第 2 章 相关理论与技术

本章首先将要介绍强化学习相关基础，主要介绍了一些值函数的方法。随后将对于末制导问题进行简要物理学描述，得到物理学方程。最后对于末制导环境进行稀疏奖励下的强化学习马尔可夫决策过程建模。

2.1 强化学习基础

强化学习是一种在交互中学习的机器学习方法。把进行学习和决策的对象称为智能体（agent），智能体外部所有与其进行交互的事物称为环境（environment）[11]。智能体与环境进行交互，感知到环境的状态（state）进而采取动作（action），这个动作会使环境的状态发生变化，并使得智能体收到一个来自于环境的即时奖励（reward），强化学习的目标就是探究如何把当前的环境状态映射到动作，使得累积收益最大化。

2.1.1 有限马尔可夫决策过程

马尔可夫决策过程（MDP）是序列决策的经典形式化表达[1]，绝大多数强化学习方法的研究都是建立在 MDP 框架之上。

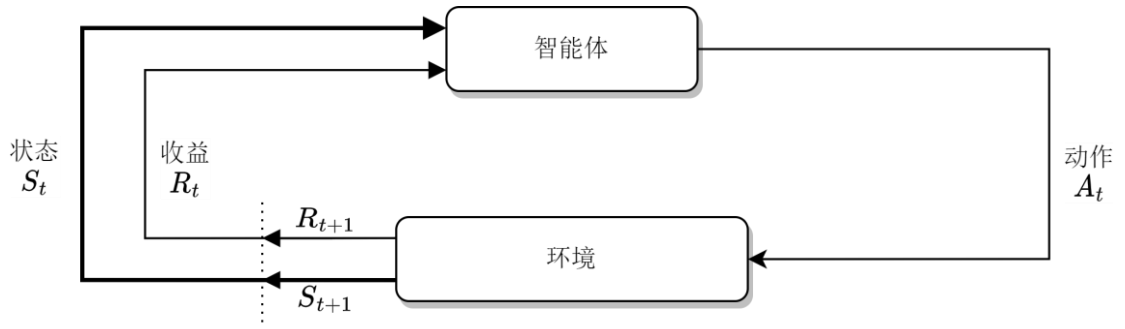

图 2-1 马尔可夫决策过程

如图 2-1 所示，在每个离散时刻 $t = 0,1,2,3,\ldots$，智能体与环境进行交互，在时刻 t，智能体观测到环境状态 $S_t \in \mathcal{S}$，并根据这个状态选择一个动作 $A_t \in \mathcal{A}(s)$，下一时刻，智能体接收到一个来自环境的即时奖励 $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$，并进入到下一个状态 S_{t+1}，由此，MDP 和智能体共同给出了一个轨迹（trajectory）

$$\tau = S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots \tag{2-1}$$

在有限 MDP 的情况下，即状态空间、动作空间和奖励的集合（$\mathcal{S}$、$\mathcal{A}$ 和 $\mathcal{R}$）都只有有限个元素时，下一时刻的状态 S_{t+1} 只与当前时刻的状态 S_t 和动作 A_t 有关，状态转移概率为

$$p(s',r|s,a) = Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} \tag{2-2}$$

对于任意 $s', s \in \mathcal{S}$，$r \in \mathcal{R}$，以及 $a \in \mathcal{A}(s)$。整个系统的动态特性由这个状态转移概率来定义。

2.1.2 强化学习的目标

强化学习的目标就是最大化期望回报，记为 G_t，在最简单的情况下，回报是奖励的总和

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \tag{2-3}$$

其中 T 是最终时刻，它适用于有最终时刻的任务，这种情况下可以把智能体与环境交互分成一系列子序列，称为幕（episode），每幕都以“终结状态”结束，下一幕如何开始与上一幕如何结束之间并无关联，最终时刻 T 通常随着幕的不同而发生变化[11]。当 $T = \infty$ 时，显然要最大化的期望回报也很有可能趋于无穷，在此引入折扣率 γ（$0 < \gamma \leq 1$），则折后期望回报为

$$G_t \doteq \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots = R_{t+1} + \gamma G_{t+1} \tag{2-4}$$

策略（policy）是从状态到每个动作的选择概率之间的映射，如果智能体在 t 时刻选择了策略 π，那么 $\pi(a|s)$ 就是在 $S_t = s$ 时 $A_t = a$ 的概率。通常利用价值函数（value function）来评估策略的好坏。状态 s 下遵循策略 π 的价值函数记为 $v_\pi(s)$，称为策略 π 的状态价值函数

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right] \tag{2-5}$$

对于所有 $s \in \mathcal{S}$。类似地，把状态 s 下遵循策略 π 采取动作 a 的价值函数记为 $q_\pi(s,a)$，称为策略 π 的动作价值函数

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a\right] \tag{2-6}$$

对于任意策略 π 和任意状态 s，s 的价值与其后继状态的价值之间存在以下关系

$$v_\pi(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{2-7}$$

称作 v_π 的贝尔曼方程，表示当前状态的价值可以由下一个时刻的价值来计算。

解决一个强化学习任务就意味着要找到一个策略来最大化期望回报，可以通过比较价值函数来判断策略的好坏，最优策略对应的最优状态价值函数记为 v_*，对于任意 $s \in \mathcal{S}$，

$$v_*(s) \doteq \max_\pi v_\pi(s) \tag{2-8}$$

对应的最优动作价值函数记为 q_*，对于任意 $s \in \mathcal{S}$，$a \in \mathcal{A}$，

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a) \tag{2-9}$$

最优价值函数也满足贝尔曼方程，称为贝尔曼最优方程，v_* 的贝尔曼最优方程如下

$$\begin{aligned} v_*(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \end{aligned} \tag{2-10}$$

q_* 的贝尔曼最优方程如下

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_a q_*(S_{t+1}, a') | S_t = s, A_t = a\right] \\ &= \sum_{s',r} \mathrm{p}(\mathrm{s}', \mathrm{r}|\mathrm{s}, \mathrm{a}) \left[r + \gamma \max_a q_*(s', a')\right] \end{aligned} \tag{2-11}$$

2.1.3 基于值函数的学习方法

2.1.3.1 动态规划

假设环境是一个有限 MDP，即状态集合 $\mathcal{S}$、动作集合 $\mathcal{A}$ 和收益集合 $\mathcal{R}$ 是有限的，并且整个系统的动态特性由对于任意 $s \in \mathcal{S}$、$a \in \mathcal{A}$、$r \in \mathcal{R}$ 和 $s' \in \mathcal{S}^+$ 的四参数概率分布 $p(s', r|s, a)$ 给出。如果环境的动态特性 p 已知，原则上可以用解线性方程组的方法求解贝尔曼最优方程的价值函数来得到最优策略。

计算一个任意策略 π 的状态价值函数 v_π 在动态规划（Dynamic Programming，DP）中称为策略评估，有时也称为预测问题。求解线性方程组有些繁琐，因此通过式（2-10）贝尔曼方程迭代计算 v_π 的近似值，每一轮迭代都更新一次所有状态的价值函数，以产生新的近似价值函数。

接下来根据这个给定策略下的价值函数进行策略改进。每个状态下都根据 $q_\pi(s,a)$ 选择一个最优的动作，即采取新的贪心策略 π' 满足

$$\begin{aligned} \pi'(s) &\doteq \operatorname{argmax}_a q_\pi(s,a) \\ &= \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]. \end{aligned} \tag{2-12}$$

策略 π 根据 v_π 去产生一个更好的策略 π'，然后再通过计算 v_π' 来得到一个更优的策略 π''，通过这样一个持续更新的策略和价值函数序列，就能找到最优策略，这种方法称为策略迭代。

2.1.3.2 蒙特卡洛方法

很多时候我们并不具有完备的环境知识，但可以通过与真实或虚拟环境交互从而采样来计算，这就是蒙特卡洛方法。在策略 π 下途径状态 s 采用随机游走的方式生成多幕序列，其中每一幕的轨迹为 $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$，对于每一幕都计算其总回报的平均值，当有越来越多的回报被观察到，平均值就会收敛于期望值。利用蒙特卡罗方法进行策略评估，然后贪心地或者逼近贪心地（软性策略如 ϵ-贪心，即绝大多数时候都选择贪心的策略，但同时以一个较小的 ϵ 概率选择动作用于探索）选择动作进行策略改进。

事实上我们希望学到的动作可以使随后的智能体行为是最优的，但如果不能搜索到其他的非最优的行动，就无法进一步改进策略，这会只是在利用（exploit）当前策略，而不是在探索（explore）环境。一个自然的方法是直接采取两个策略，一个用于学习并不断改进最终成为最优策略，称为目标策略，另一个探索环境用于采样，生成行动数据，称为行动策略，整个过程称为离轨策略学习（off-policy），如果行动策略与目标策略相同则称为同轨策略学习（on-policy）。

2.1.3.3 时序差分学习

对于预测问题，一个适用于非平稳环境的蒙特卡洛方法可以表示成

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{2-13}$$

其中 α 是步长参数。蒙特卡洛方法必须等到一幕的末尾才能确定对 $V(S_t)$ 的增量（因为只有这时 G_t 才是已知的），而时序差分（TD）结合了蒙特卡洛方法和动态规划的思想，在状态转移到 S_{t+1} 并收到 R_{t+1} 的奖励时会立即做出更新

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{2-14}$$

通过对比可以发现，蒙特卡洛更新的目标是 G_t，而 TD 更新的目标是 $R_{t+1} + \gamma V(S_{t+1})$，这种 TD 方法被称为 TD(0)，更新中如公式(2-15)的部分被称为 TD 误差

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{2-15}$$

一种经典的离轨（off-policy）策略下的时序差分算法是 Q 学习，定义为

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{2-16}$$

Q 学习的 TD 目标是当前对于 Q^* 的估计，相当于是在估计另一个贪心的策略，因此是离轨的。

2.2 末制导问题描述与强化学习建模

2.2.1 末制导问题描述

由于真实战场环境难以复现，所以本文将在仿真环境下进行测试与实验。末制导问题是一种经典的序列决策问题，因此下面使用马尔可夫决策过程（MDP）对其进行强化学习建模。

为了降低复杂度，本文将采用二维空间进行研究，其中导弹和目标飞行物均当成质点来描述其物理特性。

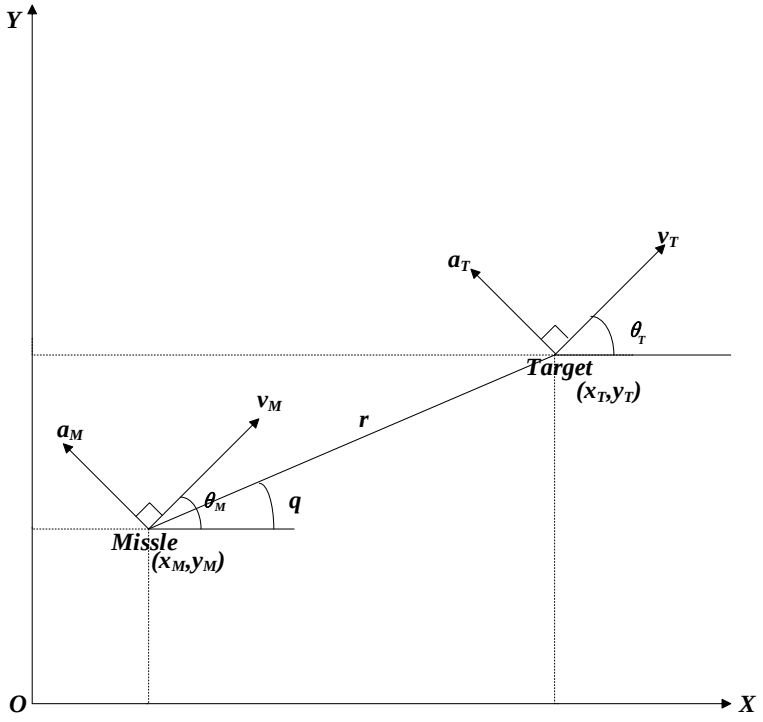

图 2-2 制导场景示意图[1]

建立如图 2-2 所示的惯性参考系，其中水平方向为 x 轴，竖直方向为 y 轴，(x_M, y_M) 和 (x_T, y_T) 分别代表导弹和目标飞行物的坐标；θ_M 和 θ_T 分别代表导弹和目标飞行物速度的倾斜角；q 代表弹目间连线与 x 轴形成的夹角，也称为视线角（LOS）；r 代表导弹和目标飞行物之间的相对距离；a_M 和 a_T 分别代表导弹和目标飞行物的法向加速度，方向垂直于速度矢量的方向；v_M 和 v_T 分别代表导弹和目标飞行物的速度矢量，箭头代表着速度矢量的方向[1]。

结合空气动力学原理，能够得到如下的物理公式：

$$\begin{cases} \dot{r} = V_T \cos(\theta_T - q) - V_M \cos(\theta_M - q) \\ r\dot{q} = V_T \sin(\theta_T - q) - V_M \sin(\theta_M - q) \end{cases} \tag{2-17}$$

$$q = arctan\left(\frac{y_T - y_M}{x_T - x_M}\right) \tag{2-18}$$

其中 $\dot{r}$ 是导弹与目标飞行物的相对速度，$\dot{q}$ 为视线转率[1]。

比例制导律的公式如下：

$$a_M = \frac{N|\dot{r}|\dot{q}}{\cos(\theta_M - q)} \tag{2-19}$$

其中 N 为导航比，在传统的比例制导律中为固定值，取值范围通常为 [2,6]。由于传统比例制导律灵活性不足，因此在这里采用广义比例制导律：

$$a_M = \frac{N_1|\dot{r}|\dot{q} + N_2 a_T cos(\theta_T - q)}{cos(\theta_M - q)} + g \cdot cos(\theta_M) \tag{2-20}$$

其中N_1、N_2 为导航比。

常规情况下对于制导效果的衡量都使用零控脱靶量（Zero Effort Miss, ZEM），即导弹和目标飞行物在不施加任何控制的情况下、保持当前运动状态时弹目之间的最小距离。但本文涉及到的运动是动态的，这种适用于静态的方式就不再合适，因此本文采取脱靶量（Miss Distance），即导弹到目标飞行物质心间的最短距离来衡量制导效果[1]。

由于目标飞行物与导弹之间是对抗关系，因此目标飞行物将采取不同的机动方式来躲避导弹的攻击，在本次仿真环境中本文设计了几种不同的机动方式如下：

（1）向上机动：

$$a_t = -n \cdot g \tag{2-21}$$

（2）向下机动：

$$a_t = n \cdot g \tag{2-22}$$

（3）正弦上下机动：

$$a_t = n \cdot g \cdot sgn\big(sin(t)\big) \tag{2-23}$$

其中 a_t 为目标飞行物的控制量也就是加速度；n 为常数，表示加速度的幅度，可以看作是超参数；上下机动方式中的 t 是时间步，通过正弦函数和指示函数对时间步进行修正[1]。在实际仿真环境测试中，除了上述三种固定的机动方式，本文还会采取每一个时间步都从这三种机动方式中随机抽取来调整目标飞行物机动方式的随机机动方式。

2.2.2 马尔可夫决策过程设计

（1）状态空间设计

状态空间是指强化学习中的所有可能的状态的集合，它决定了智能体可以观察到的环境信息，以及可以采取的动作。状态空间的设计对于强化学习的效率和性能有重要影响。如果状态空间太大或太复杂，那么智能体可能需要很多数据和时间才能探索和学习；但如果状态空间覆盖度不足，即不能充分表示当前的环境信息，那么智能体可能难以从数据中学习到有效的策略。

在仿真时应该更加重视导弹与目标飞行物之间的相对关系，防止绝对位置对结果造成影响。因此在本次仿真环境的状态空间设计中，对于弹目状态的表达都采用了相对值。

本文共设计了 6 个维度：弹目之间的相对距离 r、弹目之间的相对速度 v_{rela}、弹目之间的视线角 q、弹目间的视线偏转率 $\dot{q}$、为了对导弹速度加以限制而引入的导弹速度 v、为了对导弹偏转角加以限制而引入的弹道倾角 θ_m。该状态向量表示如下：

$$S:<r,v_{rela},q,\dot{q},v,\theta_m> \tag{2-24}$$

（2）动作空间设计

在强化学习中，动作空间是指智能体在特定环境中可以采取的所有有效动作的集合。动作空间的设计应该尽可能地简化，避免无效或冗余的动作，以减少搜索空间和提高学习速度。并且应该尽可能地覆盖所有可能的状态转移，以保证智能体能够实现预期的目标。

为了简化和尽可能地覆盖所有可能的状态转移，本文采用了导航比作为动作。由于使用的是广义比例制导律，即公式 (2-20)，其中导航比 N_1、N_2 均为常数且恒定，因此面对多变的目标机动其适应能力较差，导致制导性能有所下降。本文通过将导航比 N_1、N_2 进行离散化扩展，其中导航比 N_1 在 $\{2,3,4,5,6\}$ 中取值；导航比 N_2 在 $\{0,0.5,1,1.5,2\}$ 中取值，通过这种方式实现导航比的动态选取。在实际状态转移时需要将导航比索引映射成实际制导中的导弹加速度。

（3）奖励函数设计

稀疏奖励是指在强化学习中，智能体在状态空间中只有少数的状态能够收到反馈信号的情况。导弹只有在命中目标才会得到奖励，在其它情况下无论导弹距离目标飞行物是偏离还是靠近，都无法得到奖励，无法从环境中得到任何反馈，是一种非常经典的稀疏场景，因为本文要在稀疏奖励场景下进

行末制导律设计，因此在仿真环境中设计的奖励函数也是二值稀疏的，即：

$$R_t = \begin{cases} 1, & miss\ diatance < \delta \\ 0, & otherwise \end{cases} \tag{2-25}$$

为了保证导弹的有效杀伤力，在仿真环境中将 δ 取值为 3，单位为 m，也就是只有在脱靶量小于 3 米时才会得到一个奖励，R_t 表示 t 时刻的即时奖励。

2.3 本章小结

本章首先简要介绍了有限马尔可夫决策过程，然后介绍了强化学习的目标，即最大化期望回报。之后介绍了强化学习的一些值函数的方法，比如动态规划、蒙特卡洛方法、时序差分学习，时序差分学习中一种经典的方法是 Q 学习。

随后本章对于末制导问题进行了物理学描述，并结合空气动力学得到了物理学方程。最后利用这些物理学方程对于本次仿真实验使用的仿真末制导环境进行了马尔可夫过程建模，为后续两种末制导律的设计提供了基础。

第 3 章 基于优先经验回放（PER）的强化学习末制导律设计

本章首先要介绍深度 Q 网络，并在此基础之上引入优先经验回放的思想。随后将基于优先经验回放算法进行末制导律设计。最后本章将在仿真环境中对于优先经验回放末制导律进行测试，并将效果和 DQN 进行对比，验证该方法的有效性。

3.1 深度 Q 网络

在第 2 章介绍时序差分学习时，曾经简略提到，Q 学习是一种基于值函数的强化学习方法，它可以用一个 Q 表来存储每个状态-动作对的价值，然后通过贝尔曼方程来更新 Q 表中的值。但 Q 学习有一些缺点，比如当状态空间或动作空间很大时，Q 表会占用很多内存，而且难以收敛。此外，Q 学习只能处理离散的问题，不能处理连续的状态或动作空间。

为了解决这些问题，深度 Q 网络（DQN）被提出。DQN 是一种结合了深度神经网络和 Q 学习的算法，它可以用神经网络来近似 Q 函数，从而减少内存需求和提高泛化能力。DQN 采用了神经网络函数逼近的方法来估计动作值函数，

$$Q(s,a;\theta) \approx Q^*(s,a) \tag{3-1}$$

将权重为 θ 的这个用于函数逼近的神经网络称为 Q 网络。Q 网络可以通过最小化在每次迭代 i 时发生变化的一系列损失函数 $L_i(\theta_i)$ 来训练，

$$L_i(\theta_i) = \mathbb{E}_\pi\left[\left(y_i - Q(S_t,A_t;\theta_i)\right)^2\right] \tag{3-2}$$

即 δ^2，其中 $y_i = R_{t+1} + \gamma \max_a Q(S_{t+1},a;\theta_{i-1})$，在优化损失函数 $L_i(\theta_i)$时，上一个迭代的参数 θ_i 是固定的。

深度 Q 网络（DQN）[2]采取两个核心思想：一是经验回放（experience replay），就是将每个转换（transition） (S_t,A_t,R_t,S_{t+1}) 存储在经验池里，为了降低数据的相关性；二是固定 Q 目标（fixed Q-targets），是指在一段时间内固定目标网络中的参数，来减少训练中参数的震荡和发散。

实现时，在经验池里随机抽取样本进行训练，这样可以避免相邻训练数据的相似性，从而防止陷入局部最优。

3.2 优先经验回放（PER）

前面介绍过的 DQN 是在经验池里随机选取采样，但事实上样本的优先级对结果有着重要影响。优先经验回放（Prioritized Experience Replay, PER）[3]在 DQN 之上作出了改进，其核心思想就是对于每个转换 (s_t, a_t, r_t, s_{t+1}) 重要性标准的衡量。这里使用 TD 误差来衡量样本的的重要性，但如果仅仅采用贪心的方式，会导致经验池里初始 TD 误差较小的样本可能永远不会被采样到，且会高频重复使用 TD 误差较高的样本，导致过拟合。于是采取了一种随机采样方法（stochastic sampling method），并且由于新的 transition 存入时并没有已知的 TD 误差，默认将其设置为最大的 TD 误差，以保证所有的样本至少被抽到一次。

3.2.1 随机采样方法

对于每一个样本都计算出一个概率 p_i，分为两种方法：一是按比例确定优先级（proportional prioritization），根据 TD 误差 δ 确定概率，$p_i = |\delta_i| + \epsilon$；二是按排序确定优先级（rank-based prioritization），根据样本的 δ 从大到小排序，$p_i = \frac{1}{rank(i)}$，其中 $rank(i)$ 指第 i 个样本在全体样本中的排名。然后随机采样方法利用概率来选择样本，

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{3-3}$$

其中 α 用于控制均匀采样和贪心的偏好，$\alpha = 0$ 时为均匀采样，$\alpha = 1$ 时为贪心。

按比例确定优先级（proportional prioritization）在实现上采用了求和树（sum tree）这样的二叉树结构[12]（如图 3-1 所示）来存储存含有优先级的经验池。所有经验回放样本都只存储在叶节点上，每个叶节点都对应着一个样本，存储着它的数据和优先级。内部节点不存储数据，只保存它的子节点的优先级之和。

图 3-1 为采样 s=24（$0 \le s \le \sum_k p_k$）的过程。假设所有节点的优先级之和为 42，即根节点为 42，因此在 (0-42) 区间均匀采样，假设本次采样随机到了 s=24，那么它最终会使用 (13-25) 这个区间上的数据（即优先级为 12 的那个节点对应的数据），优先级越高的叶子节点对应的区间越长，因此被采样到的概率也越大。具体流程为：s=24 从根节点 42 向下搜索，左孩子 29 > 24，于是更新到左子树，更新后的左孩子 13<24，则更新到右子树，同时将 s=24

更新为 s=24-13=11，左孩子 12>11，于是最后选择 12 对应的数据。假如需要采样多个样本，在本例中假设需要采样 3 个样本，则把 (0-42) 分成 3 个区间 (0-14)、(14-28)、(28-42)，在三个区间上分别随机选取一个数据。

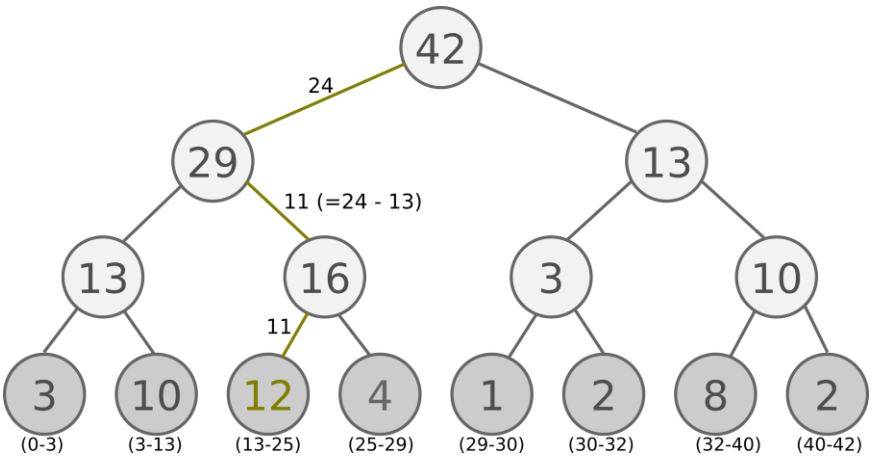

图 3-1 求和树[12]

按排序确定优先级（rank-based prioritization）在实现上类似于分层采样，将排名分成多个等概率区间，然后再各个等概率区间上均匀采样。这种方案无法反映 δ 的信息，但是稳定性较强。

3.2.2 消除偏差

除了优化经验池的存储结构，PER 在损失函数上也作出了优化。此前的损失函数为公式(3-2)，现在加入了一个优先级权重 ω_i 来消除偏差，避免对梯度造成影响，从而收敛到相同的值，

$$L_i(\theta_i) = \mathbb{E}_\pi\left[w_i\big(y_i - Q(S_t, A_t; \theta_i)\big)^2\right] \tag{3-4}$$

$$w_i = \left(\frac{1}{N}\cdot\frac{1}{p(i)}\right)^\beta \tag{3-5}$$

其中 N 为经验池中的样本数目，最后为了增加稳定性，将 w_i 除以 $\max_i w_i$ 向下缩放，以减小梯度更新幅度。$\beta \in [0,1]$，在学习快结束时，使 $\beta \to 1$。

3.3 基于优先经验回放算法的强化学习末制导律设计

依据前文介绍的 PER 算法原理，结合第 2 章的末制导马尔可夫决策过程设计，本文提出了以下基于优先经验回放算法的强化学习末制导律设计方案。

算法伪代码如下：

算法 1：基于优先经验回放算法的强化学习末制导律设计[3]

输入：状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣率 γ，小批次量 k，步长 η，经验池中样本数目 N，采样与更新之间的时间间隔 K，参数 α、β

输出：训练得到的网络参数 θ

初始化环境，指定弹目初始位置、速度、导弹弹道倾角、目标机动方式

初始化经验池 $\mathcal{H}$ 为空集

初始化用于更新网络权值 θ 而设置的参数 $\Delta = 0$

初始化起始状态 S_0

for t = 1, T **do**

利用 $\epsilon-$贪心选择导航比 N_1、N_2 的组合 $A_{t-1} \sim \pi_\theta(S_{t-1})$

将 A_{t-1} 映射称为真实的动作 A'_{t-1}

观测 S_t, R_t

在经验池 $\mathcal{H}$ 中存储 transition $(S_{t-1}, A'_{t-1}, R_t, S_t)$ 和最高优先级 $p_t = \max_{i<t} p_i$

if t ≡ 0 mod K **then**

for j = 1, k **do**

随机小批量从经验池中采样 transition $j \sim P(j) = \frac{p_j^\alpha}{\sum_i p_i^\alpha}$

计算重要性采样权值 $w_j = \left(\frac{1}{N} \cdot \frac{1}{p(j)}\right)^\beta / \max_i w_i$

计算 TD 误差 $\delta_j = R_j + Q_{target}\left(S_j, \operatorname{argmax}_a Q(S_j, a)\right) - Q(S_{j-1}, A_{j-1})$

更新 transition 的优先级 $p_j \leftarrow |\delta_j|$

计算权重变化 $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$

end for

更新权值 $\theta \leftarrow \theta + \eta \cdot \Delta$，重置 $\Delta = 0$

间歇性将权值复制到目标网络 $Q_{target} \leftarrow \theta$

end if

end for

可以看到，PER 整体上的算法结构和 DQN 类似，但 PER 和 DQN 有两个最为显著的区别：一是 PER 为经验池中的 transition 增加了优先级的衡量，优先级高的更容易被采样到；二是 PER 为损失函数增加了一个重要性采样权值，以消除偏差和更好地收敛。

本次在具体实现上选择了按比例确定优先级（proportional prioritization）的方法去采样。因此采用求和树这种数据结构来存储 PER 的经验池，所占据的区间越长，优先级就越高，被采样到的概率也就越大，详细采样过程已在 3.2.1 节的理论部分加以说明，此处不再赘述。

3.4 仿真实验

本节将介绍仿真实验具体部署细节，如超参数设置等，并将本章基于优先经验回放的末制导律设计方案与 DQN 进行对比，测试在稀疏奖励环境下该方案的有效性。

3.4.1 实验环境部署

表 3-1 和 3-2 中分别列出了初始化环境的参数值和算法的参数：

表 3-1 末制导仿真环境初始参数表

参数	值	单位
导弹初始位置	(0,11000)	m
导弹初始速度倾角	-25	degrees
导弹最大过载（加速度）	20g	m/s^2
导弹初始速度	600	m/s
目标初始位置	(6000,8000)	m
目标初始速度	300	m/s
目标初始加速度	2g	m/s^2

以上述的导弹初始位置、导弹初始速度倾角、导弹最大加速度、导弹初始速度、目标初始位置、目标初始速度、目标初始加速度作为初始条件，这些参数在每一个轮次都是相同的。接下来与环境交互进行状态转移后开始训练，训练后保存模型并进行测试，表 3-2 为 PER 算法的超参数，其中优先级参数 α 表明了对于优先经验回放的利用程度，$\alpha = 0.1$ 是一个容易得到更好结果的参数设置。权值参数 β 的设置与 α 有关。$\epsilon-$贪心参数 ϵ 表明了对于探索和利用的平衡程度。有研究者[13]证明了经验池的大小会影响结果到学习率，因此本章测试并调整了经验池容量，目前的参数设置为结果较为良好的值。

表 3-2 基于 PER 的末制导律设计方案参数表

参数	值
经验池容量	10000
网络参数更新间隔	1000
优先级参数 α	0.1
权值参数 β	0.1
$\epsilon-$贪心参数 ϵ	0.1
折扣率 γ	0.9
小批量采样数目	32

3.4.2 实验结果与分析

为了将 PER 与 DQN 进行对比，本次在仿真环境中训练了 120 轮，之后利用训练好的值函数分别进行了 100 组测试，并选择以脱靶量作为衡量方案有效性的标准，最终结果如表 3-3、3-4 所示。机动方式共有 4 种，分别是向上机动、向下机动、正弦上下机动以及在以上三种机动方式中每轮随机选取一个作为机动方式在实际测试中，训练和测试中选取的机动方式是相同的。

表 3-3 DQN 与 PER 制导律方法脱靶量统计表

制导律方法	机动方式	脱靶量(m)			
		最小值	最大值	平均值	标准差
DQN	向上机动	0.08	6.13	1.54	1.24
	向下机动	0.04	3.13	1.96	0.68
	正弦机动	0.04	5.95	1.97	1.45
	随机机动	0.05	5.89	1.89	1.35
PER	向上机动	0.06	1.2	0.58	0.27
	向下机动	0.05	0.67	0.33	0.4
	正弦机动	0.04	0.84	0.34	0.19
	随机机动	0.05	1.24	0.49	0.31

表 3-4 为脱靶量分布表，可以看到 PER 的脱靶量全部分布在 [0,3] 区间内，而测试中设置为脱靶量 3m 内视为拦截成功给予奖励，这证明基于 PER 的末制导律设计方案在绝大多数情况下都能拦截成功。

表 3-4 DQN 与 PER 制导律方法脱靶量分布表

制导律方法	机动方式	脱靶量(m)			
		[0,0.5]	[0.5,3]	[3,5]	[5,]
DQN	向上机动	19.00%	68.00%	11.00%	2.00%
	向下机动	6.00%	92.00%	2.00%	0.00%
	正弦机动	19.00%	58.00%	19.00%	4.00%
	随机机动	14.00%	67.00%	16.00%	3.00%
PER	向上机动	42.00%	58.00%	0.00%	0.00%
	向下机动	90.00%	10.00%	0.00%	0.00%
	正弦机动	78.00%	22.00%	0.00%	0.00%
	随机机动	57.00%	43.00%	0.00%	0.00%

为了更直观地看出脱靶量的分布状况，本章绘制了下图 3-2。

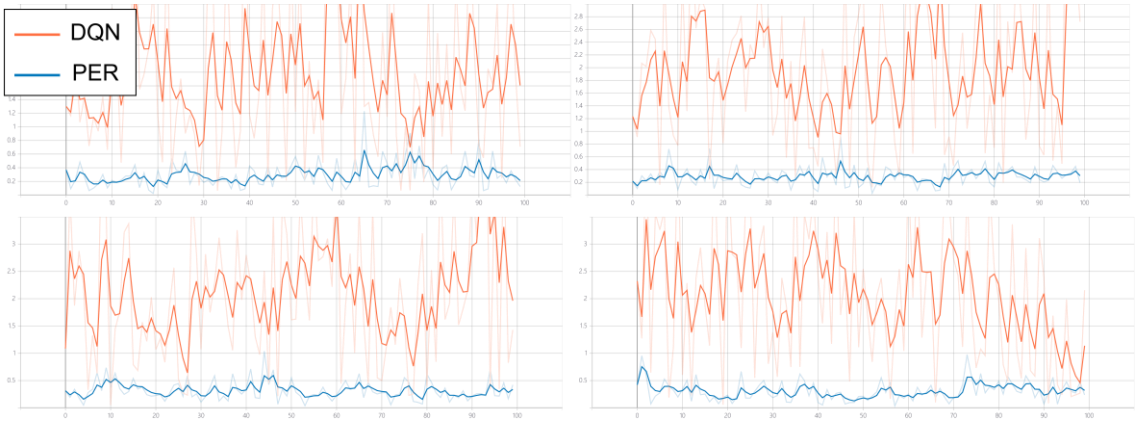

图 3-2　DQN 与 PER 的脱靶量分布

上图 3-2 为某一次测试中 DQN 与 PER 脱靶量的分布图，纵坐标代表脱靶量，横坐标代表测试的轮次，图中左上、右上、左下、右下分别代表目标采取向上机动方式、向下机动方式、正弦上下机动方式、随机机动方式时的脱靶量。可以很容易看到，整体上 PER 的脱靶量是明显要比 DQN 小的。由于 PER 进行了采样优先级排序，优先级高的样本更容易被采样到，因此在解决末制导这种稀疏奖励问题时，PER 算法可以更快地找到有效的策略，从而降低脱靶量，使其效果优于 DQN。因此 PER 是解决稀疏奖励下末制导问题的一种有效解法。

3.5 本章小结

本章首先介绍了深度 Q 网络（DQN）算法的基本原理，然后详细介绍了优先经验回放（PER）算法的原理和具体实现方法，该算法是一种改进的深度 Q 网络算法，其主要思想是为经验回放机制加入优先级的概念，使得重要性较高的样本被更频繁地训练，提高样本利用效率，从而加速学习过程。

接着，本章提出了一种基于优先经验回放算法的强化学习末制导律设计方案，并在仿真环境下进行了实验验证,通过脱靶量及其分布验证了该设计方案的有效性。在解决稀疏奖励下的末制导问题时，其效果优于 DQN。

第 4 章 基于事后经验回放（HER）的强化学习末制导律设计

本章首先将要介绍事后经验回放的原理。随后将基于事后经验回放算法进行末制导律设计。最后将在仿真环境中对于本章设计的事后经验回放末制导律进行测试，并将效果和 DQN 和 PER 进行对比。

4.1 事后经验回放（HER）

事后经验回放（Hindsight Experience Replay, HER）[4]作为解决稀疏回报问题的一个经典方法，它也采取了经验回放的设计，但不同于第 2 章提到的 DQN、PER，因为稀疏回报问题本身并没有一个很好的奖励或者 TD 误差，HER 基于 goal-conditioned 强化学习框架，将值函数从状态空间扩展到了目标空间，transition 也扩展成为了五元组 (s,a,r,s',g)。

其核心思想为从失败中学习，尽管没有成功，但可以从过程中学到一些不一样的东西[4]，它假设目标空间 $\mathcal{G}$ 与状态空间 $\mathcal{S}$ 存在着一一映射，将某个状态选作目标，目标空间是状态空间的子集。

该方法的稀疏奖励设置为

$$y = \mathrm{r_g}(\mathrm{s_t}, \mathrm{a_t}, \mathrm{g}) = \begin{cases} 0, & if\ |s_t - g| < \delta \\ -1, & otherwise \end{cases} \tag{4-1}$$

由于奖励是二值稀疏的，因此大部分的状态对于我们的目标来说都是失败的，为了提高样本利用率避免浪费，采样的每一条轨迹都应该不止用一个目标来衡量，也就是多重目标（Multi-goal）。

其核心思想为：利用当前策略在环境中交互获得一条轨迹（绝大多数情况下的奖励都是失败），然后将 $(s_t||g, a_t, r_t, s_{t+1}||g)$ 存储在经验池中，再挑选一些曾经已经达到过的“其它目标”对于这条失败轨迹中的目标和奖励作出修改，也存储在经验池中，假装这个“其它目标”本来就是当时想要完成的那个目标，这样经验池中奖励为正值的 transition 数目就因此增多。之所以可以这样做，是因为目标只会影响智能体的动作而不会影响环境自身的动态变化规律。

关于如何选取用于回放的额外的目标，HER 提出了 4 种不同的方法：[4]

1) final：将每个 episode 的最终状态 s_T 所对应的目标 $m(s_T)$作为额外目标

2) future：将同一 episode 中 k 个稍后才会被回放到的 transition 中的状态所对应的目标作为额外目标
3) episode：从当前 episode 中随机选取 k 个已经回放过的 transition 中的状态作为额外目标
4) random：从全局训练过程中遇到过的状态中随机选取 k 个作为额外目标

4.2 基于 Hindsight 思想的强化学习末制导律设计

由于稀疏场景下末制导问题的目标（goal）选取相当困难，因此本文只是采用了 Hindsight 这种经典的思想来处理末制导问题。本文利用了 Hindsight 的思想去解决末制导问题，但只是将目标选为让导弹击中目标飞行物，本文选取的目标是一个恒定不变的抽象概念，并没有将值函数扩展到目标空间；而前面介绍的 HER 算法是默认将目标空间设置为状态空间的一个子集，目标就是某个状态，然而对于末制导问题，状态具备随机性，在打中目标飞行物之前并没有一些必经的状态，因此对于目标的选取是困难的。

本文对于 Hindsight 思想的利用方式为：将本轮没有成功击中的导弹轨迹和过去所有幕中没有成功被击中的目标轨迹进行匹配，假如它们之间的脱靶量小于设定的 δ，那么就利用当前时刻匹配成功的这对弹目的位置坐标、速度大小方向来计算当前时刻的一个“虚假的”状态，并利用导弹轨迹曾经在这个时间步的动作，进行状态转移到下一个状态，最后把当前状态、动作、下一个状态、正奖励形成的四元组存放在经验池里，这样就通过 Hindsight 思想扩充了经验池里奖励为正的 transition 的数量，降低稀疏奖励对于训练的影响。该算法详细流程如下：

算法 2：基于 Hindsight 思想的强化学习末制导律设计[4]

输入：状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣率 γ，小批次量 k，步长 η，经验池中样本数目 N，正奖励 R

输出：离轨策略训练得到的网络参数

初始化环境，指定弹目初始位置、速度、导弹弹道倾角、目标机动方式

初始化经验池 R

for episode = 1, M **do**

　随机初始化一个初始状态 s_0

　for t = 0, T - 1 **do**

　　利用 $\epsilon-$贪心选择导航比 N_1、N_2 的组合 $A_t \sim \pi_\theta(S_t)$

将 A_t 映射称为真实的动作 A'_t

观测 S_{t+1}, R_{t+1}

在 R 中存储 transition $(S_t, A'_t, R_t, S_{t+1})$（标准经验回放）

利用离轨策略如 DQN 进行训练

end for

if 在刚才所有的时间步内，导弹从未击中过目标

将刚才的时间步形成的“失败”轨迹 与 过去的 episode 内所有的目标飞行物轨迹进行匹配，计算它们之间的脱靶量，如果在某一时刻小于 δ，则匹配成功，找到所有匹配成功的弹目轨迹

对于所有匹配成功的弹目轨迹：

利用脱靶量小于 δ 那一时刻弹目的位置坐标、速度、速度倾角计算此时的状态 S，并利用导弹当时的动作 A 进行状态转移得到下一个状态 S'

在 R 中存储 transition $(\mathrm{S}, \mathrm{A}, R, \mathrm{S}')$（事后经验回放）

end for

本文对于 Hindsight 思想的具体实现细节为：将当前失败的导弹轨迹和过去所有幕中失败的目标轨迹进行匹配，如果它们的脱靶量小于 δ，那么就计算当前时刻的状态、下一时刻的状态，并将（当前状态，动作，下一状态，正奖励）存储在经验池中，由于不能“轻信”这个匹配出来的虚假信息，此处的正奖励在数值上的设置是 0.1，小于常规经验回放的奖励值 1 。

之后只需要进行常规的 DQN，执行状态转移、将 transition 存储到经验池、从中随机抽样进行训练更新 Q 网络。

4.3 仿真实验

本节将介绍仿真实验具体部署细节，如超参数设置等，并将本章基于 Hindsight 思想的末制导律设计方案与 DQN 和优先经验回放进行对比，测试该稀疏奖励下该方案的有效性。

4.3.1 实验环境部署

表 4-1 和 4-2 中分别列出了初始化环境的参数值和算法的参数，表 4-1 中的参数设置和第 3 章相同，且在每一轮次都是相同的设置。

表 4-1 末制导仿真环境初始参数表

参数	值	单位
导弹初始位置	(0,11000)	m
导弹初始速度倾角	-25	degrees
导弹最大过载（加速度）	20g	m/s^2
导弹初始速度	600	m/s
目标初始位置	(6000,8000)	m
目标初始速度	300	m/s
目标初始加速度	2g	m/s^2

表 4-2 Hinsight 末制导律设计方案参数表

参数	值
经验池容量	10000
网络参数更新间隔	1000
$\epsilon-$贪心参数 ϵ	0.1
折扣率 γ	0.9
正奖励值	0.1
小批量采样数目	32

本章对于仿真环境进行了一些修改，在第 3 章的环境中，曾设置了一些标志位用于记录导弹是否在连续的时间步里一直偏离目标，如果持续偏离，则直接变成终止状态，进入下一个 episode；而在本章中虽然也设置了这些标志位，但它们不再决定最终状态，只有成功击中目标或者到达最终时间步时才会变成终止状态，这些标志位用于决定是否存储当前 transition，即虽然在持续进行状态转移，但不再将其存储在经验池中，这样做是为了收集到完整的“失败”轨迹，而又使得标准经验回放过程中不会往经验池里存储更多的值，因此即使在修改之后，两个环境下的 DQN 和 PER 效果也是相同的，方便进行效果的对比。

此外，在非随机机动方式下，在每一幕中，由于设置的目标飞行物的初始位置坐标、速度大小、倾角、机动方式都是相同的，目标飞行物在每一幕的轨迹也会是相同的。假如导弹在当前幕内没有击中目标飞行物，那么和过往所有幕的目标飞行物轨迹也不会匹配成功，因此本章仅选取在随机机动方式下进行测试。

4.3.2 实验结果与分析

机动方式均为随机机动方式，由于机动方式较为单一，为了使数据更加丰满，相对于第 3 章的测试，本章增加了多组数据用于对比。

表 4-3 三种制导律方法脱靶量统计表

制导律方法	数据组别	脱靶量(m)			
		最小值	最大值	平均值	标准差
DQN	第一组	0.02	8.2	1.45	1.44
	第二组	0.09	6.83	1.51	1.46
	第三组	0.08	5.38	1.54	1.27
	第四组	0.07	6.72	1.77	1.58
PER	第一组	0.05	1.5	0.51	0.34
	第二组	0.03	2.09	0.55	0.42
	第三组	0.05	1.24	0.49	0.31
	第四组	0.08	1.77	0.55	0.34
HER	第一组	0.03	1.24	0.37	0.23
	第二组	0.03	0.85	0.37	0.2
	第三组	0.03	1.33	0.39	0.22
	第四组	0.01	1.0	0.31	0.19

表 4-4 三种制导律方法脱靶量分布表

制导律方法	数据组别	脱靶量(m)			
		[0,0.5]	[0.5,3]	[3,5]	[5,]
DQN	第一组	27.00%	63.00%	6.00%	4.00%
	第二组	22.00%	62.00%	12.00%	4.00%
	第三组	13.00%	72.00%	12.00%	3.00%
	第四组	20.00%	63.00%	11.00%	0.00%
PER	第一组	57.00%	43.00%	0.00%	0.00%
	第二组	54.00%	46.00%	0.00%	0.00%
	第三组	57.00%	43.00%	0.00%	0.00%
	第四组	55.00%	45.00%	0.00%	0.00%
HER	第一组	77.00%	23.00%	0.00%	0.00%
	第二组	76.00%	24.00%	0.00%	0.00%
	第三组	79.00%	21.00%	0.00%	0.00%
	第四组	82.00%	18.00%	0.00%	0.00%

为了更直观地体现脱靶量的分布状况，本章绘制了下图 4-1 。

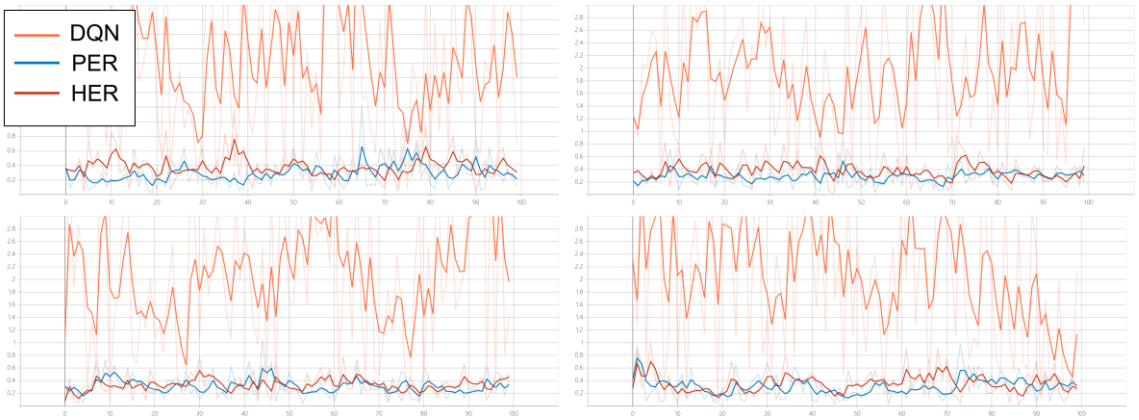

图 4-1 DQN、PER 与 HER 的脱靶量分布

上图 4-1 为某一次测试中 DQN、PER 与 HER 脱靶量的分布图，纵坐标代表脱靶量，横坐标代表测试的轮次，图中左上、右上、左下、右下分别代表目标采取向上机动方式、向下机动方式、正弦上下机动方式、随机机动方式时的脱靶量。可以看到 PER 与 HER 效果接近，几乎分布在相同的区间。

无论是基于优先经验回放还是基于事后经验回放的末制导律设计，脱靶量都可以 100% 地分布在 3 米以内，而这正是我们在训练时可以得到奖励的脱靶量，充分证明了这两种设计方案对于解决稀疏奖励场景下末制导问题的有效性。

在某次测试的 120 轮次训练中，共匹配成功 287 对弹目轨迹，经计算“虚假”状态后存储到经验池中，而经验池中总样本数目为 67831。可以看到，基于 Hindsight 思想的末制导律设计方案在测试结果上是略优于基于优先经验回放的末制导律设计方案的，但这并不能说明该方法一定总是优于优先经验回放。事实上正奖励值参数的设置是重要的，它代表着对于 Hindsight 思想的利用程度，本章也测试了在不同正奖励值设置下 Hindsight 制导方案的效果，如下表 4-5 所示。

表 4-5 不同正奖励参数值下的脱靶量统计表

制导律方法	正奖励参数值	脱靶量(m)			
		最小值	最大值	平均值	方差
HER	0.01	1.88	5.69	3.84	0.71
	0.1	0.03	0.85	0.37	0.2
	0.3	0.19	4.55	1.49	0.66
	0.5	1.49	3.45	2.37	0.38

这说明，仅在对于 Hindsight 的利用程度合适时，该方法优于 PER。此外，该方法在收敛速度上略慢于 PER，且并非总能得到良好的结果，这应该是由于机动方式随机性太大，每一条轨迹之间的差异也很大，有时难以进行有效的匹配。

此外，本章也尝试了在 PER 上应用 Hindsight 思想，其中并没有修改任何超参数设置，由于本文设置的脱靶量输出上限为 50，该结果表现不佳。

表 4-6 PER 结合 Hindsight 思想脱靶量统计表

制导律方法	数据组别	脱靶量(m)			
		最小值	最大值	平均值	标准差
HER+PER	第一组	0.06	49.97	18.55	17.05
	第二组	0.06	49.51	15.74	15.77
	第三组	0.08	49.09	16.23	16.66
	第四组	0.06	48.03	15.44	17.24

表 4-7 PER 结合 Hindsight 思想脱靶量分布表

制导律方法	数据组别	脱靶量(m)			
		[0,0.5]	[0.5,3]	[3,5]	[5,]
HER+PER	第一组	12.20%	24.39%	1.22%	62.19%
	第二组	11.90%	25.00%	2.38%	60.71%
	第三组	18.52%	23.46%	3.70%	54.32%
	第四组	21.69%	24.10%	3.61%	50.60%

有研究者[14]做过类似的测试，当 Hindsight 和优先经验回放都被激活时，多目标学习被启动，奖励可能是有噪声的，在这种情况下，TD 误差可能是一个较差的估计方法。而从本文对于 Hindsight 思想的利用上来看，在正奖励值设置为 0.1 而不是 1 的时候才能得到良好的结果，这似乎证明了不能过于“轻信”由 hindsight 得到的 transition，因此直接计算其 TD 误差当作优先级，并非良好的做法。另外，调整经验池的大小也并没有改善结果。

4.4 本章小结

本章首先详细介绍了事后经验回放（HER）算法的原理，该算法的利用了 Hindsight 思想，从失败中学习，用新的目标去替换经验中原本想要达到的目标，提高样本利用率。

接着，本章提出了一种基于 Hindsight 思想的强化学习末制导律设计方案，通过将本轮失败的导弹轨迹和过往所有轮中失败的目标轨迹匹配，脱靶量小于某个设定好的值时，则被认为匹配成功，即可以拦截成功。利用弹目轨迹信息计算出这个虚假的拦截成功状态，将其制作成四元组存储在经验池中。最后在仿真环境下进行了实验，并与 DQN 和 PER 的效果进行了对比，在某些良好的参数下其表现优于 PER，验证了该设计方案的有效性。另外测试了将 PER 与 Hindsight 思想进行结合，其结果不尽如人意。

结 论

末制导律作为制导过程中的关键部分，随着智能空战的不断发展也逐渐引起了人们的关注。真实环境中的末制导问题常常是稀疏奖励的，难以有效地学习因此效果较差。本文针对于稀疏奖励场景下的末制导问题，设计了两种基于经验回放思想的强化学习末制导律。

本文研究成果具体如下：

（1）根据末制导问题的空气动力学特性，利用简化的二维场景对其进行强化学习马尔可夫决策过程建模，对于状态空间、动作空间、奖励函数进行了详细的设计。

（2）设计并实现基于优先经验回放算法的末制导律，并在仿真环境下进行了与深度 Q 网络的对比测试，结果显示其优于深度 Q 网络。

（3）设计并实现基于 Hinsdight 思想的末制导律，并在仿真环境下进行了与深度 Q 网络和优先经验回放的对比测试，结果显示在最优参数设置下，该方法的有效性和优先经验回放算法接近。无论是基于优先经验回放还是基于事后经验回放的末制导律设计，脱靶量都可以 100% 地分布在在训练时可以得到奖励的脱靶量之内。

本文尚存在一些不足：对于事后经验回放直接采取了固定的是否拦截成功目标而不是使用目标空间，由于目标选取困难，就无法使用 goal-conditioned 分层强化学习的方法，因此仍然需要继续探索对于稀疏奖励下末制导问题中目标如何选取的问题。

参考文献

[1] 王迪. 基于回溯思想的高效强化学习末制导律设计[D].哈尔滨工业大学, 2021.

[2] V MNIH, et al. Playing Atari with Deep Reinforcement Learning[C]. NeurIPS, 2013.

[3] T SCHAUL, et al. Prioritized Experience Replay[C]. ICLR, 2016.

[4] M ANDRYCHOWISZ, et al. Hindsight Experience Replay[C]. NeurIPS, 2017.

[5] R S SUTTON, et al. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning[J]. Artificial Intelligence, 1999, 112(1-2):181–211.

[6] T SCHAUL, et al. Universal Value Function Approximators[C]. ICML, 2015.

[7] A LEVY, et al. Learning Multi-level Hierarchies With Hindsight[C]. ICLR, 2019.

[8] T MATIISEN, et al. Teacher-Student Curriculum Learning[J]. TNNLS, 2020, 31(9):3732-3740.

[9] S SUKHBAATAR, et al. Intrinsic Motivation and Automatic Curricula via Asymmetric Self-play[C]. ICLR, 2018.

[10] C FLORENSA, et al. Automatic Goal Generation for Reinforcement Learning Agents[C]. ICML, 2018.

[11] R S SUTTON, BARTO A G. Reinforcement Learning: An Introduction[M]. US: MIT Press, 2020.

[12] J JANISCH, Let’s make a DQN: Double Learning and Prioritized Experience Replay[OL](2016-11-07)[2023-05-20]. https://jaromiru.com/2016/11/07/lets-make-a-dqn-double-learning-and-prioritized-experience-replay/.

[13] RUISHAN LIU, et al. The effects of memory replay in reinforcement learning[C]. IEEE, 2018.

[14] RENZO TAN, et al. An Inquiry into Experience Replay Sampling in Deep Reinforcement Learning[C]. Proceedings of the 28th Annual Conference of the Japanese Neural Network Society, 2018.

[15] P L BACON, et al. The Option-Critic Architecture[C]. AAAI, 2017.

[16] T D KULKARNI, et al. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation[C]. NeurIPS, 2016.

[17] O NACHUM, et al. Data-Efficient Hierarchical Reinforcement Learning [C]. NeurIPS, 2018.

[18] SIYUAN LI, et al. Hierarchical reinforcement learning with advantage-based auxiliary rewards[C]. NeurIPS, 2019.
[19] S SUKHBAATAR, et al. Learning Goal Embeddings via Self-Play for Hierarchical Reinforcement Learning[C]. NeurIPS, 2018.
[20] J. ZHANG, et al. Deep reinforcement learning with successor features for navigation across similar environments[C]. IROS, 2017.
[21] H V HASSELT, et al. Deep Reinforcement Learning with Double Q-learning[C]. AAAI, 2016.
[22] A S VEZHNEVETS, et al. Feudal networks for hierarchical reinforcement learning[C]. ICML, 2017.
[23] HAORAN XU, et al. Sparse Q-Learning: Offline Reinforcement Learning with Implicit Value Regularization[C]. NeurIPS, 2022.
[24] E S HU, et al. Planning Goals for Exploration[C]. ICLR, 2023.
[25] 杨瑞, 严江鹏, 李秀. 强化学习稀疏奖励算法研究——理论与实验[J]. 智能系统学报, 2020, 15(5): 888–899.
[26] 邱锡鹏. 神经网络与深度学习[M]. 北京：机械工业出版社. 2020.