近端策略优化（Proximal Policy Optimization）

一、从同策略到异策略

- 采样
  - 对于一个随机变量，我们通常用概率密度函数来刻画该变量的概率分布特性。具体来说，给定随机变量的一个取值，可以根据概率密度函数来计算该值对应的概率（密度）。反过来，也可以根据概率密度函数提供的概率分布信息来生成随机变量的一个取值，这就是采样
  - 因此，从某种意义上来说，采样是概率密度函数的逆向应用
  - 与根据概率密度函数计算样本点对应的概率值不同，采样过程往往没有那么直接，通常需要根据待采样分布的具体特点来选择合适的采样策略
- 同策略
  - 如果要学习的智能体和环境的交互的智能体策略是相同的，那么就是同策略
- 异策略
  - 如果要学习的智能体和环境的交互的智能体策略是相同的，那么就是异策略
  - 为什么需要呢?
    - 策略梯度方法是同策略方法
    - PPO 是 PO 的变体，也是 OpenAI 默认的强化学习算法
    - PO 需要很多时间去采样数据，大多数时间都在采样数据，所以使用 PPO 去近似 PO，优化计算，近似平滑
  - 做法假设
    - 人可以从别人的经验中学习，模型也可以
    - 概述
      - 可以用另外一个策略 $\pi\theta'$ 、另外一个演员 θ' 与环境交互（θ θ' 被固定了）
      - 用 θ' 采样到的数据去训练 θ
      - 假设我们可以用 θ' 采样到的数据去训练 θ，我们可以多次使用 θ' 采样到的数据，可以多次执行梯度上升（gradient ascent），可以多次更新参数，都只需要用同一批数据
      - 因为假设 θ 有能力学习另外一个演员 θ' 所采样的数据，所以 θ' 只需采样一次，并采样多一点的数据，让 θ 去更新很多次，这样就会比较有效率
  - 具体做法
    -

假设我们有一个函数 $f(x)$，要计算从分布 p 采样 x，再把 x 代入 f，得到 $f(x)$。我们该怎么计算 $f(x)$ 的期望值呢？假设我们不能对分布 p 做积分，但可以从分布 p 采样一些数据 x^i。把 x^i 代入 $f(x)$，取它的平均值，就可以近似 $f(x)$ 的期望值。

现在有另外一个问题，假设我们不能从分布 p 采样数据，只能从另外一个分布 q 采样数据 x，q 可以是任何分布。如果我们从 q 采样 x^i，就不能使用式 (5.2)。因为式 (5.2) 是假设 x 都是从 p 采样出来的。

$$\mathbb{E}_{x\sim p}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x^i) \tag{5.2}$$

所以我们做一个修正，期望值 $\mathbb{E}_{x\sim p}[f(x)]$ 就是 $\int f(x)p(x)\mathrm{d}x$，我们对其做如下的变换：

$$\int f(x)p(x)\mathrm{d}x = \int f(x)\frac{p(x)}{q(x)}q(x)\mathrm{d}x = \mathbb{E}_{x\sim q}[f(x)\frac{p(x)}{q(x)}] \tag{5.3}$$

就可得

$$\mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right] \tag{5.4}$$

-

因为是从 q 采样数据，所以我们从 q 采样出来的每一笔数据，都需要乘一个**重要性权重（importance weight）** $\frac{p(x)}{q(x)}$ 来修正这两个分布的差异。$q(x)$ 可以是任何分布，唯一的限制就是 $q(x)$ 的概率是 0 的时候，$p(x)$ 的概率不为 0，不然会没有定义。假设 $q(x)$ 的概率是 0 的时候，$p(x)$ 的概率也都是 0，$p(x)$ 除以 $q(x)$ 是有定义的。所以这个时候我们就可以使用重要性采样，把从 p 采样换成从 q 采样。

重要性采样有一些问题。虽然我们可以把 p 换成任何的 q。但是在实现上，p 和 q 的差距不能太大。差距太大，会有一些问题。比如，虽然式 (5.4) 成立（式 (5.4) 左边是 $f(x)$ 的期望值，它的分布是 p，式 (5.4) 右边是 $f(x)\frac{p(x)}{q(x)}$ 的期望值，它的分布是 q），但如果不是计算期望值，而是计算方差，$\mathrm{Var}_{x\sim p}[f(x)]$ 和 $\mathrm{Var}_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right]$ 是不一样的。两个随机变量的平均值相同，并不代表它们的方差相同。

我们可以将 $f(x)$ 和 $f(x)\frac{p(x)}{q(x)}$ 代入方差的公式 $\mathrm{Var}[X] = E[X^2] - (E[X])^2$，可得

$$\mathrm{Var}_{x\sim p}[f(x)] = \mathbb{E}_{x\sim p}\left[f(x)^2\right] - \left(\mathbb{E}_{x\sim p}[f(x)]\right)^2 \tag{5.5}$$

$$\begin{aligned}\mathrm{Var}_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right] &= \mathbb{E}_{x\sim q}\left[\left(f(x)\frac{p(x)}{q(x)}\right)^2\right] - \left(\mathbb{E}_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right]\right)^2 \\ &= \mathbb{E}_{x\sim p}\left[f(x)^2\frac{p(x)}{q(x)}\right] - \left(\mathbb{E}_{x\sim p}[f(x)]\right)^2\end{aligned} \tag{5.6}$$

-

现在要做的就是把重要性采样用在异策略的情况中，把同策略训练的算法改成异策略训练的算法。怎么改呢？如式 (5.7) 所示，之前我们用策略 π_θ 与环境交互，采样出轨迹 τ，计算 $R(\tau)\nabla \log p_\theta(\tau)$。现在我们不用 θ 与环境交互，假设有另外一个策略 π'_θ，它就是另外一个演员，它的工作是做示范（demonstration）。

$$\nabla \bar{R}_\theta = \mathbb{E}_{\tau\sim p_{\theta'(\tau)}}\left[\frac{p_\theta(\tau)}{p_{\theta'}(\tau)}R(\tau)\nabla \log p_\theta(\tau)\right] \tag{5.7}$$

θ' 的工作是为 θ 做示范。它与环境交互，告诉 θ 它与环境交互会发生什么事，借此来训练 θ。我们要训练的是 θ，θ' 只负责做示范，负责与环境交互。我们现在的 τ 是从 θ' 采样出来的，是用 θ' 与环境交互。所以采样出来的 τ 是从 θ' 采样出来的，这两个分布不一样。但没有关系，假设我们本来是从 p 采样，

- 实际做 PO 梯度的时候，我们会把序列的概率，细化到每个 (s, a) pair 对上去进行优化，从而给不同的动作配分
  - $\mathbb{E}_{(s_t,a_t)\sim\pi_\theta}\left[A^\theta(s_t,a_t)\nabla \log p_\theta(a_t^n|s_t^n)\right]$

$$\mathbb{E}_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(s_t,a_t)}{p_{\theta'}(s_t,a_t)}A^\theta(s_t,a_t)\nabla\log p_\theta(a_t^n|s_t^n)\right] \tag{5.9}$$

其中，$A^\theta(s_t,a_t)$ 有一个上标 θ，θ 代表 $A^\theta(s_t,a_t)$ 是演员 θ 与环境交互的时候计算出来的。但是实际上从 θ 换到 θ' 的时候，$A^\theta(s_t,a_t)$ 应该改成 $A^{\theta'}(s_t,a_t)$，为什么呢？$A(s_t,a_t)$ 这一项是想要估测在某一个状态采取某一个动作，接下来会得到累积奖励的值减去基线的值。我们怎么估计 $A(s_t,a_t)$？我们在状态 s_t 采取动作 a_t，接下来会得到的奖励的总和，再减去基线就是 $A(s_t,a_t)$。之前是 θ 与环境交互，所以我们观察到的是 θ 可以得到的奖励。但现在是 θ' 与环境交互，所以我们得到的这个优势是根据 θ' 所估计出来的优势。但我们现在先不要管那么多，就假设 $A^\theta(s_t,a_t)$ 和 $A^{\theta'}(s_t,a_t)$ 可能是差不多的。

接下来，我们可以拆解 $p_\theta(s_t,a_t)$ 和 $p_{\theta'}(s_t,a_t)$，即

$$\begin{aligned}p_\theta(s_t,a_t)&=p_\theta(a_t|s_t)p_\theta(s_t)\\p_{\theta'}(s_t,a_t)&=p_{\theta'}(a_t|s_t)p_{\theta'}(s_t)\end{aligned} \tag{5.10}$$

于是我们可得

$$\mathbb{E}_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}\frac{p_\theta(s_t)}{p_{\theta'}(s_t)}A^{\theta'}(s_t,a_t)\nabla\log p_\theta(a_t^n|s_t^n)\right] \tag{5.11}$$

这里需要做的一件事情是，假设模型是 θ 的时候，我们看到 s_t 的概率，与模型是 θ' 的时候，我们看到 s_t 的概率是一样的，即 $p_\theta(s_t)=p_{\theta'}(s_t)$。因为 $p_\theta(s_t)$ 和 $p_{\theta'}(s_t)$ 是一样的，所以我们可得

$$\mathbb{E}_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}A^{\theta'}(s_t,a_t)\nabla\log p_\theta(a_t^n|s_t^n)\right] \tag{5.12}$$

Q：为什么我们可以假设 $p_\theta(s_t)$ 和 $p_{\theta'}(s_t)$ 是一样的？

A：因为我们会看到状态往往与采取的动作是没有太大的关系的。比如我们玩不同的雅达利游戏，其实看到的游戏画面都是差不多的，所以也许不同的 θ 对 s_t 是没有影响的。但更直接的理由就是 $p_\theta(s_t)$ 很难算，$p_\theta(s_t)$ 有一个参数 θ，它表示的是我们用 θ 去与环境交互，计算 s_t 出现的概率，而这个概率很难算。尤其是如果输入的是图片，同样的 s_t 可能根本就不会出现第二次。我们根本没有办法估计 $p_\theta(s_t)$，所以干脆就无视这个问题。

但是 $p_\theta(a_t|s_t)$ 很好算，我们有参数 θ ，它就是一个策略网络。我们输入状态 s_t 到策略网络中，它会输出每一个 a_t 的概率。所以我们只要知道 θ 和 θ' 的参数就可以计算 $\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}$。

式 (5.12) 是梯度，我们可以从梯度反推原来的目标函数：

$$\nabla f(x)=f(x)\nabla\log f(x) \tag{5.13}$$

注意，对 θ 求梯度时，$p_{\theta'}(a_t|s_t)$ 和 $A^{\theta'}(s_t,a_t)$ 都是常数。

所以实际上，当我们使用重要性采样的时候，要去优化的目标函数为

$$J^{\theta'}(\theta)=\mathbb{E}_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}A^{\theta'}(s_t,a_t)\right] \tag{5.14}$$

二、近端策略优化

基本款

- 我们可以把同策略换成异策略，但前提是两个分布相差没有那么多，PPO 就是为了解决这一点
- **注意，由于在 PPO 中 *θ'* 是 *θ* old，即行为策略也是 *πθ*，因此 PPO 是同策略的算法**
- 在做重要性采样的时候，*pθ* (*at*|*st*) 不能与 *pθ′* (at|st) 相差太多。做示范的模型不能与真正的模型相差太多，相差太多，重要性采样的结果就会不好。我们在训练的时候，应多加一个约束（constrain）。这个约束是 *θ* 与 *θ′* 输出的动作的 KL 散度（KL divergence），这一项用于衡量 *θ* 与 *θ′* 的相似程度

- $$J_{\mathrm{PPO}}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta \mathrm{KL}(\theta, \theta')$$
  $$J^{\theta'}(\theta) = \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t \mid s_t)}{p_{\theta'}(a_t \mid s_t)} A^{\theta'}(s_t, a_t)\right]$$

  PPO 有一个前身：**信任区域策略优化（trust region policy optimization，TRPO）**。TRPO 可表示为

  $$J_{\mathrm{TRPO}}^{\theta'}(\theta) = \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t, a_t)\right], \mathrm{KL}(\theta, \theta') < \delta \tag{5.16}$$

- TRPO 与 PPO 不一样的地方是约束所在的位置不一样，PPO 直接把约束放到要优化的式子里面，我们就可以用梯度上升的方法去最大化式 (5.16)。但 TRPO 是把 KL 散度当作约束，它希望 θ 与 θ' 的 KL 散度小于 δ。如果我们使用的是基于梯度的优化，有约束是很难处理的。TRPO 是很难处理的，因为它把 KL 散度约束当作一个额外的约束，没有放在目标（objective）里面，所以它很难计算。因此我们一般就使用 PPO，而不使用 TRPO 。PPO 与 TRPO 的性能差不多，但 PPO 在实现上比 TRPO 容易得多。

改进款

- 近端策略优化惩罚（PPO-penalty）
  - **(adaptive KL penalty)**。我们可以总结一下自适应 KL 惩罚：如果 $\mathrm{KL}(\theta, \theta^k) > \mathrm{KL}_{\max}$，增大 β；如果 $\mathrm{KL}(\theta, \theta^k) < \mathrm{KL}_{\min}$，减小 β。

    近端策略优化惩罚可表示为

    $$J_{\mathrm{PPO}}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta \mathrm{KL}(\theta, \theta^k)$$
    $$J^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \frac{p_\theta(a_t \mid s_t)}{p_{\theta^k}(a_t \mid s_t)} A^{\theta^k}(s_t, a_t) \tag{5.18}$$

- 近端策略优化裁剪（PPO-clip）
  - 如果我们觉得计算 KL 散度很复杂，那么还有一个 **PPO2** 算法，PPO2 即**近端策略优化裁剪**。近端策略优化裁剪的目标函数里面没有 KL 散度，其要最大化的目标函数为

    $$J_{\mathrm{PPO2}}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \min\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t),\ \mathrm{clip}\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right) A^{\theta^k}(s_t, a_t)\right) \tag{5.19}$$

    其中，
    - 操作符（operator）min 是在第一项与第二项里面选择比较小的项。
    - 第二项前面有一个裁剪（clip）函数，裁剪函数是指，在括号里面有 3 项，如果第一项小于第二项，那就输出 $1-\varepsilon$；第一项如果大于第三项，那就输出 $1+\varepsilon$。
    - ε 是一个超参数，是我们要调整的，可以设置成 0.1 或 0.2 。