Survey on LLM Policy Gradient

董林康

2025 年 5 月 7 日

目录

Chapter 1

Introduction

最近参考了一个复旦同学的博客 (Liu, 2025)，对 LLM 的 Policy Gradient 进行了简单的介绍，并对其在当前主流的 LLM 的 policy gradient 的方法上提供了一个 unified 的视角，觉得很有意思，但是缺点是其中的很多地方都没有给 reference，导致忽略了一些重要的工作的推导细节，这里打算自己也写一遍，介绍一些前置知识在这里，顺便可以之后作为毕设的绪论/方法/附录部分了。

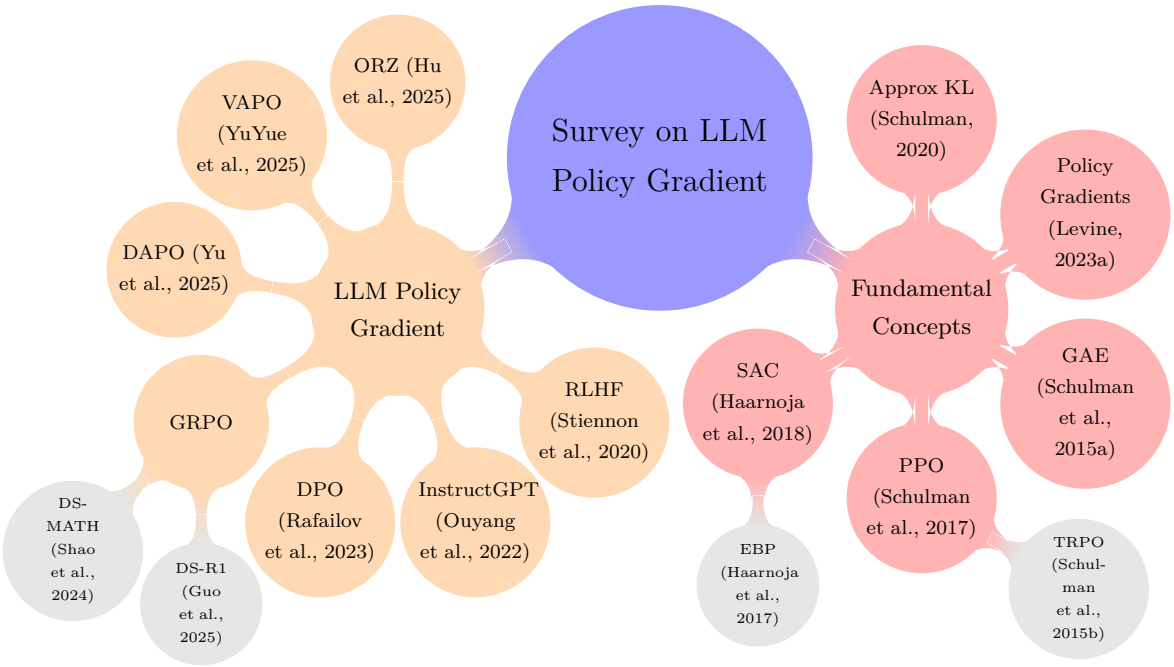

图 1.1: Mindmap of Survey on LLM Policy Gradient

Chapter 2

Preliminary

2.1 Approximating Kullback-Leibler divergence

TL;DR: 用 MC approximation 来计算 KL 散度，使用不同的 estimator 来减少方差。

2.1.1 KL Divergence

原本的 KL 散度 (Kullback-Leibler divergence) 定义为：

$$D_{\mathrm{KL}}(P||Q) = \int p(x) \log \frac{p(x)}{q(x)} dx \tag{2.1}$$

其中 P 和 Q 是两个概率分布，$p(x)$ 和 $q(x)$ 分别是它们的概率密度函数。

2.1.2 Approximation

这里将简单介绍一下的 MC 近似计算 KL 的方法 (Schulman, 2020)。
如果使用 Monte Carlo 方法来近似计算 KL 散度，可以通过以下步骤实现：

$$\mathrm{KL}[q,p] = \sum_x q(x) \log \frac{q(x)}{p(x)} = E_{x\sim q}\left[\log \frac{q(x)}{p(x)}\right] \approx \frac{1}{N}\sum_{i=1}^{N} \log \frac{q(x_i)}{p(x_i)}$$

我们称 $\log \frac{q(x)}{p(x)}$ 为一个 estimator，那么我们得到了一个 naive estimator：

$$k_1 = \log \frac{q(x)}{p(x)} = -\log r$$

其中 ratio r 的定义如下 (这种格式的 ratio 在 subsequent calculation 中会经常出现，比如 importance sampling 等，通常是新的 policy/旧的 policy)：

$$r \doteq \frac{p(x)}{q(x)}$$

那么我们有

$$\mathrm{KL}[q,p] = E_q[k_1] = E_q\left[-\log r\right]$$

对于这个 naive estimator r，我们容易知道当 p 和 q 两个分布差异比较大时（就比如 mean 差的比较多），这个 KL 散度用 MC 的方法固然可以估计出来，但是这个估计的方差就会很大（样本之间的波动会很厉害），这就导致了我们需要的样本数量大大增加。

(Schulman, 2020) 提出了另外两个 estimator 来希望减少 std，并做了简单实验来证明。

$$k_2 = \frac{1}{2}\left(\log p(x) - \log q(x)\right)^2 = \frac{1}{2}(\log r)^2 \qquad \text{(biased, but lower variance empirically)}$$

$$k_3 = (r-1) - \log r \qquad \text{(unbiased, lower variance)}$$

对于 k_2，是 biased，因为它的 expectation 是 f-divergence $\mathrm{KL}_f[p,q]$

$$\mathrm{KL}_f[p,q] = E_q\left[k_2\right] = E_q\left[\frac{1}{2}(\log r)^2\right] \neq \mathrm{KL}[p,q]$$

但是直觉告诉我们这种形式的 estimator 会有相对小一点的方差（因为很对称而且每个样本都是正数），我们会在下面简单实验证明这一点。

对于 k_3，是 unbiased 的

$$\begin{aligned} E_q\left[k_3\right] &= E_q\left[(r-1) - \log r\right] \\ &= E_q\left[r\right] - 1 - E_q\left[\log r\right] \\ &= E_q\left[-\log r\right] \end{aligned}$$

因为

$$E_q\left[r\right] - 1 = E_q\left[\frac{p(x)}{q(x)}\right] - 1 = \sum_x q(x)\frac{p(x)}{q(x)} - 1 = 0$$

那么对于 k_3 这种形式的 KL 其实有两个（不是对称的），对于 $x \sim q, r = \frac{p(x)}{q(x)}$

$$\mathrm{KL}[q,p] = \sum_x q(x)\log\frac{q(x)}{p(x)} = E_q[-\log r] \xrightarrow{\text{unbiased}} E_q[(r-1) - \log r] \tag{2.2}$$

$$\mathrm{KL}[p,q] = \sum_x q(x)\frac{p(x)}{q(x)}\log\frac{p(x)}{q(x)} = E_q[r\log r] \xrightarrow{\text{unbiased}} E_q[r\log r - (r-1)] \tag{2.3}$$

2.1.3 Experiment

这里我们参考了 (Schulman, 2020) 的实验，使用了两个不同均值的正态分布来生成数据，分别是 $p \sim N(0,1)$ 和 $q \sim N(1,1)$，然后使用不同的 estimator 来计算 KL 散度。

可以从上图文字框第二列看到，k_1 的相对方差最大，而 k_2 和 k_3 的相对方差相对较小；此外我们也可以看到第一列中 k_2 的相对偏差是最大的，而 k_1 和 k_3 的相对偏差是相对较小的。这个结果和我们之前的分析是一致的，k_1 和 k_3 是 unbiased 的，而 k_2 是 biased 的，k_3 是一个相对比较好的 estimator。

2.1.4 Commentary

降低方差的 core idea

这里的主要思想是把最初始的 KL 散度 $E_q[-\log r]$ 变成了 $E_q[f(r)]$，接着希望通过选择一个合适性质的函数 f 来降低方差。

首先对 r 的定义域进行思考，r 是一个 ratio，通常是两个分布的比值，所以它的定义域是 $[0,+\infty)$。又由于我们后面一系列的操作都是希望两个分布的差别不会太大，比如 (Schulman et al., 2017) 就通过 clipping 来限制 ratio 的范围在 $[1-\eta, 1+\eta]$，所以我们可以从 $r=1$ 的附近范围考虑选择 f 的问题：

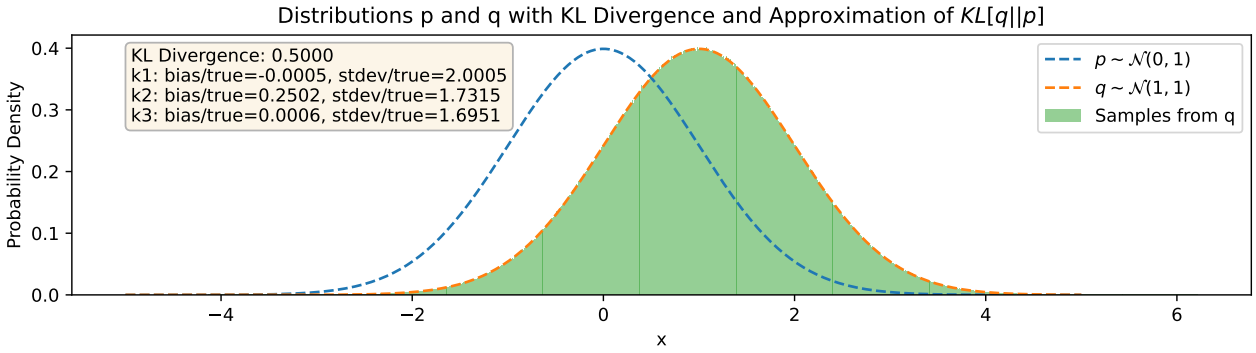

图 2.1: KL 散度的估计结果

a. $f(r) = -\log r$，此时 f 虽然是 convex 的，但是显然 $r = 1$ 附近时的样本有正数也有负数，$f(r)$ 此时是穿过 x 轴的，这样直观看起来方差会比较大；

b. $f(r) = (r-1) - \log r$，此时 f 是 convex 的，并且在 $r = 1$ 时有最小值 0，且 $f(r)$ 此时在 $r = 1$ 的切线就是 $y = 0$，r=1 附近的样本都是接近 0 的正数，$f(r)$ 的方差会小很多；

c. $f(r) = \frac{1}{2}(\log r)^2$，此时 f 是 convex 的，并且在 $r = 1$ 时有最小值 0，其函数性质和上面类似，但是回到 expectation 上面，我们发现他已经不是 unbiased 了。

为什么要计算 KL 散度？

在强化学习中，KL 散度通常用于衡量两个策略之间的差异。通过一定的 KL 散度的约束（惩罚），可以使得新策略更接近于旧策略，从而实现策略的平滑更新。也即之后我们会经常看到优化的目标函数中会有一个 KL 散度的项，比如 PPO (Schulman et al., 2017) 中 KL penalty 项：

$$\mathrm{KL}[\pi_{\theta_{old}}, \pi_\theta] = E_{a\sim\pi_{\theta_{old}}(s)}\left[-\log \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}\right]$$

其中 π_θ 是当前的策略，$\pi_{\theta_{old}}$ 是旧的策略。通过对 KL 散度进行约束（通常是一个以 β ratio 加在 objective function 上面的形式来约束的，在后面章节提及(3.2)），可以使得新策略在更新时不会偏离旧策略太远，从而实现更稳定的学习过程。

这里的 KL penalty 项在 GRPO (Shao et al., 2024) 中被记为 $\mathrm{KL}[\pi_\theta, \pi_{\mathrm{ref}}]$，这个 KL 散度不是用来约束 RL 微调过程中的新旧参数变化（GRPO 采用的是 PPO 中的 clip 来约束），而是用来约束 RL 训练出来的模型不要偏离 ref 模型太远；在 R1 (Guo et al., 2025) 中 ref 模型可以是预训练的模型，也可以是 CoT-SFT 过后的模型。这种 pretrain+SFT+RL(with KL penalty) 的做法在 (Stiennon et al., 2020) 中也有体现，目的都是希望这种方式训练出来的 π_{RL} 不要偏离 π_{SFT} 太远；而使用 KL control 去保守 RL finetune 的做法在 2017 的工作 (Jaques et al., 2017) 上就已经有了。

2.2 Policy Gradients

TL;DR: 这里将参考 (Levine, 2023a) 中的内容，来介绍一下 policy gradient 的推导过程，需要注意的是这里的推导过程和 (Sutton and Barto, 2020) 有细微的差别，不过也比较好理解。

2.2.1 The goal of RL

首先明确强化学习的目标 goal

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

其中 $\tau = (s_1, a_1, \ldots, s_T, a_T)$ 是一个 trajectory，$r(s_t, a_t)$ 是一个 reward function，T 表示到达了 terminal 所经历的步数，$p_\theta(\tau)$ 是一个概率分布，表示在参数 θ 下，trajectory 的分布

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

注：这里的 trajectory 很简洁，只包括 visits(state-action pairs)，很多地方也会写成是 (state, action, reward) 的形式，另外这里的 reward 是输入 state, action 就可以，还有的地方是输入 state, action, next state 的形式 (Schulman et al., 2015a)，其实都是一样的。

其中两种常用的表示方式，一种是无限的 horizon，一种是我们假设有终点的 finite hozizon:

$$\theta_\star = \begin{cases} \arg\max_\theta \mathbb{E}_{(s,a)\sim p_\theta(s,a)}\left[r(s,a)\right] & \text{infinite horizon case} \\ \arg\max_\theta \sum_{t=1}^{T}\mathbb{E}_{(s,a)\sim p_\theta(s,a)}\left[r(s_t,a_t)\right] & \text{finite horizon case} \end{cases}$$

下面我们来看 objective，我们使用 MC 的方法，从 p_θ 中采样出 N 个 trajectory，记为 $\tau_i, i = 1, \ldots, N$，然后我们可以得到一个近似的 objective function

$$J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_t r(s_t, a_t)\right] \approx \frac{1}{N}\sum_i\sum_t r(s_{i,t}, a_{i,t}) \tag{2.4}$$

令

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

则有

$$J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[r(\tau)\right] = \int p_\theta(\tau)r(\tau)d\tau$$

注：这个记法其实和 $\sum_\tau p_\theta(\tau)r(\tau)$ 是一样的，只不过这里是连续的分布，所以用积分来表示。

2.2.2 Direct policy differentiation

那么我们可以直接求梯度得

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

注：第二个等号成立是因为一个简单的 identity:

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

对于 $\nabla_\theta \log p_\theta(\tau)$ 我们有

$$\begin{aligned}\nabla_\theta \log p_\theta(\tau) &= \nabla_\theta \log \left(p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \right) \\ &= \underbrace{\nabla_\theta \log p(s_1)}_{0} + \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) + \sum_{t=1}^{T} \underbrace{\nabla_\theta \log p(s_{t+1}|s_t, a_t)}_{0} \\ &= \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\end{aligned}$$

那么此时的 policy gradient 就可以写成

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log p_\theta(\tau) r(\tau) \right] \\ &= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left(\sum_{t=1}^{T} r(s_t, a_t) \right) \right]\end{aligned}$$

那么结合(2.4)，我们可以得到

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

Algorithm 1 REINFORCE

1: Sample τ_i from $\pi_\theta(a_t|s_t)$, run the policy
2: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right)$
3: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

2.2.3 Understanding Policy Gradient

我们可以拿 policy gradient 和对 π_θ 求的 maximum likelihood 来做对比一下：

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

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

在这里令

$$\nabla_\theta \log \pi_\theta(\tau_i) = \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})$$

则有

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i) r(\tau_i) \quad \nabla_\theta J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i)$$

我们做的事情就其实是增加好的轨迹的概率，减少坏的轨迹的概率，相当于形式化了试错的概念。而且在这里我们基本都没用到 markov 的性质（可以在 partial observed MDPs 中用 policy gradient），只是简单的把 τ 当成一个整体来处理，只是想 maximize 好 τ 的概率，这个形式其实和 cross entropy 非常像了，按理来说我们是比较希望奖励是可以 normalize 一下的，相当于就是人类设计的一个奖励分布希望模型学进去。

2.2.4 Variance Reduction

TL;DR: 这里主要介绍两种减少方差的方法：causality 和 optimal baseline，为后面的减少策略梯度的方差 (Schulman et al., 2017, 2015b)，减少优势函数的方差Schulman et al. (2015a) 做铺垫。

$$\nabla_\theta J_{\mathrm{PG}}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t},a_{i,t})\right) \tag{2.5}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t_1=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t_1}|s_{i,t_1})\sum_{t_2=1}^{T} r(s_{i,t_2},a_{i,t_2})\right) \tag{2.6}$$

括号中两个对时间维度的求和项的方差其实是很大的，特别是对于特别长的 trajectory 来说，最直观减少方差的方法我们的思路是考虑因果性 causality，即未来的 policy 是不能影响到当前获得的 reward 的（根据这一点我们减少了起码一半的方差），那么此时就可以修改上面式子为

$$\nabla_\theta J_{\mathrm{PG}}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t_1=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t_1}|s_{i,t_1})\sum_{t_2=t_1}^{T} r(s_{i,t_2},a_{i,t_2})\right)$$

一些联系

结合之前在 Sutton 书中的概念，这个求和的 reward 其实就是 undiscounted return G_t，即

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

在 sutton 书中的 policy gradient 是用的 Q 而不是 G_t

$$Q = \mathbb{E}_\pi[G_t|s_t,a_t]$$

相当于对之后可能的结果再做了一次平均，或者可以理解成 sutton 书中考虑的角度是 $\frac{1}{N}\sum_{i=1}^{N}$ 放在后一项结合，而这里的考虑问题的角度是从 batch 搜集样本出发。

做完第一点改进之后我们来做 baseline 的改进，和 (Sutton and Barto, 2020) 13.4 节中的类似，我们可以引入一个 baseline b，使得

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

其中 b 是一个和 τ 无关的常数或者函数就可以得到一个无偏估计，因为

$$\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\nabla_\theta \log p_\theta(\tau) b\right] = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) b d\tau = \int \nabla_\theta p_\theta(\tau) b d\tau = b\nabla_\theta \underbrace{\int p_\theta(\tau)d\tau}_{1} = 0$$

接着我们来推导最优的 baseline，令

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

又已知

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

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\Big[g(\tau)[r(\tau)-b]\Big] = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[g(\tau)r(\tau)\right]$$

所以有

$$\frac{d}{db}\text{Var} = \frac{d}{db}\mathbb{E}\Big[g(\tau)^2[r(\tau)-b]^2\Big] - \underbrace{\frac{d}{db}\mathbb{E}\left[g(\tau)r(\tau)\right]^2}_{0} = -2\left(\mathbb{E}\Big[g(\tau)^2 r(\tau)\Big] - b\mathbb{E}\Big[g(\tau)^2\Big]\right) = 0$$

解得 optimal baseline 为 gradient magnitudes weighted average of the rewards:

$$b = \frac{\mathbb{E}[g(\tau)^2 r(\tau)]}{\mathbb{E}[g(\tau)^2]} \xrightarrow{\text{suboptimal}} b = \mathbb{E}_{\tau\sim p_\theta(\tau)}[r(\tau)] \approx \frac{1}{N}\sum_{i=1}^{N} r(\tau_i)$$

2.2.5 Importance Sampling

我们目前为止所学的 policy gradient 都是 on-policy 的方法，

$$J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[r(\tau)\right]$$

这样遇到的问题是神经网络每次都只能更新一点，不能利用过去的经验（只能从当前的 policy 中采样）导致效率非常低下，解决方法是不从当前策略 π_θ 中采样而是从一个行为策略 $\bar{\pi}$ 中采样，我们的目标函数就变成了

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

其中推导也很简单，就是相当于换一个分布的来计算原来积分的方法，这个方法就叫做 importance sampling，推导如下

$$\begin{aligned}
\mathbb{E}_{x\sim p(x)}\left[f(x)\right] &= \int p(x) f(x) dx \\
&= \int p(x)\frac{q(x)}{q(x)} f(x) dx \\
&= \int q(x)\frac{p(x)}{q(x)} f(x) dx \\
&= \mathbb{E}_{x\sim q(x)}\left[\frac{p(x)}{q(x)} f(x)\right]
\end{aligned}$$

我们可以将这个 ratio $\frac{p_\theta(\tau)}{\bar{p}(\tau)}$ 进一步展开，得到

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

那么我们就得到了 off-policy policy gradient 的形式

$$\nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\nabla_{\theta'}\log p_{\theta'}(\tau) r(\tau)\right] \tag{2.7}$$

$$= \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)}\nabla_{\theta'}\log p_{\theta'}(\tau) r(\tau)\right] \tag{2.8}$$

$$= \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\left(\prod_{t=1}^{T}\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\right)\left(\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(a_t|s_t)\right)\left(\sum_{t=1}^{T} r(s_t,a_t)\right)\right] \tag{2.9}$$

和之前一样我们可以考虑 causality 的影响（推导可以看附录A.1），则可以推出

$$\nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t_1=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(a_{t_1}|s_{t_1})\left(\prod_{t_2=1}^{t_1}\frac{\pi_{\theta'}(a_{t_2}|s_{t_2})}{\pi_\theta(a_{t_2}|s_{t_2})}\right)\left(\sum_{t_3=t_1}^{T} r(s_{t_3},a_{t_3})\left(\prod_{t_4=t_1}^{t_3}\frac{\pi_{\theta'}(a_{t_4}|s_{t_4})}{\pi_\theta(a_{t_4}|s_{t_4})}\right)\right)\right] \tag{2.10}$$

实际上我们往往考虑下面这个式子的一阶形式就可以了

$$\nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t_1=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_{t_1}|s_{t_1}) \left(\prod_{t_2=1}^{t_1} \frac{\pi_{\theta'}(a_{t_2}|s_{t_2})}{\pi_\theta(a_{t_2}|s_{t_2})} \right) \left(\sum_{t_3=t_1}^{T} r(s_{t_3}, a_{t_3}) \right) \right] \tag{2.11}$$

其中 $\prod_{t_2=1}^{t_1} \frac{\pi_{\theta'}(a_{t_2}|s_{t_2})}{\pi_\theta(a_{t_2}|s_{t_2})}$ 对于 T 而言是指数级的增长的，一般只用当前 step 就可以了，再加上我们将这里的 G_{t_1} 替换成 $\hat{Q}$，那么就可以得到

$$\text{on-policy}: \quad \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \hat{Q}_{i,t} \tag{2.12}$$

$$\text{off-policy}: \quad \nabla_{\theta'} J(\theta') \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{\pi_{\theta'}(a_{i,t}|s_{i,t})}{\pi_\theta(a_{i,t}|s_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(a_{i,t}|s_{i,t}) \hat{Q}_{i,t} \tag{2.13}$$

2.2.6 Commentary

PPO (Schulman et al., 2017) 中提出的 objective function

$$L^{\text{PG}}(\theta) = \hat{\mathbb{E}}_t \left[\hat{A}_t \log \pi_\theta(a_t|s_t) \right]$$

这里的 objective function 其实和 (Sutton and Barto, 2020), 13.2 节的推导不太一样。Sutton 的书里推导 policy gradient 时是从 $\nabla_\theta v_\pi$ 开始入手去 derive 的，最后得到的结果是：

$$\begin{aligned}
\nabla J(\theta) &\propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s) \\
&= \mathbb{E}_{s \sim \mu} \left[\sum_a \pi(a|s) q_\pi(s,a) \frac{1}{\pi(a|s)} \nabla \pi(a|s) \right] \\
&= \mathbb{E}_{s \sim \mu, a \sim \pi(\cdot|s)} \left[q_\pi(s,a) \nabla \log \pi(a|s) \right]
\end{aligned}$$

注：这里的 ∇ 和 π 都省略了下标 $_\theta$

可以看到这里的原来的 objective function 是没有 log 的，只是在推导的过程中加了一个 $\frac{1}{\pi(a|s)}$，然后就可以写进 ∇ 里面变成 $\nabla \log \pi(a|s)$ 了。但是参考 CS285 lecture5 (Levine, 2023a) 中的推导可以得知这里的 $\mathbb{E}_t$ 实际上是 trajectory-visit step 所求的期望 (这里的 trajectory 来自于一个 batch, 一个 visit/state-action pair 则来自于 trajectory)，而 Sutton 的推导则是 state-action step 所求的期望 (这里的 state 和 action 都是来自于分布中)，所以两者期望的下标是不同的。

完整的将 CS285 lecture5 (Levine, 2023a) 中的 objective function 写成如下的形式：

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

后续工作做进一步改进，就将 $\hat{Q}_t$ 替换成 $\hat{A}_t$，或者换成 GAE (Schulman et al., 2015a) 中的 estimator；得到这个形式的目标函数其实是从结果导向的，是因为我们的目标是希望求得 $\hat{A}_{i,t} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})$，于是我们便构造这样一个 objective function 来求解，并不是说我们一开始的目标函数就是这个形式；但是我们可以很清晰看出来其代表的数学含义其实就是 cross-entropy。

2.3 Generalized Advantage Estimation

TL;DR: 这个工作简称 GAE (Schulman et al., 2015a)，在后续 PPO(Schulman et al., 2017), Open-Reasoner-Zero (Hu et al., 2025), VAPO (YuYue et al., 2025) 工作中都用到了 GAE 作为 estimator：

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V$$

2.3.1 Preliminaries

这里介绍基本的一些符号，首先从 undiscounted 的情况出发：

Notation	Description
s_0	initial state sample from distribution ρ_0
τ	trajectory, $\tau = (s_0, a_0, s_1, a_1, \ldots)$ generated by sampling actions according to the policy $a_t \sim \pi(a_t\|s_t)$ and sampling the states by the dynamics $s_t \sim P(s_{t+1}\|s_t, a_t)$, until a terminal is reached
r_t	reward at time step t, $r_t = r(s_t, a_t, s_{t+1})$

表 2.1: Notations used in GAE

目标就是最大化总奖励 $\sum_{t=0}^{\infty} r_t$ 的期望，这里假设这个目标函数是有限的，期望是对于所有的 policy 而言的。

策略梯度的方法如下：

$$g := \nabla_\theta \mathbb{E}\left[\sum_{t=0}^{\infty} r_t\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi(a_t|s_t)\right]$$

其中 Ψ_t 的可能取值如下：

Expression of Ψ_t	Description
$\sum_{t=0}^{\infty} r_t$	total reward of the trajectory
$\sum_{t'=t}^{\infty} r'_t$	total reward follwing action a_t
$\sum_{t'=t}^{\infty} r'_t - b(s_t)$	baselined version of previous formula
$Q^\pi(s_t, a_t)$	state-action value function
$A^\pi(s_t, a_t)$	advantage function
$r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$	TD residual

表 2.2: Different expressions of the policy gradient

其中后三项出现的符号定义为：

$$V^\pi(s_t) := \mathbb{E}_{s_{t+1:\infty}, a_{t:\infty}}\left[\sum_{l=0}^{\infty} r_{t+l}\right] \quad Q^\pi(s_t, a_t) := \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}}\left[\sum_{l=0}^{\infty} r_{t+l}\right]$$

$$A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t), \quad \text{(Advantage function)}$$

其中 $\Psi_t = A^\pi(s_t, a_t)$ 在上面是最小的方差。

符号理解

这里的 $a:b$ 的意思表示从 a 到 b 的一个 range $(a, a+1, \ldots, b)$，这里对 value function 和 state-action value function 的定义和之前在 (Sutton and Barto, 2020) 书中的定义是一致的，可能有点难以一眼看出来，下面解释一下，之前我们所学的定义为：

$$G_t = \begin{cases} \sum_{l=0}^{\infty} R_{t+l} & \text{undiscounted return} \\ \sum_{l=0}^{\infty} \gamma^l R_{t+l} & \text{discounted return} \end{cases}$$

$$v_\pi(s) = \mathbb{E}[G_t | S_t = s]$$

$$q_\pi(s,a) = \mathbb{E}[G_t | S_t = s, A_t = a]$$

这里的大写字母都是代表随机变量的意思，求的期望就是对括号内的随机变量所求的，在此基础上我们可以理解 gae 中出现的两个期望的含义：
$\mathbb{E}_{s_{t+1:\infty}, a_{t:\infty}}$ 指对于所有可能的 $(a_t, s_{t+1}, a_{t+1}, \ldots, s_\infty, a_\infty)$ 路径的总 return 求期望，也即是在 s_t 之后的所有可能路径的 return 做期望；
$\mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}}$ 指对于所有可能的 $(s_{t+1}, a_{t+1}, \ldots, s_\infty, a_\infty)$ 路径的总 return 求期望，也即是在 s_t, a_t 之后的所有可能路径的 return 做期望；

GAE 引入了 *gamma* 来减少上面 estimator 的方差，注意这个 γ 和我们所学知识的 discounted MDPs 中的 bellman equation 中的衰减因子有所区别，这里的主要作用是用来减少 undiscounted return 的方差，那么三项的表达式就变成了：

$$V^{\pi,\gamma}(s_t) := \mathbb{E}_{s_{t+1:\infty}, a_{t:\infty}}\left[\sum_{l=0}^{\infty} \gamma^l r_{t+l}\right] \quad Q^{\pi,\gamma}(s_t, a_t) := \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}}\left[\sum_{l=0}^{\infty} \gamma^l r_{t+l}\right]$$

$$A^{\pi,\gamma}(s_t, a_t) := Q^{\pi,\gamma}(s_t, a_t) - V^{\pi,\gamma}(s_t).$$

discounted 下的 policy gradient 的表达式为：

$$g^\gamma := \mathbb{E}_{s_0:\infty, a_0:\infty}\left[\sum_{t=0}^{\infty} A^{\pi,\gamma}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t)\right]$$

γ 的作用

我们之前在学习的 discounted MDPs 中，γ 的作用是让一个 trajectory/episode 后面的 reward 衰减，避免了 return 无穷大的问题，容易得

$$1 + \gamma + \gamma^2 + \ldots = \frac{1}{1-\gamma}$$

e.g., 如果每一步 reward 都是 1 的话, $\gamma = 0.99$，那么我们无限步的 return 就是 100，这相当于 $\gamma = 0.99$ 只考虑了前 100 步的 reward，后面的 reward 都不用考虑了；但是在这里的 undiscounted policy gradient 单纯引入 γ 来减少方差，但是也带来了 bias，即 g^γ 是 biased estimate of undiscounted MDP

文章中提出了一个 γ-just 的概念，其实很简单，就是希望得到的是相对于 g^γ 的无偏估计就可以了，γ-just 的意思就是 γ 的值是可以选择的，仅有 γ 引入了 bias 的意思，γ-just 的充分条件是 estimator 对

于 Q_t 而言是无偏的就可以（这和我们在 Sutton and Barto (2020) 中学到的一致），可以验证下面这些的表达式是 γ-just 的：$\sum_{l=0}^{\infty}\gamma^l r_{t+l}$, $A^{\pi,\gamma}(s_t,a_t)$, $Q^{\pi,\gamma}(s_t,a_t)$, $r_t+\gamma V^{\pi,\gamma}(s_{t+1})-V^{\pi,\gamma}(s_t)$.

2.3.2 Advantage function estimation

这一节就是关于如何制造一个精确的 (无偏且方差小的)estimate $\hat{A}^t$ of discounted advantage function $A^{\pi,\gamma}(s_t,a_t)$，用来当作 policy gradient 的一部分

$$\hat{g}=\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{\infty}\hat{A}_{i,t}\nabla_\theta\log\pi_\theta(a_{i,t}|s_{i,t})$$

其中 N 表示 batch size，i 表示第 i 个 sample，t 表示第 t 步的动作。

假设我们有一个 approximate value function V(我们的价值函数就是一个神经网络)，那么 TD residual 的表达式为：

$$\delta_t^V=r_t+\gamma V(s_{t+1})-V(s_t)$$

只有当 $V=V^{\pi,\gamma}$ 才是 γ-just 的（无偏的），因为

$$\begin{aligned}\mathbb{E}_{s_{t+1}}\left[\delta_t^{V^{\pi,\gamma}}\right]&=\mathbb{E}_{s_{t+1}}\left[r_t+\gamma V^{\pi,\gamma}(s_{t+1})-V^{\pi,\gamma}(s_t)\right]\\&=\mathbb{E}_{s_{t+1}}\left[Q^{\pi,\gamma}(s_t,a_t)-V^{\pi,\gamma}(s_t)\right]=A^{\pi,\gamma}(s_t,a_t).\end{aligned}$$

我们现在知道一个 δ_t^V 可以是一个无偏的优势函数的估计了，那么是不是可以考虑一下将 k 个 δ 项组合相加的 estimate 呢，此时记为 $\hat{A}_t^{(k)}$，比较直观的想法是 (后面说这个想法怎么来的)：

$$\begin{aligned}\hat{A}_t^{(1)}&:=\delta_t^V&&=-V(s_t)+r_t+\gamma V(s_{t+1})\\\hat{A}_t^{(2)}&:=\delta_t^V+\gamma\delta_{t+1}^V&&=-V(s_t)+r_t+\gamma r_{t+1}+\gamma^2V(s_{t+2})\\\hat{A}_t^{(3)}&:=\delta_t^V+\gamma\delta_{t+1}^V+\gamma^2\delta_{t+2}^V&&=-V(s_t)+r_t+\gamma r_{t+1}+\gamma^2r_{t+2}+\gamma^3V(s_{t+3})\\\hat{A}_t^{(k)}&:=\sum_{l=0}^{k-1}\gamma^l\delta_{t+l}^V&&=-V(s_t)+r_t+\gamma r_{t+1}+\ldots+\gamma^{k-1}r_{t+k-1}+\gamma^kV(s_{t+k})\\\hat{A}_t^{\infty}&:=\sum_{l=0}^{\infty}\gamma^l\delta_{t+l}^V&&=-V(s_t)+\sum_{l=0}^{\infty}\gamma^lr_{t+l}\end{aligned}$$

我们的 Generalized Advantage Estimator GAE(γ,λ) 的表达式定义为上面这些 $\hat{A}_t^{(k)}$ 的 exponetially-weighted average:

$$\begin{aligned}\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}&:=(1-\lambda)\left(\hat{A}_t^{(1)}+\lambda\hat{A}_t^{(2)}+\lambda^2\hat{A}_t^{(3)}+\ldots\right)\\&=(1-\lambda)\left(\delta_t^V+\lambda(\delta_t^V+\gamma\delta_{t+1}^V)+\lambda^2(\delta_t^V+\gamma\delta_{t+1}^V+\gamma^2\delta_{t+2}^V)+\ldots\right)\\&=(1-\lambda)\left(\delta_t^V\left(1+\lambda+\lambda^2+\ldots\right)+\gamma\delta_{t+1}^V\left(\lambda+\lambda^2+\ldots\right)+\gamma^2\delta_{t+2}^V\left(\lambda^2+\lambda^3+\ldots\right)+\ldots\right)\\&=(1-\lambda)\left(\delta_t^V\frac{1}{1-\lambda}+\gamma\delta_{t+1}^V\frac{\lambda}{1-\lambda}+\gamma^2\delta_{t+2}^V\frac{\lambda^2}{1-\lambda}+\ldots\right)\\&=\delta_t^V+\gamma\lambda\delta_{t+1}^V+\gamma^2\lambda^2\delta_{t+2}^V+\ldots\\&=\sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V\end{aligned}$$

简单放在一起就是：

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V \tag{2.14}$$

$$\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{2.15}$$

其中有两个特殊情况：

$$\mathrm{GAE}(\gamma, 0) : \hat{A}_t^{(1)} = \delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{2.16}$$

$$\mathrm{GAE}(\gamma, 1) : \hat{A}_t^{\infty} = \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t) \tag{2.17}$$

前一种情况就是 one-step TD(or TD(0))，后者就是 Monte Carlo(or TD(1)) - baseline 的情况。我们容易知道后者是 γ-just 的（无偏的），但是由于这个求和项，会导致我们的估计出现非常大的误差；而前者这种只有一个 r 的情况下，方差是非常小的，但是显然由于 V 是一个 approximate value function 带来了 bias，（当 $V = V^{\pi,\gamma}$ 才是无偏的）：所以这里用 λ 来平衡 bias 和 variance 的 trade-off。

2.3.3 Commentary

可以发现其实 GAE 的提出和 (Sutton and Barto, 2020) 中的 Eligibility Traces 一章的内容相关性非常大的，这里的 GAE(γ, λ) 和 TD(λ) 的概念就是非常类似的。

首先我们有一个 discounted 的 return 的表达式（(Sutton and Barto, 2020) 7.1 n-step TD）：

$$G_{t:t+n} \doteq R_t + \gamma R_{t+1} + \ldots + \gamma^{n-1} R_{t+n-1} + \gamma^n V(s_{t+n})$$

这里的 G 相当于是一个 truncated 的 return，之后的 estimate 直接用 V 来代替，我们将其进行加权平均就得到了 λ-return 的表达式 ((Sutton and Barto, 2020) 12.1 λ-return)：

$$G_t^{\lambda} \doteq (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n}$$

对比这里的 GAE 的定义：

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} \doteq (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} \hat{A}_t^{(n)}$$

减少方差的思想基本是一致，就其实是将 λ-return 的表达式中的 $G_{t:t+n}$ 换成了 $\hat{A}_t^{(n)}$ 一个简单的工作。

理解加权平均

这个权重就是一个 exponential decay 的权重

$$(1-\lambda)\lambda^n = \frac{\lambda^n}{1/(1-\lambda)}$$

加权平均的意思就是我们最后还将其归一化了 ($\sum = 1$):

$$(1-\lambda)\left(1 + \lambda + \lambda^2 + \ldots\right) = (1-\lambda)\frac{1}{1-\lambda} = 1$$

最后说一些论文中的一些细节，第一个细节是作者发现 λ 的取值是要比 γ 要小的，比如 $\gamma = 0.995, \lambda = 0.96$；第二个是关于 reward shaping，GAE 中就这个维度解释了一下 GAE 的做法：本来只有 γ 的时候，模型在学习时是会忘记/不管 $l \gg 1/(1-\gamma)$ 之后的 reward，比如 $\gamma = 0.995$ 就会不太管 200 步之后的内容了，但是这样很明显会有很大的方差；GAE 加入 λ 的加权指数平均之后就减少了 long delay 带来的 noise 了，此时变成了忘记/不管 $l \gg 1/(1-\lambda\gamma)$ 之后的 reward 比如 $\gamma = 0.995, \lambda = 0.96$ 就会不太管 25 步之后的内容了，减少了方差，又一定程度上考虑了 long-term 的 reward。

2.4 Proximal Policy Optimization algorithms

这个工作一般被简称为 PPO (Schulman et al., 2017)，这里还是属于传统强化学习的范畴之中，但是这里的很多概念都非常重要，比如 clip 的思想，GAE 的思想等（TODO 先写完 PPO 再回去写 GAE），这里我们会简单介绍一下 PPO 的工作原理和一些重要的概念，之后会在LLM Policy Gradient中用到这些概念。

2.4.1 Policy gradient methods

回忆起我们强化学习的基础概念策略梯度 (Dong, 2025)，结合 PPO (Schulman et al., 2017) 中所用到的符号，我们给出 gradient estimator 的形式如下：

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right]$$

其中符号表如下：

Notation	Description
s_t	State at time step t.
a_t	Action at time step t.
π_θ	Policy(stochasitic) parameterized by θ.
$\hat{A}_t$	Advantage function estimator at time step t.
$\hat{\mathbb{E}}_t$	Empirical average over a finite batch of samples at time step t.

表 2.3: Notations used in LLM Policy Gradient

Policy gradient 的原来的 objective function 的形式如下 (PPO (Schulman et al., 2017), 2.1 节)：

$$L^{\text{PG}}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t|s_t) \hat{A}_t \right]$$

2.4.2 Trust region methods

TRPO (Schulman et al., 2015b) 工作是在上面的 PG 方法上添加 KL 约束条件和 importance sampling，其原因可以参考 CS285 lecture9 (Levine, 2023b) 的解释，Specifically, the KL divergence is a measure of how much the new policy π_θ differs from the old policy $\pi_{\theta_{old}}$; 这个约束条件的形式如下：

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \\ \text{subject to} \quad & \mathbb{E}_t \left[\text{KL}(\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)) \right] \leq \delta \end{aligned}$$

注：这里的 $\mathrm{KL}(\pi_{\theta_{old}}, \pi_\theta)$ 其实是和这种表述 $D_{KL}(\pi_{\theta_{old}}||\pi_\theta)$ 是等价的

通常也使用这样一种 penalty 的形式来表示约束条件：

$$\underset{\theta}{\text{maximize}} \quad \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t - \beta\mathrm{KL}(\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t))\right]$$

注：这种方式是因为某种替代目标函数（计算状态上的最大 KL 散度，而不是平均值）构成了策略 π 性能的下界（即悲观界限），包括后面的文章提出的 clip 方法也是这样，使得策略更新不能超过这个界限。

原始的 TRPO 是直接使用 hard constraint 来约束，因为 β 这个超参数比较难修改，文章中使用一个 adaptive β 的版本作为 baseline 之一，如下

$$\begin{aligned} L_{\mathrm{KLPEN}}(\theta) &= \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t - \beta\mathrm{KL}(\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t))\right] \\ &= \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right] - \beta\underbrace{\hat{\mathbb{E}}_t\left[\mathrm{KL}(\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t))\right]}_{d} \end{aligned}$$

$$\beta \leftarrow \begin{cases} \beta/2 & \text{if } d < d_{\text{targ}}/1.5 \\ \beta \times 2 & \text{if } d > d_{\text{targ}} \times 1.5 \\ \beta & \text{otherwise} \end{cases}$$

2.4.3 Clipped Surrogate Objective

令

$$r_t(\theta) \doteq \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

那么可知 $r_t(\theta_{old}) = 1$，

TRPO 中的工作是在有约束的情况下 maximize 目标函数：

$$L(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

PPO 中提出的 main 目标函数是：

$$L^{\mathrm{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

从图 2.2 可以看出，红色虚线就是上面目标函数的形式，这里之所以用 min 的形式是因为我们希望在 $\hat{A}_t$ 为正的时候，$r_t(\theta)$ 的值不能超过 $1+\epsilon$，而在 $\hat{A}_t$ 为负的时候，$r_t(\theta)$ 的值不能超过 $1-\epsilon$，这样就可以保证我们不会过度优化策略。

2.4.4 Algorithm

下面是 PPO 中的一些算法细节，最后采用的 objective function 如下：

$$L_t^{\mathrm{CLIP+VF+S}}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{\mathrm{CLIP}}(\theta) - c_1 L_t^{\mathrm{VF}}(\theta) + c_2 S[\pi_\theta](s_t)\right] \tag{2.18}$$

$L_t^{\mathrm{CLIP}}(\theta)$ 就是上面提出的 clipped surrogate objective function；

$L_t^{\mathrm{VF}}(\theta)$ 是 squared-error loss $(V_\theta(s_t) - V_t^{\mathrm{targ}})^2$，$V_\theta(s_t)$ 是 value function，V_t^{targ} 是 target value function；文章的解释是如果 policy 和 value function 共享一个神经网络的参数，那么就需要 combine clipped surrogate objective 和 value function error term;

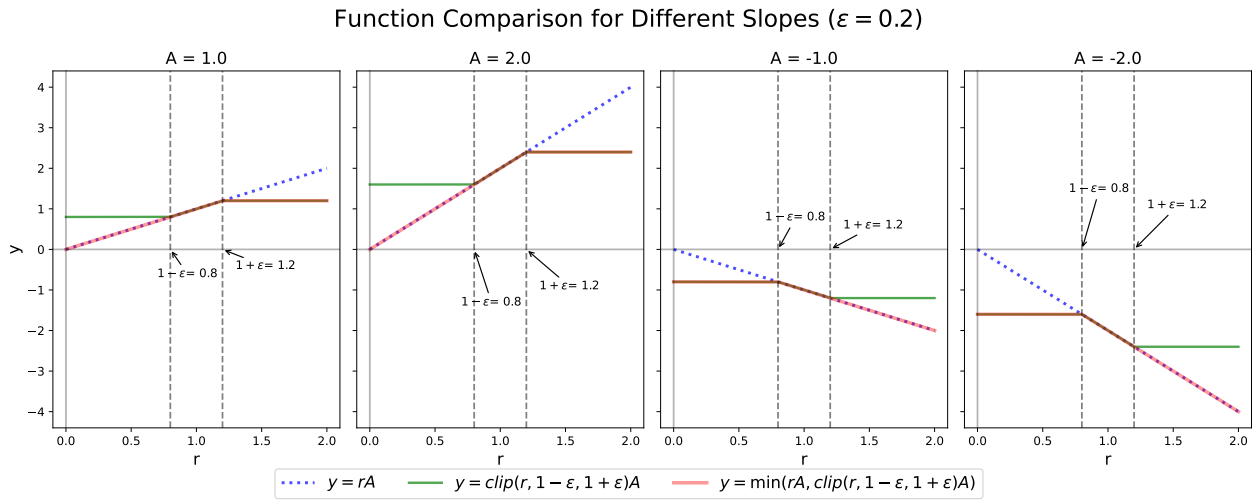

图 2.2: Clipped surrogate objective function

$S[\pi_\theta](s_t)$ 是 entropy bonus，用来保证 sufficient exploration；

c_1 和 c_2 是 coefficients，这两个参数都是大于零的参数。

Entropy bonus

自己的理解是保证每个状态下的动作概率分布的熵尽量大（更加混乱），从而保证积极的探索，这一项最早在 (Williams, 1992) 中就提出了，A3C (Mnih et al., 2016) 中记号为 H，PPO 参考二者中记号为 $S[\pi_\theta](s_t)$（包括之后的 SoftQlearning(Haarnoja et al., 2017) 和 SoftAC(Haarnoja et al., 2018) 记号为 $\mathcal{H}$），其形式如下：

$$S[\pi_\theta](s_t) = \mathcal{H}(\pi_\theta(\cdot|s_t)) = -\sum_a \pi_\theta(a|s_t)\log\pi_\theta(a|s_t)$$

在这部分 PPO 中还提出了一些估计优势函数的技巧（适合 RNN 的做法），就是采取 truncated version of generalized advantage estimator(Schulman et al., 2015a)，即固定长度为 T timesteps 的 truncated GAE，其形式如下：

$$\begin{aligned}\hat{A}_t &= -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T) \\ &= \underbrace{r_t + \gamma V(s_{t+1}) - V(s_t)}_{\delta_t} + \gamma\underbrace{(r_{t+1} + \gamma V(s_{t+2}) - V(s_{t+1}))}_{\delta_{t+1}} + \cdots + \gamma^{T-t-1}\underbrace{(r_{T-1} + \gamma V(s_T) - V(s_{T-1}))}_{\delta_{T-1}} \\ &= \delta_t + \gamma\delta_{t+1} + \cdots + \gamma^{T-t-1}\delta_{T-1} = \sum_{k=0}^{T-t-1} \gamma^k \delta_{t+k}\end{aligned}$$

其中 $t \in [0, T-1]$，T 是 truncated 的长度，$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ 是 TD error，$\hat{A}_t$ 是优势函数的估计值，$V(s_t)$ 是 value function 的估计值。

下面将原文的算法流程2直接贴在下面：

A proximal policy optimization (PPO) algorithm that uses fixed-length trajectory segments is shown below. Each iteration, each of N (parallel) actors collect T timesteps of data. Then we construct the

surrogate loss on these N T timesteps of data, and optimize it with minibatch SGD (or usually for better performance, Adam), for K epochs.

Algorithm 2 PPO, Actor-Critic Style

for iteration = 1, 2, ... **do**
  for actor = 1, 2, ..., N **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, ..., \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

2.4.5 Commentary

从最原始的 policy gradient 方法出发，

$$L^{\text{PG}}(\theta) = \hat{\mathbb{E}}_t \left[\hat{A}_t \log \pi_\theta(a_t|s_t) \right]$$

PPO (Schulman et al., 2017) 和 TRPO (Schulman et al., 2015b) 都是在做策略梯度的部分 $\log \pi_\theta(a_t|s_t)$进行改进；而从原来的 Q 函数变成优势函数、GAE (Schulman et al., 2015a) 工作都是在前半部分 $\hat{A}_t$ 改进。

但是 PPO 在 TRPO 的基础上做了进一步改进，使得目标函数可以通过一个非常简单的 clip 函数来实现 pessimistic bound 的效果，避免了 TRPO 中计算 KL 和调 β 的问题。

但是在后面的 RL4LLM 的部分中我们可以看到，大部分工作还是会有一个 KL 的约束条件来使得 π^{RL} 和 π^{SFT} 之间的差距不会太大。

2.5 Soft Actor-Critic

这个工作简称 SAC (Haarnoja et al., 2018)。这个工作是在 EBP (Haarnoja et al., 2017) 的基础上继续做的，两个都是发的 ICML，之前的工作没有打败 DDPG，这个工作拉开了 SOTA 一大截，还是很有借鉴意义的；和 PPO 不一样，这个工作是和 DDPG 一样专注于 continuous action space 的，且是 off-policy 的 (文章中分析说 PPO，GAE 这些算法是在每个 policy gradient 都要搜集一批样本所以是 on-policy 的)。

2.5.1 Preliminaries

之前的 RL 的目标函数都是 maximize reward objective

$$J(\pi) = \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t, a_t)]$$

这里的目标函数考虑 maximize entropy objective

$$J(\pi) = \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))\right] \tag{2.19}$$

Notation	Description
$\mathcal{S}$	State space (continuous).
$\mathcal{A}$	Action space (continuous).
p	$\mathcal{S}\times\mathcal{S}\times\mathcal{A}\to[0,\infty)$, state transition probability density function $p(s_{t+1}\|s_t,a_t)$
r	$\mathcal{S}\times\mathcal{A}\to[r_{\min},r_{\max}]$, reward function $r(s_t,a_t)$, bounded between $r_{\min}$ and $r_{\max}$.
s_{t+1},s_t,a_t	Next state, current state, action at time step t.
$\pi(a_t\|s_t)$	Policy, probability of taking action a_t given state s_t.
$\rho_\pi(s_t),\rho_\pi(s_t,a_t)$	State and state-action marginals of the trajectory distribution induced by policy π.

表 2.4: Notations for an infinite-horizon Markov Decision Process (MDP) in SAC.

和之前的 PPO (2.18) 只是把它当作一个熵正则化项不一样，这里的目标是要将熵的部分其最大化，其表达式如下：

$$\mathcal{H}(\pi(\cdot|s_t))=-\int_{\mathcal{A}}\pi(a|s_t)\log\pi(a|s_t)da$$

引入了熵部分的目标函数也是可以用原来的 policy iteration 的方法求解出 optimal policy 的，主要证明思路就是把熵看作是属于奖励的一部分就可以了，详细见附录 A.2。

2.5.2 Method

Q-function, policy 和 value function 均采用了神经网络来近似，下面是其算法部分设计到的符号表达：

Notation	Description
$V_\psi(s)$	State value function, parameterized by ψ.
$V_{\bar{\psi}}(s)$	Target state value function, parameterized by $\bar{\psi}$, $\bar{\psi}$ is a moving average of ψ.
$Q_\theta(s,a)$	Soft Q-function, parameterized by θ, we use double Q-learning to mitigate overestimation bias, so we have two Q-functions $Q_{\theta_1}(s,a)$ and $Q_{\theta_2}(s,a)$.
$\pi_\phi(a\|s)$	Policy, parameterized by ϕ, we use a Gaussian policy with mean and covariance given by neural networks.

表 2.5: Notations for the SAC algorithm.

本来我们可以直接根据 $Q_\theta(s,a)$ 去通过 MC 的方法直接得到对应的 V 的(A.8)

$$V(s_t)=\mathbb{E}_{a_t\sim\pi}[Q_\theta(s_t,a_t)-\log\pi(a_t|s_t)]$$

但是引入 separate 的 value function approximator $V_\psi(s)$ 对于训练过程来说是更稳定的，所以此时我们 soft value function 的网络的 objective function 就变成了

$$J_V(\psi)=\mathbb{E}_{s_t\sim\mathcal{D}}\left[\frac{1}{2}\Big(V_\psi(s_t)-\mathbb{E}_{a_t\sim\pi_\phi}[Q_\theta(s_t,a_t)-\log\pi_\phi(a_t|s_t)]\Big)^2\right] \tag{2.20}$$

其中 $\mathcal{D}$ 是一个 replay buffer，但是这里的 action 是从当前策略 π_ϕ 中采样的（而不是 replay buffer）。而对于 soft Q-function 的目标函数则是

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}\Big(Q_\theta(s_t,a_t) - \hat{Q}(s_t,a_t)\Big)^2\right] \tag{2.21}$$

其中 $\hat{Q}(s_t,a_t)$ 是一个 target Q-value，定义如下：

$$\hat{Q}(s_t,a_t) = r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\bar{\psi}}(s_{t+1})] \tag{2.22}$$

根据(A.9)，我们此时的 policy objective 就是

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}}\left[D_{\mathrm{KL}}\left(\pi_\phi(\cdot|s_t)\|\frac{\exp(Q_\theta(s_t,\cdot))}{Z_\theta(s_t)}\right)\right] \tag{2.23}$$

这里可以用了重参数采样的技巧，我们记这样的一个 transformation 为

$$a_t = f_\phi(\epsilon_t; s_t)$$

其中 ϵ_t 是一个符合高斯分布的随机变量。这样我们就可以把(2.23)变成

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}\left[\log\pi_\phi(f_\phi(\epsilon_t;s_t)|s_t) - Q_\theta(s_t, f_\phi(\epsilon_t;s_t))\right] \tag{2.24}$$

最终的算法流程见算法 3。

Algorithm 3 SAC, Soft Actor-Critic

Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $a_t \sim \pi_\phi(a|s_t)$
    $s_{t+1} \sim p(s_{t+1}|s_t,a_t)$
    $\mathcal{D} \leftarrow \mathcal{D}\cup\{(s_t,a_t,r(s_t,a_t),s_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V\hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q\hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i\in\{1,2\}$
    $\phi \leftarrow \phi - \lambda_\pi\hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} = \tau\psi + (1-\tau)\bar{\psi}$
  end for
end for

2.5.3 Commentary

重参数采样的技巧 (reparametrization trick) 其实也比较简单，

Listing 2.1: Reparametrization trick in Python

```
# policy_net: obs --> features --> latent_pi
# action_mean = policy_net.mu(latent_pi)
```

```
    # log_std = policy_net.log_std(latent_pi)
    # log_std = torch.clamp(log_std, LOG_STD_MIN, LOG_STD_MAX)
    action_std = torch.ones_like(action_mean) * log_std.exp()
    action_dist = Normal(action_mean, action_std)
    # we can sample from the action distribution and get the action
    action = action_dist.sample() # reparametrization trick
```

另外在 Stable Baselines3 (Raffin et al., 2021) 中所实现的 SAC 算法和这里的不太一样，其中实现是没有包括 value function 的，actor 就是指 policy π_ϕ，critic 就是指 Q-function Q_θ，其中因为去掉了 value function, 所以 value_target 也一并去掉了，只有 critic 和 critic_target 部分；除此之外，SAC 的实现中还包括了一个 target_entropy 的参数（传进去之后就意味着我们的 α 可以有 loss），这个参数是用来平衡熵正则化项和奖励项的权重的，具体来说就是在(2.19)中 α 的值是可以调节的。

Chapter 3

LLM Policy Gradient

下面我们开始过一遍 LLM Policy Gradient 的发展情况；

3.1 Notations

我们使用以下的符号来表示一些重要的概念，这里的符号体系参考了 InstructGPT (Ouyang et al., 2022)：

Notation	Description
(x, y)	A pair where x is the input (prompt, input prefix) and y is the output (model response).
D	A dataset, e.g., D_{pretrain} represents the pretraining dataset.
π	Policy, representing the model's output distribution. E.g., $\pi_\theta(y\|x)$ is the probability of output sequence y given input sequence x.
π_θ	A parameterized policy, where θ is the model's parameters (e.g., transformer model parameters).
$r(x, y)$	Reward function or model, output a scalar reward given inputs x and y.

表 3.1: Notations used in LLM Policy Gradient

3.2 Learning to summarize with human feedback

TL;DR: 这是 InstructGPT 工作的前身，这里将其简称为 Summarize RLHF (Stiennon et al., 2020)，基本是现在主流 RLHF 框架参考的模式 pretrain+SFT+RL。

3.2.1 Reward model

其主要的工作是先给定一个数据集 D，每个样本包含一个 post x 和两个 summary $y \in \{y_0, y_1\}$ 组成的 pair，然后通过人类反馈去标注哪个 y 好（此时获得了 human preference），假设 y_i 更好，然后根据这个数据去训练一个 reward model r_θ (input prompt x and response y, output a scalar reward r)，

objective function 如下：

$$\text{loss}(r_\theta) = -E_{(x,y_0,y_1,i)\sim D}\left[\log\sigma\left(r_\theta(x,y_i) - r_\theta(x,y_{1-i})\right)\right] \tag{3.1}$$

此时 D 为人类判断处理过后的数据集（人类偏好数据集），σ 是 sigmoid 函数。

简单分析损失函数

令 $d = r_\theta(x,y_i) - r_\theta(x,y_{1-i})$，$\text{loss} = -\log\sigma(d)$，我们可以分析一下这个损失函数的性质：
1. 当 $d \geq 0$ 时，随着 d 的增大，$\sigma(d) \to 1 \Rightarrow \text{loss} \to 0$；
2. 当 $d \leq 0$ 时，随着 d 的减小，$\sigma(d) \to 0 \Rightarrow \text{loss} \to +\infty$；
如果希望 $\text{loss}\downarrow \Rightarrow d\uparrow \Rightarrow (r_\theta(x,y_i) - r_\theta(x,y_{1-i}))\uparrow$，也即希望训练到最后的结果是 reward model 输出的 y_i 的 reward 要大于 y_{1-i} 的 reward。

3.2.2 Reward and objective function

得到奖励模型之后，会把最后输出的奖励 normalize 成 mean 为 0。接着就可以用这个奖励模型来训练 LLM 的 policy 了，不过这篇工作没有具体给出 policy gradient 的具体算法（说是用了 PPO，其中 policy，value function，reward model 都是 same size 的 transformer，value function 的参数用 reward model 初始化），只是给了 policy 训练中采取的奖励形式 (RM + KL control (Jaques et al., 2017))

$$R(x,y) = r_\theta(x,y) - \beta\log\frac{\pi_\phi^{\text{RL}}(y|x)}{\pi^{\text{SFT}}(y|x)} \tag{3.2}$$

这里的 β 是一个超参数，π_ϕ^{RL} 是当前的策略（ϕ 是当前的参数），π^{SFT} 是之前的策略。原工作没有给出 objective function，为了更好的过渡到下一篇，这里给出一个简单形式的 objective function 并分析：

$$\text{objective}(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y) - \beta\log\frac{\pi_\phi^{\text{RL}}(y|x)}{\pi^{\text{SFT}}(y|x)}\right] \tag{3.3}$$

下面对(3.3)做解释：

1. 下面三种形式的期望都是等价的 (不同论文里基本都是这样写)，都是指我们用当前策略 π_ϕ^{RL} 输入 x 得到 y，进而在这个前提下来求期望，（此时还是属于 reponse level 或者叫 prompt level 的，与 token level 相对，response level 下的 input x 就可以被视为一个 state，output y 可以被视为模型给我们的 action，但这样最终会导致一些问题，TODO 后面在 PPO 解释）：

$$E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}[*] = E_{x\sim D, y\sim\pi_\phi^{\text{RL}}(\cdot|x)}[*] = E_{x\sim D}\left[E_{y\sim\pi_\phi^{\text{RL}}(\cdot|x)}[*]\right]$$

2. 这个 objective function 其实就是 $E_\pi(R)$(R 就是公式(3.2)，下标 π 表示由这个 RL 策略生成的数据)，可以考虑用 MC 去 batch 地操作获得最后的 mean reward 再反向传播。可以看到下面操作后我们得到了 KL penalty 项，但实际上我们最后是要 maximize 这个 objective：

$$\begin{aligned}
E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y) - \beta\log\frac{\pi_\phi^{\text{RL}}(y|x)}{\pi^{\text{SFT}}(y|x)}\right] &= E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y)\right] - \beta E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[\log\frac{\pi_\phi^{\text{RL}}(y|x)}{\pi^{\text{SFT}}(y|x)}\right] \\
&= \underbrace{E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y)\right]}_{\text{maximize reward}} - \beta\underbrace{\text{KL}(\pi_\phi^{\text{RL}}||\pi^{\text{SFT}})}_{\text{minimize KL penalty}}
\end{aligned}$$

3.2.3 Commentary

Summarize RLHF (Stiennon et al., 2020) 的工作主要是对一个 post x 和两个 summary y_0, y_1 进行比较，得到一个比较 $y_0 \succ y_1$，然后去训练一个 reward model r_θ，最后再通过这个 reward model 去训练 LLM 的 policy。相当于把基础的 RLHF pipeline 搭建起来了，但是这个工作是 response level 的，即使在原文中提到是用 PPO 来训练的（我们当下一般认为 PPO 是 token level 的）

3.3 Training language models to follow instructions with human feedback

TL;DR: 这是 InstructGPT (Ouyang et al., 2022) 的工作，主要是对上面的工作做了数据方面、目标函数方面的改进。

3.3.1 Reward model

首先是在 comparison collection 方面，不再是之前那样一个 post 两个 summary 比较了，而是针对一个 post 生成 K 个 responses，那么就会有 $\binom{K}{2} = \frac{K(K-1)}{2}$ 个 comparison，然后这里还有一个细节就是他们不是将得到的所有 comparisons 全部打乱然后拿去训练 (会 overfits)，而是将同一个 prompt(task) 生成的 responses 放在一起 (在一个 batch 里面) 一块去训练。此时的 reward model 的 loss function 如下：

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D} \left[\log \sigma \left(r_\theta(x, y_w) - r_\theta(x, y_l)\right)\right]$$

这里的 r_θ 和之前一样，y_w 是 preferred response in y_w, y_l，D 是 human comparisons。$\binom{K}{2}$ 是组合数，表示在一个 batch 里面有多少个比较 (batch size)。同样在最后的 reward model 训练完成后也会进行 normalize 成 a mean score of 0。

3.3.2 Reward and objective function

原文给了 objective function，是在之前的基础上 mix 了 pretraining gradients 的形式，具体如下：

$$\text{objective}(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}} \left[r_\theta(x, y) - \beta \log \frac{\pi_\phi^{\text{RL}}(y|x)}{\pi^{\text{SFT}}(y|x)}\right] + \lambda E_{x\sim D_{\text{pretrain}}} \left[\log \pi_\phi^{\text{RL}}(x)\right]$$

对这个新加入的 prtraining 项目进行分析就是这里用的还是之前 D_{pretrain} 的预训练数据集，λ 是一个超参数，$\pi_\phi^{\text{RL}}(x)$ 是当前策略在输入 x 下的输出概率，显然 $\pi \in [0, 1]$，所以 $\log \pi_\phi^{\text{RL}}(x)$ 是一个负数，所以要 maximize 这个项的作用就是希望在训练的时候能够让模型的输出概率更大一些 (希望其尽可能接近 1)。此外也可以从熵的角度去理解，借鉴 SAC (Haarnoja et al., 2018) 的理解，我们这一步相当于是在 minimize $E_{x\sim D_{\text{pretrain}}}[-\log \pi_\phi^{\text{RL}}(x)]$，也即最小化模型在预训练数据集的熵，使其输出的动作概率更集中。

3.3.3 Commentary

InstructGPT (Ouyang et al., 2022) 相比 Summarize RLHF (Stiennon et al., 2020) 的主要改进是从对一个 prompt 一个 comparison $y_0 \succ y_1$ 相当于变成了一个 prompt 多个 comparisons，即人类给出的一个偏好序列，比如 $y_0 \succ y_1 \succ y_2 \succ \cdots \succ y_K$，那么就可以学到一个偏好序列的 reward model 了。但

依然是属于 response level 的，相比之前最大的进步在于不是做单一的 summarize 任务了，而是去做了 instruction 的任务，让其更加遵循人类的指令（相当于从而大大提高了其多任务的能力）。

My Thoughts

SAC (Haarnoja et al., 2018) 加入熵的目的是希望可以增大熵，从而增强模型的探索能力，如果要在这里做类似 SAC 的改进的话是不是也可以考虑在前面这一项的期望 $E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}$ 中加入一项 $-\log \pi_\phi^{\mathrm{RL}}(x)$，鼓励模型多去探索更多的回答方案，因为这还是属于 reponse level 的输出，如果在 token level 中加入这一项说不定会更好，获得更加好的采样多样性。

3.4 Direct Preference Optimization

TL;DR: DPO (Rafailov et al., 2023) 在之前的基础上将奖励模型的部分去掉了（认为 llm 本来就是一个 reward model）。

3.4.1 Objective

DPO 首先根据 Bradley-Terry 模型 (Bradley and Terry, 1952) 定义了一个偏好分布：

$$p(y_1 \succ y_2|x) = \frac{\exp(r(x, y_1))}{\exp(r(x, y_1)) + \exp(r(x, y_2))} \tag{3.4}$$

其中 $r(x, y)$ 是一个 general reward function，接着可以推出在 KL-constrained 条件下的 optimal solution

$$\pi_r(y|x) = \frac{\pi_{\mathrm{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)}{\sum_y \pi_{\mathrm{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)} \tag{3.5}$$

令

$$Z(x) = \sum_y \pi_{\mathrm{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right) \tag{3.6}$$

那么就可以推出 $r(x, y)$ 的表达式：

$$r(x, y) = \beta \log \frac{\pi_r(y|x)}{\pi_{\mathrm{ref}}(y|x)} + \beta \log Z(x) \tag{3.7}$$

注意

我们对于 Z 和 r 都是不知道实际的表达式的，也不会用神经网络去逼近它 (reward model 的做法)，在这里我们的 r 只是最后推出 objective function 的一个辅助工具。但是也可以观察到这里的 reward 和之前的 InstructGPT 的操作是反着来的：

$$r(x, y) = r_\phi(x, y) - \beta \log \frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)}$$

但是 DPO 是把 LLM 当成一个 reward model 来训练了，而之前的 InstructGPT 是把 LLM 当成一个 actor 来训练的。

我们将 (3.7) 代入 (3.4) 中得：

$$\begin{aligned} p(y_1 \succ y_2|x) &= \frac{\exp(r(x,y_1))}{\exp(r(x,y_1)) + \exp(r(x,y_2))} = \frac{1}{1+\exp(r(x,y_2) - r(x,y_1))} \\ &= \frac{1}{1+\exp(\beta \log \frac{\pi_r(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta \log \frac{\pi_r(y_1|x)}{\pi_{\text{ref}}(y_1|x)})} \\ &= \sigma\left(\beta \log \frac{\pi_r(y_1|x)}{\pi_{\text{ref}}(y_1|x)} - \beta \log \frac{\pi_r(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right) \end{aligned}$$

那么参考之前的训练 reward model 的 objective function (3.1)，我们的 policy objective 就变成了下面这个式子 (minimize it):

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right] \tag{3.8}$$

其中 y_w 和之前一样是 prefered 的数据，可以看出 minimize $-\mathbb{E}[\log p(y_w \succ y_l|x)]$ 其实就是 maximize $p(y_w \succ y_l|x)$ 的过程。

3.4.2 Commentary

基本有相当多的工作已经证明 DPO 的不好：DPO 只能学会让大模型减少输出不好的回答而忽略了更优的回答；训练的效果和 SFT 基模型的性能密切相关；而后续的 DAPO 工作对 kl 项的删除则更加表明了训练出来的学会 reason 的模型与基模型存在非常大的差异，但是 DPO 的 objective 直接就是写成 $(\pi_\theta; \pi_{\text{ref}})$ 的格式，即将 LLM 表示为一个 reward model $\hat{r}_\theta$:

$$\hat{r}_\theta(x,y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$$

即我们的目标是只要使得 $\hat{r}_\theta(x,y_w) > \hat{r}_\theta(x,y_l)$，但是，DPO 完全忽略了我们更加应该最大化符合人类偏好的奖励，按照 DPO 的做法我们有

$$\mathbb{E}_{\pi_\theta}\left[\hat{r}_\theta(x,y)\right] = \beta \mathbb{E}_{\pi_\theta}\left[\log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\right] = \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$$

从 KL 散度 (2.1) 的角度而言，DPO 在做的事情就是希望输出 y_w 的 D_{KL} 要大于 y_l 的 D_{KL} 而不管最后 y_w 的 D_{KL} 会到多大（因为这个 $\mathbb{E}[\hat{r}]$ 并不是 DPO 的 objective）；退一步讲，如果使用最大化奖励作为 objective，那么按照这个奖励的形式优化到最后的结果是 $D_{\text{KL}} = 0$，此时 π_θ 和 π_{ref} 并没有什么区别了。

3.5 Group Relative Policy Optimization

TL;DR: 这里主要介绍 DeepSeek 的两个工作 (Shao et al., 2024; Guo et al., 2025)，它们都使用了 Group Relative Policy Optimization (GRPO) 。

3.5.1 PPO for LLM Fine-tuning

将之前的 PPO (Schulman et al., 2017) 算法用在微调 LLM 上，最早是 InstructGPT (Ouyang et al., 2022)，它主要是 maximizing 下面的目标函数：

$$J_{\text{PPO}}(\theta) = \mathbb{E}_{q\sim P(Q),o\sim\pi_{\theta_{\text{old}}}(O|q)}\left[\frac{1}{|o|}\sum_{t=1}^{|o|}\min\left(\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t|q,o_{<t})}A_t, \text{clip}\left(\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t|q,o_{<t})}, 1-\varepsilon, 1+\varepsilon\right)A_t\right)\right] \tag{3.9}$$

Notation	Description
Q	Set of questions/prompts.
O	Set of responses/outputs.
$P(Q)$	Distribution of questions.
$\pi_\theta(O\|q)$	Distribution of responses given question q under policy π_θ.
θ	Parameters of the current policy model.
θ_{old}	Parameters of the policy model from the previous iteration.
q	Sampled question/prompt.
o	Sampled response.
o_t	The t-th token in response o.
$o_{<t}$	Tokens in response o preceding the t-th token.
A_t	Advantage function at token t, estimated using GAE.
V_ψ	Value model with parameters ψ.
r_t	Reward at token t.
r_ϕ	Reward model with parameters ϕ.
β	Hyperparameter controlling the KL penalty strength.
π_{ref}	Reference model, typically the initial SFT model.
ε	Hyperparameter for PPO clipping.

表 3.2: Notations used in PPO for LLM Fine-tuning (Equation 3.9)

A_t 的计算基于 $\{r_{\geq t}\}$ 和 value model V_ψ，$r_{\geq t}$ 是从 t 到 T 的 reward，参考(2.14)和 PPO 的做法，我们用 $\lambda = 1$ 的 GAE：

$$A_t = \sum_{l=0}^{T-1-t} \gamma^l \delta_{t+l}$$

$$\delta_t = r_t + \gamma V_\psi(q, o_{<t+1}) - V_\psi(q, o_{<t})$$

注：上面这个形式是非常适合 next-token prediction 的网络输出的，V_ψ 也可以是一个 transformer decoder 的输出；在训练过程中，value model V_ψ 需要随着 policy model π_θ 一起训练去防止 reward model r_ϕ 的过度优化；

这里和之前 (Ouyang et al., 2022) response level 的 KL penalty 不一样，将 KL penalty 算入了每一个 token 的计算中：

$$r_t = r_\phi(q, o_{\leq t}) - \beta \log \frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\text{ref}}(o_t|q, o_{<t})}$$

一些细节

注意到如果放在 token level 中，我们的状态 $s_t = (q, o_{<t})$，动作 $a_t = o_t$，所以可以看到 V_ψ 的输入是 $q, o_{<t}$，也即：

$$V_\psi(s_t) = V_\psi(q, o_{<t})$$

可以看到 $r_\phi(\cdot)$ 括号中的 o 的下标是小于等于号，也即：

$$r_\phi(s_t, a_t) = r_\phi(q, o_{<t}, o_t) = r_\phi(q, o_{\leq t})$$

3.5.2 GRPO

将 updated policy π_θ 和 old policy $\pi_{\theta_{\text{old}}}$ 之间的比值记为：

$$r_i(\theta) = \frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} \tag{3.10}$$

那么 i-th clip surrogate objective 此时为：

$$J_i^{\text{clip}}(\theta) = \min\left(r_i(\theta)A_i, \text{clip}\left(r_i(\theta), 1-\varepsilon, 1+\varepsilon\right)A_i\right) \tag{3.11}$$

其中 A_i 是 advantage，其计算如下（推导见附录A.3.1）：

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})} \tag{3.12}$$

再此基础上我们再加上一个 KL penalty (Ouyang et al., 2022)，参考(2.2)，我们将一个方差更小的 KL 无偏估计作为 penalty （推导见附录A.3.2）：

$$\mathbb{D}_i^{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - 1 \tag{3.13}$$

GRPO (Guo et al., 2025) 中所给出的 response-level objective function 如下:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(O|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\left(J_i^{\text{clip}}(\theta) - \beta\mathbb{D}_i^{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\right)\right] \tag{3.14}$$

3.5.3 Commentary

注意这里的 o_i 和之前的 o_t 不一样，o_i 是一个完整的 response，而 o_t 是一个 token；我们会对于一个 question q 采样 G 个 response o_i，然后 rollout 每一个 response 的 A_i；

GRPO 的全称叫做 Group Relative Policy Optimization，Group 的意思就是指生成 G 个 response；而 Relative 的意思就是指减去平均奖励作为相对的 advantage；这里介绍的是 (Guo et al., 2025) 中的 response-level GRPO，和最原始的 token-level GRPO (Shao et al., 2024) 有所区别，附录A.3.3中同样也给出了后者的推导。

参考文献

Jiacai Liu. Brief introduction of policy gradient in llm reasoning. Notion Link, 2025.

John Schulman. Approximating KL divergence. `http://joschu.net/blog/kl-approx.html`, 2020.

Sergey Levine. Deep reinforcement learning: Lecture 5 policy gradients. `https://rail.eecs.berkeley.edu/deeprlcourse/deeprlcourse/static/slides/lec-5.pdf`, 2023a. UC Berkeley CS 285.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015a.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015b.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

YuYue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Ru Zhang, Xin Liu, Mingxuan Wang, Yonghui Wu, and Lin Yan. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks, 2025.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model, 2025.

Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *International Conference on Machine Learning*, pages 1645–1654. PMLR, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2020.

Linkang Dong. Note of policy gradient methods. RL Note Link, 2025.

Sergey Levine. Deep reinforcement learning: Lecture 9 advanced policy gradients. `https://rail.eecs.berkeley.edu/deeprlcourse/deeprlcourse/static/slides/lec-9.pdf`, 2023b. UC Berkeley CS 285.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR, 2016.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of machine learning research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. Gpg: A simple and strong reinforcement learning baseline for model reasoning. *arXiv preprint arXiv:2504.02546*, 2025.

附录 A

Supplementary Material

A.1 Causality in Policy Gradient Methods

我们在第2.2中提到，因果性可以帮助我们减少方差。这里我们为 policy gradient 章节中出现的每个使用 causality 来减少方差的公式给一个简单的例子来直观展示减少方差的推导过程。

A.1.1 On-Policy Policy Gradient

首先是 on-policy policy gradient(without baseline)(2.6)，这部分做的工作是通过忽略当前策略之前的所有 reward，只考虑当前策略之后的 reward 来减少方差, 即，未来的策略不会对当前的奖励产生影响：

$$\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\left(\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(a_t|s_t)\right)\left(\sum_{t=1}^{T}r(s_t,a_t)\right)\right]$$
$$\Rightarrow\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t_1=1}^{T}\nabla_{\theta}\log\pi_{\theta}(a_{t_1}|s_{t_1})\sum_{t_2=t_1}^{T}r(s_{t_2},a_{t_2})\right]$$

为了使得符号更简洁，我们记

$$g_t=\nabla_{\theta'}\log\pi_{\theta'}(a_t|s_t)\quad r_t=r(s_t,a_t)$$

此时我们只关注期望中括号内的值，假设 $T=4$，那么我们有

$$\begin{aligned}\sum g_t\sum r_t=&(g_1+g_2+g_3+g_4)\cdot(r_1+r_2+r_3+r_4)\\=&g_1\cdot(r_1+r_2+r_3+r_4)\\&+g_2\cdot(r_1+r_2+r_3+r_4)\\&+g_3\cdot(r_1+r_2+r_3+r_4)\\&+g_4\cdot(r_1+r_2+r_3+r_4)\end{aligned}$$

将结果写成矩阵形式如下所示：

$$\sum g_t \sum r_t = \sum \left(\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \cdot \begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \end{bmatrix} \right)$$

我们可以看到这里相当于只计算了上三角矩阵部分的值，下面部分的奖励全设置为 0，通过 causality 我们使得当前的 on-policy policy gradient 减少了几乎一半的方差。

A.1.2 Off-Policy Policy Gradient

接下来考虑 off-policy policy gradient(2.10)，这是加入了 importance sampling 的方法，我们直观的理解，动作的采样只受当前策略的影响，而当前策略只受到过去 importance sampling 的影响，即，未来的重要性采样不会对当前的策略产生影响：

$$\begin{aligned} &\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\left(\prod_{t=1}^{T}\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\right)\left(\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(a_t|s_t)\right)\left(\sum_{t=1}^{T}r(s_t,a_t)\right)\right] \\ =&\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t_1=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(a_{t_1}|s_{t_1})\left(\prod_{t_2=1}^{t_1}\frac{\pi_{\theta'}(a_{t_2}|s_{t_2})}{\pi_\theta(a_{t_2}|s_{t_2})}\right)\left(\sum_{t_3=t_1}^{T}r(s_{t_3},a_{t_3})\left(\prod_{t_4=t_1}^{t_3}\frac{\pi_{\theta'}(a_{t_4}|s_{t_4})}{\pi_\theta(a_{t_4}|s_{t_4})}\right)\right)\right] \end{aligned}$$

同样的，我们记

$$w_t = \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \quad g_t = \nabla_{\theta'}\log\pi_{\theta'}(a_t|s_t) \quad r_t = r(s_t,a_t)$$

同样假设 $T=4$，那么我们有

$$\begin{aligned} &\prod w_t \sum g_t \sum r_t = \sum g_t \sum r_t \prod w_t \\ =&g_1\cdot(r_1+r_2+r_3+r_4)\cdot(w_1\cdot w_2\cdot w_3\cdot w_4) \\ +&g_2\cdot(r_1+r_2+r_3+r_4)\cdot(w_1\cdot w_2\cdot w_3\cdot w_4) \\ +&g_3\cdot(r_1+r_2+r_3+r_4)\cdot(w_1\cdot w_2\cdot w_3\cdot w_4) \\ +&g_4\cdot(r_1+r_2+r_3+r_4)\cdot(w_1\cdot w_2\cdot w_3\cdot w_4) \end{aligned}$$

进一步展开有

$$\begin{aligned} &g_1\cdot\Big(r_1\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_2\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_3\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_4\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)\Big) \\ +&g_2\cdot\Big(r_1\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_2\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_3\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_4\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)\Big) \\ +&g_3\cdot\Big(r_1\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_2\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_3\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_4\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)\Big) \\ +&g_4\cdot\Big(r_1\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_2\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_3\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)+r_4\cdot(w_1\cdot w_2\cdot w_3\cdot w_4)\Big) \end{aligned}$$

得到的结果写成矩阵形式如下：

$$\prod w_t \sum g_t \sum r_t \tag{A.1}$$

$$= \sum \left(\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \cdot \begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \end{bmatrix} \times \begin{bmatrix} w_1 \cdot w_2 \cdot w_3 \cdot w_4 \\ w_1 \cdot w_2 \cdot w_3 \cdot w_4 \\ w_1 \cdot w_2 \cdot w_3 \cdot w_4 \\ w_1 \cdot w_2 \cdot w_3 \cdot w_4 \end{bmatrix} \right) \tag{A.2}$$

$$= \sum \left(\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_1 \cdot w_2 \\ w_1 \cdot w_2 \cdot w_3 \\ w_1 \cdot w_2 \cdot w_3 \cdot w_4 \end{bmatrix} \cdot \begin{bmatrix} r_1 & r_2 \cdot w_2 & r_3 \cdot w_2 \cdot w_3 & r_4 \cdot w_2 \cdot w_3 \cdot w_4 \\ & r_2 & r_3 \cdot w_3 & r_4 \cdot w_3 \cdot w_4 \\ & & r_3 & r_4 \cdot w_4 \\ & & & r_4 \end{bmatrix} \right) \tag{A.3}$$

从第一个等号可以看出，使用 causality 我们对奖励的计算只考虑了上三角，对 importance sampling 的系数，我们只考虑了类似下三角的部分；而第二个等号实际上就是我们开头那个复杂的式子 (2.10)的简化展开版。

A.1.3 Simplified Off-Policy Policy Gradient

这里先用另一个角度来看待“通过 causality 减少方差”的问题，即：确定是谁和谁的 causality 的问题。如果我们认为的是奖励和 importance sampling 的 causality，那么上面(A.3)的考虑就是合理的; 如果我们考虑的是 gradient 和 importance sampling 的 causality，那么我们的计算过程就变为:

$$\prod w_t \sum g_t \sum r_t = \sum g_t \prod w_t \sum r_t \tag{A.4}$$

$$= \sum \left(\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \cdot \begin{bmatrix} w_1 \cdot w_2 \cdot w_3 \cdot w_4 \\ w_1 \cdot w_2 \cdot w_3 \cdot w_4 \\ w_1 \cdot w_2 \cdot w_3 \cdot w_4 \\ w_1 \cdot w_2 \cdot w_3 \cdot w_4 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \end{bmatrix} \right) \tag{A.5}$$

$$= \sum \left(\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_1 \cdot w_2 \\ w_1 \cdot w_2 \cdot w_3 \\ w_1 \cdot w_2 \cdot w_3 \cdot w_4 \end{bmatrix} \cdot \begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ & r_2 & r_3 & r_4 \\ & & r_3 & r_4 \\ & & & r_4 \end{bmatrix} \right) \tag{A.6}$$

可以看到这里我们通过交换顺序首先考虑 g 和 w 的 causality，然后再考虑 g 和 r 的 causality，最后得到的结果(A.6)比较(A.3)要简洁很多。此时对应的 element-wise 的表达式就是(2.11):

$$\nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t_1=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_{t_1}|s_{t_1}) \left(\prod_{t_2=1}^{t_1} \frac{\pi_{\theta'}(a_{t_2}|s_{t_2})}{\pi_\theta(a_{t_2}|s_{t_2})} \right) \left(\sum_{t_3=t_1}^{T} r(s_{t_3}, a_{t_3}) \right) \right]$$

A.1.4 First-Order Off-Policy Policy Gradient

在这个基础上我们直接用 importance sampling 的一阶近似来考虑问题会变得更简单(2.13):

$$\nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) \hat{Q}_t \right]$$

记

$$w_t = \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \quad g_t = \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) \quad Q_t = \hat{Q}_t$$

此时的 matrix form 为：

$$\sum \left(\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \cdot \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{bmatrix} \right)$$

A.1.5 TL;DR

我们这一节都是在用 T 维度上的 causality 来减少方差，先是用 on-policy policy gradient 的 g 和 r 的 causality 来减少方差:

$$\left(\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) \right) \left(\sum_{t=1}^{T} r(s_t, a_t) \right)$$
$$\Rightarrow \sum_{t_1=1}^{T} \nabla_\theta \log \pi_\theta(a_{t_1}|s_{t_1}) \sum_{t_2=t_1}^{T} r(s_{t_2}, a_{t_2})$$

接着用 off-policy policy gradient 的 g 和 r 的 causality、r 和 w 的 causality 来减少方差；接着进一步简化成利用 g 和 w 的 causality、g 和 r 的 causality 来减少方差:

$$\left(\prod_{t=1}^{T} \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \right) \left(\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) \right) \left(\sum_{t=1}^{T} r(s_t, a_t) \right)$$
$$\Rightarrow \sum_{t_1=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_{t_1}|s_{t_1}) \left(\prod_{t_2=1}^{t_1} \frac{\pi_{\theta'}(a_{t_2}|s_{t_2})}{\pi_\theta(a_{t_2}|s_{t_2})} \right) \left(\sum_{t_3=t_1}^{T} r(s_{t_3}, a_{t_3}) \left(\prod_{t_4=t_1}^{t_3} \frac{\pi_{\theta'}(a_{t_4}|s_{t_4})}{\pi_\theta(a_{t_4}|s_{t_4})} \right) \right)$$
$$\Rightarrow \sum_{t_1=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_{t_1}|s_{t_1}) \left(\prod_{t_2=1}^{t_1} \frac{\pi_{\theta'}(a_{t_2}|s_{t_2})}{\pi_\theta(a_{t_2}|s_{t_2})} \right) \left(\sum_{t_3=t_1}^{T} r(s_{t_3}, a_{t_3}) \right)$$

最后采用一阶近似的 off-policy policy gradient 来减少方差:

$$\sum_{t_1=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_{t_1}|s_{t_1}) \left(\prod_{t_2=1}^{t_1} \frac{\pi_{\theta'}(a_{t_2}|s_{t_2})}{\pi_\theta(a_{t_2}|s_{t_2})} \right) \left(\sum_{t_3=t_1}^{T} r(s_{t_3}, a_{t_3}) \right)$$
$$\Rightarrow \sum_{t_1=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_{t_1}|s_{t_1}) \frac{\pi_{\theta'}(a_{t_1}|s_{t_1})}{\pi_\theta(a_{t_1}|s_{t_1})} \left(\sum_{t_3=t_1}^{T} r(s_{t_3}, a_{t_3}) \right)$$
$$\Rightarrow \sum_{t_1=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_{t_1}|s_{t_1}) \frac{\pi_{\theta'}(a_{t_1}|s_{t_1})}{\pi_\theta(a_{t_1}|s_{t_1})} \hat{Q}_{t_1}$$

A.2 Derivation of Soft Policy Iteration

A.2.1 Soft Policy Evaluation

对于 soft policy iteration 的 policy evaluation 步骤，我们希望计算一个 policy π 的值函数，按照最大熵目标函数(2.19)。对于一个固定的 policy，soft Q-value 可以迭代地计算出来，从任意函数 $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ 开始，重复应用一个修改过的 Bellman backup operator $\mathcal{T}_\pi$，其定义如下：

$$\mathcal{T}_\pi Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}[V(s_{t+1})] \tag{A.7}$$

其中 $V(s_t)$ 是 soft state value function，定义如下：

$$V(s_t) = \mathbb{E}_{a_t \sim \pi}[Q(s_t, a_t) - \log \pi(a_t|s_t)] \tag{A.8}$$

Lemma 1. *(Soft Policy Evaluation)* 考虑 *soft Bellman backup operator* $\mathcal{T}_\pi$ (A.7) 和一个映射 $Q_0 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$，且 $|\mathcal{A}| < \infty$，定义 $Q_{k+1} = \mathcal{T}_\pi Q_k$。那么序列 Q_k 会收敛到 π 的 *soft Q-value*，当 $k \to \infty$。

对于这个 soft policy evaluation 的证明也比较简单，就是把熵也看作是奖励的一部分，记作一个 r_π，那么对于 soft Q-value 的定义就变成了一个 Bellman 方程，和之前的 Q-value 没有区别了。

$$Q_\pi(s_t, a_t) = r_\pi(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p, a_{t+1} \sim \pi}[Q_\pi(s_{t+1}, a_{t+1})]$$

A.2.2 Soft Policy Improvement

对于 policy improvement 步骤，我们更新 policy 到 soft Q-value 的指数函数。这个更新可以保证在 soft value 上是一个改进。由于我们更喜欢可处理的 policy，我们将 policy 限制在一个集合 Π 中，这个集合可以是一个参数化的分布族，比如高斯分布。为了考虑到这个约束，我们将改进后的 policy 投影到所需的 policy 集合中。虽然原则上我们可以选择任何投影，但是使用基于 Kullback-Leibler 散度的信息投影会更方便。换句话说，在 policy improvement 步骤中，对于每个状态，我们根据下面的公式更新 policy：

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} D_{KL}\left(\pi'(\cdot|s_t) \| \frac{\exp(Q_{\pi_{\text{old}}}(s_t, \cdot))}{Z_{\pi_{\text{old}}}(s_t)}\right) \tag{A.9}$$

Lemma 2. *(Soft Policy Improvement)* 设 $\pi_{old} \in \Pi$，π_{new} 是优化问题(A.9)的最优解。对于所有的 $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$，都有 $Q_{\pi_{new}}(s_t, a_t) \geq Q_{\pi_{old}}(s_t, a_t)$，且 $|\mathcal{A}| < \infty$。

证明如下，首先我们有

$$\begin{aligned}
J_{\pi_{\text{old}}}(\pi_{\text{new}}(\cdot|s_t)) &= D_{KL}\left(\pi_{\text{new}}(\cdot|s_t) \| \frac{\exp(Q_{\pi_{\text{old}}}(s_t, \cdot))}{Z_{\pi_{\text{old}}}(s_t)}\right) \\
&= \mathbb{E}_{a_t \sim \pi_{\text{new}}}\left[\log \frac{\pi_{\text{new}}(a_t|s_t)}{\frac{\exp(Q_{\pi_{\text{old}}}(s_t, a_t))}{Z_{\pi_{\text{old}}}(s_t)}}\right] \\
&= \mathbb{E}_{a_t \sim \pi_{\text{new}}}[\log \pi_{\text{new}}(a_t|s_t) - Q_{\pi_{\text{old}}}(s_t, a_t) + \log Z_{\pi_{\text{old}}}(s_t)]
\end{aligned}$$

根据 (A.9)，我们知道

$$\pi_{\text{new}}(\cdot|s_t) = \arg\min_{\pi' \in \Pi} J_{\pi_{\text{old}}}(\pi'(\cdot|s_t))$$

则有

$$\begin{aligned}
& J_{\pi_{\text{old}}}(\pi_{\text{new}}(\cdot|s_t)) = \mathbb{E}_{a_t \sim \pi_{\text{new}}}[\log \pi_{\text{new}}(a_t|s_t) - Q_{\pi_{\text{old}}}(s_t, a_t) + \log Z_{\pi_{\text{old}}}(s_t)] \\
\leq \quad & J_{\pi_{\text{old}}}(\pi_{\text{old}}(\cdot|s_t)) = \mathbb{E}_{a_t \sim \pi_{\text{old}}}[\log \pi_{\text{old}}(a_t|s_t) - Q_{\pi_{\text{old}}}(s_t, a_t) + \log Z_{\pi_{\text{old}}}(s_t)]
\end{aligned}$$

又因为 $\log Z_{\pi_{\text{old}}}(s_t)$ 是与期望的下标 a_t 无关的常数，所以可以去掉。然后我们就有

$$\mathbb{E}_{a_t \sim \pi_{\text{new}}}[\log \pi_{\text{new}}(a_t|s_t) - Q_{\pi_{\text{old}}}(s_t, a_t)] \leq \mathbb{E}_{a_t \sim \pi_{\text{old}}}[\log \pi_{\text{old}}(a_t|s_t) - Q_{\pi_{\text{old}}}(s_t, a_t)]$$

注意到 (A.8)中的 Q 是没有下标的，于是我们可以假设我们给定 $Q = Q_{\pi_{\text{old}}}$，那么在给定策略 π 下的值函数为

$$V_\pi(s_t) = \mathbb{E}_{a_t \sim \pi}[Q_{\pi_{\text{old}}}(s_t, a_t) - \log \pi(a_t|s_t)]$$

则显然有

$$
\begin{aligned}
Q_{\pi_{\text{old}}}(s_t, a_t) &= r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}\left[V_{\pi_{\text{old}}}(s_{t+1})\right] \\
&= r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}\left[\mathbb{E}_{a_{t+1} \sim \pi_{\text{old}}}\left[Q_{\pi_{\text{old}}}(s_{t+1}, a_{t+1}) - \log \pi_{\text{old}}(a_{t+1}|s_{t+1})\right]\right] \\
&\leq r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}\left[\mathbb{E}_{a_{t+1} \sim \pi_{\text{new}}}\left[Q_{\pi_{\text{old}}}(s_{t+1}, a_{t+1}) - \log \pi_{\text{new}}(a_{t+1}|s_{t+1})\right]\right] \\
&= r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}\left[V_{\pi_{\text{new}}}(s_{t+1})\right] \\
&= Q_{\pi_{\text{new}}}(s_t, a_t)
\end{aligned}
$$

Commentary

这里的证明和 SAC 原论文 (Haarnoja et al., 2018) 的证明不太一样，这里是参考了 (Sutton and Barto, 2020) policy iteration 中 dynamic programming 的方法，假定之前的 Q 是固定的，然后考虑策略 π 的变化来希望通过 Bellman Equation 来使得新的 V 可以达到最大，也即

$$
v_{\pi_{\text{new}}} = \max_{\pi' \in \Pi} \pi' @ q_{\pi_{\text{old}}}
$$

放在我们这里就是定义为：

$$
V_{\pi_{\text{new}}}(s_t) = \mathbb{E}_{a_t \sim \pi_{\text{new}}}\left[Q_{\pi_{\text{old}}}(s_t, a_t) - \log \pi_{\text{new}}(a_t|s_t)\right]
$$

通过这样定义后就不用像原论文中得一直展开，如果按照原文的定义:

$$
V_{\pi_{\text{new}}}(s_t) = \mathbb{E}_{a_t \sim \pi_{\text{new}}}\left[Q_{\pi_{\text{new}}}(s_t, a_t) - \log \pi_{\text{new}}(a_t|s_t)\right]
$$

我们会发现展开之后非常麻烦，且没有办法最后归约得到 $Q_{\pi_{\text{new}}}$，原文只是提了一句说根据 lemma 1 可以直接得到，但不严谨。

A.2.3 Soft Policy Iteration

Theorem 1. *(Soft Policy Iteration)* 重复应用 *soft policy evaluation* 和 *soft policy improvement* 到任意的 *policy* $\pi \in \Pi$，收敛到一个 *policy* π^*，使得对于所有的 *policy* $\pi \in \Pi$ 和 $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$ 都有 $Q_{\pi^*}(s_t, a_t) \geq Q_\pi(s_t, a_t)$，假设 $|\mathcal{A}| < \infty$。

参考 (Sutton and Barto, 2020)，这个定理的证明和之前的定理类似，都是通过 Bellman Equation 来证明的。

A.3 Supplementary Material for GRPO

A.3.1 Derivation of Advantage Function

3.5.1 的 token-level A_t 根据(2.17)可以写作下面形式：

$$
A_t = \sum_{l=0}^{T-1-t} \gamma^l r_{t+l} - V_\psi(q, o_{<t})
$$

这里其实可以用 MC 估计来 value model，即：

$$V(\underbrace{q, o_{<t}}_{s_t}) = \mathbb{E}_{\pi_{\theta_{\text{old}}}}[Q(\underbrace{q, o_{\leq t}}_{s_t, a_t})] = \mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\sum_{l=0}^{T-1-t} \gamma^l r_{t+l}\right]$$

那么我们有就可以去掉 V_ψ：

$$A_t = \sum_{l=0}^{T-1-t} \gamma^l r_{t+l} - \mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\sum_{l=0}^{T-1-t} \gamma^l r_{t+l}\right]$$

再进一步的，我们可以采用 rule-based reward, 对生成的每一个 response 进行格式检查或者验证答案从而进行奖励 r_i ，这样就可以去掉 reward model，且变成了 response-level advantage：

$$A_i = r_i - \mathbb{E}_{\pi_{\theta_{\text{old}}}}[r_i]$$

注：GPG Chu et al. (2025) 采取的就是上面这个不除以标准差的形式，(Guo et al., 2025) 也采取的是除以标准差的形式；

进一步的，(Guo et al., 2025) 将这个 advantage 进行归一化处理：

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})}$$

A.3.2 Derivation of KL Approximation

参考之前章节2.1 KL 散度的无偏估计器 k_3 ，实际上我们的 kl 散度等于

$$\begin{aligned} D_{KL}(\pi_\theta \| \pi_{\theta_{\text{ref}}}) &= \mathbb{E}_{\pi_\theta}\left[\log \frac{\pi_\theta(o|q)}{\pi_{\theta_{\text{ref}}}(o|q)}\right] = \mathbb{E}_{\pi_\theta}\left[-\log \frac{\pi_{\theta_{\text{ref}}}(o|q)}{\pi_\theta(o|q)}\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\frac{\pi_{\theta_{\text{ref}}}(o|q)}{\pi_\theta(o|q)} - \log \frac{\pi_{\theta_{\text{ref}}}(o|q)}{\pi_\theta(o|q)} - 1\right] \end{aligned}$$

(3.13) 写的是一个 estimator，实际上如果使用 MC 方法，我们有：

$$D_{KL}(\pi_\theta \| \pi_{\theta_{\text{ref}}}) = \mathbb{E}_{q\sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(O|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\left(\frac{\pi_{\theta_{\text{ref}}}(o_i|q)}{\pi_\theta(o_i|q)} - \log \frac{\pi_{\theta_{\text{ref}}}(o_i|q)}{\pi_\theta(o_i|q)} - 1\right)\right]$$

我们如果要最大化 $J_{\text{GRPO}}(\theta)$，就要最小化 $D_{KL}(\pi_\theta \| \pi_{\theta_{\text{ref}}})$，所以我们可以将上面的式子作为 penalty term，即在前面加一个负号放进 $J_{\text{GRPO}}(\theta)$ 去就可以了，见3.14。

A.3.3 Derivation of Token-level GRPO

实际上 token-level GRPO (Shao et al., 2024) 在 DeepSeek-Math 中就已经出现了，但是效果不如 response-level GRPO (Guo et al., 2025) 在 DeepSeek-R1 上那么出圈。理论上我们可能觉得 token-level GRPO 更加的细粒度，而且可以针对每一个步骤给奖励优化，但是累计的奖励往往带来非常大的方差导致训练不稳定，这里依然有学习的意义，因此在这里给出推导过程。

对(3.14) 我们改成 token-level 的形式：

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(O|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(J_{i,t}^{\text{clip}}(\theta) - \beta \mathbb{D}_{i,t}^{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right)\right] \tag{A.10}$$

Clip Surrogate Objective

从 $J_i^{\text{clip}}(\theta)$ (3.11)变成了：

$$J_{i,t}^{\text{clip}}(\theta) = \min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right)$$

Ratio

从 $r_i(\theta)$ (3.10) 变成了：

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}$$

KL estimator

从 $\mathbb{D}_i^{\text{KL}}$ (3.13) 变成了：

$$\mathbb{D}_{i,t}^{\text{KL}}(\pi_\theta\|\pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - \log\frac{\pi_{\text{ref}}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - 1$$

Advantage and Reward

如果是 response-level reward，advantage 和之前 A_i (3.12)一样 (相当于一个 response 所有 token 的 advantage 都是一样的)：

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})}$$

如果是 process/step/turn-level reward, 那么我们会有一个 process reward model (PRM) 为 G 个 response 的 K_i 个步骤打分，那么会返回一系列奖励（一共有 $\sum_{i=1}^{G} K_i$ 个奖励）：

$$\mathbf{R} = \{\{r_1^{\text{index}(1)}, \cdots, r_1^{\text{index}(K_1)}\}, \cdots, \{r_G^{\text{index}(1)}, \cdots, r_G^{\text{index}(K_G)}\}\}$$

其中 index(j) 表示第 j 步骤的 end token index，对 rewards 进行 normalize：

$$\tilde{r}_i^{\text{index}(j)} = \frac{r_i^{\text{index}(j)} - \text{mean}(\mathbf{R})}{\text{std}(\mathbf{R})}$$

那么此时的 advantage 为 (相当于一个 response 中的 token 是分为几段的 advantage)

$$\hat{A}_{i,t} = \sum_{\text{index}(j)\geq t} \tilde{r}_i^{\text{index}(j)}$$

思考：Bellman Equation 去哪了？

发现其实在对于 $s_t = (q, o_{<t}), a_t = o_t$ 的关系而言，我们从来都没有重复利用到类似这样的过程：

$$V(s_t) = \mathbb{E}_{a_t\sim\pi}\left[Q(s_t, a_t)\right]$$
$$Q(s_t, a_t) = r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[V(s_t)\right]$$

我们目前一直没有利用好上一次处理 $s_t = (q, o_{<t})$ 时得到的历史信息，时间维度上的下一个状态 $s_{t+1} = (q, o_{\leq t})$ 并没有上一个状态 $s_t = (q, o_{<t})$ 传达某些信息（除了在预训练阶段的 cross-entropy loss 是因为 casuality 传达信息了），因为上面的 advantage 其实是通过一个 bandit 环境得到的 single-turn 的奖励归一化而已，并不是完整的 multi-turn RL；这也怪不得在最初始提出 GRPO (Shao et al., 2024) 的时候称之为 Supervision RL，自己理解为监督学习中的一种，里面没有涉及到探索学习新知识、利用经验的部分存在其实。