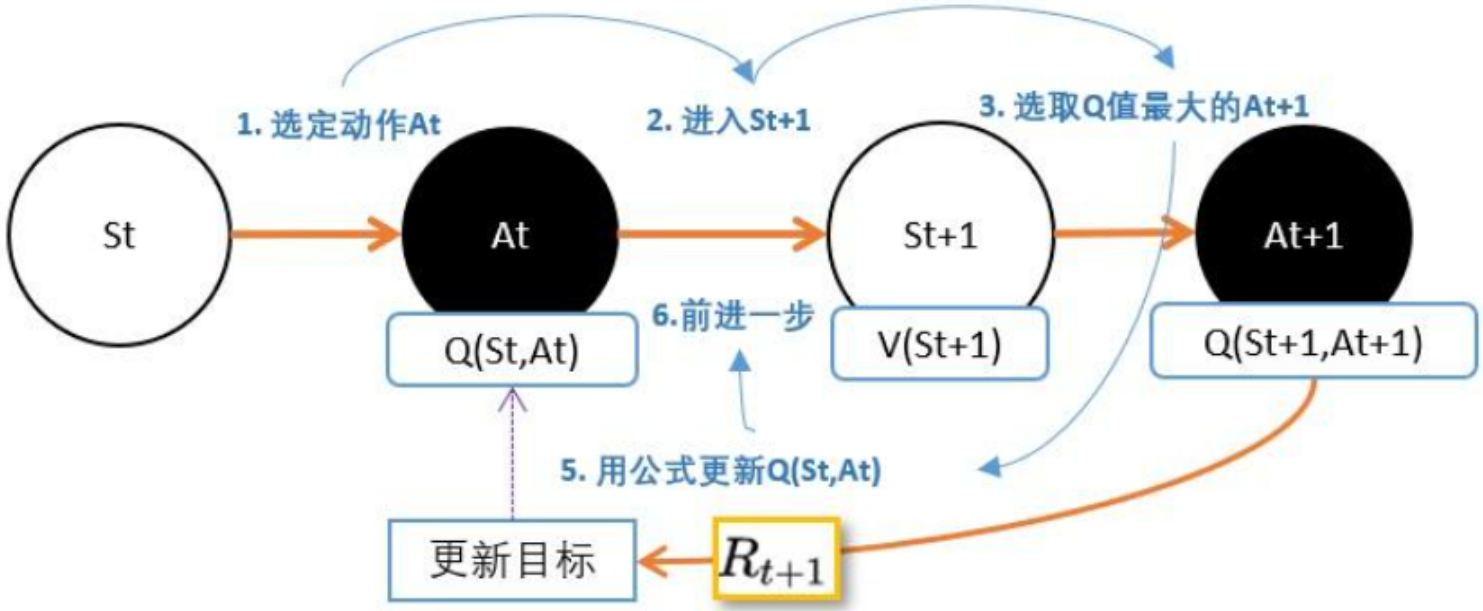
DQN 与 Actor-Critic 的结合

耿佳贺

Peking University

2025.4.9

DQN 算法



$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$$

图 1: DQN 算法

Actor-Critic 算法

Algorithm Steps

1. **Initialize** Actor and Critic network parameters θ, w.
2. **Repeat** the following steps until convergence:
   1. In state s, Actor samples action a according to $\pi_\theta(a|s)$.
   2. Execute action a, receive reward r and next state s'.
   3. Critic computes TD error: $\delta = r + \gamma V^\pi(s') - V^\pi(s)$.
   4. Critic update: $w \leftarrow w + \beta \cdot \delta \cdot \nabla_w V^\pi(s)$.
   5. Actor update: $\theta \leftarrow \theta + \alpha \cdot \nabla_\theta \log \pi_\theta(a|s) \cdot \delta$.

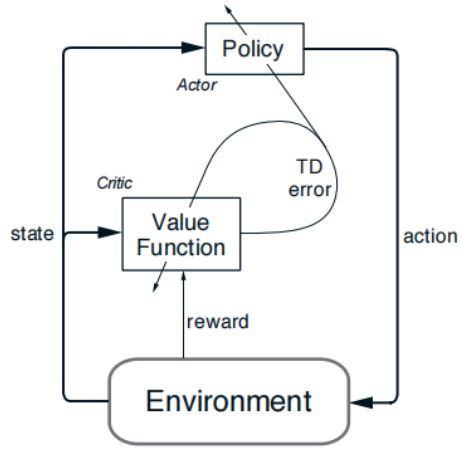


图 2: Actor-Critic

DQN vs. A-C

- **DQN**
  - 优点：Off-policy 算法，采样效率高
  - 缺点：只能处理离散、低维的动作空间，对连续、高维的动作空间，无法给出每个动作的 Q 值
- **Actor-Critic**
  - 优点：通过网络学习策略函数 π，便于处理具有高维、连续动作空间的问题
  - 缺点：On-policy 算法，采样效率低

Actor 部分

Actor 函数：

$$A_t = \pi(S_t|\theta_t^\pi) + N_t$$

- 这里的 Actor 是一个确定性的策略函数，而非随机策略——对于一个连续的动作空间，其概率分布不便于计算
- 在训练过程中引入随机噪声——平衡 Exploration 和 Exploitation

原文中使用 Ornstein-Uhlenbeck 过程产生的具有时间相关性的随机变量作为噪声，但实践证明，时间不相关的零均值高斯噪声效果也很好。

Critic 部分

- **动作** A_t: 在当前状态 S_t 下，通过 Actor 网络 $\pi(S_t|\theta^\pi)$ 选择动作 $A_t = \pi(S_t|\theta_t^\pi) + N_t$
- **计算 Q 值**: $Y_i = R_i + \gamma Q(S_{t+1}, \pi(S_{t+1}|\theta^{\pi'}|\theta^{Q'})$
- **损失函数**: $L = \frac{1}{N}\sum_i (Y_i - Q(S_i, A_i|\theta^Q)^2$

目标网络与指数平滑方法

同样地，仿照 DQN 算法，我们知道引入目标网络可以稳定训练过程。也就是说，在 DDPG 方法中，实际存在着四个网络，分别是 Actor 网络 $\pi(s|\theta^{\pi})$，Critic 网络 $Q(s,a|\theta^{Q})$，以及其对应的两个目标网络 $\pi(s|\theta^{\pi'})$，$Q(s,a|\theta^{Q'})$
目标参数的更新方式：

$$\theta^{Q'} \leftarrow \rho\theta^{Q} + (1-\rho)\theta^{Q'}$$

$$\theta^{\pi'} \leftarrow \rho\theta^{\pi} + (1-\rho)\theta^{\pi'}$$

其中参数 $\rho \ll 1$，保证了目标网络的稳定性。

DDPG 算法

算法 6.26 DDPG

超参数：软更新因子 ρ，奖励折扣因子 γ。
输入：回放缓存 $\mathcal{D}$，初始化 critic 网络 $Q(s,a|\theta^Q)$ 参数 θ^Q、actor 网络 $\pi(s|\theta^\pi)$ 参数 θ^π、目标网络 Q'、π'。
初始化目标网络参数 Q' 和 π'，赋值 $\theta^{Q'} \leftarrow \theta^Q, \theta^{\pi'} \leftarrow \theta^\pi$。
for episode $= 1, M$ **do**
  初始化随机过程 $\mathcal{N}$ 用于给动作添加探索。
  接收初始状态 S_1。
  for t $= 1, T$ **do**
    选择动作 $A_t = \pi(S_t|\theta^\pi) + \mathcal{N}_t$。
    执行动作 A_t 得到奖励 R_t，转移到下一状态 S_{t+1}。
    存储状态转移数据对 $(S_t, A_t, R_t, D_t, S_{t+1})$ 到 $\mathcal{D}$。
    令 $Y_i = R_i + \gamma(1 - D_t)Q'(S_{t+1}, \pi'(S_{t+1}|\theta^{\pi'})|\theta^{Q'})$
    通过最小化损失函数更新 Critic 网络：
      $L = \frac{1}{N}\sum_i (Y_i - Q(S_i, A_i|\theta^Q))^2$
    通过策略梯度的方式更新 Actor 网络：
      $\nabla_{\theta^\pi} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=S_i, a=\pi(S_i)} \nabla_{\theta^\pi}\pi(s|\theta^\pi)|_{S_i}$
    更新目标网络：
      $\theta^{Q'} \leftarrow \rho\theta^Q + (1-\rho)\theta^{Q'}$
      $\theta^{\pi'} \leftarrow \rho\theta^\pi + (1-\rho)\theta^{\pi'}$
  end for
end for

图 3: DDPG 算法

关键技术：截断 Double Q-Learning

通过建立两个 Q 值网络来估计下一个状态的值：

$$Q_{\phi_1}(s', a') = Q_{\phi_1}(s', \pi_{\phi_2}(s'))$$

$$Q_{\phi_2}(s', a') = Q_{\phi_2}(s', \pi_{\phi_1}(s'))$$

并使用其中的最小值来计算 Bellman 方程：

$$Y_1 = r + \gamma \min_{i=1,2} Q_{\phi_i}(s', \pi_{\phi_2}(s'))$$

截断的 Double Q-Learning 技术可以有效缓解过估计问题。

关键技术：延迟更新

降低策略网络的更新频率，在价值网络更新 d 次后再更新策略网络，以便在策略更新之前先最小化价值估计的误差，使得 Q 值函数具有更小的方差，从而获得质量更高的策略更新。

关键技术：在目标动作周围进行模糊拟合

原作者认为相似的动作应该具有相似的只估计，因此将目标动作周围的一小块区域的值进行了模糊拟合，通过在每个动作中加入截断的正态分布噪声作为正则化，可以平滑 Q 值的计算，避免过拟合。修正后的更新如下：

$$y = r + \gamma Q_{\phi}(s', \pi_{\phi'}(s') + \epsilon), \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$$

孪生延迟 DDPG 算法

算法 6.27 TD3

超参数：软更新因子 ρ、回报折扣因子 γ、截断因子 c
输入：回放缓存 $\mathcal{D}$，初始化 Critic 网络 $Q_{\theta_1}, Q_{\theta_2}$ 参数 θ_1, θ_2，初始化 Actor 网络 π_ϕ 参数 ϕ
初始化目标网络参数 $\hat{\theta}_1 \leftarrow \theta_1, \hat{\theta}_2 \leftarrow \theta_2, \hat{\phi} \leftarrow \phi$
for $t = 1$ to T do **do**
  选择动作 $A_t \sim \pi_\phi(S_t) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$
  接受奖励 R_t 和新状态 S_{t+1}
  存储状态转移数据对 $(S_t, A_t, R_t, D_t, S_{t+1})$ 到 $\mathcal{D}$
  从 $\mathcal{D}$ 中采样大小为 N 的小批量样本 $(S_t, A_t, R_t, D_t, S_{t+1})$
  $\tilde{a}_{t+1} \leftarrow \pi_{\phi'}(S_{t+1}) + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}, -c, c))$。
  $y \leftarrow R_t + \gamma(1 - D_t) \min_{i=1,2} Q_{\theta_i'}(S_{t+1}, \tilde{a}_{t+1})$
  更新 Critic 网络 $\theta_i \leftarrow \arg\min_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(S_t, A_t))^2$
  if t mod d **then**
    更新 ϕ：
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(S_t, A_t)|_{A_t = \pi_\phi(S_t)} \nabla_\phi \pi_\phi(S_t)$
    更新目标网络：
    $\hat{\theta}_i \leftarrow \rho\theta_i + (1 - \rho)\hat{\theta}_i$
    $\hat{\phi} \leftarrow \rho\phi + (1 - \rho)\hat{\phi}$
  end if
end for

图 4: 孪生延迟 DDPG 算法

Frame Title

Thanks!