

PR #6308 完整报告

verl-project/verl

[trainer] fix: combine REMAX sampled and greedy samples in one rollout request

合并时间: 2026-05-11 16:17

原文链接: <http://prhub.com.cn/verl-project/verl/pull/6308>

执行摘要

- 一句话: 合并 REMAX 采样与贪心基线样本为一个 rollout 请求
- 推荐动作: 该 PR 解决了异步 vLLM 场景下 REMAX 的竞态问题, 设计简洁且影响面小。值得 REMAX 用户更新, 也值得其他算法开发者关注 `__do_sample__` 这种按样本控制采样参数的实现模式。

功能与动机

REMAX 算法需要每个 prompt 的采样 rollout 和一个贪心基线样本。在异步 vLLM 的多轮 agent 路径中, 先后发出两个 rollout 请求时, 第二个贪心基线请求可能在 rollout 副本休眠后发出, 可能使 vLLM 引擎处于无效状态, 导致运行时失败。PR 旨在通过合并请求从根本上避免该竞态条件, 修复 issue #3027。

实现拆解

1. Agent 循环层支持按样本控制采样参数 (`verl/experimental/agent_loop/agent_loop.py`): 新增局部函数 `apply_greedy_sampling_params`, 将采样参数置为贪心 (`temperature=0`, `top_p=1.0`, `top_k=-1`); `generate_sequences` 方法识别内部控制字段 `__do_sample__`: 当该字段为 `False` 且非验证模式时, 对该样本应用贪心参数; `__do_sample__` 在转发给具体 agent 循环前被过滤移除, 避免下游报错。
2. 训练器统一 rollout 请求 (`verl/trainer/ppo/ray_trainer.py` 的 `fit` 方法和 `verl/experimental/separation/ray_trainer.py` 的 `_fit_generate` 方法): 在 REMAX 分支中, 将采样 batch 的 `__do_sample__` 置为 `True`, 使用 `gen_batch.slice(0, None)` 创建基线 batch (替代 `deepcopy`), 将其 `__do_sample__` 置为 `False`, 然后通过 `DataProto.concat` 合并两个 batch。之后统一调用一次 `generate_sequences` 进行 rollout。
3. 拆分合并输出: rollout 完成后, 从合并输出中通过 `slice(0, num_sampled_prompts)` 切分采样输出, 并移除 `__do_sample__` 字段。基线输出仅用于计算 `reward_baselines`, 不再合并回主训练 batch。原代码中的 `gen_max` 计时器被移除, 因为生成开销已包含在统一的 `gen` 计时器中。
4. 清理旧代码: 移除了不再需要的 `deepcopy` 导入, 以及 `separation/ray_trainer.py` 中未使用的 `torch` 导入。
5. 保持兼容性: 不涉及公共 API 或配置变更, 现有 REMAX 配置文件无需修改。

关键文件:

- verl/experimental/agent_loop/agent_loop.py (模块 Agent 循环; 类别 source; 类型 core-logic; 符号 apply_greedy_sampling_params, generate_sequences) : 新增按样本控制采样参数的支持, 是合并请求的核心接口变更。
- verl/trainer/ppo/ray_trainer.py (模块 PPO 训练器; 类别 source; 类型 dependency-wiring; 符号 fit) : REMAX 合并请求的主要逻辑实现, 将采样和贪心基线 batch 合并后统一 rollout, 再拆分。
- verl/experimental/separation/ray_trainer.py (模块 分离训练器; 类别 source; 类型 dependency-wiring; 符号 _fit_generate) : 与 ray_trainer.py 同步的分离训练器变更, 保持一致。

关键符号: apply_greedy_sampling_params, generate_sequences, fit, _fit_generate

关键源码片段

verl/experimental/agent_loop/agent_loop.py

新增按样本控制采样参数的支持, 是合并请求的核心接口变更。

```

async def generate_sequences(self, batch: DataProto) -> DataProto:
    # ... 前面代码 ...
    config = self.rollout_config
    validate = batch.meta_info.get("validate", False)
    sampling_params = dict(
        temperature=config.temperature,
        top_p=config.top_p,
        top_k=config.top_k,
        repetition_penalty=1.0,
        logprobs=config.calculate_log_probs,
    )

    def apply_greedy_sampling_params(params: dict[str, Any]) -> None:
        # 将采样参数置为贪心模式: 确定性解码
        params["top_p"] = 1.0
        params["top_k"] = -1
        params["temperature"] = 0

    if validate:
        sampling_params["top_p"] = config.val_kwargs.top_p
        sampling_params["top_k"] = config.val_kwargs.top_k
        sampling_params["temperature"] = config.val_kwargs.temperature

    # ... 中间代码 ...

    # NOTE: __do_sample__ 是 REMAX 合并请求的内部控制字段,
    # 不要传递给具体 agent 循环, 避免未知参数错误。
    per_sample_do_sample = batch.non_tensor_batch.get("__do_sample__")
    tasks = []
    for i in range(len(batch)):
        trace_this_sample = i in traced_indices

```

```

kwargs = {k: v[i] for k, v in batch.non_tensor_batch.items() if k != "__do_sample__"}
sample_sampling_params = dict(sampling_params)
# 若非验证模式且该样本标记为不做采样 ( 即贪心基线 ), 则应用贪心参数
if not validate and per_sample_do_sample is not None and not bool(per_sample_do_sample[i]):
    apply_greedy_sampling_params(sample_sampling_params)
tasks.append(
    asyncio.create_task(
        self._run_agent_loop(sample_sampling_params, trajectory_info[i], trace=trace_this_sample, **kwargs)
    )
)
outputs = await asyncio.gather(*tasks)
# ... 后续代码 ...

```

verl/trainer/ppo/ray_trainer.py

REMAX 合并请求的主要逻辑实现, 将采样和贪心基线 batch 合并后统一 rollout, 再拆分。

```

# ... 前面代码 ...
rollout_n = self.config.actor_rollout_ref.rollout.n
gen_batch_output = gen_batch.repeat(repeat_times=rollout_n, interleave=True)

if self.config.algorithm.adv_estimator == AdvantageEstimator.REMAX:
    # NOTE: REMAX 需要每个 prompt 的采样 rollout 和一个贪心基线。
    # 将它们合并到一次 agent-loop/vLLM 请求中, 避免副本休眠后发起第二次请求,
    # 这会使异步 vLLM 多轮 agent 引擎处于无效状态。
    gen_batch_output.non_tensor_batch["__do_sample__"] = np.ones(len(gen_batch_output),
        dtype=bool)
    # 通过 slice 创建基线 batch, 避免 deepcopy 开销
    gen_baseline_batch = gen_batch.slice(0, None)
    gen_baseline_batch.non_tensor_batch["__do_sample__"] = np.zeros(len(gen_baseline_batch),
        dtype=bool)
    combined_gen_batch = DataProto.concat([gen_batch_output, gen_baseline_batch])
    num_sampled_prompts = len(gen_batch_output)
else:
    combined_gen_batch = gen_batch_output
    num_sampled_prompts = len(gen_batch_output)

with marked_timer("gen", timing_raw, color="red"):
    if curr_step_profile:
        self.llm_server_manager.start_profile()
    # 仅一次 rollout 调用
    combined_gen_output = self.async_rollout_manager.generate_sequences(combined_gen_batch)
    self.checkpoint_manager.sleep_replicas()
    if curr_step_profile:
        self.llm_server_manager.stop_profile()
    timing_raw.update(combined_gen_output.meta_info["timing"])
    combined_gen_output.meta_info.pop("timing", None)

```

```
# 切分出采样部分（前 num_sampled_prompts 行）
gen_batch_output = combined_gen_output.slice(0, num_sampled_prompts)
if "__do_sample__" in gen_batch_output.non_tensor_batch:
    gen_batch_output.pop(non_tensor_batch_keys=["__do_sample__"])
# 后续 REMAX 分支从合并输出中取出基线部分计算 reward_baselines
# ... 后续代码 ...
```

评论区精华

- 使用 `DataProto.slice` 替代 `deepcopy`: gemini-code-assist bot 建议将 `deepcopy(gen_batch)` 替换为 `gen_batch.slice(0, None)`, 认为 `slice` 更高效且足够安全。作者接受并在后续提交中实现。
- 移除 `gen_max` 计时器: bot 建议保留该计时器以跟踪基线处理开销; 作者回应称合并后实际生成开销已由 `gen` 计时器覆盖, `gen_max` 仅测量轻量后处理, 不再匹配原始语义, 因此无需保留。最终未采纳, 但审核者 wuxibin89 批准了 PR。
- 未来迁移计划: wuxibin89 指出 `main_ppo.py` 和 `ray_trainer.py` 即将被 `main_ppo_sync.py` 弃用, 作者表示后续另开 PR 支持 ReMax in `main_ppo_sync.py`, 得到赞同。
 - 使用 `DataProto.slice` 替代 `deepcopy (performance)`: 作者接受并提交了两个 commit 分别修改两个 trainer 文件。
 - 移除 `gen_max` 计时器的合理性 (`performance`): 作者未采纳该建议, 审核者 wuxibin89 最终批准, 无进一步异议。
 - 后续迁移到 `main_ppo_sync.py (design)`: 未在本次 PR 中解决, 预计后续另开 PR。

风险与影响

- 风险:
 - 回归风险低: 变更仅影响 REMAX 分支, PPO/GRPO 等算法流程完全不变。
 - batch size 小幅增大: 每个 prompt 额外插入一条基线样本, 可能略增显存占用, 但影响很小。
 - 内部字段冲突风险: `__do_sample__` 以下划线开头标示内部使用, 若其他功能注入相同字段可能冲突, 但当前无其他使用者。
 - 缺少单元测试: 仅通过本地脚本验证, 未添加自动化测试, 未来重构可能引入潜在问题。
- 影响:
 - 用户: 使用 REMAX 进行多轮 agent 训练的用户直接受益, 原 issue #3027 的错误不再出现; 其他算法用户不受影响。
 - 系统: 减少一次 rollout 调用和 replicas sleep, 可能略微提升吞吐。
 - 团队: 代码结构简化, 但后续迁移到 `main_ppo_sync.py` 时需要同步变更。
 - 风险标记: 缺少测试覆盖, 核心路径变更, 内部字段冲突风险

关联脉络

- 暂无明显关联 PR