PPO × Family 第二讲技术问题 QA

课程组整合梳理了自第二节课发布以来的相关技术问题答疑，希望能启发更多对于决策智能和强化学习的思考，也欢迎大家贡献问题或者积极参与讨论~

Q0: 有没有第二节课内容的大白话总结?

A0: 决策问题环境类型的多样性带来了不同类型的动作空间，而处理不同的动作空间需要不同的策略建模方式：

小节	技术要点	代表决策任务
离散动作空间	• 离散动作空间计算图 + logit • 构建概率分布及采样动作	火箭回收 视频游戏（Atari，Procgen）
连续动作空间	• 连续动作空间计算图/梯度流 • PPO 和 DDPG 在设计理念和数据属性两方面的对比	机器人控制（MuJoCo） 无人机姿态控制
多维离散空间	• multi-discrete 等四种网络结构的对比 • multi-discrete 和 discrete 在探索和利用角度的分析	交通信控 键盘操作的游戏
混合动作空间	• 各类动作空间离散化方法 • Hybrid PPO (HPPO) • HPPO + 自回归/分离预测器	导航控制 即时战略游戏（星际争霸2） MOBA 类游戏（DOTA2）

总之，对一个实际的决策问题环境选用适合的建模方式，转化为标准的 MDP 形式，是解决决策问题最关键的步骤。所谓，阵而后战，兵法之常；运用之妙，存乎一心。

Q1: 第二节课代码题的第一题，就是在给出的代码段里（即 `reparam_grad` 函数），根据重参数化方法对应得到的梯度公式，用 `numpy` 实现公式，并运行整个程序进行对比，这道题是这么理解的吗?

A1: 这道题就是对应梯度公式实现相应逻辑，可以参考代码段上面的 `naive_grad` ，这个函数就是实现了朴素方法的梯度公式。整体这道题的目的是希望通过这个例子来对比解释重参数化到底是如何减小梯度方差，通过直观的数据对比来说明这一点。

Q2: 第二节课的应用 demo 都挺有意思的，但是实际运行起来配环境时出现了不少bug，并且究竟运行成什么样才算效果好，作为初学者不太有概念，这方面课程官方能提供一些帮助吗？

A2: 课程组也注意到了这个问题，对于第二节课的题解和第三节课的实践题，我们会做以下改进：

- 对于环境安装，课程组会给出完整的安装指南 + 可以直接运行环境 demo 的 docker + 可以直接运行环境 demo 的在线 notebook（类似 colab 和 kaggle 上的 notebook）
- 对于训练过程，课程组会从第三节课作业起预先给出完整的训练日志和相应的操作录屏，帮助大家建立一个更直观的认识
- 对于代码和使用到的一些开源库，我们会安排一组系列小视频来讲解相关的代码（包括但不限于 gym 环境定义，DI-engine 代码详解， DI-treetensor 和 wandb 使用指南等等）

Q3: 请问一下，课程2中的辅助材料，也是需要学习的吗？这些材料和动作空间的关系是？

A3: 辅助材料是作为补充使用的，根据个人兴趣和时间安排决定即可。不过课程组建议，就算不深入看也应该简单读一下了解大概内容，以后如果遇到类似的问题可以借助补充材料拓展知识。对于第二节的补充材料，主要对应三部分的内容：

1. 重参数化部分详解连续动作空间输出形式的设计，并对比 PPO 和 SAC 在这方面的异同点。
2. DDPG 对比部分主要深入解释 PPO 中重要性采样（IS）的设计动机，虽然都是做连续动作控制但是两者优化思路不同，因此设计就大相径庭，进而实践中 DDPG 和 PPO 的超参数设置方式也有很大区别。
3. HyAR 部分则是给出了动作空间表征学习的一个例子，这也是目前复杂动作空间研究的主要方向。

Q4: 想问下课上的内容，为什么可以用 Behavior Cloning（BC）来测试网络架构利用数据能力的高低呢？课程里面说的这种专家数据的 BC 具体是怎么做的呢？是用专家数据拟合程度给 reward 么？还是说用 PPO 的网络直接做监督学习呢？

A4: 这里是使用了 PPO 应用于 multi-discrete action space 的两种神经网络模型架构，一种是标准的 discrete 形式，另一种是 multi-discrete。不同模型使用相同的专家数据去做监督学习，实验中发现 multi-discrete 形式的网络学习效果更好，说明这种网络结构设计更有优势。

Q5: 请问 DPPO (Distributed Proximal Policy Optimization) 在更新模型的时候，优势值一般要不要做归一化呢？例如有看到过类似下方的代码实现：

```
adv = (adv - adv.mean()) / (adv.std() + 1e-6)  # sometimes helpful
```

并且，如果要做归一化，应该分批次做吧，因为 DPPO 每次训练的数据来自于多个 worker ，应该是在各自 worker 内产生的数据进行归一化吧？不然就混乱了。

A5: 优势值归一化（adv norm）需要考虑 reward 的数值范围：

- 如果 reward 绝对值在 0-100 之内其实影响不大；
- 如果绝对值大于这个范围，且 reward 波动的确实很明显（注意要和稀疏 reward 区分），那适合用 adv norm。然后这种比较直接的 adv norm，应该 batch 越大统计量越准，所以在实现中，像 DPPO 的话，应该是训练的多卡之间 allreduce 同步 mean 和 std，然后再 norm。

Q6: 加 adv norm 对解决训练过程中出现 NaN 导致崩溃有帮助吗？

A6: 出现 NaN 有很多种原因，其中之一就是 adv norm；具体情况是，如果数据多样性很差，一个 batch 里的数据太相近太过相似，那么算出来的 std 就很接近于0，这样 norm 操作时一除就导致 NaN 。

Q7: 在 DPPO 中添加 adv norm，有两种实现方式：

- **类似课程组之前将的 allreduce [可参考博客] 出各个 worker 所有的 adv ，然后求 mean 和 std ，**
- **每个 batch 的数据中，是由不同的 worker 贡献的，单独在各自 worker 贡献的数据内求 mean 和 std ，然后对各自 worker 贡献的数据进行归一化，**

这两个哪个好呢？我自己做了下实验，发现这两种没啥差别。

A7: mean 和 std 计算用的样本肯定是越多越准，但因为 RL 本身数据分布就一直在变，所以可能有些场景里对最终性能影响不大，就跟你的实验结果一样。你要想真正深究这个问题，应该要去可视化这两种设定下算出来的 mean 和 std 的变化情况，再分析这个变化对于智能体性能的影响，并在不同类型的环境上做对比看能不能找到普适结论；决策问题（环境）之间的差异性太大了，所以经验性结论经常变化，但是分析方法和手段掌握了之后，具体问题具体分析就好，没有什么玄学。

Q8: 想问下 PPO 的连续动作的方差，一般是用神经网络训练输出，还是给固定值（或者从固定值开始线性衰减）？我自己试验了一下，发现方差给固定值居然要比用神经网络学出方差要好。对此我 google 了一下，发现 OpenAI 的开源库中使用 PPO 算法时，居然说他们使用了固定的连续动作的方差 sigma，而不是学出来的，链接如下：
https://spinningup.openai.com/en/latest/spinningup/rl_intro.html

A8: PPO 连续动作的 sigma，其实在不同版本的实现里一共有三种，根据不同情况使用：

- fixed：固定 sigma，常用于一些特殊控制任务，如果对环境的 sigma 的设置有足够的先验知识可以这样做
- independent：即为一个可优化的网络参数，但是和 state 无关，是一个独立参数。这是一般 PPO 常用的情形
- state conditioned：由 state 输入通过一定的网络层生成，这种情况在 SAC 中常用，PPO中较少见。不过有 paper 在 MuJoCo 环境上做过对比实验，至少在 MuJoCo 这样的控制环境上差别不大

三种类型的代码对比可以参考这里的代码: https://github.com/opendilab/DI-engine/blob/main/ding/model/common/head.py#L965

中文版的注释详解可以看这个: https://opendilab.github.io/PPOxFamily/continuous_zh.html

Q9： 第二讲课程中提到 CityFlow 环境的奖励空间是：

“每辆在进入路口时需要等待红灯的车都会产生负的奖励，同时等待时间越长，负奖励越大；每辆行驶出路口的车则会产生另一种正的奖励 。两部分相结合就促使智能体学习到最大化路口吞吐量的策略。 ”.

想问一下，这个奖励函数的具体形式是怎么样？在代码中没有找到这个多维离散动作空间的训练代码。

A9： 具体形式可以参考课程第二讲文字稿中的相关信息 https://mp.weixin.qq.com/s/OsygT69_jD-4RpnobWe19g，具体代码的话可以先参考这个仓库 https://github.com/opendilab/DI-smartcross/blob/main/smartcross/envs/cityflow_env.py

Q10： 对 Muti-discrete 动作空间的代码实现有一些疑惑，是否有相关补充资料？

A10： 这部分的补充材料，会有模型修改、动作空间修改、学习函数修改三个方面：

- 模型修改 https://github.com/opendilab/DI-engine/blob/main/ding/model/template/vac.py#L131-L151
- 动作空间修改 https://github.com/opendilab/DI-smartcross/blob/main/smartcross/envs/cityflow_env.py#L15-L30
- 学习函数 https://github.com/opendilab/DI-engine/blob/main/dizoo/common/policy/md_ppo.py#L80-L87

另外可以参考这里的例子，Multi-discrete 的网络结构+如何采样动作：
https://github.com/opendilab/PPOxFamily/blob/gh-pages/chapter2_action/discrete_tutorial_zh.py#L58

Q11： 重参数梯度中提到，将重参数化后的形式代入，我们原先的随机目标函数可以变为如下所示：

$$L(\theta)=\mathbb{E}_{q_\theta(\mathbf{z})}(l(\theta,\mathbf{z}))=\int l(\theta,\mathbf{z})q_\theta(\mathbf{z})\mathrm{d}\mathbf{z}$$
$$=\int l(\theta,r(\theta,\boldsymbol{\epsilon}))q(\boldsymbol{\epsilon})\mathrm{d}\boldsymbol{\epsilon}=\mathbb{E}_{q(\boldsymbol{\epsilon})}(l(\theta,r(\theta,\boldsymbol{\epsilon})))$$

这里的推导有更详细的说明吗？为什么 $q_\theta(\mathbf{z})\mathrm{d}\mathbf{z}$ 可以替换为 $q(\boldsymbol{\epsilon})\mathrm{d}\boldsymbol{\epsilon}$？

A11： 关于这个地方的推导，可以参考我们最新更新后的补充材料，添加了一些补充说明内容，主要用到一些概率论相关的知识：
https://github.com/opendilab/PPOxFamily/blob/main/chapter2_action/chapter2_supp_reparameterization.pdf

> **B 重参数化证明细节说明**
>
> 为了证明改变随机变量前后，以下公式仍然成立，需要使用以下的一些概率论与测度的知识：
>
> $$L(\theta)=\mathbb{E}_{q_\theta(\mathbf{z})}(l(\theta,\mathbf{z}))=\int l(\theta,\mathbf{z})q_\theta(\mathbf{z})\mathrm{d}\mathbf{z}$$
> $$=\int l(\theta,r(\theta,\boldsymbol{\epsilon}))q(\boldsymbol{\epsilon})\mathrm{d}\boldsymbol{\epsilon}=\mathbb{E}_{q(\boldsymbol{\epsilon})}(l(\theta,r(\theta,\boldsymbol{\epsilon})))$$
>
> 概率空间 $(\Omega,\mathcal{F},\mathcal{P})$ [1] 是一个测度为 1 的空间，其中样本空间 Ω，事件集合 $\mathcal{F}$，和测度函数 $\mathcal{P}$，有：
>
> $$\mathcal{P}(\Omega)=1$$
>
> 对于任意的事件 A，或是这些事件的并集，$A\in\mathcal{F}$，有 $\mathcal{P}(A)\in[0,1]$。

> 对于随机变量 x 和 y ，如果有 $y=f(x)$ 始终成立，那么对于所有 $x=c$ 的事件 $A_{x=c}$ ，显然与 $y=f(x)=f(c)$ 是同一类事件 $A_{y=f(c)}$ ，所以其概率测度 [2] 相同。对于这些事件的并集也相同。
>
> 那么对于连续的随机变量的任意此类事件的概率测度，则有如下公式成立：
>
> $$\mathcal{P}(A_{x=c})=\int_{A_{x=c}} p_X(x)\mathrm{d}x=\int dp=\int_{A_{y=f(c)}} p_Y(y)\mathrm{d}y=\mathcal{P}(A_{y=f(c)})$$
>
> 其中 $p_X(x)$ 和 $p_Y(y)$ 是对应的连续变量的概率密度， $\mathrm{d}p$ 为对应的概率质量。
>
> 因为上式对于任意基本事件范围的求和或积分都成立，那么将等式两边添加任意相同的权重系数 $l(y)$ 之后，其求和或积分也成立，所以有：
>
> $$\int l(y)p_Y(y)\mathrm{d}y=\int l(f(x))p_X(x)\mathrm{d}x$$
>
> 于是对应着所需证明的等式成立。

Q12: 如果想用 PPO 算法处理 Graph 类型的观察空间，例如把 Encoder 的神经网络替换成 GNN。但是在 update 批处理 obs 时，由于 obs 存储在一个 list 中，没法直接转为 tensor。请问有什么比较好的处理方式吗？

A12:

这种就是 RL 里复杂环境经常出现的结构化数据，可以将这个 list 转化为 dict（list 可以看作是整数索引的 dict），然后使用我们提供的 treetensor 工具来建模。具体的链接可以参考这里

- https://github.com/opendilab/DI-engine#general-data-container-treetensor
- https://github.com/opendilab/DI-treetensor

Q13: 请问在 PPO 算法中，当在离散动作训练过程中出现 NaN 时，有哪些有效的处理方法？是否应该对概率进行范围裁剪（clip）？一般情况下，该范围应如何确定？

A13:

通常情况下，在离散动作训练过程中很少会出现 NaN 的情况，因为很少会有某个离散动作对应的对数概率接近 NaN 或无穷大（Inf）。如果确实出现了 NaN 的情况，可以通过观察离散概率分布的变化过程来进行分析。在 TensorBoard 中使用直方图（histogram）功能可以查看离散概率分布在训练过程中的具体情况。

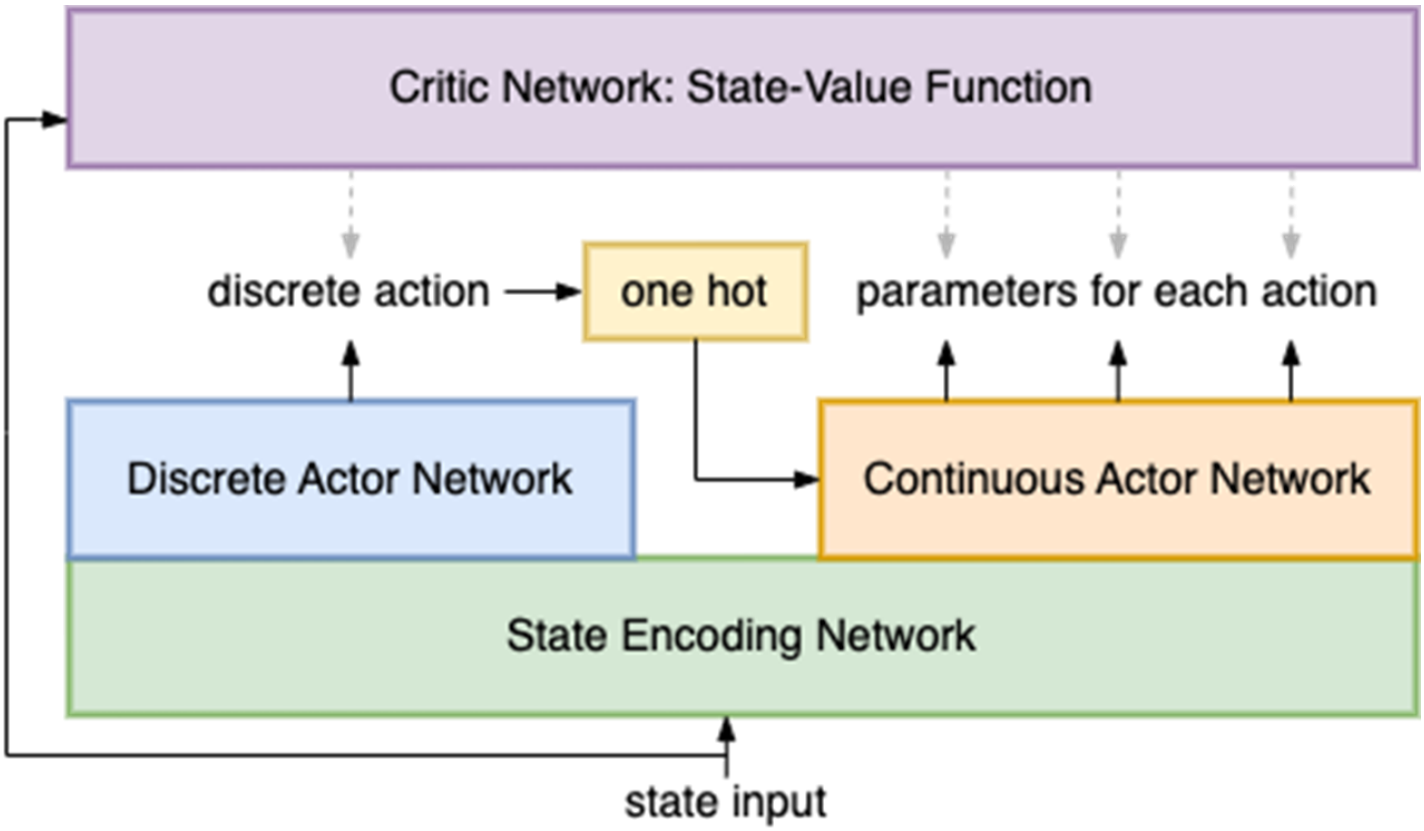
至于是否应该对概率进行范围裁剪（clip），这取决于具体的情况。范围裁剪可以确保概率值在一个合理的范围内，并防止出现过大或过小的概率值。裁剪范围的确定应该根据实际问题和实验结果来决定。一种常见的做法是将概率裁剪到较小的范围，例如 [ε, 1-ε]，其中 ε 是一个较小的正数。然而，需要注意的是，裁剪范围不应设置得过窄，以允许一定程度的不确定性和探索。

Q14: HPPO 中，如果 continuous actor network 输出连续动作并 detach，和 state 拼接起来作为 discrete actor network 的输入，这样与原始的 HPPO 会有效果的差异吗？

A14:

这样的处理方式并不能完全解决问题，因为实际上只是在前向传播时提供了连续动作的信息，但在反向传播过程中，应该返回给连续动作网络的那部分梯度并没有被传递回去。是否进行 detach 操作仍然要根据实际问题来决定。如果离散动作和连续动作之间具有很强的相关性，那么最好还是让梯度回传到连续动作网络中。然而，如果离散动作和连续动作之间相关性较弱，那么断开计算图（detach）也是可行的。对于理解原理而言，这种方法可以作为一个对比实验，但在实际问题中，需要根据实验结果来决定如何使用该方法。

Q15: 在课程第二讲中提到，HPPO（Hybrid Proximal Policy Optimization）的一种变体将连续动作作为离散网络的输入，这可能会导致梯度出现问题。那么，如果将离散动作作为连续动作网络的输入，是否就不存在这个问题了呢？



A15:

这个问题不是简单的离散输入连续或者连续输入的问题，因为离散动作和连续动作之间存在特定的依赖关系，比如假设 3 个离散动作和 3 个连续参数，且一个离散动作对应一个连续参数（例如控制方向盘离散动作+方向盘角度连续动作），从某个离散动作的角度来讲，HPPO 这种变体会将所有的连续参

数都作为离散部分的输入，那么和这个离散动作不相关的其他连续参数，都会带来干扰和影响。真正需要做的是，设计一种方便实现的机制，使得只有与离散动作最相关的连续参数作为其输入，而不相关的连续参数不参与其中。建议参考课程第二讲作业，从梯度角度给出了这个问题的分析思路。

Q16: 处理离散动作空间的 PPO 为什么使用 Softmax 而不是 Gumbel Softmax?

A16:

在 PPO 中，采样出来的动作并不需要用于相关损失函数的计算，因此不需要重参数化来确保这部分梯度能够回传。PPO 中的采样动作仅用于与环境进行交互和生成输出动作。在这种情况下，使用简洁明了的 softmax 函数就足够了。

Reference

[1] Xu M, Quiroz M, Kohn R, et al. Variance reduction properties of the reparameterization trick[C]//The 22nd International Conference on Artificial Intelligence and Statistics. PMLR, 2019: 2711-2720.

[2] Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor[C]//International conference on machine learning. PMLR, 2018: 1861-1870.