PR #1827 完整报告

radixark/miles

fix: run PPO GAE over trainable tokens only

合并时间：2026-07-29 09:25

原文链接：http://prhub.com.cn/radixark/miles/pull/1827

执行摘要

- 一句话：修复 PPO GAE 仅在可训练 token 上计算
- 推荐动作：值得精读。这是 PPO 在 agentic / 多轮场景下训练正确性的关键修复：" 压缩子序列再跑 batched GAE" 的设计让 FLA 风格 chunked kernel 零改动复用，属于可迁移的模式；docstring 对 masked 位置 0 值、下游 whitening / OPD 偏移、mask-weighted 统计量的边界界定也很值得学习。建议与 PR#1916（多轮样本语义）、PR#1759（session server 样本组装）一起阅读，理解 loss_mask 从 rollout 到 loss 的完整链路。

功能与动机

PR body 明确指出当前 PPO GAE 把每个 response token 都当作 MDP 转移，并在最后一个 response token 注入终止奖励，完全无视 loss_mask。多轮 / agentic rollout 中工具调用与环境观察 token 被 mask（loss_mask == 0），由此产生三个问题：

1) 若响应以 masked token 结尾，标量奖励落在 masked 位置，需经 (gamma * lambd)^gap 衰减才能到达可训练 token，gamma = 0 时直接丢失； 2) masked 位置的 KL 形奖励与 critic value delta 泄漏进相邻可训练 token 的 advantage，且每个 masked token 多累积一步 gamma * lambd 衰减，稀释长观察块上的信用； 3) 截断语义从未定义。该 PR 的目标是让 GAE 只在可训练 token 子序列上运行，并将截断的零 bootstrap 语义文档化、用测试钉死。

实现拆解

1. 入口接线：把 loss_masks 透传到 GAE 批处理函数。miles/backends/training_utils/loss_hub/advantages.py 的 compute_advantages 在 args.advantage_estimator == "ppo" 分支调用 get_advantages_and_returns_batch 时新增 loss_masks=loss_masks 传参（此前该函数完全不接收 mask 参数）。这一层只改 1 行，但让下游拿到 mask 语义成为可能；GRPO / GSPO 与 reinforce 系列分支不受影响。
2. 核心改造：在 get_advantages_and_returns_batch 中按可训练 token 压缩序列。math_utils.py 中该函数新增 loss_masks 参数并在头部断言 B == len(loss_masks)。随后用 loss_masks[i][: response_lengths[i]].nonzero() 求出每个样本的可训练索引，将 values / rewards 按索引打包成 [B, max(trainable_lengths)] 的 packed 张量；终止奖励从 full_rewards[i, L-1] 改为 packed_rewards[i, K-1] 注入，即落在最后一个可训练 token；全 masked 样本 max_len == 0 时直接返回空双零张量，终止奖励被丢弃。由于压缩发生在 batched scan 之前，vanilla_gae / chunked_gae（FLA 风格 chunked kernel）完全不用改动。

3. 结果散射与 CP 兼容。GAE 结果先写回全 0 的 [R_i] 向量（masked 位置保持 0），再在 cp_size > 1 时用 slice_log_prob_with_cp 切分到当前 rank。相比旧实现直接对 padded 行切片，新实现对散射后的完整序列做 CP 切分，语义更直观；两 rank 结果一致由 Gloo 测试覆盖。
4. 语义固化与测试配套。新增 tests/fast/backends/training_utils/test_ppo_gae_masks.py，以串行参考实现 _reference_masked_gae 为基准，6 组测试全部在 chunked True/False 下参数化：压缩子序列与参考一致、carry 跨 mask gap 只衰减一个 gamma * lambd 且与 gap 长度无关、masked 尾部时终止奖励落在最后一个可训练 token（gamma=0 场景）、全 masked 样本输出 0 且不干扰同 batch 邻居、全 1 mask 与旧 vanilla_gae 逐位一致、gamma = lambd = 1 时 telescoping 到 sum(rewards) + terminal - V_t 钉死截断零 bootstrap 语义。test_ppo_cp_advantages.py 新增 _run_ppo_masked_case / _worker_masked_case 两个 2-rank Gloo 用例，验证 masked GAE 在 CP 下与单 rank baseline 完全一致，并对 gamma = lambd = 0 给出精确期望值。
5. 文档边界。docstring 完整记录了语义边界：masked 位置输出 0、下游 whitening / OPD 仍可能使其非零、whitening 统计量为 mask-weighted 所以注入的 0 不偏置统计量，最终正确性依赖 policy loss 对 inactive token 重新置零与所有 loss reducer 按 loss_mask 加权。

关键文件：

- miles/backends/training_utils/loss_hub/math_utils.py（模块 优势计算；类别 source；类型 core-logic；符号 get_advantages_and_returns_batch）：核心改动文件。get_advantages_and_returns_batch 新增 loss_masks 参数，把每个样本压缩到可训练 token 子序列后跑 GAE，终止奖励注入最后一个可训练 token，masked 位置输出 0，全 masked 样本直接返回 0；docstring 完整定义截断零 bootstrap 与下游 whitening / OPD 语义。
- tests/fast/backends/training_utils/test_ppo_gae_masks.py（模块 GAE 测试；类别 test；类型 test-coverage；符号 _trivial_parallel_state, _reference_masked_gae, _compute, test_masked_gap_matches_compressed_reference）：新增主测试文件（+209 行），以串行参考实现 _reference_masked_gae 为基准，6 组测试全部在 chunked True/False 下参数化，把压缩子序列等价性、carry 跨 gap 单次衰减、终止奖励落点、全 masked 样本、全 1 mask 回归、截断零 bootstrap 等语义全部钉死。
- tests/fast/backends/training_utils/test_ppo_cp_advantages.py（模块 CP 测试；类别 test；类型 test-coverage；符号 _run_ppo_masked_case, _worker_masked_case, test_ppo_masked_gae_matches_single_rank_baseline）：新增 2-rank Gloo masked 用例（_run_ppo_masked_case / _worker_masked_case / test_ppo_masked_gae_matches_single_rank_baseline），验证 CP 下 masked GAE 与单 rank baseline 完全一致、masked 位置为 0，并对 gamma = lambd = 0 给出精确期望值，证明压缩散射与 CP 切分可以正确组合。
- miles/backends/training_utils/loss_hub/advantages.py（模块 优势估计；类别 source；类型 core-logic；符号 compute_advantages）：改动仅 1 行，但这是 loss_masks 进入 GAE 计算链路的入口：compute_advantages 的 PPO 分支把 loss_masks 透传给 get_advantages_and_returns_batch，使掩码语义真正接入训练管线。

关键符号：get_advantages_and_returns_batch, compute_advantages

关键源码片段

miles/backends/training_utils/loss_hub/math_utils.py

核心改动文件。get_advantages_and_returns_batch 新增 loss_masks 参数，把每个样本压缩到可训练 token 子序列后跑 GAE，终止奖励注入最后一个可训练 token，masked 位置输出 0，全 masked 样本直接返回 0；docstring 完整定义截断零 bootstrap 与下游 whitening / OPD 语义。

```
# miles/backends/training_utils/loss_hub/math_utils.py
# get_advantages_and_returns_batch 的 PPO 分支核心逻辑（head 版本）

    # CP 场景先用 all_gather_with_cp 恢复完整响应序列，非 CP 直接使用本 rank 数据
    # full_values_list[i] 与 full_rewards_list[i] 的 shape 均为 [resp_len_i]

    # Compress each sample to its trainable positions so that masked
    # tokens do not act as MDP transitions in the GAE recursion.
    # 可训练索引 = loss_mask 非零位置；mask 超出 response 长度的部分被截断
    trainable_indices = [
        loss_masks[i][: response_lengths[i]].to(device).nonzero(as_tuple=True)[0] for i in range(B)
    ]
    trainable_lengths = [idx.numel() for idx in trainable_indices]

    # 按可训练 token 数 padding，而不是按 response 长度
    # 长观察块（masked gap）不再撑大 GAE 扫描长度
    max_len = max(trainable_lengths)

    packed_values = torch.zeros(B, max_len, device=device, dtype=dtype)
    packed_rewards = torch.zeros(B, max_len, device=device, dtype=dtype)

    for i in range(B):
        K = trainable_lengths[i]
        if K > 0:
            idx = trainable_indices[i]
            # 压缩：只保留可训练位置的 value 与 reward（包括 KL shaping）
            packed_values[i, :K] = full_values_list[i][idx]
            packed_rewards[i, :K] = full_rewards_list[i][idx]
            # 终止奖励注入最后一个可训练 token，而不是最后一个 response token
            # 否则 masked 尾部 + gamma = 0 时终止奖励会完全丢失
            packed_rewards[i, K - 1] += terminal_rewards[i]

    # 全 masked 样本：没有任何可训练 token，直接返回空的双零结果
    # 其终止奖励被丢弃，避免把垃圾信号训练进模型
    if max_len == 0:
        packed_advantages = torch.zeros(B, 0, device=device, dtype=dtype)
        packed_returns = torch.zeros(B, 0, device=device, dtype=dtype)
    elif not chunked:
```

```
            packed_advantages, packed_returns = vanilla_gae(
                rewards=packed_rewards, values=packed_values, gamma=gamma, lambd=lambd,
            )
        else:
            # FLA 风格 chunked kernel 无需改动：压缩后的 packed 序列就是普通连续序列
            packed_advantages, packed_returns = chunked_gae(
                rewards=packed_rewards, values=packed_values, gamma=gamma, lambd=lambd,
            )

        # 将压缩结果 scatter 回原始长度，masked 位置保持 0
        for i in range(B):
            resp_len = response_lengths[i]
            K = trainable_lengths[i]

            adv_full = torch.zeros(resp_len, device=device, dtype=dtype)
            ret_full = torch.zeros(resp_len, device=device, dtype=dtype)
            if K > 0:
                idx = trainable_indices[i]
                adv_full[idx] = packed_advantages[i, :K]
                ret_full[idx] = packed_returns[i, :K]

            # cp_size > 1 时再按 qkv_format 用 slice_log_prob_with_cp 切分到当前 rank
            #  （CP 切分分支从略，逻辑与旧实现一致）
```

tests/fast/backends/training_utils/test_ppo_gae_masks.py

新增主测试文件（+209 行），以串行参考实现 _reference_masked_gae 为基准，6 组测试全部在 chunked True/False 下参数化，把压缩子序列等价性、carry 跨 gap 单次衰减、终止奖励落点、全 masked 样本、全 1 mask 回归、截断零 bootstrap 等语义全部钉死。

```
# tests/fast/backends/training_utils/test_ppo_gae_masks.py
# 串行参考实现：在压缩后的可训练 token 子序列上跑标准 GAE

def _reference_masked_gae(
    values: torch.Tensor,
    rewards: torch.Tensor,
    mask: torch.Tensor,
    terminal_reward: float,
    gamma: float,
    lambd: float,
) -> tuple[torch.Tensor, torch.Tensor]:
    """Serial GAE over the compressed subsequence of trainable tokens."""
    idx = mask.nonzero(as_tuple=True)[0]
    advantages = torch.zeros_like(values)
    returns = torch.zeros_like(values)
    if idx.numel() == 0:
        # 全 masked 样本：返回全 0，终止奖励被丢弃
        return advantages, returns

    # 压缩：只保留可训练位置
```

```
    v = values[idx]
    r = rewards[idx].clone()
    # 终止奖励加到最后一个可训练 token
    r[-1] += terminal_reward

    K = v.numel()
    compressed_adv = torch.zeros_like(v)
    lastgaelam = 0.0
    for t in reversed(range(K)):
        # 截断语义：最后一个可训练 token 之后的 value 视为 0（零 bootstrap）
        next_value = v[t + 1] if t < K - 1 else 0.0
        delta = r[t] + gamma * next_value - v[t]
        lastgaelam = delta + gamma * lambd * lastgaelam
        compressed_adv[t] = lastgaelam

    # 散射回原始位置，masked 位置保持 0
    advantages[idx] = compressed_adv
    returns[idx] = compressed_adv + v
    return advantages, returns
```

评论区精华

审查共 3 条评论，全部收敛，无未解决疑虑：

1. Shi-Dong 的自评与文档修正：在 docstring review 中指出下游 on-policy distillation（OPD）与 advantage whitening 仍可能把 masked 位置的 0 变成非零，原表述不严谨。该观察直接落地为第 3 个 commit（8643926）：docstring 澄清 whitening 统计量是 mask-weighted 的、注入的 0 不偏置统计量，正确性最终由各 loss 的 mask 加权保证。
2. guapisolo 的疑问与撤回：在 math_utils.py 第 700 行（max_len = max(trainable_lengths)）询问：对于以 loss_mask=1 结尾的 rollout 样本（如 Tool response），如果是误加的 mask 应在 session server 或 convert_samples_to_train_data 中裁剪；如果是有意为之，则 max_len 的取法可能不合适。随后自行撤回："Ignore this cmt. My idea is not correct."，未进入实现变更。
3. 最终 guapisolo（"Goodfix. LGTM."）与 Zhichenzzz 均 APPROVED。

- 下游 whitening / OPD 可能把 masked 位置的 0 变成非零 (documentation): 第 3 个 commit（8643926）澄清 docstring：whitening 统计量本身是 mask-weighted 的，注入的 0 不会偏置统计量；正确性最终由 policy loss 对 inactive token 重新置零、所有 loss reducer 按 loss_mask 加权来保证。
- 以 loss_mask=1 结尾的样本是否应在 session server 裁剪 (question): guapisolo 随后自行撤回："Ignore this cmt. My idea is not correct." 未进入实现变更。

风险与影响

- 风险：
  1. 核心训练路径变更：get_advantages_and_returns_batch 是所有 PPO 训练的必经路径，改动影响所有线上 PPO 实验。全 1 mask 场景有逐位一致回归测试兜底，但带 mask

样本的 advantage 数值会系统性变化——这是意图内的行为修正，不过与旧checkpoint / 旧实验曲线对比时会失真。

2. 依赖 loss_mask 长度对齐：函数对 loss_masks[i][: response_lengths[i]] 做截断——若 loss_mask 与 response 长度不一致（过长静默截断、过短漏掉尾部可训练 token），会产生静默错误。建议后续在 session server 或样本转换处统一保证 mask 长度。
3. 正确性依赖下游 loss：函数输出 masked 位置为 0，但 docstring 也承认 whitening / OPD 会把这些位置变为非零；最终正确性依赖 policy loss 对 inactive token 重新置零、所有 loss reducer 按 loss_mask 加权。任何未按 mask 加权的损失路径都会引入偏差。
4. CP 覆盖有限：压缩与散射在每个 rank 重复执行（输入已 all-gather），2-rank Gloo 测试覆盖了 THD 下 masked 场景，但 3+ rank、非均匀分片、BSHD 布局的 masked 组合未覆盖。
5. 性能：max_len 从 max(response_lengths) 变为 max(trainable_lengths)，长观察块不再撑大 GAE 扫描长度，通常更省；但压缩 / 散射引入了额外索引与 scatter 开销，对全 1 mask 的经典单轮训练是纯开销。

- 影响：
  1. 训练质量（用户侧）：多轮、agentic、tool-use 类 PPO 训练的 advantage 信用分配被修正——终止奖励不再衰减或丢失，masked 观察块不再稀释 credit，全 masked 样本不再产生垃圾梯度。
  2. 系统侧：改动集中在 loss_hub 两个源码文件与两个测试文件，无配置、无部署、无 schema 变更；vanilla_gae / chunked_gae 内核不动，chunked 路径零回归风险。函数签名新增必填参数 loss_masks，compute_advantages 及测试调用方已同步，但外部直接调用 get_advantages_and_returns_batch 的代码需补参。
  3. 团队侧：该修复与 session 样本组装（PR#1759）、多轮样本语义（PR#1916）形成链路，为 agentic PPO 训练的正确性补齐了关键一环；docstring 的语义边界描述可作为同类 mask 处理的参考模板。 - 风险标记：核心训练路径变更，依赖 loss_mask 长度对齐，正确性依赖下游 loss 掩码，多 rank 分片覆盖有限

关联脉络

- PR #1916 (1/2) refactor(rollout): drop --generate-multi-samples and its per-turn sample semantics: 该 PR 改变了多轮轨迹的样本语义（统一返回标量 Sample），多轮 / agentic rollout 中工具与环境观察 token 进入训练样本并带 loss_mask，正是本 PR 修复的 GAE 缺陷的触发场景。
- PR #1759 (2/2) refactor(session): assemble training samples on the session server; records never leave it: 训练样本开始在 session server 组装，loss_mask 的生成与传递链路由此确立；本 PR 首次在 GAE 计算中消费 loss_mask，与样本组装形成完整闭环。
- PR #1829 fix: require explicit off-policy correction for async PPO training: 同为 PPO 训练正确性主题的修复（防止静默错训），与本 PR 一起构成 PPO 训练管线正确性加固的一对变更。