PR #6153 完整报告

verl-project/verl

[algo, fsdp, megatron, cfg] fix: wire up sum_pi_squared for optimal_token_baseline

合并时间：2026-04-27 13:47

原文链接：http://prhub.com.cn/verl-project/verl/pull/6153

执行摘要

- 一句话：连接 sum_pi_squared 以修复 optimal_token_baseline advantage estimator 的运行时崩溃
- 推荐动作：该 PR 是典型的"功能连接"式 bugfix，展示了如何从配置到引擎再到训练器完整地贯通一个张量。尤其值得关注的是 Megatron TP 下非破坏性实现的设计权衡。对于需要自定义 advantage 估计的开发者，这是一个很好的参考模板。

功能与动机

optimal_token_baseline advantage estimators 需要 Σπ²（对数概率梯度范数代理）才能正常工作，但 actor_config.calculate_sum_pi_squared 从未被 worker 代码读取和填充，导致任何启用该算法的运行在 compute_advantage 中因 data.batch['sum_pi_squared'] 缺失而崩溃。详见 PR body 及其引用的 issue 搜索链接。

实现拆解

1. 配置提升与 YAML 更新：将 calculate_sum_pi_squared 从 FSDPActorConfig 和 McoreActorConfig 提升到共享的 ActorConfig 基类，更新 actor.yaml、dp_actor.yaml、megatron_actor.yaml 和所有 _generated_*.yaml。
2. FSDP 引擎集成：在 prepare_model_outputs 中读取 calculate_sum_pi_squared 标志，在非 fused-kernel 路径中调用 calculate_sum_pi_squared_from_logits 计算 Σπ²，并在 ulysses SP 和非 SP 路径中正确 gather/unpad 并打包为嵌套张量。同时处理 no-rmpad 路径。
3. Megatron 引擎集成：在 tensor_parallel.py 中新增 vocab_parallel_sum_pi_squared 函数（基于 logsumexp 恒等式，非破坏性实现），并在 Megatron 引擎的 forward_step 中 logits_processor 内在 entropy 计算之前调用它，确保输入张量在后续破坏性操作前被安全读取。
4. 训练器数据流：在 _compute_old_log_prob 中读取 calculate_sum_pi_squared 标志，通过 assign_non_tensor 传入 micro_batch，从模型输出中提取 sum_pi_squared，进行 padding 转换后打包进 old_log_prob 字典，使其最终到达 compute_advantage。

关键文件：

- verl/workers/engine/fsdp/transformer_impl.py（模块 FSDP 引擎；类别 source；类型 core-logic）：核心实现：在 FSDP 引擎中读取 calculate_sum_pi_squared 标志，从

logits 计算 Σπ²，并处理 ulysses SP gather 和嵌套张量打包。

- verl/utils/megatron/tensor_parallel.py（模块 TP 工具；类别 source；类型 core-logic；符号 vocab_parallel_sum_pi_squared）：新增 vocab_parallel_sum_pi_squared 函数，在 Tensor Parallel 分片环境下非破坏性地计算 Σπ²。
- verl/workers/engine/megatron/transformer_impl.py（模块 Megatron 引擎；类别 source；类型 dependency-wiring）：在 Megatron 引擎的 forward_step 中集成 sum_pi_squared 计算，在 entropy 之前调用 vocab_parallel_sum_pi_squared。
- verl/trainer/ppo/ray_trainer.py（模块 PPO 训练器；类别 source；类型 core-logic）：训练器集成：在 _compute_old_log_prob 中读取 calculate_sum_pi_squared 标志，从输出提取 sum_pi_squared 并传递给 advantage 计算。
- tests/utils/test_torch_functional.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_calculate_sum_pi_squared_from_logits, test_calculate_sum_pi_squared_from_logits_extreme_values）：新增 CPU 参数化测试，验证 calculate_sum_pi_squared_from_logits 的形状、数值正确性和极端数值稳定性。
- tests/special_distributed/test_tensor_dict.py（模块 分布式测试；类别 test；类型 test-coverage；符号 test_vocab_parallel_sum_pi_squared）：新增分布式 TP 测试，验证 vocab_parallel_sum_pi_squared 的输出与单卡参考一致，并断言非破坏性。

关键符号：vocab_parallel_sum_pi_squared, test_calculate_sum_pi_squared_from_logits, test_calculate_sum_pi_squared_from_logits_extreme_values, test_vocab_parallel_sum_pi_squared

关键源码片段

verl/workers/engine/fsdp/transformer_impl.py

核心实现：在 FSDP 引擎中读取 calculate_sum_pi_squared 标志，从 logits 计算 Σπ²，并处理 ulysses SP gather 和嵌套张量打包。

```
def prepare_model_outputs(self, output, output_args, micro_batch: TensorDict, logits_processor_
func):
    use_remove_padding = tu.get_non_tensor_data(data=micro_batch, key="use_remove_
    padding", default=True)
    pad_mode = tu.get_non_tensor_data(data=micro_batch, key="pad_mode", default=
    DatasetPadMode.NO_PADDING)
    use_fused_kernels = tu.get_non_tensor_data(data=micro_batch, key="use_fused_kernels",
    default=False)
    calculate_entropy = tu.get_non_tensor_data(data=micro_batch, key="calculate_entropy",
    default=False)
    # 读取 sum_pi_squared 标志
    calculate_sum_pi_squared = tu.get_non_tensor_data(
        data=micro_batch, key="calculate_sum_pi_squared", default=False
    )
    distillation_use_topk = tu.get_non_tensor_data(data=micro_batch, key="distillation_use_topk",
    default=False)

    # 提前检查：fused kernel 不暴露完整 logits，无法计算 Σπ²
```

```
if calculate_sum_pi_squared and use_fused_kernels:
    raise NotImplementedError(
        "calculate_sum_pi_squared=True is not supported with use_fused_kernels=True: "
        "fused kernels do not materialize the full logits tensor needed for Σπ²."
    )

model_output = {}
input_ids = micro_batch["input_ids"]

if use_remove_padding:
    input_ids_rmpad_rolled = output_args["input_ids_rmpad_rolled"]
    temperature_rmpad = output_args["temperature_rmpad"]

    if use_fused_kernels:
        log_probs = output.log_probs.squeeze(0)
        entropy_rmpad = output.entropy.squeeze(0)
    else:
        logits_rmpad = output.logits.squeeze(0)
        logits_rmpad.div_(temperature_rmpad.clamp(min=1e-8).unsqueeze(-1).to(logits_rmpad.
        dtype))

        # 计算 log_probs，熵等（省略）
        # ...

        # 计算 Σπ²（新增部分）
        if calculate_sum_pi_squared:
            sum_pi_squared_rmpad = verl_F.calculate_sum_pi_squared_from_logits(logits_rmpad)

    # Ulysses SP gather
    if self.use_ulysses_sp:
        # gather log_probs, entropy（省略）
        if calculate_sum_pi_squared:
            sum_pi_squared_rmpad = gather_outputs_and_unpad(
                sum_pi_squared_rmpad,
                gather_dim=0,
                unpad_dim=0,
                padding_size=pad_size,
            )

    if pad_mode == DatasetPadMode.NO_PADDING:
        cu_seqlens = input_ids.offsets()
        # 打包 log_probs, entropy（省略）
        if calculate_sum_pi_squared:
            sum_pi_squared = torch.nested.nested_tensor_from_jagged(sum_pi_squared_rmpad,
            cu_seqlens)

else:
    # no-rmpad 路径
    if calculate_sum_pi_squared:
```

```
            sum_pi_squared = verl_F.calculate_sum_pi_squared_from_logits(logits)

    # 将 sum_pi_squared 写入 model_output
    if calculate_sum_pi_squared:
        model_output["sum_pi_squared"] = sum_pi_squared

    # ... 后续返回等
```

verl/utils/megatron/tensor_parallel.py

新增 vocab_parallel_sum_pi_squared 函数，在 Tensor Parallel 分片环境下非破坏性地计算 Σπ²。

```
def vocab_parallel_sum_pi_squared(vocab_parallel_logits: torch.Tensor) -> torch.Tensor:
    """Compute Σπ² (sum of squared probabilities) when logits are sharded across tp ranks.

    Used by ``optimal_token_baseline`` advantage estimators as the path-variance proxy:
    ``w_t = 1 - 2*π_t + Σπ²``.

    Args:
        vocab_parallel_logits: (..., vocab_size // tp_size)

    Returns: (...,)

    Implementation is non-destructive (does not mutate ``vocab_parallel_logits``) so it
    can be safely called before ``vocab_parallel_entropy`` / ``vocab_parallel_log_probs``
    which would otherwise consume the same tensor.
    """
    tp_group = mpu.get_tensor_model_parallel_group()

    # 计算全局最大值用于 logsumexp 稳定性
    logits_max = vocab_parallel_logits.max(dim=-1, keepdim=True).values
    dist.all_reduce(logits_max, op=dist.ReduceOp.MAX, group=tp_group)

    # 减去最大值并 exponentiate（不修改原始张量）
    shifted = vocab_parallel_logits - logits_max
    exp_shifted = shifted.exp() # 非破坏性：没有使用 exp_()

    # 跨 TP ranks all-reduce 求和以得到全局 softmax 分母
    sum_exp = exp_shifted.sum(dim=-1, keepdim=True)
    dist.all_reduce(sum_exp, group=tp_group)

    # 跨 TP ranks all-reduce 求和以得到全局概率平方和
    sum_exp_squared = exp_shifted.pow(2).sum(dim=-1, keepdim=True)
    dist.all_reduce(sum_exp_squared, group=tp_group)

    # Σπ² = sum(exp(2*(logit - max))) / (sum(exp(logit - max)))^2
    return (sum_exp_squared / sum_exp.pow(2)).squeeze(dim=-1)
```

评论区精华

1. sum_pi_squared_checkpointing 配置字段位置错误：gemini-code-assist[bot] 指出 sum_pi_squared_checkpointing 在 FSDPActorConfig 中却通过 engine_config 访问，可能导致 AttributeError。作者在后续 commit 中移除了该配置选项，简化了逻辑。
2. vocab_parallel_entropy 的前向 / 反向操作顺序说明：gemini-code-assist[bot] 指出注释声称 vocab_parallel_entropy 在 forward 中用 exp_() 改变输入，但实际 exp_() 在副本上；不过 backward 会改变输入，因此 sum_pi_squared 必须在 entropy 之前调用。作者更新了注释，明确非破坏性约束。

- sum_pi_squared_checkpointing 配置字段位置错误 (correctness): 作者在第三 commit 中移除了 sum_pi_squared_checkpointing 选项，简化了逻辑，消除了问题。
- vocab_parallel_entropy 的前向 / 反向操作顺序说明 (correctness): 作者更新了注释，明确 sum_pi_squared 的非破坏性并确保在 entropy 之前调用，保持当前安全顺序。

风险与影响

- 风险：新增计算增加了前向传播的计算开销（softmax 再平方求和），但仅在主动启用时发生；与 fused kernels 不兼容，已明确抛出 NotImplementedError；Megatron 端涉及两次 all_reduce 通信原语（max 和 sum），会增加通信量；分布式 TP 测试通过非破坏性断言确保了与 entropy 计算的安全顺序。整体风险较低，因为默认关闭且测试覆盖了单机和分布式场景。
- 影响：对用户：需显式设置 actor_rollout_ref.actor.calculate_sum_pi_squared=True 才能启用，默认 False，无侵入性。对系统：无 breaking change，所有现有配置保持兼容。对团队：该 PR 完善了 optimal_token_baseline 算法的数据流水线，使该功能从不可用变为可用。
- 风险标记：核心路径变更，缺少 fused kernel 兼容，新增 all_reduce 通信

关联脉络

- 暂无明显关联 PR