无人机飞越冰湖

方桂安，刘梦莎，袁菁一，陈谊聪，刘玥，左宗伟，高雨飞，漆佳嬉

摘 要—本次机器人综合实验的主题是无人机飞越冰湖。针对传统无人机避障算法需要构建离线三维地图等问题，使用了基于马尔可夫决策过程的 Qlearning，Sarsa 和 Double DQN (DDQN)，分别在 Deterministic 模式和 Stochastic 模式的 4x4 以及 8x8 地图训练了对应的 RL 模型，并通过性能对比，最终成功在实机演示中控制无人机飞越冰湖。

关键词—冰湖问题，无人机，强化学习

I. 概述

UAV 中的小型多旋翼无人机因为其体积小、动作灵活、适宜在复杂环境下完成机动，从而广泛应用于各种军用以及民用领域。科学技术的发展，也使得针对无人机的导航指导与控制技术得到了更大提升。然而，在陌生环境中实时进行自主导航却是极具挑战性的问题。在结构不能确定、光线条件多变的未知三维环境中，实现自主规避障碍物仍然是亟待解决的问题。

传统的无人机避障算法需要构建离线三维地图，在全局地图的基础上以障碍点为约束，采用路径搜索算法计算出最优路径。有的避障算法虽然避免了繁杂的地图构建工作，但是需要手动调整大量的参数且机器人在避障的过程中不能利用避障经验进行自我迭代。随着机器学习的发展，研究人员将有监督学习引入无人机的避障之中，将避障看作是一个基于监督学习的分类问题，但需要对每个样本标签进行标注，这样无疑费时费力。在无人机避障过程中，如何能够充分利用无人机三维空间信息，避免环境构建工作，简化算法模型，提升无人机自主避障能力以及无人机避障效率，最终实现无人机更加安全高效地到达指定目的地仍然是目前需要解决的问题。

方桂安，20354027，(e-mail: fanggan@mail2.sysu.edu.cn)。
刘梦莎，20354091，(e-mail: liumsh6@mail2.sysu.edu.cn)。
袁菁一，20354262，(e-mail: yuanjy36@mail2.sysu.edu.cn)。
陈谊聪，20354023，(e-mail: chenyc57@mail2.sysu.edu.cn)。
刘　玥，20354229，(e-mail: liuy2236@mail2.sysu.edu.cn)。
左宗伟，20354274，(e-mail: zuozw@mail2.sysu.edu.cn)。
高雨飞，20354031，(e-mail: gaoyf37@mail2.sysu.edu.cn)。
漆佳嬉，18364076，(e-mail: qijx@mail2.sysu.edu.cn)。

随着机器学习的发展，强化学习被科研人员应用于无人机的控制之中。强化学习主要通过与外界交互来优化自身的行为，它比传统机器学习更具优势：

1) 训练前不需要对数据进行标注处理，能更有效地解决环境中存在的特殊情况；
2) 可把整个系统作为一个整体，实现端到端输入输出，从而使其中的一些模块具有更强的鲁棒性；
3) 强化学习与其他机器学习方法相比较更容易学习到一系列行为。

因此，使用强化学习算法的优势在于不依赖于传统非机器学习所需要的离线地图，以及监督学习所需要的人工标注的数据集，通过深度学习模型学习输入数据和输出动作的映射关系，使智能体具备处理高维连续空间下的决策问题的能力，避免复杂离线地图构建工作。

基于此，我们小组通过查阅资料，对多种强化学习算法就冰湖问题进行了仿真实验、对比性能，并最终将其中效果最好的 Sarsa 应用于实机飞越冰湖演示。

II. 算法介绍

A. RL

强化学习（RL）是机器学习的一个分支，agent 通过与环境交互来进行学习。这是一个以目标为导向的学习过程，agent 从其行为的结果中学习。

agent 可以根据最好的结果探索不同行为，也可以利用获得了良好回报的先前行为。如果 agent 探索不同的行为（action），则很有可能 agent 将获得较差的回报，因为所有的行为并不都是最佳的。如果 agent 仅利用已知的最佳行为，那么错过最佳行为（action）的可能性也很大。因此，需要选择合适的算法在这两者间取得平衡。

一个典型 RL 算法的步骤如下：

1. 首先，agent 通过执行行为与环境进行交互；

2. agent 执行行为并从一种状态移动到另一种状态；

3. 然后 agent 将根据其执行的行为获得回报;

4. 根据回报，agent 将知道行为是好的还是坏的;

5. 如果 action 是好的，也就是说，如果 agent 得到了积极的 reward，那么 agent 将更喜欢执行该 action，否则 agent 将尝试执行导致积极 reward 的其他 action。

B. Q-learning

强化学习包括状态 (state)、奖励 (Reward)、行为 (Action)。因为在算法中加入一个叫做 Q 表的东西，Q-Learning 因此命名。

Q 即为 $Q(s,a)$ ，就是在某一时刻的 s 状态下 ($s \in S$) ，采取动作 $a(a \in A)$ 动作能够获得收益的期望，环境会根据 agent 的动作反馈相应的回报 reward r ，所以算法的主要思想是 Q 表随着状态、动作而更新，当 Q 表更新不再发生改变时，就可以根据环境选择对应最大的值所对应的动作，从而采取动作，进行与环境交互。

Q-learning 算法的更新准则如下式所示:

$$\mathrm{Q[S,A]} = (1-\alpha)*\mathrm{Q[S,A]} + \alpha*(\mathrm{R} + \gamma * \max \mathrm{Q}\left[\mathrm{S_{next}}, :\right])$$

构建 Q 表算法步骤:

Step 1 给定参数 γ 和 reward 矩阵 R.

Step 2 令 $Q := \mathbf{0}$.

Step 3 For each episode:

3.1 随机选择一个初始的状态 s.

3.2 若未达到目标状态, 则执行以下几步:

(1) 在当前状态 s 的所有可能行为中选取一个行为 a.

(2) 利用选定的行为 a, 得到下一个状态 $\widetilde{s}$.

(3) 按照 (1.1) 计算 $Q(s,a)$.

(4) 令 $s := \widetilde{s}$.

Agent 利用上述算法从经验中进行学习. 每一个 episode 相当于一个 training session. 在一个 training session 中, agent 探索外界环境, 并接收外界环境的 reward, 直到达到目标状态. 训练的目的是要强化 agent 的 “大脑” (用 Q 表示). 训练得越多, 则 Q 被优化得更好. 当矩阵 Q 被训练强化后, agent 便很容易找到达到目标状态的最快路径了.

利用训练好的矩阵 Q, 我们可以很容易地找出一条从任意状态 s_0 出发达到目标状态的行为路径, 具体步骤如下:

1. 令当前状态 $s := s_0$.

2. 确定 a, 它满足 $Q(s,a) = \max_{\widetilde{a}}\{Q(s,\widetilde{a})\}$.

3. 令当前状态 $s := \widetilde{s}$ ($\widetilde{s}$ 表示 a 对应的下一个状态).

4. 重复执行步 2 和步 3 直到 s 成为目标状态.

C. Sarsa

Sarsa 算法的的决策部分与 Qlearning 相同，都是通过 Q 表的形式进行决策，在 Q 表中挑选值较大的动作值施加在环境中来换取奖惩，也就是根据计算出来的 Q 值来作为选取动作的依据，两者不同的是行为更新准则是有差异的。Sarsa 不会去选取他估计出来的最大 Q 估计值，而是直接选取估计出来的 Q 值。

Sarsa 算法的更新准则如下式所示:

$$\mathrm{Q[S,A]} = (1-\alpha)*\mathrm{Q[S,A]} + \alpha*(\mathrm{R} + \gamma * \mathrm{Q}\left[\mathrm{S_{next}}, \mathrm{A_{next}}\right])$$

具体的改变就是在于

$$\max \mathrm{Q}\left[\mathrm{S_{next}}, :\right] \text{ 变成了 } \mathrm{Q}\left[\mathrm{S_{next}}, \mathrm{A_{next}}\right]$$

其实就是整个算法看得更加清晰一点，即明确知道状态转移之后的步骤的结果来迭代，而不是直接用 max 值来迭代。

Sarsa 选择的是一条最安全的道路，远离陷阱。Q-learning 选择的是一条最快的道路，尽快到达出口。

这在一定程度上提高了 Q-Learning 的泛化能力，避免陷入局部最优解。

D. Double DQN (DDQN)

Q-learning 算法和 sarsa 算法都用到了 Qtable，记录着在每一个状态下，各个动作的 Q 值。但它们的缺陷是，只适用于离散的情况，遇到连续状态时 Qtable 就没办法解决。

DQN 将 Qtable 换成一个深度神经网络，允许连续状态的表示。

DQN 的更新准则为:

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

DQN 有一个显著的问题，就是 DQN 估计的 Q 值往往会偏大。这是由于 Q 值是以下一个 s’ 的 Q 值的最大值来估算的，但下一个 state 的 Q 值也是一个估算值，也依赖它的下一个 state 的 Q 值，这就导致了 Q 值往往会有偏大的的情况出现。

Double Deep Q Network(DDQN) 是对 DQN 算法的改进。

它有两个 Q 网络，因为两个 Q 网络的参数有差别，所以对于同一个动作的评估也会有少许不同。我们选取评估出来较小的值来计算目标。这样就能避免 Q 值往往会有偏大的情况发生。

Q1 网络推荐能够获得最大 Q 值的动作；Q2 网络计算这个动作在 Q2 网络中的 Q 值。

DDQN 的算法输入：迭代轮数 T ，状态特征维度 n ，动作集 A ，步长 α ，衰减因子 γ ，探索率 ϵ ，当前 Q 网络 Q ，目标 Q 网络 Q' ，批量梯度下降的样本数 m ，目标 Q 网络参数更新频率 C 。

输出: Q 网络参数

1. 随机初始化所有的状态和动作对应的价值 Q. 随机初始化当前 Q 网络的所有参数 w, 初始化目标 Q 网络 Q' 的参数 $w'=w$ 。清空经验回放的集合 D 。

2. for i from 1 to T ，进行迭代。

a) 初始化 S 为当前状态序列的第一个状态，拿到其特征向量 $\phi(S)$.

b) 在 Q 网络中使用 $\phi(S)$ 作为输入，得到 Q× 级络的所有动作对应的 Q 值输出。用 ϵ-贪婪法在当前 Q 值输出中选择对应的动作 A.

c) 在状态 S 执行当前动作 A, 得到新状态 S' 对应的特征向量 $\phi\left(S'\right)$ 和奖励 R, 是否终止状态 is_end.

d) 将 $\{\phi(S), A, R, \phi\left(S'\right), is\_end\}$ 这个五元组存入经验回放集合 D.

e) $S=S'$.

f) 从经验回放集合 D 中采样 m 个样本 $\left\{\phi\left(S_j\right), A_j, R_j, \phi\left(S_j'\right), is\_end_j\right\}, j=1,2.,,,m$ ，计算当前目标 Q 值 y_j ：

$$y_j=\begin{cases} R_j \\ R_j+\gamma Q'\left(\phi\left(S_j'\right), \arg\max_{a'} Q\left(\phi\left(S_j'\right), a, w\right), w'\right) \end{cases} \tag{1}$$

g) 使用均方差损失函数 $\frac{1}{m}\sum_{j=1}^{m}\left(y_j-Q\left(\phi\left(S_j\right), A_j, w\right)\right)^2$ ，通过神经网络的梯度反向传播来更新 Q 网络的所有参数 w

h) 如果 $i\%C=1$, 则更新目标 Q× 网络参数 $w'=w$

i) 如果 S' 是终止状态，当前轮迭代完毕，否则转到步骤 b)

III. 实验结果与分析

A. Deterministic 模式

在 Deterministic 模式下，我们画出了 Q-learning 和 Sarsa 两个算法的收敛曲线，如图 1和图 4。我们发现，两个算法都在 epoch 大约为 150-200 之间收敛，但观察具体数据可发现，与 Q-learning 算法相比，Sarsa 算法收敛较快。

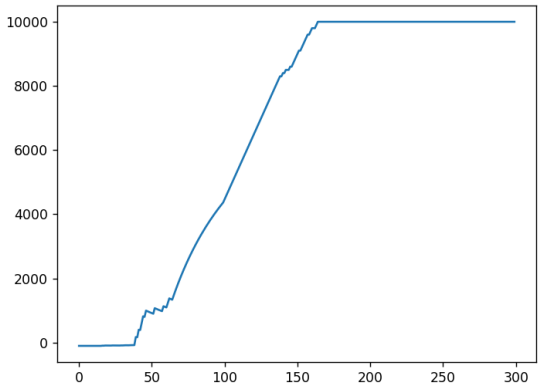

Fig. 1. Deterministic 模式 Q-learning 算法收敛曲线

此外，我们还记录了算法的运行时间。对于固定的 10^4 个 epoch，Sarsa 算法需要的时间是 8.165 秒，Q-learning 算法需要的时间是 10.271 秒。总体来看，在 Deterministic 模式下，Sarsa 算法表现较好，较快收敛。

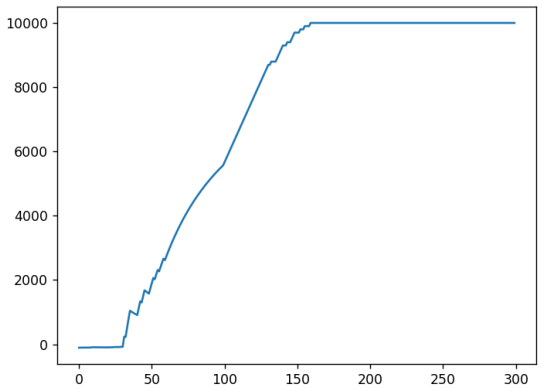

Fig. 2. Deterministic 模式 Sarsa 算法收敛曲线

B. Stochastic 模式

在 Stochastic 模式下，同样地，我们画出了两种算法的收敛曲线。但我们发现，在 Stochastic 模式下，相对

于前一种模式，算法后期振荡较大，特别是 Q-learning 算法，表现出较差的稳定性。Stochastic 模式下，agent 执行一个动作后，环境的状态转移是一个随机过程，即状态转换的最终结果不仅仅取决于动作，还受到环境的影响（例如，风、冰面打滑等），而 Deterministic 模式下，环境总是执行 agent 所采取的动作。由于 Stochastic 模式状态转移的随机性，s_{t+1} 不再只与动作和 s_t 有关，使得 agent 学习难度增大，收敛变得困难。

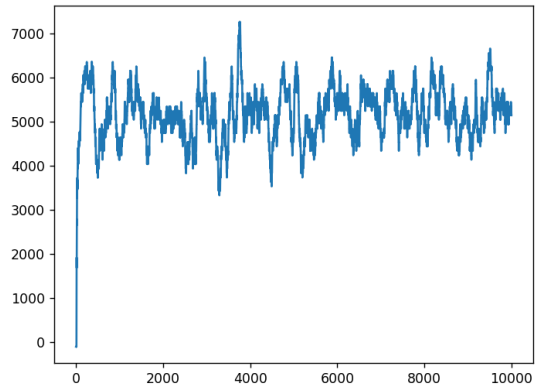

Fig. 3. Stochastic 模式 Q-learning 算法收敛曲线

与 Stochastic 模式下 Sarsa 算法对比，可以发现，Q-learning 算法能达到的 reward 低于 Sarsa，其后期稳定性也远差于 Sarsa。

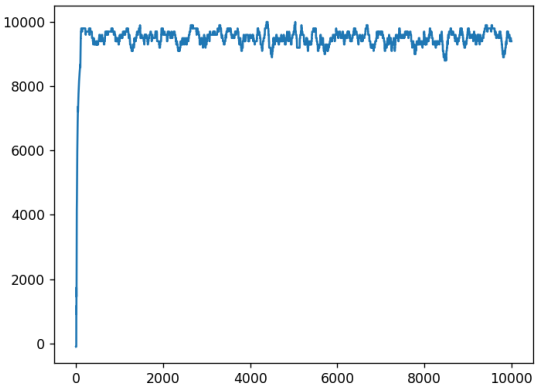

Fig. 4. Stochastic 模式 Sarsa 算法收敛曲线

C. 分析

Q-Learning 的目的是学习特定 State 下、特定 Action 的价值。是建立一个 Q-Table，以 State 为行、Action 为列，通过每个动作带来的奖赏更新 Q-Table。

Q-Learning 是 off-policy 的。异策略是指行动策略和评估策略不是一个策略。Q-Learning 中行动策略是 $\epsilon - greedy$ 策略，要更新 Q 表的策略是贪婪策略。

Sarsa 采用 Q-table 的方式存储动作值函数；而且决策部分和 Q-Learning 是一样的，也是采用 $\epsilon - greedy$ 策略。不同的地方在于 Sarsa 的更新方式是不一样的。

1.Sarsa 是 on-policy 的更新方式，它的行动策略和评估策略都是 $\epsilon - greedy$ 策略。

2.Sarsa 是先做出动作后更新。

总而言之，Q-Learning 算法，先假设下一步选取最大奖赏的动作，更新值函数。然后再通过 $\epsilon - greedy$ 策略选择动作。Sarsa 算法，先通过 $\epsilon - greedy$ 策略执行动作，然后根据所执行的动作，更新值函数。Sarsa 算法更加保守，在意每一步的决策，而 q-learning 算法的目的就是使得 Q 值最大化。因此，q-learning 有时会因为选择带来的惩罚而陷入较差的循环中，在此问题中，其收敛性和稳定性都不如 Sarsa。

IV. 问题与解决

A. 算法的性能评估

对于强化学习性能的评价一般有两种方式，一种是计算平均得分（reward），即在 agent 执行一定的步数后，记录 agent 所获得的所有奖励值并对其求平均；另一种是计算平均 Q 值，即测试性能前就确定好一定数量的状态动作对，测试时对已经确定好的状态动作对的 Q 值求平均。在本实验中，我们选择了求平均得分的方法。但从绘制的 reward-epoch 曲线中我们发现一个问题，虽然从曲线中我们可以得出随着训练轮数的增加模型性能的变化趋势，但曲线的局部平稳性欠佳，十分容易出现“抖动”，即相邻 epoch 的得分往往存在较大差距。经过分析，我们认为，强化学习每次迭代更新策略参数后，对应的训练数据分布也发生了变化，导致训练数据的分布往往与当前的策略分布有一定的差距，并且训练数据分布的更新也存在滞后性，从而使算法在某时刻下对某些状态有较好的表现而对另一些状态有较差的表现。为了解决上述问题，我们考虑到取多次实验的平均结果来绘制曲线的方法，例如每个数据点都是进行 50 次实验后取平均的结果。但出于实验条件的限制，且强化学习的训练本身耗费的时间巨大，所以我们适当减少了试验次数，采取了平均 2~3 次实验结果的方式，从而得到相对较为平滑的曲线。

B. 算法的收敛性

在默认的训练图上，QLearning、Sarsa、DDQN 都能得到比较好的结果，但是在测试的地图上，发现 DDQN 出现了三步跳水和在终点之前停止不前等问题，造成了 DDQN 的训练效果的收敛性并不是很好，通过修改 reward 也没能得到进一步提升，因此最终实机验收选取 Sarsa 和 QLearning 模型验收。

C. Stochastic 地图的实机测试

一开始采用的是在环境内设置随机偏移，这在模拟地图上没有问题。但是具体到实机测试中，由于并没有给无人机发送随机偏移运动的指令，因此随机偏移仅仅是在环境内实现，而并没有映射到现实中无人机的运动上。经过修改，在飞机通过强化学习得到运动指令 a 后，为 a 引入随机可能性，再用随机后的结果进行状态转移，完成了 Stochastic 的实机测试。

V. 实验总结

通过本次实验任务，我们小组成员第一次接触并认真研究了强化学习的相关知识，受益匪浅。

刚刚开始这项任务的时候，由于缺乏对冰湖问题的深刻分析，我们片面地寻找强化学习中的 sota 算法，实现 Q-learning 和 Sarsa 之后，我们花费了大量时间在尝试 DQN 及其变种网络中，但最终效果并不理想，甚至会出现仿真时走到终点前止步不前的状态。

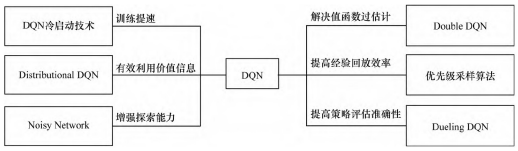

Fig. 5. DQN 算法的改进

目前，深度强化学习方法存在以下 3 个方面的局限性：

1) 深度强化学习理论支撑不够。谷歌的 DeepMind 团队于 2015 年在《自然》杂志上发表的文章虽然取得了较好的应用效果，但没有证明 DQN 的收敛性，并且到目前为止在 DQN 或其他深度强化学习方法的基础上的改进工作也没有很好地解决该问题。
2) 样本采样率低。样本采样率低使得深度强化学习方法有时在实际应用中效果不佳。导致该问题的主要原因有两个：一是完成任务需要收集大量数据；二是训练过程中利用当前数据的有用信息效率低。
3) 在连续动作空间中应用有限。目前主流的深度强化学习方法大多适用于离散动作空间，对于机器人的机械臂路径规划等连续动作空间的任务还处于初步研究阶段，理论支撑不够，因此应用十分有限。

对此，我们认为应该针对以下方面进行研究，解决问题

1) 设计有效的奖励函数。强化学习通过最大化奖励值来获得最优策略，那么策略是否最优取决于奖励函数。现阶段奖励函数由专家学者凭借专业知识设计，面对路径规划领域日益复杂多变的应用环境，不合理的奖励函数也会使得到的最优策略不合理。有学者提出元学习（Meta Learning，ML）等方式，让智能体尝试在面对环境或任务变换的情况下，从合理的策略中不断完善奖励函数。因此，设计有效的奖励函数是未来发展的热点之一。
2) 解决强化学习的探索-利用困境。一方面，探索的目的是通过不断探索新的环境信息来获得更高的奖励值，以此避免陷入局部最优解。另一方面，利用探索是指用已学习到的信息来选择奖励值最高的动作。探索-利用的困境即使得探索信息与利用信息两者之间得到平衡，目前常用的解决方法为 $\mathcal{E}$-greedy 算法。该算法的基本原理是使智能体以 $\mathcal{E}$ 为概率随机探索信息，并以 $1-\mathcal{E}$ 为概率利用信息，通过不断的学习，$\mathcal{E}$ 会不断衰减以保证后期的学习效率。$\mathcal{E}$-greedy 算法简单易实现，但随机探索效率低，因此如何解决强化学习的探索-利用困境有待进一步研究。
3) 研究强化学习方法与常规方法的结合方法。每种强化学习方法在路径规划应用中都存在自身局限性，为了弥补单一方法的不足，通过不同方法之间相互结合的优势互补可以得到一些性能更好的方法，如传统路径规划算法、图形学算法、智能仿生学算法以及强化学习算法之间的有效结合，相互取长补短后均具有一定的发展前景。

如今，基于 DQN 的深度强化学习模型已经较为完善，策略梯度方法得到了广泛应用，而 ML 领域的其他算法也被不断地应用到 DRL 算法的相关模型中。但作为机器学习的一个新兴领域，深度强化学习现仍处于发展阶段，仍有很多问题值得进一步深入研究，我们也会努力学习相关知识，争取在前沿领域里探索。