第七讲 Q&A

Q0:有没有第七节课内容的大白话总结?

A0:

小节	要点
权衡偏差与方差的GAE	• 价值函数经典学习方式 • GAE
稳扎稳打之Recompute	• 无需梯度计算，节省时间和显存 • 无序额外维护时间顺序关系 • 在 Epoch 较大时尤其有用

巧夺天工之Entropy	• 策略早熟
	• PPO 中控制策略探索能力的两个关键点
	  • 选择动作时的 Temperature
	  • 计算损失函数时的 Entropy Bonus

免死金牌之Grad Clip	• PPO 训练波动的原因 • Grad Clip 具体实现
笨鸟先飞之Init	• 神经网络初始化设计的五大因素 • 为什么不使用全零或固定方差的初始化 • 为什么要初始化考虑非线性函数的类型 • 正交初始化 • 策略输出层的特殊初始化
亡羊补牢之Extra Clip	• Dual clip 技巧 • Value clip 技巧
合理利用随机	• 环境的随机种子 • 工具库导致的随机 • 收集数据和评估测试对随机性要求区别

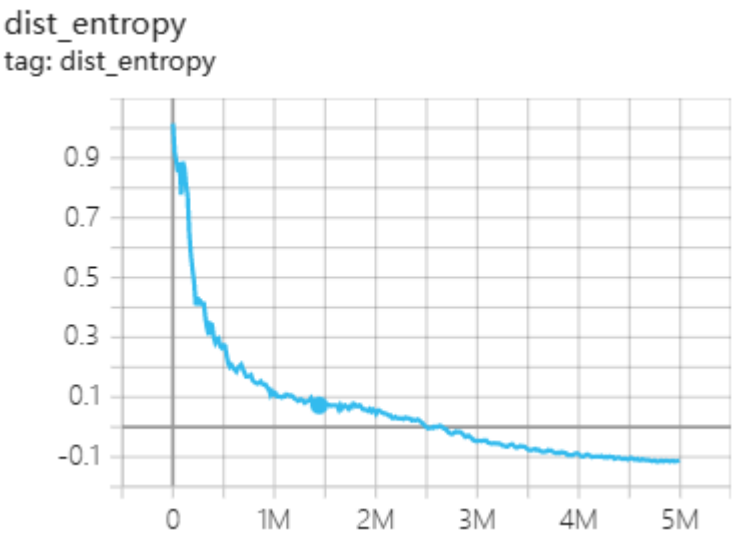
Q1: 公众号文章《Off-Policy 版本的 PPO 算法，训练效率可不会 off》中提到“当采样数据的使用次数小于 1 时，智能体的表现提升的速率最快。”，那么在采样数据非常困难时一般采用什么解决方法？通过类似 Off-policy DQN 的方法对较少的数据进行多次重复训练吗？当采样数据的训练次数小于 1 时，PPO 效果最好，此时 PPO 是不是可以近似于没有重要性采样的 PG 算法？

A1:

A1.1 在采样数据非常困难的情况下，一种常见的解决方法是尝试使用 PPO 的 off-policy 版本。但需要注意的是，这种 off-policy 版本并没有得到理论上的充分支持，它只是实践中的一种变体。在使用 off-policy 版本时，需要谨慎控制数据的 staleness（陈旧程度）和 reuse（重用），以确保训练的稳定性和有效性。

A1.2 此外，当采样数据的训练次数小于 1 时，PPO 并不能近似于没有重要性采样的 PG（Policy Gradient）算法。PPO 和 PG 的设计目的和训练数据分布存在很大差异。虽然从最终的损失函数的角度来看可能存在相似之处，但在原理上仍然需要分开理解，它们在概念和训练方法上存在明显差异。

Q2: 这样的熵收敛曲线，是不是就属于策略早熟呢？



A2:

可能表明策略早熟的熵收敛曲线特征：

- 早期快速下降：在训练的初期，策略熵迅速下降，并很快趋于稳定。这可能表明智能体过早地收敛到一个确定性策略，缺乏进一步的探索。
- 平稳收敛：策略熵在训练的后期达到一个相对稳定的值，并保持在该值附近。这可能表明智能体没有进一步改进策略，陷入了一个局部最优解。

- 低熵值：策略熵的最终稳定值较低，接近于零。这表示智能体在选择行动时的不确定性极小，基本上变成了确定性策略，没有足够的探索行为。

如果以上特征同时出现，或者熵在训练的早期迅速下降并在后期保持较低的稳定值，可能就是存在策略早熟的问题。所以如图的熵收敛曲线大概率是策略早熟，一般来说最好的 entropy 曲线是，先缓慢下降（充分探索）再较快速度下降（找到某种较优的解）。可以根据你的任务具体的 action space，结合这个经验来控制 entropy weight。

Q3:理论题第一题中，GAE-λ 的数学表达式中，λ 和 γ 分别起到什么作用？GAE-λ 和 TD-λ 之间有什么区别和联系？

A3:

通过表达式，我们可以发现，λ 的大小主要负责调节各个 N-step 优势函数的数值占 GAE 数值的权重，当 λ 接近 0 时，只有当前 step 的优势函数起作用，而当 λ 接近 1 时，多个 step 的优势函数都会对 GAE 产生影响。较大的 λ 值可能会导致更平滑的优势估计，而较小的 λ 值则可能更加关注当前步骤的优势。

而 γ 的大小负责每个优势函数计算中，各个时刻的奖励沿时间的折扣的大小，较大的 γ 值意味着更重视未来的奖励，即更远的时间步长上的奖励仍然具有较高的权重。而较小的 γ 值会导致更快地减小奖励的权重，更加重视即时奖励。

因为 GAE-λ 的每一项都是 TD-λ 的当前价值函数 target 减去当前状态的价值函数，因此我们可以将 GAE-λ 视为 TD-λ 形式的优势函数的表达式，然后通过系数 λ 的大小来控制各个 step 长度的优势函数占整体 GAE 数值的权重。

Q4:理论题第四题中，蒙特卡洛估计与时序差分法估计在理论上的主要区别是什么，在算法实现中，会有区别吗？

A4:

在理论方法上，MC 和 TD 两种方法对价值函数的估计的主要区别在于，前者通过累次叠加整条轨迹的奖励，从而获得对价值函数的无偏估计，而后者通过叠加有限长度的轨迹奖励，并使用价值函数进行自举归纳，作为对价值函数的估计，在训练过程中，该估计和真实的价值函数存在偏差，但偏差会随着训练的收敛而逐渐降低。

在算法实现中，计算 MC 法估计的价值函数的损失，需要将完整的轨迹奖励信息放入计算函数。而对于 TD 法，则除了需要来自环境反馈的奖励信息之外，还需要额外计算 N 步之后的状态下的状态价值。

Q5:理论题第五题中，神经网络参数正交初始化的核心数学原理是什么？在 pytorch 中，应该如何实现它们？假如神经网络的参数不是一个方阵，参数正交初始化应该怎么做？

A5:

神经网络参数正交初始化的核心数学原理是正交矩阵之间的乘法运算依然是一个正交矩阵，从而保持了其矩阵范数的有界，这对稳定神经网络的初始阶段的训练是有意义。在 pytorch 库中，可以使用“torch.nn.init.orthogonal_”函数来实现神经网络参数的初始化。

当神经网络的参数不是一个方阵，比如假定神经网络的参数矩阵 W 是一个 $D_1 \times D_2$ 的矩阵，其中 $D_2 > D_1$ ，假如它被奇异值分解成以下形式的矩阵：

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1D_2} \\ w_{21} & w_{22} & \cdots & w_{2D_2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{D_1 1} & w_{D_1 2} & \cdots & w_{D_1 D_2} \end{bmatrix} = U \begin{bmatrix} \lambda_{11} & & & \\ & \lambda_{22} & & \\ & & \ddots & \\ & & & \lambda_{D_1 D_1} \\ & & & \vdots \\ & & & 0_{D_2 D_1} \end{bmatrix} V^T = U \Lambda V^T$$

其中， U 和 V 分别是 $D_1 \times D_1$ ， $D_2 \times D_2$ 的正交矩阵，经过正交初始化参数矩阵 W ，使得其特征值都为 1，可以让 $W^N = W \times W \times W \cdots \times W$ 的矩阵范数依然保持在一定范围内，从而有利于神经网络训练。