Actor-Critic算法

结合了DQN和策略梯度的想法，使用DQN学习价值函数，然后指导策略网络学习，最终的输出还是策略。

在策略梯度算法中，可以把梯度写成一个一般的形式：

$$g = \mathbb{E}\left[\sum_{t=0}^{T} \psi_t \nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

其中ψ_t有多种形式：

1. $\sum_{t'=0}^{T} \gamma^{t'} r_{t'}$：轨迹的总回报；
2. $\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$：动作$a_t$之后的回报；
3. $\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} - b(s_t)$：基准线版本的改进；
4. $Q^{\pi_\theta}(s_t, a_t)$：动作价值函数；
5. $A^{\pi_\theta}(s_t, a_t)$：优势函数；
6. $r_t + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$：时序差分残差。

对于形式4，可以考虑DQN学习价值函数的方法，使用策略网络+价值网络共同指导策略的学习。

Actor 的更新采用策略梯度的原则，那 Critic 如何更新呢？我们将 Critic 价值网络表示为V_ω，参数为ω。于是，我们可以采取时序差分残差的学习方式，对于单个数据定义如下价值函数的损失函数：

$$\mathcal{L}(\omega) = \frac{1}{2}(r + \gamma V_\omega(s_{t+1}) - V_\omega(s_t))^2$$

与 DQN 中一样，我们采取类似于目标网络的方法，将上式中$r + \gamma V_\omega(s_{t+1})$作为时序差分目标，不会产生梯度来更新价值函数。因此，价值函数的梯度为：

$$\nabla_\omega \mathcal{L}(\omega) = -(r + \gamma V_\omega(s_{t+1}) - V_\omega(s_t)) \nabla_\omega V_\omega(s_t)$$

然后使用梯度下降方法来更新 Critic 价值网络参数即可。