

**本科生毕业论文（设计）**



**题 目 多智能体均值-方差线性组合最小强化学习算法预研**

**学 院 计算机学院**

**专 业 计算机科学与技术**

**学生姓名 王钰**

**学 号 2020141420210 年级 2021**

**指导教师 郭宏亮**

**教务处制表**

**2025 年 4 月 21 日**

多智能体均值-方差线性组合最小强化学习算法预研

专业 计算机科学与技术

学生姓名 王钰 指导教师 郭宏亮

**[摘要]** 当下，随着人工智能的飞速发展，多智能体强化学习（Multi-Agent Reinforcement Learning, MARL）逐步成为研究者们关注的热点。现实的应用场景中通常同时存在多个决策个体，因此，MARL方法对于构建在现实场景中更可靠的智能体有着极其重要的意义。当前的MARL方法仍然面临着多智能体环境随机性过大，从而智能体的学习不稳定的挑战。多智能体环境中的随机性主要来源于两个方面：第一，由于环境的随机性，智能体们在特定状态做出特定动作的情况下，从环境中获得的奖励可能是随机的；第二，由于环境中存在多个智能体，单个智能体获得的奖励因为其它智能体的存在而具有随机性。然而，现有的MARL方法主要基于时序差分学习或策略梯度学习的方法对智能体获得的回报的均值进行建模，也就无法描述多智能体环境中智能体获得的奖励的随机性，无法让智能体更鲁棒地学习到最优决策。为了解决上述挑战，本文提出一种全新的强化学习的建模方法—多智能体级联时序差分（Multi-Agent Cascaded Temporal Difference, MA-CTD），对智能体获得的回报的均值和方差同时进行建模，让智能体根据回报的期望和随机性的大小做出更智能的决策。实验表明，本文提出的MA-CTD方法可以让智能体更鲁棒地学习到最优决策。

**[关键词]** 多智能体强化学习；环境随机性；均值-方差建模；多智能体级联时序差分

**Multi-Agent Mean-Variance Linear Combination Minimization Reinforcement Learning Algorithm**

Computer Science

Student: Yu Wang Adviser: Hongliang Guo

**[Abstract]** With the rapid advancement of artificial intelligence, Multi-Agent Reinforcement Learning (MARL) has gradually become a focal point of research. In real-world application scenarios, multiple decision-making agents often coexist simultaneously. Therefore, MARL methods hold significant importance in constructing more reliable agents for practical scenarios. However, current MARL methods still face the challenge of excessive randomness in multi-agent environments, which leads to insufficient robustness in agents' learning processes. The randomness in multi-agent environments primarily stems from two aspects. First, due to the stochastic nature of the environment, the rewards agents receive from the environment for taking specific actions in specific states may be random. Second, the presence of multiple agents in the environment introduces additional randomness, as the rewards received by a single agent are influenced by the actions of other agents. Existing MARL methods mainly model the mean of the rewards obtained by agents using Temporal Difference learning or Policy Gradient learning. Therefore, existing MARL methods fail to capture the randomness of rewards in multi-agent environments, thereby preventing agents from learning optimal decisions in a more robust manner. To address the aforementioned challenges, this paper proposes a novel reinforcement learning modeling method—Multi-Agent Cascaded Temporal Difference (MA-CTD). This method models both the mean and variance of the rewards obtained by agents, enabling them to make smarter decisions based on the expected rewards and the magnitude of randomness. Experiments demonstrate that the proposed MA-CTD method allows agents to learn optimal decisions more robustly.

**[Key Words]** MARL, Environmental Stochasticity, Mean-Variance Modeling, MA-CTD

目录

[多智能体均值-方差线性组合最小强化学习算法预研 1](#_Toc196231369)

[1 背景介绍 5](#_Toc196231370)

[2 相关研究综述 6](#_Toc196231371)

[2.1 单智能体强化学习算法研究综述 6](#_Toc196231372)

[2.2 多智能体强化学习算法研究综述 8](#_Toc196231373)

[2.2.1 完全合作式多智能体强化学习 8](#_Toc196231374)

[2.2.2 完全竞争式多智能体强化学习 9](#_Toc196231375)

[2.2.3 合作-竞争混合式多智能体强化学习 9](#_Toc196231376)

[3 强化学习基本公式 9](#_Toc196231377)

[3.1 强化学习基本公式 10](#_Toc196231378)

[3.2 时序差分学习 11](#_Toc196231379)

[3.3 多智能体强化学习基本公式 13](#_Toc196231380)

[3.3.1 多智能体强化学习基本概念 13](#_Toc196231381)

[3.3.2 多智能体强化学习经典算法简介：价值分解网络 13](#_Toc196231382)

[4 MA-CTD算法 14](#_Toc196231383)

[4.1 单个智能体的MA-CTD算法及策略选择 15](#_Toc196231384)

[4.1.1 MA-CTD算法理论推导 15](#_Toc196231385)

[4.1.2 MA-CTD算法下智能体的策略选择 18](#_Toc196231386)

[4.2 深度强化学习下的MA-CTD算法 18](#_Toc196231387)

[4.3 多智能体强化学习下MA-CTD的应用 20](#_Toc196231388)

[4.4 MA-CTD算法复杂度分析 22](#_Toc196231389)

[5 实验 23](#_Toc196231390)

[5.1 实验多智能体环境介绍 23](#_Toc196231391)

[5.2 算法实现细节 26](#_Toc196231392)

[5.3 实验结果 26](#_Toc196231393)

[5.4 消融实验 29](#_Toc196231394)

[5.4.1 风险系数消融实验 29](#_Toc196231395)

[5.4.2 学习率、方差学习率消融实验 32](#_Toc196231396)

[5.5 讨论 33](#_Toc196231397)

[6 结论 34](#_Toc196231398)

[7 参考文献 35](#_Toc196231399)

[声 明 38](#_Toc196231400)

[学位论文使用授权书 38](#_Toc196231401)

# 背景介绍

近年来，人工智能相关的技术取得了飞速的发展，并在多个领域展现出很大的应用潜力。在人工智能的发展过程中，多智能体系统作为一种人工智能技术重要的应用范式，逐渐成为了研究和实践的热点[1]。多智能体系统是由多个智能体组成的，这些智能体能够在环境中自主感知、决策和行动，以实现个体或集体的目标。与传统的单智能体系统相比，多智能体系统具有更高的灵活性和适应性，从而能够更好地应对复杂的现实环境。例如，在分布式机器人系统中，多个机器人需要合作以完成复杂的任务[2]；在游戏开发中，多个非玩家角色需要进行交互以给玩家更好的游戏体验[3]。MARL作为多智能体系统的核心技术之一，近年来逐渐成为人工智能领域的研究热点。在多智能体环境中，多个智能体需要同时与环境交互，每个智能体根据自身对环境的观测（可能包括其他智能体的行为）做出决策。这种复杂的交互关系使得MARL问题比单智能体强化学习更具挑战性。

随着现代深度学习技术和算力的发展，学者们对MARL的研究取得了显著进展。当下，将深度神经网络与强化学习相结合的深度强化学习已经成为了一种MARL的范式[4]。这种结合使得每个智能体都可以拥有深度神经网络的特征提取能力，也就使得智能体可以处理高维状态空间和动作空间；通过端到端的学习方式，智能体可以通过神经网络自动提取环境中的关键特征，显著提升了智能体的效率和性能。

然而，现有的MARL方法仍然面临着一个主要挑战：现有MARL方法对贝尔曼方程[5]的依赖限制了其对复杂多智能体环境中奖励分布的全面刻画，导致这些现有的MARL方法难以有效应对多智能体环境中高随机性带来的学习不稳定性。在多智能体环境中，智能体的行为不仅受到自身策略与环境的影响，还受到其他智能体策略的不确定性的影响。也就是说，智能体从环境中获得的奖励的随机性来源于两个方面：一是环境本身的随机性，即智能体在特定状态下采取特定动作时，从环境中获得的奖励可能是随机的；二是其他智能体的行动带来的随机性，即单个智能体获得的奖励可能受到其他智能体行为的影响。这种强随机性的奖励会导致智能体学习过程十分不稳定，甚至导致算法难以收敛。现有MARL方法主要依赖贝尔曼方程对智能体从环境中获得的累计奖励，即回报的均值进行建模，让智能体预测行动可以获得的回报的均值从而选择最优策略。然而，由于多智能体环境中的强随机性，回报的分布可能是高方差的。因此，智能体仅根据回报的均值选择的策略是不鲁棒的，从而导致智能体学习的不稳定性。

针对现有的MARL方法所面临的上述挑战，本文提出了MA-CTD算法。MA-CTD算法的核心在于同时建模回报的均值和方差，即让智能体预测行动可以获得的回报的均值和方差。MA-CTD算法的建模使得智能体能够在考虑回报的期望的同时兼顾回报的不确定性，即回报的方差；从而让智能体做出的决策更加鲁棒，增强其在复杂环境中的学习稳定性。在贝尔曼方程下回报的均值的状态-动作价值函数更新公式的基础上，本文推导出了回报的方差的状态-动作价值函数的表达式以及更新公式，使得智能体能够动态地估计其在每个状态或状态-动作对下的回报的完整分布特征。为了应对复杂多智能体环境中的高维状态空间和动作空间，本文进一步将MA-CTD方法扩展到深度强化学习的框架下，为每个智能体设计了计算回报的均值和方差的两个深度神经网络，并分别推导了这两个深度神经网络的损失函数，使得在深度强化学习的框架下智能体可以完成端到端的学习并保证对回报的分布的建模。同时，MA-CTD可以看作一种扩展性强的即插即用的算法，现有的MARL方法可以通过在其原算法基础上引入MA-CTD的均值-方差建模来进一步提升智能体学习的稳定性。

本文后面的部分将按以下顺序组织：第2章介绍现有的强化学习和多智能体强化学习方法的相关研究；第3章详细对MA-CTD的相关公式进行推导；第4章展示MA-CTD算法的实验环境并分析实验结果；第5章对本文进行总结；第6章为本文的参考文献。

# 相关研究综述

多智能体强化学习算法是在单智能体强化学习算法的基础之上发展而来的，因此，本章节将分为单智能体强化学习算法研究综述和多智能体强化学习算法研究综述，并总结现有方法的优缺点。

## 单智能体强化学习算法研究综述

在单智能体强化学习中，智能体需要学习到最优策略，使得它在与环境交互的过程中可以获得尽可能大的累计奖励；在单智能体强化学习方法中，Q-learning是最流行的强化学习方法之一[6]。Q-learning算法让智能体学习每个状态下每个动作可能获得的回报的均值，从而做出最优决策。通常，Q-learning方法会让智能体采用ε-贪心策略进行动作选择[7]。

Watkins博士在1989年首次提出了Q-learning算法。Watkins博士[8]将智能体与环境交互并学习的过程建模为马尔可夫决策过程（Markov Decision Process, MDP），并结合了时序差分、动态规划等思想，提出了Q-learning的概念。Q-learning通过在程序内维护一张二维的Q表格，来计算智能体在状态s下采取动作a可以获得的回报的期望；每次与环境交互时，程序都会通过状态、动作以及从环境中获得的实际奖励来更新Q表格的一部分。在Q-learning被提出之后，Watkins等人[9]证明了Q-learning算法的收敛性：假如智能体重复探索过了所有状态下所有可能的动作，那么Q-learning中的Q表格会收敛到最优的状态-动作价值函数。

在Q-learning提出后，许多学者针对Q-learning算法做了相关改进。Q-learning算法需要在内存中维护一张状态-动作的二维Q表格，当状态、动作数量过多或者智能体处于连续观测空间时，Q-learning算法就会产生极高的内存开销；所以Martin[10]提出了神经拟合Q值（Neural Fitted Q, NFQ）算法，用神经网络计算在状态s下执行动作a的Q值，代替从Q表格中查询在状态s下执行动作a的Q值，很大程度上节省了内存并提高了数据利用的效率。Mnih等人[11]提出了深度Q网络（Deep Q-Network, DQN），DQN可以直接将每个状态的视觉图像作为输入，并且根据状态s计算所有动作的分数，通过经验回放让智能体根据过往的经验更新网络参数，并将智能体与环境的交互存放到经验池中，以便后续智能体进行学习；DQN在Atari游戏上的表现远优于当时的其它算法。值得说明的是，NFQ算法中神经网络的输入是状态s和动作a，输出是对应的Q值，也就是说对每个状态下的每个动作都要用神经网络计算其Q值；而DQN算法中神经网络的输入是状态s，输出是动作空间维度的向量，表示当前状态下每个动作对应的Q值；因此，DQN中对神经网络的利用显然是更高效的。

在DQN算法提出后，许多学者从不同角度对DQN进行了改进。Van等人[12]发现DQN中最终预测出的Q值过高的现象，于是他们提出了双网络DQN（Double DQN，DDQN），DDQN包含一个用于计算Q值的网络（策略网络）和一个计算目标值的网络（目标网络），每次训练时只更新策略网络的参数，一段时间后，目标网络的参数与策略网络同步；这样就保证了计算的Q值不会越来越高，减小了Q值的过估计。Wang等人[13]提出了对偶网络的DQN（Dueling DQN），将网络输出的Q值分解为动作Q值和状态Q值，这两个Q值加和起来就是智能体在某状态下选择某动作的Q值；这种网络架构除了让智能体学习到好的决策外，还能让智能体学习到环境中什么状态是好的状态。在一些场景中，Q值和当前状态高度相关，在某些状态下无论选择什么样的动作的Q值都相差不大，Dueling DQN的设计就可以让智能体在这些场景下学习到更好的策略。

上述强化学习算法的目标都是找到最优的状态-动作价值函数Q，即让智能体预测在状态s下选择动作a可以获得的回报的均值。然而，在环境中存在的随机性过大的情况下，仅仅计算回报的均值是不够的。因此，Guo等人[14]提出了级联时序差分学习的方法，利用两个级联的Q表格，让智能体接连预测在状态s下选择动作a可以获得的回报的均值和方差，并推导出了Q表格更新的公式；他们的算法在实际的交通路网上高效地解决了可靠最短路径的问题。

## 多智能体强化学习算法研究综述

在多智能体强化学习中，每个智能体都需要学习到自己的最优策略，使得所有智能体一起行动时多智能体系统可以获得尽可能大的累积奖励[15]。随着多智能体强化学习方法广泛应用到人工智能相关领域，用于解决路径规划、博弈、任务分配等问题，越来越多的研究人员将目光聚集在了多智能体强化学习方法上[16]。多智能体强化学习方法可以根据智能体之间的关系大致分为三类：完全合作式、完全竞争式以及合作-竞争混合式[17]。在完全合作式多智能体强化学习中，所有智能体需要相互协作以完成共同的团队任务，它们的目标是最大化团队累积奖励；在完全竞争式多智能体强化学习中，每个智能体只关注自身行为，它们的目标是最大化自身累积奖励；在合作-竞争混合式多智能体强化学习中，每个智能体既要考虑自身利益也要考虑团队利益，它们的优化目标是在最大化自身累积奖励的同时通过与其他智能体的合作最大化团队累计奖励。

对于完全合作式、完全竞争式或是合作-竞争混合式，存在一种通用的多智能体强化学习方法，也就是独立Q学习（Independent Q-Learning, IQL）[18]。IQL算法中，每个智能体都会把其它智能体看作环境的一部分，也就是说，每个智能体都在解决一个单智能体强化学习任务；对于不同的任务，只需要修改每个智能体获得奖励的方式即可。然而，由于在IQL的建模中，环境中存在其它智能体，所以环境是非稳态的，算法的收敛性无法从理论上得到证明；且智能体之间无法进行任何形式的信息共享，然而，这种算法在工程实践中仍有一定价值。接下来，本文将分别对完全合作式、完全竞争式以及合作-竞争混合式这三类多智能体强化学习方法进行介绍。

### 完全合作式多智能体强化学习

在完全合作式多智能体强化学习中，智能体们会因为完成团队目标而统一获得奖励，不会有个体的奖励。这种共享团队奖励的方式会带来问题，即一个智能体由于只能获得自己的局部观测，从而无法确定是因为自己的行为或者其它智能体的行为而获得奖励；除此之外，由于部分智能体可以学习到比较好的策略以完成团队任务，其它智能体就会变得“懒惰”。为了解决上述问题，Sunehag等人[19]提出了价值分解网络（Value Decomposition Networks, VDN），将团队的Q值分解为各个智能体的Q值之和，每个智能体用深度神经网络来拟合自己的Q值；这样，每个智能体就可以分配到合理的奖励值，且一定程度上避免了懒惰智能体的问题。Rashid等人在VDN的基础上提出了QMIX[20]。Rashid等人通过混合网络将团队的Q值分解为各个智能体的Q值的复杂非线性组合，并在训练中加入全局信息，使得每个智能体学习到的策略更精确；且通过保证混合网络参数的非负性，保证团队Q值与每个智能体计算的Q值的关系是单调的，从而保证集中策略和分散策略的一致性。Kyunghwan等人[21]提出了QTRAN，为价值分解提供了更多了理论证明，并且将每个智能体的价值进行进一步映射，保证每个智能体学习到的状态-价值函数可以逼近最优。

### 完全竞争式多智能体强化学习

在完全竞争式多智能体强化学习中，不同的智能体的目标通常是不同的，甚至互相之间进行博弈，也就是每个智能体都需要学习到比其它智能体更好的策略，才可能获得更高的奖励。Micheal[22]针对二人零和随机博弈场景提出了Minimax-Q算法，使用Q-learning中的时序差分来迭代更新minimax算法的求解公式，即让每个智能体最大化在博弈中最差情况下的期望奖励值，最终学习到纳什均衡策略。Hu[23]等人针对多智能体博弈设计了Nash Q-learning，旨在在多智能体相互竞争或合作的环境中找到纳什均衡作为各个智能体的最优策略；其中，每个智能体在给定其它智能体策略的情况下，通过求解纳什均衡，选择一个能最大化自身收益的动作，并更新状态-动作价值函数。

### 合作-竞争混合式多智能体强化学习

在合作-竞争混合式多智能体强化学习中，智能体需要兼顾自身和团队的利益。Ryan等人[24]提出了多智能体深度确定性策略梯度（Multi-Agent Deep Deterministic Policy Gradient, MADDPG）算法。MADDPG算法是一种基于DDPG算法的扩展，用于解决多智能体环境中的协作和竞争问题。MADDPG采用演员-评论家架构，并让每个智能体在训练中都可以了解其他智能体的策略信息，以改善学习效果和稳定性。Mnih等人[25]提出了异步优势演员-评论家（Asynchronous Advantage Actor-Critic, A3C）算法；A3C算法是一种策略梯度算法，它通过并行执行多个智能体来提高训练效率，并允许异步更新全局模型。A3C结合了价值和策略梯度方法，并使用n步回报来更新策略和价值函数，从而在复杂的围棋、星际争霸等任务中取得了显著的效果。

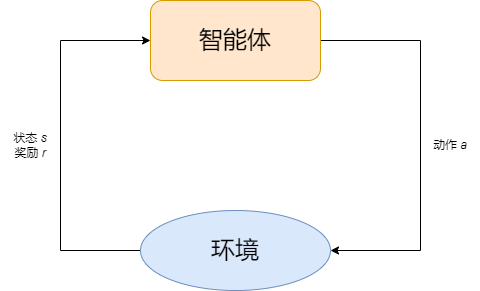
总的来说，现有的单智能体强化学习方法与多智能体强化学习方法均能很好地借助Q表格或神经网络来构建可靠的智能体。然而，现有算法只依赖贝尔曼方程对回报的期望进行建模，不能完整的描述奖励的随机分布情况。

# 强化学习背景知识

本章将主要介绍单智能体、多智能体强化学习中的基础概念。首先，本章节将介绍单智能体强化学习下的基础概念；然后介绍单智能体强化学习的经典方法—时序差分学习。最后，本文将介绍多智能体强化学习的基础概念，并以多智能体强化学习的经典算法VDN为例进行详细介绍。

## 单智能体强化学习基本公式

接下来，本文将简要介绍强化学习中的基础概念。强化学习是机器学习的一个重要分支，强化学习的目标是通过智能体与环境的交互来让智能体学习最优行为策略。智能体通过在环境中采取动作来获得奖励，并通过最大化累积获得的奖励（回报）来优化其行为策略。强化学习中智能体与环境交互的过程通常使用马尔可夫决策过程（Markov Decision Process，MDP）来描述。



图表 1：强化学习中智能体与环境交互的MDP描述

如图表1所示，智能体通过在环境中做出动作与环境进行交互，并且得到环境的状态以及获得的奖励。MDP是强化学习中描述智能体与环境交互的数学模型。MDP假设环境具有马尔可夫性，即未来的状态仅依赖于当前状态和动作，而与过去的历史状态和动作无关；即：

公式中即为状态转换的概率，表示时刻的状态。

因此，一个MDP可以用一个五元组表示。其中（state）是状态空间，（action）是动作空间，是在状态下采取动作可以转换到状态的转换概率，智能体每次在状态下采取动作转换到状态实际获得的奖励记为，是折扣因子。

在强化学习中，智能体需要根据一定的策略来选择动作。因此，定义策略；智能体遵循策略在状态下采取动作的概率为。

接下来，定义智能体从环境中获得的回报为，则回报的计算公式为：

公式中，其中表示时刻，表示回报。也就是说，回报定义为智能体从时刻的状态开始至结束可以获得的累计折扣回报。

在强化学习中，智能体学习的目标是找到最优策略，使得从任意状态开始的期望回报最大化，即：

为了让智能体学习到最优策略，就需要引入价值函数的概念。价值函数是衡量状态或状态-动作对长期价值的函数，因此，让智能体学习到最优策略就等价于学习可以正确描述状态或状态-动作对长期价值的价值函数。价值函数的核心在于评估在特定策略下，从某个状态或状态-动作对开始的期望累积奖励，因此，价值函数可以分为状态价值函数和动作-状态价值函数。状态价值函数可以被定义为：

其中是指遵从策略，智能体从状态开始到结束，可以获得的累计折扣奖励。

同理，可以定义状态-动作价值函数为：

其中是指遵从策略，智能体从状态选择动作到结束，可以获得的累计折扣奖励。

## 单智能体时序差分学习

对一般的强化学习环境，从状态下采取动作的转换概率可以看作1，也就是说，从某种状态下采取某种动作，获得的结果一定是相同的。上一章节定义的两个价值函数均满族贝尔曼方程。根据贝尔曼方程，状态价值函数可以推导为：

在实际的强化学习环境中，可以用智能体在时刻实际获得的奖励来作为的估计值；由状态价值函数的定义可知，。因此，状态价值函数最终可以表达为：

同理，动作-状态价值函数可以推导为：

用智能体在时刻实际获得的奖励来作为的估计值；再由状态-动作价值函数的定义可得，。因此，状态-动作价值函数最终可以表达为：

对于上面的公式，被称为时序差分目标，它可以看作是时刻获得的奖励和时刻状态-动作价值函数值的无偏估计。因此，可定义时序差分误差为：

时序差分误差刻画了实际获得的回报与预期获得的回报的差异。为了让智能体学习到实际可以获得的回报，就需要让智能体的状态-动作价值函数进行相应更新：

在公式中，智能体的状态-动作价值函数值将会根据实际获得的回报更新，是学习率，控制状态-动作价值函数更新的速度。这样，智能体就可以通过与环境的交互不断更新其状态-动作价值函数，也就可以通过状态-动作价值函数计算每个状态-动作下可能获得的回报，从而做出最优决策。

## 多智能体强化学习背景知识

### 多智能体强化学习基本概念

在MARL中，多个智能体需要在共享环境中同时做出决策，以实现个体或集体的目标。与单智能体强化学习不同，多智能体环境中的智能体需要考虑其他智能体的行为和策略，且很有可能无法观测到全部的环境，这使得问题更加复杂。为了更好地描述多智能体环境中的交互过程，分布式部分可观测马尔可夫决策过程（Decentralized Partially Observable Markov Decision Process, Dec-POMDP）[26]被广泛用于建模MARL问题。

在Dec-POMDP中，每个智能体只能观测到部分环境状态，并且需要根据自身的观测做出决策。Dec-POMDP的定义可以用一个七元组来表示。状态空间表示环境的所有可能状态的集合。每个状态描述了环境在某一时刻的完整信息。动作空间表示所有智能体可以采取的动作的集合。每个动作描述了智能体在某一时刻可以执行的操作。观测空间表示每个智能体可以观测到的观测值的集合。每个观测描述了智能体在某一时刻从环境中获得的部分信息。状态转移概率表示在状态下，所有智能体采取动作后转移到状态的概率。观测概率表示在状态下，智能体采取动作后观测到的概率。奖励函数表示在状态下，所有智能体采取动作后获得的即时奖励。是回报的折扣因子。

### 多智能体强化学习经典算法简介：价值分解网络

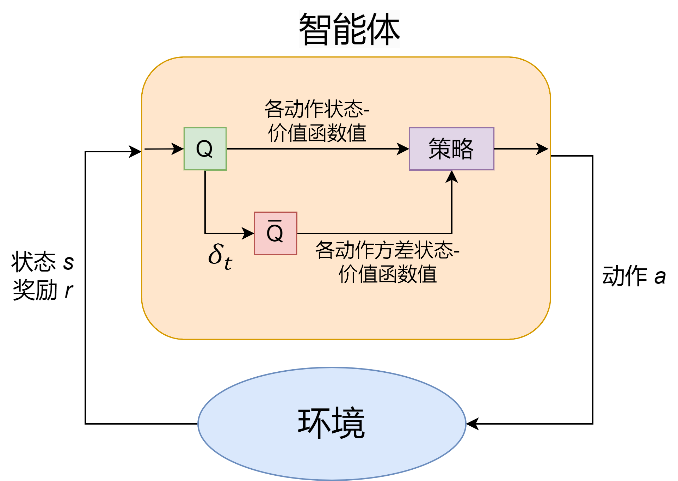
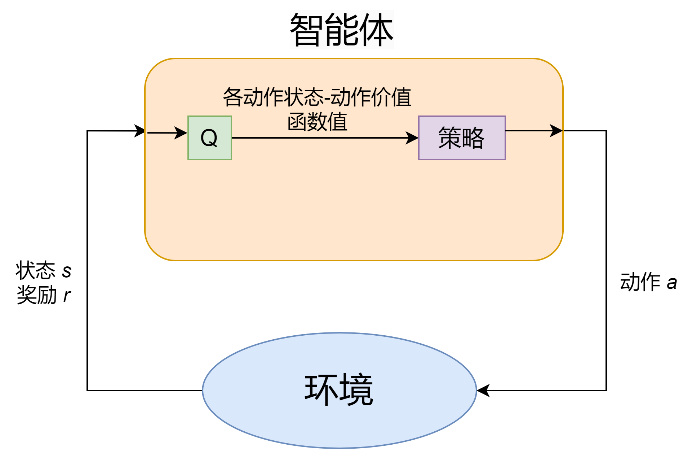
在MARL方法中，VDN[19]是一种经典的完全合作式MARL方法。VDN算法的核心思想是将全局的联合价值函数分解为每个智能体的局部价值函数之和。假设存在个智能体，每个智能体有一个局部观察和动作，其局部价值函数为 ，那么全局的联合价值函数可以表示为：

其中是全局状态，是所有智能体的联合动作。这种将团队奖励分解为各个智能体的奖励之和的分解方式使得复杂的全局价值函数能够通过各个智能体的局部价值函数之和进行近似，大大降低了问题的求解难度。在训练阶段，VDN 算法采用集中训练的方式：所有智能体的局部价值函数通过全局价值函数进行联合优化，从而确保智能体能够学习到全局最优策略。接下来，VDN算法通过收集智能体与环境交互的经验数据，利用经验回放机制对神经网络进行训练，使得全局价值函数能够准确地评估各个智能体的联合动作的价值。在执行阶段，VDN 算法采用分散执行的方式。每个智能体根据自己的局部价值函数独立选择动作，而无需考虑其他智能体的动作。这种分散执行的模式使得智能体能够在实际应用中快速做出决策，使得算法有很大的应用价值。

# MA-CTD算法

MA-CTD算法旨在对多智能体环境下智能体获得的回报的均值和方差同时建模。由于多智能体环境是由多个单智能体构成的，所以本文首先推导针对单个智能体的MA-CTD，并说明这种建模下智能体的策略选择。接下来，为了让智能体可以处理更复杂的环境，本文推导出使用深度神经网络时的单个智能体回报的均值-方差建模。在本章中，本文将均值-方差建模扩展到多智能体的情况下，并以VDN为例说明MA-CTD算法是如何扩展现有MARL方法的。最后，本文将MA-CTD-VDN算法的复杂度与VDN算法的复杂度进行对比分析。

为了便于理解，本文首先直观地对MA-CTD算法进行概述。在一般的时序差分学习算法中，智能体首先与环境交互，获取环境的状态和奖励。根据环境的状态，通过状态-动作价值函数计算出各个状态-动作对的回报的均值，并根据回报的均值通过动作选择策略来选择最优的动作，最后利用环境的奖励对状态-动作价值函数进行更新。这一过程如图2的左图所示。在MA-CTD算法中，智能体仍然首先与环境交互，获取环境的状态和奖励。根据环境的状态，智能体通过状态-动作价值函数计算出各个状态-动作对的回报的均值，进一步通过方差状态-动作价值函数计算出各个状态-动作对的回报的方差，并根据回报的均值和方差通过动作选择策略来选择最优的动作，最后利用环境的奖励对状态-动作价值函数和方差状态-动作价值函数进行更新。这一过程如图2的右图所示。



图表 2：时序差分算法与MA-CTD算法的对比。

也就是说，在相同的环境下，MA-CTD算法通过构建方差状态-动作价值函数的公式来让智能体在预测回报的均值之外还能预测回报的方差，从而让智能体做出的决策更少受环境随机性的干扰，从而使智能体的学习更稳定、决策更鲁棒。

接下来，本文将从方差状态-动作价值函数的定义开始，推导出方差状态-动作价值函数在训练中的更新公式，从而使智能体可以在训练过程中通过学习来方差状态-动作价值函数。

## 单智能体的MA-CTD算法及策略选择

### MA-CTD算法理论推导

在多智能体环境中，每个智能体都会有一个状态-动作价值函数来预测在状态下采取动作可以获得的回报的均值。然而，多智能体环境中的随机性很大，仅仅预测回报的均值不足以刻画智能体可能获得的奖励的分布。因此，本文提出MA-CTD算法，为每个智能体增加方差状态-动作价值函数来预测在状态下采取动作可以获得的回报的方差，从而完整刻画智能体可能获得的奖励的分布。

状态-动作价值函数预测的是回报的均值，因此其定义为。同理，方差状态-动作价值函数预测的是回报的方差，因此其可以定义为：

根据贝尔曼方程，。于是可以推导为：

由定义得，。于是可以推导为：

在等式右边加上并减去一个，得到：

可以发现，等式右边的期望中可以拼凑出在状态-价值函数的贝尔曼方程中的时序差分误差项：

然后，根据期望的平方的性质对上式进行拆解：

将上式中的写作，可得：

对上式中的期望的平方进一步拆解，得到：

根据定义，，故的值为0。因此，上式可进一步化简为：

根据定义，，因此，上式进一步变化为：

根据定义，，因此，上式进一步化简为：

在实际应用中，折扣因子通常是一个很接近1的数。在本文中，设置，就可以认为是一个接近0的数。为了进一步简化公式，舍弃掉上式中含有的项，可以得到最终的方差状态-动作价值函数的贝尔曼方程为：

与状态-动作价值函数在时序差分学习中的应用类似，可以定义方差状态-动作价值函数的时序差分误差项（以下简称为方差时序差分误差项）为：

方差时序差分误差项刻画了智能体实际获得回报的方差与计算出的回报的方差之间的差异。为了让智能体学习到实际获得的回报的方差，就需要让智能体的方差状态-动作价值函数进行相应更新：

在上面的公式中，智能体的方差状态-动作价值函数值将会根据实际获得的回报的方差更新，是方差学习率，控制方差状态-动作价值函数更新的速度。这样，方差状态-动作价值函数就可以和状态-动作价值函数一同进行更新。值得注意的是，方差状态-动作价值函数的更新公式中存在，也就是说方差状态-动作价值函数的更新依赖于状态-动作价值函数的值。因此，在智能体的训练中，每次方差状态-动作价值函数的更新要在状态-动作价值函数更新之后进行，类似级联的关系，故本文将这种对均值-方差进行建模的方式命名为多智能体级联时序差分。

### MA-CTD算法下智能体的策略选择

在MA-CTD算法下，智能体可以同时预测回报的均值和方差，而不仅仅是预测回报的均值，因此，MA-CTD下智能体的策略选择是考虑更周全的。在状态下，记最优策略选择的动作为，一般的强化学习算法下有：

使回报的均值最大的动作即为最优策略下的动作。在MA-CTD算法下，智能体要考虑的则是回报的均值和方差的线性组合，即：

智能体更应该选择更确定能获得更大回报的动作，所以某个动作的方差越小越好；因此，智能体的动作选择策略是回报减去风险系数与方差的乘积。回报的均值和方差的线性组合最大的动作即为最优策略下的动作。本文将公式中的称为风险系数；显然，越大，回报的方差在策略选择时所占的比重就会越大，智能体就会更倾向于选择随机性更小、带来的回报更确定的动作。有关风险系数值的讨论将在实验部分通过消融实验说明。

在训练过程中，本文采取常用的-贪心策略：设定一个值（），智能体有的概率随机选择一个动作，的概率选择使回报的均值和方差的线性组合最大的动作；这样就可以保证智能体在利用学习到的策略的同时仍有可能探索新的策略。在训练过程中，会随训练轮数的增加逐渐减小；这样可以保证智能体在学习后期充分利用学习到的策略，从而使算法更好的收敛。

## 深度强化学习下的MA-CTD算法

在现代的多智能体环境中，状态空间和动作空间通常是高维的。比如，当智能体从环境中获取的状态时图像、网格等数据时，状态空间就是高维的。传统的时序差分学习方法需要维护一个状态-动作价值表格，这在高维空间中会导致内存开销过大；因此，传统的时序差分学习方法在处理高维状态空间和动作空间时面临巨大的挑战。为了解决这一挑战，深度强化学习方法通过深度神经网络来近似Q函数或策略函数，从而避免了维护一个巨大的Q表格，能够有效地处理高维输入。

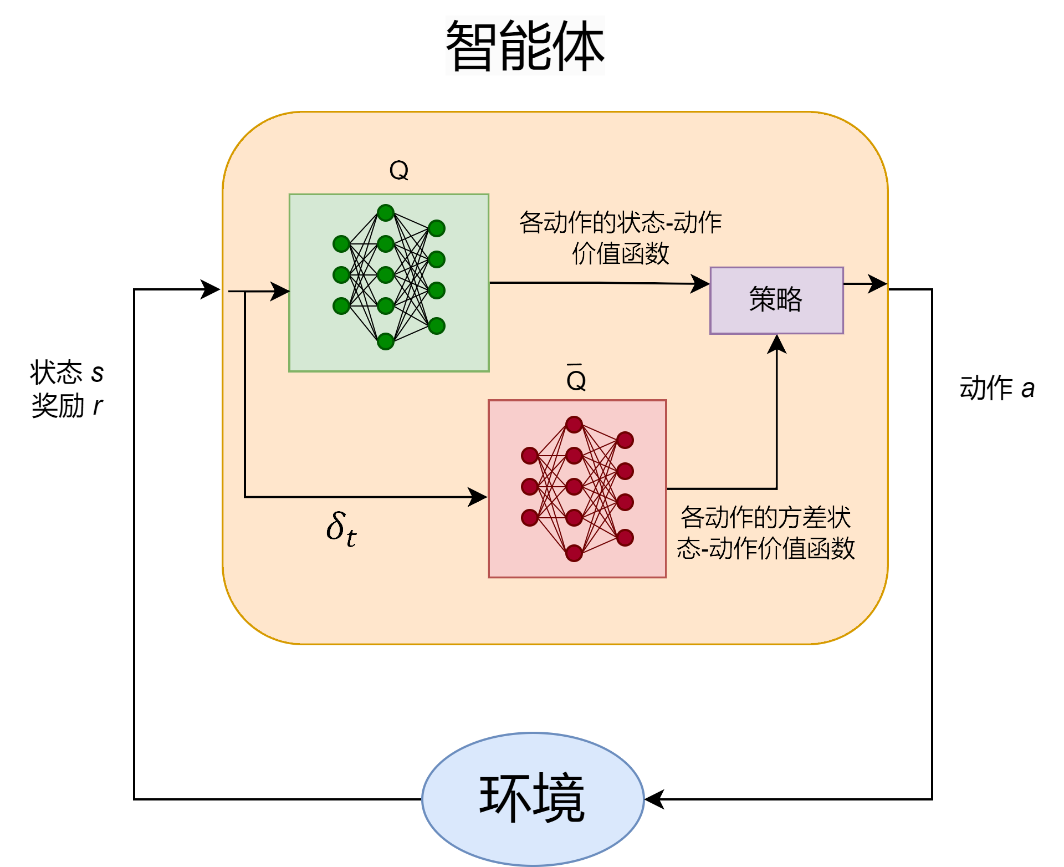
DQN[11]是深度强化学习的一个经典算法，它通过将深度神经网络与Q-learning算法结合，让智能体在高维状态空间下进行学习成为可能。DQN算法直接将智能体观测到的状态作为智能体内部的神经网络（以下称为策略网络）的输入，并且让策略网络输出此状态下所有动作对应的Q值；记策略网络的参数为，则在DQN中，每个状态-动作对的Q值可以表示为。为了提高学习的稳定性，DQN算法引入了经验回放机制。经验回放机制即利用经验池存储智能体与环境交互的经验，并从经验池中随机采样进行训练；经验回放机制打破了数据之间的相关性，从而满足一般深度学习中样本独立同分布的假设。智能体与环境交互的每一次交互都会将经验存储在一个经验回放池 中;每次训练时，智能体从经验回放池中随机采样一批数据进行训练。为了避免Q值估值过高的问题，DQN引入了目标网络机制；目标网络是一个与策略网络结构相同的网络，但其参数不随着训练更新，而是每轮将策略网络的参数复制过来；记目标网络的参数为，则目标网络输出的每个状态-动作对的Q值可以表示为。在DQN中，策略网络的损失函数为：

也就是说，损失函数的目标是最小化时序差分误差。通过损失函数的监督，策略网络的参数就可以随着训练更新。

在此基础上，本文将MA-CTD扩展到深度强化学习的框架下。在上部分推导的MA-CTD公式中，智能体通过两个Q函数来分别预测某个状态-动作对获得的回报的均值和方差。在深度强化学习的框架下，本文通过为智能体设计两个神经网络来分别拟合预测回报的均值和方差的Q函数。在DQN的基础之上，令预测回报的方差的策略网络（以下简称为方差策略网络）的参数为，则每个状态-动作对获得的回报的方差可以表示为。同理，本文为避免值估值过高，为方差策略网络引入了方差目标网络；方差目标网络与方差策略网络结构相同，其参数不随着训练更新，而是定期从方差策略网络复制过来。记方差目标网络的参数为，则方差目标网络输出的每个状态-动作对的值为。在深度强化学习框架下，方差时序差分误差可表示为：

因此，方差策略网络的损失函数为：

也就是说，方差策略网络的损失函数的目标就是最小化方差时序差分误差。通过损失函数的监督，方差策略网络的参数就可以随着训练更新。在深度强化学习框架下，MA-CTD算法如下所示：



图表 4：深度强化学习下的MA-CTD算法。

值得注意的是，策略网络和方差策略网络是两个不同的网络，且需要在不同的损失函数的监督下更新参数；因此，这两个网络训练时应采用不同的优化器。方差策略网络的更新依赖于策略网络计算出的Q值；因此，方差策略网络采用的学习率应略小于策略网络的学习率，且方差目标网络的更新频率应该慢于目标网络的更新频率，这样可以保证方差策略网络学习的稳定性。

## 多智能体强化学习下MA-CTD的应用

在多智能体环境中，我们为每个智能体设计两个神经网络（或Q表格），分别用来预测智能体在某个状态-动作对可能获得的回报的均值和方差；智能体将根据回报的均值和方差选择策略。在每次与环境交互后，对智能体的神经网络（或Q表格）进行更新。由于神经网络较Q表格更为复杂，因此本文接下来给出MA-CTD算法在深度强化学习框架下的算法伪代码：

表格 1：MA-CTD算法伪代码

|  |
| --- |
| 算法输入：状态-动作价值函数以及方差状态-动作价值函数的学习率和；初始策略网络和方差策略网络和；初始目标网络和方差目标网络和；训练轮数；智能体；初始空经验池；经验池最大容量；训练批次大小；目标网络更新频次；  算法输出：经过训练后的策略网络和方差策略网络和。 |
| 1. 初始化当前轮数； 2. 拷贝网络参数到网络，拷贝网络参数到网络； 3. while do 4. while 当前环境未结束 do 5. for in do 6. 从环境中获取当前观测； 7. 根据MA-CTD下的-贪心策略选择动作； 8. 执行动作，并从环境中获取下个观测以及奖励； 9. 将此次与环境交互的经验存储到经验池中； 10. if do 11. 将最远一次经验从经验池中删除； 12. if do 13. 根据策略网络和方差策略网络的损失函数及学习率更新两个网络的参数； 14. if do 15. 将目标网络和方差目标网络的参数与其相应的策略网络同步； 16. 算法结束，得到训练好的策略网络和方差策略网络和。 |

由以上论述，可以发现MA-CTD算法还有着“即插即用”的优秀特性，这意味着现有的MARL算法可以引入额外的方差网络来扩展其功能，并用本文推导出的方差网络损失函数来对方差网络进行训练。这种插入不仅可以保存原MARL方法的优势，还可以进一步提升原MARL方法下智能体决策的鲁棒性以及学习的稳定性；且方差网络的插入仅使原MARL方法增加了一个方差网络，对原MARL方法的效率不会由显著影响。

接下来，本文将以上面介绍过的VDN为例，介绍MA-CTD算法和VDN的结合：MA-CTD-VDN。在VDN中，每个智能体都是由一个深度神经网络构成的，假设神经网络的参数为，团队的Q值可以表示为各个智能体的Q值之和，即：

训练过程中，可以根据以下的损失函数来对各个智能体的网络参数进行联合优化：

上式中即为损失函数，是智能体在时刻收到的团队奖励。结合MA-CTD的建模，为每个智能体添加一个用于预测方差值的深度神经网络；则团队的方差值可以表示为各个智能体的方差值之和，即：

训练过程中，可以根据MA-CTD推导出的方差网络的损失函数对各个智能体方差网络的参数进行联合优化：

在损失函数和的监督下，MA-CTD-VDN中智能体的网络参数就可以进行更新，每个智能体就可以按照回报的均值-方差的线性组合进行策略选择。

## MA-CTD算法复杂度分析

本章将以上面提到的VDN作为基本算法，对MA-CTD-VDN算法与VDN算法的复杂度进行对比分析。令训练轮数为，环境中智能体的数量为，智能体在每轮中与环境交互次，假设每个智能体的网络参数的数量都是。接下来，本文将分别对VDN和MA-CTD-VDN算法的时间复杂度和空间复杂度进行分析。

首先，算法的时间复杂度通常指的是算法中浮点运算（Floating-point operation，flop）的次数，而一次flop通常被定义为两个浮点数之间的一次加减乘除运算。为了简化分析过程，这里将策略网络进行一次前向传播的时间复杂度记作。对于VDN算法，训练共轮，每轮中个智能体会做出次决策，每次决策智能体利用策略网络的前向传播计算状态-动作对的值；每隔轮，目标网络的参数会与策略网络进行同步。因此，VDN算法训练的时间复杂度为：

对于MA-CTD-VDN算法，假设方差策略网络和策略网络架构一致。在轮训练中，个智能体会做出次决策，每次决策智能体利用策略网络和方差策略网络的前向传播计算状态-动作对的和值；每隔轮，目标网络和方差目标网络的参数会与策略网络和方差策略网络进行同步。因此，MA-CTD-VDN算法训练的时间复杂度为：

在一般计算时间复杂度的多项式中，通常只保留次数之和最高的项且忽略该项的常数。因此，对VDN算法和MA-CTD-VDN算法的时间复杂度进行整理，可以发现，它们的时间复杂度都是：

接下来，本文对算法的空间复杂度进行分析。算法的空间复杂度通常指算法运行过程中所占的内存的大小。在VDN算法中，每个智能体都需要维护目标网络和策略网络。因此，VDN算法的空间复杂度为。

在MA-CTD-VDN算法中，每个智能体还需要额外维护方差目标网络和方差策略网络。因此，MA-CTD-VDN算法的空间复杂度为。

一般计算空间复杂度的多项式也会只保留次数之和最高的项且忽略该项的常数。因此，VDN算法和MA-CTD-VDN算法的空间复杂度都是：

也就是说，MA-CTD算法仅仅是添加一个额外的网络让智能体进行回报的方差的预测，MA-CTD-VDN算法的时间、空间复杂度与VDN算法是相同的。更一般地来说，MA-CTD算法仅会在原算法的基础上添加用于预测方差的网络或表格。因此，MA-CTD算法作为即插即用的算法，并不会增加原算法的时间、空间复杂度。

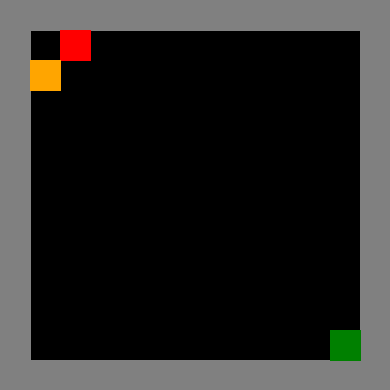
# 实验

为了验证MA-CTD算法的有效性，本文将实现IQL[18]和VDN[19]两种算法作为基准算法，并在此基础上实现MA-CTD-IQL算法与MA-CTD-VDN算法，在本文设计的高方差的环境下比较它们与基准算法的性能。接下来，本文将详细介绍设计的环境、算法实现细节、实验结果、MA-CTD算法的消融实验，并关于实验做出讨论。

## 多智能体实验环境介绍

本文将在真实环境下验证提出的MA-CTD算法的有效性；因此，本文将先对实验采用的多智能体强化学习环境进行介绍。为了直观地验证MA-CTD算法的有效性，本文设计了两个稀疏奖励的环境；在稀疏奖励的情况下，智能体训练中接收到的有效的奖励很少，训练过程很可能是不稳定、高方差的。因此，在稀疏奖励环境下MA-CTD算法的有效性可以被更好地说明。接下来，本文将分别介绍设计的两个环境，分别命名为寻宝和收集。

寻宝环境是一个基于网格世界的环境，环境尺寸为 11×11 的网格。环境中的智能体被标记为红色和橙色以便于区分，四周墙壁被标记为灰色，待收集的宝物被标记为绿色，固定在地图的右下角；如下图所示。



图表 5：寻宝环境图片

游戏的目标是：两个智能体中的一个收集到宝物即可；规定智能体走到宝物格子即判定为收集。游戏在以下两种情况会结束：某一个智能体收集到宝物或某一智能体移动超过70步。每个智能体可以选择 4 种动作：向上、向下、向左或向右移动，如果只能提与墙壁碰撞，则智能体会保持不动。

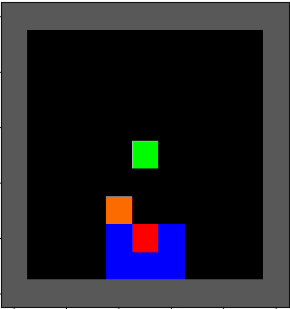
在游戏中，智能体与环境交互可以获得的奖励为：

表格 2：寻宝环境奖励设置。

|  |  |  |
| --- | --- | --- |
| 事件/奖励 | 智能体1 | 智能体2 |
| 智能体1找到宝物 | +1 | +1 |
| 智能体2找到宝物 | +1 | +1 |
| 任一智能体移动一步 | -0.001 | -0.001 |

显然，在寻宝环境中，两个智能体的最优决策是：在不触碰墙壁的情况下，向下或向右移动，直至其中一个智能体可以收集宝物。

收集环境也是一个基于网格世界的环境，环境尺寸为 11×11 的网格。环境中的智能体被标记为红色和橙色以便于区分，待收集食物的被标记为绿色，家被标记为蓝色，墙壁被标记为灰色，如下图所示。



图表 6：收集环境图片。

游戏的目标是：其中一个智能体需要吃掉食物，然后两个智能体都需要返回家。游戏在以下两种情况下会结束：食物被吃掉且两个智能体都到达家区域时，或者某个智能体移动超过70步。为表现智能体之间的合作，当食物被吃掉时，如果两个智能体都紧挨着食物，它们将获得额外的奖励。每个智能体可以选择 4 种动作：向上、向下、向左或向右移动，如果移动到墙壁或与其他智能体碰撞，智能体会保持在原地。

在游戏中，两个智能体会随机在家中的某个位置出生，食物会随机在家以外的位置刷新。值得注意的是，当一个智能体吃掉食物且另一个智能体在食物的两格之内时，两个智能体获得的团队奖励会更高一些。因此，两个智能体的最优决策为：两个智能体都向着食物的方向移动，一个智能体到达食物附近时，如果另一个智能体没有在食物附近，则在附近移动等待另一个智能体；当食物被收集时，两个智能体一起向家的方向移动。

智能体与环境交互获得的奖励为：

表格 3：收集环境奖励设置。

|  |  |  |
| --- | --- | --- |
| 事件/奖励 | 智能体1 | 智能体2 |
| 智能体1吃掉食物 | +10 | +0 |
| 智能体2吃掉食物 | +0 | +10 |
| 吃掉食物后两个智能体都到家 | +20 | +20 |
| 智能体1吃掉食物且智能体2在附近 | +6 | +6 |
| 智能体2吃掉食物且智能体1在附近 | +6 | +6 |
| 任一智能体移动一步 | -0.1 | -0.1 |
| 智能体1撞到墙壁 | -1 | -0 |
| 智能体2撞到墙壁 | -0 | -1 |
| 两个智能体碰撞 | -5 | -5 |

## 算法实现细节

本章节将详细描述本文中IQL、VDN、MA-CTD-IQL和MA-CTD-VDN这四种算法的实现细节。首先，算法中一些超参数的设计如下表所示。

表格 4：算法超参数设置。

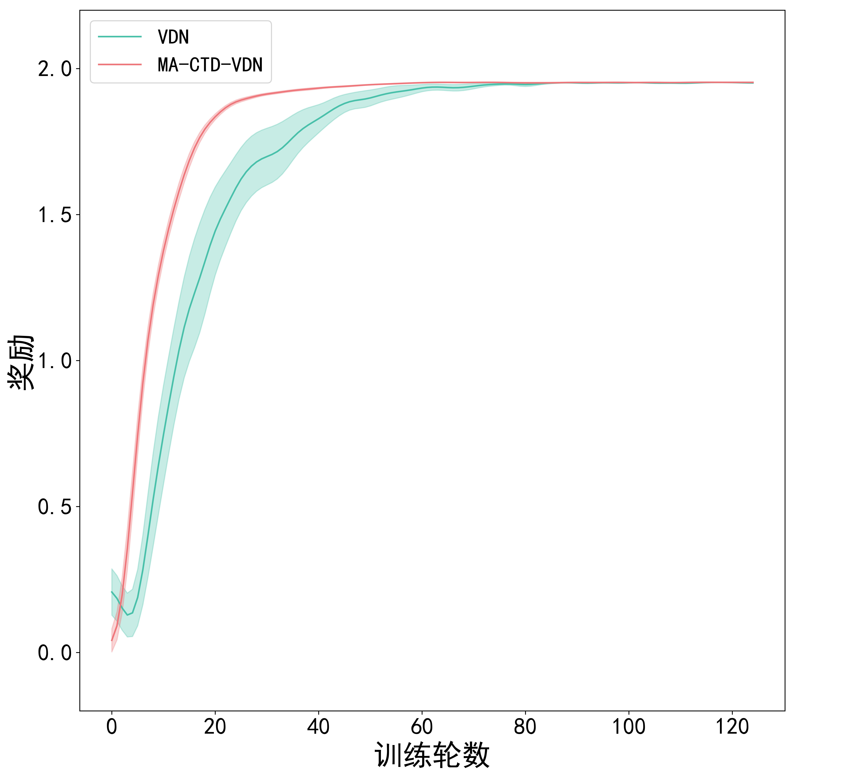
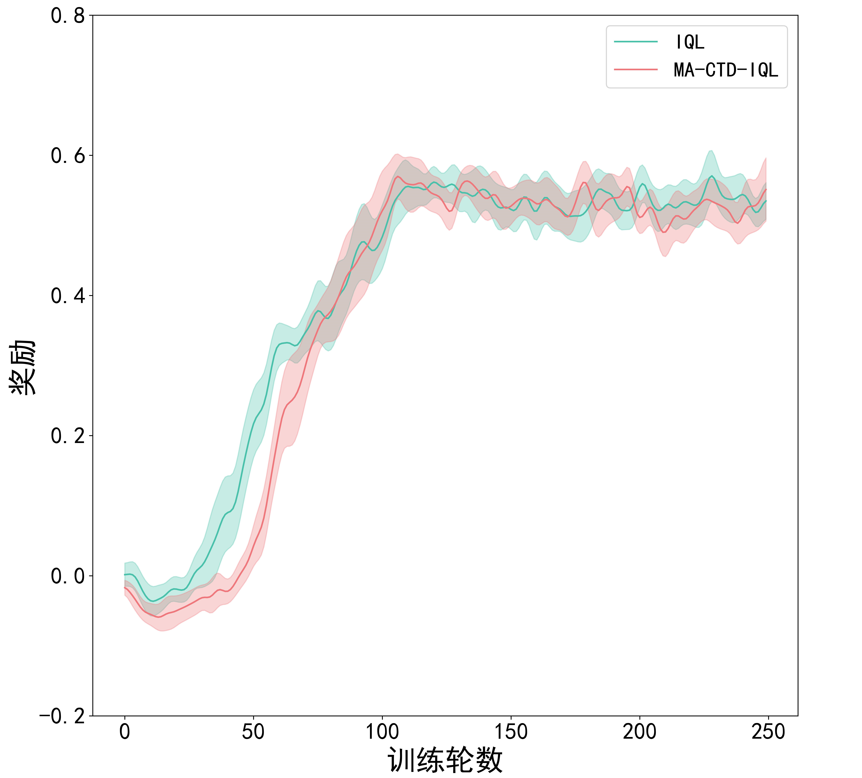
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 超参数/算法 | IQL | VDN | MA-CTD-IQL | MA-CTD-VDN |
| 学习率 | 1e-2 | 1e-3 | 1e-2 | 1e-3 |
| 方差学习率 | - | - | 1e-4 | 1e-4 |
| 风险系数 | - | - | 1e-1 | 1e-1 |
| 训练轮数 | 50000 | 25000 | 50000 | 25000 |
| 折扣因子 | 0.99 | 0.99 | 0.99 | 0.99 |

在IQL和MA-CTD-IQL算法中，每个智能体独立维护一张Q表格，Q表格的所有值都被初始化为0。在VDN和MA-CTD-VDN算法中，每个智能体维护一个具有三个全连接层的神经网络，神经网络参数初始化为全0；训练过程中，批次大小被设置为32，经验回放池的大小被设置为20000，目标网络的参数每10轮与策略网络同步，所有网络参数都是用Adam[27]进行优化。在两种算法的训练中，每轮的值是由以下的衰减公式计算得到的：

上面的公式中，表示取和的最大值。上式每轮结束后被执行一次运算，的值被初始化为。

## 实验结果

本章节将通过对比实验的结果展示MA-CTD算法的有效性。首先，本文在寻宝环境中实现了IQL、VDN、MA-CTD-IQL和MA-CTD-VDN四种算法。为了对比实验的公平，本文保证IQL和MA-CTD-IQL算法的训练轮数、学习率等设置是一样的，VDN和MA-CTD-VDN的算法设置是一样的。由于本文的环境设计均为合作式MARL环境，因此，本文以团队奖励作为算法的奖励指标。在寻宝环境中，各个算法训练中奖励的阴影折线图如下图所示。



图表 7：IQL、VDN算法与这两种算法结合MA-CTD后的算法在寻宝环境上训练的阴影折线图。

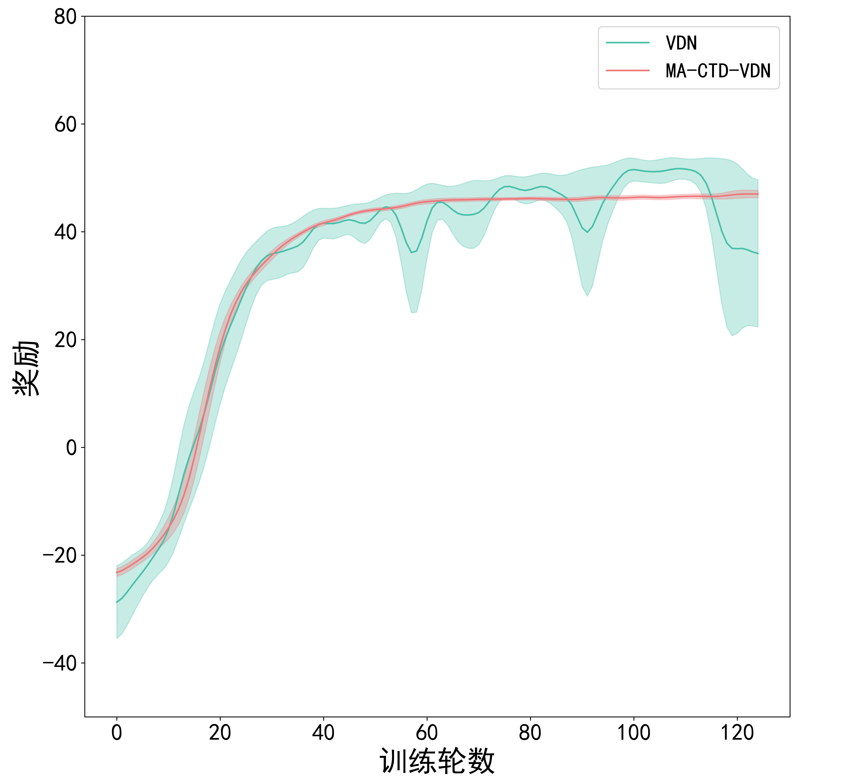
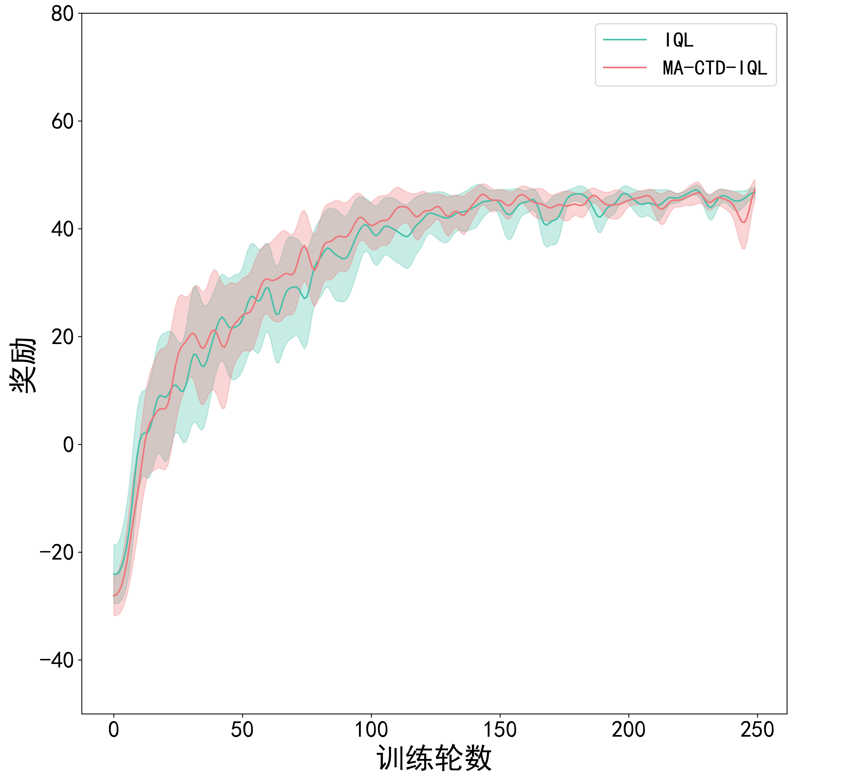
各个算法训练中奖励的阴影折线图如上图所示。本文在寻宝环境上将每个算法代码分别运行了五次，最终得到了这样的阴影折线图。图中横轴代表轮数，纵轴代表团队获得的奖励；由于轮数过多影响可视化的效果，所以本文对阴影折线图进行了平滑处理。为了方便对比，本文将IQL算法和MA-CTD-IQL算法的阴影折线图合并在一起，将VDN算法和MA-CTD-VDN算法的阴影折线图合并在一起。在阴影折线图中，深色的实线代表五次训练中获得奖励的平均值，浅色的阴影部分代表五次训练中获得的奖励的方差。因此，阴影折线图的比较可以反映出算法性能的优劣。为了更具体地比较，本文从数据上计算了每种算法的阴影折线图中的奖励的方差，以反映智能体训练过程中的稳定性；并计算了最后100轮的奖励的均值，作为该算法智能体最终获得的奖励。具体的计算数据如下表所示。

表格 5：算法在寻宝环境上奖励与方差的对比。

|  |  |  |
| --- | --- | --- |
| 算法 | 最终奖励 | 奖励方差 |
| IQL | 0.5341 | 7.8973 |
| MA-CTD-IQL | **0.5517** | **6.8843** |
| VDN | 1.9528 | 2.0545 |
| MA-CTD-VDN | **1.9547** | **0.9587** |

上表中，最终奖励即每个算法最后100轮的奖励的均值，奖励方差即阴影折线图中浅色的阴影部分面积。在五次训练中，智能体的-贪心策略、环境的随机种子、经验回放池中采样等随机因素会导致每次训练的结果都有所差异。因此，这五次训练得到的奖励的方差可以很好地反映算法的稳定性。从训练数据中可以看出，结合了MA-CTD算法建模的IQL、VDN算法训练过程中的奖励方差明显更小。其中，MA-CTD-VDN算法的奖励方差小于VDN算法的奖励方差的一半；从阴影折线图中也可以看出，MA-CTD-VDN算法的阴影部分明显少于VDN算法的阴影部分。除此之外，结合了MA-CTD算法建模的IQL、VDN算法最终得到的奖励也会略高于原IQL、VDN算法。而IQL、MA-CTD-IQL两个算法较VDN、MA-CTD-VDN两个算法的表现差，是由于IQL算法建模简单，将每个智能体以外的智能体视作环境的一部分，从而导致环境的非稳态性，导致算法难以收敛。然而，IQL和VDN这两种算法在结合MA-CTD建模之后，训练过程的稳定性以及最终学习到的策略的好坏都有一定提升。

接下来，本文在收集环境上实现了IQL、VDN、MA-CTD-IQL和MA-CTD-VDN四种算法。同样，为了对比实验的公平性，IQL和MA-CTD-IQL算法、VDN和MA-CTD-VDN算法的配置是相同的。实验结果如下所示。



图表 8：IQL、VDN算法与这两种算法结合MA-CTD后的算法在收集环境上训练的阴影折线图。

同样，为了更具体地比较，本文将各个算法的最终奖励和奖励方差进行了计算，计算结果如下表所示：

表格 6：算法在收集环境上奖励与方差的对比。

|  |  |  |
| --- | --- | --- |
| 算法 | 最终奖励 | 奖励方差 |
| IQL | 44.88 | 14.2292 |
| MA-CTD-IQL | **46.40** | **12.6167** |
| VDN | 48.12 | 13.6971 |
| MA-CTD-VDN | **49.00** | **7.5413** |

从上表中可以看出，结合了MA-CTD算法建模的IQL、VDN算法训练过程中的奖励方差明显更小。其中，MA-CTD-VDN算法的奖励方差大约仅有VDN算法的一半。在收集环境中，结合了MA-CTD算法建模的IQL、VDN算法最终得到的奖励也会略高于原IQL、VDN算法。

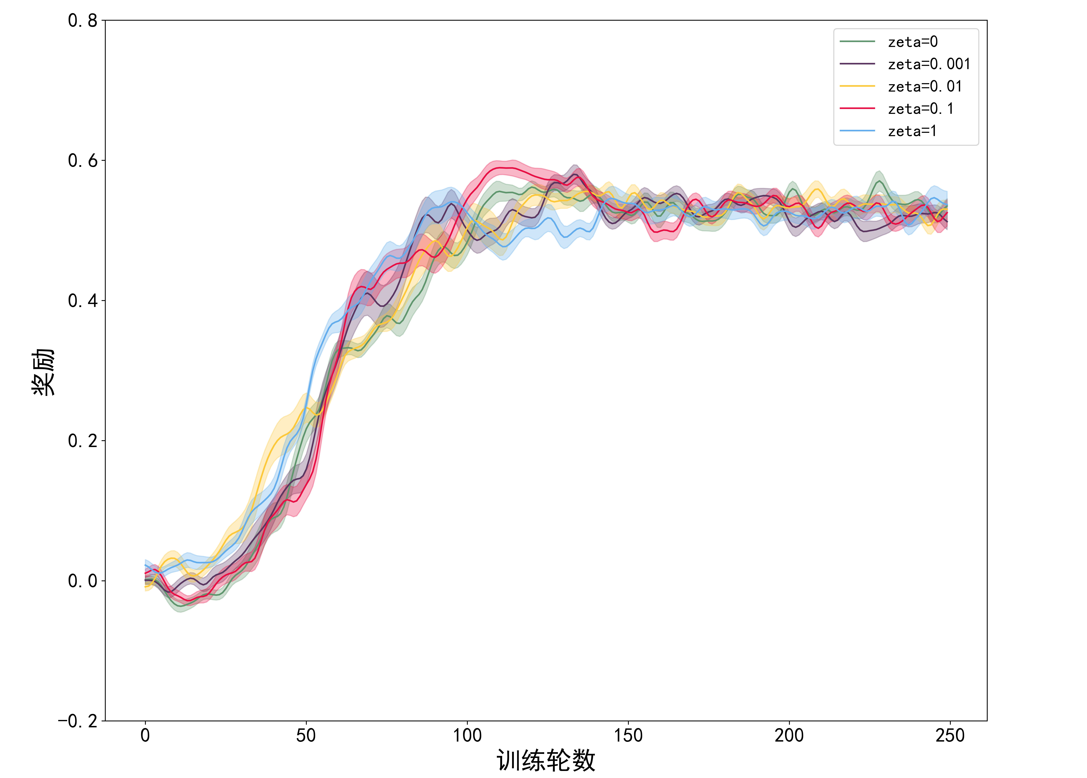
综合以上实验，可以得出结论：MA-CTD算法的均值-方差建模可以有效地提高现有算法的性能。在与MA-CTD算法结合后，原算法，即IQL和VDN算法中智能体最终获得的奖励略有提升，训练过程中奖励折线的方差有明显降低。智能体最终获得的奖励更高，说明MA-CTD算法通过让智能体根据回报的均值和方差综合选择动作，从而学习到更鲁棒的策略；智能体训练过程中奖励折线的方差明显降低，说明MA-CTD算法通过对方差进行建模减少了环境等随机性的干扰，使得智能体学习过程更稳定。

## 消融实验

MA-CTD算法的设计使得算法中多了两个关键参数，即风险系数和方差网络（或方差表格）的学习率。因此，本章节旨在对这两个参数的设置进行消融实验，选取最合适的设置。值得注意的是，不同MARL算法的学习率设置是不一样的，因此，方差学习率在不同的算法中应该也是不同的。因此，本文的消融实验主要探讨和的倍数关系，而不单单是值的设置。

### 风险系数消融实验

首先，本文将对风险系数的取值进行消融实验。为了测试不同的值对算法的影响，本文在寻宝环境上训练不同值的MA-CTD-IQL算法，并对比它们的性能。因此，本文设置值为0、0.001、0.01、0.1、1，分别测试算法的性能。值得注意的是，当时，智能体的策略将不考虑回报的方差，此时的MA-CTD-IQL算法就等同于IQL算法。值为0的MA-CTD-IQL算法训练的阴影折线图见图表6；值为0.001、0.01、0.1和1时算法训练的阴影折线图如下图所示。



图表 9：ζ的不同取值对应的MA-CTD-IQL算法在寻宝环境下训练的阴影折线图。

由于这些阴影折线图的重复地方较多，本文对这些阴影折线图中的阴影部分做了等比例的缩小，以便更清晰的展示。不同值下算法具体的表现如下表所示。

表格 7：不同ζ取值下MA-CTD-IQL算法的奖励与方差的对比。

|  |  |  |
| --- | --- | --- |
| 值 | 最终奖励 | 奖励方差 |
| 0 | 0.5341 | 7.8973 |
| 0.001 | 0.5303 | 6.9853 |
| 0.01 | 0.5373 | 7.0711 |
| 0.1 | **0.5517** | **6.8843** |
| 1 | 0.5098 | 7.9650 |

从上表可以看出，值为0.1时算法的表现最好；值为0.001和0.01时MA-CTD-IQL算法表现相近；值为1时MA-CTD-IQL算法的奖励方差（7.9650）甚至高于IQL算法（7.8973），最终奖励也更低；这说明了值的取值十分重要。从消融实验结果可以看出，值设置为0.1时可以使算法达到最好的性能，设置为0.01时也会获得比原算法更好的性能。

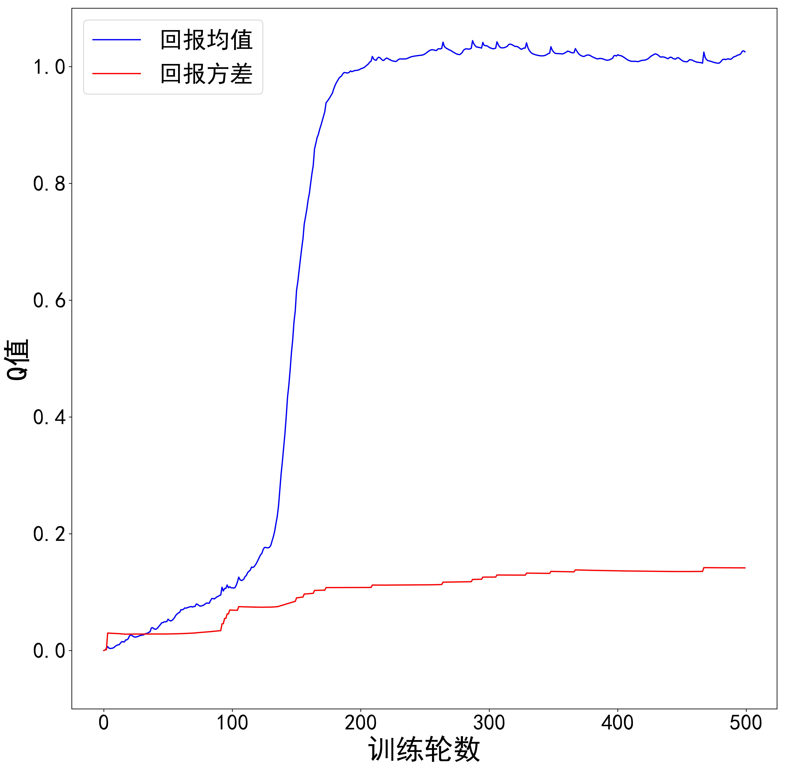
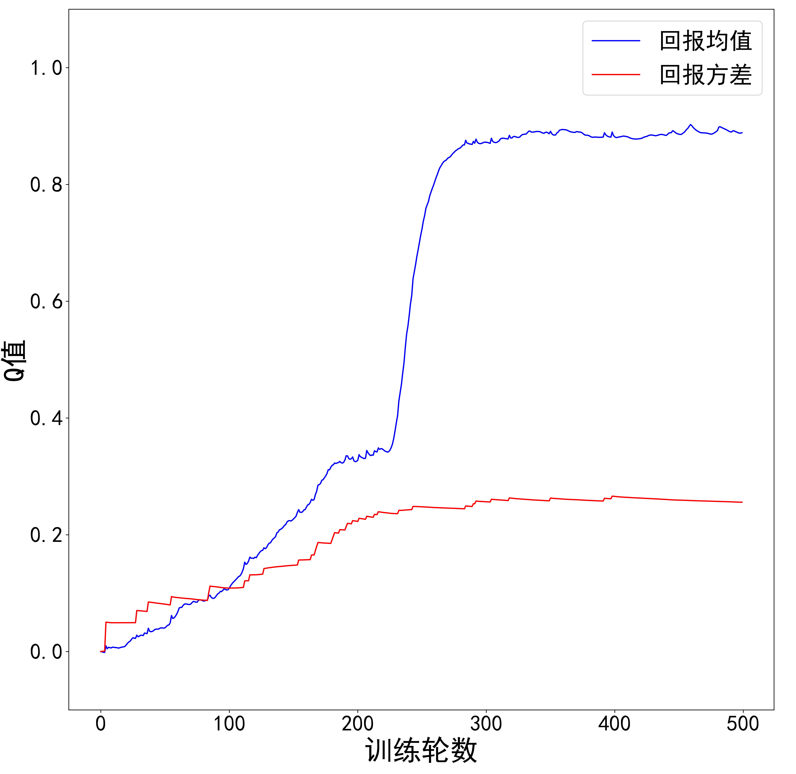
为了更好验证值设置为0.1的普适性，本文在寻宝环境上对MA-CTD-VDN算法不同的取值进行了消融实验，实验结果如下表所示：

表格 8：不同ζ值下MA-CTD-VDN算法的奖励与方差的对比。

|  |  |  |
| --- | --- | --- |
| 值 | 最终奖励 | 奖励方差 |
| 0 | 1.9528 | 2.0545 |
| 0.001 | 1.9532 | 1.5348 |
| 0.01 | 1.9530 | 0.9819 |
| 0.1 | **1.9547** | **0.9587** |
| 1 | 1.4951 | 2.2670 |

可以看出，值设置为0.1是MA-CTD-VDN算法的表现仍然是最好的；也就是说，在MA-CTD算法中，将值设置为0.1可以使算法达到最好的性能。在前文中推导出，智能体应该以的值做出决策；根据的最优取值，可以得出结论：智能体应该更关注获得回报的期望值（），将获得回报的方差（）作为辅助决策的依据。如果智能体给回报的方差更多考虑，即值设置的过大，则会导致算法收敛性更差，训练过程也更加不稳定。

更进一步，本文将以特定的动作-状态对为例，展示回报的期望和方差在学习过程中的变化。考虑MA-CTD-IQL算法中某个智能体处于寻宝环境的起点时，做出向上和向下移动的动作的状态-动作对；其中向下移动是更靠近目标的，向上移动则是于墙壁发生碰撞。采用值为0.1的最优设置，在训练过程中，这两个状态-动作对的和的值的变化如下图所示。

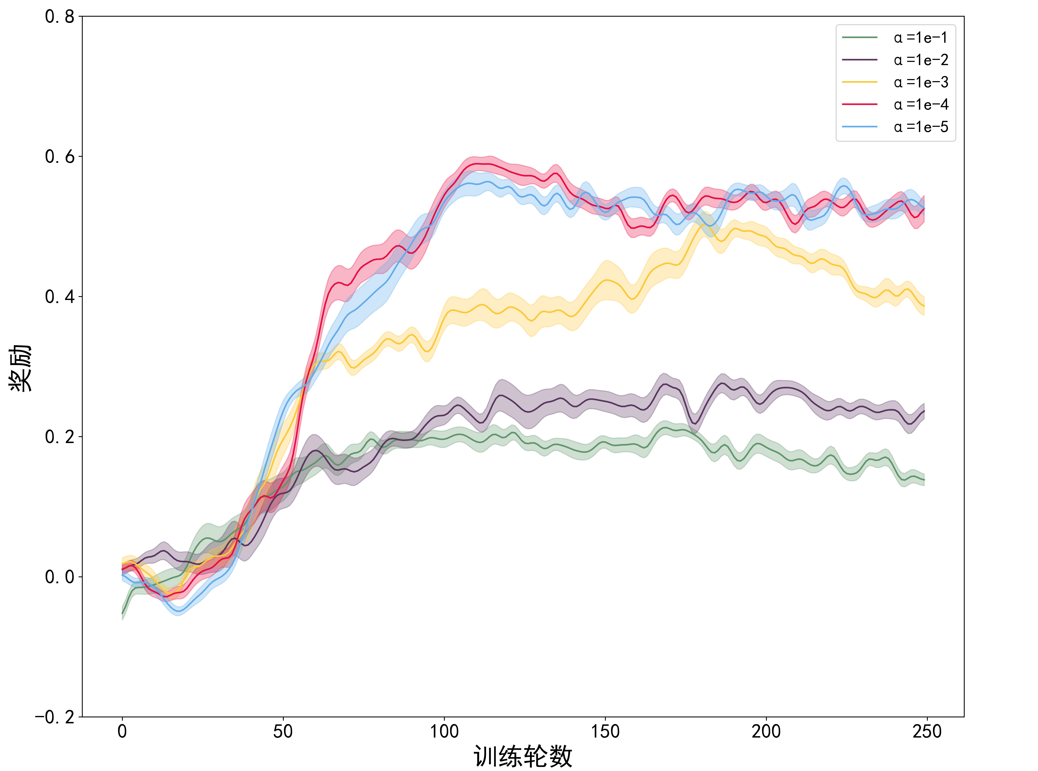


图表 10：训练过程中某一智能体和的值的变化。

上面两图中蓝色的线代表值的变化，红色的线代表值的变化；左边的图片是在起点处向上移动的情况，右边的图片是在起点处向下移动的情况。由于向上移动相当于没有移动，会获得负奖励；而向下移动虽然也会获得负奖励，但向下移动使得智能体与目标的距离更近；所以向下移动的值要大于向上移动的值。向上移动相当于没有移动，向下移动使智能体与终点更近；也就是说，向下移动会导致智能体与终点距离的缩短，使得智能体到终点路线上的不确定性变小；所以向下移动的值要小于向下移动的值。因此，根据智能体的的动作选择策略，智能体最终会做出向下移动的决策。

### 学习率、方差学习率消融实验

接下来，本文将对MA-CTD算法中的学习率、方差学习率的关系进行消融实验。为了探索和的倍率关系，本文调整了寻宝环境上MA-CTD-IQL算法的参数：先将固定，探索不同取值，即不同和的倍率对算法的影响。首先，本文将固定1e-2，并设置了1e-1、1e-2、1e-3、1e-4、1e-5这五个取值。在不同的取值下MA-CTD-IQL算法训练的阴影折线图如下图所示。



图表 11：不同和的倍率对应的MA-CTD-IQL算法在寻宝环境下训练的阴影折线图。

从上图可以看出，不同的和的倍率对算法的影响是较大的。更具体来说，各个取值下的MA-CTD-IQL表现如下表所示：

表格 8：不同和的倍率下算法的奖励与方差的对比。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 取值 | 和比值 | 最终奖励 | 奖励方差 |
| 1e-2 | 1e-1 | 1:10 | 0.1464 | 5.3655 |
| 1e-2 | 1e-2 | 1:1 | 0.2416 | 5.7509 |
| 1e-2 | 1e-3 | 1:0.1 | 0.4015 | 6.8907 |
| 1e-2 | 1e-4 | **1:0.01** | **0.5517** | **6.8843** |
| 1e-2 | 1e-5 | 1:0.001 | 0.5305 | 7.1789 |

从上表中可以看出，取值1e-4，即和的比值为1:0.01时算法的效果是最好的。虽然和的比值为1:10、1:1时算法在训练中产生的奖励方差小，但是算法收敛性很差，最终奖励很低。而和的比值为1:0.001时，最终奖励和奖励方差比和的比值为1:0.01时略差一些。为了更好地说明和的比值对算法产生的影响，而不是某个固定的值下的结果，本文接下来调整了值为1e-3，并按照1:10、1:1、1:0.1、1:0.01、1:0.001的比值设置的值，进行了额外的消融实验，实验结果如下表所示。

表格 9：不同和的倍率下算法的奖励与方差的对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 取值 | 和比值 | 最终奖励 | 奖励方差 |
| 1e-3 | 1e-2 | 1:10 | 0.0673 | 3.3693 |
| 1e-3 | 1e-3 | 1:1 | 0.0343 | 3.1966 |
| 1e-3 | 1e-4 | 1:0.1 | 0.4030 | 5.5797 |
| 1e-3 | 1e-5 | **1:0.01** | **0.4721** | **5.4107** |
| 1e-3 | 1e-6 | 1:0.001 | 0.4144 | 5.8277 |

从上表中可以看出，仍然是设置和的比值为1:0.01时算法的效果是最好的。当和的比值为1:10、1:1时，算法中智能体获得的最终奖励仍然很低，说明算法没有收敛。也就是说，智能体应该更主要学习回报的期望值（），在期望值的基础上再学习回报的方差（）。

## 讨论

在实验章节的最后，本文将对提出的MA-CTD算法、对比实验结果以及消融实验结果进行讨论。

本文针对多智能体环境下随机性过大的挑战提出了MA-CTD算法。现有的MARL算法基于贝尔曼方程，对智能体获得某个观测时做出某个动作可以获得的回报的期望，即进行建模。然而，在随机性过大的情况下，回报这一随机变量的分布很可能是高方差的。因此，现有算法仅仅对回报的期望的建模不足以精确刻画多智能体环境中智能体可能获得的回报的情况。因此，本文从一般的贝尔曼方程出发，进一步推导出回报的方差的表达公式，并将回报的方差的时序差分更新公式进行了创新性的推导；这就使得智能体可以对回报的均值和方差进行建模，并根据回报的均值和方差进行决策。

由于当下深度神经网络的火热，现有MARL算法大多是基于深度神经网络的。因此，本文更进一步将MA-CTD算法扩展到了深度强化学习的情况下，推导出了智能体通过神经网络预测回报的方差时神经网络训练的损失函数。这样，智能体就可以在现有MARL算法的基础上，通过增加一个网络来对回报的方差进行建模。

为了更好的验证MA-CTD算法的有效性，本文设计了两个高方差的多智能体环境，并将现有MARL方法和结合了MA-CTD算法的MARL方法进行对比实验。实验结果表明，MA-CTD算法在插入到现有MARL方法后，可以帮助智能体学习到更好的策略，且在高方差的环境下学习过程更稳定。也就是说，MA-CTD算法对智能体获得的回报的均值和方差同时建模的思路可以帮助智能体应对环境或自身算法中随机性的干扰。

在MA-CTD算法的实现中，涉及风险系数以及方差学习率的设定。通过本文的消融实验表明，风险系数设置为0.1是最优的参数设置，即智能体应该以的策略选择动作。也就是说，在智能体选择动作时，应该主要关注回报的均值；如果某个高回报的动作方差过大，智能体就不会选择这个高回报的动作。消融实验还表明，MA-CTD中网络（或表格）与网络（或表格）的学习率比值应该保持在1：0.01的设置。根据MA-CTD的公式推导，的更新公式依赖的更新公式中的时序差分误差项，也就是说，更新公式中的时序差分误差项的准确性会影响的学习。因此，的学习率应该低于的学习率，使得可以在较准确的时序差分误差项的基础上进行更新。

# 结论与展望

多智能体环境中环境、多个智能体的共同决策等因素的随机性过大，而现有的MARL方法基于贝尔曼方程对智能体获得的回报的期望进行建模，无法对随机性进行建模。因此，现有MARL方法仍然面临着过大的随机性导致的智能体学习过程不稳定，学习到的策略不够鲁棒的挑战。

针对上述挑战，本文提出可以对多智能体环境中的随机性进行建模的MA-CTD算法。MA-CTD算法通过对智能体获得的回报的期望和方差同时建模，让智能体对回报有更精准的刻画。本文从智能体获得回报的期望（）出发，推导出智能体获得的回报的方差（）的表达公式，并进一步推导出训练过程中的更新公式，从而同时对智能体获得的回报的期望和方差进行建模。更进一步，本文将MA-CTD算法扩展到深度强化学习的情况，使得现有MARL方法可以直接结合MA-CTD算法对回报的方差进行建模，从而提升算法性能。在稀疏奖励、高方差的多智能体环境上的实验验证表明，结合了MA-CTD算法的现有MARL方法的学习过程更加稳定，且最终学习到的决策更加鲁棒。

然而，MA-CTD算法中通过限制折扣因子的取值完成了近似，这就导致智能体在方差的学习中会存在一定误差。在未来的工作中，我计划通过更多数学工具对MA-CTD算法中回报的方差建模的公式进行优化；并且在更复杂的多智能体环境中，将MA-CTD算法与最先进的MARL方法结合进行应用。

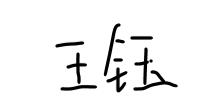
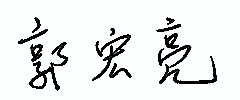
# 参考文献

1. Wooldridge, M. (2009). An introduction to multiagent systems. John wiley & sons.
2. Gautam, A., & Mohan, S. (2012, August). A review of research in multi-robot systems. In *2012 IEEE 7th International Conference on Industrial and Information Systems (ICIIS)* (pp. 1-5). IEEE.
3. Chen, D., Wang, H., Huo, Y., Li, Y., & Zhang, H. (2023). Gamegpt: Multi-agent collaborative framework for game development. *arXiv preprint arXiv:2310.08067*.
4. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484-489.
5. Bellman, R. (1966). Dynamic programming. *Science*, *153*(3731), 34-37.
6. Thrun, S., & Littman, M. L. (2000). Reinforcement learning: an introduction. AI Magazine, 21(1), 103-103.
7. Jang, B., Kim, M., Harerimana, G., & Kim, J. W. (2019). Q-learning algorithms: A comprehensive classification and applications. IEEE Access, 7, 133653-133667.
8. Watkins, C. J. C. H. (1989). Learning from delayed rewards.
9. Watkins, C. J., & Dayan, P. (1992). Q-learning. Machine Learning, 8, 279-292.
10. Riedmiller, M. (2005). Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In Machine Learning: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. proceedings 16 (pp. 317-328). Springer Berlin Heidelberg.
11. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.
12. Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 30, No. 1).
13. Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016, June). Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning* (pp. 1995-2003). PMLR.
14. Guo, H., Hou, X., & Peng, Q. (2021). CTD: Cascaded temporal difference learning for the mean-standard deviation shortest path problem. IEEE Transactions on Intelligent Transportation Systems, 23(8), 10868-10886.
15. Oroojlooy, A., & Hajinezhad, D. (2023). A review of cooperative multi-agent deep reinforcement learning. Applied Intelligence, 53(11), 13677-13722.
16. 罗彪,胡天萌,周育豪,等.多智能体强化学习控制与决策研究综述[J/OL].自动化学报,1-30[2024-11-24].https://doi.org/10.16383/j.aas.c240392.
17. Zhang, K., Yang, Z., & Başar, T. (2021). Multi-agent reinforcement learning: A selective overview of theories and algorithms. Handbook of Reinforcement Learning and Control, 321-384.
18. Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., ... & Vicente, R. (2017). Multiagent cooperation and competition with deep reinforcement learning. PloS one, 12(4), e0172395.
19. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., ... & Graepel, T. (2017). Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:1706.05296.
20. Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., & Whiteson, S. (2020). Monotonic value function factorisation for deep multi-agent reinforcement learning. Journal of Machine Learning Research, 21(178), 1-51.
21. Son, K., Kim, D., Kang, W. J., Hostallero, D. E., & Yi, Y. (2019, May). Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In International Conference on Machine Learning (pp. 5887-5896). PMLR.
22. Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In Machine Learning Proceedings 1994 (pp. 157-163). Morgan Kaufmann.
23. Hu, J., & Wellman, M. P. (2003). Nash Q-learning for general-sum stochastic games. Journal of Machine Learning Research, 4(Nov), 1039-1069.
24. Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in Neural Information Processing Systems, 30.
25. Mnih, V. (2016). Asynchronous Methods for Deep Reinforcement Learning. arXiv preprint arXiv:1602.01783.
26. Amato, C., Chowdhary, G., Geramifard, A., Üre, N. K., & Kochenderfer, M. J. (2013, December). Decentralized control of partially observable Markov decision processes. In *52nd IEEE Conference on Decision and Control* (pp. 2398-2405). IEEE.
27. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

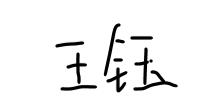
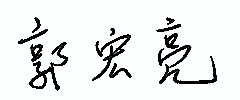
作者签名：  导师签名：

年 月 日

学位论文使用授权书

本学位论文作者完全了解四川大学有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或相关机构送交论文的原件、复印件和电子版，允许论文被查阅和借阅。本人授权四川大学将本学位论文的全部或部分内容编入有关数据库进行信息技术服务，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文，并用于学术活动。

（涉密学位论文在解密后适用于本授权书）

作者签名：  导师签名：

年 月 日