DDPG 算法

本章讲 DDPG 算法和 TD3 算法。DDPG 算法可以看作是 DQN 算法在连续动作空间上的一种扩展，但结构上仍然更像是 Actor-Critic 算法。TD3 算法则是对 DDPG 算法的一种改进，主要是为了克服 DDPG 算法在实际应用中存在的一些缺点，包括过估计偏差、训练不稳定等问题。

DPG 算法

在经典策略梯度算法中，我们学习的是一个随机性策略（stochastic policy），即通过学习一个概率分布 $\pi_\theta(a|s)$ 来选择动作 a。对于连续动作空间，我们通常会选择高斯分布来表示这个概率分布，但是这种策略往往计算成本较高，且对动作采样的方差较大，导致训练过程不稳定。

在某些情况下，使用确定性策略（deterministic policy）可能会更加高效。确定性策略直接将状态映射到一个具体的动作，而不是一个概率分布，如式 (1) 所示。

$$a = \mu_\theta(s) \tag{1}$$

对应的策略梯度表达式如式 (2) 所示。

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_a Q\left(s_t, a\right)|_{a=\mu_\theta(s_t)} \nabla_\theta \mu_\theta\left(s_t\right)\right] \tag{2}$$

这就是确定性策略梯度（Deterministic Policy Gradient，DPG）算法的核心，其中 ρ^β 是策略的初始分布。

DDPG 算法

基于 DPG 方法，DDPG 算法在引入神经网络近似的同时，增加了一些要素，如式 (3) 所示。

$$\text{DDPG} = \text{DPG} + \text{target network} + \text{experience replay} + \text{OU noise} \tag{3}$$

其中目标网络和经验回放在 DQN 算法中已经介绍过了，这里我们主要讲讲 OU 噪声，引入噪声的目的是为了增加策略的探索性，从而提高算法的性能和稳定性，如式 (4) 所示。

$$dx_t = \theta\left(\mu - x_t\right)dt + \sigma dW_t \tag{4}$$

其中 x_t 是 OU 过程在时间 t 的值，即当前的噪声值，μ 是回归到的均值，θ 是 OU 过程的回归速率，σ 是 OU 过程的扰动项，dW_t 是布朗运动（Brownian motion）或者维纳过程（Wiener process），是一个随机项，表示随机高斯噪声的微小变化。

代入到 DDPG 算法中，最终的动作选择如式 (5) 所示。

$$a_t = \mu_\theta(s_t) + x_t \tag{5}$$

OU 噪声是一种具有回归特性的随机过程，其与高斯噪声相比的优点在于：

- **探索性**：OU 噪声具有持续的、自相关的特性。相比于独立的高斯噪声，OU 噪声更加平滑，并且在训练过程中更加稳定。这种平滑特性使得OU噪声有助于探索更广泛的动作空间，并且更容易找到更好的策略。
- **控制幅度**：OU 噪声可以通过调整其参数来控制噪声的幅度。在 DDPG 算法中，可以通过调整 OU 噪声的方差来控制噪声的大小，从而平衡探索性和利用性。较大的方差会增加探索性，而较小的方差会增加利用性。
- **稳定性**：OU 噪声的回归特性使得噪声在训练过程中具有一定的稳定性。相比于纯粹的随机噪声，在 DDPG 算法中使用OU 噪声可以更好地保持动作的连续性，避免剧烈的抖动，从而使得训练过程更加平滑和稳定。
- **可控性**：由于 OU 噪声具有回归特性，它在训练过程中逐渐回归到均值，因此可以控制策略的探索性逐渐减小。这种可控性使得在训练的早期增加探索性，然后逐渐减小探索性，有助于更有效地进行训练。

最后，DDPG 算法的整体流程如图 1 所示，**完整的代码实现可参考实战部分的内容。**

DDPG 算法

初始化 critic 网络 $Q\left(s, a \mid \theta^{Q}\right)$ 和 actor 网络 $\mu(s|\theta^{\mu})$ 的参数 θ^{Q} 和 θ^{μ}
初始化对应的目标网络参数，即 $\theta^{Q'} \leftarrow \theta^{Q}, \theta^{\mu'} \leftarrow \theta^{\mu}$
初始化经验回放 D
for 回合数 $= 1, M$ **do**
  交互采样：
  选择动作 $a_t = \mu\left(s_t \mid \theta^{\mu}\right) + \mathcal{N}_t$，$\mathcal{N}_t$ 为探索噪声
  环境根据 a_t 反馈奖励 s_t 和下一个状态 s_{t+1}
  存储样本 (s_t, a_t, r_t, s_{t+1}) 到经验回放 D 中
  更新环境状态 $s_{t+1} \leftarrow s_t$
  策略更新：
  从 D 中取出一个随机批量的 (s_i, a_i, r_i, s_{i+1})
  求得 $y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right)$
  更新 critic 参数，其损失为：$L = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i \mid \theta^{Q}\right)\right)^2$
  更新 actor 参数：$\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_i \nabla_a Q\left(s, a \mid \theta^{Q}\right)\big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu\left(s \mid \theta^{\mu}\right)\Big|_{s_i}$
  软更新目标网络：$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$，$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$
end for

图 1: DDPG 算法流程

注意，跟 DQN 算法中使用的硬更新不同，DDPG 算法中目标网络的更新一般使用软更新（Soft Update）的方式，这样可以使得目标网络的参数更加平滑地变化，从而提高训练的稳定性和收敛性。

TD3 算法

DDPG 算法虽然在连续动作空间中表现不错，但有个很大的缺点，那就是它容易出现 Q 值的过估计问题（overestimation），这会导致策略学习的不稳定，甚至发散。为了解决这个问题，TD3 算法（Twin Delayed DDPG）提出了三种关键的改进技巧，一是双 Q 网络，二是延迟更新，三是目标策略平滑。

双 Q 网络

在 DDPG 算法中，只有一个 Critic 网络来估计动作的价值，这可能会导致 Q 值的过估计问题。为了解决这个问题，TD3 算法引入了双 Q 网络的概念，即使用两个独立的 Critic 网络来估计动作的价值。在计算目标值 y 时，取两个 Q 值中的较小值作为目标值，从而减少过估计的风险，如式 (6) 所示。

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i'(s', \mu_{\phi'}(s'))} \tag{6}$$

对应的损失函数如式 (7) 所示。

$$L(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim D}\left[\left(Q_{\theta_i}(s,a) - y\right)^2\right] \quad i = 1, 2 \tag{7}$$

延迟更新

延迟更新（Delayed Policy Updates）是指在 TD3 算法中，Actor 网络的更新频率要低于 Critic 网络的更新频率。具体来说，Actor 网络每隔一定数量的时间步才会更新一次，而 Critic 网络则在每个时间步都进行更新。 这样做的目的是为了减少策略更新的频率，从而提高训练的稳定性。

举个例子，Critic 就好比领导，Actor 则好比员工，领导不断给员工下达目标，员工不断地去完成目标，如果领导的决策经常失误，那么员工就很容易像无头苍蝇一样不知道该完成哪些目标。

因此，一个好的解决方式就是让领导学得比员工更快，领导学得更快了之后下达目标的失误率就会更低，这样员工就能够更好地完成目标了，从而提高整个团队的效率。在实践中，Actor 的更新频率一般要比 Critic 的更新频率低一个数量级，例如 Critic 每更新 10 次，Actor 只更新 1 次。

目标策略平滑

目标策略平滑（Target Policy Smoothing）是指在计算目标值 y 时，给目标策略添加一些噪声，从而使得目标策略更加平滑，减少过估计的风险，也叫噪声正则（noise regularization）。

在延迟更新中，我们的主要思想是通过提高 Critic 的更新频率来减少值函数的估计误差，也就是“降低领导决策的失误率”。但是这样其实是治标不治本的做法，因为它只是让 Critic 带来的误差不要过分地影响到了 Actor，而没有考虑改进 Critic 本身的稳定性。

因此，我们也可以给 Critic 引入一个噪声提高其抗干扰性，这样一来就可以在一定程度上提高 Critic 的稳定性，从而进一步提高算法的稳定性和收敛性。注意，这里的噪声是在 Critic 网络上引入的，而不是在输出动作上引入的，因此它跟 DDPG 算法中的噪声是不一样的。具体来说，我们可以在计算 TD 误差的时候，给目标值 y 加上一个噪声，并且为了让噪声不至于过大，还增加了一个裁剪（clip），如式 (8) 所示。

$$y = r + \gamma Q_{\theta'}\left(s', \pi_{\phi'}\left(s'\right) + \epsilon\right) \epsilon \sim \operatorname{clip}(N(0, \sigma), -c, c) \tag{8}$$

其中 $N(0, \sigma)$ 表示均值为 0，方差为 σ 的高斯噪声，ϵ 表示噪声，clip 表示裁剪函数，即将噪声裁剪到 $[-c, c]$ 的范围内，c 是一个超参数，用于控制噪声的大小。

思考

DDPG 算法是 off-policy 算法吗？为什么？

跟 DQN 一样，DDPG 算法，主要结合了经验回放、目标网络和确定性策略，是典型的 off-policy 算法。

软更新相比于硬更新的好处是什么？为什么不是所有的算法都用软更新？

好处：**平滑目标更新**：软更新通过逐渐调整目标网络的参数，使其向主网络的参数靠近，而不是直接复制主网络的参数。这样做可以降低目标的变化幅度，减少了训练中的不稳定性；**降低方差**；**避免振荡**：软更新可以减少目标网络和主网络之间的振荡，这有助于更稳定地收敛到良好的策略。缺点：**速度**：软更新会使目标网络变得更加缓慢；**探索和稳定性权衡**：一些算法可能更倾向于使用硬更新，因为它们可能需要更频繁地探索新的策略，而不依赖于过去的经验。硬更新允许在每次更新时完全用新策略替代旧策略；**算法需求**：某些算法可能对硬更新更敏感，而且硬更新可能是这些算法的关键组成部分。综上，软更新和硬更新都有其用途，选择哪种方式取决于具体的问题和算法要求。

相比于 DDPG 算法，TD3 算法做了哪些改进？请简要归纳。

双Q网络： TD3 使用了两个独立的 Q 网络，分别用于估计动作的价值。这两个Q网络有不同的参数，这有助于减少估计误差，并提高了训练的稳定性；**目标策略噪声**：与 DDPG 不同，TD3 将噪声添加到目标策略，而不是主策略。这有助于减小动作值的过估计误差；**目标策略平滑化**：TD3 使用目标策略平滑化技术，通过对目标策略的参数进行软更新来减小目标策略的变化幅度。这有助于提高稳定性和训练的收敛性。**延迟策略更新**：TD3 引入了延迟策

略更新，意味着每隔一定数量的时间步才更新主策略网络。这可以减小策略更新的频率，有助于减少过度优化的风险，提高稳定性。

TD3 算法中 Critic 的更新频率一般要比 Actor 是更快还是更慢？为什么？

答：Critic 网络的更新频率要比 Actor 网络更快，即延迟策略更新。延迟策略更新的目的是减小策略更新的频率，以避免过度优化和提高训练的稳定性。因为 Critic 网络的更新频率更高，它可以更快地适应环境的变化，提供更准确的动作价值估计，从而帮助 Actor 网络生成更好的策略。