

1 DQN

1.1 DQN Method

DQN (Deep Q-Network) 算法最终更新的目标是让动作价值函数 $Q_{\omega}(s, a)$ 逼近时序差分 (TD) 目标 $r + \gamma \max_{a'} Q_{\omega}(s', a')$ ，其中 ω 表示网络参数， s 为当前状态， a 为当前动作， r 为即时回报， s' 为下一状态， $\gamma \in [0, 1]$ 为折扣因子。

由于 TD 误差目标本身包含神经网络的输出，因此在更新网络参数 ω 的同时，目标值也会随之不断改变，这容易导致神经网络训练的不稳定性（如参数震荡、收敛缓慢）。

为解决这一问题，DQN 引入了**目标网络 (Target Network)** 的核心思想：暂时固定 TD 目标中的 Q 网络，避免目标值随训练网络实时波动。

具体实现

需构建两套结构完全相同但参数更新不同步的 Q 网络

1. 训练网络 $Q_{\omega}(s, a)$ ：用于计算损失函数中的预测项 $Q_{\omega}(s, a)$ ，通过梯度下降实时更新参数 ω ；
2. 目标网络 $Q_{\omega^{-}}(s, a)$ ：用于计算 TD 目标项 $r + \gamma \max_{a'} Q_{\omega^{-}}(s', a')$ ，其中 ω^{-} 为目标网络参数。目标网络不随训练网络实时更新，而是每隔 C 步（超参数）复制一次训练网络的当前参数，即 $\omega^{-} \leftarrow \omega$ ，以此保证目标值的稳定性。

此时，DQN 的损失函数修正为：

$$L(\omega) = \frac{1}{N} \sum_{i=1}^N \left[Q_{\omega}(s_i, a_i) - \left(r_i + \gamma \max_{a'} Q_{\omega^{-}}(s_{i+1}, a') \right) \right]^2$$

其中 N 为经验回放池的采样批量大小。

1.2 DQN Algorithm

Algorithm 1 DQN 算法完整流程

Require: 环境状态空间 \mathcal{S} 、动作空间 \mathcal{A} 、折扣因子 γ 、学习率 α 、探索率 ϵ (衰减策略)、目标网络更新间隔 C 、经验回放池容量 M 、采样批量 N 、训练总序列数 E 、每个序列最大时间步 T

Ensure: 训练稳定的动作价值网络 $Q_{\omega}(s, a)$

```

1: 随机初始化训练网络参数  $\omega$ , 构建  $Q_{\omega}(s, a)$ 
2: 复制参数  $\omega^- \leftarrow \omega$ , 初始化目标网络  $Q_{\omega^-}(s, a)$ 
3: 初始化经验回放池  $R$  (容量为  $M$ )
4: 初始化探索率  $\epsilon$  (如  $\epsilon_{\text{init}} = 1.0$ )
5: for  $e = 1$  to  $E$  do
6:   获取环境初始状态  $s_1$  (将状态转换为网络输入格式, 如向量/矩阵)
7:   重置当前序列的累计回报  $R_{\text{total}} = 0$ 
8:   for  $t = 1$  to  $T$  do
9:     生成随机数  $\text{rand} \sim U(0, 1)$ 
10:    if  $\text{rand} < \epsilon$  then
11:      随机选择动作  $a_t \in \mathcal{A}$  (探索)
12:    else
13:      选择贪婪动作  $a_t = \arg \max_{a \in \mathcal{A}} Q_{\omega}(s_t, a)$  (利用)
14:    end if
15:    执行动作  $a_t$ , 获得即时回报  $r_t$ 、下一状态  $s_{t+1}$ 、终止标志  $\text{done}$ 
16:     $R_{\text{total}} = R_{\text{total}} + r_t$ 
17:    将经验元组  $(s_t, a_t, r_t, s_{t+1}, \text{done})$  存入  $R$  (若容量超  $M$ , 替换旧经验)
18:    if  $\text{len}(R) \geq N$  then
19:      从  $R$  中随机采样  $N$  个经验  $\{(s_i, a_i, r_i, s_{i+1}, \text{done}_i)\}_{i=1}^N$ 
20:      对每个采样经验计算 TD 目标  $y_i$ :

$$y_i = \begin{cases} r_i & \text{若 } \text{done}_i = \text{True} \text{ (终止状态)} \\ r_i + \gamma \max_{a' \in \mathcal{A}} Q_{\omega^-}(s_{i+1}, a') & \text{否则} \end{cases}$$

21:      计算损失函数  $L = \frac{1}{N} \sum_{i=1}^N (y_i - Q_{\omega}(s_i, a_i))^2$ 
22:      基于梯度下降更新训练网络参数  $\omega$  ( $\omega \leftarrow \omega - \alpha \nabla_{\omega} L$ )
23:    end if
24:    if  $t \bmod C == 0$  then
25:      复制训练网络参数到目标网络:  $\omega^- \leftarrow \omega$ 
26:    end if
27:    if  $\text{done}$  then
28:      输出当前序列回报  $R_{\text{total}}$ , 跳出时间步循环
29:    else
30:       $s_t = s_{t+1}$  (状态转移)
31:    end if
32:  end for
33:   $\epsilon = \epsilon \times \epsilon_{\text{decay}}$  (如  $\epsilon_{\text{decay}} = 0.995$ )
34:   $\epsilon = \max(\epsilon_{\text{min}}, \epsilon)$  (设置最小探索率, 如  $\epsilon_{\text{min}} = 0.01$ )
35: end for
```

- 经验回放池: 避免样本相关性导致的训练不稳定, 通过随机采样打破经验的时序关联;
- 目标网络: 固定 TD 目标值, 减少参数更新时的目标波动, 提升收敛稳定性;
- ϵ -贪婪策略: 平衡探索 (发现新动作) 与利用 (选择已知最优动作), 通过衰减 ϵ 逐步降低探索比例。

2 Double DQN

DQN 算法通过经验回放和目标网络两大机制，解决了高维状态空间下 Q 学习的训练不稳定性问题，但仍存在一个关键缺陷——Q 值过估计（Overestimation Bias）。

这种过估计源于 DQN 的目标值计算方式：使用同一目标网络同时完成“动作选择”和“价值评估”，神经网络的逼近误差会被 max 操作放大，导致 Q 值系统性偏高，进而引发策略次优、收敛速度减慢等问题。

为解决这一问题，便有了 Double DQN（DDQN）算法，其核心思想源于经典的 Double Q-Learning：将动作选择与价值评估解耦，通过两个独立的网络分别完成这两项任务，从而打破过估计的正向反馈循环，显著提升 Q 值估计的准确性和算法稳定性。

“解耦”（Decoupling）指的是将 DQN 中“动作选择”和“价值评估”这两个原本绑定在同一网络的任务，拆分给两个独立网络分别完成，避免单一网络的偏差被放大，核心是“职责分离、相互独立”。

2.1 DDQN and DQN

对比维度	DQN 算法	DDQN 算法
目标值计算逻辑	目标网络同时负责动作选择与评估	在线网络选动作，目标网络做评估
目标值公式	$y_t^{DQN} = r_t + \gamma \max_{a'} Q_{\theta^-}(s_{t+1}, a')$	$y_t^{DDQN} = r_t + \gamma Q_{\theta^-}(s_{t+1}, \arg \max_{a'} Q_{\theta}(s_{t+1}, a'))$
过估计风险	高（单一网络偏差被 max 放大）	低（双网络解耦，偏差相互抵消）
网络结构	双网络（训练网络 + 目标网络）	双网络（在线网络 + 目标网络，结构一致）
实现成本	基础版本	仅修改目标值计算，无额外开销

表 1: DQN and DDQN

DDQN 核心机制：解耦动作选择与价值评估

DDQN 沿用 DQN 的“双网络”架构，但赋予两个网络明确的分工：

1. 在线网络 ($Q_{\theta}(s, a)$)：负责动作选择，即通过 $\arg \max_{a'} Q_{\theta}(s_{t+1}, a')$ 确定下一状态的最优动作，反映当前训练策略的偏好；
2. 目标网络 ($Q_{\theta^-}(s, a)$)：负责价值评估，即对在线网络选择的最优动作进行 Q 值计算，提供稳定的目标值基准。

这种分工的关键优势在于：即使两个网络都存在估计误差，它们同时高估同一动作价值的概率极低，从而有效抑制过估计偏差的累积。

2.2 DDQN Method

DDQN 的损失函数形式与 DQN 一致，仍采用均方误差 (MSE) 损失，但目标值替换为解耦计算后的 y_t^{DDQN} ：

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N \left[Q_{\theta}(s_i, a_i) - y_i^{DDQN} \right]^2$$

其中， y_i^{DDQN} 为 DDQN 的目标值，分两种情况计算：

$$y_i^{DDQN} = \begin{cases} r_i & \text{若 } done_i = \text{True} \text{ (终止状态, 无未来奖励)} \\ r_i + \gamma Q_{\theta-}(s_{i+1}, \arg \max_{a'} Q_{\theta}(s_{i+1}, a')) & \text{否则} \end{cases}$$

$\gamma \in [0, 1]$ 为折扣因子， N 为经验回放池的采样批量大小。

DDQN 算法优势：

- 显著缓解过估计：实验证明在 Atari 游戏等任务中，DDQN 的平均 Q 值更接近真实值，策略更稳健；
- 兼容性强：可与 DQN 的其他改进（如优先经验回放、Dueling 架构）无缝结合；
- 稳定性提升：目标值计算更可靠，避免因过估计导致的策略震荡，收敛速度更快。

2.3 DDQN Algorithm

Algorithm 2 Double DQN (DDQN) 完整算法流程

Require: 环境状态空间 \mathcal{S} 、动作空间 \mathcal{A} 、折扣因子 γ 、学习率 α 、探索率 ϵ (衰减策略)、目标网络更新间隔 C 、经验回放池容量 M 、采样批量 N 、训练总序列数 E 、每个序列最大时间步 T

Ensure: 低偏差的动作价值网络 $Q_\theta(s, a)$

```

1: 随机初始化在线网络参数  $\theta$ , 构建  $Q_\theta(s, a)$  (负责动作选择与预测)
2: 复制参数  $\theta^- \leftarrow \theta$ , 初始化目标网络  $Q_{\theta^-}(s, a)$  (负责价值评估)
3: 初始化经验回放池  $R$  (容量为  $M$ )
4: 初始化探索率  $\epsilon$  (如  $\epsilon_{\text{init}} = 1.0$ , 最小探索率  $\epsilon_{\text{min}} = 0.01$ )
5: for  $e = 1$  to  $E$  do
6:   获取环境初始状态  $s_1$  (转换为网络输入格式, 如向量/矩阵)
7:   重置当前序列累计回报  $R_{\text{total}} = 0$ 
8:   for  $t = 1$  to  $T$  do
9:     生成随机数  $\text{rand} \sim U(0, 1)$ 
10:    if  $\text{rand} < \epsilon$  then
11:      随机选择动作  $a_t \in \mathcal{A}$  (探索)
12:    else
13:      在线网络选贪婪动作:  $a_t = \arg \max_{a \in \mathcal{A}} Q_\theta(s_t, a)$  (利用)
14:    end if
15:    执行动作  $a_t$ , 获得即时回报  $r_t$ 、下一状态  $s_{t+1}$ 、终止标志  $\text{done}$ 
16:     $R_{\text{total}} = R_{\text{total}} + r_t$ 
17:    将经验元组  $(s_t, a_t, r_t, s_{t+1}, \text{done})$  存入  $R$  (超容量时替换旧经验)
18:    if  $\text{len}(R) \geq N$  then
19:      从  $R$  中随机采样  $N$  个经验  $\{(s_i, a_i, r_i, s_{i+1}, \text{done}_i)\}_{i=1}^N$ 
20:      对每个采样经验, 用在线网络选择下一状态最优动作:  $a_i^* = \arg \max_{a' \in \mathcal{A}} Q_\theta(s_{i+1}, a')$ 
21:      用目标网络计算 DDQN 目标值  $y_i^{\text{DDQN}}$ :

$$y_i^{\text{DDQN}} = \begin{cases} r_i & \text{若 } \text{done}_i = \text{True} \\ r_i + \gamma Q_{\theta^-}(s_{i+1}, a_i^*) & \text{否则} \end{cases}$$

22:      计算损失函数  $L = \frac{1}{N} \sum_{i=1}^N (y_i^{\text{DDQN}} - Q_\theta(s_i, a_i))^2$ 
23:      基于梯度下降更新在线网络参数:  $\theta \leftarrow \theta - \alpha \nabla_\theta L$ 
24:    end if
25:    if  $t \bmod C == 0$  then
26:      复制在线网络参数到目标网络:  $\theta^- \leftarrow \theta$ 
27:    end if
28:    if  $\text{done}$  then
29:      输出当前序列回报  $R_{\text{total}}$ , 跳出时间步循环
30:    else
31:       $s_t = s_{t+1}$  (状态转移)
32:    end if
33:  end for
34:   $\epsilon = \epsilon \times \epsilon_{\text{decay}}$  (如  $\epsilon_{\text{decay}} = 0.995$ )
35:   $\epsilon = \max(\epsilon_{\text{min}}, \epsilon)$  (限制最小探索率)
36: end for
```

算法说明

- 目标网络更新: 仍采用“硬更新”策略 (每隔 C 步复制参数), 也可采用“软更新” ($\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^-$, τ 为小权重如 0.001), 进一步提升稳定性;
- 经验回放池: 功能与 DQN 一致, 用于打破样本时序相关性, 避免参数更新偏向近期经验;

- 适用场景：适用于所有 DQN 可解决的任务，尤其在奖励稀疏、动作空间复杂的环境中，过估计缓解效果更显著（如机器人导航、复杂游戏 AI）；
- 扩展方向：DDQN 是 Rainbow DQN（集成改进算法）的核心组件之一，可与优先经验回放（PER）、竞争网络（Dueling DQN）等结合，进一步提升性能。