PR #2778 完整报告

radixark/miles

fix(metrics): scope episode rewards by adapter

合并时间：2026-08-28 05:05

原文链接：http://prhub.com.cn/radixark/miles/pull/2778

执行摘要

- 一句话：按 AdapterRef 隔离奖励聚合键，修复 Multi-LoRA 指标偏差
- 推荐动作：该 PR 是一个小而精准的修复，值得精读，特别是理解 AdapterRef 在 Multi-LoRA 中的关键作用。测试覆盖了核心场景，推动了指标正确性。

功能与动机

PR 描述指出：每个 Multi-LoRA 适配器拥有独立的 rollout 数据源，因此 (group_index, rollout_id) 并非全局唯一。如果不将适配器身份纳入键，rollout/episode_raw_reward 可能会在某个适配器产生更多训练样本时静默地对其过度加权。这是对 PR #2765 中已识别问题的后续修复。

实现拆解

1. 修改聚合键结构：在 miles/ray/rollout/metrics.py 的 _compute_training_sample_metrics 函数中，将 rewards_by_rollout 字典的键从 tuple[str, int | None, int] 改为 tuple[AdapterRef | None, str, int | None, int]，并在生成 rollout_key 时，在元组前面插入 sample.adapter。
2. 更新文档字符串：在函数 docstring 中补充说明适配器身份用于隔离 ID，因为每个 Multi-LoRA 数据源独立编号。
3. 新增回归测试：在 tests/fast/ray/rollout/test_metrics.py 的 TestTrainingSampleMetrics 类中新增 test_rollout_ids_are_scoped_by_adapter 测试，构造两个适配器（adapter-a 和 adapter-b）使用相同 rollout_id=10 的样本，验证 episode_raw_reward 被正确隔离计算。
4. 导入调整：在源码和测试文件中新增 AdapterRef 的导入。

关键文件：

- miles/ray/rollout/metrics.py（模块 指标计算；类别 source；类型 core-logic；符号 _compute_training_sample_metrics）：核心逻辑变更，修改了奖励聚合键以包含适配器身份
- tests/fast/ray/rollout/test_metrics.py（模块 测试；类别 test；类型 test-coverage；符号 test_rollout_ids_are_scoped_by_adapter）：新增回归测试，验证适配器隔离行为

关键符号：_compute_training_sample_metrics

关键源码片段

miles/ray/rollout/metrics.py

核心逻辑变更，修改了奖励聚合键以包含适配器身份

```
# miles/ray/rollout/metrics.py

def _compute_training_sample_metrics(args: Any, samples: list[Sample]) -> dict[str, float | int]:
    """按 rollout 等权平均奖励，并统计训练样本数。

    会话压缩 (session compaction) 可能将一个 rollout 拆分成多个训练样本。
    样本数包含所有行，但奖励先对共享同一 rollout ID 的同行平均，
    避免长 rollout 仅因样本多而获得更大的指标权重。

    Multi-LoRA 场景下，每个适配器独立编号 rollout ID，因此必须用
    AdapterRef 隔离，否则不同适配器的相同局部 ID 会被错误地合并。
    """
    # 键结构： (adapter, 类型标识 , group_index, 局部编号 )
    rewards_by_rollout: dict[tuple[AdapterRef | None, str, int | None, int], list[float]] = {}
    use_metadata_reward = bool(samples and samples[0].metadata and "raw_reward" in
    samples[0].metadata)

    for position, sample in enumerate(samples):
        # 优先使用 rollout_id；否则用 sample.index；最后退化为 position。
        # 无论哪种情况，都先放入 sample.adapter，确保适配器维度隔离。
        if sample.rollout_id is not None:
            rollout_key = (sample.adapter, "rollout", sample.group_index, sample.rollout_id)
        elif sample.index is not None:
            rollout_key = (sample.adapter, "sample", sample.group_index, sample.index)
        else:
            rollout_key = (sample.adapter, "position", sample.group_index, position)

        raw_reward = sample.metadata["raw_reward"] if use_metadata_reward else sample.get_
        reward_value(args)
        # 仅收集数值型奖励，非数值（如 None）直接跳过。
        if isinstance(raw_reward, Number):
            rewards_by_rollout.setdefault(rollout_key, []).append(raw_reward)

    # 先对每个 rollout 桶求平均，再对所有 rollout 求平均，确保等权。
    rollout_rewards = [sum(rewards) / len(rewards) for rewards in rewards_by_rollout.values()]
    return {
        "num_training_samples": len(samples),
        "episode_raw_reward": sum(rollout_rewards) / len(rollout_rewards) if rollout_rewards else
        0.0,
    }
```

tests/fast/ray/rollout/test_metrics.py

新增回归测试，验证适配器隔离行为

```
# tests/fast/ray/rollout/test_metrics.py

def test_rollout_ids_are_scoped_by_adapter(self):
    args = make_args(reward_key=None)
    # 构造两个不同适配器，使用相同的 rollout_id=10，但奖励不同。
    adapter_a = AdapterRef(name="adapter-a", slot=0)
    adapter_b = AdapterRef(name="adapter-b", slot=1)
    samples = [
        # adapter-a 的 rollout 10 有三个样本，奖励均为 1.0
        make_sample(group_index=0, rollout_id=10, adapter=adapter_a, reward=1.0),
        make_sample(group_index=0, rollout_id=10, adapter=adapter_a, reward=1.0),
        make_sample(group_index=0, rollout_id=10, adapter=adapter_a, reward=1.0),
        # adapter-b 的 rollout 10 只有一个样本，奖励为 0.0
        make_sample(group_index=0, rollout_id=10, adapter=adapter_b, reward=0.0),
    ]

    out = _compute_training_sample_metrics(args, samples)

    # 若不按适配器隔离，则 4 个样本会合并为两个桶 → 平均为 (1.0+1.0+1.0)/3 = 0.75
    # 正确隔离后为两个桶，每个桶各自平均，最终平均为 (1.0+0.0)/2 = 0.5
    assert out["episode_raw_reward"] == pytest.approx(0.5)
```

评论区精华

review 评论中，claude[bot] 经过代码审查后报告“未发现问题”，同时 guapisolo 和 maocheng23 均给予了批准。没有实质性的讨论线程。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。主要风险在于 sample.adapter 的类型可能为 None，但代码中已使用 AdapterRef | None 联合类型，且现有逻辑对 None 值已有处理。此外，改动仅影响指标计算，不涉及训练核心路径，回归风险较小。
- 影响：影响限于训练日志中的 episode_raw_reward 和 num_training_samples 指标计算，尤其对 Multi-LoRA 训练场景影响显著，修复了潜在的错误加权问题。对单适配器场景，由于 sample.adapter 为 None，行为与之前一致。对系统性能无影响。
- 风险标记：指标计算变更，Multi-LoRA 场景，测试覆盖新增

关联脉络

- PR #2765 修复 Multi-LoRA 奖励聚合问题：PR body 明确指出这是对 PR #2765 中发现的预先存在问题的后续修复。