PR #6228 完整报告

verl-project/verl

[reward, trainer] feat: support multi-output trajectories in async reward scoring

合并时间：2026-05-09 20:18

原文链接：http://prhub.com.cn/verl-project/verl/pull/6228

执行摘要

- 一句话：支持多输出轨迹异步奖励评分
- 推荐动作：此 PR 值得精读，尤其是设计权衡部分（修改 reward manager 接口 vs 调用 score_batch）。它展示了如何在保持向后兼容的同时引入灵活的多输出支持，对理解异步奖励架构有帮助。

功能与动机

PR body 原话：When using the TransferQueue trainer with multi-turn agent loops, each prompt may produce a list of AgentLoopOutputs. Previously _compute_score was called only on the final output, constructing a batch of size 1. This change passes all outputs to _compute_score so reward managers can be extended to score intermediate turns, while the default behaviour (scoring only the last output via data[-1:]) is preserved for all existing reward managers. Also fixes a pre-existing bug in AgentLoopWorkerTQ._agent_loop_postprocess where _compute_teacher_logprobs was called with the raw output parameter (which may be a list[AgentLoopOutput]) instead of the resolved final_output.

实现拆解

1. AgentLoopWorker._compute_score 接口重构：将签名从接收单个 output 和多个分离 tensor 改为接收 list[AgentLoopOutput] 和 kwargs。内部遍历所有输出，重新计算每个输出的 prompts、responses、input_ids、attention_mask、position_ids，并通过 pad_sequence 打包成批量 TensorDict，最后只将奖励赋值给 outputs[-1]。
2. 同步训练器 _agent_loop_postprocess 简化：在 verl/trainer/main_ppo_sync.py 中，移除手动构造最终输出 tensors 的逻辑，直接调用 self._compute_score(outputs, kwargs=kwargs)，并将 _compute_teacher_logprobs 的调用参数从原始 output 修正为 final_output。
3. RewardLoopWorker.compute_score 放宽断言：移除 assert len(data)==1，允许传入多输出序列，并在使用蒸馏奖励模型时通过 data[-1:] 仅取最后一项。
4. 所有内置奖励管理器 (naive, dapo, gdpo, limited, remote) 的 run_single 方法：将开头的 assert len(data)==1 替换为 data = data[-1:]，保证默认行为只评分最后输出，同时为自定义管理器留下扩展空间。

5. 测试与文档同步更新：修改 test_rate_limited_reward_manager_on_cpu.py 以适配新接口，并更新 docs/advance/reward_loop.rst 文档反映新的参数格式。

关键文件：

- verl/experimental/agent_loop/agent_loop.py（模块 代理循环；类别 source；类型 core-logic；符号 _compute_score）：核心变更点：重构 _compute_score 以支持多输出批量评分，是本 PR 的主要实现。
- verl/trainer/main_ppo_sync.py（模块 训练器；类别 source；类型 core-logic；符号 _agent_loop_postprocess）：同步训练器的 _agent_loop_postprocess 简化，移除手动构造 tensors 并修复 teacher logprobs 参数。
- verl/experimental/reward_loop/reward_loop.py（模块 奖励循环；类别 source；类型 core-logic；符号 compute_score）：RewardLoopWorker.compute_score 移除单数据断言，并确保蒸馏奖励模型只取最后一个输出。
- verl/experimental/reward_loop/reward_manager/dapo.py（模块 奖励管理器；类别 source；类型 core-logic；符号 run_single）：DAPO 奖励管理器移除 assert，改为 data[-1:] 以保证默认只评分最后输出。
- verl/experimental/reward_loop/reward_manager/gdpo.py（模块 奖励管理器；类别 source；类型 core-logic；符号 run_single）：GDPO 奖励管理器移除 assert，改为 data[-1:]。
- verl/experimental/reward_loop/reward_manager/limited.py（模块 奖励管理器；类别 source；类型 core-logic；符号 run_single）：Limited 奖励管理器移除 assert，改为 data[-1:]。
- verl/experimental/reward_loop/reward_manager/naive.py（模块 奖励管理器；类别 source；类型 core-logic；符号 run_single）：Naive 奖励管理器移除 assert，改为 data[-1:]。
- verl/experimental/reward_loop/reward_manager/remote.py（模块 奖励管理器；类别 source；类型 core-logic；符号 run_single）：Remote 奖励管理器移除 assert，改为 data[-1:]。
- tests/experimental/reward_loop/test_rate_limited_reward_manager_on_cpu.py（模块 测试；类别 test；类型 test-coverage）：测试适配新版接口，使用 numpy 数组而非标量。
- docs/advance/reward_loop.rst（模块 文档；类别 docs；类型 documentation）：文档更新反映新的接口参数。

关键符号：_compute_score, _agent_loop_postprocess, compute_score, run_single

关键源码片段

verl/experimental/agent_loop/agent_loop.py

核心变更点：重构 _compute_score 以支持多输出批量评分，是本 PR 的主要实现。

```
async def _compute_score(self, outputs: list[AgentLoopOutput], kwargs: dict) -> None:
    # Compute reward score for all outputs; default reward manager uses last output only.
    enable_async_reward = self.reward_loop_worker_handles is not None
    final_output = outputs[-1]
```

```
    if final_output.reward_score is None and enable_async_reward:
        # Gather tensors from each output and pad into a batch
        all_prompts, all_responses, all_input_ids, all_attention_mask, all_position_ids = [], [], [], [],
        []
        for output in outputs:
            prompts = torch.tensor(output.prompt_ids, dtype=torch.int64)
            responses = torch.tensor(output.response_ids, dtype=torch.int64)
            input_ids = torch.cat([prompts, responses], dim=0)
            attention_mask = torch.ones_like(input_ids, dtype=torch.int64)
            multi_modal_inputs = self._compute_multi_modal_inputs(output, input_ids)
            position_ids = self._compute_position_ids(
                input_ids.unsqueeze(0), attention_mask.unsqueeze(0), multi_modal_inputs
            ).squeeze(0)
            all_prompts.append(prompts)
            all_responses.append(responses)
            all_input_ids.append(input_ids)
            all_attention_mask.append(attention_mask)
            all_position_ids.append(position_ids)

        n = len(outputs)
        data = TensorDict({
            "prompts": torch.nn.utils.rnn.pad_sequence(all_prompts, batch_first=True, padding_
            value=0),
            "responses": torch.nn.utils.rnn.pad_sequence(all_responses, batch_first=True, padding_
            value=0),
            "attention_mask": torch.nn.utils.rnn.pad_sequence(all_attention_mask, batch_first=True,
            padding_value=0),
            "input_ids": torch.nn.utils.rnn.pad_sequence(all_input_ids, batch_first=True, padding_
            value=0),
            "position_ids": torch.nn.utils.rnn.pad_sequence(all_position_ids, batch_first=True,
            padding_value=0),
        })
        # Send batch to reward loop worker; reward manager selects last item
        selected_reward_loop_worker_handle = random.choice(self.reward_loop_worker_handles)
        result = await selected_reward_loop_worker_handle.compute_score.remote(data)
        final_output.reward_score = result # assign score to final output
```

verl/trainer/main_ppo_sync.py

同步训练器的 _agent_loop_postprocess 简化，移除手动构造 tensors 并修复 teacher logprobs 参数。

```
  async def _agent_loop_postprocess(
      self, output: AgentLoopOutput | list[AgentLoopOutput], validate, **kwargs
  ) -> None:
      """Put agent loop outputs into TransferQueue."""
      uid, session_id = kwargs["uid"], kwargs["session_id"]
      outputs = output if isinstance(output, list) else [output]
      if not outputs:
          logger.warning(f"Empty output for prompt {uid}_{session_id}")
```

```
        return

    # Pass all outputs to _compute_score; default reward manager scores only the last.
    await self._compute_score(outputs, kwargs=kwargs)

    final_output = outputs[-1]
    # TODO: Support output:list[AgentLoopOutput]
    await self._compute_teacher_logprobs(
        final_output,
        prompt_ids=final_output.prompt_ids,
        response_ids=final_output.response_ids,
        validate=validate,
        sample_kwargs=kwargs,
    )

    if final_output.reward_score is not None:
        for output in outputs[:-1]:
            output.reward_score = final_output.reward_score
            output.extra_fields["reward_extra_info"] = final_output.extra_fields["reward_extra_info"]

    # NOTE: agent loop may has multiple outputs, put each output into TransferQueue.
    # key format: {uid}_{session_id}_{index}
    keys, fields, tags = [], [], []
    for i, output in enumerate(outputs):
        prompts = torch.tensor(output.prompt_ids, dtype=torch.int64)
        responses = torch.tensor(output.response_ids, dtype=torch.int64)
        input_ids = torch.cat([prompts, responses], dim=0)
        attention_mask = torch.ones_like(input_ids, dtype=torch.int64)
        multi_modal_inputs = self._compute_multi_modal_inputs(output, input_ids)
        position_ids = self._compute_position_ids(
            input_ids.unsqueeze(0), attention_mask.unsqueeze(0), multi_modal_inputs
        ).squeeze(0)

        keys.append(f"{uid}_{session_id}_{i}")
        field = output.as_dict()
        field.update(kwargs)
        field.pop("multi_modal_data", None)
        # ... remaining transfer logic
```

verl/experimental/reward_loop/reward_loop.py

RewardLoopWorker.compute_score移除单数据断言，并确保蒸馏奖励模型只取最后一个输出。

```
  async def compute_score(self, data: DataProto) -> dict:
    if self.config.reward.custom_reward_function.path is not None:
        # directly use user-customized reward function
        return await self.reward_manager.run_single(data)
    else:
        if self.config.reward.reward_model.enable:
            # we assume the rm is disrm
```

```
            # genrm must set custom_reward_function
            return await self.compute_score_disrm(data[-1:])
        else:
            return await self.reward_manager.run_single(data)
```

评论区精华

- 自定义奖励函数兼容性（wuxibin89 提问）：担心移除 assert len(data)==1 后自定义奖励函数可能收到多输出。guillemgt 回应：非自定义管理器通过 data[-1:] 保持相同行为；自定义管理器可自行处理多输出。
- 批量评分 vs 逐次评分设计选择（yyDing1 建议改用 compute_score_batch）：guillemgt 指出逐次评分会假设输出独立，无法支持需要联合考虑所有输出的奖励函数（如 arXiv:2505.20622 算法 1），因此选择修改 reward manager 接口。
- 多步 teacher logprobs 缺失（gemini-code-assist[bot] 提出）：在线蒸馏场景下中间步缺少 teacher logprobs。guillemgt 承认这是待办事项，但超出本 PR 范围。
- 自定义奖励函数兼容性 (question): 确认兼容，自定义管理器可自行扩展。
- 使用 compute_score_batch 替代修改 reward manager (design): 保持当前方案，即修改 reward manager。
- 多步 teacher logprobs 缺失 (correctness): 标记为 TODO，后续处理。

风险与影响

- 风险：
  - 自定义奖励管理器兼容性风险：如果用户的外部奖励管理器仍保留 assert len(data)==1，使用多输出轨迹时会报错。建议用户在升级时检查自定义 reward manager 的 run_single 方法。
  - 冗余计算：_compute_score 中为每个输出重新计算 multi_modal_inputs 和 position_ids（这些在 _agent_loop_postprocess 中已算过），存在轻微性能浪费，但影响不大。
  - 多输出 teacher logprobs 缺失：对于启用在线蒸馏的用户，中间轨迹的 teacher logprobs 未被计算，可能影响蒸馏效果。已在代码中标记 TODO。
- 影响：
  - 用户影响：使用 TransferQueue trainer 和多轮 agent loop 的用户可直接受益，无需更改配置即可让奖励管理器接收完整轨迹；现有单输出训练流程完全向后兼容。
  - 系统影响：涉及 agent_loop、trainer、reward_loop 及所有内置 reward manager，但每个文件改动量很小（+66/-60），测试覆盖通过。
  - 团队影响：为后续扩展多轮奖励评分提供了清晰的接口基点，设计讨论（批量 vs 逐次）可作为团队尝参。
  - 风险标记：自定义 reward manager 兼容性，多输出 teacher logprobs 缺失，冗余计算

关联脉络

- 暂无明显关联 PR