PR #7300 完整报告

verl-project/verl

[algo] fix: carry running_return through observation spans in REINFORCE++ (#7278)

合并时间：2026-08-10 14:23

原文链接：http://prhub.com.cn/verl-project/verl/pull/7300

执行摘要

- 一句话：修复 REINFORCE++ 多轮观察段奖励丢失
- 推荐动作：值得精读。这是一个小而精准的正确性修复：展示了如何用与 GAE 一致的 carry-through 模式统一处理 response_mask 的双重语义，避免在同一代码库中出现两套不一致的 mask 行为。测试设计覆盖了核心回归、gamma 折扣、padding、batch 和单轮兼容性，可作为 RL 算法修改的测试范例。建议重点关注 running_return 选择逻辑与测试中对 observation/padding 的区分。

功能与动机

issue #7278 给出了精确复现：在相同 4 个 assistant token 中间插入 2 个 observation token（mask=0）后，gamma=1 时 valid-token return 从 [1.0, 1.0, 1.0, 1.0] 退化为 [0.0, 0.0, 1.0, 1.0]。issue 指出 AgentLoopManager.generate_sequences 的 response_mask=0 同时表示工具 observation 与尾部 padding，而 REINFORCE++ 反向扫描对每个位置直接乘 mask，等于把 observation 当作 episode 终点；GAE 已有正确的 carry-through 处理（254-256 行），REINFORCE++ 应复用同一模式。

实现拆解

1. 根因定位：verl/trainer/ppo/core_algos.py 的 compute_reinforce_plus_plus_outcome_advantage（约 718 行）在反向扫描中，每步先计算 running_return = token_level_rewards[:, t] + gamma * running_return，随后立即执行 running_return = running_return * response_mask[:, t]。AgentLoopManager.generate_sequences 中工具观察与尾部 padding 的 response_mask 均为 0，因此 observation 段会把 running_return 清零，等价于在此处提前结束轨迹，使 observation 之前的 action token 无法拿到最终 outcome reward。
2. 修复方案：将“更新后清零”改为 GAE（同文件 254-256 行）已有的 carry-through 模式。新增中间变量 new_running_return，returns[:, t] 仅对 mask=1 的位置写入，running_return 则按 mask 选择：mask=1 时推进为 new_running_return，mask=0 时保持旧值。这样 observation 段既不会终结轨迹，也不会消耗一步 gamma 折扣。
3. 兼容性论证：当 response_mask 全为 1 时，新公式 running_return = new_running_return * 1 + running_return * 0 退化为原逻辑，单轮训练行为完全不变。
4. 测试配套：新增 tests/trainer/ppo/test_reinforce_pp_multiturn_on_cpu.py，共 6 个用例，分别验证：跨 observation 段奖励传播、observation 位置返回 0、尾部 padding 保持 0、

gamma 折扣跨段仅按有效 token 计算、batch 维度独立处理、无 observation 时结果不变。既有 tests/trainer/ppo/test_core_algos_on_cpu.py 23 个用例全部通过，未触发回归。

关键文件：

- verl/trainer/ppo/core_algos.py（模块 训练核心；类别 source；类型 core-logic；符号 compute_reinforce_plus_plus_outcome_advantage）：核心修复文件：compute_reinforce_plus_plus_outcome_advantage 的反向 return 扫描从“每步乘 mask 清零”改为 GAE 式 carry-through，observation 位置不再阻断奖励传播。
- tests/trainer/ppo/test_reinforce_pp_multiturn_on_cpu.py（模块 单测；类别 test；类型 test-coverage；符号 _config, TestReinforcePPMultiTurn, test_observation_span_does_not_block_returns, test_observation_positions_have_zero_returns）：新增 6 个回归测试，直接覆盖 issue 复现场景及 gamma、padding、batch、单轮兼容等边界，构成修复的验证支柱。

关键符号：compute_reinforce_plus_plus_outcome_advantage

关键源码片段

verl/trainer/ppo/core_algos.py

核心修复文件：compute_reinforce_plus_plus_outcome_advantage 的反向 return 扫描从“每步乘 mask 清零”改为 GAE 式 carry-through，observation 位置不再阻断奖励传播。

```
# verl/trainer/ppo/core_algos.py
# REINFORCE++ 优势估计：反向扫描计算 return
@register_adv_est(AdvantageEstimator.REINFORCE_PLUS_PLUS)
def compute_reinforce_plus_plus_outcome_advantage(
    token_level_rewards: torch.Tensor,
    response_mask: torch.Tensor,
    config: Optional[AlgoConfig] = None,
    **kwargs,
) -> tuple[torch.Tensor, torch.Tensor]:
    assert config is not None
    gamma = config.gamma
    with torch.no_grad():
        returns = torch.zeros_like(token_level_rewards)
        running_return = 0

        for t in reversed(range(token_level_rewards.shape[1])):
            # 先计算本位置更新后的累计 return
            new_running_return = token_level_rewards[:, t] + gamma * running_return

            # 仅对 mask = 1 的有效 token 写入 return；mask = 0 的观察与填充位置保持 0
            returns[:, t] = new_running_return * response_mask[:, t]

            # Carry-through：mask = 1 时推进 running_return，mask = 0 时原样保留。
            # 观察段既不终结轨迹，也不消耗 gamma 折扣，与 GAE（同文件 254-256 行）保持一致
            running_return = (
                new_running_return * response_mask[:, t]
```

```
                + running_return * (1 - response_mask[:, t])
            )

            advantages = verl_F.masked_whiten(returns, response_mask)
            advantages = advantages * response_mask

        return advantages, returns
```

tests/trainer/ppo/test_reinforce_pp_multiturn_on_cpu.py

新增 6 个回归测试，直接覆盖 issue 复现场景及 gamma、padding、batch、单轮兼容等边界，构成修复的验证支柱。

```
# tests/trainer/ppo/test_reinforce_pp_multiturn_on_cpu.py
# 核心回归：插入观察段后，valid token 的 return 应与无观察序列一致
def test_observation_span_does_not_block_returns(self):
    config = _config(gamma=1.0)

    compact_mask = torch.ones(1, 4)
    compact_rewards = torch.tensor([[0.0, 0.0, 0.0, 1.0]])

    expanded_mask = torch.tensor([[1.0, 1.0, 0.0, 0.0, 1.0, 1.0]])
    expanded_rewards = torch.tensor([[0.0, 0.0, 0.0, 0.0, 0.0, 1.0]])

    _, compact_returns = compute_reinforce_plus_plus_outcome_advantage(
        compact_rewards, compact_mask, config=config
    )
    _, expanded_returns = compute_reinforce_plus_plus_outcome_advantage(
        expanded_rewards, expanded_mask, config=config
    )

    torch.testing.assert_close(
        compact_returns[compact_mask.bool()],
        expanded_returns[expanded_mask.bool()],
    )
```

评论区精华

PR 本身没有实质 review 评论，wuxibin89 直接 APPROVED；评论仅 CLAassistant 的签署检查（作者已签）。真正的技术讨论发生在 issue #7278 中：社区详细分析了 mask 双重语义（observation 与 padding 均为 0），对比了 GAE 的正确实现，并给出了可重复的最小复现脚本——这些直接决定了修复路径的选择。

- REINFORCE++ observation span 处 running_return 被清零导致奖励丢失 (correctness): 采用 GAE（core_algos.py 254-256 行）的 carry-through 模式：mask=0 时保留 running_return，observation 不终结轨迹也不消耗 gamma 折扣。修复后复现结果与紧凑序列一致，单轮行为不变。

风险与影响

- 风险：变更位于 trainer 核心优势计算路径，所有使用 REINFORCE++ 的训练都会经过该函数。具体风险： 1) running_return 的语义从“每步清零”变为“按 mask 选择”，需确保尾部 padding（同样 mask=0）不会被错误地当作 observation 跳过——新公式下 padding 后 running_return 也不清零，但因为 padding 之后没有后续 token，return 不受影响；测试 test_trailing_padding_stays_zero 覆盖了该点。 2) 多轮训练中 gamma 折扣不再跨 observation 消耗，可能改变长序列的优势数值分布，需要对超参重新观察。 3) 当前只有 CPU 上的单元测试，缺少真实多轮 e2e（如工具调用）训练对比，建议补充。
  - 影响：影响所有使用 REINFORCE++ 做多轮 RL（Agent/ 工具调用）的用户：修复前 observation 之前的 action 因丢失 reward 而梯度被压低，可能造成训练不收敛或次优策略；修复后奖励正确传播，与 GAE 的行为语义对齐。单轮训练（无 observation）结果完全不变，因此对现有单轮实验无回归风险。仓库层面，core_algos.py 是共享算法文件，但改动向后兼容。新增测试文件为后续多轮 RL 回归提供了基础。
  - 风险标记：核心算法路径变更，多轮训练行为变化，缺少真实多轮 e2e 验证

关联脉络

- PR #7337 [ci] chore: add three baselines for npu's nightly ci: 与本 PR 共用文件 tests/trainer/ppo/test_reinforce_pp_multiturn_on_cpu.py：本 PR 新增的回归测试在后续被扩展为 NPU nightly CI 基线，说明该测试已纳入持续验证体系。