LLM 中的强化学习：从 Actor-Critic 到 RLHF / DPO / GRPO

Song mm

计算机学院
浙江大学

2025 年 12 月 25 日

提纲

提纲

今天的主线任务：RL 和 LLM

- AI 中训练模型可以拆成两个核心问题：
  - **模型结构**：怎么吃数据？
  - **损失函数**：你的目标是什么？
- 本次内容以**直觉与通俗解释**为主，数学推导只保留关键形式。
- 先复习从 **PG** 到 **Actor-Critic 框架**，然后介绍后续的 **TRPO / PPO** 改进。
- 最后讲**大模型的 RL**：**RLHF**、**DPO**、**GRPO** 在干什么。

提纲

Actor-Critic

基于值函数的策略迭代算法

- **Critic**：学一个值函数 $V(s)$ 去估计状态价值
- **Actor**：学一个策略函数 $\pi(a|s)$，我们的训练目标
- 所有函数皆可神经网络：策略函数网络 + 值函数网络，然后交替优化。

那么首先就有一个问题，这是一个 on-policy 还是 off-policy 的算法呢？

回答这个问题我们只需要看生成样本的策略和你更新的策略是不是一个策略，自己生成用来更新自己就是 on-ploicy，否则就是 off-policy。所以回答是：**Actor-Critic 可以是 on-policy 也可以是 off-policy**，但在后面为了样本效率，大家其实一般都是用的 **off-policy**。

Off-policy 的代价：重要性采样

- off-policy 下，数据来自行为策略 μ，但你要优化的是目标策略 π
- 因为采样分布不一致，期望/梯度不能直接用，需要用重要性采样“纠偏”

一般重要性采样：用 q 的样本估计 p 下的期望

$$\mathbb{E}_{x\sim p}[f(x)] = \int f(x)p(x)\,dx = \int f(x)\frac{p(x)}{q(x)}q(x)\,dx = \mathbb{E}_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right].$$

对应到 off-policy：用 μ 的样本估计 π 下的期望

令行为策略为 μ、目标策略为 π，则

$$\mathbb{E}_{(s,a)\sim d^{\pi}}[g(s,a)] = \mathbb{E}_{(s,a)\sim d^{\mu}}\left[\frac{d^{\pi}(s,a)}{d^{\mu}(s,a)}\,g(s,a)\right],$$

其中 $d^{\pi}(s,a)$ 表示在策略 π 下出现 (s,a) 的分布。

动作条件概率的 ratio

动作条件概率的 ratio

由于 $d^\pi(s)$ 难以直接估计，常用近似忽略状态分布比值：

$$\frac{d^\pi(s,a)}{d^\mu(s,a)} = \frac{\pi(a|s)d^\pi(s)}{\mu(a|s)d^\mu(s)} \approx \frac{\pi(a|s)}{\mu(a|s)}.$$

于是得到常见的重要性采样比率（ratio）：

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)}.$$

- 原理：**按“样本在目标策略下有多可能出现”进行加权**
- 注意：若 π 与 μ 差太大，上面的约等于就不能近似

从 Value Based 到 Policy Gradient

- 强化学习方法可分为两大类：
  - **基于值函数的方法**
  - **基于 PG 的方法**
- 基于值函数的方法学习一个 $Q(s,a)$，然后选择，不断的改进 Q 值估计最终学习到最优策略

$$a^* = \arg\max_a Q(s,a)$$

- 代表算法：Q-learning、SARSA、DQN

策略参数化

- 基于策略的方法直接学习策略

$$\pi_\theta(a|s)$$

- 用一个神经网络输出：

$$\pi_\theta(a|s) = \text{softmax}(f_\theta(s))$$

- 不再选“最优动作”，而是按概率采样：

$$a \sim \pi_\theta(\cdot|s)$$

- 概率大的动作更可能被选中

定义优化目标

- 一条轨迹

$$\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$$

- 在策略 π_θ 下出现该轨迹的概率

$$p_\theta(\tau) = p(s_0) \prod_{t=0}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

- 轨迹回报

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$$

- 最大化期望回报：

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

策略梯度推导

目标函数：

$$J(\theta) = \sum_{\tau} p_\theta(\tau) R(\tau)$$

对参数求导：

$$\nabla_\theta J(\theta) = \sum_{\tau} \nabla_\theta p_\theta(\tau) R(\tau)$$

使用对数导数技巧：

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

代入：

$$\nabla_\theta J(\theta) = \sum_{\tau} p_\theta(\tau) R(\tau) \nabla_\theta \log p_\theta(\tau)$$

可采样的策略梯度形式

由于

$$\log p_\theta(\tau) = \sum_{t=0}^{T} \log \pi_\theta(a_t|s_t)$$

最终得到：

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

使用梯度上升更新策略：

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

思想

PG 其实就是一个以**回报期望最大化**为优化目标的**牛顿解法**，关键是计算 Reward

REINFORCE：Monte Carlo 计算完整 Reward

- 思路：采样完整 rollout，算回报 G_t，再做梯度更新
- 优点：简单直接
- 缺点：
  - 纯采样估计期望 $\Rightarrow$ **方差大、训练不稳定**
  - 想稳定 $\Rightarrow$ 需要很多采样 $\Rightarrow$ **样本效率低、收敛慢**

改进想法

减小方差的方法，使用优势估计提到 Reward，不采样到底避免步骤太多产生的误差 (TD)

从 REINFORCE 到 Advantage

- REINFORCE 使用回报 $R(\tau)$，方差非常大
- Actor–Critic 用状态价值作为 baseline：

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$$

- 策略梯度变为：

$$\nabla J(\theta) = \mathbb{E}\Big[A(s_t, a_t)\nabla \log \pi_\theta(a_t|s_t)\Big]$$

疑问

减小方差可以理解，但为什么策略梯度这样没变？

Baseline 不改变策略梯度（Policy Gradient Invariance）

性质（Baseline Invariance）
对于任意仅依赖于状态的标量函数 $b(s)$，有

$$\nabla_\theta J(\theta) = \mathbb{E}_{S\sim\eta,\, A\sim\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(A|S)\; Q^\pi(S,A)\Big] = \mathbb{E}_{S\sim\eta,\, A\sim\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(A|S)\big(Q^\pi(S,A) - b(S)\big)\Big]$$

$$\mathbb{E}_{S,A}[\nabla_\theta \log \pi_\theta(A|S)\; b(S)] = 0$$

$$\begin{aligned}
&= \sum_s \eta(s) \sum_a \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)\; b(s) \\
&= \sum_s \eta(s) \sum_a \nabla_\theta \pi_\theta(a|s)\; b(s) \\
&= \sum_s \eta(s) b(s) \nabla_\theta \sum_a \pi_\theta(a|s) \\
&= \sum_s \eta(s) b(s) \nabla_\theta 1 = 0
\end{aligned}$$

两种极端的 Advantage 估计

一步 TD 误差：

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

$$\hat{A}_t^{TD} = \delta_t$$

Monte-Carlo 优势：

$$\hat{A}_t^{MC} = G_t - V(s_t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} - V(s_t)$$

偏差–方差权衡

- TD：低方差，但若 V 不准则有偏
- MC：无偏，但高方差

N-step Advantage：在 TD 与 MC 之间

$$\hat{A}_t^{(N)} = \left(\sum_{k=0}^{N-1} \gamma^k r_{t+k+1} + \gamma^N V(s_{t+N}) \right) - V(s_t)$$

- $N = 1$ ： TD(0)
- $N \to \infty$ ： Monte Carlo
- **增大 $N \to$ 偏差 $\downarrow$，方差 $\uparrow$**

我们需要一种方法，能连续地控制这个 trade-off

GAE：用 λ 控制偏差与方差

TD 误差：

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

广义优势估计：

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

- $\lambda = 0$：

$$\hat{A}_t = \delta_t \quad \text{(TD，有偏低方差)}$$

- $\lambda = 1$：

$$\hat{A}_t \approx G_t - V(s_t) \quad \text{(MC，无偏高方差)}$$

- $0 < \lambda < 1$：平衡偏差与方差

GAE 的实际计算（递归形式）

GAE 的定义

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

在计算上不实用，但可以用递归高效计算。

第 1 步：计算 TD 误差

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

若 s_{t+1} 是终止状态，则定义 $V(s_{t+1}) = 0$。

第 2 步：从序列末尾反向递推

设序列长度为 T，初始化

$$\hat{A}_T^{GAE} = 0$$

然后对 $t = T-1, \ldots, 0$ 递归：

$$\hat{A}_t^{GAE} = \delta_t + \gamma\lambda \hat{A}_{t+1}^{GAE}$$

提纲

Actor-Critic 还有什么不好的地方？

Actor–Critic 给出了梯度方向：

$$\nabla_\theta J(\theta) = \mathbb{E}[A(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)]$$

用学习率做参数更新：

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta_k)$$

问题：步长 α 怎么选？

两种典型失败模式

- **Overshooting**：步子太大，越过最优策略，掉入次优区域
- **Undershooting**：步子太小，收敛极慢

为什么 RL 中 overshooting 特别危险？

监督学习中数据固定，下一轮可以纠正；但在强化学习中，**策略变了，数据分布也变了**。一次错误的大更新可能导致未来只看到“烂数据”，形成灾难性正反馈，无法恢复。

自然策略梯度

朴素想法：限制参数更新幅度防止 overshooting

$$\|\theta_{\text{new}} - \theta_{\text{old}}\|_2 \leq \epsilon$$

直觉：只要参数变化不大，策略就不会“变太多”。

但这不是一个好方法

不同策略分布对参数变化的**敏感度不同**。相同的欧氏距离变化，可能导致：

- 某些策略分布几乎不变
- 某些策略分布发生巨大变化

更合理的想法：限制“策略分布本身”的变化幅度就像限制欧氏距离一样：

$$D(\pi_{\theta_{\text{old}}}, \pi_{\theta_{\text{new}}}) \leq \delta$$

其中 $D(\cdot,\cdot)$ 是分布之间的距离/散度。实践中最常用的是 KL 散度：

$$D_{\text{KL}}(\pi_{\theta_{\text{old}}} \parallel \pi_{\theta_{\text{new}}}) \leq \delta$$

自然策略梯度

目标：在限制策略变化的前提下最大化回报

$$\max_{\theta} \; J(\theta) \quad \text{s.t.} \quad D_{\mathrm{KL}}(\pi_{\theta_{\text{old}}} \,\|\, \pi_{\theta}) \le \delta$$

用拉格朗日松弛转为无约束问题

$$\max_{\theta} \; J(\theta) - \beta \, D_{\mathrm{KL}}(\pi_{\theta_{\text{old}}} \,\|\, \pi_{\theta})$$

在 θ_{old} 附近作泰勒展开一阶展开目标函数，二阶展开 KL 散度，可得到近似问题：

$$\max_{\Delta\theta} \; g^{\top}\Delta\theta - \frac{\beta}{2}\Delta\theta^{\top} F \Delta\theta$$

其中

$$g = \nabla_{\theta} J(\theta), \qquad F = \text{Fisher 信息矩阵}$$

最优解（自然策略梯度方向）

$$\Delta\theta \propto F^{-1} g$$

$$\tilde{\nabla} J(\theta) = F^{-1} \nabla J(\theta)$$

自然策略梯度算法更新公式

更新公式（由 KL 约束推出步长）

设 $g = \nabla_\theta J(\theta_{\text{old}})$，$F$ **为 Fisher 矩阵，自然梯度方向** $d = F^{-1}g$。

$$\Delta\theta = \alpha d, \qquad D_{\text{KL}} \approx \frac{1}{2}\Delta\theta^\top F \Delta\theta \le \delta$$

$$\alpha = \sqrt{\frac{2\delta}{g^\top F^{-1} g}}, \qquad \Delta\theta = \sqrt{\frac{2\delta}{g^\top F^{-1} g}}\, F^{-1} g$$

$$\theta_{\text{new}} = \theta_{\text{old}} + \Delta\theta$$

TRPO：为什么自然策略梯度还不够？

自然策略梯度（NPG）试图通过 KL 约束控制更新幅度，但在实践中存在问题：

NPG 的三个核心缺陷

- **KL 约束可能失效**二阶近似可能不准，真实 KL 可能大于设定阈值
- **计算代价高** Fisher 矩阵是 $|\theta| \times |\theta|$，求逆代价 $O(|\theta|^3)$
- **没有保证真的变好**大量近似导致即使沿自然梯度走，回报也可能下降

TRPO 的目标：**给出一个“每步都保证改进”的更新规则**。

TRPO：策略改进的数学表达

TRPO 从两个策略的期望回报差开始：

$$\eta(\pi') - \eta(\pi)$$

通过优势函数，可以写成：

$$\eta(\pi') - \eta(\pi) = \sum_s \rho_{\pi'}(s) \sum_a \pi'(a|s) A_\pi(s,a)$$

其中优势函数定义为：

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

由于 $\rho_{\pi'}(s)$ 难以计算，用 $\rho_\pi(s)$ 近似：

$$\eta(\pi') - \eta(\pi) \approx \sum_s \rho_\pi(s) \sum_a \pi'(a|s) A_\pi(s,a)$$

TRPO: Surrogate Advantage 与单调改进

将上式写成期望并引入重要性采样:

$$L_\pi(\pi') = \mathbb{E}_{s\sim\rho_\pi,\, a\sim\pi}\left[\frac{\pi'(a|s)}{\pi(a|s)} A_\pi(s,a)\right]$$

TRPO 证明:

$$\eta(\pi') - \eta(\pi) \geq L_\pi(\pi') - C \max_s D_{\mathrm{KL}}(\pi(\cdot|s)\|\pi'(\cdot|s))$$

其中常数 C 是 TRPO 论文算出来的，但太复杂不看。

因此 TRPO 求解:

$$\max_{\pi'} L_\pi(\pi') \quad \text{s.t.} \quad D_{\mathrm{KL}}(\pi(\cdot|s)\|\pi'(\cdot|s)) \leq \delta$$

单调改进定理

如果我们提高这个下界，则真实回报 $\eta(\pi')$ 也至少提高同样多。

提纲

PPO：别推导了，直接做个好用近似

- TRPO 告诉我：**优势增益被 KL 限制**
- 那我就：
  - 直接用 KL 做惩罚（Penalty）
  - 或者干脆把 $L_\pi(\pi')$ 中的重要性权重裁剪（Clip）
- 这就是 PPO 系列“各种变种”的 idea

Clip 为什么能 Work，甚至成为现在主流

大概思路就是 KL 散度计算和重要性采样的式子超像，而且像激活函数，AI 太熟悉了

PPO-Penalty：自适应 KL 惩罚系数

- 设定目标 KL 阈值
- KL 太大（更新太猛）⇒ 增大惩罚系数，让变化变小
- KL 太小（更新太保守）⇒ 减小惩罚系数，让变化别太小
- 超参依然存在：阈值、倍数规则等

优化目标形式

$$\max_{\theta} \mathbb{E}\left[L(\theta) - \beta \mathrm{KL}(\pi_{\mathsf{old}} || \pi_{\theta})\right]$$

PPO-Clip：重要性采样“ratio 裁剪”

- 观察：重要性采样 ratio 的变化和 KL “看着差不多”
- 于是直接裁剪 ratio

$$L^{\text{clip}}(\theta) = \mathbb{E}\Big[\min\big(r_t(\theta)A_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\big)\Big]$$

- 常见 $\epsilon \approx 0.2$（经验值），其中 $(r_t(\theta)$ 是重要性采样系数

提纲

RLHF 是什么？

Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback

- 关键词：**用人类偏好构建损失函数**来微调模型
- 直觉：如果最终给人用，优化目标就应体现人的偏好

- 为什么以前不这么干？
  - 预训练目标天然是 next-token prediction，不需要人标
  - 大模型微调很贵：算力 + 高质量偏好数据更贵

RLHF 训练 pipeline

1. **SFT**：先做监督微调，让模型能“像样回答”
2. **RM/PM**：收集偏好对比数据，训练 Reward/Preference Model
3. **RL**：模型 RM 提供的 reward 下继续优化策略

总结

RLHF $\approx$ “给大模型套上 PPO”，关键在 **reward model**。

为什么要 Reward/Preference Model？

- 人很难对单条回答给一致的绝对分数（5 分/7 分不统一）
- 但更容易做**相对偏好**：A 比 B 好
- 所以训练 PM 用二元比较数据：(x, y^+, y^-)
- 一个二元的偏好概率 (BT 模型)：

$$P(y^+ \succ y^- \mid x) = \frac{\exp\big(r_\phi(x, y^+)\big)}{\exp\big(r_\phi(x, y^+)\big) + \exp\big(r_\phi(x, y^-)\big)} = \sigma\big(r_\phi(x, y^+) - r_\phi(x, y^-)\big)$$

- 直觉：**让模型学会“隐式评分标准”**

把 RL 放进“文本生成”的定义里

- 状态 s：当前上下文（历史 tokens）
- 动作 a：下一个 token
- 动作空间：词表大小（万级；有的更大）
- 轨迹：从 prompt 开始一路生成到 EOS / max_len
- Reward：由 RM/规则项等给出（不同算法差异从这里开始）

RLHF 的优化目标

- 目标：尽可能在 RM 打分体系中得到更高分数，同时别偏离参考模型太多
- $$\max_{\theta} \ \mathbb{E}\big[r_\phi(x,y)\big] - \beta\,\mathbb{E}\big[\mathrm{KL}(\pi_\theta(\cdot|x)\|\pi_{\mathsf{ref}}(\cdot|x))\big]$$
- 是不是很像 PPO 的公式？
- RLHF 的难点之一：PM/RM 训练不稳 $\Rightarrow$ PPO 也会崩

优化动机

能不能跳过训练 PM / Critic，直接用偏好数据，把策略训练出来 ??

提纲

偏好对齐的总优化目标

不论 PPO 还是 DPO，偏好对齐阶段的总目标相同：

$$\max_{\pi_\theta} \mathbb{E}_{(x,y)\sim\pi_\theta}\left[r_\phi(x,y) - \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\right]$$

其中：

- π_θ：要训练的对齐模型
- r_ϕ：奖励模型
- π_{ref}：参考模型（SFT）

目标：**绕过奖励模型，直接训练** π_θ

固定奖励函数 r 下的最优对齐模型

固定 r，对上式求最优解：

$$\pi_r^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

其中配分函数

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

这给出了在任意奖励函数 r 下的最优对齐模型的显式解。

从 π_r^* 反推奖励函数

由

$$\pi_r^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\tfrac{1}{\beta}r(x,y)\right)$$

可解出：

$$r(x,y) = \beta\log\frac{\pi_r^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta\log Z(x)$$

由于 $Z(x)$ 与 π_θ 无关，可忽略。因此：**奖励函数由对齐模型隐式表示**。

成对偏好: Bradley–Terry 模型

数据形式: $\langle x, y_w, y_l \rangle$ (y_w=chosen, y_l=reject)
BT 模型:

$$P(y_w \succ y_l) = \frac{e^{r(x,y_w)}}{e^{r(x,y_w)} + e^{r(x,y_l)}}$$

奖励模型的极大似然目标:

$$\max_r \ \mathbb{E}\Big[\log \sigma(r(x, y_w) - r(x, y_l))\Big]$$

代入 $r(x,y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$ **得到 DPO**:

$$\max_{\pi_\theta} \mathbb{E}\left[\log \sigma\Big(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\Big)\right]$$

多回答偏好：Plackett–Luce 模型

标注排序：

$$\tau = (y_1 \succ y_2 \succ \cdots \succ y_K)$$

PT **概率**：

$$P(\tau) = \prod_{k=1}^{K} \frac{e^{r(x,y_k)}}{\sum_{j\geq k} e^{r(x,y_j)}}$$

代入 $r(x,y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$ **得到多回答 DPO 目标**：

$$\max_{\pi_\theta} \mathbb{E} \sum_{k=1}^{K} \log \frac{\left(\frac{\pi_\theta(y_k|x)}{\pi_{\text{ref}}(y_k|x)}\right)^\beta}{\sum_{j\geq k} \left(\frac{\pi_\theta(y_j|x)}{\pi_{\text{ref}}(y_j|x)}\right)^\beta}$$

DPO 的核心思想

- RLHF：训练 $r_\phi \to$ PPO 优化 π_θ
- DPO：用 π_θ 隐式表示 r，直接优化 π_θ

Your Language Model is secretly a Reward Model

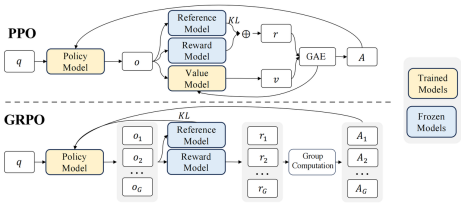
BT / PT 偏好模型 + 显式解　绕过奖励模型　直接从偏好数据训练对齐模型

提纲

GRPO：我不懂具体多好是好，我比大多数好就行

- Critic / PM 训练更新太麻烦，我搞不懂
- 那我直接：
  1. 对同一个 prompt **采样多条** response
  2. 用 reward model（或规则）给每条打分
  3. 在组内算“相对优势”做更新
- 组内优势：

$$\hat{A}_i = \frac{r_i - \mu(r)}{\sigma(r) + \epsilon}$$



GRPO 的优点与问题

优点

- 简化结构：**不显式训练 critic**
- 思路清晰：组内相对比较，容易实现

问题

- 组内 reward 全一样（比如全 0）⇒ 学不到东西
- 依赖多次采样 ⇒ 吃显存/吃时间
- 把奖励平均到 token 上 ⇒ 长序列优势可能被稀释
- reward 含惩罚项时：可能短但错的反而更“划算”

提纲

总结回顾

- Actor-Critic：**结构** = actor + critic；**目标** = 用优势更新策略
- GAE：在 MC/TD 间折中，降低方差、提升稳定性
- TRPO/PPO：核心是**限制策略更新幅度**
- **大模型对齐**：
  - RLHF：**RM + PPO**
  - DPO：**直接用偏好优化策略**
  - GRPO：**删 critic，用组内相对优势**

Q&A

谢谢!

提纲

参考文献

- 知乎用户 8Gs7DZ, *https://zhuanlan.zhihu.com/p/614115887*.
- Schulman et al., *Trust Region Policy Optimization*, 2015.
- Schulman et al., *Proximal Policy Optimization Algorithms*, 2017.
- Training a Helpful and Harmless Assistant with RL from Human Feedback.
- Rafailov et al., *Direct Preference Optimization*, 2023.
- Group-based policy optimization style methods.