

PPO 完整推导与设计动机

含 Actor 与 Critic 学习机制：从策略梯度、PPO-Clip、GAE 到机器人控制

2026 年 10 月 18 日

目录

第一部分	从预期回报到策略梯度	2
第二部分	Baseline、Value 与 Advantage	3
第三部分	性能差分、Surrogate Objective 与重要性采样	4
第四部分	TRPO：用 Trust Region 控制更新	6
第五部分	PPO-Clip 的核心推导	7
第六部分	GAE：Advantage 究竟如何估计？	9
第七部分	完整 Actor-Critic 损失	13
第八部分	连续动作 PPO 与机器人控制	15
第九部分	一次 PPO Iteration 的完整执行过程	17
第十部分	超参数与实际诊断	19
第十一部分	与 REINFORCE、Actor-Critic、TRPO 的关系	20
第十二部分	最终公式总览与研究启示	20

课程首页 · English · PDF

项目	说明
主题	PPO
修订日期	2026-10-19
许可	CC BY-NC-SA 4.0 · Xiong Zeng and contributors

第一部分 从预期回报到策略梯度

阅读目标：逐步理解 PPO 为什么这样设计，而不仅仅是记住其损失函数。本文覆盖完整数学推导、各步骤的设计动机、数值示例、实现要点，以及在机器人连续控制与残差策略学习中的用法。

核心思想：提高好动作的概率、降低坏动作的概率，同时避免一次策略更新过大，使得根据旧策略数据建立的局部性能模型仍然可信。

阅读路线：回报最大化 → Policy Gradient → Baseline / Advantage → 性能差分与重要性采样 → TRPO → PPO-Clip → GAE → Actor-Critic 完整损失 → 机器人应用与实现。

记号约定：主要考虑折扣无限时域 MDP, $\gamma \in [0, 1]$; $J(\theta)$ 表示从初始状态分布出发的折扣回报。涉及实际 PPO 的时间步平均时，会明确说明与严格折扣占用分布之间的区别。 $\hat{A}_t$ 是利用旧策略 Rollout 计算、在本轮策略更新中固定的 Advantage 估计。

学习主线：Critic 根据固定的 Bootstrap/GAE Target 学会估计长期回报 → 用 TD Error / GAE 为旧动作计算相对优劣 → Actor 根据优势梯度更新动作概率分布 → PPO-Clip 抑制过度策略变化 → 新策略重新采样。**Critic 做回归，Actor 做动作分布的带裁剪策略优化；二者通过固定的 Advantage 信号协同。**

1. 定义强化学习优化问题

考虑马尔可夫决策过程 (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0).$$

其中 $s_t \in \mathcal{S}$ 为状态, $a_t \in \mathcal{A}$ 为动作, $P(s_{t+1} | s_t, a_t)$ 为环境转移核, r_t 为奖励, ρ_0 是初始状态分布, $\pi_\theta(a | s)$ 是参数为 θ 的随机策略。学习目标是

$$\theta^* = \arg \max_{\theta} J(\theta), \quad J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]. \quad (1)$$

轨迹记为 $\tau = (s_0, a_0, s_1, a_1, \dots)$, 其折扣回报为 $R(\tau) = \sum_t \gamma^t r_t$ 。于是

$$J(\theta) = \int p_{\theta}(\tau) R(\tau) d\tau.$$

动机与困难： θ 首先改变动作分布，再通过环境动力学影响整条未来轨迹：

$$\theta \longrightarrow a_0 \longrightarrow s_1 \longrightarrow a_1 \longrightarrow \dots \longrightarrow R(\tau).$$

对于碰撞、摩擦、接触切换和未知动力学, $R(\tau)$ 一般不能简单地通过环境对 θ 反向传播。因此需要一种不要求环境动力学可微的梯度估计方法。

2. 利用 Log-Derivative Trick 推导 Policy Gradient

对期望求梯度：

$$\nabla_{\theta} J(\theta) = \int \nabla_{\theta} p_{\theta}(\tau) R(\tau) d\tau.$$

使用恒等式

$$\nabla_{\theta} p_{\theta}(\tau) = p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau), \quad (2)$$

得到

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}} [R(\tau) \nabla_{\theta} \log p_{\theta}(\tau)]. \quad (3)$$

这称为 Likelihood-Ratio / Score-Function Trick。它把概率密度的梯度转为对数概率的梯度。下一个关键点是：MDP 轨迹概率恰好可以分解。

$$p_{\theta}(\tau) = \rho_0(s_0) \prod_{t=0}^{\infty} \pi_{\theta}(a_t | s_t) P(s_{t+1} | s_t, a_t).$$

取对数并求导。如果初始分布、环境转移核不依赖 θ ，则

$$\begin{aligned} \log p_{\theta}(\tau) &= \log \rho_0(s_0) + \sum_t \log \pi_{\theta}(a_t | s_t) + \sum_t \log P(s_{t+1} | s_t, a_t), \\ \boxed{\nabla_{\theta} \log p_{\theta}(\tau)} &= \boxed{\sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)}. \end{aligned} \quad (4)$$

所以形式上有

$$\boxed{\nabla_{\theta} J(\theta) = \mathbb{E} \left[R(\tau) \sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right]}. \quad (5)$$

这里严格处理无穷时域时需要适当的可积性与极限交换条件；也可以先在有限时域推导后取极限。

这一转换解决了什么？ 我们只需要对策略网络的 $\log \pi_{\theta}(a_t | s_t)$ 求梯度。机器人动力学和奖励仍然只用于前向采样，不必可微。

3. 因果性与 Reward-to-Go

式 (5) 的每个动作梯度都乘以整条轨迹回报。但 a_t 不能影响发生在 t 以前的奖励。利用 score function 的零均值性质，过去奖励对应的交叉项期望为零，于是可改写成

$$G_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_k, \quad \boxed{\nabla_{\theta} J(\theta) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) G_t \right]}. \quad (6)$$

动机： 对 a_t 的更新应由它能够影响的未来后果决定。去除过去奖励通常可以降低估计方差，而不改变梯度期望。式中的 γ^t 与原始折扣目标严格一致；实际 PPO 代码经常直接对采样时间步平均，属于相应采样分布下的优化约定，不应混为完全相同的无穷时域表达式。

第二部分 Baseline、Value 与 Advantage

4. 为什么可以减去 Baseline?

虽然使用 Reward-to-Go 比整条轨迹回报更合理，但 G_t 仍会因未来随机性而显著波动。考虑引入只依赖状态的 Baseline $b(s_t)$ ：

$$g_t = \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) [G_t - b(s_t)].$$

证明它不改变策略梯度的期望：

$$\begin{aligned}
& \mathbb{E}_{a \sim \pi_\theta(\cdot|s)} [\nabla_\theta \log \pi_\theta(a|s) b(s)] \\
&= b(s) \int \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) da \\
&= b(s) \int \nabla_\theta \pi_\theta(a|s) da \\
&= b(s) \nabla_\theta \int \pi_\theta(a|s) da \\
&= b(s) \nabla_\theta 1 = 0.
\end{aligned} \tag{7}$$

因此

$$\boxed{\mathbb{E}[\nabla \log \pi_\theta G_t] = \mathbb{E}[\nabla \log \pi_\theta (G_t - b(s_t))]} \tag{8}$$

动机：去除与“当前动作相对质量”无关的整体回报水平，从而减小随机波动。要注意：不是任何 Baseline 都能保证减小方差；严格的最优方差 Baseline 通常还与 score gradient 的平方范数有关。 $V^\pi(s)$ 是实用而自然的选择。

5. 为什么使用 Value Function 作为 Baseline?

定义状态价值、动作价值和优势函数：

$$\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi[G_t | s_t = s], \\
Q^\pi(s, a) &= \mathbb{E}_\pi[G_t | s_t = s, a_t = a], \\
\boxed{A^\pi(s, a)} &= \boxed{Q^\pi(s, a) - V^\pi(s)}.
\end{aligned} \tag{9}$$

$V^\pi(s)$ 衡量“从该状态按现有策略继续执行的平均回报”； $Q^\pi(s, a)$ 衡量“先执行指定动作再按策略继续执行的平均回报”。因此：

- $A^\pi(s, a) > 0$ ：这个动作比当前策略的平均动作更好，应倾向于提高其概率；
- $A^\pi(s, a) < 0$ ：这个动作比平均动作更差，应倾向于降低其概率；
- $A^\pi(s, a) = 0$ ：在该估计下没有相对优劣信号。

在准确的 Q^π, V^π 下，得到 Advantage Policy Gradient：

$$\boxed{\nabla_\theta J(\theta) = \mathbb{E} \left[\sum_t \gamma^t \nabla_\theta \log \pi_\theta(a_t | s_t) A^{\pi_\theta}(s_t, a_t) \right]} \tag{10}$$

从而引出 Actor-Critic：Actor π_θ 负责产生动作；Critic V_ϕ 近似状态价值；Advantage 连接二者。使用不准确的 Value Baseline 本身不会改变理想 score-function 公式的期望，但使用自举构造 Advantage 时会引入额外估计偏差。

第三部分 性能差分、Surrogate Objective 与重要性采样

6. 旧数据为什么不能无限重复使用?

Policy Gradient 假设采样数据与当前策略一致。若使用 π_{θ_0} 采样后更新到 $\pi_{\theta_1}, \pi_{\theta_2}$ ，旧 Rollout 依然服从 π_{θ_0} ，并不服从新策略。每次梯度步都重新采样又非常昂贵。

PPO 的目标之一是：在**一批由旧策略采集的数据上，进行多次相对安全的一阶更新**。要做到这一点，必须区分状态分布的变化和给定状态下动作分布的变化。

7. Performance Difference Lemma 的逐步证明

定义归一化折扣状态访问分布：

$$d_{\pi}(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s \mid \pi). \quad (11)$$

对连续状态空间，上式应理解为对应概率测度/密度。对于任意新旧策略，性能差分引理给出**精确恒等式**：

$$J(\pi_{\theta}) - J(\pi_{\text{old}}) = \frac{1}{1 - \gamma} \mathbb{E}_{\substack{s \sim d_{\pi_{\theta}} \\ a \sim \pi_{\theta}}} [A^{\pi_{\text{old}}}(s, a)]. \quad (12)$$

证明：由旧策略的 Bellman 方程，

$$A^{\pi_{\text{old}}}(s_t, a_t) = \mathbb{E}[r_t + \gamma V^{\pi_{\text{old}}}(s_{t+1}) - V^{\pi_{\text{old}}}(s_t) \mid s_t, a_t].$$

沿**新策略**的轨迹求折扣和：

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

后两个级数望远镜相消，只剩 $-V^{\pi_{\text{old}}}(s_0)$ （假设末端项消失），因此

$$\mathbb{E}_{\pi_{\theta}} \left[\sum_t \gamma^t A^{\pi_{\text{old}}}(s_t, a_t) \right] = J(\pi_{\theta}) - J(\pi_{\text{old}}).$$

再用 $d_{\pi_{\theta}}$ 的定义重写折扣时间求和，即得到式 (12)。

重要直觉：新策略的回报提升，等于它在自身将访问的状态、执行的动作上，取得的“相对于旧策略”的 Advantage 总和。

8. 将真实性能改进近似为 Surrogate Objective

式 (12) 的困难在于，期望中的状态来自新策略：

$$s \sim d_{\pi_{\theta}}.$$

但我们手上只有旧策略数据。如果新旧策略足够接近，可以作**局部近似**：

$$d_{\pi_{\theta}}(s) \approx d_{\pi_{\text{old}}}(s).$$

忽略正常数 $1/(1 - \gamma)$ 后，定义

$$L_{\text{sur}}(\theta) = \mathbb{E}_{\substack{s \sim d_{\pi_{\text{old}}} \\ a \sim \pi_{\theta}}} [A^{\pi_{\text{old}}}(s, a)]. \quad (13)$$

它比直接优化新策略的整条轨迹分布容易得多：状态分布固定，只需要研究给定状态下动作概率的变化。在旧参数附近，准确 Advantage 对应的 Surrogate 与真实目标具有相同的一阶局部方向（差一个正常数）。

核心前提：只有当 π_{θ} 不远离 π_{old} 时，这种状态分布替换才比较可信。这为后面的 Trust Region 和 Clipping 提供了动机。

9. Importance Sampling 推导 Probability Ratio

式 (13) 仍然包含 $a \sim \pi_\theta$ ，而数据中的动作来自旧策略。给定状态 s ，利用重要性采样：

$$\begin{aligned}\mathbb{E}_{a \sim \pi_\theta}[A^{\pi_{\text{old}}}(s, a)] &= \int \pi_\theta(a | s) A^{\pi_{\text{old}}}(s, a) da \\ &= \int \pi_{\text{old}}(a | s) \frac{\pi_\theta(a | s)}{\pi_{\text{old}}(a | s)} A^{\pi_{\text{old}}}(s, a) da.\end{aligned}$$

定义概率比：

$$r_\theta(s, a) = \frac{\pi_\theta(a | s)}{\pi_{\text{old}}(a | s)}. \quad (14)$$

于是得到 CPI (Conservative Policy Iteration) 形式的代理目标：

$$L^{\text{CPI}}(\theta) = \mathbb{E}_{\substack{s \sim d_{\pi_{\text{old}}} \\ a \sim \pi_{\text{old}}}} [r_\theta(s, a) A^{\pi_{\text{old}}}(s, a)]. \quad (15)$$

实际估计写为 $\mathbb{E}_t[r_t(\theta) \hat{A}_t]$ 。

概率比的解释：若旧动作概率为 0.2、新概率为 0.3，则 $r = 1.5$ ；若新概率变为 0.1，则 $r = 0.5$ 。正 Advantage 鼓励 r 上升，负 Advantage 鼓励 r 下降。

边界与限制：上述重要性采样要求新策略对旧策略概率为零的动作不赋正概率（绝对连续性）；对于一般 Gaussian 分布，通常满足动作密度支持条件。重要性采样仅精确修正了给定状态下的动作分布，并未修正 d_{π_θ} 与 $d_{\pi_{\text{old}}}$ 的差异。因此，它没有消除式 (13) 的状态分布近似误差。

第四部分 TRPO：用 Trust Region 控制更新

10. 直接最大化 CPI 为什么危险？

对于正 Advantage 样本， $r_t \hat{A}_t$ 随 r_t 增长：若 $\hat{A} = 5$ ， $r = 1.2, 2, 10$ 对应 6, 10, 50。代理目标似乎鼓励无限放大好动作的概率。但这些 Advantage 是根据旧策略计算的；如果新策略改变很大，旧估计不一定再可信。

尤其在机器人操作中，动作分布的一次较大改变可能引发接触切换、掉落甚至完全不同的未来状态访问。故 L^{CPI} 增大 **不保证** 真实 J 增大。

11. KL Divergence 与 TRPO 约束

使用 KL Divergence 衡量同一状态下两策略的分布差异：

$$D_{\text{KL}}(\pi_{\text{old}} \| \pi_\theta) = \mathbb{E}_{a \sim \pi_{\text{old}}} \left[\log \frac{\pi_{\text{old}}(a | s)}{\pi_\theta(a | s)} \right].$$

TRPO 的实用优化形式是

$$\begin{aligned} \max_{\theta} \quad & \mathbb{E}_t[r_t(\theta) \hat{A}_t] \\ \text{s.t.} \quad & \mathbb{E}_{s \sim d_{\pi_{\text{old}}}} [D_{\text{KL}}(\pi_{\text{old}}(\cdot | s) \| \pi_\theta(\cdot | s))] \leq \delta. \end{aligned} \quad (16)$$

δ 决定信赖域大小。其动机不是“策略改变越小越好”，而是在一个代理目标足够可信的局部区域内寻找改进。

TRPO 的理论分析给出含有最大状态策略差异惩罚项的性能下界，典型结构为

$$J(\pi_\theta) \geq J(\pi_{\text{old}}) + \frac{L_{\text{sur}}(\theta)}{1-\gamma} - C \max_s D_{\text{KL}}(\pi_{\text{old}} \parallel \pi_\theta), \quad (17)$$

其中 C 依赖回报/Advantage 的界和 γ ；具体常数以及 KL 方向取决于采用的界。**注意：**理论中的最坏状态差异控制，强于工程实现的平均 KL 约束；实际神经网络 TRPO 不能据此保证每次更新单调提升。

12. TRPO 局部二阶近似与 Natural Gradient

令 $\Delta\theta = \theta - \theta_{\text{old}}$ 。在旧参数附近进行泰勒展开：

$$L^{\text{CPI}}(\theta) \approx L^{\text{CPI}}(\theta_{\text{old}}) + g^\top \Delta\theta, \quad g = \nabla_\theta L^{\text{CPI}}|_{\theta_{\text{old}}}.$$

KL 在相同策略处为零且一阶梯度为零，二阶近似为

$$\bar{D}_{\text{KL}} \approx \frac{1}{2} \Delta\theta^\top F \Delta\theta,$$

其中 F 是在旧参数处计算的 Fisher 信息矩阵。为避免歧义，先定义 $u_{\text{old}}(s, a) = \nabla_\theta \log \pi_\theta(a | s)|_{\theta=\theta_{\text{old}}}$ ，则

$$F = \mathbb{E}_{s \sim d_{\pi_{\text{old}}}, a \sim \pi_{\text{old}}} [u_{\text{old}}(s, a) u_{\text{old}}(s, a)^\top].$$

局部问题变成

$$\boxed{\max_{\Delta\theta} g^\top \Delta\theta \quad \text{s.t.} \quad \frac{1}{2} \Delta\theta^\top F \Delta\theta \leq \delta.} \quad (18)$$

写出 Lagrangian：

$$\mathcal{L}(\Delta\theta, \eta) = g^\top \Delta\theta - \eta \left(\frac{1}{2} \Delta\theta^\top F \Delta\theta - \delta \right).$$

一阶条件 $g - \eta F \Delta\theta = 0$ 导致 $\Delta\theta = \eta^{-1} F^{-1} g$ 。代入活动约束（假设 F 正定、 $g \neq 0$ ）得

$$\boxed{\Delta\theta^* = \sqrt{\frac{2\delta}{g^\top F^{-1} g}} F^{-1} g.} \quad (19)$$

方向 $F^{-1}g$ 正是 Natural Policy Gradient 的核心形式。实际训练不能显式反演一个巨大矩阵，因此需要 Fisher-vector product、共轭梯度、线搜索等技术；非正定或病态情况下还需阻尼。**PPO 的设计动机：**保留保守策略更新的思想，但避免 TRPO 的复杂二阶/约束优化。

第五部分 PPO-Clip 的核心推导

13. 从 KL Penalty 到 Clipped Surrogate

最直接的一阶化方式是 PPO-Penalty：

$$L^{\text{KLPEN}}(\theta) = \mathbb{E}_t [r_t(\theta) \hat{A}_t] - \beta \mathbb{E}_t [D_{\text{KL}}(\pi_{\text{old}} \parallel \pi_\theta)]. \quad (20)$$

但 β 太小会限制不足，太大则更新过慢，还需要调整罚系数。PPO-Clip 改为限制“过大的概率比带来的额外目标收益”。设

$$\text{clip}(r, 1 - \epsilon, 1 + \epsilon) = \min\{1 + \epsilon, \max\{r, 1 - \epsilon\}\}.$$

最常用的 PPO 目标为

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right]. \quad (21)$$

常见起点为 $\epsilon = 0.1 \sim 0.2$ ，但应当根据环境、动作维度和监测的 KL 调整。这里的 **min** 与 **clip** 必须一起理解。

14. 正 Advantage：好动作不能无限被奖励

若 $\hat{A}_t > 0$ ，希望提高该动作概率，即使 $r_t > 1$ 。式 (21) 化简为

$$L_t^{\text{CLIP}} = \min(r_t, 1 + \epsilon) \hat{A}_t = \begin{cases} r_t \hat{A}_t, & r_t \leq 1 + \epsilon, \\ (1 + \epsilon) \hat{A}_t, & r_t > 1 + \epsilon. \end{cases} \quad (22)$$

设 $\hat{A}_t = 2, \epsilon = 0.2$:

r_t	原始 $r_t \hat{A}_t$	PPO 目标
0.8	1.6	1.6
1.0	2.0	2.0
1.1	2.2	2.2
1.2	2.4	2.4
1.5	3.0	2.4
2.0	4.0	2.4

当 $r_t > 1.2$ 时，这个样本不再因为概率继续上升而增加代理目标，梯度在此方向上为零。**设计动机**：避免对看起来“好”的旧策略样本过度提高概率。

15. 负 Advantage：坏动作也不能无限被压低

若 $\hat{A}_t < 0$ ，希望降低动作概率，即降低 r_t 。由于乘以负数会反转大小关系，

$$L_t^{\text{CLIP}} = \max(r_t, 1 - \epsilon) \hat{A}_t = \begin{cases} (1 - \epsilon) \hat{A}_t, & r_t < 1 - \epsilon, \\ r_t \hat{A}_t, & r_t \geq 1 - \epsilon. \end{cases} \quad (23)$$

设 $\hat{A}_t = -2, \epsilon = 0.2$:

r_t	原始 $r_t \hat{A}_t$	PPO 目标
0.5	-1.0	-1.6
0.7	-1.4	-1.6
0.8	-1.6	-1.6
1.0	-2.0	-2.0
1.2	-2.4	-2.4
1.5	-3.0	-3.0

PPO 最大化目标，因此负 Advantage 时降低 r 有利；但当 $r < 0.8$ 时不再额外奖励进一步压低该动作的概率。

16. 为什么一定要有 min，而不是只用 clip?

假设只有 $\text{clip}(r, 1 - \epsilon, 1 + \epsilon)\hat{A}$ 。若 $\hat{A} > 0$ 、 $r = 0.5$ 、 $\epsilon = 0.2$ ，说明好动作的概率已经被错误地大幅降低，但单独的 clip 会输出 $0.8\hat{A}$ ，在 $r < 0.8$ 区域梯度为零，无法纠正错误方向。

加入 min 后，由于 $r\hat{A} < 0.8\hat{A}$ ，实际使用原始项 $r\hat{A}$ ，因此仍有正梯度推动 r 回升。负 Advantage 下的错误方向同理。

更新方向正确但幅度过大： 停止提供额外优化收益；
更新方向不利于 Advantage： 保留纠偏梯度。

而且由 $\min(x, y) \leq x$ ，逐样本成立 $L^{\text{CLIP}} \leq L^{\text{CPI}}$ ：这是对未裁剪代理目标的保守下界，不等于对真实回报 J 的严格下界。

17. 从梯度看 PPO 在何处停止更新

因为旧策略分母在本次优化中不变，

$$\nabla_{\theta} r_t(\theta) = r_t(\theta) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t). \quad (24)$$

忽略折点处的不可微性（优化器采用次梯度），单样本梯度为

$$\nabla_{\theta} L_t^{\text{CLIP}} = \begin{cases} 0, & \hat{A}_t > 0, r_t > 1 + \epsilon, \\ 0, & \hat{A}_t < 0, r_t < 1 - \epsilon, \\ \hat{A}_t r_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t), & \text{其余区域.} \end{cases} \quad (25)$$

当 $\theta = \theta_{\text{old}}$ ， $r_t = 1$ ，因此初始局部梯度与 CPI/Advantage Policy Gradient 一致（差相应归一化常数）。PPO 不改变初始策略改进方向，而是在重复更新、概率比逐渐偏离 1 时抑制过大更新收益。

18. PPO 不能保证 Ratio 或 KL 被硬性约束

$\epsilon = 0.2$ 不意味着更新后始终有 $0.8 \leq r_t \leq 1.2$ 。Clipping 修改的是损失，而不是把策略投影到可行集合。共享神经网络参数会同时影响其他状态和样本；多个 Mini-batch Epoch 后，平均 KL 依然可能过大。

因此常在训练中监测

$$\bar{D}_{\text{KL}}(\pi_{\text{old}} \| \pi_{\theta})$$

并使用 Target-KL Early Stopping，必要时结合学习率调整、Gradient Clipping。需要分清三种裁剪：**Ratio Clipping** 控制代理收益；**Gradient Clipping** 控制梯度范数；**Action Clipping** 限制真实执行动作范围。三者不是一回事。

第六部分 GAE: Advantage 究竟如何估计?

19. Monte Carlo Advantage 的动机与代价

PPO 中使用的 $\hat{A}_t$ 不是凭空给定。按定义 $A^{\pi} = Q^{\pi} - V^{\pi}$ ，最直接的方法是把样本回报 G_t 作为 Q^{π} 的样本估计：

$$\hat{A}_t^{\text{MC}} = G_t - V_{\phi}(s_t), \quad G_t = \sum_{l=0}^{T-t-1} \gamma^l r_{t+l}. \quad (26)$$

对完整终止轨迹，在正确策略分布下， G_t 的条件期望等于 $Q^{\pi}(s_t, a_t)$ 。它不需要用 Critic 对未来进行自举，因此目标相对直接。然而，如果抓取任务直到第 100 步成功才给出奖励，早期动作的 G_t 会受到大量后续随机因素影响，梯度方差可能很大。

问题：能否尽早利用较少步数的信息估计 Advantage，而不依赖整条未来轨迹？

20. 由 Bellman 方程得到 One-Step TD Advantage

Bellman 关系为

$$Q^\pi(s_t, a_t) = \mathbb{E}[r_t + \gamma V^\pi(s_{t+1}) \mid s_t, a_t].$$

因此，一步 TD Residual:

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t) \quad (27)$$

可用作 Advantage 估计:

$$\hat{A}_t^{(1)} = \delta_t. \quad (28)$$

它只需要一个转移样本 s_t, a_t, r_t, s_{t+1} 。未来随机回报被 $V_\phi(s_{t+1})$ 代替，通常可以降低方差；代价是 Critic 不准确时会引入自举偏差。对于真正终止状态，未来价值为零；对于人为时间截断，则通常仍需 Bootstrap。

估计器	主要优点	主要缺点
Monte Carlo	减少未来价值自举偏差	长期回报导致高方差
One-Step TD	方差较低、反馈及时	更依赖 Critic 的预测准确性

21. 从 One-Step 推导 N-Step Advantage

一步估计:

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

两步估计:

$$\begin{aligned} \hat{A}_t^{(2)} &= r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t) \\ &= [r_t + \gamma V(s_{t+1}) - V(s_t)] \\ &\quad + \gamma [r_{t+1} + \gamma V(s_{t+2}) - V(s_{t+1})] \\ &= \delta_t + \gamma \delta_{t+1}. \end{aligned} \quad (29)$$

三步估计同理:

$$\hat{A}_t^{(3)} = \delta_t + \gamma \delta_{t+1} + \gamma^2 \delta_{t+2}.$$

推广到 n 步:

$$\hat{A}_t^{(n)} = \sum_{l=0}^{n-1} \gamma^l \delta_{t+l}. \quad (30)$$

较小 n 更多依赖 Value Bootstrap，较大 n 更多依赖真实未来奖励。**GAE 的出发点**：不要在单一的 n 上做硬选择，而是平滑组合多个 N-Step 估计器。

22. GAE 的指数权重与求和交换推导

令 $0 \leq \lambda < 1$, 定义

$$\hat{A}_t^{\text{GAE}(\gamma, \lambda)} = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} \hat{A}_t^{(n)}. \quad (31)$$

这里权重总和为 1: $(1 - \lambda) \sum_{n \geq 1} \lambda^{n-1} = 1$ 。代入 N-Step 形式:

$$\begin{aligned} \hat{A}_t^{\text{GAE}} &= (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} \sum_{l=0}^{n-1} \gamma^l \delta_{t+l} \\ &= \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} (1 - \lambda) \sum_{n=l+1}^{\infty} \lambda^{n-1} \\ &= \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} \lambda^l. \end{aligned}$$

因此:

$$\hat{A}_t^{\text{GAE}(\gamma, \lambda)} = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}. \quad (32)$$

也就是:

$$\hat{A}_t = \delta_t + \gamma \lambda \delta_{t+1} + (\gamma \lambda)^2 \delta_{t+2} + \dots$$

两个极端: $\lambda = 0$ 时只剩 δ_t , 即 One-Step TD; $\lambda \rightarrow 1$ 时趋于所有后续 TD Residual 之和。对于完整终止轨迹, $\lambda = 1$ 对应 $G_t - V(s_t)$ 。实用 PPO 常把 $\gamma = 0.99, \lambda = 0.95$ 作为起点。

GAE 并非数学上保证任何情况下方差都更低。它提供可调的偏差-方差折中: λ 较小通常更依赖 Critic, λ 较大通常更依赖长期采样回报。

23. GAE 的 $O(T)$ 反向递推

式 (32) 具有递归结构:

$$\hat{A}_t = \delta_t + \gamma \lambda \hat{A}_{t+1}. \quad (33)$$

若 $d_t = 1$ 表示 MDP 在该步真正终止:

$$\begin{aligned} \delta_t &= r_t + \gamma(1 - d_t)V_{\phi}(s_{t+1}) - V_{\phi}(s_t), \\ \hat{A}_t &= \delta_t + \gamma \lambda(1 - d_t)\hat{A}_{t+1}. \end{aligned} \quad (34)$$

这样从 $T - 1$ 向前循环一次就能完成全部计算, 复杂度为 $O(T)$ 。固定长度 Rollout 的最后一个状态若并未真正终止, 应使用 $V(s_T)$ Bootstrap; 而递归不能跨越一次 Episode Reset 错误地把下一条 Episode 的 Advantage 累加进来。实现时应区分 **terminated** (真实终止)、**truncated** (外部时间限制) 和 **Rollout Boundary** (采样段边界)。

24. 用 GAE 构造 Critic 训练目标

GAE 给出 Advantage 估计：

$$\hat{A}_t \approx Q^{\pi_{\text{old}}}(s_t, a_t) - V_{\text{old}}(s_t).$$

自然定义 TD(λ) Return：

$$\hat{R}_t^\lambda = \hat{A}_t^{\text{GAE}} + V_{\phi_{\text{old}}}(s_t). \quad (35)$$

这里 $V_{\phi_{\text{old}}}$ 是计算这一批 Rollout 目标时保存的旧 Value，整个 PPO 更新阶段必须视为常量。这一技巧让 Actor 的 Advantage 估计与 Critic 的回归目标保持一致。

24.1 Critic 的学习信号：Bellman Target 与半梯度

Critic 不知道真实的 $V^{\pi_{\text{old}}}(s)$ ，但可以从旧策略产生的样本 (s_t, a_t, r_t, s_{t+1}) 构造近似监督信号。对于不是真正终止的下一状态，一步自举目标为

$$y_t^{\text{TD}} = r_t + \gamma V_{\phi_{\text{old}}}(s_{t+1}), \quad \delta_t = y_t^{\text{TD}} - V_{\phi_{\text{old}}}(s_t).$$

若 $d_t = 1$ 代表真实的 MDP 终止，则使用 $y_t^{\text{TD}} = r_t + \gamma(1 - d_t)V_{\phi_{\text{old}}}(s_{t+1})$ 。人工时间上限、Rollout 切分和真正终止不能直接使用同一个 Mask：若环境只是截断而非终止，应从截断前的真实最后观测 Bootstrap，不能误用 Reset 后的状态。

把目标视为常量，Critic 通过回归学习：

$$\ell_t(\phi) = \frac{1}{2} (V_\phi(s_t) - \text{stopgrad}(y_t^{\text{TD}}))^2,$$

$$\nabla_\phi \ell_t = (V_\phi(s_t) - y_t^{\text{TD}}) \nabla_\phi V_\phi(s_t).$$

这就是常说的 **Semi-Gradient TD**：目标中的旧 Critic 值不接受本次反向传播；但预测项 $V_\phi(s_t)$ 接受梯度。若让梯度同时穿过目标 $V_\phi(s_{t+1})$ ，优化的就不再是通常的固定目标 TD 回归。注意：PPO 通常不要求像 DQN 那样再维护一个长期滞后的 Target Network；一次 Rollout 内保存旧值并固定回归目标即可。

24.2 为什么 PPO 常用 TD(λ) Return，而非仅用 One-Step TD?

第 22-23 节已推导 GAE。它与 Critic Target 存在一致的代数关系：

$$\hat{R}_t^\lambda = V_{\text{old}}(s_t) + \hat{A}_t^{\text{GAE}}.$$

对一段没有真实终止的连续轨迹，可等价写成

$$\hat{R}_t^\lambda = r_t + \gamma \left[(1 - \lambda) V_{\text{old}}(s_{t+1}) + \lambda \hat{R}_{t+1}^\lambda \right],$$

在采样段尾部用合适的 $V_{\text{old}}(s_T)$ 初始化 Bootstrap；遇到真正终止则令后续回报为零。 $\lambda = 0$ 退化为 One-Step TD Target；对完整终止轨迹， $\lambda = 1$ 退化为 Monte Carlo Return。中间值同时利用多步奖励与 Bootstrap，这既可作为 Actor 的 Advantage 信号，也可形成 Critic 的回归目标。

关键区分：“计算 GAE 的旧 Value”与“当前要训练的 V_ϕ ”是两个时间快照；PPO 一轮优化中， $\hat{A}_t$ 、 $\hat{R}_t^\lambda$ 和旧动作 Log Probability 均固定，即使 Critic 参数已更新也不在同一轮内偷偷重算它们。

第七部分 完整 Actor-Critic 损失

25. Actor Loss

本轮旧策略 Rollout 上，先计算并固定 $\hat{A}_t$ （实践中经常标准化）。Actor 最大化

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

在 PyTorch 等以最小化为接口的框架中：

$$\boxed{\mathcal{L}_{\text{actor}} = -L^{\text{CLIP}}}. \quad (36)$$

如果 $\hat{A}_t$ 通过当前可训练 Critic 的计算图产生，应在 Actor Loss 中对其 **detach**，避免无意把 Actor 更新传播到 Advantage 目标。

25.1 Actor 没有正确动作标签，为什么仍能学习？

行为克隆需要专家标签 a_t^* ，PPO 不需要。它只有自己采样的动作及其**相对质量** $\hat{A}_t$ 。先看未裁剪代理目标：

$$L_t^{\text{CPI}}(\theta) = \rho_t(\theta) \hat{A}_t, \quad \rho_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\text{old}}(a_t | s_t)}.$$

因为旧分母和 Advantage 被固定，

$$\boxed{\nabla_\theta [-L_t^{\text{CPI}}] = -\hat{A}_t \rho_t(\theta) \nabla_\theta \log \pi_\theta(a_t | s_t)}.$$

在本轮更新刚开始时 $\theta = \theta_{\text{old}}$ ，有 $\rho_t = 1$ ，Clipping 尚未生效，因此 Actor 与基本 Advantage Policy Gradient 的方向一致：

- $\hat{A}_t > 0$ ：提高这次采样动作的概率密度，**并非**将动作强行回归到它；
- $\hat{A}_t < 0$ ：降低这次动作的概率密度；
- $\hat{A}_t \simeq 0$ ：该样本给出的局部策略信号很弱。

对参数做梯度下降后，神经网络会改变许多状态与动作的概率，因此上面讨论的是**单样本、局部的梯度贡献**，不保证某个动作最终概率必然按该方向变化。Clipping 只在样本的概率比朝有利方向变化过多时，停止提供额外收益；第 17 节已经给出精确的分段梯度。

25.2 一个 Actor 更新数值例子

设某个采样动作在旧策略下的概率为 $\pi_{\text{old}}(a_t | s_t) = 0.20$ ， $\epsilon = 0.2$ ， $\hat{A}_t = +2$ 。若新概率变成 0.22，则 $\rho_t = 1.1$ ，该样本的未裁剪收益为 2.2；若变成 0.30，则 $\rho_t = 1.5$ ，PPO-Clip 对该样本只计 $1.2 \times 2 = 2.4$ ，进一步增大这一概率不再增加其代理收益。对于负 Advantage，过度**降低**概率时也出现相应平台。这里为了直观以离散动作的概率作例子；连续 Gaussian Policy 应使用**概率密度**与联合 Log Probability，而非单点概率。

26. Critic Loss

Critic 通过回归 $\hat{R}_t^\lambda$ 学习状态价值：

$$\boxed{\mathcal{L}_{\text{critic}}(\phi) = \mathbb{E}_t [(V_\phi(s_t) - \hat{R}_t^\lambda)^2]}. \quad (37)$$

也可以直接回归 Monte Carlo Return。某些实现另设 Value Clipping 或 $\frac{1}{2}$ 系数，但它们不是 PPO-Clip 目标的必要组成。Critic 过差会导致 Advantage 的符号或量级不准确，因此 PPO 的训练稳定性很大程度依赖 Value Function 的估计质量。

26.1 Critic 的误差如何真正改变网络参数?

为简化梯度形式, 考虑带 $\frac{1}{2}$ 系数的逐样本 MSE:

$$\ell_t^V(\phi) = \frac{1}{2}(V_\phi(s_t) - \hat{R}_t^\lambda)^2, \quad \nabla_\phi \ell_t^V = (V_\phi(s_t) - \hat{R}_t^\lambda) \nabla_\phi V_\phi(s_t).$$

例如 $V_{\text{old}}(s_t) = 4$ 、 $V_{\text{old}}(s_{t+1}) = 5$ 、 $r_t = 2$ 、 $\gamma = 0.9$ 、且没有终止, One-Step Target 为

$$y_t^{\text{TD}} = 2 + 0.9 \times 5 = 6.5, \quad \delta_t = 6.5 - 4 = 2.5.$$

若把 $V(s_t)$ 暂时视为一个可单独优化的标量, 则 $\partial \ell / \partial V = 4 - 6.5 = -2.5$, 梯度下降会推动预测值上升。对真实神经网络, 更新通过 $\nabla_\phi V_\phi(s_t)$ 分配到共享参数, 因此不会保证任何特定状态的预测单调改变。

学习的不只是即时奖励: $V_\phi(s_t)$ 预测的是当前策略下的折扣长期回报; 即使当前奖励为零, 后继状态的 Bootstrap 或未来多步奖励也能传递学习信号。Critic 属于近似的、自举的监督回归, 不是已知真值标签上的普通回归。

26.2 Critic Loss 降了, Value 就一定正确吗?

不一定。Target 依赖旧 Critic、有限轨迹和随机奖励; Actor 更新又会改变下轮所对应的策略与状态分布。低训练 MSE 不等于 Value 无偏, 也不等于真实 Episode Return 提升。**Explained Variance** 常定义为

$$\text{EV} = 1 - \frac{\text{Var}(\hat{R}_t^\lambda - V_\phi(s_t))}{\text{Var}(\hat{R}_t^\lambda)},$$

当目标的方差接近零时该指标不稳定; 且目标本身仍可能有偏。应结合成功率、实际回报与 Target / Value 的尺度一起诊断。某些实现对 Value 增量做 Clipping, 但它不保证真实价值精度提升。

27. Entropy Bonus 的动机

策略可能过早收缩到狭窄动作分布, 停止尝试其他抓取方式。使用策略熵

$$H(\pi_\theta(\cdot | s)) = -\mathbb{E}_{a \sim \pi_\theta}[\log \pi_\theta(a | s)]. \quad (38)$$

鼓励合理探索。其平均值记为 $L^{\text{entropy}} = \mathbb{E}_t[H(\pi_\theta(\cdot | s_t))]$ 。高熵并不总是更好, 尤其是机器人部署阶段; Entropy Coefficient 可能取零, 也可能随训练退化。

27.1 熵正则如何参与 Gaussian 方差学习?

一维未压缩 Gaussian $\mathcal{N}(\mu, \sigma^2)$ 的熵为

$$H(\pi) = \frac{1}{2} \log(2\pi e \sigma^2) = \frac{1}{2} \log(2\pi e) + \log \sigma, \quad \frac{\partial H}{\partial \log \sigma} = 1.$$

因此, 最小化总损失中的 $-c_e H$, 会在 $c_e > 0$ 时提供增加 $\log \sigma$ 的梯度倾向; 而 Advantage 的梯度可能使方差增大或减小, 具体见第 30 节。**探索并不是方差越大越好:** 高噪声会妨碍接触稳定性, 也可能使机器人执行动作触及安全限制。多维独立 Gaussian 的熵为各维之和; 经 Tanh 变换后的动作熵不能不加修改地等同于变换前 Gaussian 熵。

28. PPO 的最终训练目标

把 Actor、Critic、Entropy 组合, 得到一个常见的联合最大化目标:

$$L^{\text{PPO}}(\theta, \phi) = L^{\text{CLIP}}(\theta) - c_v \mathcal{L}_{\text{critic}}(\phi) + c_e L^{\text{entropy}}(\theta). \quad (39)$$

对应的最小化形式为

$$\mathcal{L}_{\text{total}} = -L^{\text{CLIP}} + c_v \mathcal{L}_{\text{critic}} - c_e L^{\text{entropy}}. \quad (40)$$

- c_v : 控制 Value 回归损失权重;
- c_e : 控制 Entropy Bonus 权重;
- 独立 Actor 与 Critic 参数时可分别更新; 共享网络时需要考虑梯度耦合。

常见实现对每个训练 Mini-batch 做多轮更新, 但旧 Log Probability、Advantage 和 Return Target 在该 Rollout 的 PPO Epoch 内保持固定。

28.1 梯度图: 谁更新谁?

在使用独立参数的 Actor 和 Critic 时, 一轮采样后先完成 GAE 计算, 然后形成两条彼此不同的监督路径:

$$\begin{aligned} \pi_{\theta_{\text{old}}} &\longrightarrow (s_t, a_t, r_t, s_{t+1}, \log \pi_{\text{old}}, V_{\text{old}}) \\ &\quad \downarrow \text{TD errors and GAE} \\ \hat{A}_t &\xrightarrow{\text{stopgrad}} L^{\text{CLIP}}(\theta) \longrightarrow \text{Actor parameters } \theta, \\ \hat{R}_t^\lambda = \hat{A}_t + V_{\text{old}}(s_t) &\xrightarrow{\text{stopgrad}} \mathcal{L}_{\text{critic}}(\phi) \longrightarrow \text{Critic parameters } \phi. \end{aligned}$$

为什么停止 Advantage 的梯度? Actor 应因改变 $\pi_{\theta}(a_t | s_t)$ 而提高代理目标, 而非通过修改 Critic 的 V_{ϕ} 或回归标签来“投机”降低 Actor Loss。固定 Advantage 并不会阻止 Critic 根据自己的 Loss 学习。

若两个网络各自独立, 可分别用不同优化器和学习率更新。若共享视觉编码器或 Backbone, 联合 Loss 对共享参数的梯度来自 Actor、Critic 和 Entropy 三项; **stopgrad** 仍用于冻结**优势与回归目标**, 但不会阻断 Critic Loss 自身对共享 Backbone 的合理梯度。共享表示降低推理和参数开销, 也可能导致策略与价值梯度竞争; 这并不是 PPO 必须采用的架构。

28.2 机器人中的 Asymmetric Actor–Critic

在仿真训练中, Actor 可以只输入部署时可获得的相机图像、本体感觉或估计状态, 而 Critic 使用仿真器可访问的真实物体姿态、接触信息等 Privileged State。用符号表示为

$$\pi_{\theta}(a_t | o_t), \quad V_{\phi}(x_t), \quad x_t \text{ 可包含训练阶段特权信息.}$$

这类 Asymmetric Actor–Critic 能减少价值估计难度, 但**绝不能让部署 Actor 无意依赖特权输入**; 如存在历史信息、部分可观测性和非马尔可夫观测, 也要明确 Value 使用的信息集与 Bootstrap 条件。它是训练架构选择, 而不是 PPO-Clip 算法的必备条件。

第八部分 连续动作 PPO 与机器人控制

29. Gaussian Policy 的对数概率与 Ratio

机器人动作通常是连续向量, 例如关节位置增量、目标速度或关节力矩。典型 Actor 使用对角 Gaussian:

$$\pi_{\theta}(a | s) = \mathcal{N}(\mu_{\theta}(s), \text{diag}(\sigma_1^2, \dots, \sigma_n^2)). \quad (41)$$

则联合 Log Probability 为

$$\log \pi_{\theta}(a | s) = -\frac{1}{2} \sum_{i=1}^n \left[\frac{(a_i - \mu_i)^2}{\sigma_i^2} + 2 \log \sigma_i + \log(2\pi) \right]. \quad (42)$$

数值稳定的 Ratio 计算为

$$r_t = \exp(\log \pi_\theta(a_t | s_t) - \log \pi_{\text{old}}(a_t | s_t)). \quad (43)$$

实现重点：先把各动作维度的 Log Probability 求和，得到整个动作向量的联合 Log Probability，再指数化算 Ratio；不能对各维 Ratio 分别 Clip 后当作标准 PPO 联合策略目标。若使用 Tanh-Squashed Gaussian，需按变量变换计算正确的 Log Probability；若环境直接将采样动作截断到硬边界，也要审慎区分“采样动作”和“实际执行动作”的密度。

30. Gaussian Policy 的梯度如何推动动作均值？

考虑一维固定方差模型：

$$a \sim \mathcal{N}(\mu_\theta(s), \sigma^2),$$

$$\frac{\partial \log \pi_\theta(a | s)}{\partial \mu_\theta} = \frac{a - \mu_\theta(s)}{\sigma^2}. \quad (44)$$

因此未裁剪区域的 Policy Gradient 对均值参数的主要贡献为

$$\nabla_\theta J \propto \mathbb{E} \left[\hat{A} \frac{a - \mu_\theta(s)}{\sigma^2} \nabla_\theta \mu_\theta(s) \right]. \quad (45)$$

当 $\hat{A} > 0$ ，更新推动分布均值朝采样动作移动；当 $\hat{A} < 0$ ，更新推动均值远离该动作。PPO 的 Clipping 再限制单个样本从此类概率变化中获得的额外收益。若 σ 可训练，还有关于方差的梯度项。

30.1 除了均值，Log Standard Deviation 怎样学习？

允许 Gaussian 的标准差可训练，令 $\eta_\theta(s) = \log \sigma_\theta(s)$ 。一维 Log Probability 为

$$\log \pi_\theta(a | s) = -\frac{1}{2} \left[\frac{(a - \mu_\theta(s))^2}{e^{2\eta_\theta(s)}} + 2\eta_\theta(s) + \log(2\pi) \right].$$

对均值和对数标准差分别求导：

$$\frac{\partial \log \pi_\theta}{\partial \mu_\theta} = \frac{a - \mu_\theta}{\sigma_\theta^2}, \quad \frac{\partial \log \pi_\theta}{\partial \eta_\theta} = \frac{(a - \mu_\theta)^2}{\sigma_\theta^2} - 1.$$

未被 Clipping 截断的样本，在梯度上升方向给这两项乘以 $\hat{A}_t \rho_t$ 。对于 $\hat{A}_t > 0$ ，如果 $|a - \mu| > \sigma$ ，该样本会倾向于增加 $\log \sigma$ ；如果 $|a - \mu| < \sigma$ ，则倾向于减小 $\log \sigma$ 。 $\hat{A}_t < 0$ 时方向相反。可见策略不是机械地“好动作就缩小噪声”，而是同时学习概率分布的**位置**和**宽度**。

机器人 PPO 中常见两种参数化：

1. **State-Independent Log Std**：网络预测 $\mu_\theta(s)$ ，另把每个动作维度的 η_i 当作可训练参数；在许多 Locomotion 配置中常用，结构简单。
2. **State-Dependent Log Std**：网络同时预测 $\mu_\theta(s)$ 和 $\eta_\theta(s)$ ；表达能力更强，但数值稳定性与训练难度可能不同。

实际可对 Log Std 设上下界，并检查策略熵和实际控制噪声。训练时从策略采样动作（通常为随机）；评估/部署时常用均值动作或等价的确定性映射，但这不是 PPO 数学要求的唯一选择。若使用 Tanh-Squashed Gaussian，密度计算需要一致地考虑相应变量变换。

31. 机器人 Residual PPO 示例

假设有基础轨迹跟随控制器

$$a_t^{\text{base}} = \pi_{\text{base}}(s_t),$$

另外用 PPO 学习随机残差

$$\Delta a_t \sim \pi_{\theta}(\cdot | s_t),$$

最终执行

$$a_t = a_t^{\text{base}} + \Delta a_t. \quad (46)$$

残差可表示关节目标位置、速度、末端位姿的修正。

此时要用残差动作分布计算

$$r_t = \frac{\pi_{\theta}(\Delta a_t | s_t)}{\pi_{\text{old}}(\Delta a_t | s_t)}.$$

如果残差经过缩放、非线性变换或安全过滤，必须明确哪些变量的密度用于策略优化、哪些是实际控制命令。PPO 不要求接触动力学可微，但需要可重复的环境 Rollout、合理奖励和足够准确的状态/价值估计。

第九部分 一次 PPO Iteration 的完整执行过程

32. 从采样到更新的十个步骤

1. 冻结旧策略：令 $\pi_{\text{old}} = \pi_{\theta_k}$ ，用它采集若干 Episode 或定长 Rollout。
2. 保存采样信息：记录 (s_t, a_t, r_t, s_{t+1}) 、终止标记、 $\log \pi_{\text{old}}(a_t | s_t)$ 和 $V_{\text{old}}(s_t)$ 。
3. 计算 TD Error: $\delta_t = r_t + \gamma V_{\text{old}}(s_{t+1}) - V_{\text{old}}(s_t)$ ，对真实终止进行正确 Mask。
4. 反向计算 GAE: $\hat{A}_t = \delta_t + \gamma \lambda \hat{A}_{t+1}$ ，同时正确处理 Episode Boundary。
5. 计算 Critic Target: $\hat{R}_t^{\lambda} = \hat{A}_t + V_{\text{old}}(s_t)$ ，之后固定该目标。
6. 可选标准化 Advantage: $\tilde{A}_t = (\hat{A}_t - \mu_A) / (\sigma_A + \varepsilon_{\text{num}})$ 。
7. 在 Mini-batch 上计算新 Log Probability: 对于已采样的旧动作 a_t ，计算当前 $\log \pi_{\theta}(a_t | s_t)$ 。
8. 计算 Ratio 与 Clipped Actor Loss: $r_t = \exp(\log \pi_{\theta} - \log \pi_{\text{old}})$ ，然后应用式 (21)。
9. 更新 Actor 与 Critic: 最小化联合损失或分别优化，在同一批数据上重复若干 Epoch，监测 KL、Clip Fraction、Value Loss。
10. 重新采样: 更新完毕后，用新策略生成下一批数据；不把任意久远的旧数据当作标准 On-Policy PPO 的普通 Rollout 重复训练。

32.1 同一批 Rollout 中，Actor 与 Critic 的参数到底何时变化？

建议把第 32 节的十个步骤理解为两个不同的阶段：

1. 数据采样与 GAE 阶段：环境状态、采样动作和奖励随时间变化，随后一次性计算 $\hat{A}_t$ 与 $\hat{R}_t^{\lambda}$ 。采样时的行为策略 π_{old} 和用于形成 Target 的 V_{old} 保持为固定快照。
2. K 轮 Mini-batch 优化阶段：只更新当前 Actor 参数 θ 、Critic 参数 ϕ 以及由它们产生的新 Log Probability 和 Value。存储的旧动作、旧 Log Probability、 $\hat{A}_t$ 、 $\hat{R}_t^{\lambda}$ 始终不变。
3. 下一次 Rollout 阶段：更新后的 Actor 成为新的行为策略，采集新数据；不把很久以前的旧 Rollout 当作标准 PPO 的新采样继续训练。

这样能解释一个常见混淆：在同一轮 PPO Epoch 中，Critic 的预测可以改变，但用来训练 Critic 的回归 Target 不跟着改变。如果训练时动态重算旧 Log Probability 或 Advantage，优化目标会不再对应于最初的旧策略快照。

33. 简洁且可直接改造的 PyTorch 实现

下面代码给出 PPO 核心 Actor Loss。输入 `new_log_prob`、`old_log_prob` 均为动作向量的联合 Log Probability，且旧 Log Probability 和 Advantage 在本轮更新中固定。

```
import torch

def ppo_actor_loss(new_log_prob, old_log_prob,
                  advantages, clip_eps=0.2):
    # The old log probability and advantage are fixed rollout data.
    old_log_prob = old_log_prob.detach()
    advantages = advantages.detach()

    log_ratio = new_log_prob - old_log_prob
    ratio = torch.exp(log_ratio)

    objective_unclipped = ratio * advantages
    objective_clipped = torch.clamp(
        ratio, 1.0 - clip_eps, 1.0 + clip_eps
    ) * advantages

    return -torch.minimum(
        objective_unclipped, objective_clipped
    ).mean()
```

计算 GAE 的最小后向递推如下。此示例中的 `bootstrap_mask[t]` 控制 δ_t 是否使用下一状态价值；`continue_mask[t]` 控制 GAE 是否跨越此步递归，真实终止、超时与 Rollout 切分须由环境正确提供。

```
import torch

def compute_gae(rewards, values, next_values,
               bootstrap_mask, continue_mask,
               gamma=0.99, lam=0.95):
    """All tensors are indexed by rollout time."""
    T = rewards.shape[0]
    advantages = torch.zeros_like(rewards)
    running = torch.zeros_like(rewards[0])

    for t in reversed(range(T)):
        delta = (rewards[t]
                + gamma * bootstrap_mask[t] * next_values[t]
                - values[t])
        running = (delta
                  + gamma * lam * continue_mask[t] * running)
        advantages[t] = running

    returns = advantages + values
    return advantages, returns
```

如果使用并行环境，这些变量可以具有形状 `[T, N_env]`；`continue_mask` 不能跨不同 Episode 串接。用于生成 `returns` 的旧 `values` 和 `next_values` 应与计算 Rollout 目标时的冻结 Critic 一致。

对于对角 Gaussian：

```
# dist: torch.distributions.Normal, action shape [batch, action_dim]
new_log_prob = dist.log_prob(actions).sum(dim=-1)
# entropy shape [batch]
entropy = dist.entropy().sum(dim=-1)
```

这段代码只展示核心公式；完整训练还需要 Rollout Buffer、环境步进、网络结构、优化器、正确的 `terminated/truncated` 处理与指标记录。

33.1 从三种损失得到一次可反向传播的完整更新

以下补充代码只演示已准备好一个固定 Rollout Mini-batch 时的三个 Loss。它不替代上面的 GAE / 终止 Mask 处理，也不是完整训练包。

```
import torch

def ppo_joint_losses(actor, critic, obs, actions,
                    old_log_prob, advantages, returns,
                    clip_eps=0.2, value_coef=0.5,
                    entropy_coef=0.0):
    # actor(obs) returns [batch, action_dim] mean and log_std.
    mean, log_std = actor(obs)
    dist = torch.distributions.Independent(
        torch.distributions.Normal(mean, log_std.exp()), 1
    )
    new_log_prob = dist.log_prob(actions)
    actor_loss = ppo_actor_loss(
        new_log_prob, old_log_prob.detach(),
        advantages.detach(), clip_eps
    )
    value_pred = critic(obs).squeeze(-1)
    value_loss = 0.5 * (value_pred - returns.detach()).square().mean()
    entropy = dist.entropy().mean()
    total_loss = actor_loss + value_coef * value_loss - entropy_coef * entropy
    return total_loss, actor_loss, value_loss, entropy
```

若 Actor、Critic 参数互不共享，可以分别计算梯度并调用各自优化器；若共享 Backbone，则 `total_loss.backward()` 会汇总对共享参数的梯度。示例使用 `Independent(Normal, 1)` 得到动作向量的联合 Log Probability；如果网络的动作分布实现不同，必须对应调整 Density 的计算。这里的 `returns` 必须由旧 Value + GAE 在优化前计算。

第十部分 超参数与实际诊断

34. 常见起始超参数及其含义

参数	常见起点	控制什么
γ	0.99	未来奖励的时间折扣
GAE λ	0.95	Advantage 的偏差-方差折中
Clip ϵ	0.1-0.2	过大 Probability Ratio 的额外代理收益
Actor 学习率	3×10^{-4}	Actor 优化速度
PPO Epoch 数	3-10	一批旧数据的重复更新次数
Target KL	0.01-0.03	是否提前停止 Actor 更新
Entropy 系数	0 或小正数	探索强度
Value 权重	约 0.5	Critic 回归损失相对权重

这些是便于启动实验的经验量级，不是普适最优参数。RSL-RL 等大规模并行运控实现可能使用不同学习率、Batch 尺寸、Clip 和 Epoch 设置。

γ 与 λ 的根本区别： γ 决定优化目标如何折扣远期奖励，而 λ 决定 Advantage 估计多大程度依赖远期采样 TD 信息。二者共同以 $(\gamma\lambda)^l$ 出现在 GAE 中，但理论角色不同。

重要诊断量还包括 Episode Return、Success Rate、Value Loss、Explained Variance、Entropy、Approximate KL、Clip Fraction、梯度范数。出现“代理 Loss 看似改善、真实成功率下降”时，应重点检查数据分布漂移、Critic 误差、KL 过大和奖励设计。

34.1 将诊断指标映射到 Actor-Critic 的真实学习问题

- **Value Loss 大 / Explained Variance 低**：可能是 Critic Target 噪声大、网络表达能力不足、Bootstrap 偏差或奖励尺度变化。优先核对 Return Target 与 Value 的范围、真正终止的 Mask、Value 学习率和 Normalization。
- **Actor Loss 下降，但成功率不升**：代理目标不必然等同于真实性能。应比较新 Rollout 的 Return、Approximate KL、Clip Fraction 和奖励定义，并审查 Advantage 估计的噪声。
- **Approximate KL 或 Clip Fraction 长期偏大**：通常提示在固定 Rollout 上 Actor 更新太快。检查 Actor Learning Rate、每批 PPO Epoch 数、Mini-batch 大小和 Target-KL Early Stopping。
- **Entropy 很快接近零**：检查可训练 Log Std 是否收缩过快、Log Std 的上下界、动作分布形式及 Entropy Coefficient。
- **机器人接触任务中策略抖动**：可能来自过大的探索噪声、不合适的动作尺度或控制频率；应联动检查 Action Scale、Log Std、控制周期以及关节/接触约束。

注意两类学习率和两个时间尺度：Actor 的学习率控制策略漂移；Critic 的学习率控制 Value 对变化中的目标的追踪速度。Critic 太慢可能损坏 GAE，太快可能导致高噪声目标下的过拟合或优化不稳定。不能仅凭 Critic Loss 数值选择两者；应同时观察性能和 KL。更多数据复用提高样本利用率，但也增加偏离采样策略的风险。

第十一部分 与 REINFORCE、Actor-Critic、TRPO 的关系

35. 算法演化：每一步解决一个问题

算法	关键公式 / 想法	主要动机
REINFORCE	$\nabla \log \pi_{\theta} G$	不对环境求导也能优化策略
Advantage Actor-Critic	$\nabla \log \pi_{\theta} \hat{A}$	用状态价值作为 Baseline 降低方差
TRPO	最大化 $\mathbb{E}[r\hat{A}]$ ，约束 KL	保持代理目标在可信的局部区域
PPO-Clip	$\min(r\hat{A}, \text{clip}(r)\hat{A})$	用简单一阶更新抑制过大概率改变的收益
PPO + GAE	$\hat{A}_t = \sum_l (\gamma\lambda)^l \delta_{t+l}$	调节 Advantage 估计的偏差和方差

PPO 本质上依然是 **On-Policy Policy Gradient / Actor-Critic**。在每轮 Rollout 的初始点，它的 Clipped Objective 与普通 CPI 代理目标的一阶梯度相同；真正的区别发生在对同一批数据继续做多次更新、Probability Ratio 逐渐偏离 1 之后。

PPO 在仿真机器人控制中常见，是因为可以并行采样许多环境、训练结构简单、不要求动力学可微，且天然支持 Gaussian 连续动作。但在真实机器人上，样本采集昂贵，SAC 等 Off-Policy 算法可能更具交互样本效率。对于 Diffusion / Flow Matching Action Policy，标准 PPO 需要的显式且可计算 Log Probability 往往不易获得，通常需要专门改造。

第十二部分 最终公式总览与研究启示

核心公式 1：策略梯度

$$\nabla_{\theta} J \propto \mathbb{E} [\nabla_{\theta} \log \pi_{\theta}(a | s) A^{\pi}(s, a)].$$

动机：不必对不可微动力学反向传播。

核心公式 2：重要性采样

$$r_t = \pi_{\theta}(a_t | s_t) / \pi_{\text{old}}(a_t | s_t).$$

动机：在旧策略动作样本上构造新策略的局部代理目标。

核心公式 3：TRPO 的信赖域思想

$$\bar{D}_{\text{KL}}(\pi_{\text{old}} \| \pi_{\theta}) \leq \delta.$$

动机：控制代理目标因分布变化而失真的风险。

核心公式 4: PPO Clipped Objective

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

动机：保守限制过大概率变化能获得的代理收益，保留修正不利更新方向的梯度。

核心公式 5: GAE

$$\hat{A}_t = \sum_{l \geq 0} (\gamma \lambda)^l \delta_{t+l}, \quad \delta_t = r_t^{\text{reward}} + \gamma V(s_{t+1}) - V(s_t).$$

动机：在长期样本回报和短期 Bootstrap 间进行平滑折中。这里用 r_t^{reward} 区分即时奖励与 PPO 的概率比 $r_t(\theta)$ 。

最终最小化损失：

$$\mathcal{L}_{\text{PPO}} = -L^{\text{CLIP}} + c_v \mathcal{L}_{\text{critic}} - c_e L^{\text{entropy}}.$$

研究层面的核心结论：PPO 并不是一个新的环境梯度计算方法。它使用 Policy Gradient 确定局部性能改进方向，用 Importance Sampling 在旧数据上构造代理目标，用 Clipping 抑制过度优化，并用 GAE 改善 Advantage 估计。GAE 主要解决梯度估计质量的问题；Clipping 主要解决策略更新稳定性的问题。二者功能不同、相互补充。

参考文献与延伸阅读

1. Schulman, J. et al. (2017). *Proximal Policy Optimization Algorithms*. arXiv:1707.06347. <https://arxiv.org/abs/1707.06347>.
2. Schulman, J. et al. (2015). *Trust Region Policy Optimization*. ICML. <https://arxiv.org/abs/1502.05477>.
3. Schulman, J. et al. (2015). *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. ICLR 2016. <https://arxiv.org/abs/1506.02438>.
4. OpenAI Spinning Up. *PPO Documentation*. <https://spinningup.openai.com/en/latest/algorithms/ppo.html>

版权与许可：CC BY 4.0。

代码说明：代码片段仅用于解释算法机制，并非经过完整环境验证的可直接训练脚本。