PR #6149 完整报告

verl-project/verl

[trainer] fix: support non-last ragged dim in nested tensor rebuild

合并时间：2026-04-27 10:03

原文链接：http://prhub.com.cn/verl-project/verl/pull/6149

执行摘要

- 一句话：修复嵌套张量重建对非最后 ragged 维的支持
- 推荐动作：该 PR 值得精读，展示了如何以最小改动修复一个因硬编码假设导致的通用性问题。读者可重点关注 nested_tensor_from_tensor_list 中 cat_dim 的推导以及 getattr 回退模式。同时建议跟进 review 中未解决的问题，尤其考虑添加 ragged_dim 参数到 nested_tensor_from_jagged 调用，并评估默认回退值是否应改为 1。

功能与动机

Issue #6152 报告，在在线策略蒸馏（OPD）等路径中，teacher_logprobs 张量的形状为 (batch, [seq_len], topk)，其 ragged 维度是 1（非最后）。PR #6127 引入的 nested_tensor_from_tensor_list 在 torch.cat 时硬编码 dim=-1，导致此类张量在 chunk/index_select 时崩溃。本 PR 修复此问题。

实现拆解

1. 放宽 ragged_idx 断言：在 nested_tensor_from_tensor_list 中，将原先只允许 ragged_idx 等于最后一个维度的断言改为允许 1 ≤ ragged_idx ≤ sample_dim，并计算 cat_dim = ragged_idx - 1 用于 torch.cat。
2. 传播 _ragged_idx 属性：在 concat_nested_tensors、chunk_tensordict、index_select_tensor_dict 中，从输入的嵌套张量读取 _ragged_idx 属性（若不存在则回退到 dim() - 1），并传递给 nested_tensor_from_tensor_list，确保重建后的张量继承正确的 ragged 维度。
3. 添加回归测试：在 tests/test_protocol_v2_on_cpu.py 中新增两个测试，分别验证 chunk_tensordict 和 index_select_tensor_dict 在 ragged_idx=1 的 3D 嵌套张量上表现正确。

关键文件：

- verl/utils/tensordict_utils.py（模块 数据工具；类别 source；类型 core-logic；符号 nested_tensor_from_tensor_list, concat_nested_tensors, chunk_tensordict, index_select_tensor_dict）：核心修改文件，包含 nested_tensor_from_tensor_list、concat_nested_tensors、chunk_tensordict、index_select_tensor_dict 四个函数的改动，实现对任意 ragged 维度的支持。

- tests/test_protocol_v2_on_cpu.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_chunk_tensordict_preserves_3d_nested_tensor_layout_with_non_last_ragged_idx, test_index_select_tensor_dict_preserves_3d_nested_tensor_layout_with_non_last_ragged_idx）：新增两个回归测试，验证非最后 ragged 维的嵌套张量在 chunk 和 index_select 操作中的正确性。

关键符号：nested_tensor_from_tensor_list, concat_nested_tensors, chunk_tensordict, index_select_tensor_dict

关键源码片段

verl/utils/tensordict_utils.py

核心修改文件，包含 nested_tensor_from_tensor_list、concat_nested_tensors、chunk_tensordict、index_select_tensor_dict 四个函数的改动，实现对任意 ragged 维度的支持。

```
def nested_tensor_from_tensor_list(
    tensors: list[torch.Tensor], ragged_idx: int | None = None
) -> torch.Tensor:
    """从张量列表构建嵌套张量，支持指定任意 ragged 维度。

    Args:
        tensors: 张量列表，每个张量可以是 2D+（包含 batch 维度）。
        ragged_idx: ragged 维度的索引（1-based），默认为最后一个维度。

    Returns:
        具有 jagged layout 的嵌套张量，其 `_ragged_idx` 属性被正确设置。
    """
    assert len(tensors) > 0, "Must provide at least one tensor"
    sample_dim = tensors[0].dim()

    # ragged_idx 默认取最后一个维度（保持向后兼容）
    if ragged_idx is None:
        ragged_idx = sample_dim
    # 放宽断言：允许 ragged 维度在 [1, sample_dim] 之间
    assert 1 <= ragged_idx <= sample_dim, (
        f"ragged_idx must be in [1, {sample_dim}]. Got {ragged_idx=} and {sample_dim=}"
    )

    if sample_dim == 1:
        return torch.nested.as_nested_tensor(tensors, layout=torch.jagged)

    # 根据 ragged_idx 计算 torch.cat 的维度（0-based）
    cat_dim = ragged_idx - 1
    # 沿 ragged 维度拼接 values
    values = torch.cat(tensors, dim=cat_dim)
    # 获取各样本在 ragged 维度的长度
    lengths = torch.tensor(
```

```
        [tensor.shape[cat_dim] for tensor in tensors],
        dtype=torch.long,
        device=values.device,
    )
    offsets = torch.zeros(len(tensors) + 1, dtype=torch.long, device=values.device)
    torch.cumsum(lengths, dim=0, out=offsets[1:])

    nested_tensor = torch.nested.nested_tensor_from_jagged(
        values=values, offsets=offsets
    )
    # 设置 _ragged_idx 属性，供后续操作使用
    nested_tensor._ragged_idx = ragged_idx
    return nested_tensor
```

tests/test_protocol_v2_on_cpu.py

新增两个回归测试，验证非最后 ragged 维的嵌套张量在 chunk 和 index_select 操作中的正确性。

```
def test_chunk_tensordict_preserves_3d_nested_tensor_layout_with_non_last_ragged_idx():
    """回归测试：chunk_tensordict 必须能处理 ragged 维度不在最后一维的嵌套张量。"""
    topk = 64
    # 模拟 teacher_logprobs: (batch, [seq_len], topk)
    elements = [
        torch.randn(5, topk),
        torch.randn(8, topk),
        torch.randn(3, topk),
        torch.randn(7, topk),
    ]
    # 显式指定 ragged_idx=1，表示第一个维度（seq_len）是 ragged 维度
    teacher_logprobs = tu.nested_tensor_from_tensor_list(elements, ragged_idx=1)

    input_ids = torch.nested.as_nested_tensor(
        [torch.arange(5), torch.arange(8), torch.arange(3), torch.arange(7)],
        layout=torch.jagged,
    )
    td = tu.get_tensordict({"input_ids": input_ids, "teacher_logprobs": teacher_logprobs})

    # 将 4 个 sample 等分为 2 组
    chunks = tu.chunk_tensordict(td, chunks=2)

    # 验证每个块的 _ragged_idx 得到保留
    assert chunks[0]["teacher_logprobs"]._ragged_idx == 1
    # 验证数据内容正确
    assert torch.equal(chunks[0]["teacher_logprobs"].unbind(0)[0], elements[0])
    assert torch.equal(chunks[0]["teacher_logprobs"].unbind(0)[1], elements[1])
    assert chunks[1]["teacher_logprobs"]._ragged_idx == 1
    assert torch.equal(chunks[1]["teacher_logprobs"].unbind(0)[0], elements[2])
    assert torch.equal(chunks[1]["teacher_logprobs"].unbind(0)[1], elements[3])
```

评论区精华

Review 中 gemini-code-assist[bot] 提出了以下重点关注点：

- nested_tensor_from_jagged 缺少 ragged_dim 参数：当 ragged_idx != 1 时，默认 ragged_dim=1 的构造函数可能引发错误，该问题未在 PR 中修复。
- 默认回退值争议：gemini 建议将所有 getattr(nt, "_ragged_idx", nt.dim()-1) 中的回退值改为 1，以兼容标准 3D 嵌套张量；但作者维持了 dim()-1 以确保向后兼容。
- chunk_tensordict 中硬编码切片：padded_chunks[i][j, :seq_len] 假设 ragged 维度是第一个，当 ragged_idx 不唯一时可能错误；gemini 建议使用 narrow(cat_dim, 0, seq_len)，但未采纳。
- index_select_tensor_dict 缺少 workaround：与 chunk_tensordict 不同，该函数未处理 PyTorch 中 3D+ 嵌套张量 unbind(dim=0) 可能抛出 RuntimeError 的情况，存在潜在崩溃风险。
  - nested_tensor_from_jagged 的 ragged_dim 参数缺失 (correctness): 未修改代码，PR 已合并，风险未被解决。
  - 默认回退 ragged_idx 值选择 (design): 采用 dim()-1 作为默认回退。
  - chunk_tensordict 中切片使用硬编码第一维度 (correctness): 未修改，可能存在错误数据重建的风险。
  - index_select_tensor_dict 未处理 3D+ 嵌套张量 unbind 失败 (correctness): 未添加 workaround，存在潜在崩溃风险。

风险与影响

- 风险：尽管 PR 解决了主要崩溃问题，仍存在风险：
  - ragged_dim 参数未传递：nested_tensor_from_jagged 默认 ragged_dim=1，若传入 ragged_idx=2 的 4D 张量，将错误地将第二维视为 ragged 维度，导致后续操作异常。
  - 默认回退 dim()-1 不安全：对于标准 3D 嵌套张量（如 (batch, [seq_len], hidden)），若 _ragged_idx 属性缺失，回退到 dim()-1 将使用错误维度重建，可能引发静默数据损坏或运行时错误。
  - 缺失 3D+ workaround：index_select_tensor_dict 未实现 chunk_tensordict 中的 padded_tensor 回退路径，在遇到 3D+ 嵌套张量时仍可能崩溃。
  - 测试覆盖不足：测试仅覆盖 ragged_idx=1 的情况，未覆盖 ragged_idx=2 等场景，也未覆盖 concat_nested_tensors 路径。
  - 影响：
  - 用户影响：修复了在线策略蒸馏（OPD）等路径中使用 teacher_logprobs 等 3D 嵌套张量时的崩溃问题，使相关训练流程恢复正常。
  - 系统影响：修改了核心张量重建工具，提升了其对任意 ragged 维度的通用性，但也引入了上述风险。
  - 团队影响：变更集中在 tensordict_utils.py，影响范围明确，但后续开发者需注意回退值的选择和参数传递的完整性。

- 风险标记：ragged_dim 参数未显式传递，默认回退 dim()-1 不安全，缺失 3D+ workaround，测试覆盖不全

关联脉络

- PR #6127 [trainer] fix: preserve jagged tensor layout when rebuilding nested tensors with same sequence length: 引入了 nested_tensor_from_tensor_list 函数，其中硬编码 dim=-1 导致本 PR 修复的 bug。
- PR #6152 [trainer] bug: #6127 breaks distillation — nested_tensor_from_tensor_list assumes ragged dim is always the last: 报告了蒸馏路径中的崩溃问题，直接催生了本修复 PR。