**持续学习和强化学习相结合的多智能体协同方法研究**

**作者姓名 202406XSJK079**

**指导教师姓名、职称**

**申请学位类别**

**持续学习和强化学习相结合的多智能体协同方法研究**

**作者姓名：**202406XSJK079

**一级学科：**

**二级学科（研究方向）：**

**学位类别：**

**指导教师姓名、职称：**

**学　　院：**

**提交日期：**2024年3月

**西安电子科技大学**

**硕士学位论文**

**学　号**

**密　级　 公开**

**学校代码 10701**

**分类号 TP39**

By

Supervisor: Title:

March 2024

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in

**Research on Multi-Agent Cooperative Method Based on the Combination of Continual Learning and Reinforcement Learning**

摘要

自21世纪以来，人工智能技术取得了迅猛的发展，已经渗透到人类生活的各个领域。作为人工智能的代表技术之一，强化学习能够实现自主的学习和决策，而无需依赖人类专家的判断，为众多的现实问题提供了智能化的解决方案。然而，随着任务的变化，智能体无法自动调整自身的行为决策。当顺序学习多个任务时，按照传统的强化学习算法训练的智能体会遭受灾难性遗忘的问题，即在学习新任务的过程中会逐渐忘记旧任务的知识。

解决灾难性遗忘问题的核心目标，是确保智能体在学习新任务的同时能够保留旧任务的知识，从而在多个任务上均有良好的表现。在单智能体领域，相关研究已经取得了较为成熟的成果。然而，在涉及多个智能体的复杂环境中，目前尚未找到有效的解决方案。多智能体强化学习不仅涵盖单个智能体的学习，还需要考虑多个智能体之间的相互作用。因此，如何实现多智能体的持续强化学习，已成为当前强化学习领域面临的重要挑战。

本文对多智能体强化学习进行了深入研究，并提出了有效的算法来解决上述挑战，以帮助多智能体更好地执行多种任务。本文主要完成以下工作：

（1）本文对多智能体强化学习的经典算法及其相关应用进行了深入调研，研究了持续学习的前沿算法，并对其进行了分类，为后续多智能体持续强化学习算法的提出做好了充足的准备。

（2）针对多智能体在顺序学习多任务时面临的灾难性遗忘问题，本文将持续学习中的正则化方法引入多智能体强化学习中，基于MADDPG框架提出了OwDP算法，在新任务训练阶段，通过知识蒸馏方法把旧任务的知识转移到新模型，同时引入联合PCA约束，在参数级别上限制智能体在旧任务上决策协同性的降低。实验结果表明，OwDP算法能有效缓解灾难性遗忘问题，帮助智能体保留旧任务的知识，从而实现多智能体持续强化学习。

（3）针对多智能体在顺序学习多任务时面临的灾难性遗忘问题，本文将持续学习中的经验重放方法引入多智能体强化学习中，基于MADDPG框架提出了MADDPG-OP算法，采用混合小批次度量评估样本对旧任务的代表程度，进而选择具有较高代表度的样本，构成旧任务的核心集。在新任务训练阶段，通过PER策略重放核心集中的样本，以缓解多智能体对旧任务的遗忘，保留旧知识。实验结果表明，MADDPG-OP算法能够避免智能体在旧任务中性能的不均衡，确保所有智能体都得到充分训练，从而实现多智能体持续强化学习。

（4）对本文提出的两种多智能体持续强化学习算法进行了系统的比较，分析了它们按顺序学习任务的成本和性能，并判断了各自的适用场景。

**关 键 词**：多智能体强化学习， 持续学习， 正则化， 经验重放

ABSTRACT

Since the 21st century, artificial intelligence technology has made rapid development and has penetrated into all aspects of human life. As one of the representative technologies of artificial intelligence, reinforcement learning can achieve autonomous learning and decision-making without relying on the judgment of human experts, providing intelligent solutions for many practical problems. However, as tasks change, agents cannot automatically adjust their own behavior decisions. When learning multiple tasks sequentially, agents trained using traditional reinforcement learning algorithms will suffer from the problem of catastrophic forgetting, which means they will gradually forget knowledge of old tasks while learning new ones.

The core goal of addressing the catastrophic forgetting problem is to enable agents to retain the knowledge of old tasks while acquiring new ones, thereby ensuring their excellent performance across multiple tasks. In the field of single agent, related research has achieved more mature results. However, in the complex environment involving multiple agents, no effective solution has been found. Multi-agent reinforcement learning not only involves the learning of a single agent, but also needs to consider the interaction between multiple agents. Therefore, how to achieve multi-agent continual reinforcement learning has become an important challenge in the field of reinforcement learning.

This paper makes an in-depth study of multi-agent reinforcement learning and proposes effective algorithms to solve the above challenges, in order to help multi-agent perform a variety of tasks better. The principal accomplishments of this paper are outlined below:

(1) In this paper, the classical algorithms and related applications of multi-agent reinforcement learning are investigated in depth, and the cutting-edge algorithms of continual learning are studied and classified, which provides sufficient preparation for the subsequent proposal of multi-agent continual reinforcement learning algorithms.

(2) Aiming at the catastrophic forgetting problem faced by multi-agent when learning multiple tasks sequentially, this paper introduces the regularization method in continual learning into multi-agent reinforcement learning and proposes an OwDP algorithm based on MADDPG framework. In the new task training phase, the knowledge of old tasks is transferred to the new model by knowledge distillation, and joint PCA constraint is introduced to limit the reduction of decision coordination of agents on old tasks at the parameter level. The experimental results show that the OwDP algorithm can effectively alleviate the catastrophic forgetting problem and help agents retain the knowledge of old tasks, thus realizing multi-agent continual reinforcement learning.

(3) Aiming at the catastrophic forgetting problem faced by multi-agent when learning multiple tasks sequentially, the experience replay method in continual learning is introduced into multi-agent reinforcement learning, and a MADDPG-OP algorithm is proposed based on MADDPG framework. Mixed mini batch metric is used to evaluate the degree of representation of samples to the old tasks, and then the samples with a high representation are selected to form the coreset for the old tasks. During the training phase of new tasks, the samples in the coreset are replayed by the PER strategy to alleviate the forgetting of the old tasks and retain the old knowledge. The experimental results show that the MADDPG-OP algorithm can avoid the imbalance of agent performance in the old tasks and ensure that all agents are fully trained, thus realizing multi-agent continual reinforcement learning.

(4) This paper systematically compares the two multi-agent continual reinforcement learning algorithms proposed in this paper, analyzes the cost and performance of their sequential learning tasks, and judges their respective applicable scenarios.

**Keywords**: Multi-agent reinforcement learning, Continual learning, Regularization, Experience replay

插图索引

[图2.1 智能体与环境的交互过程 11](#_Toc162185829)

[图2.2 车杆环境的Q网络结构 15](#_Toc162185830)

[图2.3 DQN算法的训练过程 16](#_Toc162185831)

[图2.4 Actor-Critic算法架构 18](#_Toc162185832)

[图2.5 DDPG算法架构 18](#_Toc162185833)

[图2.6 MADDPG算法架构 21](#_Toc162185834)

[图2.7 EWC算法训练轨迹 23](#_Toc162185835)

[图2.8 LwF算法网络结构 24](#_Toc162185836)

[图2.9 相同状态空间中的不同任务 28](#_Toc162185837)

[图2.10 OWL算法中具有多头结构的策略网络 28](#_Toc162185838)

[图3.1 OwDP算法架构 31](#_Toc162185839)

[图3.2 Actor网络结构 32](#_Toc162185840)

[图3.3 PCA维度转换 35](#_Toc162185841)

[图3.4 联合PCA约束示意图 36](#_Toc162185842)

[图3.5 OwDP算法验证环境 40](#_Toc162185843)

[图3.6 OwDP算法对比实验中持续训练智能体的平均奖励变化曲线 43](#_Toc162185844)

[图3.7 OwDP算法对比实验中智能体执行任务的平均奖励比较 44](#_Toc162185845)

[图3.8 OwDP算法消融实验中持续训练智能体的平均奖励变化曲线 47](#_Toc162185846)

[图3.9 OwDP算法消融实验中智能体执行任务的平均奖励比较 48](#_Toc162185847)

[图3.10 OwDP算法多智能体在环境中的表现 50](#_Toc162185848)

[图4.1 悬崖漫步环境 55](#_Toc162185849)

[图4.2 MADDPG-OP算法架构 58](#_Toc162185850)

[图4.3 经验重放机制 59](#_Toc162185851)

[图4.4 核心集样本的生成与使用 60](#_Toc162185852)

[图4.5 MADDPG-OP算法执行流程 63](#_Toc162185853)

[图4.6 MADDPG-OP算法对比实验中持续训练智能体的平均奖励变化曲线 66](#_Toc162185854)

[图4.7 MADDPG-OP算法对比实验中智能体执行任务的平均奖励比较 68](#_Toc162185855)

[图4.8 MADDPG-OP算法消融实验中持续训练智能体的平均奖励变化曲线 70](#_Toc162185856)

[图4.9 MADDPG-OP算法消融实验中智能体执行任务的平均奖励比较 71](#_Toc162185857)

[图4.10 MADDPG-OP算法多智能体在环境中的表现 73](#_Toc162185858)

[图4.11 持续强化学习算法训练阶段的平均奖励变化曲线 75](#_Toc162185859)

[图4.12 持续强化学习算法智能体执行任务的平均奖励比较 76](#_Toc162185860)

表格索引

[表2.1 LwF算法 25](#_Toc162185861)

[表3.1 OwDP算法 39](#_Toc162185862)

[表3.2 OwDP算法对比实验中智能体执行任务的平均奖励统计结果 45](#_Toc162185863)

[表3.3 OwDP算法消融实验设置 46](#_Toc162185864)

[表3.4 OwDP算法消融实验中智能体执行任务的平均奖励统计结果 49](#_Toc162185865)

[表4.1 MADDPG-OP算法 64](#_Toc162185866)

[表4.2 MADDPG-OP算法对比实验中智能体执行任务的平均奖励统计结果 69](#_Toc162185867)

[表4.3 MADDPG-OP算法消融实验设置 70](#_Toc162185868)

[表4.4 MADDPG-OP算法消融实验中智能体执行任务的平均奖励统计结果 72](#_Toc162185869)

[表4.5 OwDP算法和MADDPG-OP算法主要指标对比 74](#_Toc162185870)

符号对照表

|  |  |
| --- | --- |
| 符号 | 符号名称 |
|  | 优势函数 |
|  | 动作集合 |
|  | 动作 |
|  | 旧任务核心集 |
|  | 从时刻开始的累计回报 |
|  | 智能体数量 |
|  | 联合观测集合 |
|  | 联合观测 |
|  | 状态转移函数 |
|  | 动作价值函数 |
|  | 奖励函数 |
|  | 状态集合 |
|  | 状态 |
|  | 序列长度 |
|  | 时刻 |
|  | 确定性策略函数 |
|  | 状态价值函数 |
|  | 价值网络学习率 |
|  | 策略网络学习率 |
|  | 折扣因子 |
|  | 随机动作选择概率 |
|  | 策略网络参数 |
|  | 策略目标网络参数 |
|  | 价值网络参数 |
|  | 价值目标网络参数 |
|  | 目标网络学习率 |
|  | 时序差分误差 |
|  | 新旧任务重要性系数 |
|  | 优先经验重放补偿因子 |
|  | 随机性策略函数 |
|  | 最优随机性策略函数 |
|  | 梯度 |

缩略语对照表

|  |  |  |
| --- | --- | --- |
| 缩略语 | 英文全称 | 中文对照 |
| AI | Artificial Intelligence | 人工智能 |
| CF | Catastrophic Forgetting | 灾难性遗忘 |
| CL | Continual Learning | 持续学习 |
| CTDE | Centralized Training with Decentralized Execution | 中心化训练去中心化执行 |
| DDPG | Deep Deterministic Policy Gradient | 深度确定性策略梯度 |
| DPG | Deterministic Policy Gradient | 确定性策略梯度 |
| DQN | Deterministic Policy Gradien | 深度Q网络 |
| EBLL | Encoder Based Lifelong Learning | 基于编码器的终身学习 |
| EWC | Elastic Weight Consolidation | 塑性权重巩固 |
| FC | Full Connected | 全连接 |
| GAN | Generative Adversarial Network | 生成对抗网络 |
| GEM | Gradient Episodic Memory | 梯度情景记忆 |
| iCaRL | Incremental Classifier and Representation Learning | 增量分类器和表示学习 |
| IPPO | Independent Proximal Policy Optimization | 独立近端策略优化 |
| KD | Knowledge Distillation | 知识蒸馏 |
| LwF | Learning without Forgetting | 无遗忘学习 |
| MADDPG | Multi-Agent Deep Deterministic Policy Gradient | 多智能体深度确定性策略梯度 |
| MADDPG-OP | Multi-Agent Deep Deterministic Policy Gradient with Online Coreset Selection | 基于在线核心集选择的多智能体深度确定性策略梯度 |
| MPE | Multi-agent Particle Environment | 多智能体粒子环境 |
| MDP | Markov Decision Process | 马尔可夫决策过程 |
| OCS | Online Coreset Selection | 在线核心集选择 |
| Online EWC | Online Elastic Weight Consolidation | 在线塑性权重巩固 |
| OwDP | OWL with Distillation & PCA | 基于知识蒸馏和主成分分析的无冲突持续强化学习 |
| OWL | COntinual RL Without ConfLict | 无冲突持续强化学习 |
| PCA | Principal Component Analysis | 主成分分析 |
| PER | Prioritized Experience Replay | 优先经验重放 |
| PG | Policy Gradient | 策略梯度 |
| POMDP | Partially Observable Markov Decision Process | 部分可观测的马尔可夫决策过程 |
| PPO | Proximal Policy Optimization | 近端策略优化 |
| RePR | Reinforcement-Pseudo-Rehearsal | 强化伪排练 |
| RL | Reinforcement Learning | 强化学习 |
| TD | Temporal Difference | 时序差分 |
| VDN | Value-Decomposition Networks | 价值分解网络 |
| WA | Weight Aligning | 权重对齐 |
| WSN | Winning SubNetworks | 胜出子网络 |

目录

[摘要 I](#_Toc162185871)

[ABSTRACT III](#_Toc162185872)

[插图索引 V](#_Toc162185873)

[表格索引 VII](#_Toc162185874)

[符号对照表 IX](#_Toc162185875)

[缩略语对照表 XI](#_Toc162185876)

[第一章 绪论 1](#_Toc162185877)

[1.1 研究背景及意义 1](#_Toc162185878)

[1.2 国内外研究现状 2](#_Toc162185879)

[1.2.1 强化学习 2](#_Toc162185880)

[1.2.2 持续学习 5](#_Toc162185881)

[1.3 本文研究内容 7](#_Toc162185882)

[1.4 本文组织结构 8](#_Toc162185883)

[第二章 强化学习和持续学习理论基础 11](#_Toc162185884)

[2.1 强化学习 11](#_Toc162185885)

[2.1.1 马尔可夫决策过程 11](#_Toc162185886)

[2.1.2 价值函数与贝尔曼方程 12](#_Toc162185887)

[2.1.3 基于值函数的方法 13](#_Toc162185888)

[2.1.4 基于策略的方法 16](#_Toc162185889)

[2.2 多智能体强化学习 19](#_Toc162185890)

[2.2.1 部分可观测的马尔可夫决策过程 20](#_Toc162185891)

[2.2.2 多智能体深度确定性策略梯度算法 20](#_Toc162185892)

[2.3 持续学习 22](#_Toc162185893)

[2.3.1 正则化方法 22](#_Toc162185894)

[2.3.2 重放方法 26](#_Toc162185895)

[2.3.3 持续强化学习方法 27](#_Toc162185896)

[2.4 本章小结 29](#_Toc162185897)

[第三章 基于正则化的多智能体持续强化学习算法 31](#_Toc162185898)

[3.1 问题分析 31](#_Toc162185899)

[3.2 基于正则化的多智能体持续强化学习算法 31](#_Toc162185900)

[3.2.1 Online EWC约束 32](#_Toc162185901)

[3.2.2 知识蒸馏 34](#_Toc162185902)

[3.2.3 构建联合PCA约束 35](#_Toc162185903)

[3.2.4 算法流程 38](#_Toc162185904)

[3.3 实验及结果分析 40](#_Toc162185905)

[3.3.1 实验环境 40](#_Toc162185906)

[3.3.2 实验设置 41](#_Toc162185907)

[3.3.3 算法有效性检验 41](#_Toc162185908)

[3.3.4 消融实验 46](#_Toc162185909)

[3.3.5 行为分析 49](#_Toc162185910)

[3.4 本章小结 50](#_Toc162185911)

[第四章 基于经验重放的多智能体持续强化学习算法 51](#_Toc162185912)

[4.1 问题分析 51](#_Toc162185913)

[4.2 基于经验重放的多智能体持续强化学习算法 51](#_Toc162185914)

[4.2.1 混合小批次度量 51](#_Toc162185915)

[4.2.2 优先经验重放 55](#_Toc162185916)

[4.2.3 算法网络更新 57](#_Toc162185917)

[4.2.4 算法流程 61](#_Toc162185918)

[4.3 实验及结果分析 65](#_Toc162185919)

[4.3.1 算法有效性检验 65](#_Toc162185920)

[4.3.2 消融实验 69](#_Toc162185921)

[4.3.3 行为分析 73](#_Toc162185922)

[4.4 持续强化学习算法对比 73](#_Toc162185923)

[4.5 本章小结 77](#_Toc162185924)

[第五章 总结与展望 79](#_Toc162185925)

[5.1 总结 79](#_Toc162185926)

[5.2 展望 79](#_Toc162185927)

[参考文献 81](#_Toc162185928)

[作者简介 87](#_Toc162185930)

# 第一章 绪论

## 研究背景及意义

人工智能（Artificial Intelligence，AI）利用计算机、数学和生物学等学科知识，模拟人类思维过程，使机器能够实现学习、决策、推理和规划等多种智能行为。近年来，随着计算能力、大数据和机器学习算法的迅速发展，人工智能取得了重大突破，引起了社会的广泛关注和研究兴趣。人工智能在金融[1]、游戏[2]、人机交互[3]、通信[4]等领域有着重要的应用，已经成为我们日常生活的一部分，彻底改变了人类的生活方式和生产方式。2024年2月，OpenAI公司研发的Sora文生视频模型[5]的问世更是产生了全球性的影响。

以深度学习为代表的人工智能技术在图像分类[6]、自然语言处理[7]和语音识别[8]领域取得了显著成就，显示出深度学习具备强大的表示和感知能力。然而，在许多场景中，人工智能系统面临的任务更加复杂，仅仅具备表示和感知能力是不足以满足这些任务需求的，还需要具备一定的决策能力。而强化学习是一种能够提供这种决策能力的方法[9]。与深度学习不同，强化学习通过智能体与环境交互所获得的数据进行训练，使智能体能够自主学习到较优的策略，进而高效地解决现实生活和生产中所面临的序贯决策问题。智能体从环境中获得反馈来改进和优化自身的行为，类似于人类在学习过程中通过奖励和惩罚来判断行为的好坏，这种学习模式表现出更加高级的人工智能形式。如今，强化学习已经成为人工智能领域的重要部分，并且取得了丰富的成果。在游戏领域，网易伏羲实验室和天津大学联合研发的游戏测试框架Wuji融合了深度强化学习算法和进化算法，实现了游戏的自动化测试，展示了人工智能和强化学习的巨大潜力[10]。在聊天机器人领域，DeepMind团队研发的Sparrow系统具有人类一样的对话能力，并能显著减少不安全和不恰当的回复，为人类带来更加高质量的交流体验[11]。此外，强化学习在医疗服务[12]、机器人控制[13]等领域也有广泛的应用。这些标志性成果不仅推动了强化学习和人工智能算法的发展，也为解决现实生活中的复杂问题提供了新的思路，极大地便利了人们的生活。

强化学习的智能体能够自主地学习和决策，而无需依赖人类专家的判断，这为更多的现实问题提供了智能化的解决方案。然而，生活中会存在一些更加复杂的任务，例如足球比赛[14]、多人游戏[15]和无人驾驶[16]等，这些任务无法由单个智能体单独完成，而需要智能体之间相互合作。在军事领域，无人机在战争中扮演了侦察、预警、电子干扰[17]等角色，并取得了惊人的效果，充分证明了无人机作战的军事价值。随着无人机性能和续航能力的不断发展，使用无人机群执行联合任务成为可能，而无人机群的协同合作也成为一个重要挑战。以上场景都可以建模为多智能体系统。在自然界中，蚂蚁社会有着典型的群体协作关系：蚁后负责繁殖后代，工蚁负责维持蚁群的正常运转，兵蚁负责抵御外敌，它们各司其职、相互配合、相互协调，实现了蚁群的生存和繁衍。类似于蚁群，如何实现多个智能体分工合作、信息共享[18]和协调行动[19]，成为当前多智能体系统的重要研究内容。多智能体系统比单智能体情况更复杂，因为智能体在与环境交互的同时，也在和其他智能体直接或间接地交互。因此，多智能体系统需要解决协作、竞争与合作平衡[20]和非平稳性[21]等问题，不能简单地将单智能体强化学习的算法应用到多智能体中。深度学习凭借其在高维问题中出色的特征学习和处理能力，在图像识别和目标检测[22]等领域取得了许多成功应用。在强化学习中引入深度学习已成为了必然趋势，深度强化学习将深度神经网络强大的表达能力和强化学习的探索试错能力相结合，为解决多智能体系统存在的问题提供了重要途经，因此深度强化学习方法的研究也成为了热点之一。

采用强化学习算法训练的多智能体能够较好地完成单个任务。当面临多种任务时，为每个任务训练一个单独的多智能体系统是相当费时且不经济的。相反，只使用一个多智能体系统来执行这些任务，可能会导致灾难性遗忘（Catastrophic Forgetting, CF）[23]的发生，即在学习新任务后，多智能体可能会忘记如何执行旧任务。一个有效的解决方案是引入持续学习（Continual Learning, CL）方法，允许多智能体在学习新任务的同时保留旧任务的知识，从而使得多智能体能够按照顺序学习各个任务，并在多个任务上均具有良好的表现。

解决多智能体执行多任务时灾难性遗忘问题的关键，在于将持续学习和强化学习相结合。目前，有许多研究员提出了有效的持续强化学习算法，但这些算法主要集中在单智能体领域，而对于多智能体强化学习，尚未出现被广泛认可的持续学习算法。与单智能体相比，多智能体的持续强化学习面临更高的数据维度和更加不稳定的训练环境等挑战[24]。本文旨在基于多智能体强化学习算法，深入研究多智能体在顺序学习多任务时所面临的灾难性遗忘问题，目标是提出有效的算法来解决该问题，以帮助多智能体更好地执行多种任务。

## 国内外研究现状

### 强化学习

在1956年，达特茅斯学院会议的举行，正式确立了人工智能领域的研究。经过几十年的发展，人工智能取得了巨大的进展。强化学习是人工智能的核心技术之一，它能够让智能体通过与环境的交互来学习和改进自己的行为，这是实现人工智能的重要一步。

心理学认为，人类的学习过程是在环境奖励的刺激下进行的。在这个过程中，个体能够逐渐形成对当前反应好坏程度的记忆，进而学会如何最大化获得的奖励[25]。强化学习的思想正是基于上述理念，强化学习的智能体通过与环境不断交互，学会最大化环境的预期奖励，从而完成预设的任务。强化学习的研究最早可追溯于1954年，Minsky首次提出“强化学习”的概念[26]。马尔可夫决策过程（Markov Decision Process, MDP）是强化学习的基础，它基于一个假设：环境是一个通过状态来描述的系统，通过执行动作可以控制系统的状态。在该系统中，智能体的下一个状态只取决于当前状态和当前状态下选择的动作，而不会受过去的状态和动作的影响。这种性质的存在可以极大地简化运算，只利用当前的状态和动作信息即可预测未来的状态。20世纪50年代，美国数学家贝尔曼提出贝尔曼方程[27]。基于这一方程，研究者们使用MDP对离散化的最优控制问题进行了形式化的描述，为现代强化学习的发展奠定了坚实的基础。

根据状态转移概率是否已知，强化学习算法可分为两类：基于模型的强化学习和无模型的强化学习[28]。基于模型的强化学习指智能体所处的环境已知，那么智能体无需与环境交互，而直接使用动态规划算法[29]即可求出最优策略。但在大多数场景下，环境都是未知的，此时便无法使用动态规划算法求解。在这种情况下，需要智能体主动去探索环境，通过采样的数据来学习最优策略，这种学习方法称为无模型的强化学习，它不需预先了解环境的状态转移概率。时序差分（Temporal Difference, TD）算法[30]是一种常见的无模型的强化学习算法，它使用与环境交互获得的大量数据，利用贝尔曼方程，通过比较当前状态与后续状态的价值估计值的差异来更新价值函数，从而逐步优化智能体的策略。

强化学习的目标是最大化智能体获得的环境奖励。在未知的环境中，无法直接求出智能体的总奖励，因此，无模型强化学习的目标在于寻找一个最优策略，旨在最大化智能体的预期总奖励。实现这一目标有两种基本手段：一是计算所有状态下每个动作的预期奖励，智能体根据当前状态选择预期奖励最大的动作去执行，这种方法被称为基于值函数的强化学习；二是直接学习一个策略函数，该函数可以根据每个状态输出最优动作，这种方法被称为基于策略的强化学习。

基于值函数方法的核心在于学习最优的动作价值函数，而TD算法是优化动作价值函数的重要方法之一。1989年，Watkins提出基于TD算法的Q-learning算法[31]，该算法维护一个Q表，用来存储当前策略下所有状态动作对的价值，智能体选择当前状态Q值最大的动作去执行，这是早期强化学习的一个重要突破。早期的强化学习方法通常使用表格来存储数据。然而，这种做法只适用于状态和动作都是离散，并且空间都较小的情况，当状态和动作数量非常大时，这种存储方式便不再适用。

为了解决连续状态和动作下的强化学习问题，不少研究员提出将深度学习和强化学习融合，形成深度强化学习。2013年，Mnih等人提出了深度Q网络（Deep Q-Network, DQN）[32]，用神经网络拟合动作价值函数Q，并在Atari游戏中得到验证，由此强化学习开启了新的十年。为了解决传统DQN算法对Q值过高估计的问题，2016年，Hasselt等人提出Double DQN算法[33]，使用两套独立的网络，一套网络选取最优动作，另一套网络计算该动作的价值。Dueling DQN[34]是DQN算法的另一种改进算法，它将动作价值函数Q分解为状态价值函数V和优势函数A，从而有效地区分了状态值和动作值的重要性。同时，该算法在每次更新网络时，都会更新状态价值函数V，这也会间接地促进其他动作Q值的更新。相比于DQN算法，这种更新策略能够更加快速、准确地学习Q值函数。

基于策略方法的核心在于学习最优的策略函数。1999年，Sutton提出策略梯度（Policy Gradient, PG）算法[35]，通过将目标函数对策略求导得到梯度，再用梯度上升的方法最大化策略在环境中的期望回报。在计算策略梯度时需要用到动作价值函数Q，REINFORCE算法[36]提供了一种通过蒙特卡洛采样来估算Q值的方法。除了用蒙特卡洛方法估计Q值，还可以用神经网络拟合Q值，这正是Actor-Critic算法[37]所做的。该算法包含两个部分：Actor和Critic。Actor是策略函数，其任务是与环境交互，并且在Critic的指导下使用策略梯度方法更新自身策略；Critic是价值函数，利用Actor与环境交互收集的数据学习更准确的价值函数，并指导Actor进行策略更新。2015年，Lillicrap提出深度确定性策略梯度（Deep Deterministic Policy Gradient, DDPG）算法[38]，该算法结合了深度神经网络，能够有效地处理高维、连续动作空间的问题。同时，为了提高训练的稳定性和鲁棒性，该算法引入了经验回放和目标网络的机制。2017年，Schulman提出近端策略优化（Proximal Policy Optimization, PPO）算法[39]，核心思想是在训练过程中保持策略更新的幅度有限，以避免过大的策略更新引起的不稳定性。

如果环境中存在其他智能体，那么任务上升为多智能体强化学习。在这种情况下，多个智能体同时在环境中行动和交互，并且每个智能体自身的策略也在不断更新。因此，对单个智能体来说，即使在相同的状态执行相同的动作，其状态转移和得到的环境奖励的分布也可能不同。针对这种复杂的强化学习问题，主要有3种思路：完全中心化方法、完全去中心化方法和中心化训练去中心化执行（Centralized Training with Decentralized Execution, CTDE）方法。

完全中心化：将多智能体系统看成一个超级智能体，超级智能体的状态为多智能体的联合状态，动作为多智能体的联合动作。对超级智能体来说，环境依然是稳态的，但这种方法可能会导致维度灾难。2016年，Sukhbaatar等人提出CommNet算法[40]，每个智能体能够接收其他所有智能体的通信内容作为其输入的一部分，增强了智能体之间的信息传递。

完全去中心化：将多智能体系统看成多个独立的智能体，各智能体独立学习，互不影响，这种方法不会导致维度灾难，但可能因为条件过于理想化而使模型无法达到收敛。2020年，Christian等人提出IPPO（Independent Proximal Policy Optimization）算法[41]，将PPO算法扩展到多智能体强化学习中。IPPO算法的核心思想是每个智能体使用独立的PPO算法进行决策，忽略其他智能体的存在。这样做的好处是降低了智能体的输入维度，极大地提高了训练速度。然而，IPPO算法可能无法实现全局最优，因为它忽略了智能体之间的相互协作。

中心化训练去中心化执行：在训练阶段，利用一些单个智能体无法获得的整体信息以提升训练效果；而在执行阶段，不再依赖整体信息，而是根据智能体自身的策略执行动作。CTDE算法主要分为两种：一种是基于值函数的方法，例如价值分解网络（Value-Decomposition Networks, VDN）算法[42]，将所有智能体的联合动作价值函数分解为各智能体的动作价值函数之和，解耦了多智能体系统的复杂关系；QMIX算法[43]是VDN算法的改进版本，通过使用神经网络来学习整体的联合动作价值函数如何分解为单智能体的动作价值函数，而不是简单地将单智能体的动作价值函数累加。另一种是基于Actor-Critic的方法，例如多智能体深度确定性策略梯度（Multi-Agent Deep Deterministic Policy Gradient, MADDPG）算法[44]，它是对DDPG算法的扩展，使其适用于多智能体强化学习的场景。在MADDPG算法中，每个智能体都独立实现了一个DDPG算法。在训练阶段，Critic网络将智能体系统的联合状态和联合动作作为输入，并为Actor网络提供指导；在执行阶段，每个智能体的Actor网络独立执行。

### 持续学习

随着深度学习的迅速发展和大模型的出现，人工智能在众多场景中取得广泛成功，在某些情况下甚至超越了人类的水平。然而，在面临序列任务时，模型在经过训练后的表现可能无法达到预期，出现灾难性遗忘问题，即忘记如何执行先前的任务。解决这个问题的一个方法，就是引入持续学习算法，以便模型在学习新任务时能保留旧任务的知识。这样，模型就能灵活地适应不断变化的任务。

持续学习涉及面临一系列按顺序到来的任务，其目标是从这些任务中学习和积累知识，从而逐步扩充和提升模型的能力。与多任务学习[45]不同，持续学习的任务是按顺序到达的，也就是说每个时刻只有一个任务可见，在后续学习阶段中，已经学习过任务不会再次出现。人类认知就是持续学习的例证，因为人类在一生中能够不断学习大量的知识。例如，人类在儿童时期学会了母语，即使长大后再学习英语等其他语言，也不会忘记自己的母语。事实上，母语可能会进一步帮助学习其他语言。因为在学习新知识时，人类可以审视旧知识，这有助于加快对新知识的学习速度。新知识不会对旧知识造成干扰，这种能力使人类能够不断整合知识，扩充自己的认知。持续学习与人类学习具有很大的相似性，因此，它被认为是实现通用人工智能的重要一步[46]。

持续学习方法可以分为类增量学习和任务增量学习[47]。类增量学习指在学习过程中逐渐添加新的类别，而任务增量学习指在学习过程中逐渐添加新的任务，每个任务可以包含多个类别。持续学习的核心在于平衡可塑性和稳定性，可塑性指的是整合新知识的能力，而稳定性指的是保留先前知识的能力。然而，学习新任务可能会导致灾难性遗忘，即旧任务的知识部分或完全丧失。解决这个问题是非常有挑战性的，也是当前持续学习方法研究的重点之一。研究人员已经提出了许多持续学习方法，按照避免灾难性遗忘策略的不同，可以将这些方法分为重放方法、正则化方法和参数隔离方法。

重放方法采用两种方式处理样本：一种是直接存储原始样本，另一种是使用生成性模型生成伪样本。在学习新任务时，通过重放先前的任务样本来减轻遗忘。iCaRL（Incremental Classifier and Representation Learning）算法[48]专注于类增量学习，其核心思想是选择并存储每个类别中最接近特征均值的样本。在训练过程中，iCaRL算法通过最小化新类别的估计损失，同时降低存储样本在先前模型和当前模型预测标签之间的蒸馏损失，从而实现持续学习。梯度情景记忆（Gradient Episodic Memory, GEM）算法[49]专注于任务增量学习，利用旧任务样本和梯度定义可行区域，并将当前任务的梯度投影到由先前任务梯度勾勒出的可行区域中，从而约束新任务的更新，以避免干扰以前的任务。在线核心集选择（Online Coreset Selection, OCS）算法[50]引入小批次相似性和样本多样性准则，自适应地选择样本以构建核心集，进而有效保留先前任务的知识。权重对齐（Weight Aligning, WA）算法[51]证明了灾难性遗忘的一个关键原因：最后一个全连接（Full Connected, FC）层中权重的倾斜。为应对这一问题，WA算法首先利用新数据和旧数据联合训练网络，随后对FC层中的权重进行校正。与以往方法不同，WA算法无需引入额外参数，因为它直接利用了倾斜权重本身的信息。

正则化方法通过在损失函数中引入一个额外的正则化项，来巩固之前学到的知识，从而在学习新任务时避免遗忘。LwF（Learning without Forgetting）算法[52]采用知识蒸馏（Knowledge Distillation, KD）[53]策略来保留先前任务的知识。在训练新任务之前，记录新任务数据输入网络后的输出，记为软标签；在后续的训练过程中，通过使新网络的输出接近软标签，来提取并保留先前任务的知识。然而，随着不断添加多样化的任务，相对于先前学习任务的分布偏移可能会逐渐积累误差，最终导致对先前任务的遗忘。基于编码器的终身学习（Encoder Based Lifelong Learning, EBLL）算法[54]通过保留先前任务的重要低维特征表示来扩展LwF算法。对于每项任务，EBLL算法通过自编码器进行端到端的优化，将特征投影到较低维度的流形上。在训练过程中，引入了一个附加的正则化项，以防止当前特征投影偏离先前任务的最优投影。通过这种方式，EBLL算法能够在学习新任务的同时保留先前任务的重要特征表示，为持续学习的应用提供了一种新的思路和框架。塑性权重巩固（Elastic Weight Consolidation, EWC）算法[55]估计网络中每个参数的重要性，并在学习新任务时优先保护重要参数，使其接近旧任务上的取值。这样，即使在学习新任务时进行参数更新，重要参数的变化幅度仍会被限制在一个合理的范围内，从而保留旧任务的知识。

参数隔离方法通过为各任务分配独立的模型参数来防止遗忘。当学习新任务时，可以为其增设新的分支，同时保持先前任务的参数不变。PackNet算法[56]在任务训练完成后评估权重的重要性，并通过剪枝操作删除冗余的连接和不重要的权重。随后，重新进行任务训练以保证获得较好的性能。在后续任务的训练中，可以使用先前任务的权重，但不能对其进行修改，而只能修改先前任务被剪枝的权重。这种方法确保了先前任务的权重在后续任务中保持不变，从而保留了先前任务的知识。胜出子网络（Winning SubNetworks, WSN）算法[57]学习与每个任务相关联子网络的模型权重，同时尝试重用先前子网络的权重。不同于PackNet算法复用旧任务的所有权重，WSN算法在新任务中选择性复用旧任务权重，有效避免了知识迁移的偏差。

近些年，也涌现出一些持续强化学习算法。这些算法主要集中在单一智能体领域，通过在强化学习算法的基础上引入持续学习机制，使智能体在学习新任务的同时保留旧任务的知识。Atkinson等人提出了强化伪排练（Reinforcement-Pseudo-Rehearsal, RePR）模型[58]，该模型具有双记忆系统：长期系统和短期系统，其中，短期系统负责强化学习，而长期系统负责持续学习，这样可以将持续学习与强化学习分开处理。此外，RePR模型还包含一个伪排练系统，通过生成对抗网络（Generative Adversarial Network, GAN）[59]生成伪样本并进行重放。无冲突持续强化学习（COntinual RL Without ConfLict, OWL）算法[60]适用于不同任务具有相同的状态空间的场景。OWL算法通过共享特征提取层，并使用单独的策略头来模拟多模态目标，能够在不同任务之间有效地平衡共享和个性化，从而提高持续强化学习的性能。L2D2-C（Lifelong Learning Distributed Decentralized Collective）算法[61]假设在分布式强化学习场景下，所有智能体共享一个随机初始化的主干网络，但各自存储独立的掩码，以应对自己遇到的任务。同时，L2D2-C算法确保智能体在面对每个任务时，能够向其他智能体查询该任务的信息，并选择在该任务上表现最佳的智能体的掩码，将其集成到自身策略中。

## 本文研究内容

经过对研究现状的调查，发现多智能体强化学习方法发展迅速，并在越来越多的场景中得到应用。然而，这些方法所获得的模型是静态的，无法随着任务的改变自动调整智能体的行为决策。这是因为传统训练方式必然会导致智能体在学习新任务时灾难性地遗忘旧任务。因此，本文深入研究了多智能体强化学习领域的持续学习方法，主要完成了以下工作：

（1）本文对多智能体强化学习的经典算法及其相关应用进行了深入调研，研究了持续学习的前沿算法，并对其进行了分类，为后续多智能体持续强化学习算法的提出提供了充足的准备。

（2）针对多智能体在顺序学习多任务时面临的灾难性遗忘问题，本文将持续学习中的正则化方法引入多智能体强化学习中，基于MADDPG框架提出了OwDP算法，在新任务训练阶段，通过知识蒸馏把旧任务的知识转移到新模型，同时引入联合PCA约束，在参数级别上限制智能体在旧任务上决策协调性的降低。实验结果表明，OwDP算法能有效缓解灾难性遗忘问题，帮助智能体保留旧任务的知识，从而实现多智能体持续强化学习。

（3）针对多智能体在顺序学习多任务时面临的灾难性遗忘问题，本文将持续学习中的经验重放方法引入多智能体强化学习中，基于MADDPG框架提出了MADDPG-OP算法，采用混合小批次度量评估样本在旧任务的小批次中的代表程度，进而选择具有较高代表度的样本，构成旧任务的核心集。在新任务训练阶段，通过PER策略重放核心集中的样本，以缓解多智能体对旧任务的遗忘，保留旧的知识。实验结果表明，MADDPG-OP算法能够避免智能体在旧任务中性能的不均衡，确保所有智能体都得到充分训练，从而实现多智能体持续强化学习。

（4）对本文提出的两种多智能体持续强化学习算法进行了系统的比较，分析了它们持续学习任务的成本和性能，并判断了各自的适用场景。

## 本文组织结构

本文共分为五节，具体结构如下：

第一章，绪论。本章首先介绍了论文的研究背景和意义，并详细介绍了强化学习和持续学习方法的国内外研究现状。接着，说明了多智能体持续强化学习方法存在的问题，并提出了本文针对这些问题的解决方法。最后，给出了本文的组织结构。

第二章，强化学习和持续学习基础。本章首先概述了强化学习的基本概念，探讨了强化学习的经典算法。随后，对比了多智能体强化学习的与单智能体的差异，并详细介绍了多智能体强化学习的建模方法和经典算法。最后，引入了持续学习的概念，阐述了持续学习的分类，并分别介绍了每类持续学习方法中的经典算法。

第三章，基于正则化的多智能体持续强化学习算法。本章首先指出了简单地将OWL算法扩展到多智能体领域遇到的问题，并针对这些问题提出了OwDP算法。接着，详细介绍了OwDP算法的知识蒸馏和联合PCA约束机制，分别从策略和参数级别限制了多智能体对旧任务性能的降低，并介绍了网络更新方式和算法流程。最后，通过对比实验和消融实验，验证了OwDP算法在多智能体持续强化学习中的优越性。

第四章，基于经验重放的多智能体持续强化学习算法。本章首先探讨了将OCS算法扩展到多智能体领域面临的挑战，并针对这些挑战提出了MADDPG-OP算法。接着，详细介绍了MADDPG-OP算法如何采用混合小批次度量选取旧任务样本以构成核心集，以及如何使用PER策略重放核心集样本，并阐述了网络更新方式和算法流程。之后，通过对比实验，验证了MADDPG-OP算法在多智能体持续强化学习中的优越性。最后，深入对比了本文提出的两种算法的优缺点，并分析了各自的适用场景。

第五章，总结与展望。本章总结了全文的主要工作内容，反思了本文研究的不足之处，并为未来工作指出了方向。

# 第二章 强化学习和持续学习理论基础

## 强化学习

### 马尔可夫决策过程

强化学习是通过智能体与环境交互来学习如何实现目标的方法。与监督学习和无监督学习不同，强化学习方法不需要大量的样本，而是在智能体与不断环境交互过程中的学习。智能体与环境的交互方式如图2.1。在每一轮交互中，智能体首先接收环境提供的当前状态信息；然后，基于这个状态，智能体通过自身决策给出动作；最后，环境在接收到智能体选择的动作后，生成奖励信号反馈给智能体，并转移到新的状态。智能体进入下一轮与环境的交互，不断重复。



图2.1 智能体与环境的交互过程

马尔可夫决策过程是强化学习的重要概念，它是一类具有马尔可夫性质的随机过程，即下一状态只与当前状态有关，而不会受过去状态的影响，用公式表示为，其中，表示当前状态转移到下一状态的概率。在强化学习中，可以使用马尔可夫决策过程对智能体和环境的交互建模。

马尔可夫决策过程由元组构成，其中，是状态集合，是动作集合，是状态转移函数，表示在状态执行动作后转移到状态的概率，是奖励函数，表示在状态执行动作获得的环境奖励，是折扣因子，引入折扣因子是因为远期利益具有不确定性。定义从时刻开始，直到终止状态，智能体的回报为：



其中，为智能体在时刻获得的奖励。

### 价值函数与贝尔曼方程

在马尔可夫决策过程中，智能体根据当前状态选择一个动作，这个过程需要策略的指导。智能体的策略通常用字母表示，策略表示在状态时选择动作的概率。由于马尔可夫性质的存在，策略只依赖于当前状态，而不需要考虑过去的状态。智能体策略可以分为确定性策略和随机性策略。当智能体策略为确定性策略时，智能体选择某个动作的概率为1，而选择其他动作的概率为0，因此智能体只执行一个确定的动作。当智能体策略为随机性策略时，智能体输出所有动作的概率分布。通过从这个分布中随机采样，智能体得到一个具体的动作去执行，因此智能体执行的动作是不确定的。

智能体累计回报的期望与策略有关。用表示马尔可夫决策过程中基于策略的状态价值函数，该函数衡量了智能体从状态出发经策略指导获得的期望回报，计算公式为：



同时，还可以用表示基于策略的动作价值函数，该函数衡量了智能体在状态执行动作后经策略指导获得的期望回报，计算公式为：



可以观察到，状态价值函数和动作价值函数存在一定的关系，即状态的价值等于在该状态下基于策略选取各个动作的概率与这些动作各自价值的乘积之和，用公式描述为：



对于状态价值函数，当前状态的价值和下个状态的价值也存在一定的关系，即当前状态的价值包含下一状态的价值，两者满足递归关系，用公式描述为：



式称为状态价值函数的贝尔曼方程，该方程说明当前状态的价值等于即时奖励与下个状态折扣回报的期望之和。同理，可以得到动作价值函数的贝尔曼期望方程：



贝尔曼方程是强化学习的重要组成部分，也是很多经典算法的基础。

强化学习的目标找到一个策略，使得从初始状态开始的累计奖励的期望最大化。这个策略就是最优策略，可以用表示。基于最优策略，可以定义最优状态价值函数和最优动作价值函数：



### 基于值函数的方法

基于值函数方法的核心在于学习最优的状态价值函数或动作价值函数，之后从或中推导出最优策略。这些方法包含动态规划方法、蒙特卡洛方法和时序差分方法。动态规划是一种基于模型的方法，然而在大多数强化学习场景中，马尔可夫决策过程的状态转移函数往往无法被精确定义，这使得动态规划无法使用。蒙特卡洛方法和时序差分方法作为无模型的方法，在强化学习领域具有广泛的适用性。两者都是基于贝尔曼方程，利用后续状态的价值来更新当前状态的价值，但具体更新方式有所区别。蒙特卡洛方法的更新方式为：



其中，为时刻的状态，为学习率。时序差分算法的更新方式为：



其中，为时刻的状态。可以观察到，蒙特卡洛方法必须在整个序列结束后才能进行更新，而时序差分方法在当前步结束即可进行更新。

**（1）Q-Learning算法**

Q-Learning算法是一种基于TD方法的算法，适用于动作和状态都是离散，且空间都比较小的环境。在智能体训练之前，先创建一个Q表，用于存储当前策略下所有状态动作对的价值，然后将表中所有项初始化为0。在智能体训练过程中，用贪婪策略选择Q表中当前状态下价值最大的动作，即。智能体用该动作与环境交互，获得环境奖励并转移到新的状态，然后按下式更新Q表：



如此重复，直到Q表中的Q值不再变化，或者变化小于某个阈值。

智能体用贪婪策略在Q表中选择价值最大的动作，可能会导致某些状态动作对永远不会出现，以至于其动作价值一直为0，影响后续策略的提升。一个常用的解决办法是使用-贪婪策略：有的概率选择随机动作，的概率选择价值最大的动作，用公式描述为：



其中，为动作集合的大小。

**（2）Sarsa算法**

Sarsa算法同样属于TD方法的一种。与Q-Learning算法相比，两者最显著的区别在于TD的更新方式。Sarsa算法的更新方式为：



其中，为智能体在时刻基于-贪婪策略选择的动作。

**（3）DQN算法**

传统的强化学习方法在状态和动作空间较小的场景表现良好。然而，当状态或动作的数量较大时，在计算机中存储这样的Q表是不现实的。随着深度学习的快速发展，强化学习迎来重要的发展方向。深度学习的引入使得强化学习能够处理高维、连续的状态和动作空间，扩展了强化学习的适用场景。

在2013年NIPS会议上，DeepMind团队展示了DQN算法，该算法利用卷积神经网络处理像素输入，并成功应用于Atari游戏的DQN算法。DQN算法利用深度神经网络拟合Q值，使智能体能够通过该网络估计该状态下所有动作的价值，而不再需要通过表格查询，这个网络被称为Q网络。

当状态连续而动作离散时，Q网络的输入是状态，输出是每个动作的价值。以Gym平台[62]的车杆（CartPole）环境为例，该环境包含一辆小车，小车上有一个可以自由活动的杆，任务目标是通过左右移动小车使杆保持竖直状态。智能体的状态是一个4维向量，包含小车位置、小车速度、杆的角度和杆顶端的速度，每维都是连续的；动作空间大小为2，即小车向左或向右移动。图2.2展示了车杆环境的Q网络结构，其中网络输入是智能体的4维状态，输出是该状态下两个动作的价值。当状态和动作都连续时，Q网络的输入是状态和动作，输出一个标量，表示在状态下选择动作的价值。



图2.2 车杆环境的Q网络结构

DQN算法对Q-learning算法还有两个重要的改进：经验回放和目标网络。

经验回放方法通过维护一个经验回放池来改进Q-learning算法。在每次智能体与环境交互时，会将状态、动作、奖励和下一状态的四元组数据存储到经验回放池中。之后，再从经验回放池随机采样若干个数据对Q网络进行训练。通过使用经验回放，可以打破样本间的关联性，并提高样本利用率，从而提高Q网络的训练效率。

目标网络方法使用两套Q网络：训练网络（原网络）和目标网络。训练网络在训练每一步都会更新，而目标网络保留比较旧的网络参数，每隔若干步与训练网络同步一次。在训练过程中，利用训练网络计算当前状态的动作价值，并以此来选择价值最高的动作；利用目标网络计算下个状态的动作价值，进而计算TD目标，以此来更新训练网络。通过使用目标网络，有效解决了Q值高估的问题，降低了训练的不稳定性。

Q网络的更新也是基于TD算法的，使向目标靠近，其中，为Q训练网络，为Q目标网络，为Q训练网络参数，为Q目标网络参数。于是，对于一组数据，Q训练网络的损失函数为：



其中，为采样个数。DQN算法的训练过程如图2.3所示。



图2.3 DQN算法的训练过程

### 基于策略的方法

Q-learning算法和DQN算法需要学习值函数，并通过学习好的值函数来导出最优策略。然而，还有一类算法可以直接学习策略，并基于这个策略选择动作，这就是基于策略的方法。

基于策略的方法将策略参数化，用表示，可以简写为，其中，为策略参数。动作的选择不再基于状态动作价值，而是基于策略函数的输出。将某个状态输入到策略函数，可以直接得到动作的概率分布。基于这个概率分布，对动作的选择有两种基本的方法：确定性策略和随机性策略。确定性策略直接选择概率最大的动作去执行；而随机性策略会对动作概率分布采样，选择采样到的动作去执行。

策略学习的目标是最大化策略在环境中的期望回报，目标函数为：



其中，表示智能体初始状态，表示策略指导下的状态价值函数。将目标函数对策略参数求导，可以得到梯度，公式如下：



其中，表示策略的状态访问分布，即策略下智能体访问到的状态的分布，表示在策略下的动作价值函数。然后，用梯度上升的方法最大化目标函数，从而获得最优策略：



其中，为学习率。

**（1）REINFORCE算法**

容易观察到，计算策略梯度需要用到，一种常用的方法是通过蒙特卡洛采样对进行估计，将动作价值近似为累计回报。在一个序列采样结束后，策略梯度计算公式为：



其中，是该序列的长度。这就是著名的REINFORCE算法，它可以利用采样序列的数据直接计算出策略参数的梯度，并根据梯度信息更新当前策略。然而，REINFORCE算法存在一些缺点：首先，需要在智能体与环境完成一轮交互后才能计算策略参数的梯度，这大大降低了策略的训练效率；其次，蒙特卡洛方法会使梯度估计的方差过大，从而导致训练的不稳定。

**（2）Actor-Critic算法**

除了利用蒙特卡洛方法估计动作价值外，还可以采用基于值函数的方法学习一个价值函数，并利用这个价值函数来指导策略的更新。Actor-Critic算法是一个将价值学习和策略学习相融合的算法框架，通常与深度学习相结合使用。该算法框架分为Actor（策略网络）和Critic（价值网络）两部分，具体结构如图2.4所示。Actor网络的主要任务是接收环境提供的状态信息，并基于生成相应的动作。在Critic网络的指导下，Actor网络采用策略梯度方法不断优化其策略。同时，Critic网络利用Actor网络与环境交互产生的数据进行学习，目标在于构建一个更准确的价值函数，从而有效指导Actor网络进行策略更新。



图2.4 Actor-Critic算法架构

**（3）DDPG算法**

Liiicp将确定性策略梯度（Deterministic Policy Gradient, DPG）算法[63]和DQN算法结合，提出DDPG算法。该算法也属于一种Actor-Critic算法，包含策略网络和价值网络两部分。与本文前面介绍的基于策略的算法不同，DDPG算法属于确定性策略算法，策略网络直接输出确定的动作。因此，策略网络不再表示为，而是。价值网络仍表示为。

DDPG算法采用了DQN算法两个成功的方法：经验回放和目标网络，整体结构如图2.5。



图2.5 DDPG算法架构

这四个网络有不同的功能定位：

1. Actor训练网络，与环境交互，接收环境的状态并选择动作反馈给环境；同时，在Critic训练网络的指导下更新策略参数。
2. Actor目标网络，表示为，为其网络参数，接收经验回放池的下一状态并选择下一动作，以辅助Critic训练网络更新。
3. Critic训练网络，表示为，为其网络参数，接收经验回放池的状态和Actor训练网络选择的动作，以指导Actor训练网络更新。
4. Critic目标网络，表示为，为其网络参数，接收经验回放池的下一状态和Actor目标网络选择的动作，计算目标Q值，以辅助Critic训练网络更新。

Critic训练网络采用TD算法更新，即最小化目标损失：



其中，为采样个数。Actor训练网络根据Critic训练网络的反馈的Q值计算梯度，从而改进策略，其策略梯度为：



为了避免网络参数更新的不稳定性，DDPG算法采用软更新的方式更新目标网络，即让目标网络缓慢接近训练网络，更新公式为：



其中，是一个比较小的数。

## 多智能体强化学习

在单智能体强化学习中，环境通常是稳态的，这意味着奖励函数和状态转移函数始终保持不变。然而，在多智能体环境中，智能体不仅与环境交互，还与其他智能体直接或间接地交互。这导致即使智能体在相同状态执行相同动作，状态转移和奖励的分布也可能是不同的。这对强化学习提出了更高的要求。

### 部分可观测的马尔可夫决策过程

在多智能体强化学习中，智能体通常无法看到全局信息，而仅能利用局部观测信息去做决策。这种场景可以建模为部分可观测的马尔可夫决策过程（Partially Observable Markov Decision Process, POMDP）：该过程由元组构成，其中，是全局状态的集合；是智能体联合观测的集合，联合观测，其中是第个智能体的局部观测信息；是智能体联合动作的集合，联合动作，其中是第个智能体的动作；是状态转移函数，表示在全局状态执行联合动作后转移到全局状态的概率；是智能体奖励函数的集合，奖励函数表示第个智能体在全局状态执行动作获得的奖励；是折扣因子。多智能体与环境的交互过程可以描述如下：每个智能体从环境的全局状态中获取局部观测信息，然后根据选择合适的动作。环境接收所有智能体的联合动作，并根据这些动作转移到下一状态，然后将奖励返回给各智能体。每个智能体的目标是最大化累计回报：



其中，是第个智能体的回报，是结束时刻，是第个智能体在时刻的奖励。

### 多智能体深度确定性策略梯度算法

Ryan Lowe等人提出基于Actor-Critic框架的多智能体深度确定性策略梯度（Multi-Agent Deep Deterministic Policy Gradient , MADDPG）算法，将DDPG算法推广到多智能体场景。MADDPG算法遵从CTDE范式：在训练阶段，智能体使用全局信息优化训练效果；而在执行阶段，每个智能体独立地根据自身策略做出动作。

图2.6描述了MADDPG算法的网络结构。假设有个智能体，每个智能体都拥有各自独立的Actor网络和Critic网络，其中，网络为第个智能体的策略网络，记为：，为策略网络参数，其输入为第个智能体的局部观测值；网络为第个智能体的价值网络，记为：，为价值网络参数，该网络用于对策略进行评价，以指导网络的学习，其输入为所有个智能体的联合局部观测值和联合动作。MADDPG算法同样引入了目标网络的概念，采用软更新的方式更新Actor和Critic目标网络的参数，以实现更稳定的学习。



图2.6 MADDPG算法架构

从经验池中抽取样本，该样本的数据构成为：，其中，分别为当前时刻个智能体的联合观测值、联合动作和获得的奖励值，为下一时刻个智能体的联合观测值。在MADDPG算法中，训练网络参数更新的损失函数为：



其中，是目标网络，是目标网络，。

Actor网络在Critic网络的指导下，最大化策略在环境中的期望回报，则训练网络策略学习的目标函数为：



其中，，。可得更新训练网络参数的梯度为：



## 持续学习

基于深度学习的人工智能方法在众多任务中取得了良好的泛化性能，在个别任务中的表现甚至超过了人类。然而，深度学习方法生成的模型是静态的，无法随着时间的推移扩展其行为，这是一个很大的局限。为了解决这个问题，持续学习应运而生。持续学习要解决的问题是：在一组任务上按顺序训练模型，如何使模型在学习下一个新任务时，能够继续保持执行所有先前任务的性能。

在持续学习中，“任务”指的是具有新一批数据的独立训练阶段，它可以是一个新的真实任务，如添加一个新的猫狗分类任务；也可以是一个新的类别，比如在牛羊分类中添加一个新的猫类别。按照这种“任务”的不同，可以将持续学习分为任务增量学习和类增量学习。对于持续学习模型，设是第个任务的输入分布，是第个任务的输出分布，对不同“任务”而言，。任务增量学习定义为且，即新任务标签与旧任务标签不同；类增量学习定义为，即新任务标签包含旧任务标签，也就是在原任务添加了一个新类。本文要解决的多智能体协同决策问题属于任务增量学习，因此本文将重点介绍任务增量学习方法。

根据任务特定信息的存储和使用方式，将持续学习区分为三类：正则化方法、重放方法和参数隔离方法。因本文主要使用了正则化方法和重放方法，因此下面将重点介绍这两种持续学习方法。

### 正则化方法

正则化方法避免了存储原始输入，而是在损失函数中引入一个额外的正则化项，从而在学习新数据时巩固以前的知识。

**（1）EWC算法**

Kirkpatrick提出EWC算法，让模型参数受限地变动，阻碍对旧任务重要参数的变化。假设持续学习任务的顺序为A->B，如图2.7所示，在学习新任务B的同时，EWC算法通过将参数限制在以为中心的旧任务A的低误差区域来保护任务A的性能，其中，为网络对任务A训练的最优参数。如果只根据任务B采取梯度步骤（蓝色箭头），将使任务B的损失最小化，但会破坏任务A的性能。相反，如果对模型参数施加完全相同的约束，即L2范数约束（绿色箭头），则会导致限制太过严格，使得模型在记住任务A的同时无法学习任务B，从而牺牲了任务B的性能。EWC算法通过计算网络各参数对任务A的重要性，对网络参数施加不同的约束，从而在学好任务B的同时，确保任务A的性能不会明显下降（红色箭头）。



图2.7 EWC算法训练轨迹

设为网络参数，是一个列向量。对引入EWC约束，使用贝叶斯公式从参数的先验概率和数据的后验概率计算参数的后验概率：



其中，是全部任务的数据。假设数据被分为两个独立的部分，一部分来自旧任务A（），另一部分来自新任务B（），公式可以重写为：



EWC算法的目标为最大化，即最大化公式等号右边的三项，其中，是常数项；最大化等价于找到使出现概率最大的，可用训练任务B的损失函数代替；为与旧任务A有关的项，应用拉普拉斯近似，将近似为均值为的高斯分布。由此，最大化近似为最大化：



其中，为训练新任务B的损失函数，设置旧任务与新任务相比的重要性，对角精度矩阵由Fisher信息矩阵的对角线给定。

EWC算法为历史每个任务都保留一个对角精度矩阵，随着任务数量的持续增长，可能会造成较大的内存开销。为了将EWC算法推广到多任务情形，Online EWC算法[64]提出递归地应用拉普拉斯近似，在任务A和B之后，学习第三个任务C时，贝叶斯后验分解为：



其中，为来自任务C的数据。因此，最大化近似为最大化：



其中，设置旧任务与新任务相比的重要性，分别为对任务A、B的对角精度矩阵，为按顺序学习旧任务A、B后的网络参数。可以将其推广到更多任务的场景。

**（2）LwF算法**

LwF算法利用知识蒸馏方法，只使用新任务的数据，将先前任务的知识传递给新模型，从而保持执行旧任务的能力。算法结构如下图：



图2.8 LwF算法网络结构

图2.8中，为共享参数，为旧任务参数，为新任务参数。设新任务样本为，其中为数据，为数据标签。定义和组成的网络为新网络，和组成的网络为旧网络。在新任务训练前，数据在旧网络的输出为，即软标签；在训练过程中，数据在旧网络的输出为，其中，为训练过程中的共享参数和旧任务参数。由于共享参数和旧任务参数在训练阶段会发生变化，因此在没有添加约束条件的情况下，。通过限制接近软标签来实现知识蒸馏，将旧模型知识转移到新模型，损失函数采用交叉熵损失，如下式：



设样本在新网络输出为，其中，为训练过程中新任务参数。在新任务的训练中，损失函数促使新网络输出与标签保持一致。对于分类任务，同样采用交叉熵损失，如下式：



因此，LwF算法的整体损失函数为：



其中，为的所有参数，为新旧任务的损失平衡的权重值。

LwF算法流程如下表所示：

表2.1 LwF算法

|  |
| --- |
| 算法名称：LwF算法 |
| 输入： |
| ：共享参数 |
| ：旧任务参数 |
| ：新任务数据及其标签 |
| 初始化： |
| 计算软标签 |
| 随机初始化新任务参数 |
| 训练： |
| 计算 |
| 计算 |
| 更新参数 |
| 输出： |
|  |

### 重放方法

重放方法采用两种方式处理样本：一种是直接存储原始样本，另一种是使用生成性模型生成伪样本。在学习新任务时，通过重放先前的任务样本来减轻遗忘。重放缓冲区是重放方法中的关键技术，用于存储和回放先前任务的样本。它涉及选择保留哪些样本，在样本选择策略中，常见的方法有随机选择和基于重要性选择。此外，重放缓冲区的大小可以根据需求进行调整。当模型学习新的任务时，可以从重放缓冲区中采样一批旧任务的样本进行训练，以保持模型对先前任务的知识并防止遗忘。

OCS算法提出了一种自适应地选择样本来构建重放缓冲区（或核心集）的方法。该算法引入了小批次相似性和样本多样性的准则，以选择最能代表旧任务小批次样本集的样本，并在新任务训练时重放这些样本，从而保留先前任务的知识。由于从整个样本集中获取具有代表性的子集是计算昂贵且困难的，OCS算法将小批次样本集视为样本集的近似，并且在每次小批次迭代中选择少量具有代表性的样本。

设为当前任务的小批次样本集，为小批次样本集的第个数据点及其对应的标签，为持续学习的神经网络，为网络参数，则可以将对于网络训练的平均梯度表示为，对于网络训练的梯度表示为。和之间的小批次相似性由下式给出：



其中，表示向量的L2范数。

设为将小批次样本集去除后的样本集，则和之间的样本多样性由下式给出：



其中，为小批次样本集的大小，表示在且的范围内求和。小批次相似性将小批次视为当前任务数据集的近似，并比较数据点的梯度向量与小批次样本集之间的小批次级相似性。可以看到，小批次相似度描述数据点梯度向量与小批次样本集平均梯度向量夹角的余弦值，它衡量给定数据样本在每个训练步骤中对当前小批次的表示程度。将每个数据点的样本多样性表示为样本本身与同一小批次中的其他样本之间的平均相异度(即负相似性的平均值)。因此，公式中的样本多样性度量为负，范围为。

设为先前任务的核心集，为从核心集采样获得的数据集，其对于当前网络训练的平均梯度为。和之间的核心集亲和性由下式给出：



核心集亲和性是选择当前小批次中与先前任务核心集梯度向量的夹角最小的样本，认为重放这些样本不会降低模型在先前任务中性能。即使之前的任务在训练结束后无法再次访问，但通过选择性地填充的重放缓冲区，可以有效地保留先前任务的关键信息。综上所述，设是小批次样本数，训练任务的核心集选择公式可以表示为：



其中，表示选择最大的个样本，是控制模型可塑性和稳定性的超参数。

可以看到，在式中，最大化的三项具有以下含义：前两项表示选择最代表当前任务的样本，以便在后续任务的训练中保留当前任务的知识；第三项表示选择与先前任务核心集最相似的样本，以避免重放当前任务的样本对先前任务的性能产生影响。需要注意，在第一个任务训练期间，没有以前任务的干扰，OCS算法选择最大限度地提高小批次相似性和样本多样性的个样本。

### 持续强化学习方法

2013年，Deep Mind公司将深度学习应用到强化学习领域，开启了强化学习的新篇章。在过去的十年，涌现出很多强化学习算法。但这些算法获得的模型是静态的，无法随着任务的改变而调整智能体的行为决策。这是因为智能体在被训练能够执行某一任务后，采用传统训练方式继续学习新任务，必然会遭遇旧任务的灾难性遗忘。将强化学习和持续学习相结合，能够使智能体在面对连续不断的任务时更加灵活和高效，在多个任务中展现出更强大的学习能力和适应能力。

Samuel Kessler等人提出OWL算法，针对不同任务具有相同状态空间的场景。在图2.9中有一个这样的例子：四幅图中的智能体均位于正中间（十字位置），1、2属于一个任务，设为任务A，目标位于智能体右下方的房间，所以智能体向下移动会获得高奖励值，向左移动会获得低奖励值（或负奖励值）；3、4属于另一个任务，设为任务B，目标位于左上方的房间，所以智能体向左移动会获得高奖励值，向下移动会获得低奖励值（或负奖励值）。可以看出，任务A和任务B的状态空间是相同的，但目标是相反的。智能体在执行任务A和任务B时，最有效的策略是根据所需要执行的任务将智能体引导至不同的目标方向，即对具有相同状态空间的不同任务，可以给出不同的最优决策。如果采用传统的持续强化学习算法，按照任务A->任务B或者任务B->任务A的顺序，对智能体的策略模型进行训练，则会导致干扰的发生，因为策略模型参数会进行相反的优化。



图2.9 相同状态空间中的不同任务

对于这种场景，OWL算法的解决方案是使用如图2.10所示的策略网络结构，该网络的特征提取层（Feature Extraction）为多个任务共享，如：被任务A和任务B共享。而多个线性层（Linear）构成多头结构以适应不同的任务。在图2.10中，对特征提取层添加EWC约束，通过限制该层重要参数的变化来减轻对旧任务遗忘。



图2.10 OWL算法中具有多头结构的策略网络

## 本章小结

本章主要介绍了强化学习和持续学习的基本概念和经典算法。

强化学习使用马尔可夫决策过程对环境进行建模，并通过价值函数和贝尔曼方程推导当前状态和下一状态之间的关系。强化学习方法可以分为基于价值的方法和基于策略的方法，本章分别介绍了Q-learning、Sarsa和REINFORCE等经典算法。随着深度学习在强化学习中的应用，强化学习取得了快速的发展，本章介绍了基于价值的DQN算法和基于策略的Actor-Critic算法以及DDPG算法。

多智能体强化学习的环境可以建模为POMDP，在这种设定下，每个智能体只能获取全局状态的局部观测信息。本章介绍了基于CTDE范式的MADDPG算法的基本原理，为后续研究及实验提供了理论基础。

持续学习的目标是让模型能在顺序学习任务的过程中能够扩展新任务知识，同时不忘记旧任务。本章将持续学习方法分为三类：重放方法、正则化方法和参数隔离方法，并介绍了几种代表性算法。之后，对持续学习和强化学习的融合算法做了简单介绍，这些算法旨在让智能体获得执行多任务的能力。

# 第三章 基于正则化的多智能体持续强化学习算法

传统的持续强化学习算法在单个智能体场景中表现出良好的性能，然而在面对多智能体场景时却无法发挥作用。为了解决这一问题，本章首先引入持续学习中的正则化方法，提出了一种基于正则化的多智能体持续强化学习算法。该算法通过对智能体策略网络的关键参数进行限制性更新，避免了多智能体学习多任务时性能的下降。

## 问题分析

在2.3.3节中介绍的OWL算法通过为策略网络引入正则化约束，并为不同任务设置不同的策略头，显著提高了智能体在学习多任务时的稳定性和适应性。然而，简单地将该算法扩展到多智能体场景，忽略了智能体之间的协同性，这将导致算法无法很好地收敛。因此，实现多智能体持续强化学习，需要设计适当的方法来促进多个智能体之间的协同和合作。

## 基于正则化的多智能体持续强化学习算法



图3.1 OwDP算法架构

本章基于MADDPG框架，将持续学习中的正则化方法应用到多智能体强化学习中，提出了一种基于正则化的多智能体持续强化学习算法。图3.1为算法的总体框架，整个框架包括4个部分，分别为环境、经验池、多智能体和正则化持续学习机制。在该框架中，多智能体按照当前策略与环境交互，将状态、动作和奖励等交互数据存入经验池，作为训练多智体策略网络Actor的样本来源。对多智能体的Actor网络进行持续强化学习训练，即：正则化持续学习机制，是整个框架的核心，如图3.1中绿色方框部分。

针对多智能体持续强化学习机制的实现，本章在MADDPG强化学习方法的基础上，结合OWL算法和知识蒸馏，为每个智能体的Actor网络添加Online EWC约束和知识蒸馏，以缓解Actor网络学习新任务时对已学旧任务的遗忘。同时从全局的角度，为了保持多智能体在旧任务中策略的协同性，提出基于所有智能体Actor网络计算联合PCA约束，并将该约束应用到Actor网络参数更新的过程中。鉴于上述实现方法，这里将本章的多智能体持续强化学习方法命名为：OWL with Distillation & PCA，简称OwDP算法。

### Online EWC约束

本章将OWL算法和知识蒸馏引入到MADDPG框架的策略网络Actor中，以实现Actor网络的持续学习。Actor网络的具体结构如图3.2所示，将该网络划分为特征提取层和线性层，其中，特征提取层在不同任务之间共享，线性层为每个任务单独使用。Online EWC约束作用于Actor网络的特征提取层，其作用是通过与旧任务参数的比较，限制对旧任务重要参数的变化。而知识蒸馏作用于Actor网络的特征提取层和线性层，目的是将旧任务的知识转移给新网络。



图3.2 Actor网络结构

在图3.2中，蓝色线框代表针对旧任务已训练好的Actor网络，包括网络的特征提取层和线性层；橙色和黄色线框分别代表在新任务上待训练的Actor网络的特征提取层和线性层。该Actor网络包含层，其中前层被划分为特征提取层，第层则是线性层。

OWL算法采用EWC约束来限制特征提取层参数的变化，从而巩固旧知识的参数，有效地缓解了持续学习中的灾难性遗忘问题。EWC约束需要比较准确地估计不同参数对旧任务的重要性，并基于这个重要性系数对网络参数添加约束，如式。在任务数量增加时，需要保存之前每个任务的梯度信息以进行参数重要性的计算，这会增加计算成本。因此，本章采用了Online EWC约束，根据任务的相对重要性和影响程度自动更新参数对旧任务的重要性，避免了保存多个旧任务的梯度信息，以降低计算的复杂度。

由于不同任务只共享Actor网络的特征提取层，因此只需计算Actor网络特征提取层中各参数的重要性。假设有个智能体，每个智能体都具有图3.2所示的相同的Actor网络结构，记为：。个智能体持续学习的任务顺序为：任务A->任务B->任务C。以第个智能体为例，令表示网络特征提取层的参数，其同样为列向量。在学习任务B时，用对角精度矩阵来估计对任务B的重要性，的计算步骤如下：

1. 根据当前智能体的局部观测，通过策略网络生成相应的动作。
2. 将个智能体当前的状态和个Actor网络：生成的动作传递给网络，将网络输出作为目标函数，如式。
3. 将目标函数对中每个参数求二阶导，其中，为的第个参数，为的参数个数。
4. 取多次数据，计算二阶导的平均值，将这些值作为对角精度矩阵的对角线元素。

需要注意的是，本章对角精度矩阵的计算需要在网络和网络针对任务B训练完成后进行。智能体对每个任务单独保存一个对角精度矩阵可能造成较大的内存开销，因此，在学习任务B后，可以将A、B两个任务的对角精度矩阵合并为：



其中，，设置旧任务与新任务相比的重要性，为对任务A的对角精度矩阵。再学习任务C时，对于每个智能体的Actor网络，在损失函数中添加下式的正则化项：



其中，为网络特征提取层针对旧任务训练的最优参数，正则化项系数可以反映中不同参数对旧任务的重要程度，且随着持续学习新任务自动更新。式可称为Online EWC约束。可以看出，将Online EWC约束引入MADDPG框架Actor网络的特征提取层中，可以实现对旧任务参数的保护和约束，以促进持续学习和知识共享。

### 知识蒸馏

知识蒸馏[53]最早在模型压缩领域被提出，可用于将复杂网络的策略提取到一个简单网络中，目前已成为解决持续学习问题的重要方法。这种方法使用两个模型：一个是训练好的教师模型，通过使用软标签标注数据，为学生模型提供正则化约束；另一个是未经训练的学生模型，该模型要向教师模型学习。在训练新任务之前，记录新任务数据在教师模型的输出，随后在训练期间使用该输出来提取先前任务知识。知识蒸馏的损失函数可以表示为式。

与EWC算法关注神经网络中具体参数的更新空间范围不同，知识蒸馏方法更关注神经网络输出的不变性，通过将已有模型的知识传递给新模型，保留先前任务的知识和泛化能力，加速新任务的学习。

为了进一步提高多智能体持续强化学习的性能，本章在MADDPG框架下，对Actor网络引入知识蒸馏。设学习顺序为任务A->任务B，将学习任务A后的网络称为旧网络，记为：，其特征提取层的参数为，线性层的参数为。再学习任务B时，网络学习的基本步骤为：从任务B的经验池中随机采样，获得第个智能体的局部观测值，将其传入旧网络，其输出被视为软标签。同时，设当前网络的特征提取层和旧网络的线性层组合的网络为新网络，记为：，将传入该网络，使输出尽可能接近软标签。知识蒸馏采用交叉熵损失，损失函数为：



### 构建联合PCA约束

主成分分析（Principal Component Analysis, PCA）[65]是一种使用广泛的数据降维算法，主要思想是在原有维特征的基础上重新构造出维特征，其中。这维特征包含数据的绝大部分特征信息，也被称为主成分。PCA算法的工作就是从原始空间中顺序地找一组相互正交的新坐标轴，实现对数据特征的降维处理。新坐标轴的选择与数据是密切相关的。首先，选择数据方差最大的方向作为第一个新坐标轴；接着，在与第一个新坐标轴垂直的平面上，选择方差最大的方向作为第二个新坐标轴；然后，在与前两个新坐标轴都垂直的平面上，选择方差最大的方向作为第三个新坐标轴。按此方法类推，可以得到一系列新坐标轴。如图3.3所示，红色实点为数据点，为原坐标轴，为PCA算法转换的新坐标轴，且数据在方向的方差最大。因此，可以选取方向为数据的新特征。



图3.3 PCA维度转换

在多智能体持续强化学习的过程中，多智能体在旧任务上决策的协同性往往会随着新任务的训练而被破坏。为解决这一问题，在智能体训练过程中，可以每隔若干步对策略网络的参数进行采样，并将这些参数构成一个整体的参数采样样本集。在该样本集中，如果参数在某些方向上的方差较小，说明参数在这些方向上已经训练得比较好，则在这些维度方向上的取值特征将是确保多智能体协调决策性能的主要参数取值特征。如果多智能体在后续学习新任务的过程中，从整体上对这些维度上的参数训练进行约束，可以减轻对旧任务决策协同性能的破坏。

基于这一思想，本章提出了一种对策略网络构建联合PCA约束的方法，该方法利用PCA算法分析训练过程中参数在不同方向上的变化情况，并选择方差最小的个方向，限制这些方向上参数的变化有助于维持多智能体之间决策的协调。然后，通过引入约束项，限制参数在这些方向的大幅变化，从而限制多智能体在旧任务上协同性的降低。

具体而言，针对MADDPG框架，对于个智能体，每个智能体的Actor网络均有层，其中前层为特征提取层。将个Actor网络特征提取层的每一层参数分别进行合并，作为新的特征提取层，如图3.4中的蓝色线框。这里称新的特征提取层为联合特征提取层，其同样会有层。将联合PCA约束作用于层的联合特征提取层的参数训练中，图3.4中橙色和黄色方框分别代表在新任务上待训练网络的特征提取层和线性层。



图3.4 联合PCA约束示意图

首先需要对这层联合特征提取层的训练轨迹分别采样，以构成采样样本集。以第层联合特征提取层为例说明具体的采样过程：假设第层联合特征提取层的参数维度为，即个Actor网络第层特征提取层的参数维度之和为，若采样次，可得第层联合特征提取层参数的采样样本集，用矩阵表示，该矩阵为行列。

根据上述采样过程，依次可得层联合特征提取层的采样矩阵：，在这些采样矩阵的基础上，结合PCA算法计算联合PCA约束。具体计算步骤如下：

1. 对矩阵去中心化，即每一维特征减去各自的平均值。
2. 计算协方差矩阵，该协方差矩阵为行列。
3. 求协方差矩阵的特征值和特征向量，并将特征值按从大到小排列。
4. 选择最大的个特征值作为特征值矩阵的对角线元素，该矩阵为行列，非对角线元素为0。然后，将对应的个特征向量作为行向量组成特征向量矩阵，特征向量矩阵为行列。
5. 在和的基础上，构建联合PCA约束项：



其中，表示向量的L2范数，表示对角线元素为的对角矩阵与对角矩阵相减，为联合特征提取层参数，为在新任务中待训练的第层联合特征提取层参数，为第层联合特征提取层参数针对旧任务训练所得的最优参数，为足够大的系数，为用来放松对部分参数的约束系数。式中的第一部分严格约束联合特征提取层参数在所有维度方向的更新，第二部分可放松对部分维度方向的约束，这里指方差大的维度方向。

综合起来，式所示联合PCA约束项的主要目的是限制联合特征提取层参数在方差较小的维度方向的更新。在学习新任务的过程中，通过对策略网络的层联合特征提取层加入联合PCA约束项，可减少多智能体在旧任务上决策协同性能的破坏。

如果限制联合特征提取层参数在某些维度方向上的变化，完全可以用式这种更加简洁的方式代替式。但这种方法需要保存所有特征向量，即特征向量矩阵将是行列，其中是第层联合特征提取层参数的维度。当网络规模较大时，可能导致特征向量矩阵难以存储和计算。因此，本章采用式这种间接的方法来计算联合PCA约束项。



在式中，分别为联合PCA约束计算步骤（4）中所有特征值组成的向量和所有特征向量组成的矩阵，常数与对角矩阵相除表示与对角矩阵中每个元素相除得到的新对角矩阵。

### 算法流程

综合以上描述，本章在MADDPG框架下提出了OwDP算法，以实现多智能体持续强化学习。因不同任务之间共享特征提取层，故只对特征提取层计算EWC约束。知识蒸馏损失与特征提取层和线性层均有关。综合传统强化学习、EWC约束和知识蒸馏，网络的目标为最大化下式：



其中，的定义如式，为Online EWC约束和知识蒸馏相对新任务的重要性。

的EWC约束损失为式，可得EWC约束的梯度为：



的知识蒸馏损失为式，可得知识蒸馏的梯度公式为：



的梯度如式，结合式和式，OwDP算法网络参数更新的梯度公式变为：



接下来，从整体角度对多个智能体Actor网络构成的联合特征提取层的参数进行更新，以保持多智能体对旧任务决策的协同性。根据式计算联合PCA约束，用该约束的负梯度更新联合特征提取层的参数：



OwDP算法流程如下表所示：

表3.1 OwDP算法

|  |
| --- |
| 算法名称：OwDP算法 |
| **for** 任务序号 = 1→任务总数 **do** |
| 随机初始化个智能体的Critic网络 |
| **if** = 1 |
| Actor网络被随机初始化 |
| **else** |
| Actor网络的特征提取层从旧网络中加载，线性层随机初始化 |
| 创建旧Actor网络，加载先前任务的特征提取层和线性层 |
| 加载旧任务的对角精度矩阵 |
| 加载特征值矩阵和特征向量矩阵 |
| **end if** |
| **for** 序列 epi = 1→max\_epi **do** |
| 获取所有智能体的初始观测 |
| **for** 时间步 step = 1→STEP **do** |
| 根据当前策略与环境交互获得样本并存储到经验池D中 |
| 从D中随机抽取样本 |
| **for** 智能体  **do** |
| 更新训练网络 |
| 根据式和式分别计算EWC约束和知识蒸馏的梯度 |
| 根据式计算网络参数更新的梯度，更新训练网络 |
| **end for** |
| 根据式计算联合PCA约束的梯度，更新联合特征提取层 |
| 更新Actor、Critic目标网络 |
| **end** **for** |
| 采样联合特征提取层参数值，添加到采样样本集 |
| **end** **for** |
| 计算或合并对角精度矩阵，保存所有智能体Actor网络的参数 |
| 根据采样样本集计算特征值矩阵和特征向量矩阵 |
| **end for** |

## 实验及结果分析

本章实验平台为多智能体粒子环境（Multi-agent Particle Environment，MPE），该平台通过控制二维空间中不同粒子的运动来完成一系列任务，目前已被广泛应用于各类多智能体强化学习算法的验证。

### 实验环境

本章选用MPE库中simpleTag、simplePush和simpleAdversary三个测试场景，并对测试场景的智能体数量和奖励函数等做了一定的修改，分别命名为myTag、myPush和myAdversary，使其更适合验证多智能体持续强化学习算法。智能体训练和算法验证规则为：按顺序在myTag、myPush和myAdversary场景中训练多智能体，由于这3个场景的目标不同，对智能体而言为3个不同的任务，这里分别命名为任务1、任务2和任务3。之后，测试多智能体在当前任务及所有历史任务中的表现，验证本章OwDP算法是否能够明显抑制灾难性遗忘问题，以实现多智能体持续强化学习。

|  |  |
| --- | --- |
| 1. myTag | 1. myPush |
| 1. myAdversary | |

图3.5 OwDP算法验证环境

如图3.5(a)为myTag测试场景：猎物（Prey）需要尽可能躲避捕食者（Predator），以获得环境反馈的正奖励，猎物与捕食者或静止的障碍物（Obstacle）碰撞会受到相应的惩罚，即得到负奖励。在该测试场景中，猎物为我方智能体，采用OwDP算法学习策略，捕食者采用预训练的策略行动。

如图3.5(b)为myPush测试场景：Adversary在自身靠近目标地标Goal时获得环境反馈的正奖励，在GoodAgent靠近Goal时获得负奖励。因此，Adversary需要尽可能阻止GoodAgent靠近Goal。Obstacle为静止的障碍物。在该测试场景中，Adversary为我方智能体，采用OwDP算法学习策略，GoodAgent采用预训练的策略行动。

如图3.5(c) 为myAdversary测试场景：GoodAgent在自身靠近目标地标Goal时获得环境反馈的正奖励，在Adversary靠近Goal时获得负奖励。Obstacle为静止的障碍物。只有GoodAgent知道哪个地标是目标地标Goal，因此，GoodAgent必须学会“分头行动”，掩护Goal来欺骗Adversary。在该测试场景中，GoodAgent为我方智能体，采用OwDP算法学习策略，Adversary采用预训练的策略行动。

### 实验设置

本章设置MPE平台的上述三个场景中双方智能体数量均为3，myTag场景中障碍物数量为2，myPush场景和myAdversary场景中地标数量均为3，其中有1个地标为目标地表。

在myTag场景中，我方智能体（猎物）的奖励为与捕食者的距离之和，即距离越大奖励越高。同时，我方智能体若与捕食者相撞，会得到-10的奖励；若与障碍物相撞，会得到-3的奖励。

在myPush场景中，我方智能体（Adversary）的奖励机制设定如下：奖励与敌方智能体与目标地标的最小距离成正比，即敌方距离目标地标越远，我方智能体奖励越高；奖励函数中还引入了我方智能体与目标地标距离的负值，鼓励我方智能体靠近目标地标。

在myAdversary场景中，我方智能体（GoodAgent）的奖励机制设定如下：当敌方智能体与目标地标的最小距离小于智能体大小的2倍时，我方智能体会得到-1的奖励；当我方智能体与目标地标的距离小于智能体大小的2倍时，我方智能体会得到+5的奖励。

在持续学习训练中，Actor网络和Critic网络的学习率均为0.001，目标网络更新率为0.01，折扣因子为0.95，训练轮次为5000，每轮训练的步长为25。

实验台式机硬件：CPU为i7-12700K，显卡为NVIDIA GeForce RTX 3050，内存为16G；软件：Python 3.9.13，PyTorch 1.13.0，CUDA 11.6，MPE平台 0.0.1。

### 算法有效性检验

为验证本章所提出算法框架的有效性，在三个任务中进行对比实验, 对比的方法包括：

1. OwDP算法：本章提出的多智能体持续强化学习算法；
2. 传统算法1，记为Traditional 1：在MADDPG下，针对每个任务单独训练一个策略网络，可作为验证持续强化学习算法的上限；
3. 传统算法2，记为Traditional 2：在MADDPG下，不采用任何持续学习算法，智能体在学习所有任务时共享一个策略网络；
4. OWL算法：将文献[60]中的单智能体持续强化学习算法：OWL算法，直接应用在多智能体持续强化学习中；
5. 随机动作策略，记为Random：智能体不经学习训练，而是采用随机动作，可作为验证持续强化学习算法的下限。

按照顺序：myTag->myPush->myAdversary，即：任务1->任务2->任务3，对智能体在持续学习训练过程中所得的奖励值进行比较，实验结果如图3.6，图中横坐标为训练轮次，纵坐标为当前训练阶段智能体获得的平均奖励值。

从图3.6(a)看出，在任务1中，OwDP算法经过1500轮训练后开始收敛，奖励达到最大值；OWL算法在训练前期的奖励较低，但经过3000轮训练后开始收敛，奖励逐渐接近其他算法。由于任务1是多智能体学习的第一个任务，在训练任务1时OwDP、Traditional 1与Traditional 2这3种算法等价，因此奖励变化是同一条曲线。Random策略采用随机动作，无需训练，被视为持续学习任务的算法下限。为了方便其他算法与这一下限进行对比，特将其绘制在图中。

图3.6(b)为智能体学习完任务1后继续学习任务2的结果，可以看出，OwDP算法因知识蒸馏和联合PCA的双重约束，初期奖励增长较慢，但经过3000轮训练后收敛，与Traditional 1算法的奖励基本持平；Traditional 2算法是在任务1得到模型的基础上继续训练任务2，没有任何的约束，训练速度较Traditional 1算法慢；OWL算法经过500轮训练即得到较高的奖励，但之后并没有太大提升，与其他算法有一定的差距。

图3.6(c)为智能体学习完任务1和任务2后继续学习任务3的结果，可以看出，Traditional 1算法收敛速度最快，且奖励最高；Traditional 2算法训练速度稍慢，但最终奖励与Traditional 1算法基本相同；OwDP算法训练速度与Traditional 2算法相当，最终奖励稍低于Traditional 1算法和Traditional 2算法；OWL算法训练效果最差，远低于其他算法。

从实验结果可见，本章提出的OwDP算法在新任务中表现良好，这说明该算法能够有效地学习新任务的知识。

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | 1. 任务1 | | 1. 任务2 | | 1. 任务3 | |

图3.6 OwDP算法对比实验中持续训练智能体的平均奖励变化曲线

|  |
| --- |
| 1. 任务1 |
| 1. 任务2 |
| 1. 任务3 |

图3.7 OwDP算法对比实验中智能体执行任务的平均奖励比较

为了进一步对比不同算法的性能，本章采用了持续学习算法中常用的比较方式和展示形式。图3.7分别展示了在持续学习过程中，智能体执行三个任务获得的平均奖励值，横坐标轴表示不同任务训练阶段，如：横坐标取值1代表训练完任务1；纵坐标轴为智能体执行某一任务的平均奖励值。例如，图3.7(a)中黑色圆圈中的点表示，利用Traditional 2算法训练完任务2后执行任务1时得到的奖励。因Traditional 1算法对每个任务各保存一个模型，所以训练新任务后，实际上对旧任务使用的还是旧模型，算法性能不随任务的改变而变化。

从图3.7(a)看出，在训练任务1后，所有算法（除Random策略）执行任务1时均有较高奖励，说明这些算法都能够有效学习新任务。学习新任务的效果在图3.6中已有描述，故在后续图表中，将不再专门描述。在训练任务2时，Traditional 2算法只在旧模型的基础上训练，没有约束地更新参数，故在执行任务1时奖励值直线下降，甚至接近于Random随机策略；OWL算法的奖励值有一定下降，但因使用了持续学习算法，奖励值下降不大；同样，OwDP算法奖励值也有所下降，但高于Traditional 2算法和OWL算法。在训练任务3后，Traditional 2算法执行任务1的奖励值有所回升，但仍与Traditional 1算法有较大差距，这种回升可能是因为任务1和任务3之间存在一定的相似性；OWL算法奖励值严重下降；OwDP算法仍有较高奖励值。

从图3.7(b)看出，在训练任务3后，OwDP算法在执行任务2时仍可得到较高的奖励，而Traditional 2算法和OWL算法奖励值下降较大。

从图3.7(c)看出，在训练任务3后，除OWL算法外，其他算法的智能体在执行任务3时均有较高奖励。

从实验结果可见，本章提出的OwDP算法在旧任务中表现良好，这说明该算法能够有效地保留旧任务的知识。

图3.7中的具体平均奖励值如表3.2所示：

表3.2 OwDP算法对比实验中智能体执行任务的平均奖励统计结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 算法 | 任务1 | | | 任务2 | | 任务3 |
| 阶段1 | 阶段2 | 阶段3 | 阶段2 | 阶段3 | 阶段3 |
| Traditional 1 | **-2.7** | **-2.7** | **-2.7** | **-42.5** | **-42.5** | **39.7** |
| Traditional 2 | -2.7 | -19.4 | -11.6 | -42.9 | -90.6 | 37.8 |
| OwDP | **-2.7** | **-3.9** | **-5.6** | **-44.3** | **-49.1** | **36.9** |
| OWL | -3.2 | -6.3 | -19.2 | -48.4 | -88.4 | 8.5 |
| Random | -24.4 | -24.4 | -24.4 | -102.4 | -102.4 | -17.8 |
| 注：“阶段x”表示训练完任务x后获得的模型 | | | | | | |

从表3.2看出，对于本章提出的OwDP算法，智能体在新任务上性能良好，执行新任务时的平均奖励仅次于为每个任务单独训练一个模型的方法，而高于其他所有对比算法，且在持续学习中，用新任务训练的策略模型去执行旧任务，其平均奖励值下降较小，这表明OwDP算法抑制了持续学习中的灾难性遗忘问题。而OWL算法在新任务中的平均奖励较低，且用新任务训练的策略执行旧任务，其平均奖励值下降明显，这说明OWL算法在顺序学习多任务时会遭遇灾难性遗忘，其难以解决多智能体持续强化学习问题。相较而言，本章提出的OwDP算法性能良好，且不需要扩展模型，参数量远少于为每个任务单独训练一个模型的方法，为多智能体持续强化学习问题提供了一种的解决方法。

### 消融实验

本章提出的OwDP算法包含了两种针对多智能体持续强化学习的改进机制：知识蒸馏和联合PCA约束。为研究这两种机制对算法性能的影响，设计了消融实验，实验设置如表3.3所示。该实验通过在OwDP算法上去除某一种改进机制，比较测试结果。

表3.3 OwDP算法消融实验设置

|  |  |  |
| --- | --- | --- |
| 算法 | 知识蒸馏 | 联合PCA约束 |
| OwDP | ⚪ | ⚪ |
| OwDP-D | × | ⚪ |
| OwDP-P | ⚪ | × |
| 注：⚪表示包含，×表示不包含 | | |

如图3.8为在消融实验中，智能体在不同算法的持续训练中所得到的平均奖励值。因OwDP、OwDP-D和OwDP-P算法对任务1的训练过程是相同的，如图3.6(a)的OwDP曲线，故不再单独绘制。图3.8(a)为智能体训练完任务1再训练任务2的奖励结果，在5000轮训练后，OwDP算法与其他两种算法能够达到基本相同的奖励。图3.8(b)为智能体训练完任务1和任务2再训练任务3的奖励结果，在5000轮训练后，三种算法同样奖励相当。

实验结果表明，添加更多的约束不会导致OwDP算法的新任务学习能力下降，与OwDP-D和OwDP-P算法相比，它们均能实现相同的效果。

|  |
| --- |
| 1. 任务2 |
| 1. 任务3 |

图3.8 OwDP算法消融实验中持续训练智能体的平均奖励变化曲线

图3.9展示了消融实验中智能体执行任务时所获得奖励的变化，与图3.7相同，图3.9中横坐标为任务训练阶段，纵坐标为在不同训练阶段的策略模型执行任务的奖励。从图3.9(a)看出，在训练完任务2和任务3后，三个算法执行任务1的奖励均有一定程度的降低，但OwDP算法降低最少。从图3.9(b)看出，在训练完任务2后，OwDP-P算法和OwDP-D算法执行任务2的奖励均高于OwDP算法，但在训练完任务3后，两种算法执行任务2的奖励下降都比较严重。从图3.9(c)看出，在训练完任务3后，执行任务3奖励最高的是OwDP-P算法和OwDP算法。可见，OwDP算法在旧任务中表现最好。

|  |
| --- |
| 1. 任务1 |
| 1. 任务2 |
| 1. 任务3 |

图3.9 OwDP算法消融实验中智能体执行任务的平均奖励比较

图3.9中的具体平均奖励值如表3.4所示：

表3.4 OwDP算法消融实验中智能体执行任务的平均奖励统计结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 算法 | 任务1 | | | 任务2 | | 任务3 |
| 阶段1 | 阶段2 | 阶段3 | 阶段2 | 阶段3 | 阶段3 |
| OwDP | **-2.7** | **-3.9** | **-5.6** | -44.3 | **-49.1** | 36.9 |
| OwDP-P | -2.7 | -4.9 | -7.9 | **-43.1** | -57.0 | **37.4** |
| OwDP-D | -2.7 | -7.7 | -9.2 | -44.1 | -50.4 | 33.4 |
| 注：“阶段x”表示训练完任务x后获得的模型 | | | | | | |

从表3.4看出，本章提出的OwDP算法中的两种持续学习机制均能够显著提高算法性能，使智能体在新任务和旧任务上表现更好。消融实验结果表明，OwDP算法适用于解决多智能体持续强化学习问题。

### 行为分析

在使用OwDP算法按顺序训练三个任务后，让智能体分别在3个场景中执行对应任务，以从行为层面全面检验算法的实际效果。

（1）myTag场景：如图3.10(a)，我方智能体（绿色粒子）依然能够正确执行任务1，如：我方智能体能够顺利躲避敌方（红色粒子）的追击；在被两个敌方追到场景边缘这种极端情况下，我方智能体能从两敌方中间穿过并逃脱。

（2）myPush场景：如图3.10(b)，我方智能体（红色粒子）依然能够正确执行任务2，如：对于靠近目标地标的敌方（绿色粒子），我方智能体能够选择最佳的方向将其推开，如图中圆圈部分所示，而其他智能体守卫目标地标的其他方向。

（3）myAdversary场景：如图3.10(c)，我方智能体（绿色粒子）依然能够在记住任务1、2的情况下，正确执行任务3，如：我方智能体成功将敌方（红色粒子）欺骗到错误的地标（黑色障碍物）。

可以看到，在顺序训练所有任务后，我方多智能体在每个历史任务中仍然能够做出正确的决策，并准确执行任务。这表明OwDP算法可以很好地平衡模型的稳定性和可塑性，能够在有效学习新任务的同时保留旧任务的知识，从而实现多智能体持续强化学习。

|  |  |  |
| --- | --- | --- |
| 1. myTag | 图表, 散点图, 气泡图  描述已自动生成   1. myPush | 1. myAdversary |

图3.10 OwDP算法多智能体在环境中的表现

## 本章小结

针对多智能体持续强化学习的问题背景，本章引入持续学习中的正则化方法，基于MADDPG框架提出了OwDP算法。该算法对OWL算法进行了改进，并引入了知识蒸馏和联合PCA约束等正则化方法，限制了对旧任务重要参数的大幅变化，提高了多智能体执行多任务的效果，并有效地解决了多智能体持续强化学习的灾难性遗忘问题。

接着，在myTag、myPush和myAdversary三个强化学习场景中进行了对比实验。实验结果表明，当多智能体按顺序学习三个场景的任务时，OwDP算法相较于其他对比算法具有明显优势，在一定程度上可以保持多智能体在旧任务的协调性，并保留旧知识。

最后，在上述三个场景进行了消融实验。结果证明，OwDP算法的两种机制都不同程度地增强了算法的性能，能够最大限度保留旧任务的知识，兼顾了模型的稳定性和可塑性。

# 

# 第四章 基于经验重放的多智能体持续强化学习算法

传统的持续强化学习算法无法有效应对多智能体场景，第三章提出了一种基于正则化的多智能体持续强化学习算法很好地解决了这一问题。除了正则化方法，还有一种常用的持续学习方法：经验重放，它通过存储先前任务的样本，并在新任务学习过程中重放先前样本，来减轻灾难性遗忘。相比于正则化方法，经验重放方法需要更多的存储空间，但往往也有着更好的效果。本章引入经验重放方法，提出了一种基于经验重放的多智能体持续强化学习算法。

## 问题分析

在2.3.2节中介绍的OCS算法能够自适应选择最能代表当前任务的样本，从而保留该任务的知识。然而，简单地将该算法扩展到多智能体场景，忽略了样本对多个智能体重要性不同的问题，这可能导致某些智能体因样本不足而无法充分训练，从而忘记旧任务的知识，发生灾难性遗忘。

## 基于经验重放的多智能体持续强化学习算法

在MADDPG框架的基础上，本章将多头机制应用到策略网络，并结合持续学习中的经验重放方法，提出了一种基于经验重放的多智能体持续强化学习算法。多头机制在LwF算法中已有介绍，将智能体的Actor网络划分为特征提取层和线性层，其中，所有任务共享特征提取层，而拥有单独的线性层，如图2.8。

本章基于MADDPG框架，将持续学习中OCS算法扩展到多智能体领域，结合个体小批次度量和团队小批次度量选择旧任务样本，平衡每个智能体的样本数量，并在新任务训练时，通过优先经验重放（Prioritized Experience Replay, PER）策略重放这些样本以减轻遗忘，提出MADDPG-OP（MADDPG with OCS & PER）算法，保证多个智能体均能得到充分训练。

### 混合小批次度量

在2.3.2节中的OCS算法可以通过计算小批次相似性、样本多样性和核心集亲和性，选择保存最能代表当前任务，且对历史任务影响较小的样本，从而提高存储效率，保留该任务的知识。但在多智能体强化学习中，模型拥有多个Actor网络，OCS算法难以选出对多个网络同时有效的样本。本章提出为每个智能体维护一个核心集，命名为，其中，为我方智能体的数量。核心集样本的选择公式分为两部分，一个是个体小批次度量，设为；一个是团队小批次度量，设为。定义混合小批次度量的计算公式如下：



其中，为任务的小批次样本集，为小批次样本集的第个样本的状态及该状态下的动作，为从第个智能体先前任务的核心集中采样获得的核心子集，为第个智能体的混合小批次度量，用于衡量对当前任务及历史任务的表达程度，为超参数，用于平衡个体小批次度量和团队小批次度量。

个体小批次度量可以评估给定样本对第个智能体在任务及之前所有任务样本的代表程度。个体小批次度量如下式所示：



其中，为将小批次样本集去除后的样本子集，表示样本在小批次样本集中对第个智能体的个体相似性，表示样本在小批次样本集中对第个智能体的个体多样性，表示在核心子集中对第个智能体的个体亲和性，是平衡旧任务和新任务的超参数。式的前两项代表样本对第个智能体在当前任务样本中的代表程度，第三项代表样本对第个智能体在历史任务样本中的代表程度。

个体相似性用来评估样本与样本集之间小批次级别的相似性，它测定给定样本在每个训练步对第个智能体在任务样本中的代表程度。个体相似性如下式所示：



其中，表示样本在每个训练步对第个智能体策略网络的梯度向量，表示小批次样本集在每个训练步对第个智能体策略网络的平均梯度向量，以上梯度向量可将样本代入式获得。

个体多样性用来评估样本与样本子集之间小批次级别的不相似性，它衡量给定样本在每个训练步对第个智能体策略网络的梯度向量与其他样本的相异度，以此选择更加多元的样本。个体多样性如下式所示：



个体亲和性用来评估样本与第个智能体的核心子集之间小批次级别的相似性，它衡量给定样本在每个训练步对第个智能体策略网络的梯度向量与核心子集样本的相似度，以此选择更加接近核心集的样本。个体亲和性如下式所示：



其中，表示核心子集在每个训练步对第个智能体策略网络的平均梯度向量。

而团队小批次度量可以评估给定样本在每个训练步对除第个智能体外其他所有智能体训练任务及之前所有任务样本的代表程度。团队小批次度量如下式所示：



其中，表示样本在小批次样本集中对除第个智能体外其他所有智能体的团队相似性，表示样本在小批次样本集中对除第个智能体外其他所有智能体的团队多样性，表示样本在核心子集中对除第个智能体其他所有智能体的团队亲和性，是平衡旧任务和新任务的超参数。式的前两项代表样本对除第个智能体之外其他所有智能体在当前任务样本中的代表程度，第三项代表样本对除第个智能体之外其他所有智能体在历史任务样本中的代表程度。

团队相似性用来评估样本在每个训练步对除第个智能体之外其他所有智能体在任务样本中的代表程度。团队相似性如下式所示：



其中，表示样本对第个智能体策略网络的梯度向量，表示小批次样本集对第个智能体策略网络的平均梯度向量。

团队多样性用来评估样本在每个训练步对除第个智能体之外其他所有智能体策略网络的梯度向量与其他样本的相异度。团队多样性如下式所示：



团队亲和性用来评估样本在每个训练步对除第个智能体之外其他所有智能体策略网络的梯度向量与核心子集样本的相似度。团队亲和性如下式所示：



综上所述，训练任务的第个智能体的核心集样本选择公式可以表示为：



其中，表示选择最大的个样本，是小批次样本数，为当前训练步选出的个核心集样本。

在智能体学习过程中，需要为每个智能体维护一个核心集，以第个智能体的核心集选择为例：对任务的小批次样本集中的每个样本，使用个体小批次度量评估其对智能体在所有任务训练的表达程度，从小批次样本集中抽取对智能体训练最有利的样本；同时，使用团队小批次度量评估其对其他所有智能体（除第个智能体）在所有任务中训练的表达程度，在选择样本时兼顾其他智能体的训练，最大限度地提高样本利用率，降低样本存储量。

相较于为所有智能体维护一个核心集的方法（以下简称为OCS-E方法），MADDPG-OP算法能够明显减少核心集的存储空间。假设在OCS-E方法中，为了保证智能体能够充分保留旧任务的知识，需要存储个样本。MADDPG-OP算法提出了一个不同的策略，即将样本按智能体进行分配和存储，总共保存个核心集。然而，它并非简单地将个样本平均分配到个核心集中，使得每个核心集存储个样本。在MADDPG-OP算法中，每个智能体的核心集样本选择不仅考虑自身的训练，还兼顾了其他智能体的需求。这样，每个核心集的选择都更加精细和有针对性，从而可以降低样本的存储量，但达到与OCS-E方法相同甚至更好的效果。同时，在选择核心集样本的过程中，可能会出现两个或两个以上的智能体选择相同的样本存入自身核心集的情况。为了避免不必要的空间占用，MADDPG-OP算法采取了一种优化策略：在一个智能体的核心集中实际存储该样本，而其他智能体的核心集只存储指向该样本的指针，由此提高了存储效率。此外，OCS-E方法面临样本选择不均衡的问题，这可能导致每个智能体的训练效果不尽相同，进而在旧任务上表现下降。MADDPG-OP算法通过确保每个智能体存储相同数量的样本，有效避免了这一问题，从而保证了各智能体在旧任务上的表现稳定。

### 优先经验重放

在建立核心集后，先前任务的信息已经被保留下来。如何使用核心集的样本也是一个重要的问题。一个简单的方法是对核心集进行均匀分布采样，即每个样本有相同的概率被选中。然而，这忽略了样本的重要程度，在很多场景下，均匀分布采样并不能高效地利用样本。例如，图4.1是悬崖漫步环境。该环境包含个状态，每个状态下智能体都可以采取两个动作：正确的动作和错误的动作，且两种动作的概率均为0.5。智能体执行正确的动作会转移到下一状态（用实线表示），执行错误的动作会导致游戏终止且回到初始状态（用虚线表示）。只有智能体在状态执行正确的动作转移到状态1时，才能获得奖励值1，其他状态转移的奖励均为0。如果智能体从状态1出发，随机游走，那么智能体获得奖励值的概率将为。在这种情况下，如果使用均匀分布采样，并用这些样本去训练智能体，可能会导致训练时间过长。因此，可以尝试使用非均匀采样代替均匀采样，即计算经验回放池中的每个样本的重要性，然后根据重要性抽取样本，并将这些样本用于智能体的训练。



图4.1 悬崖漫步环境

4.2.1节提出在MADDPG框架下，在旧任务中采用混合小批次度量选出最契合于小批次的样本放入核心集。在新任务训练时，重放核心集中旧任务的样本。本节提出采用PER策略来抽取核心集中的样本，用样本在新任务模型的TD误差的绝对值来判断其重要性，并根据重要性从高到低重放这些样本，的定义如下式：



其中，是训练网络，是目标网络，是目标网络，。

设智能体当前学习的新任务为，在训练过程中要重放的旧任务的第个智能体的核心集。在获取核心集的所有样本的TD误差后，根据该值计算样本在经验重放时被选中的概率。设为核心集中任务的样本数，则该样本集中第个样本被选中的概率的计算公式如下：



其中，为根据TD误差对中的样本排序后，第个样本在中的序号，可见越大，被选中的概率越高。从核心集中抽取样本来训练旧任务，目标是使尽可能小，因此这种重要性定义是合理的。

无论是MADDPG算法的经验回放池，还是OCS算法的核心集，都是为了解决数据相关性问题，使样本尽可能独立同分布。然而，在引入PER策略后，核心集中的样本分布又引入了新的偏差，增加了样本间的依赖性。为了弥补这种偏差，可以引入重要性采样权重，对Critic训练网络的学习率加以调整，定义如下式：



其中，为补偿因子，可以取之间的值，当时，PER策略的误差得到完全的补偿。采用梯度上升法更新Critic训练网络，对网络学习率添加的权重项，更新方式如下：



其中，为Critic训练网络学习率，为Critic训练网络参数。可以看到，在样本的抽样概率较大时，网络对该样本的学习率越小。

在核心集中，每个样本都会额外存储其对应的TD误差，利用该值根据式可以计算该样本的选中概率。然后，根据这个概率分布从核心集中抽取样本，用于训练模型后，会得到该样本新的TD误差，之后用该值更新原来的。

但在初始阶段，即新任务刚开始训练时，核心集的样本没有在新网络中训练过，也没有对应的TD误差。这时，可以将其设置为极大值，以保证每个样本至少被抽取一次。之后，再根据抽取样本在新网络中的TD误差更新。

可以发现，在新任务刚开始训练时，因未对核心集中旧任务样本进行重放，因此每个样本的TD误差均为最大值，可不存储。而在新任务训练完成后，样本的TD误差不会用在后续任务的训练中，因此也不必存储。综上，可存放在内存中，在使用完毕后自动销毁，从而达到节省空间的目的。

### 算法网络更新

如图4.2所示，MADDPG-OP算法采用中心化训练去中心化执行框架，每个智能体Critic网络的输入为所有智能体的局部观测，而Actor网络的输入为智能体自身的局部观测。在训练时，每个训练步从经验回放池采样一批样本，将这些数据输入Actor网络和Critic网络的同时，通过混合小批次度量选择最能代表当前任务和历史任务训练的样本，将其放入核心集，并在后续任务训练中重放核心集样本。同时，如果当前任务不是顺序训练的第一个任务，在训练阶段，还会从每个智能体的核心集中按照PER策略抽取样本，传递给对应智能体，以减轻智能体对旧任务的遗忘。

在MADDPG-OP算法中，通过引入经验重放机制和PER策略，实现了多智能体的持续强化学习。与传统强化学习的经验回放不同，持续学习的经验重放机制更加注重保留智能体在旧任务中获得的知识，避免灾难性遗忘。经验重放包含两个阶段：将样本放入核心集以及从核心集取出样本。每个智能体都维护一个独立的核心集，且每个核心集的容量相等。以第个智能体为例：如图4.3，每个任务在核心集中分配到相同大小的空间。在训练任务时，从强化学习算法的经验回放池中随机抽取样本，用于智能体在新任务中的训练，并通过混合小批次度量选择最能代表当前任务且对历史任务影响最小的样本，将其存入该智能体核心集中对应任务的空间中。这里，为下一联合观测对应的联合最优动作，存储的目的在于避免在新任务中训练额外的Actor网络，从而显著降低训练时间。实验结果表明，用代替联合最优动作不会降低智能体在旧任务上的性能。之后，通过PER策略，从核心集中任务1到任务的样本中按重要性抽取样本，用于智能体在旧任务中的训练，并将新的TD误差返回给PER策略。



图4.2 MADDPG-OP算法架构

需要注意，重要性抽样同样是针对智能体而言的。例如，对第个智能体抽取的样本为，按照混合小批次度量机制，该样本依然要交给所有智能体去训练，以充分利用核心集样本，确保智能体在旧任务上的性能。但PER策略只需记录第个智能体的TD误差，因为对于第个智能体的核心集，只需关注核心集样本对第个智能体的重要性，而其他智能体的训练由对应核心集负责。

|  |  |  |  |
| --- | --- | --- | --- |
| 核心集 | | | |
| 任务1 |  | 任务2 |  |
|  |  |
|  |  |
|  |  |
|  |  | 任务 |  |
|  |  |
|  |  |
|  |  |

图4.3 经验重放机制

在图4.3中，表示该核心集中保存的第个任务的第个样本的智能体联合观测，为核心集为每个任务保存的样本数。在该联合观测下，多智能体共同执行联合动作，并从环境中获得奖励。随后，环境转移到下一状态，多智能体获得新的联合观测。基于新的联合观测，Actor网络指导多智能体生成新的联合动作。

智能体的训练可以分为两个阶段：强化学习阶段和持续学习阶段。如图4.4，在强化学习阶段，从MADDPG框架的经验回放池中选取样本，这些样本帮助Critic网络和Actor网络更新，进而促进智能体学习新任务的相关知识；在持续学习阶段，从核心集中按重要性抽取样本，这些样本的作用在于保留旧任务的知识。两个阶段交替进行，共同构成多智能体持续强化学习的完整体系。

设当前学习任务，在强化学习阶段，智能体采用传统强化学习算法更新网络参数，一旦经验回放池积累了足够的样本，就从中随机选取小批次样本，并传递给Critic网络和Actor网络。Critic网络和Actor网络的损失函数分别如式和式。在每个训练步中，需要记录这一小批次样本集在第个智能体Actor网络的平均梯度信息，以及其中每个样本单独的梯度信息，从而计算样本对第个智能体小批次相似性和多样性。除此之外，从核心集中随机选取一个小批次的子集，将其输入Actor网络中。Actor网络基于这些输入生成对应的动作信息，并传递给Critic网络进行评估。Critic网络在完成评估后，会将梯度信息回传给Actor网络后，记录此时的梯度信息。接着，由、和计算小批次样本集中每个样本的混合小批次度量，并从中选出度量值最大的个样本，将它们加入核心集中为任务预留的空间，以便在后续任务使用。



图4.4 核心集样本的生成与使用

如图4.4，在持续学习阶段，当学习任务时，每个训练步需要执行次如下步骤来使用核心集样本：

1. 设当前为第次步骤。根据式计算核心集中任务的所有样本的选中概率，并基于这个概率抽取样本，样本的数据构成为。
2. 为每个智能体训练一个额外的Critic网络（包含训练网络和目标网络），称旧Critic网络，该网络用于评估Actor网络在任务中的表现，第个智能体旧Critic训练网络的损失函数如下式：



其中，为第个智能体的旧Critic训练网络，为其网络参数，为旧Critic目标网络，为其网络参数，为多智能体在任务中联合观测状态下执行联合动作后第个智能体获得的环境奖励。

1. 计算样本的TD误差，设当前样本为从第个智能体核心集抽取的样本，则的计算如式。为了简化表达，图4.4中只标出了第1个智能体的计算。



1. 将第个智能体Actor网络的特征提取层和负责旧任务的线性层组合，得到的网络称旧Actor网络，记为：。将步骤（1）抽取的样本输入旧Actor网络，在旧Critic网络的指导下，更新和，以确保网络在处理旧任务时仍能保持良好的性能。第个智能体旧Actor网络学习的目标函数如下式：



其中，，。

### 算法流程

MADDPG-OP算法模型在顺序学习多个任务时，每个任务都遵循同一个训练流程，该流程主要包括初始化、回放池经验收集、模型训练、核心集经验收集和模型保存五个阶段。

在初始化阶段，算法首先初始化环境以及智能体的Actor网络和Critic网络，并为经验回放池和当前任务的核心集分配空间，做好训练前的准备。

进入回放池经验收集阶段，多智能体从环境中获得局部观测信息并生成联合动作传递给环境，环境响应这些动作后转移到新的状态，并反馈给智能体新的局部观测信息和奖励。这些信息被组合为并存入经验回放池。在回放池经验收集阶段初期，智能体只与环境交互产生数据，而不进行模型训练。之后，与模型训练和核心集经验收集阶段同步进行。

当经验回放池中积累了足够的数据后，算法进入模型训练阶段。在这一阶段，需要对所有智能体Critic网络和Actor网络的参数进行更新。算法从经验回放池随机抽取小批次样本，并采用传统多智能体强化学习方法训练智能体，以学习新任务。设当前任务为顺序学习的第个任务，如果，算法还会从核心集中按重要性抽取历史任务的样本，并按任务训练Actor网络及其对应的线性层，以保留旧任务的知识。同时，计算该样本新的TD误差并更新原来的。

核心集经验收集与模型训练是同步进行的。在每个训练步中，计算小批次样本集中每个样本对第个智能体的混合小批次度量，并选择度量值最大的个样本加入该智能体核心集中任务的空间。这个过程同样需要从核心集中抽取样本，但与模型训练阶段有所不同。在本阶段，是从核心集整体中抽取样本，而非按任务进行抽取。同时，抽取策略也由模型训练阶段的按重要性抽取改为随机抽取，因为本阶段抽取的样本主要用于计算混合小批次度量，以选择最接近核心集的样本，而不是直接用于更新网络。

最后，在模型训练达到设定的最大回合数max\_epi后，算法保存模型参数并结束训练。

MADDPG-OP算法的执行流程如图4.5，可以看到，在整个流程中共有两次从核心集中抽取样本。其中，“从核心集抽取核心子集”采用随机策略，目的是计算新任务样本的混合小批次度量，之后选择度量值最高的样本存入核心集；而“通过PER策略抽取任务的样本”采用按重要性抽取样本的策略，目的是重放先前任务的样本以减轻遗忘。



图4.5 MADDPG-OP算法执行流程

MADDPG-OP算法流程如下表所示：

表4.1 MADDPG-OP算法

|  |
| --- |
| 算法名称：MADDPG-OP算法 |
| 初始化核心集 |
| **for** 任务序号 = 1→任务总数 **do** |
| 初始化环境，随机初始化个智能体的Critic网络。 |
| **if** |
| Actor网络被随机初始化 |
| **else** |
| Actor网络的特征提取层从旧网络中加载，线性层随机初始化 |
| 创建额外的线性层，参数从中加载 |
| **end if** |
| **for** 序列 epi = 1→max\_epi **do** |
| 获取所有智能体的局部观测 |
| **for** 时间步 step = 1→STEP **do** |
| 根据当前策略与环境交互获得样本并存储到经验池D中 |
| 从D中随机抽取样本子集 |
| **for** 智能体 = 1→ **do** |
| 根据式和式更新训练网络和训练网络 |
| 从当前智能体的核心集中随机抽取核心子集 |
| 根据式计算样本子集中每个样本的混合小批次度量 |
| 将度量值最大的个样本存入当前智能体核心集中任务的预留空间 |
| **end for** |
| **for** 旧任务序号 = 1→ **do** |
| 从核心集中按重要性抽取任务的样本 |
| 计算该样本的TD误差并更新 |
| **for** 智能体 = 1→ **do** |
| 更换训练网络的线性层，初始化额外的网络。 |
| 根据式和式更新网络和额外的网络 |
| **end for** |
| **end for** |
| **end for** |
| **end for** |
| 保存所有智能体Actor网络的参数 |
| **end for** |

## 实验及结果分析

为了简化实验设置，本章选择了与第三章相同的实验平台、实验环境和智能体状态动作设计。这样的一致性设置有助于减少其他因素对实验结果的影响，更加准确地评估MADDPG-OP算法的效果，并为后续算法的研究和改进提供一个基准。

### 算法有效性检验

为验证本章所提出算法框架的有效性，在myTag、myPush和myAdversary三个任务中进行对比实验，对比的方法包括：

1. MADDPG-OP算法：本章提出的多智能体持续强化学习算法；
2. 传统算法1，记为Traditional 1：在MADDPG下，针对每个任务单独训练一个策略网络，可作为验证持续强化学习算法的上限；
3. 传统算法2，记为Traditional 2：在MADDPG下，不采用任何持续学习算法，智能体在学习所有任务时共享一个策略网络；
4. OCS算法：将文献[50]中的持续学习算法：OCS算法，直接应用在MADDPG下的多智能体持续强化学习中；
5. 随机动作策略，记为Random：智能体不经学习训练，而是采用随机动作，可作为验证持续强化学习算法的下限。

按照顺序：myTag->myPush->myAdversary，即：任务1->任务2->任务3，对智能体在持续学习训练过程中所得的奖励值进行比较，实验结果如图4.6，图中横坐标为训练轮次，纵坐标为当前训练阶段智能体获得的平均奖励值。

从图4.6(a)可以看出，由于任务1是智能体学习的第一个任务，在训练任务1时MADDPG-OP、OCS、Traditional 1与Traditional 2这4种方法等价，因此奖励变化是同一条曲线。Random策略采用随机动作，无需训练，被视为持续学习任务的算法下限。为了方便其他算法与这一下限进行对比，特将其绘制在图中。

图4.6(b)为智能体学习完任务1后继续学习任务2的结果，可以看出，MADDPG-OP算法和OCS算法因在学习任务2时重放任务1的样本，初期奖励增长较慢。但经过3000轮训练后，MADDPG-OP算法开始收敛，奖励达到最大值，与Tradition 1算法的奖励基本持平；OCS算法在训练初期奖励高于MADDPG-OP算法，但在训练2800轮左右被反超；Tradition 1算法、Tradition 2算法和Random算法与第三章相同，不再描述。

|  |
| --- |
| 1. 任务1 |
| 1. 任务2 |
| 1. 任务3 |

图4.6 MADDPG-OP算法对比实验中持续训练智能体的平均奖励变化曲线

图4.6(c)为智能体学习完任务1和任务2后继续学习任务3的结果，可以看出，MADDPG-OP算法和OCS算法在训练速度上大致相当，训练初期奖励增长较慢。两者的表现更接近于Tradition 2算法，但最终奖励稍低于Tradition 1算法和Tradition 2算法。

从实验结果可见，本章提出的MADDPG-OP算法在新任务中表现良好，这说明该算法能够有效地学习新任务的知识。

为了进一步对比不同算法的性能，本章采用了持续学习算法中常用的比较方式和展示形式。图4.7分别展示了在持续学习过程中，智能体执行三个任务获得的平均奖励值。图4.7中横坐标轴表示不同任务训练阶段，如：横坐标取值1代表训练完任务1；纵坐标轴为智能体执行某一任务的平均奖励值。

从图4.7(a)看出，在训练任务1后，所有算法（除Random策略）执行任务1时均有较高奖励，说明这些算法都能够有效学习新任务。学习新任务的效果在图4.6中已有描述，故在后续图表中，将不再专门描述。在训练任务2时，OCS算法对每个智能体保留的样本存在不均衡的现象，导致即使部分智能体在旧任务中表现良好，但仍有部分智能体对旧任务的性能出现显著下降，进而使得平均奖励值有一定降低；而MADDPG-OP算法注重在旧任务采样时保持每个智能体样本的均衡性，因此所有智能体在旧任务中均有良好表现，平均奖励值也接近于Tradition 1算法。在训练任务3后，OCS算法在执行任务1时的平均奖励值出现了严重下降，而MADDPG-OP算法仍有较高的奖励值，这进一步验证了样本均衡性在经验重放方法中的优势。

从图4.7(b)看出，在训练任务3后，MADDPG-OP算法在执行任务2时仍可得到较高的奖励值，而OCS算法奖励值出现了较大下降。

从图4.7(c)看出，在训练任务3后，所有算法（除Random策略）的智能体在执行任务3时均有较高奖励值。

从实验结果可见，本章提出的MADDPG-OP算法在旧任务中表现良好，这说明该算法能够有效地保留旧任务的知识。

|  |
| --- |
| 1. 任务1 |
| 1. 任务2 |
| 1. 任务3 |

图4.7 MADDPG-OP算法对比实验中智能体执行任务的平均奖励比较

图4.7中的具体平均奖励值如表4.2所示：

表4.2 MADDPG-OP算法对比实验中智能体执行任务的平均奖励统计结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 算法 | 任务1 | | | 任务2 | | 任务3 |
| 阶段1 | 阶段2 | 阶段3 | 阶段2 | 阶段3 | 阶段3 |
| Traditional 1 | **-2.7** | **-2.7** | **-2.7** | **-42.5** | **-42.5** | **39.7** |
| Traditional 2 | -2.7 | -19.4 | -11.6 | -42.9 | -90.6 | 37.8 |
| MADDPG-OP | **-2.7** | **-3.6** | **-5.0** | **-44.4** | **-47.5** | **35.5** |
| OCS | -2.7 | -9.1 | -15.6 | -44.7 | -68.4 | 35.6 |
| Random | -24.4 | -24.4 | -24.4 | -102.4 | -102.4 | -17.8 |
| 注：“阶段x”表示训练完任务x获得的模型 | | | | | | |

从表4.2看出，对于本章提出的MADDPG-OP算法，在强化学习中，智能体在执行新任务时获得的平均奖励接近为每个任务单独训练一个模型的方法。此外，在任务1和任务2中MADDPG-OP算法的平均奖励高于OCS算法，只有在任务3中稍差于OCS算法。这表明MADDPG-OP算法的持续学习策略基本没有影响新任务的学习，智能体仍然能够较好学习新任务。在持续学习中，用新任务训练的策略模型去执行旧任务，MADDPG-OP算法智能体获得的平均奖励值下降较小，这表明MADDPG-OP算法抑制了持续学习中的灾难性遗忘问题的发生。而OCS算法在新任务中的平均奖励与MADDPG-OP算法基本持平，但用新任务训练的策略执行旧任务时，其平均奖励值下降明显，严重低于算法上限，这说明OCS算法没有明显抑制灾难性遗忘，难以解决多智能体持续强化学习问题。相较而言，本章提出的MADDPG-OP算法在新任务和旧任务中均表现良好，且不需要扩展模型，参数量远少于为每个任务单独训练一个模型的方法，这为多智能体持续强化学习问题提供了一种的解决方法。

### 消融实验

本章提出的MADDPG-OP算法包含了两种针对多智能体持续强化学习的改进机制：混合小批次度量和PER策略。为研究这两种机制对算法性能的影响，设计了消融实验，实验设置如表4.3所示。该实验通过在MADDPG-OP算法上去除某一种改进机制，比较测试结果。去除混合小批次度量机制，意味着智能体在选取核心集样本时采用随机策略，不再考虑样本与小批次的关系，且所有智能体共享一个核心集；而去除优先经验重放机制，表示智能体在重放旧任务数据时，忽略了样本对智能体训练的重要性，而是采用随机策略选取样本。

表4.3 MADDPG-OP算法消融实验设置

|  |  |  |
| --- | --- | --- |
| 算法 | 混合小批次度量 | 优先经验重放 |
| MADDPG-OP | ⚪ | ⚪ |
| MADDPG-O | ⚪ | × |
| MADDPG-P | × | ⚪ |
| 注：⚪表示包含，×表示不包含 | | |

如图4.8为在消融实验中，智能体在不同算法的持续训练中所得到的平均奖励值。因MADDPG-OP、MADDPG-O和MADDPG-P算法对任务1的训练过程是相同的，如图4.6(a)的MADDPG-OP曲线，故不再单独绘制。

|  |
| --- |
| 1. 任务2 |
| 1. 任务3 |

图4.8 MADDPG-OP算法消融实验中持续训练智能体的平均奖励变化曲线

|  |
| --- |
| 1. 任务1 |
| 1. 任务2 |
| 1. 任务3 |

图4.9 MADDPG-OP算法消融实验中智能体执行任务的平均奖励比较

图4.8(a)为智能体训练完任务1继续训练任务2的奖励结果，三种算法均在3000轮左右时达到收敛，且在5000轮训练后达到基本相同的奖励。这说明，不同的样本选取和重放方式对于新任务学习的影响并不显著。

图4.8(b)为智能体训练完任务1和任务2后继续训练任务3的奖励结果，可以看到，MADDPG-O算法在4000轮左右达到收敛，比其他两种算法稍早，但在5000轮训练后达到基本相同的奖励。

实验结果表明，改进经验重放样本的保存和重放策略不会导致MADDPG-OP算法的新任务学习能力下降，与MADDPG-O和MADDPG-P算法相比，它们能实现相同的效果。

图4.9展示了消融实验中智能体执行任务时所获得奖励的变化，图4.9中横坐标为任务训练阶段，纵坐标为在不同训练阶段的策略模型执行任务的奖励。

从图4.9(a)看出，在训练完任务2和任务3后，三个算法执行任务1的奖励均有一定程度的降低，但MADDPG-OP算法降低最少。

从图4.9(b)看出，在训练完任务2后，MADDPG-P和MADDPG-OP算法执行任务2的奖励较高，MADDPG-O算法奖励稍低；在训练完任务3后，MADDPG-P算法执行任务2的奖励下降较大，MADDPG-OP算法奖励最高，MADDPG-O算法稍低；

从图4.9(c)看出，在训练完任务3后，三个算法执行任务3的奖励接近，其中MADDPG-OP算法最高。

图4.9中的具体平均奖励值如表4.4所示：

表4.4 MADDPG-OP算法消融实验中智能体执行任务的平均奖励统计结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 算法 | 任务1 | | | 任务2 | | 任务3 |
| 阶段1 | 阶段2 | 阶段3 | 阶段2 | 阶段3 | 阶段3 |
| MADDPG-OP | **-2.7** | **-3.6** | **-5.0** | -44.4 | **-47.5** | **35.5** |
| MADDPG-O | -2.7 | -4.9 | -8.5 | -45.9 | -48.4 | 34.8 |
| MADDPG-P | -2.7 | -7.1 | -12.5 | **-43.9** | -52.5 | 35.2 |
| 注：“阶段x”表示训练完任务x后获得的模型 | | | | | | |

从表4.4看出，本章提出的MADDPG-OP算法中的两种持续学习机制均能够显著提高算法性能，使智能体在新任务和旧任务上表现更好。消融实验结果表明，MADDPG-OP算法适用于解决多智能体持续强化学习问题。

### 行为分析

在使用MADDPG-OP算法按顺序训练三个任务后，让智能体分别在3个场景中执行相应任务，以从行为层面全面检验算法的实际效果。

如图4.10，分别展示了在顺序训练完任务1、任务2和任务3后，多智能体在各个任务中的表现。从图中可以清楚地看到，在每个任务中，智能体均能做出正确的判断，并准确执行任务。这表明MADDPG-OP算法能够很好地平衡模型的稳定性和可塑性，有效学习新任务，同时保留旧任务的知识，从而实现多智能体持续强化学习。

|  |  |  |
| --- | --- | --- |
| 1. myTag | 1. myPush | 1. myAdversary |

图4.10 MADDPG-OP算法多智能体在环境中的表现

## 持续强化学习算法对比

本文实际上提出了两种解决多智能体持续强化学习中灾难性遗忘问题的方法：OwDP算法和MADDPG-OP算法。OwDP算法采用正则化方法，在每个任务训练结束时保存EWC约束系数（即对角精度矩阵）和联合PCA约束系数，并在训练新任务时添加这些约束，同时结合知识蒸馏，以确保多智能体在旧任务上的协同性。MADDPG-OP算法采用经验重放方法，在每个任务的训练过程中计算混合小批次度量，并根据此度量保存该任务的核心集样本。当训练新任务时，通过PER策略来重放这些核心集样本，以避免智能体在旧任务中性能的不均衡。从3.3.3节和4.3.1节的实验看出，相较于对比算法，两者新旧任务上均有出色的性能，都很好的解决了多智能体持续强化学习中的灾难性遗忘问题。为了更清晰地对比这两种算法在空间和时间上的表现差异，表4.5展示了在MPE平台下这两种算法的主要指标：

表4.5 OwDP算法和MADDPG-OP算法主要指标对比

|  |  |  |
| --- | --- | --- |
| 主要指标 | OwDP算法 | MADDPG-OP算法 |
| 空间占用 | 2.69MB | 12.48MB |
| 训练时长（5000轮） | 2650s | 3220s |

通过观察可以发现，MADDPG-OP算法在空间占用和训练时长方面均大于OwDP算法，这是因为计算混合小批次度量和样本重放重要性都需要花费大量时间，且存储样本所需的空间也很大。因此，从这些方面来看，OwDP算法更具优势。

图4.11展示了按照顺序：myTag->myPush->myAdversary，即：任务1->任务2->任务3，对两种算法的智能体在持续学习训练过程中所得的奖励值进行比较。

|  |
| --- |
| 1. 任务1 |
| 1. 任务2 |
| 1. 任务3 |

图4.11 持续强化学习算法训练阶段的平均奖励变化曲线

从图4.11(a)看出，任务1作为持续学习的第一个任务，由于不需要添加正则化约束或者重放先前任务的样本，因此两个算法的奖励曲线与传统强化学习无异。图4.11(b)为智能体学习完任务1后继续学习任务2的结果，可以看出，在训练初期，MADDPG-OP算法奖励增长较OwDP算法慢，但在训练3000轮左右时两者奖励基本持平，并达到收敛。图4.11(c)为智能体学习完任务1和任务2后继续学习任务3的结果，可以看出，两个算法的奖励交替出现较大值，并在训练4500轮左右时达到收敛，且最终奖励值基本相等。

从实验结果可见，在学习新任务时，OwDP算法和MADDPG-OP算法从环境获得的奖励相当，这表明两者均能在新任务中表现出良好的性能。

为了进一步对比不同算法的性能，图4.12分别展示了在持续学习过程中，智能体执行三个任务获得的平均奖励值。图4.12中横坐标轴表示不同任务训练阶段，如：横坐标取值1代表训练完任务1；纵坐标轴为智能体执行某一任务的平均奖励值。

从图4.12看出，对于这三个任务而言，MADDPG-OP算法的平均奖励始终高于OwDP算法，这表明在学习完新任务后，MADDPG-OP算法在旧任务中表现更优，对旧任务知识的保留也更加出色。

以上分析显示，OwDP算法在多智能体的训练过程中相较于MADDPG-OP算法所需的时间和空间更少，但在学习新任务后在旧任务的表现稍差。相反，MADDPG-OP算法为了保留旧任务的核心集，需要占用比EWC约束系数和联合PCA约束系数更大的存储空间；同时，在学习新任务时，由于需要重放旧任务样本，也会消耗更多的时间。尽管如此，MADDPG-OP算法在旧任务的表现更好。

因此，在实际应用中，选择使用哪种算法需要综合考虑具体的应用场景。如果时间空间资源有限，推荐采用OwDP算法；而若需要更好的持续学习性能，则建议优先考虑使用MADDPG-OP算法。

|  |
| --- |
| 1. 任务1 |
| 1. 任务2 |
| 1. 任务3 |

图4.12 持续强化学习算法智能体执行任务的平均奖励比较

## 本章小结

针对多智能体持续强化学习的问题背景，本章引入持续学习中的经验重放方法，基于MADDPG框架提出了MADDPG-OP算法。该算法对OCS算法进行了改进，采用混合小批次度量，结合样本对个体和团体训练的贡献度来选取旧任务样本。在智能体学习新任务时，算法采用PER策略选择并重放这些旧任务的样本，使智能体在学习新任务的同时也能兼顾旧任务，从而实现持续学习。

之后，在myTag、myPush和myAdversary三个场景中进行了对比实验。实验结果表明，当多智能体按顺序学习三个场景的任务时，MADDPG-OP算法相较于其他对比算法具有明显优势，在一定程度上能够使所有智能体都得到充分训练，从而避免了部分智能体对旧任务性能较差的问题。

接着，在上述三个场景进行了消融实验。结果证明，MADDPG-OP算法的两种机制都不同程度地增强了算法地性能，能够最大限度保留旧任务的知识，兼顾了模型的稳定性和可塑性。

最后，对本文提出的两种多智能体持续强化学习算法进行了比较，分析了它们持续学习的成本和性能，并判断了各自的适用场景。

# 第五章 总结与展望

## 总结

以强化学习为代表的人工智能技术正在逐步改变人们的生活，人们对人工智能的期待也越来越高。然而，对于复杂的多智能体系统而言，智能体只能完成单一的任务，当面临多任务挑战时，智能体可能会出现灾难性遗忘问题。本文深入分析了多智能体系统中各智能体之间的关系，并结合已有的持续学习算法，主要完成了以下工作：

（1）本文对多智能体强化学习的经典算法及其相关应用进行了深入调研，研究了持续学习的前沿算法，并对其进行了分类，为后续多智能体持续强化学习算法的提出提供了充足的准备。

（2）针对多智能体在顺序学习多任务时面临的灾难性遗忘问题，本文将持续学习中的正则化方法引入多智能体强化学习中，基于MADDPG框架提出了OwDP算法，通过知识蒸馏在策略级别限制对旧任务重要参数的改变，同时引入联合PCA约束，以缓解多智能体在旧任务上决策协同性的降低。实验结果表明，该算法能够有效帮助智能体保留旧任务的知识，从而实现多智能体持续强化学习。

（3）针对多智能体在顺序学习多任务时面临的灾难性遗忘问题，本文将持续学习中的经验重放方法引入多智能体强化学习中，基于MADDPG框架提出了MADDPG-OP算法，采用混合小批次度量，结合样本对个体和团体的代表程度来选取旧任务的样本。在新任务的训练阶段，通过PER策略选择并重放这些样本，以减轻多智能体对旧任务的遗忘。实验结果表明，该算法能够确保所有智能体得到充分的训练，从而实现多智能体持续强化学习。

（4）对本文提出的两种多智能体持续强化学习算法进行了系统的比较，分析了它们持续学习任务的成本和性能，并判断了各自的适用场景。

## 展望

本文对多智能体持续强化学习算法进行了深入研究，并取得了一些预期成果。然而，受限于时间和实验条件，本文的研究还存在一些不足之处，未来的工作将集中在以下方面：

（1）本文侧重于提高多智能体持续强化学习算法的性能，避免灾难性遗忘。然而，本文并未对算法的收敛速度进行深入研究，即如何在更短的时间内达到收敛状态。为了改进这一点，在未来的工作中，可以引入更多的探索机制，以引导智能体更加高效地探索状态空间和动作空间，从而更快达到收敛状态。

（2）本文为了验证算法有效性进行了大量的实验，但这些实验都局限于MPE平台，尚未验证算法在现实大规模复杂场景下的性能。因此，下一步的研究可以将算法扩展到多无人机协同等更广泛领域，增加状态空间、动作空间的维度以及智能体的数量，增加环境复杂度，从而解决现实场景中的问题。

（3）第四章提出了一种基于经验重放的多智能体持续强化学习算法，该算法重放先前任务的样本，有助于减轻多智能体对旧任务的遗忘。然而，这种方法需要存储较多的样本，因此占用了较多的空间，这对于某些存储容量较小的设备来说可能是不利的。因此，如何降低样本占用空间，将是后续工作中值得深入研究的内容。例如，可以通过对所有历史任务的样本进行聚类的方式，将相似样本归为一类，并只保留每个类别中的少量样本。这种方法可以有效地减少存储和计算的开销，同时保留了历史任务的代表性样本。

参考文献

1. Pallathadka H, Ramirez-Asis E H, Loli-Poma T P, et al. Applications of artificial intelligence in business management, e-commerce and finance[J]. Materials Today: Proceedings, 2023, 80(3): 2610-2613.
2. Sunarya P A. Machine learning and artificial intelligence as educational games[J]. International Transactions on Artificial Intelligence, 2022, 1(1): 129-138.
3. Kosch T, Welsch R, Chuang L, et al. The Placebo Effect of Artificial Intelligence in Human–Computer Interaction[J]. ACM Transactions on Computer-Human Interaction, 2023, 29(6): 1-32.
4. Zuo Y, Guo J, Gao N, et al. A survey of blockchain and artificial intelligence for 6G wireless communications[J]. IEEE Communications Surveys & Tutorials, 2023, 25(4): 2494-2528.
5. Hannon B, Kumar Y, Gayle D, et al. Robust Testing of AI Language Model Resiliency with Novel Adversarial Prompts[J]. Electronics, 2024, 13(5): 842.
6. Touvron H, Bojanowski P, Caron M, et al. Resmlp: Feedforward networks for image classification with data-efficient training[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022, 45(4): 5314-5321.
7. Otter D W, Medina J R, Kalita J K. A survey of the usages of deep learning for natural language processing[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020, 32(2): 604-624.
8. Haeb-Umbach R, Heymann J, Drude L, et al. Far-field automatic speech recognition[J]. Proceedings of the IEEE, 2020, 109(2): 124-148.
9. 李凯文, 张涛, 王锐等. 基于深度强化学习的组合优化研究进展[J]. 自动化学报, 2021, 47(11): 2521-2537.
10. Zheng Y, Xie X, Su T, et al. Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning[C]//IEEE/ACM International Conference on Automated Software Engineering. New York, NY: IEEE, 2019: 772-784.
11. Glaese A, McAleese N, Trębacz M, et al. Improving alignment of dialogue agents via targeted human judgements[R/OL]. https://arxiv.org/abs/2209.14375[2023].
12. 王雪松, 王荣荣, 程玉虎. 安全强化学习综述[J]. 自动化学报, 2023 , 49(09): 1813-1835.
13. 林谦, 余超, 伍夏威等. 面向机器人系统的虚实迁移强化学习综述[J]. 软件学报, 2024, 35(02): 711-738.
14. Liu B, Pu Z, Zhang T, et al. Learning to Play Football from Sports Domain Perspective: A Knowledge-embedded Deep Reinforcement Learning Framework[J]. IEEE Transactions on Games, 2022, 15(14): 648-657.
15. Kaufmann E, Bauersfeld L, Loquercio A, et al. Champion-level drone racing using deep reinforcement learning[J]. Nature, 2023, 620(7976): 982-987.
16. Hu Y, Fu J, Wen G. Safe Reinforcement Learning for Model-Reference Trajectory Tracking of Uncertain Autonomous Vehicles With Model-Based Acceleration[J]. IEEE Transactions on Intelligent Vehicles, 2023, 8(3): 2332-2344.
17. Ning Z, Yang Y, Wang X, et al. Multi-agent deep reinforcement learning based uav trajectory optimization for differentiated services[J]. IEEE Transactions on Mobile Computing, 2023, 23(2) : 1503-1520.
18. Li K, Wang X, He Q, et al. Task computation offloading for multi-access edge computing via attention communication deep reinforcement learning[J]. IEEE Transactions on Services Computing, 2023, 16(4): 2985-2999.
19. Oroojlooy A, Hajinezhad D. A review of cooperative multi-agent deep reinforcement learning[J]. Applied Intelligence, 2023, 53(11): 13677-13722.
20. Wong A, Bäck T, Kononova A V, et al. Deep multiagent reinforcement learning: Challenges and directions[J]. Artificial Intelligence Review, 2023, 56(6): 5023-5056.
21. Zaman M A, Miehling E, Başar T. Reinforcement learning for non-stationary discrete-time linear–quadratic mean-field games in multiple populations[J]. Dynamic Games and Applications, 2023, 13(1): 118-164.
22. Wang C Y, Bochkovskiy A, Liao H Y M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2023: 7464-7475.
23. French R M. Catastrophic forgetting in connectionist networks[J]. Trends in Cognitive Sciences, 1999, 3(4): 128-135.
24. 孙长银, 穆朝絮. 多智能体深度强化学习的若干关键科学问题[J]. 自动化学报, 2020, 46(7): 1301-1312.
25. 蔡华俭, 黄梓航, 林莉等. 半个多世纪来中国人的心理与行为变化——心理学视野下的研究[J]. 心理科学进展, 2020, 28(10): 1599-1618..
26. Minsky M L. Theory of neural-analog reinforcement systems and its application to the brain-model problem[M]. Princeton, NJ: Princeton University, 1954.
27. Dangl T, Wirl F. Investment under uncertainty: calculating the value function when the Bellman equation cannot be solved analytically[J]. Journal of Economic Dynamics and Control, 2004, 28(7): 1437-1460.
28. Ribeiro J G, Rodrigues G, Sardinha A, et al. TEAMSTER: Model-based reinforcement learning for ad hoc teamwork[J]. Artificial Intelligence, 2023, 324(1): 4013-4022.
29. Szepesvári C, Littman M L. Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms[C]//Proceedings of International Conference on Machine Learning. New York, NY: ACM, 1996.
30. Sutton R S. Learning to predict by the methods of temporal differences[J]. Machine Learning, 1988, 3: 9-44.
31. Watkins C J C H, Dayan P. Q-learning[J]. Machine Learning, 1992, 8: 279-292.
32. Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[R/OL]. https://arxiv.org/abs/1312.5602[2023].
33. Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Menlo Park, CA: AAAI, 2016.
34. Wang Z, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning[C]//Proceedings of International Conference on Machine Learning. New York, NY: ACM, 2016: 1995-2003.
35. Sutton R S, McAllester D, Singh S, et al. Policy gradient methods for reinforcement learning with function approximation[C]//Advances in Neural Information Processing Systems. New York, NY: Curran Associates, 1999.
36. Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine Learning, 1992, 8: 229-256.
37. Konda V, Tsitsiklis J. Actor-critic algorithms[C]//Advances in Neural Information Processing Systems. New York, NY: Curran Associates, 1999.
38. Yang Z, Merrick K, Jin L, et al. Hierarchical deep reinforcement learning for continuous action control[J]. IEEE Transactions on Neural Networks and Learning Systems, 2018, 29(11): 5174-5184.
39. Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[R/OL]. https://arxiv.org/abs/1707.06347[2023].
40. Sukhbaatar S, Fergus R. Learning multiagent communication with backpropagation[C]//Advances in Neural Information Processing Systems. New York, NY: Curran Associates, 2016: 2252-2260.
41. de Witt C S, Gupta T, Makoviichuk D, et al. Is independent learning all you need in the starcraft multi-agent challenge?[R/OL]. https://arxiv.org/abs/2011.09533[2023].
42. Sunehag P, Lever G, Gruslys A, et al. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward[C]//Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. Berlin, German: Springer, 2018: 2085-2097.
43. Rashid T, Samvelyan M, De Witt C S, et al. Monotonic value function factorisation for deep multi-agent reinforcement learning[J]. The Journal of Machine Learning Research, 2020, 21(1): 7234-7284.
44. Lowe R, Wu Y I, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments[C]//Advances in Neural Information Processing Systems. New York, NY: Curran Associates, 2017: 6382-6393.
45. Zhang Y, Yang Q. A survey on multi-task learning[J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 34(12): 5586-5609.
46. 王松, 买日旦·吾守尔, 古兰拜尔·吐尔洪等. 基于知识架构的持续学习情感分类方法[J]. 计算机工程, 2023, 49(02): 112-118.
47. De Lange M, Aljundi R, Masana M, et al. A continual learning survey: Defying forgetting in classification tasks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021, 44(7): 3366-3385.
48. Rebuffi S A, Kolesnikov A, Sperl G, et al. icarl: Incremental classifier and representation learning[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2017: 2001-2010.
49. Lopez-Paz D, Ranzato M A. Gradient episodic memory for continual learning[C]//Advances in Neural Information Processing Systems. New York, NY: Curran Associates, 2017: 6470-6479.
50. Yoon J, Madaan D, Yang E, et al. Online Coreset Selection for Rehearsal-based Continual Learning[C]//International Conference on Learning Representations. Ithaca, NY: ICLR, 2022.
51. Zhao B, Xiao X, Gan G, et al. Maintaining discrimination and fairness in class incremental learning[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2020: 13208-13217.
52. Li Z, Hoiem D. Learning without forgetting[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 40(12): 2935-2947.
53. Li S, Lin M, Wang Y, et al. Distilling a powerful student model via online knowledge distillation[J]. IEEE Transactions on Neural Networks and Learning Systems, 2022, 34(11): 8743-8752.
54. Rannen A, Aljundi R, Blaschko M B, et al. Encoder based lifelong learning[C]//Proceedings of the IEEE International Conference on Computer Vision. Piscataway, NJ: IEEE, 2017: 1320-1328.
55. Kirkpatrick J, Pascanu R, Rabinowitz N, et al. Overcoming catastrophic forgetting in neural networks[J]. Proceedings of the National Academy of Sciences, 2017, 114(13): 3521-3526.
56. Mallya A, Lazebnik S. Packnet: Adding multiple tasks to a single network by iterative pruning[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2018: 7765-7773.
57. Kang H, Mina R J L, Madjid S R H, et al. Forget-free continual learning with winning subnetworks[C]// Proceedings of International Conference on Machine Learning. New York, NY: ACM, 2022: 10734-10750.
58. Atkinson C, McCane B, Szymanski L, et al. Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting[J]. Neurocomputing, 2021, 428(7): 291-307.
59. Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in Neural Information Processing Systems. New York, NY: Curran Associates, 2014: 2672-2680.
60. Kessler S, Parker-Holder J, Ball P, et al. Same state, different task: Continual reinforcement learning without interference[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Menlo Park, CA: AAAI, 2022: 7143-7151.
61. Nath S, Peridis C, Ben-Iwhiwhu E, et al. Sharing Lifelong Reinforcement Learning Knowledge via Modulating Masks[R/OL]. https://arxiv.org/abs/2305.10997[2023].
62. Brockman G, Cheung V. Openai gym[R/OL]. https://arxiv.org/abs/1606.01540[2023].
63. Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms[C]// Proceedings of International Conference on Machine Learning. New York, ACM: PMLR, 2014: 387-395.
64. Schwarz J, Czarnecki W, Luketina J, et al. Progress & compress: A scalable framework for continual learning[C]// Proceedings of International Conference on Machine Learning. New York, NY: ACM, 2018: 4528-4537.
65. Zhao B, Dong X, Guo Y, et al. PCA dimensionality reduction method for image classification[J]. Neural Processing Letters, 2022, 54(1): 347–368.

作者简介

##### 基本情况

##### 教育背景

##### 攻读硕士学位期间的研究成果

###### 参与科研项目及获奖

1. 装备预研基金项目, 基于多源探测数据的\*\*\*, 2020.01-2022.03, 验收效果良好, 主要项目负责人。
2. 西安电子科技大学学业奖学金，一等奖，2021年10月。