1. 基本原理

SAC（Soft Actor Critic）算法是一种基于最大熵的无模型的深度强化学习算法，适合于真实世界的机器人学习技能，它解决了离散动作空间和连续性动作空间的强化学习问题。SAC 算法在最大化未来积累奖励的基础上引入了最大熵的概念，加入熵的目的是增强鲁棒性和智能体的探索能力，该算法的目的是使未来积累奖励值和熵最大化，使得策略尽可能随机，即每个动作输出的概率尽可能的分散，而不是集中在一个动作上。

SAC 算法的目标函数表达式如下：

$$J = \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho^{\pi}}\left[r(s_t,a_t) + \alpha H\left(\pi(\cdot|s_t)\right)\right]$$

其中 T 表示智能体与环境互动的总时间步数，ρ^{π}表示在策略π下的分布，$H(\cdot)$代表熵值，α代表超参数，它的目的是控制最优策略的随机程度和权衡熵相对于奖励的重要性

2. 公式推导

SAC 是一种基于最大化熵理论的算法，由于目标函数中加入熵值，使得该算法的探索能力和鲁棒性得到了很大的提升，尽可能在奖励值和熵值（即策略的随机性）之间取得最大化平衡。Agent 因选择动作的随机性而获得更高的奖励值，以使它不要过早收敛到某个次优确定性策略，即局部最优解。熵值越大，对环境的探索就越多，避免了策略收敛至局部最优，从而可以加快后续的学习速度

因此，最优策略的 SAC 公式定义为：

$$\pi^{*} = arg\max_{\pi} E_{s_t,a_t\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^{t} r(s_t,a_t) + \alpha H\left(\pi(\cdot|s_t)\right)\right]$$

$$H\left(\pi(\cdot|s_t)\right) = E[-ln\,\pi(\cdot|s_t)]$$

其中π用来更新已找到最大总奖励的策略；α是熵正则化系数，用来控制熵的重要程序；$H\big(\pi(\cdot|s_t)\big)$代表熵值，熵值越大，智能体对环境的探索度越大

SAC 的 Q 值可以用基于熵值改进的贝尔曼方差来计算，状态-动作价值函数的定义为：

$$Q(s_t, a_t) = E_{s_{t+1}\sim D}[r(s_t, a_t) + \gamma V^{\pi}(s_{t+1})]$$

其中，从经验回放池 D 中采样获得，状态价值函数定义为：

$$V(s_t) = E_{a_t\sim\pi}[Q(s_t, a_t) - \alpha ln\, \pi(\cdot|s_t)] = E_{a_t\sim\pi}[Q(s_t, a_t) + H(\pi(\cdot|s_t))]$$

它表示在某个状态下预期得到的奖励

SAC 中包含 5 个神经网络：策略网络$\pi_{\phi}(a_t|s_t)$，软状态价值网络$V_{\psi}(s_t)$，目标状态价值网络$V_{\overline{\psi}}(s_t)$，以及 2 个软 Q 网络$Q_{\theta_{1,2}}(s_t, a_t)$，它们分别由$\phi, \psi, \overline{\psi}, \theta_1, \theta_2$参数化。为了分别找到最优策略，将随机梯度下降法应用于它们的目标函数中。对软状态价值网络的更新为：

$$J_V(\psi) = E_{s_t\sim D}\left[\frac{1}{2}\left(V_{\psi}(s_t) - E_{a_t\sim\pi_{\phi}}\left[\min_{i=1,2} Q_{\theta_i}(s_t, a_t) - \alpha ln\, \pi_{\phi}(a_t|s_t)\right]\right)^2\right]$$

此外，还采用了类似于双 Q 网络的形式，软 Q 值的最小值取两个由θ_1和θ_2参数化的 Q 值函数，这样有助于避免过高估计不恰当的 Q 值，以提高训练速度。软 Q 值函数通过最小化贝尔曼误差来更新：

$$J_Q(\theta) = E_{(s_t, a_t)\sim D}\left[\frac{1}{2}\big(Q_{\theta_{1,2}}(s_t, a_t) - \big[r(s_t, a_t) + V_{\overline{\psi}}(s_{t+1})\big]\big)^2\right]$$

策略网络通过最小化 KL 散度来更新：

$$J_{\pi}\big(\phi\big) = E_{a_t\sim\pi,\, s_t\sim D}\left[ln\, \pi_{\phi}(s_t, a_t) - \min_{i=1,2} Q_{\theta_i}(s_t, a_t)\right]$$

SAC 算法总流程如下：

- 用随机的网络参数ω_1, ω_2和θ分别初始化 Critic 网络$Q_{\omega_1}(s,a)$, $Q_{\omega_2}(s,a)$和 Actor 网络$\pi_\theta(s)$
- 复制相同的参数$\omega_1^- \leftarrow \omega_1$，$\omega_2^- \leftarrow \omega_2$，分别初始化目标网络$Q_{\omega_1^-}$和$Q_{\omega_1^-}$
- 初始化经验回放池R
- **for** 序列$e = 1 \to E$ **do**
- 获取环境初始状态s_1
- **for** 时间步$t = 1 \to T$ **do**
- 根据当前策略选择动作$a_t = \pi_\theta(s_t)$
- 执行动作a_t，获得奖励r_t，环境状态变为s_{t+1}
- 将(s_t, a_t, r_t, s_{t+1})存入回放池R
- **for** 训练轮数$k = 1 \to K$ **do**
- 从R中采样N个元组$\{(s_i, a_i, r_i, s_{i+1})\}_{i=1,\dots,N}$
- 对每个元组，用目标网络计算$y_i = r_i + \gamma \min_{j=1,2} Q_{\omega_j^-}(s_{i+1}, a_{i+1}) - \alpha \log \pi_\theta(a_{i+1}|s_{i+1})$，其中$a_{i+1} \sim \pi_\theta(\cdot|s_{i+1})$
- 对两个 Critic 网络都进行如下更新：对$j = 1, 2$，最小化损失函数$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q_{\omega_j}(s_i, a_i))^2$
- 用重参数化技巧采样动作$\tilde{a}_i$，然后用以下损失函数更新当前 Actor 网络：

$$L_\pi(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\alpha \log \pi_\theta(\tilde{a}_i|s_i) - \min_{j=1,2} Q_{\omega_j}(s_i, \tilde{a}_i)\right)$$

- 更新熵正则项的系数α
- 更新目标网络：

$$\omega_1^- \leftarrow \tau\omega_1 + (1-\tau)\omega_1^- \quad \omega_2^- \leftarrow \tau\omega_2 + (1-\tau)\omega_2^-$$

- **end for**
- **end for**
- **end for**