PR #6348 完整报告

verl-project/verl

[algo] fix: vectorized grpo low-variance scaling

合并时间：2026-05-14 21:47

原文链接：http://prhub.com.cn/verl-project/verl/pull/6348

执行摘要

- 一句话：修复向量化 GRPO 低方差组的缩放异常
- 推荐动作：该 PR 值得精读，尤其是 groupwise.py 中方差计算的数值稳定性改进，以及 core_algos.py 中 eps 传递的设计权衡。建议熟悉 GRPO 实现细节的工程师关注。变更小而精确，测试覆盖充分，可以快速合并。

功能与动机

PR body 指出 compute_grpo_vectorized_outcome_advantage() 旨在避免 Python 分组开销的同时匹配标准 GRPO 优势实现。但对于低方差奖励组，向量化路径会悄无声息地将归一化优势缩小数个数量级（示例中 canonical GRPO 给出 (-0.6196, 0.6196)，而向量化版本仅 (-0.0050, 0.0050)）。这个 bug 不会导致崩溃或报错，但会削弱低方差组的策略梯度信号，影响训练效果。

实现拆解

1. 重构 verl/utils/groupwise.py 中的 group_mean_std 函数：将方差计算从 s2 - (s1*s1)/count 的朴素公式改为基于中心化分数的 centered = scores - mean[gidx] 与 index_add_ 累加。新公式具有更好的数值稳定性，尤其适用于组内奖励值相近的低方差场景。
2. 修改 verl/trainer/ppo/core_algos.py 中 compute_grpo_vectorized_outcome_advantage 的调用：将传给 group_mean_std 的 eps 参数从原先的 epsilon 函数参数（默认 1e-6）改为硬编码 0.0。这样做的目的是去除内置的方差下限，让调用侧已有的 std_g[g] + epsilon 成为唯一的数值稳定措施，避免双重稳定导致缩放异常。
3. 新增低方差回归测试：在 tests/trainer/ppo/test_core_algos_on_cpu.py 中添加 test_grpo_vectorized_matches_original_for_low_variance_rewards，使用 PR body 中的极低方差样例（四个奖励值 1.0, 1.00001, 2.0, 2.00001 分为两组）验证向量化 GRPO 和原始 GRPO 输出一致。在 tests/utils/test_groupwise.py 中添加 test_group_mean_std_low_variance_matches_torch_std，直接验证 group_mean_std 的输出与 PyTorch 标准 torch.std 一致。

关键文件：

- verl/utils/groupwise.py（模块 工具函数；类别 source；类型 core-logic）：核心数值逻辑修复：改用中心化求和公式计算方差，提升低方差场景的稳定性。

- verl/trainer/ppo/core_algos.py（模块 训练器；类别 source；类型 core-logic）：修复向量化 GRPO 函数，将 eps 改为 0.0，消除双重稳定化。
- tests/trainer/ppo/test_core_algos_on_cpu.py（模块 核心算法；类别 test；类型 test-coverage；符号 test_grpo_vectorized_matches_original_for_low_variance_rewards）：新增低方差回归测试，验证向量化 GRPO 与原始 GRPO 输出一致。
- tests/utils/test_groupwise.py（模块 分组工具；类别 test；类型 test-coverage；符号 test_group_mean_std_low_variance_matches_torch_std）：新增对 group_mean_std 低方差行为的单元测试，验证其与 torch.std 一致。

关键符号：group_mean_std, compute_grpo_vectorized_outcome_advantage

关键源码片段

verl/utils/groupwise.py

核心数值逻辑修复：改用中心化求和公式计算方差，提升低方差场景的稳定性。

```
# verl/utils/groupwise.py (group_mean_std 函数，关键变更部分 )

ones = torch.ones_like(scores, dtype=torch.float32)

count = torch.zeros(G, device=target, dtype=torch.float32).index_add_(0, gidx, ones)
s1 = torch.zeros(G, device=target, dtype=torch.float32).index_add_(0, gidx, scores)
# 原实现 : s2 = ... index_add_(0, gidx, scores * scores)
# var_num = s2 - (s1 * s1) / count ( 朴素求和，低方差时精度损失严重 )

mean = s1 / count.clamp_min(1.0)
# 新实现 : 先计算中心化分数，再用 index_add 累加平方和，数值稳定性更好
centered = scores - mean[gidx]
var_num = torch.zeros(G, device=target, dtype=torch.float32).index_add_(
    0, gidx, centered * centered
)

denom = (count - 1.0).clamp_min(1.0)
var = var_num / denom
std = torch.sqrt(torch.clamp(var, min=eps))

# 单例组回退：mean=0, std=1
single = count <= 1.0
if torch.any(single):
    mean = mean.clone()
    std = std.clone()
    mean[single] = 0.0
    std[single] = 1.0

return mean, std, count
```

verl/trainer/ppo/core_algos.py

修复向量化 GRPO 函数，将 eps 改为 0.0，消除双重稳定化。

```
# verl/trainer/ppo/core_algos.py (compute_grpo_vectorized_outcome_advantage)

with torch.no_grad():
    scores = token_level_rewards.sum(dim=-1)
    g = as_torch_index(index, device=scores.device)
    # 关键修复：eps=0.0 避免内置方差底板，使调用侧 (std_g[g] + epsilon) 成为唯一稳定化措施
    mean_g, std_g, _ = group_mean_std(scores, g, eps=0.0, device=scores.device)
    if norm_adv_by_std_in_grpo:
        scalars = (scores - mean_g[g]) / (std_g[g] + epsilon)
    else:
        scalars = scores - mean_g[g]
    advantages = scalars.unsqueeze(-1) * response_mask
    return advantages, advantages
```

tests/trainer/ppo/test_core_algos_on_cpu.py

新增低方差回归测试，验证向量化 GRPO 与原始 GRPO 输出一致。

```
# tests/trainer/ppo/test_core_algos_on_cpu.py ( 新增测试 )

def test_grpo_vectorized_matches_original_for_low_variance_rewards():
    # 两个 prompt，每个生成 2 个 response，组内奖励值极其接近
    token_level_rewards = torch.tensor(
        [[1.0], [1.00001], [2.0], [2.00001]], dtype=torch.float32
    )
    response_mask = torch.ones_like(token_level_rewards)
    index = np.array(["prompt-a", "prompt-a", "prompt-b", "prompt-b"], dtype=object)

    # canonical GRPO（逐组循环）
    adv1, ret1 = compute_grpo_outcome_advantage(
        token_level_rewards=token_level_rewards,
        response_mask=response_mask,
        index=index,
    )
    # vectorized GRPO
    adv2, ret2 = compute_grpo_vectorized_outcome_advantage(
        token_level_rewards=token_level_rewards,
        response_mask=response_mask,
        index=index,
    )

    # 修复后两者应一致；修复前低方差组偏差数个数量级
    assert torch.allclose(adv1, adv2, rtol=1e-5, atol=1e-6)
    assert torch.allclose(ret1, ret2, rtol=1e-5, atol=1e-6)
```

评论区精华

无人工 review 评论。自动代码审查 bot (gemini-code-assist[bot]) 仅简要复述了变更内容，未提出问题。wuxibin89 直接批准。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。变更集中在数值计算路径，且通过新增的回归测试覆盖了关键的低方差场景。但需注意：
  - group_mean_std 改用中心化求和可能影响其他调用方（如 RLOO vectorized），但 RLOO 路径通过已有测试 test_rloo_and_vectorized_equivalence 覆盖，且该测试未修改，说明新公式兼容已有场景。
  - 硬编码 eps=0.0 意味着任何潜在的被零除风险完全由调用侧的 + epsilon 承担，若以后某处调用 group_mean_std 时未确保加 epsilon，可能引入除零错误。当前在 GRPO vectorized 中已正确处理，但依赖约定。
  - 影响：只影响使用 algorithm.adv_estimator=grpo_vectorized 的训练实验。修复后，向量化 GRPO 在低方差奖励组中的优势值将与原始 GRPO 完全一致，策略梯度信号恢复正常。不受影响的其他路径包括原始 GRPO、RLOO 及其向量化版本。影响范围小，但修复本身对正确的 GRPO 实现至关重要。
  - 风险标记：数值稳定性修复，低方差路径回归

关联脉络

- 暂无明显关联 PR