PR #7245 完整报告

verl-project/verl

[algo, cfg, doc] feat: add DRO losses

合并时间：2026-08-05 04:25

原文链接：http://prhub.com.cn/verl-project/verl/pull/7245

执行摘要

- 一句话：新增 DRO 策略损失及 dro_beta 配置
- 推荐动作：值得快速精读，核心价值在于展示如何新增一个注册式策略损失并与统一聚合、rollout 校正权重对接。实现短小清晰，可作为后续扩展其他 policy loss 的样板。建议关注 agg_loss 的复用方式以及 dro_beta 的运行时校验设计；若团队计划推广 DRO，建议补充 GPU 冒烟测试和 beta 超参建议。

功能与动机

PR 正文说明目标是"添加已注册的策略目标：Direct Reward Optimization (DRO)，带可配置的二次 log-ratio 惩罚"，并给出了使用示例。分支名 verl_tinker_importance_sampling_dro 和第一版提交表明最初还计划加入 importance-sampling 损失，但第二次提交将其移除，最终仅保留 DRO，使变更聚焦在单一算法目标上。

实现拆解

实现拆解如下：

1. 注册 DRO 损失函数：在 verl/trainer/ppo/core_algos.py 中新增 compute_policy_loss_dro，通过 @register_policy_loss("dro") 注册。损失表达式为 -(log_prob * advantages - 0.5 * beta * log_ratio.square())，其中 log_ratio = log_prob - old_log_prob；若 dro_beta 缺失或非正则直接抛出 ValueError。该函数复用统一的 agg_loss 聚合入口（支持 token-mean、sample-mean 等模式）与 config.global_batch_info，并支持可选的 rollout_is_weights 逐 token 加权，与现有损失保持行为一致。
2. 扩展配置契约：在 verl/workers/config/actor.py 的 PolicyLossConfig 中新增字段 dro_beta: Optional[float] = None，同时按 reviewer 意见将 loss_mode 的文档字符串补全为所有已注册选项（vanilla、dppo_tv、dppo_kl、gspo、sapo、gpg、clip_cov、kl_cov、geo_mean、dro、cispo、bypass_mode）。
3. 同步 YAML 配置：在 verl/trainer/config/actor/actor.yaml 的 policy_loss 段加入 dro_beta: null 及注释；四个 _generated_ppo_*_trainer.yaml（generic、megatron、torchtitan、veomni）也各加一行 dro_beta: null，确保不同后端实例化 PolicyLossConfig 时不会因缺键报错。

4. 测试与文档配套：新增 tests/trainer/ppo/test_dynamic_policy_losses_on_cpu.py，包含 test_dro_matches_direct_formula_and_requires_positive_beta，用 torch.testing.assert_close 校验公式与 mask 平均，并验证未设置 dro_beta 时抛出 ValueError；新增 docs/algo/dro.md 说明公式与用法，并加入 docs/index.rst 目录树。根据 PR 正文，本地因缺少 pytest 环境未能实际运行测试，仅通过 pre-commit。

关键文件：

- verl/trainer/ppo/core_algos.py（模块 策略损失；类别 source；类型 core-logic；符号 compute_policy_loss_dro）：新增 DRO 损失的核心实现与注册入口，是本次功能的主干。
- tests/trainer/ppo/test_dynamic_policy_losses_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 _actor_config, test_dro_matches_direct_formula_and_requires_positive_beta）：新增 CPU 测试，验证 DRO 公式与正数 dro_beta 校验，是质量保障的核心。
- verl/workers/config/actor.py（模块 配置；类别 source；类型 configuration；符号 PolicyLossConfig）：PolicyLossConfig 新增 dro_beta 字段并补全 loss_mode 文档，是配置契约的核心变更。
- verl/trainer/config/actor/actor.yaml（模块 配置；类别 config；类型 configuration）：基础 actor 配置模板中补充 dro_beta 默认值及注释，是配置分发的源。
- verl/trainer/config/_generated_ppo_megatron_trainer.yaml（模块 配置；类别 config；类型 configuration）：四个生成配置之一，保证 Megatron 后端能解析 dro_beta 字段。
- verl/trainer/config/_generated_ppo_torchtitan_trainer.yaml（模块 配置；类别 config；类型 configuration）：TorchTitan 后端生成配置同步更新 dro_beta 字段。
- verl/trainer/config/_generated_ppo_trainer.yaml（模块 配置；类别 config；类型 configuration）：通用 PPO 生成配置同步更新 dro_beta 字段。
- verl/trainer/config/_generated_ppo_veomni_trainer.yaml（模块 配置；类别 config；类型 configuration）：VeOmni 后端生成配置同步更新 dro_beta 字段。
- docs/algo/dro.md（模块 文档；类别 docs；类型 documentation）：新增 DRO 算法文档，包含公式与配置示例，是用户上手的关键入口。
- docs/index.rst（模块 文档；类别 docs；类型 documentation）：将 dro.md 挂入文档目录树，保证文档可访问。

关键符号：compute_policy_loss_dro

关键源码片段

verl/trainer/ppo/core_algos.py

新增 DRO 损失的核心实现与注册入口，是本次功能的主干。

```
@register_policy_loss("dro")
def compute_policy_loss_dro(
    old_log_prob: torch.Tensor,
    log_prob: torch.Tensor,
    advantages: torch.Tensor,
    response_mask: torch.Tensor,
```

```
    loss_agg_mode: str = "token-mean",
    config: Optional[ActorConfig] = None,
    rollout_is_weights: torch.Tensor | None = None,
) -> tuple[torch.Tensor, dict[str, Any]]:
    # 校验配置：DRO 必须显式提供正数 dro_beta，否则直接报错，避免静默使用错误超参
    assert config is not None
    assert config.policy_loss is not None

    beta = config.policy_loss.dro_beta
    if beta is None or beta <= 0:
        raise ValueError("policy_loss.dro_beta must be a positive value when using DRO")

    # log_ratio 表示新旧策略的对数似然差，用于二次惩罚项
    log_ratio = log_prob - old_log_prob
    # DRO 目标：优势加权对数似然减去 0.5 * beta * log_ratio^2 惩罚；
    # 外层取负使梯度上升方向与目标最大化一致
    pg_losses = -(log_prob * advantages - 0.5 * beta * log_ratio.square())

    # 若提供 rollout 校正权重，则逐 token 乘到损失上，支持 off-policy 数据修正
    if rollout_is_weights is not None:
        pg_losses = pg_losses * rollout_is_weights

    # 统一走 agg_loss 聚合，兼容 token-mean / sample-mean 等模式及全局 batch 信息
    pg_loss = agg_loss(
        loss_mat=pg_losses, loss_mask=response_mask, loss_agg_mode=loss_agg_mode, **config.
        global_batch_info
    )
    pg_metrics = {
        # 暴露近似 KL 作为监控指标，帮助判断新旧策略偏差程度
        "actor/ppo_kl": verl_F.masked_mean(-log_ratio, response_mask).detach().item(),
    }
    return pg_loss, pg_metrics
```

tests/trainer/ppo/test_dynamic_policy_losses_on_cpu.py

新增 CPU 测试，验证 DRO 公式与正数 dro_beta 校验，是质量保障的核心。

```
def _actor_config(*, loss_mode: str, dro_beta=None) -> ActorConfig:
    # 构造最小 ActorConfig，只保留计算 DRO 损失所需字段，降低测试耦合
    return ActorConfig(
        strategy="fsdp",
        rollout_n=1,
        ppo_micro_batch_size_per_gpu=1,
        clip_ratio=0.2,
        clip_ratio_low=0.2,
        clip_ratio_high=0.2,
        loss_agg_mode="token-mean",
        policy_loss=PolicyLossConfig(loss_mode=loss_mode, dro_beta=dro_beta),
    )

def test_dro_matches_direct_formula_and_requires_positive_beta():
```

```
    old_log_prob = torch.tensor([[-1.0, -0.7, -0.2]])
    log_prob = torch.tensor([[-0.8, -0.9, -0.1]], requires_grad=True)
    advantages = torch.tensor([[2.0, -1.5, 9.0]])
    response_mask = torch.tensor([[1.0, 1.0, 0.0]]) # 最后一个 token 被 mask 掉
    beta = 0.05
    config = _actor_config(loss_mode="dro", dro_beta=beta)

    loss, _ = compute_policy_loss_dro(old_log_prob, log_prob, advantages, response_mask,
    "token-mean", config)
    # 用直接公式手工计算期望值，并验证与实现一致
    log_ratio = log_prob - old_log_prob
    expected = (-(log_prob * advantages - 0.5 * beta * log_ratio.square()) * response_mask).
    sum()
    expected = expected / response_mask.sum()
    torch.testing.assert_close(loss, expected)

    # 未配置 dro_beta 时应抛出 ValueError，且错误信息包含 dro_beta 字样
    with pytest.raises(ValueError, match="dro_beta"):
        compute_policy_loss_dro(
            old_log_prob,
            log_prob,
            advantages,
            response_mask,
            "token-mean",
            _actor_config(loss_mode="dro"),
        )
```

评论区精华

Review 中仅有一条实质评论：审核者 Luosuu 在 verl/workers/config/actor.py 第 85 行指出 loss_mode 的文档字符串需要列出全部选项（原文："list all options here"），作者 wyettzeng 回复 "done" 并已补全。该问题属于文档完善类，未引发算法或设计层面的争论。

- loss_mode 文档字符串选项列表补全 (documentation): 已将 loss_mode 选项补全为 'vanilla', 'dppo_tv', 'dppo_kl', 'gspo', 'sapo', 'gpg', 'clip_cov', 'kl_cov', 'geo_mean', 'dro', 'cispo', 'bypass_mode'。

风险与影响

- 风险：技术风险主要集中在以下方面：

1. 缺少分布式 /GPU 验证：新增测试仅在 CPU 上覆盖 token-mean 一种聚合模式，未覆盖 sample-mean、seq-mean 等模式以及真实 PPO 训练下 global_batch_info 的跨设备聚合路径，存在隐性回归风险。
2. 配置错误延迟暴露：dro_beta 默认 None，且校验在损失函数执行时进行（运行时 ValueError），而非在配置解析阶段拦截；用户配置了 loss_mode=dro 却忘记设置 dro_beta 时，会在训练启动后较晚阶段才报错。

3. 无 clipping 的稳定性依赖 beta：DRO 不使用 PPO 的 clip 机制，其稳定性完全依赖 dro_beta 的惩罚强度；若用户设置过小的 beta，对数似然比可能发散。当前文档虽给出公式，但缺少 beta 取值建议。
4. 本地测试未跑通：PR 正文明确说明测试环境缺少 pytest 且系统 PyTorch 不可用（libnvshmem_host.so.3 缺失），虽然作者声明 pre-commit 通过，但测试的实际执行依赖 CI。 - 影响：影响范围限定在策略损失选择层：

- 对用户：新增一种可直接通过配置启用的策略损失 dro，使用方式与现有损失一致，不改变默认行为（loss_mode 默认仍为 vanilla，dro_beta 默认 None）。
- 对系统：PolicyLossConfig 新增一个可选字段，四个 _generated_*.yaml 同步补齐，不会破坏既有配置加载；agg_loss 的复用保证了聚合语义的一致性。
- 对团队：提供了一篇简短的算法文档 docs/algo/dro.md，降低了使用和后续扩展门槛；但算法论文参考（文章链接未在文档中给出）和超参指导仍显不足。

总体影响程度为中低，属于功能增量而不是架构变更。

- 风险标记：运行时而非配置期校验，新算法缺少分布式验证，仅 CPU 测试覆盖部分聚合模式

关联脉络

- PR #7197 [algo, cfg] feat: add token-sum loss aggregation: 在 core_algos.py 中引入统一的 token-sum 聚合模式，本 PR 的 DRO 直接复用 agg_loss 与该聚合框架，二者在损失聚合机制上紧密关联。