)$. 所以最大似然估计的梯度为:

$$\nabla_\theta J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right)$$

而策略梯度方法的梯度为:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left[\left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right) \right]$$

不难发现, 两者的差别仅在于奖励函数项 $r(\tau)$. 如此, 我们可以得出结论, 策略梯度的理念就是为了提高 高奖励轨迹的概率, 降低 低奖励轨迹的概率. 如图3所示:

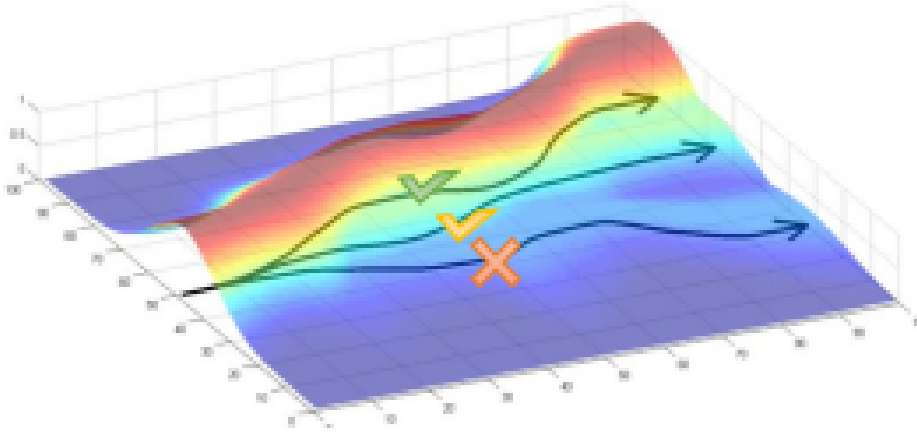


图 3: 策略梯度的理念

另外, 我们也可从策略梯度方法的梯度的原始形式进行理解:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_\theta \log p_\theta(\tau_i) r(\tau_i)$$

其中:

- $\nabla_\theta p_\theta(\tau_i)$ 反映了轨迹 τ_i 的发生概率随参数 θ 变化的方向, 参数在该方向更新时, 若沿着正方向, 则该轨迹的概率会增大, 反之减小
- $r(\tau_i)$ 控制更新的方向与大小程度, 若 $r(\tau_i)$ 为正且大, 则更新参数后, τ_i 的概率就会增大, 反之则会抑制该轨迹的概率.

5 部分可观察马尔可夫决策过程的梯度下降过程

对于部分可观察马尔可夫决策过程, 策略为 $\pi_\theta(a_t|o_t)$. 与完全观察 MDP 相比, 主要区别在于状态满足马尔可夫特性, 但观察通常不满足. 不过我们仍然可以轻松写出梯度:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left[\left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|o_{i,t}) \right) \left(\sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right) \right]$$

因为实际上马尔科夫特性并没有被用到,所以在策略梯度方法中,我们无需对 POMDP 进行更改.但需要注意的是,对于演员-评论家方法与 Q-Learning 方法,我们是需要更改的,这个后面会讲.

6 策略梯度方法的问题

那么策略梯度方法有没有什么问题呢?答案是肯定的,比如其存在高方差的问题,考虑图4的情况:

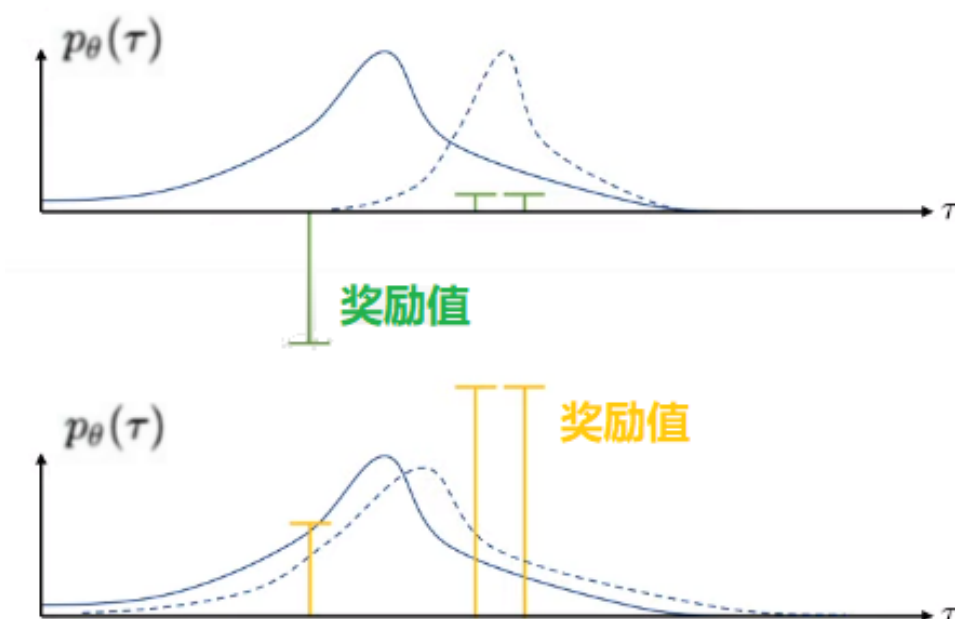


图 4: 高方差问题

其中实线为改进前策略,虚线为改进后策略,横轴表示轨迹 τ ,纵轴表示轨迹的分布概率 $p_{\theta}(\tau)$.先以上边的图为例,如果我们采样三次,当存在左边的负值奖励值(且负值较大),右边两个较小的奖励采样时,那么轨迹的概率分布会发生较大变化,即变尖且右移很多,因为负奖励值较大.改进之后,负奖励的轨迹就几乎不会再出现了,这显然不是我们想要的.如果我们的奖励函数来自于马尔可夫决策过程,那么即使我们给函数添加一个偏置(常数),最终的最优策略并不会因此改变.我们可以对图4中上面的图添加一个常数偏置,使其变为下面的图,那么此时进行采样并更新策略,概率分布则会变得更‘平缓’,因此方差会变大.

策略梯度估计器有非常高的方差,这使得样本的选择变得极为重要,因为不同的值会导致差异较大的策略梯度.需要注意的是,如果取样的次数趋近于无穷,则策略梯度估计器将总能生成正确的答案,所以添加常数到奖励函数中并不会产生任何变化,因此并不可行.但在有限取样次数中,添加一个偏置到奖励函数中仍然是有效的.

上述特性使得策略梯度方法难以使用, 若要在实际中使策略梯度方法奏效, 我们就不得不去考虑降低如此高的方差.

7 使用因果关系减少方差

我们可以使用 **因果关系 (causality)** 这一策略来减少方差. 因果关系指的是: 当 $t < t'$, 时间 t' 的策略无法影响在时间 t 的奖励, 换句话说, 我们现在所做的事不会影响以前获得的奖励. 需要注意的是, 这与马尔科夫特性不同, 马尔可夫特性强调的是未来的状态与过去的状态的无关, 这一点有时正确有时错误, 而因果关系总是正确的.

回忆策略梯度方法的梯度:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \right) \left(\sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right) \right]$$

提出策略梯度的求和, 我们可以重写这一等式为:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \left[\nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \left(\sum_{t'=1}^T r(s_{i,t'}, a_{i,t'}) \right) \right]$$

注意此时求和目标的变化:

- $\left(\sum_{t'=1}^T r(s_{i,t'}, a_{i,t'}) \right)$ 为某轨迹的总奖励
- $\left[\nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \left(\sum_{t'=1}^T r(s_{i,t'}, a_{i,t'}) \right) \right]$ 为某轨迹 t 时刻的策略梯度与其总奖励的乘积
- $\sum_{t=1}^T \left[\nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \left(\sum_{t'=1}^T r(s_{i,t'}, a_{i,t'}) \right) \right]$ 为某轨迹所有时刻求和
- $\sum_{i=1}^N \sum_{t=1}^T \left[\nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \left(\sum_{t'=1}^T r(s_{i,t'}, a_{i,t'}) \right) \right]$ 为所有轨迹求和

接下来因为 $t' < t$ 时的奖励都不重要, 也就是 t 时刻的策略只会影响 t 时刻 **以后** 的奖励函数, 我们可以得到:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \left[\nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \left(\sum_{t'=t}^T r(s_{i,t'}, a_{i,t'}) \right) \right]$$

因为我们从求和中移除了一些项, 这使得最终的方差减小了. 这里我们定义 **未来奖励 (reward-to-go)** $\hat{Q}_{i,t}$ 为:

$$\hat{Q}_{i,t} = \sum_{t'=t}^T r(s_{i,t'}, a_{i,t'})$$

则我们的梯度可以写成:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \left[\nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \hat{Q}_{i,t} \right]$$

8 使用基准线来减少方差

在小节6中我们可以发现, 奖励函数的设计对轨迹优化的影响较大, 那么如果我们将奖励函数减去一个平均值, 用平均值来评估轨迹的好坏, 这样就可以避免奖励函数的方差. 这里的平均值可以取所有轨迹的平均值. 也就是:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log p_{\theta}(\tau) [r(\tau) - b]$$

其中 $b = \frac{1}{N} \sum_{i=1}^N r(\tau)$ 为平均值 (基准线, baseline). 这样的操作会使得坏的轨迹有负的奖励, 好的轨迹有正的奖励. 这一方法是合理的, 因为我们可以计算基准线那一项的期望:

$$E[\nabla_{\theta} \log p_{\theta}(\tau) b] = \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) b d\tau$$

利用链式法则, 易知:

$$\nabla \log(f(x)) = \frac{1}{f(x)} \cdot \nabla f(x)$$

代入原式中, 可得:

$$\int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) b d\tau = \int \nabla_{\theta} p_{\theta}(\tau) b d\tau = b \nabla_{\theta} \int p_{\theta}(\tau) d\tau = b \nabla_{\theta} 1 = 0$$

则我们可以说减去一个基准线, 在期望中仍是无偏的. 平均奖励作为基准线虽然不是最佳的, 但是已经是一个比较好的选择了.

9 最优基准线

但是我们仍可以更进一步. 首先让我们分析其方差, 众所周知, 方差公式为:

$$\text{Var}[x] = E[x^2] - E[x]^2$$

而目标函数的梯度为:

$$\nabla_{\theta} J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[\nabla_{\theta} \log p_{\theta}(\tau) (r(\tau) - b) \right]$$

则其方差为:

$$\text{Var} = E_{\tau \sim p_{\theta}(\tau)} \left[\left(\nabla_{\theta} \log p_{\theta}(\tau) (r(\tau) - b) \right)^2 \right] - E_{\tau \sim p_{\theta}(\tau)} \left[\nabla_{\theta} \log p_{\theta}(\tau) (r(\tau) - b) \right]^2$$

对于第二项, 因为我们上面证明了基准线项的期望为 0, 所以第二项实际上等于 $E_{\tau \sim p_{\theta}(\tau)} \left[\nabla_{\theta} \log p_{\theta}(\tau) r(\tau) \right]$. 为了方便推导, 我们令:

$$g(\tau) = \nabla_{\theta} \log p_{\theta}(\tau)$$

那么此时我们对方差求关于 b 的梯度, 可得:

$$\begin{aligned} \frac{d\text{Var}}{db} &= \frac{d}{db} E \left[g(\tau)^2 (r(\tau) - b)^2 \right] \\ &= \frac{d}{db} \left(E[g(\tau)^2 r(\tau)^2] - 2E[g(\tau)^2 r(\tau) b] + b^2 E[g(\tau)^2] \right) \\ &= -2E[g(\tau)^2 r(\tau)] + 2bE[g(\tau)^2] \end{aligned}$$

我们只需令其结果等于 0, 即可得到最小方差值. 则最优基准线为:

$$b^* = \frac{E[g(\tau)^2 r(\tau)]}{E[g(\tau)^2]}$$

本质上来说, 基准线是关于梯度大小的加权平均. 所以在实际中, 我们经常只使用奖励期望作为基准线.

10 Off-policy 策略梯度法

10.1 为什么说策略梯度是 on-policy 的?

让我们回顾一下策略梯度方法:

$$\theta^* = \arg \max_{\theta} J(\theta)$$

$$J(\theta) = E_{\tau \sim p_{\theta}(\tau)}[r(\tau)]$$

$$\nabla_{\theta} J(\theta) = E_{\tau \sim p_{\theta}(\tau)}[\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]$$

在求梯度的时候, 轨迹服从 $p_{\theta}(\tau)$ 会出问题, 因为每次我们改变 θ , 我们都需要重新抽取样本. 在 REINFORCE 算法中, 我们无法跳过步骤 1 - 从 $\pi_{\theta}(a_t|s_t)$ 中采样 $\{\tau^i\}$. 如果在深度神经网络中, 这会成为一个大问题, 因为深度神经网络中每次使用梯度更新都只能改变一点点 (因为系统是非线性的, 我们无法一次迈出很大一步). 这一特性使得策略梯度方法的代价非常高昂, 但是如果我们就是头铁要使用策略梯度, 则我们可以使用 **重要性抽样 (importance sampling)** 技术.

10.2 重要性抽样

如果我们没有来自于最新概率分布 $p_{\theta}(\tau)$ 的样本, 只有其他 (之前的) 概率分布 $\bar{p}_{\theta}(\tau)$ 的样本, 那么我们是否能够通过 $\bar{p}_{\theta}(\tau)$ 的样本来计算 $p_{\theta}(\tau)$ 下的目标函数和其梯度呢? 我们可以使用重要性抽样:

$$E_{x \sim p(x)}[f(x)] = \int p(x) f(x) dx \quad (1)$$

$$= \int \frac{q(x)}{q(x)} p(x) f(x) dx \quad (2)$$

$$= \int q(x) \frac{p(x)}{q(x)} f(x) dx \quad (3)$$

$$= E_{x \sim q(x)} \left[\frac{p(x)}{q(x)} f(x) \right] \quad (4)$$

这时我们有:

$$J(\theta) = E_{\tau \sim \bar{p}(\tau)} \left[\frac{p_{\theta}(\tau)}{\bar{p}_{\theta}(\tau)} r(\tau) \right]$$

其中最新概率分布 $p_\theta(\tau)$ 与某些其他概率分布 $\bar{p}_\theta(\tau)$ 的比值为:

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

10.3 梯度推导

那么此时我们就可以使用重要性采样来推导 off-policy 策略梯度方法的梯度了. 我们的目的是已知参数 θ , 去估计某些新参数 θ' :

$$J(\theta) = E_{\tau \sim p_\theta(\tau)} [r(\tau)] \quad (5)$$

$$J(\theta') = E_{\tau \sim p_\theta(\tau)} \left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)} r(\tau) \right] \quad (6)$$

注意在估计新参数 θ' 时, 轨迹采样仍然服从 $p_\theta(\tau)$. 不难发现 $J(\theta')$ 中只有 $p_{\theta'}(\tau)$ 与 θ' 有关. 那么我们可以对 $J(\theta')$ 求梯度:

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim p_\theta(\tau)} \left[\frac{\nabla_{\theta'} p_{\theta'}(\tau)}{p_\theta(\tau)} r(\tau) \right]$$

这时我们需要之前提过的小技巧:

$$p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) = \nabla_\theta p_\theta(\tau)$$

代入可得:

$$\begin{aligned} E_{\tau \sim p_\theta(\tau)} \left[\frac{\nabla_{\theta'} p_{\theta'}(\tau)}{p_\theta(\tau)} r(\tau) \right] &= E_{\tau \sim p_\theta(\tau)} \left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)} \nabla_{\theta'} \log p_{\theta'}(\tau) r(\tau) \right] \\ &= E_{\tau \sim p_\theta(\tau)} \left[\left(\prod_{t=1}^T \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \right) \left(\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) \right) \left(\sum_{t=1}^T r(s_t, a_t) \right) \right] \end{aligned}$$

同时我们还可以考虑因果关系, 可以得到:

$$= E_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) \left(\prod_{t'=1}^t \frac{\pi_{\theta'}(a_{t'}|s_{t'})}{\pi_\theta(a_{t'}|s_{t'})} \right) \left(\sum_{t'=t}^T r(s_{t'}, a_{t'}) \left(\prod_{t''=t}^{t'} \frac{\pi_{\theta'}(a_{t''}|s_{t''})}{\pi_\theta(a_{t''}|s_{t''})} \right) \right) \right]$$

我们可以发现中间的项 $\prod_{t'=1}^t \frac{\pi_{\theta'}(a_{t'}|s_{t'})}{\pi_\theta(a_{t'}|s_{t'})}$ 从 1 累乘到 t , 这是因为未来的动作不会影响到当前的比值权重, 但这一项会导致最终结果在 T 内指数型增长/减少 (毕竟是累乘, 无论比值是小于 1 还是大于 1, 如果时间比较长, 那么乘积会很大/很小).

如果我们无视第二个累乘项, 那么我们就可以得到 **策略迭代算法 (policy iteration algorithm)**:

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) \left(\prod_{t'=1}^t \frac{\pi_{\theta'}(a_{t'}|s_{t'})}{\pi_\theta(a_{t'}|s_{t'})} \right) \left(\sum_{t'=t}^T r(s_{t'}, a_{t'}) \right) \right]$$

10.4 对重要性采样的一阶近似 (存疑)

为了避免梯度消失/爆炸的问题, 我们可以重新考虑目标函数, 原先 on-policy 梯度策略的为:

$$J(\theta) = \sum_{t=1}^T E_{(s_t, a_t) \sim p_\theta(s_t, a_t)} [r(s_t, a_t)] = \sum_{t=1}^T E_{s_t \sim p_\theta(s_t)} \left[E_{a_t \sim \pi_\theta(a_t|s_t)} [r(s_t, a_t)] \right]$$

按照这个思路, 则 off-policy 梯度策略为:

$$J(\theta') = \sum_{t=1}^T E_{s_t \sim p_\theta(s_t)} \left[\frac{p_{\theta'}(s_t)}{p_\theta(s_t)} E_{a_t \sim \pi_\theta(a_t|s_t)} \left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} r(s_t, a_t) \right] \right]$$

虽然这样就不再有累乘问题, 但同时又带来另一个问题 - 我们需要知道 $p_{\theta'}(s_t)$, 但我们可以无视 $\frac{p_{\theta'}(s_t)}{p_\theta(s_t)}$ 这一项来近似处理. 虽然这一操作会导致结果不再无偏, 但在实际中, 若 θ 与 θ' 相近, 就没什么问题.

11 使用自动微分的策略梯度

自动微分法是一种介于符号微分和数值微分的方法, 数值微分强调一开始直接代入数值近似来求解; 符号微分强调直接对代数进行求解, 最后才代入数值; 而自动微分法将符号微分法应用于最基本的算子, 如常数, 幂函数, 指数函数, 三角函数等等, 然后代入数值, 保留中间结果, 最后再应用于整个函数. 因此其相当灵活, 可以做到完全向用户隐藏微分求解过程, 由于它只对基本函数或常数运用符号微分法则, 所以它可以灵活结合编程语言的循环结构, 条件结构等, 使用自动微分和不使用自动微分对代码总体改动非常小, 并且由于它的计算实际是一种图计算, 可以对其做很多优化, 这也是为什么该方法在现代深度学习系统中得以广泛应用的原因. 对于梯度

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \left(\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \hat{Q}_{i,t} \right)$$

其中, 计算 $\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})$ 是很费内存的, 且计算量也大. 因此我们可以使用图结构来利用 PyTorch 或 TensorFlow 中的自动微分器. 考虑最大似然估计的目标函数:

$$J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \log \pi_\theta(a_{i,t}|s_{i,t})$$

其梯度为:

$$\nabla_\theta J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})$$

因为 $\hat{Q}_{i,t}$ 本身与参数无关, 所以我们可以构造一个 ‘虚拟的加权目标函数’:

$$\tilde{J}(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \log \pi_\theta(a_{i,t}|s_{i,t}) \hat{Q}_{i,t}$$

因为 $\hat{Q}_{i,t}$ 是权重, 则我们直接使用自动微分器来求梯度就行了. 最大似然估计的 TensorFlow 伪代码如下:

```
1 # Given:
2 # actions -(N*T) x Da tensor of actions
3 # states -(N*T) x Ds tensor of states
4 # Build the graph:
5 # This should return (N*T) x Da tensor of action logits
6 logits = policy.predictions(states)
```

```

7 negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=
    actions, logits=logits)
8 loss = tf.reduce_mean(negative_likelihoods)
9 gradients = loss.gradients(loss, variables)

```

Listing 1: Maximum Likelihood pseudo-code

策略梯度的 TensorFlow 伪代码如下:

```

1 # Given:
2 # actions - (N*T) x Da tensor of actions
3 # states - (N*T) x Ds tensor of states
4 # q_values - (N*T) x 1 tensor of estimated state-action values
5 # Build the graph:
6 # This should return (N*T) x Da tensor of action logits
7 logits = policy.predictions(states)
8 negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=
    actions, logits=logits)
9 weighted_negative_likelihoods = tf.multiply(negative_likelihoods, q_values)
10 loss = tf.reduce_mean(weighted_negative_likelihoods)
11 gradients = loss.gradients(loss, variables)

```

Listing 2: Policy Gradient pseudo-code

其中 `q_values` 就是我们的 $\hat{Q}_{i,t}$.

12 实际中的梯度策略方法

- 记住梯度高方差!
 - 这一点与监督学习不同
 - 梯度会有很多噪声
- 也许需要用到比监督学习大很多的批次 (batch) 大小
- 调整学习率比较困难
 - Adam 或许还可以, 但是动量 SGD 之类的优化器会非常难用

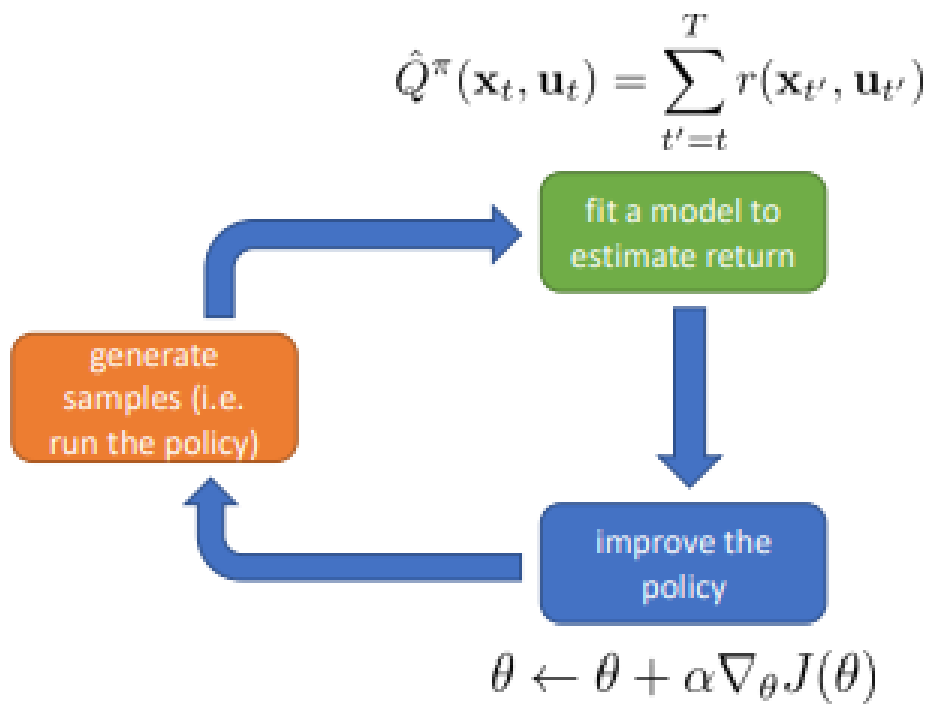


图 5: 策略梯度方法的过程

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>