PPO × Family 第四讲技术问题 Q&A

Q0: 有没有第四节课内容的大白话总结?

A0:

小节	要点	示例任务
奖励空间概述	• 奖励空间的两个特点 • 奖励塑形 • 模仿学习	/
稀疏奖励和好奇心机制 (ICM)	• 好奇心机制 • ICM — 什么对决策是重要的 • 什么是适于探索的表征	迷宫探索
稀疏奖励和随机蒸馏 (RND)	• ICM 可能失败的场景 • 预测误差的影响因素 • RND 设计思想	
多尺度奖励	• 多尺度奖励空间的定义 • 多尺度奖励空间导致的问题 • 解决多尺度奖励的三大经典方法 (reward clip, PopArt, Value Rescale)	自动驾驶

（表1：课程第四讲一图流总结。）

Q1: 第四讲提到的 exploration 方法似乎是构造引导智能体探索新环境的奖励来缓解稀疏奖励问题，这种 intrinsic reward 随着训练是否最后会趋于 0？另外 ICM 和一些经典的问题（比如老虎机）中的 exploration 方法，比如根据近似的价值函数选取动作来使 regret 值尽快收敛的方法有什么关联?

A1: 在使用内在奖励 (intrinsic reward) 的算法中，随着智能体的训练，内在奖励可能会趋于 0。因为当智能体逐渐熟悉环境并且可以从外在奖励中获得更多的收益时，内在奖励对于指导其探索和学习的作用会逐渐减弱。但是一般来说，不会刻意设置机制使得 intrinsic reward 趋于0（如设置逐步衰减），因为大部分强化学习任务并没有一个确定的收敛终点，一般只是训练一定环境步数或者优化迭代数看对应的结果。

Bandit 算法涉及的是一些经典的探索方法，比较适用于 Bandit 场景，而对于一般常见的多步决策环境且使用 model free RL 算法，intrinsic reward 是更有效的方法。更详细的信息可以参考我们做的这个与 Exploration methods in Reinforcement Learning (ERL) 相关的知识库 [1]。

Q2: 在 ICM 和 RND 这两个经典的内在奖励相关算法之后，还有什么新的衍生算法吗?

A2: 课程中介绍的 ICM，RND 方法提出通过预测问题的误差来度量状态的新颖性，为新颖性大的状态提供一个大的内在奖励，促进探索，这些方法在许多稀疏奖励/探索困难的任务上取得了不错的效果，但是存在一个问题：随着智能体训练步数的增加，预测问题的预测误差开始减小，内在奖励的激励信号变小，即不再鼓励智能体再次访问某些状态，但是有可能这些状态正是获得外在奖励所必须访问的。

为了解决前述探索信号逐渐衰减的问题，在论文 Never Give Up: Learning Directed Exploration Strategies [2] 中，NGU (Never Give Up) 智能体提出了一种新的内在奖励产生机制，融合了2个角度的新颖性：即单局角度上的局内内在奖励和 life-long 角度上的局间内在奖励，此外还提出通过同时学习一组具有不同探索程度的策略 (directed exploratory policies) 来进一步增加探索的多样性。
注：有关 NGU 的具体细节可见我们的系列博客：强化学习探索家 | 集大成者 NGU （1）[3]。

Q3: 第四节课作业理论题第一题中的价值函数 Q 与价值函数 Z 有什么区别？为什么 Z 是一个分布?

A3: 这里是因为 Q 和 Z 的计算公式是有差别的，前者是期望的形式，而后者没有期望。这使得前者是一个数值，而后者是去表示一个分布。在很多实际的决策问题里， 奖励可能不是一个确定的值而是一个分布，比如游戏的里的一些随机性规则，拾取某些宝箱，可能获得不同大小的奖励，不一定是一些固定的数值，可能是遵循某个范围内的分布。Q 天然应该是一个分布，只是以前为了简化，都当作一个期望值来进行计算，而当需要处理奖励本身的内在随机性（intrinsic randomness）问题时，就需要引入 distributional RL 的思路来建模分布，从而做到对价值函数更准确地建模，以及风险（方差）建模等等的处理。

Q4：Value Rescale 在随机性 MDP 问题上如何使用，会出现什么问题?

A4： 在随机 MDP 上，当映射函数 h 和折扣因子 γ 满足如下要求时，贝尔曼算子 $\mathcal{T}_h$ 也是 contraction：

- 映射函数 h 严格单调递增，
- 映射函数 h 是利普希茨连续的 (Lipschitz continuous) ，其利普希茨常数记为 L_h ，
- 映射函数的逆函数 h^{-1} 严格单调递增，
- 映射函数的逆函数 h^{-1} 是利普希茨连续的 (Lipschitz continuous)，其利普希茨常数记为 $L_{h^{-1}}$ ，

- 折扣因子 $\gamma < \frac{1}{L_h L_{h^{-1}}}$。

但是与确定性 MDP 不同，随机性 MDP 上 $\mathcal{T}_h$ 的不动点不一定是 $h \circ Q^*$。

例如，原文映射函数 $h(x) = sign(x)(\sqrt{|x|+1} - 1)$ 满足上述条件，且 h 的利普希茨常数为 $L_h = \frac{1}{2} + \epsilon$，$h^{-1}$ 的利普希茨常数为 $L_{h^{-1}} = \frac{1}{\epsilon}$。因此当折扣因子 $\gamma < 1 - \frac{1}{2\epsilon + 1}$ 时，贝尔曼算子为 contraction [4]。

Q5: 请问对于不同算法而言，TD-error 是越低说明学习效果越好吗?

A5：并不一定。通常来说，对于同一算法，TD-error 越低表示算法的预测精度越高，但是在具体情况下，不同算法的 TD-error 目标可能有所不同。例如，在某些情况下，为了平衡探索和利用，算法可能会有较大偏差，导致 TD-error 略高，并不能说 TD-error 越低效果越好。而对于不同算法，因为数据的分布，奖励的设计等都有很大的差异，因此一般无法直接比较。

Q6: 混合动作空间的 PPO 这里的监控指标，为什么课程代码里监控指标选择只看最大，而不是加在一起看整体呢?

课程示例代码具体位置：

```
elif self._action_space == 'hybrid':
    # discrete part (discrete policy loss and entropy loss)
    ppo_discrete_batch = ppo_policy_data(
        output['logit']['action_type'], batch['logit']['action_type'], batch['action']['action_type'],
        adv, batch['weight']
    )
    ppo_discrete_loss, ppo_discrete_info = ppo_policy_error(ppo_discrete_batch, self._clip_ratio)
    # continuous part (continuous policy loss and entropy loss, value loss)
    ppo_continuous_batch = ppo_data(
        output['logit']['action_args'], batch['logit']['action_args'], batch['action']['action_args'],
        output['value'], batch['value'], adv, batch['return'], batch['weight']
    )
    ppo_continuous_loss, ppo_continuous_info = ppo_error_continuous(
        ppo_continuous_batch, self._clip_ratio
    )
    # sum discrete and continuous loss
    ppo_loss = type(ppo_continuous_loss)(
        ppo_continuous_loss.policy_loss + ppo_discrete_loss.policy_loss, ppo_continuous_loss.value_loss,
        ppo_continuous_loss.entropy_loss + ppo_discrete_loss.entropy_loss
    )
    ppo_info = type(ppo_continuous_info)(
        max(ppo_continuous_info.approx_kl, ppo_discrete_info.approx_kl),
        max(ppo_continuous_info.clipfrac, ppo_discrete_info.clipfrac)
    )
```

（图1：混合动作空间 PPO 的代码实现片段）

A6: 因为 appro_kl 和 clip_frac 反映了新旧策略的差异程度，我们调参的时候是希望它们都别超过某一范围，那么这个时候选择最大值进行观测是较为便捷的方式。最大值没超过，那整体一定在合理范围内。此外 On Policy PPO 如果有多次 appro_kl 超过 0.1，就要着手调整超参数了，适当降低每次训练的 epoch。

Q7：请问课程中，老师提到的材料有哪些?

A7： 【PPO × Family】第四课：解密稀疏奖励空间_哔哩哔哩_bilibili 中提到的链接详情如下：

01: 38 对于奖励空间更详细的解释，大家可以参考论文：

http://aaai-rlg.mlanctot.info/papers/AAAI22-RLG_paper_38.pdf

04: 37 关于 DOTA2 奖励塑性的具体例子，搭配Link：

https://openai.com/five/

06: 07 对于模仿学习更详细的了解，大家可以参考论文：

https://arxiv.org/pdf/2106.12177.pdf

06: 43 有关模仿学习方法的细节和相关的研究资料，大家可以参考本节课提供的补充资料：

逆强化学习补充材料：
https://github.com/opendilab/PPOxFamily/tree/main/chapter4_reward/chapter4_supp_irl.pdf

行为克隆补充材料：
https://github.com/opendilab/PPOxFamily/tree/main/chapter4_reward/chapter4_supp_bc.pdf

09: 24 尝试设计好奇心机制的方法和结果可参考论文：

https://arxiv.org/pdf/1705.05363.pdf

11: 13 具体好奇心机制和内在奖励的定义，以及如何运用到强化学习方法中可参考论文：

https://arxiv.org/pdf/1705.05363.pdf

15: 02 想要了解其他提取特征的方法，大家可以参考：

https://arxiv.org/pdf/1808.04355.pdf

https://zhuanlan.zhihu.com/p/473676311

16: 17 具体对第二类设计内在奖励的经典方法 RND 的解释，大家可以参考：

https://arxiv.org/pdf/1810.12894.pdf

17: 50 详细随机蒸馏问题的解释和对比理解可参考论文：

https://arxiv.org/pdf/1810.12894.pdf

19: 58 想要了解如何设计出随机蒸馏问题以及它的新颖之处可参考：

https://zhuanlan.zhihu.com/p/485476646

22: 54 将 ICM 和 RND 结合到 PPO 中的完整示例，搭配 Link：

https://opendilab.github.io/PPOxFamily/

23: 38 Minigrid 的相关材料，详细解释以及教程，搭配 Link：

https://github.com/Farama-Foundation/Minigrid

https://di-engine docs.readthedocs.io/zh_CN/latest/13_envs/minigrid_zh.html

24: 34 Minigrid 的完整视频demo，大家都可以在他们的GitHub仓库中找到：

https://github.com/opendilab/PPOxFamily/issues/44

27: 52 对于 Pop-Art 更详细的解释和分析，大家可以参考论文：

https://arxiv.org/pdf/1602.07714.pdf

32: 45 想要更详细了解 Value Rescale 可参考论文：

https://arxiv.org/pdf/1805.11593.pdf

34: 58 如何实现Value Rescale的正向和逆向操作，以及如何运用到 PPO 算法中的代码完整示例搭配Link：

https://opendilab.github.io/PPOxFamily/

35: 11 有关于将 PPO 算法和一系列 reward 处理方法运用到 MetaDrive 实践中的材料，大家可以在以下链接寻找：

https://github.com/metadriverse/metadrive

35: 57 详细的 MetaDrive 中的奖励空间定义可参考：

https://di-engine-docs.readthedocs.io/zh_CN/latest/13_envs/metadrive_zh.html

36: 39 MetaDrive 的完整视频 demo，大家都可以在他们的 GitHub 仓库中找到：

https://github.com/opendilab/PPOxFamily/issues/44

Q8： 好奇心算法的实现是否有案例可以参考？

A8： 可以参考 DI-engine 中实现的好奇心算法 ICM：

- 代码链接 https://github.com/opendilab/DI-engine/blob/main/ding/reward_model/icm_reward_model.py
- 相关中文博客 https://zhuanlan.zhihu.com/p/474259159

Q9: 在 PPO 中是否需要 value 的方差小于 1？如果不是必须的话？在 value 方差大于 1 时，value*mean_std.std 会使 value 的方差进一步扩大，此时再使用基于 value 计算的 unnormalized_returns 来更新 mean_std，就会产生连锁反应，导致 mean_std 的 std 不断增大，最终导致 value 出现 inf 的问题。

A9：这应该与使用朴素的 value normalization 的缺陷有关。在朴素的 value normalization 中，我们通过将 value 除以其标准差来对其进行归一化。但是如果 value 的方差很大，会使归一化后的值在训练期间变得更加不稳定，可能会导致较差的效果。如果 value 方差较大，可以考虑使用其他归一化方法或者技巧，或者其他的技巧来控制 value 的大小和方差，以提高算法的稳定性。比如：

- 控制 running mean_std 的更新幅度。例如，使用一个衰减系数来控制 running mean_std 的更新幅度。可以使用一个小于1的衰减系数来加权历史 mean_std 值和当前 mean_std 值，使得当前 mean_std 值对更新的贡献较小。这样做可以减少 mean_std 的方差，并提高算法的稳定性。还可以使用其他的技巧来控制 mean_std 的大小和方差，例如使用一个可调节的系数来调整 mean_std 的大小，或者使用分位数归一化等方法来归一化 value。这些技巧可以帮助我们更好地利用 value 信息并提高算法的性能和稳定性。
- 进一步，升级成 Pop-Art 的实现形式来保证这个操作的平稳性。Pop-Art 是一种基于 running mean 和 running std 的归一化方法，它可以自适应地调整归一化的尺度和中心点，使得归一化后的值的方差和均值保持稳定。这样做可以使 running mean 和 running std 更加平稳，并且能够更好地适应 value 方差较大的情况。

Reference

[1] Awesome Exploration Methods in Reinforcement Learning, https://github.com/opendilab/awesome-exploration-rl#a-taxonomy-of-exploration-rl-methods, 2022

[2] Badia, Adrià Puigdomènech, et al. "Never give up: Learning directed exploration strategies." *arXiv preprint arXiv:2002.06038* (2020)。 https://arxiv.org/abs/2002.06038

[3] 强化学习探索家 | 集大成者 NGU （1）, https://zhuanlan.zhihu.com/p/551992517, 2022

[4] Pohlen, Tobias, et al. "Observe and look further: Achieving consistent performance on Atari." *arXiv preprint arXiv:1805.11593* (2018). https://arxiv.org/abs/1805.11593