UC Berkeley CS285 第 6 讲: 演员-评论家算法

翻译/总结: BruceQQC

版本: 1

日期: 2022 年 5 月 7 日

1 回忆: 策略梯度法

上一讲我们了解到策略梯度方法的过程如图1所示:

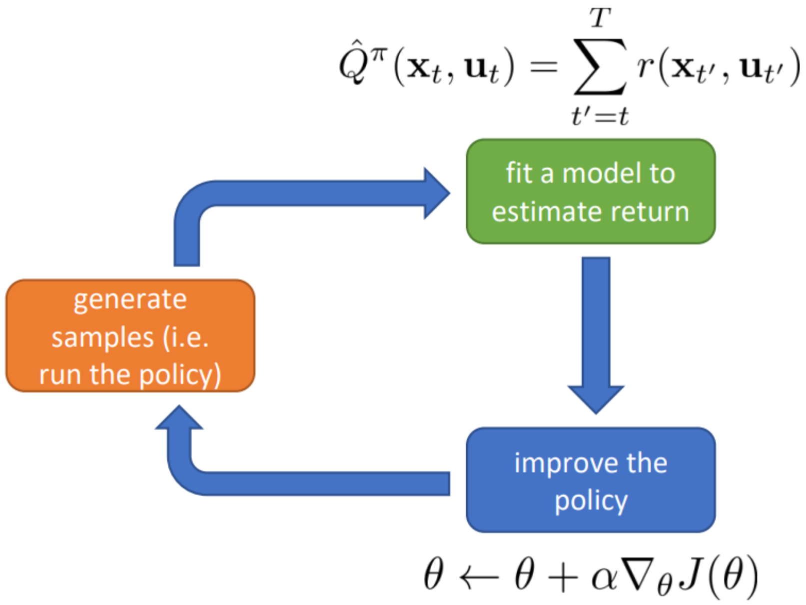

图 1: 策略梯度法过程

策略梯度法一个典型是 REINFORCE 算法, 步骤如图2所示:

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i) \right) \left(\sum_{t'=t}^{T} r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

图 2: REINFORCE 算法

应用因果关系 (causality), 我们可以得到目标函数的梯度:

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

我们使用 $\hat{Q}_{i,t}^{\pi} = \sum_{t'=1}^{T} r(s_{i,t'}, a_{i,t'})$ 表示在策略 π_θ 下的第 i 条轨迹, 其在 t 时刻, 状态 $s_{i,t}$ 下采取动作 $a_{i,t}$ 之后的奖励和. 这是对 '未来奖励' 的估计, 如图3所示:

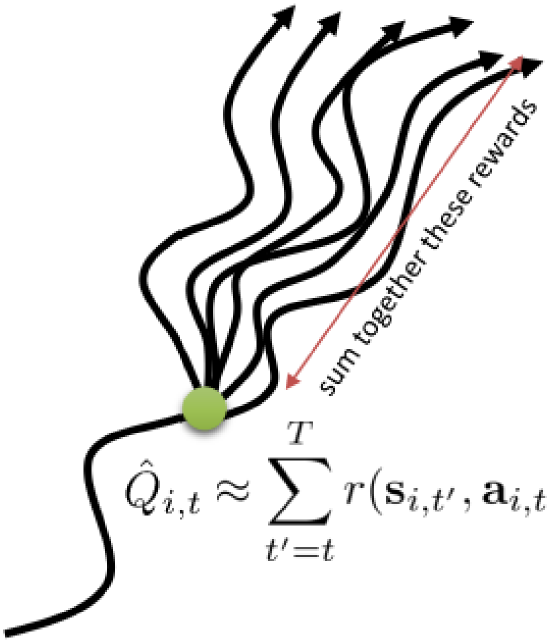

图 3: '未来奖励'

需要注意的是, 在策略 π_θ 下, 在状态 s_t 采取动作 a_t, 可能会有多条轨迹.

2 提升策略梯度法

既然在策略梯度法中, 我们对 '未来奖励' 的期望值进行了估计, 那么我们可以尝试去获得一个更好的估计. **真正的** '未来奖励' 的期望值为:

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

代入会得到较低方差的代价函数:

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

我们还可以更进一步 - 应用基准线:

$$\nabla_\theta J(\theta) \approx \sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(Q(s_t, a_t) - b\right)\right)$$

这里基准线的选择有不少, 我们可以令其为奖励的平均值; 或者是 Q 值的平均值 $b_t = \frac{1}{N}\sum_i Q(s_{i,t}, a_{i,t})$; 亦或是令基准线只与状态有关, 这样做的好处是我们可以在所有该状态下开始的概率上计算平均奖励, 说得这么绕, 其实也就是值函数:

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

则策略梯度可表示为:

$$\nabla_\theta J(\theta) \approx \sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_\theta \log\pi_\theta(a_{i,t}|s_{i,t})\Big(Q(s_t, a_t) - V(s_t)\Big)\right)$$

这样, 我们就得到了 **优势函数 (advantage function)** $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$, 其代表了对于给定策略 π, 在状态 s_t 采取动作 a_t 能比该策略平均情况的未来奖励期望值好多少的估计. 因此在梯度中, 优势函数乘以 $\log\pi$ 是有意义的, 因为这说明我们想要提升 **采取比状态中平均动作好的动作**的概率, 反之亦然. 总结一下, 我们有:

- 状态-动作 Q 函数 $Q^\pi(s_t, a_t)$ 与状态和动作有关, 其表示了对于给定策略 π, 在状态 s_t 下采取动作 a_t 的未来奖励之和的期望
- 状态值函数 $V^\pi(s_t)$ 仅与状态有关, 其表示了对于给定策略 π, 在状态 s_t 下未来奖励之和的平均值
- 优势函数 $A^\pi(s_t, a_t)$ 表示了对于给定策略 π, 在状态 s_t 下采取动作 a_t 能比该策略平均情况的未来奖励期望值好多少

3 拟合值函数

原始的策略梯度法中使用的基准线是无偏的, 我们在第 4 讲中证明过, 但是由于单次样本估计, 导致其方差太高. 为了降低方差, 我们使用了优势函数来估计代价函数的梯度, 优势函数估计得越好, 方差就越低, 但要注意这一方法不是 **无偏的**. 那么我们要如何估计优势函数呢? 回忆三个函数的定义:

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

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

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$$

既然优势函数由值函数和 Q 函数组成, 那么如果我们能找到值函数和 Q 函数之间的关系, 则只需估计值函数或 Q 函数即可. 这里我们需要对值函数和 Q 函数有透彻的理解:

- 值函数是从状态 s 开始, 遵循策略 π 所得到的预期奖励 (这儿是根据策略 π 对 **所有的**动作取期望)
- Q 函数是从状态 s 开始, 遵循策略 π, 采取行动 a 所得到的与其奖励 (它专注于在特定状态的特定动作)

所以 Q^π 与 V^π 之间的关系为 $V^\pi(s) = \sum_{a\sim\mathcal{A}}\pi_\theta(a|s)\cdot Q^\pi(s, a)$, 也就是我们对每个动作得

到的奖励值乘以在策略 $\pi(a|s)$ 下采用该动作的概率, 然后进行求和. 这里为了方便理解, 我们可以举网格世界 (grid world) 的例子, 我们将 (上/下/左/右) 的概率与 (上/下/左/右) 前一步的状态值相乘.

根据 Q 函数的定义, 对于给定 t 时刻的状态和动作, t 时刻的奖励就能确定, 则我们可以将 t 时刻的奖励提出来:

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \sum_{t'=t+1}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})\ |s_t, a_t]$$

求和项实际为值函数在状态 s_{t+1} 的期望, 如果我们用轨迹样本来估计这个期望, 做近似, 则可得优势函数为:

$$A^\pi(s_t, a_t) \approx r(s_t, a_t) + V^\pi(s_{t+1}) - V^\pi(s_t)$$

这样的话, 优势函数就只与值函数有关, 我们只需使用神经网络拟合出 V^π, 就可近似求得 Q 函数和优势函数了, 如图4所示:

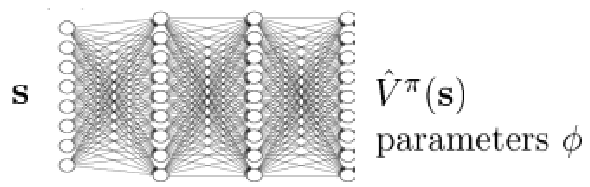

图 4: 使用神经网络拟合值函数

这就是演员-评论家算法, 如图5所示:

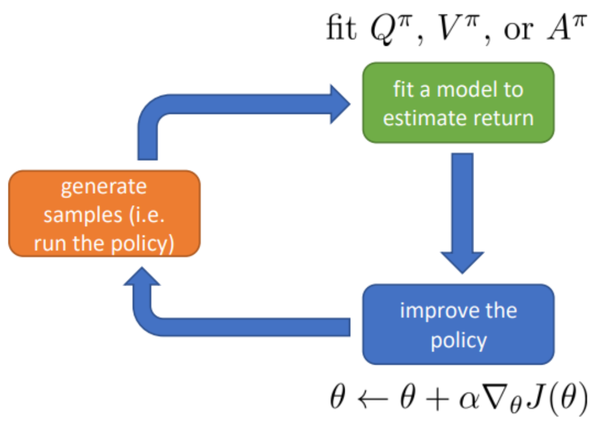

图 5: 演员-评论家算法, 这里我们选择拟合 V^π

4 策略评估

现在让我们考虑如何评估策略. 拟合值函数的过程本身是一种策略评估, 也就是通过值函数我们可以评价策略的好坏, 因为值函数的表达式为 $V^\pi(s_t) = E_{a_t\sim\pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]$, 而目标函数 $J(\theta) = E_{s_1\sim p(s_1)}[V^\pi(s_1)]$ 从初始状态开始. 这我们不难发现, 目标函数是值函

数的期望, 所以拟合值函数也会给我们带来目标函数. 我们可以使用 **蒙特卡洛策略评估法 (Monte Carlo Policy Evaluation)** 来获得值函数的近似 - 多次运行策略, 沿着由该策略生成的轨迹进行奖励求和, 然后使用其作为一个无偏但高方差的, 该策略的, 整体奖励估计. 当只进行一次轨迹采样时:

$$V^\pi(s_t) \approx \sum_{t'=t}^{T} r(s_{t'}, a_{t'})$$

当进行多次轨迹采样时:

$$V^\pi(s_t) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t'=t}^{T} r(s_{t'}, a_{t'})$$

注意每次进行采样时, 我们都需要重置仿真环境到 s_t.

4.1 使用神经网络近似值函数

但如果我们使用神经网络来近似值函数呢? 这么做在近似值函数的时候会减少方差, 因为即使我们不能访问同一个状态两次, 神经网络实际上也能够 '意识到' 我们在不同轨迹访问的不同状态的相似之处, 则当神经网络试图取估计这两个状态的值时, 其会得到相似的结果, 这是一般化 (generalization) 的特性, 如图6所示:

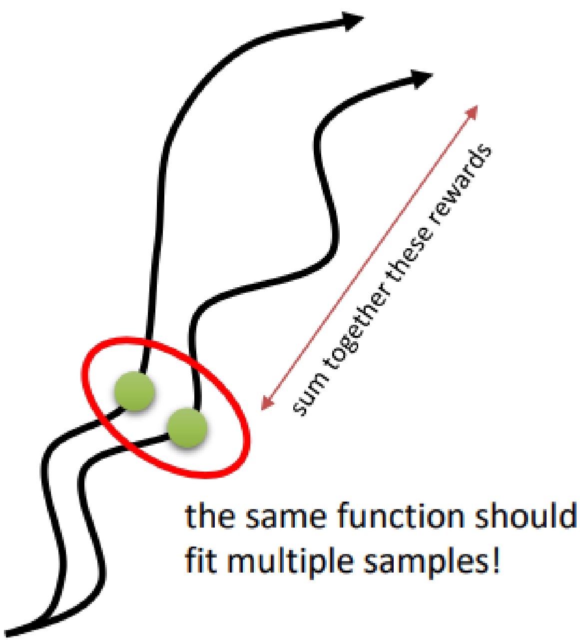

图 6: 相似状态的估计也相似

进而值函数拟合的方法就是收集训练数据:

$$\left\{\left(s_{i,t},\ y_{i,t} := \sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'})\right)\right\}$$

然后使用最小二乘损失函数 $\mathcal{L}(\phi) = \frac{1}{2}\sum_i ||\hat{V}_\phi^\pi(s_i) - y_i||^2$ 进行监督回归训练.

4.2 自助估计法近似值函数

我们可以做得更好吗? 答案是使用 **自助估计 (Bootstrapped Estimate)**. 我们理想的目标为:

$$
\begin{aligned}
y_{i,t} &= E_{\pi_\theta}[r(s_{i,t'}, a_{i,t'})|s_{i,t}] \\
&\approx r(s_{i,t}, a_{i,t}) + \sum_{t'=t+1}^{T} E_{\pi_\theta}[r(s_{t'}, a_{t'})|s_{i,t}] \\
&\approx r(s_{i,t}, a_{i,t}) + V^\pi(s_{i,t+1}) \\
&\approx r(s_{i,t}, a_{i,t}) + \hat{V}_\phi^\pi(s_{i,t+1})
\end{aligned}
$$

这里 $\hat{V}_\phi^\pi(s_{i,t+1})$ 我们可以直接使用之前拟合的值函数. 这样训练集:

$$
\left\{\left(s_{i,t},\ y_{i,t} := r(s_{i,t}, a_{i,t}) + \hat{V}_\phi^\pi(s_{i,t+1})\right)\right\}
$$

得到了大幅度的简化. '自助' 指的是使用下一刻的值函数来估计当前时刻的值函数. 自助估计法有更低的方差, 因为我们使用了之前拟合的 $\hat{V}_\phi^\pi(s_{i,t+1})$, 但又更高的偏置, 因为 $\hat{V}_\phi^\pi(s_{i,t+1})$ 可能不对.

4.3 批次演员-评论家算法

最简单的 AC 算法为批次演员-评论家算法 (batch actor-critic algorithm), 算法流程如图7所示:

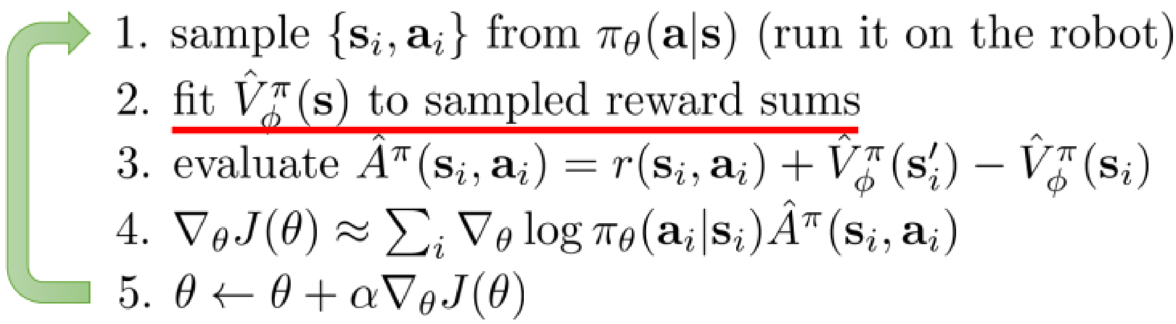

图 7: 批次演员-评论家算法

可以使用蒙特卡洛估计或自助估计来获取第二步的 $\hat{V}_\phi^\pi(s_{i,t+1})$.

4.4 对比演员-评论家算法与策略梯度法

两方法仅在绿色模块 'fit a model to estimate return' 处有所不同. 策略梯度法通过估计值 $\hat{Q}_{i,t}$ 来计算目标函数的梯度, 重点在于估计计算. 而对于演员-评论家算法, 其是通过拟合 Q^π, V^π 和 A^π, 进而得到更好的梯度估计, 重点在于拟合模型.

5 折扣因子

上述讨论基于有限域的情况, 但是在无限域中, 例如训练人型机器人学习行走, 其动作是连续的, 无限的, 那么 $\hat{V}^{\pi}_{\phi}$ 可能会变得无限大. 这时我们需要引入 **折扣因子 (discount factor)** 来解决这一问题, 也就是越早获得的奖励比越晚获得的奖励要好, 这也容易理解, 即今天获得 100 万美金和 100 年后获得 100 万美金的快乐程度不一样. 我们原来的目标 $y_{i,t}$ 为:

$$y_{i,t} \approx r(s_{i,t}, a_{i,t}) + \hat{V}^{\pi}_{\phi}(s_{i,t+1})$$

添加折扣因子之后, 我们的目标 $y_{i,t}$ 变为:

$$y_{i,t} \approx r(s_{i,t}, a_{i,t}) + \gamma\hat{V}^{\pi}_{\phi}(s_{i,t+1})$$

折扣因子 $\gamma \in [0, 1]$, 通常可以设置成 0.99. 引入折扣因子会改变 MDP, 使其添加一个额外的 '死亡状态', 即 '死亡' 后奖励一直为 0, 如图8所示:

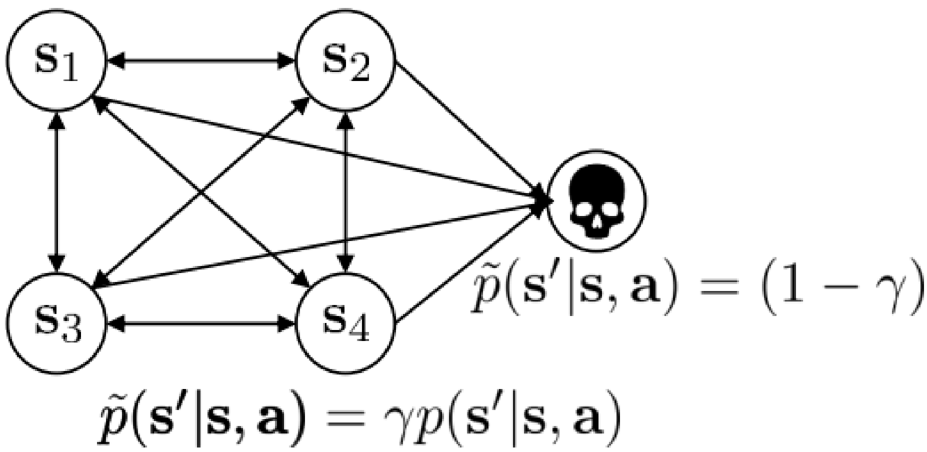

图 8: 带有 '死亡状态' 的 MDP

上图中, 我们原本含有 s_1, s_2, s_3, s_4 4 个状态, 引入折扣因子后多了一个 '死亡状态', 所有其他状态都可以 **单向到达** '死亡状态'. 原有状态的转移概率更新为:

$$\tilde{p}(s'|s, a) = \gamma\, p(s'|s, a)$$

且 $\tilde{p}(s_{\text{death}}|s, a) = 1 - \gamma$. 稍作修改, 我们也可以将折扣因子应用到蒙特卡洛策略梯度中:

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

$$\text{选项 2: } \nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T}\gamma^{t-1} r(s_{i,t}, a_{i,t})\right)\right)$$

需要注意的是, 这两个选项并不一样, 选项 1 仅对 t' 部分进行折扣, 而选项 2 则是根据策略梯度的定义. 在考虑因果关系之后, 可以看出两个选项的不同. 对选项 2, 我们将部分

折扣因子取出, 可得:

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

这样就可以看出选项 2 不仅对奖励做了折扣, 还对梯度也进行了折扣, 这样使得时间越往后, 策略的影响力越小, 这显然不是我们想要的, 所以通常会采用选项 1.

5.1 使用折扣因子的演员-评论家算法

而在演员-评论家算法中, 我们可以在评估 $\hat{A}^\pi$ 时应用折扣因子:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(r(s_{i,t},a_{i,t})+\gamma\hat{V}_\phi^\pi(s_{i,t+1})-\hat{V}_\phi^\pi(s_{i,t})\right)\right)$$

我们可以对比批次演员-评论家算法和在线演员-评论家算法, 如图9所示:

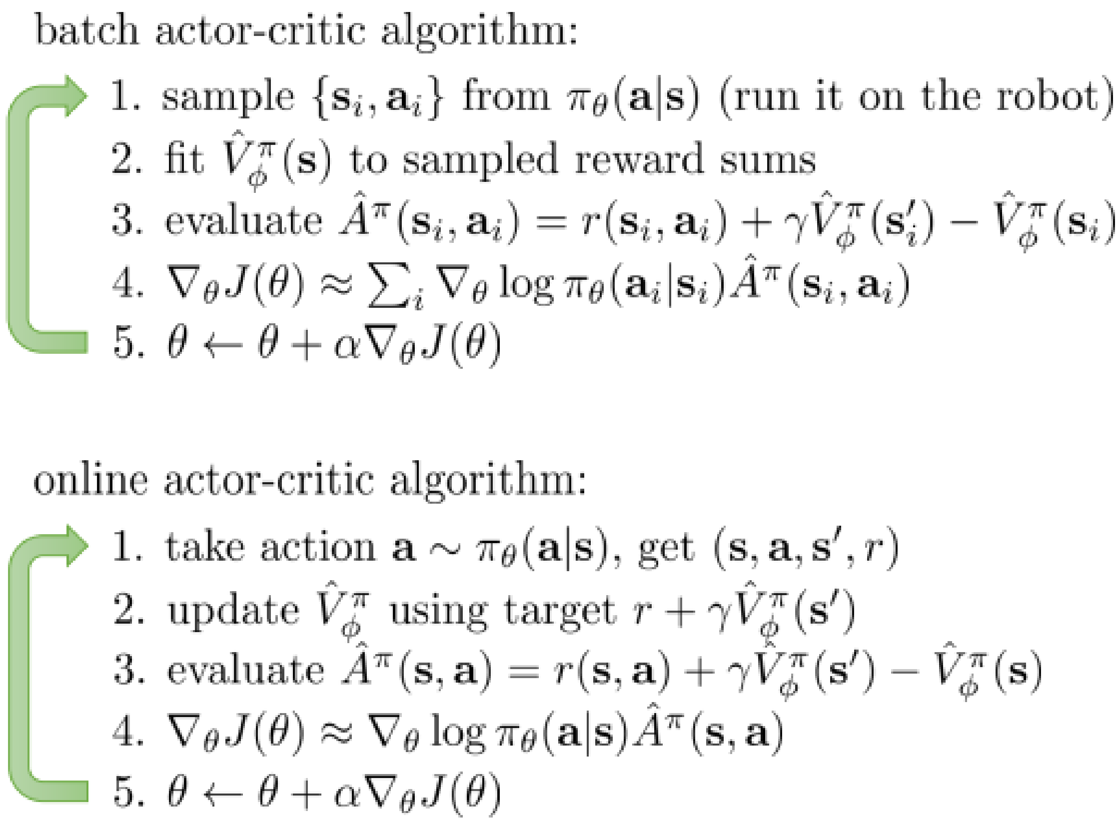

图 9: 批次演员-评论家算法与在线演员-评论家算法

对于批次演员-评论家算法, 引入折扣因子时, 只需对步骤 3 进行更新即可. 但对于在线演员-评论家算法, 因为其在步骤 1 得到的不是多个采样, 而是一个轨迹样本 (s, a, s', r). 则我们需要在步骤 2 中使用该轨迹样本先更新值函数 $\hat{V}_\phi^\pi(s)$, 然后在步骤 3 中更新优势函数 $\hat{A}_\phi^\pi(s, a)$.

6 网络设计

6.1 结构设计

对于网络结构, 我们可以使用值函数 $\hat{V}_\phi^\pi(s)$ 与策略 $\pi_\theta(a|s)$ 分离的神经网络, 如图10所示:

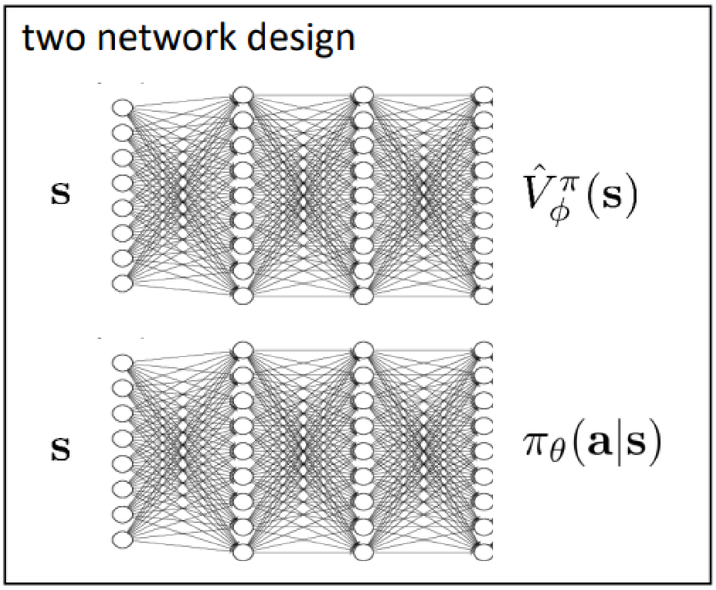

图 10: 两网络结构

这么做的优点是结构简单, 而且稳定性较高. 但缺点是演员网络与评论家网络之间无法共享特征. 那么就有了第二种网络设计, 如图11所示:

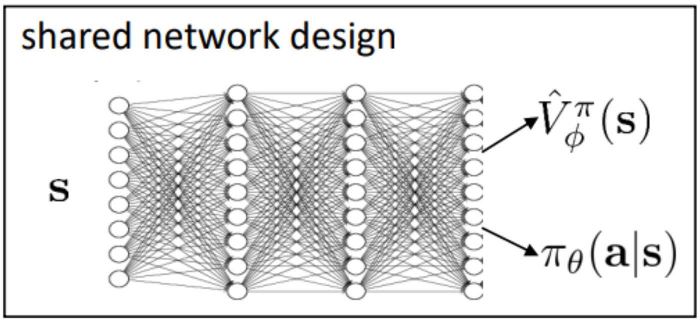

图 11: 共享网络结构

6.2 并行训练设计

此外, 对于在线演员-评论家, 因为其每次都只使用一条轨迹进行策略迭代, 所以效率并不高, 我们可以采用多个 ‘worker’ 来并行化步骤 2 和步骤 4. 有两种并行的方法, 如图所示:

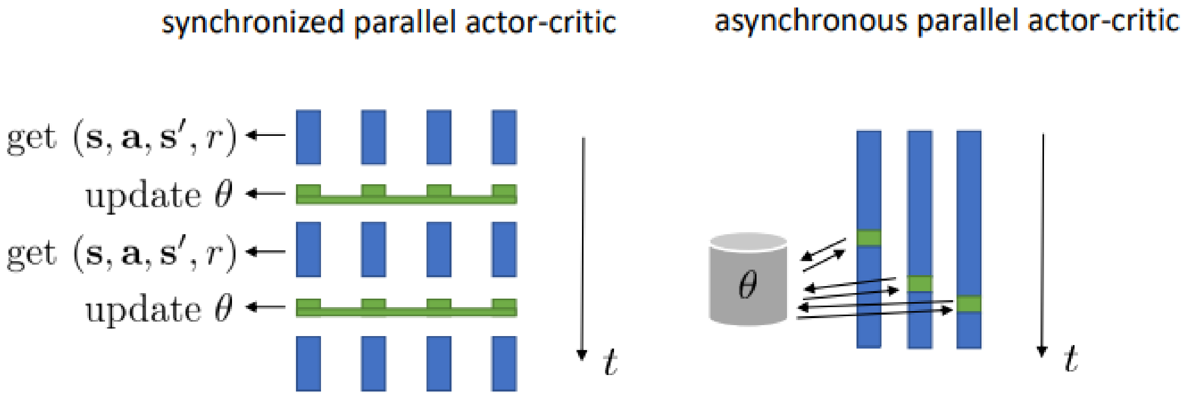

图 12: 同步与异步并行方法

第一种是 **同步并行 (synchronized parallel)**, 其分配多个 'worker', 每个 'worker' 各自生成自己的轨迹样本, 等待所有样本生成结束, 然后使用 **所有的**样本来更新参数 θ, 接着循环; 第二种是 **异步并行 (asynchronous parallel)**, 其分配多个 'worker', 每个 'worker' 各自生成自己的轨迹样本, 只要收集到了足够多的样本就对参数 θ 进行更新. 这两种方法在数学上并不等价, 在实际中, 异步并行往往更快, 但结果会略有偏.

7 Off-policy 演员-评论家算法

在异步并行演员-评论家算法中, 我们可以使用有稍 '旧' 一些演员网络生成的转移, 那么接下来让我们考虑一个问题 - 我们可以彻底移除 on-policy 的假设吗? 如果我们能以某种方式使用 **由更 '旧' 的演员网络生成的转移**, 那么也许我们甚至可以不采用多线程, 而是使用同一个演员产生的不同的 (之前的) 转移. 这就是 **off-policy 演员-评论家算法**的原理, 如图13所示:

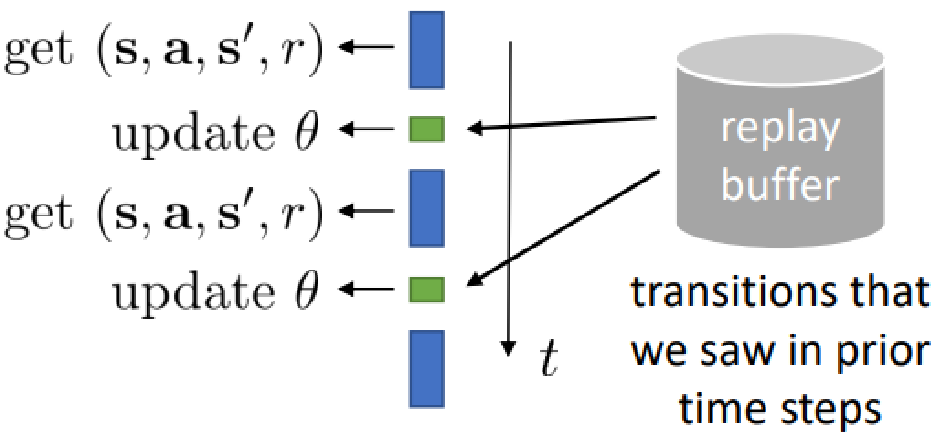

图 13: Off-policy 演员-评论家算法

上图中我们使用一个 **经验回放池 (replay buffer)**, 从其中加载批次 (batch). 这样我们就不

必使用最新的转移, 只需从经验回放池中采样整个批次即可. 如此, 对在线演员-评论家算法应用这一想法, 可得如下算法步骤:

1. 采取动作 $a \sim \pi_\theta(a|s)$, 获得 (s, a, s', r), 存储在经验回放池 $\mathcal{R}$ 中
2. 从经验回放池 $\mathcal{R}$ 中采样一个批次 $\{s_i, a_i, r_i, s_i'\}$
3. 对每个 s_i, 使用目标 $y_i = r_i + \gamma \hat{V}_\phi^\pi(s_i')$ 来更新 $\hat{V}_\phi^\pi$, 其中该批次下参数 ϕ 的损失函数为 $\mathcal{L}(\phi) = \frac{1}{N}\sum_i ||\hat{V}_\phi^\pi(s_i) - y_i||^2$, 其中 N 为批次大小
4. 评估 $\hat{A}^\pi(s_i, a_i) = r(s_i, a_i) + \gamma \hat{V}_\phi^\pi(s_i') - V_\phi^\pi(s_i)$
5. $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_i \nabla_\theta \log \pi_\theta(a_i|s_i)\hat{A}^\pi(s_i, a_i)$
6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

步骤 6 之后回到步骤 1, 以此循环. 但是上述的算法是错误的, 在步骤 3 中, 因为 a_i 不是来自于最新策略 π_θ, 则 s_i' 也不是最新演员网络采取动作所得到的结果. 所以这里我们的目标 y_i 是错误的. 改进方法也简单, 既然动作与状态不匹配, 那么我们只要限制动作即可. 也就是采用 Q 函数来替代这里的值函数, 差别在于 Q 函数需要 **在状态 s_t 下采取动作** a_t (然后遵守策略, 得到总奖励). 所以步骤 3 可以改为:

3. 对每个 s_i, a_i, 使用目标 $y_i = r_i + \gamma \hat{V}_\phi^\pi(s_i')$ 来更新 $\hat{Q}_\phi^\pi$, 其中损失函数为 $\mathcal{L}(\phi) = \frac{1}{N}\sum_i ||\hat{V}_\phi^\pi(s_i) - y_i||^2$

但是这样的修改也会导致新的问题 - 因为我们现在对 $\hat{Q}_\phi^\pi$ 进行更新,那么怎么得到 $V_\phi^\pi(s_i')$ 呢? 回忆值函数是 Q 函数的在某状态下所有动作奖励的期望, 所以我们可以设置新目标 $y_i = r_i + \gamma \hat{Q}_\phi^\pi(s_i', a_i')$, 也就是使用下一个状态 s_i' 中采取的动作 a_i'. 但需要注意的是这里的 a_i' 并 **不是**来源于经验回放池 $\mathcal{R}$, 而是我们在 $\mathcal{R}$ 中采样获得 a_i, s_i, s_i', 然后通过运行最新的策略 $\pi_\theta(a_i'|s_i')$ 来获得 a_i'.

在步骤 5 中也有错误, 因为 a_i 并不是来自于最新的策略 π_θ, 则我们不能这样计算策略梯度. 改正方法与步骤 3 中的相似, 但这次我们的对象是 a_i. 具体来说, 我们从 $\pi_\theta(a|s_i)$ 中采样 a_i^π, 注意区分 a_i^π 和 a_i. 这里 a_i 是从经验回放池采样的, 而 a_i^π 为 **如果状态 s_i 来自于经验回放池**, 则当前策略会产生的动作. 接下来我们就可以应用 a_i^π:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_i \nabla_\theta \log \pi_\theta(a_i^\pi|s_i)\hat{A}^\pi(s_i, a_i^\pi)$$

这样就对了, 因为现在 a_i^π 来自于 π_θ, 这是策略 π_θ 下对期望的无偏估计. 在实际中, 我们并不是优势函数, 而仅使用 $\hat{Q}^\pi(s_i, a_i^\pi)$, 这样做比较方便, 但随之而来的是高方差的问题. 但是这里是没问题的, 因为我们并不需要与仿真器交互来采样 a_i', 所以实际上可以通过生成更多的动作样本来降低方差 (而无需生成更多的状态样本), 在这种情形下, 我们不需要多次运行仿真, 仅需要多次运行神经网络即可. 相比之下, 这种高方差是可以接受的.

7.1 还有问题吗

还真有问题 - s_i 不是来自于 $p_\theta(s)$, 但对于这一点我们无能为力. 为什么这个问题不算大呢? 直观来讲, 这是因为我们最终想要的是 $p_\theta(s)$ 上的最优策略, 而我们的经验回放器包

含了来自于最新策略和之前策略的采样, 所以我们只是多做了一些工作, 但没有漏掉重要的东西.

8 使用评论家函数作为基准线

演员-评论家算法与策略梯度法的比较如图14所示:

Actor-critic: $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\left(r(\mathbf{s}_{i,t},\mathbf{a}_{i,t}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1}) - \hat{V}_\phi^\pi(\mathbf{s}_{i,t})\right)$

+ lower variance (due to critic)
- not unbiased (if the critic is not perfect)

Policy gradient: $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\left(\left(\sum_{t'=t}^{T}\gamma^{t'-t} r(\mathbf{s}_{i,t'},\mathbf{a}_{i,t'})\right) - b\right)$

+ no bias
- higher variance (because single-sample estimate)

图 14: 演员-评论家算法 vs. 策略梯度法

演员-评论家算法有着较低的方差, 但估计时并非无偏; 策略梯度法虽然方差较高, 但估计是无偏的. 为了将这两种算法的优势整合到一起, 我们可以使用评论家函数 (值函数) 作为基准线. 则整合后的目标函数如下:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\left(\sum_{t'=t}^{T}\gamma^{t'-t} r(s_{i,t'},a_{i,t'})\right) - \hat{V}_\phi^\pi(s_{i,t})\right)\right)$$

既无偏, 又低方差 (因为基准线更接近于奖励).

9 与动作有关的基准线

我们还可以从动作的的角度去考虑基准线. 优势函数的基本定义为: $A^\pi(s_t,a_t) = Q^\pi(s_t,a_t) - V^\pi(s_t)$, 则我们可以这么估计优势函数, 如图15所示:

$\hat{A}^\pi(\mathbf{s}_t,\mathbf{a}_t) = \sum_{t'=t}^{\infty}\gamma^{t'-t} r(\mathbf{s}_{t'},\mathbf{a}_{t'}) - V_\phi^\pi(\mathbf{s}_t)$

+ no bias
- higher variance (because single-sample estimate)

$\hat{A}^\pi(\mathbf{s}_t,\mathbf{a}_t) = \sum_{t'=t}^{\infty}\gamma^{t'-t} r(\mathbf{s}_{t'},\mathbf{a}_{t'}) - Q_\phi^\pi(\mathbf{s}_t,\mathbf{a}_t)$

+ goes to zero in expectation if critic is correct!
- not correct

图 15: 估计优势函数的两种方法

注意在使用 Q 函数的估计中, 估计的值是错误的, 这是因为我们需要补偿一个误差项. 标准的基准线在期望中可以积分至 0, 但与动作有关的基准线不再能积分至 0, 而是积分至一个误差项, 我们需要额外添加一个修正项使其无偏. 我们可以将这两种估计方法进行

组合:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\hat{Q}_{i,t} - Q_\phi^\pi(s_{i,t}, a_{i,t})\right)\right)$$
$$+\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\nabla_\theta E_{a\sim\pi_\theta(a_t|s_{i,t})}\left[Q_\phi^\pi(s_{i,t}, a_t)\right]\right)$$

第二项为修正项, 这样如果基准线与动作无关, 则第二项为 0.

10 方差截断

如图16所示, 从某个状态出发, 不同的轨迹随着时间偏差越来越大, 这也是一个从小方差到大方差的过程, 我们可以在方差变大之前, 将轨迹截断, 只求第 n 步之前的优势函数:

$$\hat{A}_n^\pi(s_t, a_t) = \sum_{t'=t}^{t+n}\gamma^{t'-t}r(s_{t'}, a_{t'}) - \hat{V}_\phi^\pi(s_t) + \gamma^n\hat{V}_\phi^\pi(s_{t+n})$$

这里 $n > 1$ 通常是个更好的选择.

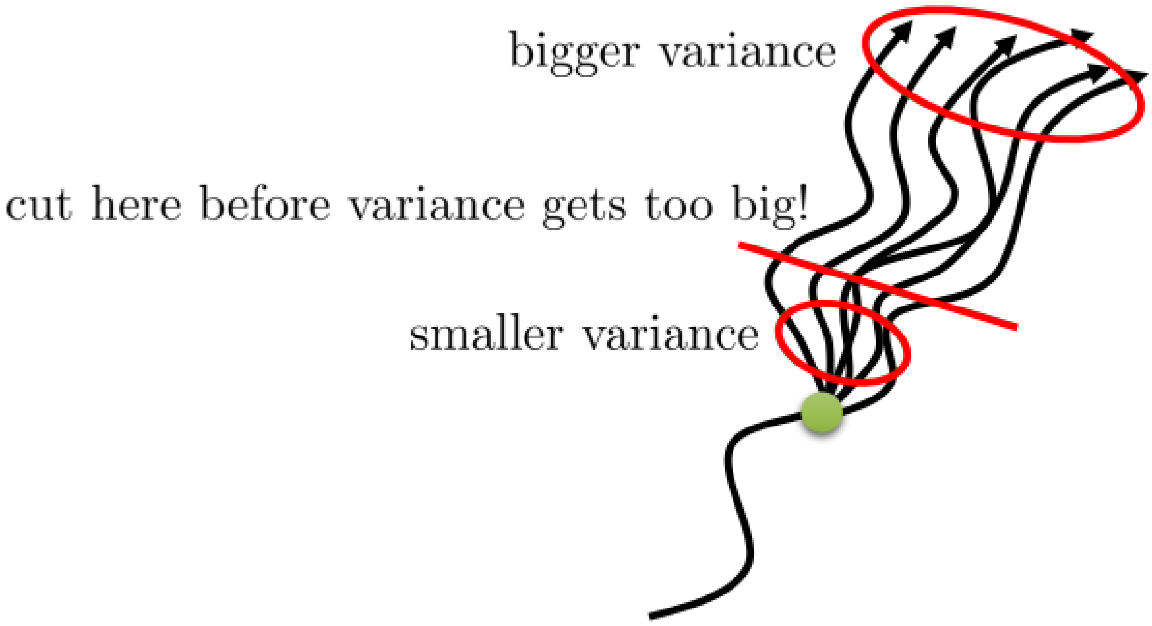

图 16: 方差截断

11 广义优势函数估计

广义优势函数估计 (Generalized advantage estimate) 本质上为 n 步截断方法的推广, 该方法不再选择一个单一的截断步数, 而是对所有的步数进行一个加权平均, 即:

$$\hat{A}_{\text{GAE}}^\pi(s_t, a_t) = \sum_{n=1}^{\infty} w_n \hat{A}_n^\pi(s_t, a_t)$$

其中 w_n 为权重. 由于希望降低方差, 因此一种比较合理的方式是让权重也指数降低, 即 $w_n \propto \lambda^{n-1}$, 其中 $\lambda \in (0,1)$ 是一个底数参数, 权重相加和为 1. 代入后我们可得:

$$\hat{A}^{\pi}_{\text{GAE}}(s_t, a_t) = \sum_{n=1}^{\infty} (\gamma\lambda)^{t'-t} \delta_{t'}$$

$$\delta_{t'} = r(s_{t'}, a_{t'}) + \gamma \hat{V}^{\pi}_{\phi}(s_{t'+1}) - \hat{V}^{\pi}_{\phi}(s_{t'})$$

其中 λ 很像折扣因子, 如果它比较小, 那么我们会在早期进行截断, 则偏差大方差小.

12 参考

原课程链接: https://rail.eecs.berkeley.edu/deeprlcourse/

文字版本翻译: https://zhuanlan.zhihu.com/p/45368769