

涉密论文  公开论文

# 浙江大学

## 本科生毕业论文



题目 多智能体人机博弈攻防布局

学生姓名 谢文韬

学生学号 3150103626

指导教师 潘纲、王志坚

年级与专业 2015级计算机科学与技术

所在学院 计算机科学与技术

递交日期 2019年5月27日



## 浙江大学本科生毕业论文（设计）承诺书

1. 本人郑重地承诺所呈交的毕业论文（设计），是在指导教师的指导下严格按照学校和学院有关规定完成的。
2. 本人在毕业论文（设计）中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，也不包含为获得浙江大学或其他教育机构的学位或证书而使用过的材料。
3. 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。
4. 本人承诺在毕业论文（设计）工作过程中没有伪造数据等行为。
5. 若在本毕业论文（设计）中有侵犯任何方面知识产权的行为，由本人承担相应的法律责任。
6. 本人完全了解浙江大学有权保留并向有关部门或机构送交本论文（设计）的复印件和磁盘，允许本论文（设计）被查阅和借阅。本人授权浙江大学可以将本论文（设计）的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编本论文（设计）。

作者签名：

导师签名：

签字日期： 年 月 日 签字日期： 年 月 日



## 致谢

首先非常感谢两位毕业设计指导老师，潘纲老师和王志坚老师对我学术上的指导和无私的帮助，从论文的调研、选题、研究和写作上都进行了认真负责的把关，自己也非常幸运经历了这样一段有意义的科研工作。由衷感谢浙江大学计算机学院对我全方位的培养，我永远以成为浙江大学计算机学院的一分子为荣。

感谢 CCNT 实验室的邢东博士对我在强化学习相关问题的疑难进行的解答，感谢大三两位 SRTP 的同学郭经纬和张云淞进行的技术支持和配合。感谢 OpenAI 和 DeepMind 两个组织的科学家对强化学习的研究成果和教程，在完成毕业论文的过程中给了我众多的启发，让我少走了许多弯路。

最后，我要感谢自己的亲友在我大学四年，尤其是即将毕业的这段时间给予的支持和鼓励，我也希望自己能不辜负他们的期望。



## 摘要

随着人工智能领域的发展，智能机器人、无人驾驶和游戏智能等领域取得了众多研究进展，基于强化学习的理论也被广泛应用。在多智能体领域，单智能体的算法不再适用，多智能体间缺乏博弈的思想，环境的噪声影响算法的性能，这些问题需要通过合理的算法设计来解决。

本文通过一个经典的枪战博弈游戏，求解其存在的近似纳什均衡策略，改进了两个基于策略梯度的多体强化学习算法。改进近端策略优化通过工程细节的优化，提升了算法的奖励值和致胜程度。改进多智能体深度确定性策略梯度算法通过经验回放机制的改进，相较于原始算法提升了收敛速度、奖励值和致胜程度，并在与近似纳什均衡的对抗中取得了相当的致胜程度。改进算法提升性能的同时还为判断对手理性程度提供了依据，在特定环境中的理性行为掌握也达到了预期。

**关键词：**多智能体；强化学习；博弈论；经验回放



## Abstract

With the development of artificial intelligence,intelligent robot,automatic driving and game intelligence witness a number of progress.The theory based on reinforcement learning is widely applied to solve problems.In the field of multi-agents,algorithms in single-agent hardly work and there is a lack of game thinking among agents.The noise and uncertainty also challenges the algorithm performance,so reasonable algorithm design is needed to handle it.

The thesis optimizes two reinforcement learning algorithm based on policy gradient to fight against the approximate nash equilibrium in a classic gunfight game.Firstly,by improving the engineering details of Proximal Policy Optimization(PPO),it gains more reward and winning rate.Furthermore,the core work on optimize the policy in experience replay of Multi-Agent Deep Deterministic Policy Gradient(MADDPG) improve the speed of convergence,reward and winning rate.When it fights against the approximate nash equilibrium,it achieves a comparable winning rate.The optimized algorithm also provides evidences to judge the rationality of the opponent.The mastery of rational behavior in a given environment also met expectations.

**keywords:** Multi-Agents;Reinforcement Learning;Game Theory;Experience Replay



## 目录

### 第一部分 毕业论文

1	绪论	1
1.1	课题背景	1
1.2	研究现状	2
1.3	本文贡献	4
1.4	本文结构	4
2	多智能体相关理论技术	5
2.1	纳什均衡理论	5
2.2	Actor-Critic 架构	6
2.3	近端策略优化	8
2.4	优先级经验回放	9
2.5	多体强化理论	10
2.6	本章小结	10
3	多智能体博弈主要工作	11
3.1	总体思路	11
3.2	问题建模	11
3.3	博弈求解	12
3.4	算法设计	18
3.5	平台架构	26
3.6	本章小结	27
4	实验与分析	28
4.1	实验目标	28
4.2	实验方法	28
4.3	结果分析	29
4.4	本章小结	34
5	结论	35
6	参考文献	36

附录 .....	39
作者简历 .....	41
本科生毕业论文（设计）任务书 .....	43
本科生毕业论文（设计）考核 .....	45

## 第二部分 毕业论文开题报告

一、 文献综述 .....	1
1 背景介绍 .....	1
1.1 研究意义 .....	1
1.2 研究背景和发展脉络 .....	2
2 国内外研究现状 .....	4
2.1 研究方向及进展 .....	4
2.2 存在问题 .....	7
3 研究展望 .....	8
4 参考文献 .....	9
二、 开题报告 .....	13
1 问题提出的背景 .....	13
1.1 课题背景 .....	13
1.2 同类研究现状 .....	14
1.3 项目来源及概括 .....	16
2 本研究的意义和目的 .....	17
2.1 研究意义 .....	17
2.2 研究目标 .....	17
3 项目的主要内容和技术路线 .....	17
3.1 主要内容 .....	17
3.2 技术路线 .....	18
3.3 可行性分析 .....	24
4 研究计划进度安排及预期目标 .....	29

---

4.1 进度安排.....	29
4.2 预期目标.....	29
5 参考文献 .....	31
 三、 外文翻译 .....	35
1 引言 .....	35
2 相关工作 .....	37
3 研究背景 .....	38
3.1 部分可见的马尔可夫游戏.....	38
3.2 确定性策略梯度算法 .....	39
3.3 多智能体深度确定性策略梯度 .....	39
3.4 内部驱动的强化学习和分层 DQN.....	40
4 基于通讯媒介的 MADDPG .....	40
4.1 问题推导和提出的方法 .....	40
4.2 学习算法.....	41
5 实验 .....	43
5.1 环境 .....	43
5.2 与基线算法的比较 .....	44
6 结论 .....	45
7 参考文献 .....	46
 四、 外文原文 .....	49
毕业论文（设计）文献综述和开题报告考核 .....	69



# **第一部分**

## **毕业论文**



# 1 絮论

## 1.1 课题背景

智能体的构建和训练是人工智能的重要分支，主要采用的策略是通过强化学习的方式，使得智能体通过和环境的交互不断优化自身的策略以期望得到更大的奖励。这一思想应用在各个学科，包括统计学、运筹学、博弈论和计算机科学等领域<sup>[1]</sup>。而在训练智能体的过程中，涉及到众多博弈的思想，使得强化学习的目的是为了更好地解释有限状态和条件下的一些平衡问题，获得当前状态和环境条件下的最优策略。

主流的观念认为强化学习主要应用的领域在游戏人工智能上，受 AlphaGo<sup>[2]</sup>影响，众多科研工作围绕构建基于强化学习的智能体算法，击败人类玩家或达到最佳的算法效果。但是其实游戏只是强化学习一个有效的载体，是研究智能体训练的落脚点，为智能体从虚拟环境到真实世界应用的迁移提供思路<sup>[3]</sup>。

从任务目标来看，强化学习分为训练智能体和环境之间交互的经典环境探索型任务，以及加入智能体间交互的多智能体任务。前者主要强调智能体对于环境的认知和探索，后者在此基础上加入了智能体间关系的构建。智能体之间包括合作型关系和对抗型关系，合作型智能体需要在环境中争取共同的奖励；而对抗型智能体需要在环境内互相制衡来最大化自身收益的同时，还要限制对方的收益<sup>[4]</sup>。因此，如何使得智能体训练过程中具备这些特质，智能体之间的交互方式是值得研究的课题。如果能找到一套通用的训练和管理智能体的方法，对智能体的训练、控制和执行都会有极大的优越性。

基于多智能体强化学习的研究，有利于促进对人类的思维、感知和探索等行为的理解，使得构建更加智能的应用。在室内服务、资源勘探、无人驾驶和竞技对抗等众多领域<sup>[1]</sup>，有多个智能体交互的应用场景，智能体可以通过训练来代替人类完成更加复杂的任务。强化学习也推动整个机器学习领域的发展，和强化学习密切相关的思想也应用在其他人工智能领域，例如计算机视觉和安全领域广泛应用的生成对抗网络 (Generative adversarial nets, GAN)。

## 1.2 研究现状

### 1.2.1 深度强化学习

深度学习自从 2010 年发展以来，带动了机器学习领域的众多发展，其中也包括强化学习。在传统强化学习中，状态和动作空间在离散非连续的情况下，如果维度不高尚且可以解决，但是高维连续的空间使用类似传统 Q-learning 的 Q 表来迭代是不可行的，因此深度学习将 Q 表的计算演化成了神经网络的拟合问题。谷歌在 2015 年提出了 DQN，利用 CNN(Convolutional Neural Network, 卷积神经网络) 和基于奖励的标签机制来解决奖励的稀疏性问题，通过借鉴经验池 (experience replay) 的方法来解决状态序列的相关性以及非静态分布问题<sup>[5]</sup>。DQN 的架构思路如图 1.1 所示，将传统的状态和动作转化为 Q 表的值转变为根据状态信息计算深度 Q 网络，随后有大量的工作在修改 DQN 的，包括 DDQN(双向 Q 网络)、DRQN(反向 Q 网络) 和 DQN 的集大成者 Rainbow<sup>[6]</sup> 等。

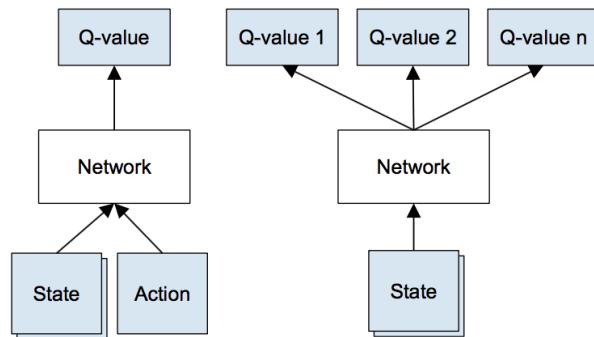


图 1.1: DQN 与传统 Q-learning 的架构区别<sup>[6]</sup>

### 1.2.2 基于策略梯度的算法

区别于 Q-learning 系列传统基于值 (Value-based) 的方法，还有一类基于策略梯度的算法，这和梯度下降等重要思想密切相关。其从传统强化学习中的 IGA 类算法发展到基于策略梯度的 Actor-Critic 机制<sup>[7]</sup>。算法分为两个部分，包括一个策略梯度函数构成的 Actor 模块，另一部分是一个价值网络构成的 Critic 模块，就像体操比赛中的裁判员给运动员的行为打分一样，通过价值网络的评估辅助更新策略梯度。Actor-Critic 架构已经发展成了强化学习的主流架构，OpenAI 相

继推出了 A2C(Advantage Actor-Critic)<sup>[8]</sup> 和 A3C(Asynchronous Advantage Actor-Critic)<sup>[8]</sup>，A3C 即为 A2C 的异步并行版本，在最新的论文中经常作为基线算法，性能和致胜程度在多个测试平台上都达到了最佳。基于 Actor-Critic 还有广泛应用的主流算法，包括 DDPG(Deep Deterministic Policy Gradient)<sup>[9]</sup> 和 PPO(Proximal Policy Optimization)<sup>[10]</sup>，其中 DDPG 和其多智能体版本是研究多智能体人机博弈的重要工具，而 PPO 则是旨在解决强化学习的有效性、简洁性和能耗问题，被当作一种其他强化学习的简便替代方案，并在复杂的游戏环境 dota2 中取得了成功<sup>[10]</sup>。为了解决星际争霸等复杂环境的强化学习问题，学界提出了分层强化学习的理论，取得了一定的进展，但是其巨大的开销和复杂度也产生了其特有的瓶颈。

### 1.2.3 多智能体强化学习

多智能体的研究很多还停留在世纪之交的一些成果，在如今深度强化学习中还未完全发掘，特别是涉及到的博弈思想。多智能体算法的基本假设是根据对手是否完全理性，基于对手理性的时候保证自己的最低收益，如 Littman 在 1994 年提出的 Minimax-Q<sup>[11]</sup>；基于贪心的最大化收益算法有 2001 年 Singh 提出的 IGA<sup>[12]</sup>，保证对手非理性时可以获取最大期望的收益。深度学习的发展促进了多智能体深度强化学习算法 (MADRL)<sup>[13]</sup>，有一系列针对性的学术成果。由于单智能体的策略在学习多智能体策略上直接移植会造成不适应的问题，比如环境和其他智能体状态的高度无序性，进而阻碍了经验回放的应用。在 2003 年曾经提出过集中训练，分散执行 (CTDE)<sup>[13]</sup> 的思想，用来解决多智能体系统中的无序问题，使得智能体更好地提升自身的观察和行为，更好估计训练时的策略和价值函数，这种思路也被用在了 MADDPG 的思想中<sup>[9]</sup>。

### 1.2.4 存在问题

(1) 强化学习的研究现状受益于深度学习的发展，随着算力和算法的革新取得了一定的进展，但是多数的研究还是依赖于神经网络的设计和大规模的搜索，这在 AlphaGo 系列中也得到了体现<sup>[2]</sup>。换言之，强化学习本身的性能和速度都有很大的提升空间，而不是仅仅依赖大量的算力。

(2) 从单智能体到多智能体算法的迁移中，存在大量无序信息和维度灾难的问题，在许多研究中选择性忽视了这个问题，但是这也对性能和致胜程度造成了一定的影响，这在未来也是必须要考虑的问题。

(3) 强化学习的算法和博弈论本身密切相关，结合博弈论的强化学习工作还缺乏突破性进展，而这一类型的进展也是人工智能会议最关注的问题之一。

### 1.3 本文贡献

本文从一个博弈可解的人机枪战攻防游戏出发，求解游戏的博弈纳什均衡解，改进 MADDPG 算法和 PPO 算法进行自我强化，检验算法的收敛速度和致胜程度，并应用在其他类似的多智能体环境中，具体贡献为：

- (1) 改进 OpenAI baseline PPO1 的实现，提升其性能
- (2) 设计了基于奖励和 TD-error 的经验回放机制，优化训练的收敛性，并为判断对手的理性程度提供依据。
- (3) 通过博弈的纳什均衡解检验算法的有效性，将博弈理论与强化学习相结合，应用在类似的多智能体人机博弈中。

### 1.4 本文结构

本文第二节着重介绍在进行多智能体研究中应用的经典理论和原理，第三节介绍求解问题的过程和算法设计，第四部分进行实验验证和分析结果，第五部分总结全文。

## 2 多智能体相关理论技术

为了选取与博弈思想更加契合的强化学习思路,本文选取了基于 Actor-Critic 架构的算法,这也是如今最主流和应用最广的架构<sup>[14]</sup>。基于 Actor-Critic 架构的算法有很多种,考虑可复现性和通用性,选取 OpenAI 提出的近端策优化 (PPO) 和多智能体深度确定性策略梯度 (MADDPG)。PPO 算法思路简洁,易于实现,在多种任务下可以快速收敛,但存在无法经验回放和合作对抗实现复杂的局限性,因此适合快速验证博弈问题的可解性,优化后可以作为一个非常有效的对比基线。而 MADDPG 算法的经验回放机制适应多体博弈环境下的要求,但是其原始的随机采样机制和部分改进仍然提升有限,因此本文核心工作是改进 MADDPG 算法。

### 2.1 纳什均衡理论

在战场环境下的每个智能体拥有其对应的目的和实现目的的方法,由此可以建模其行为空间。根据建模出的行为空间,计算每智能体个 action 映射到系统结果的函数,从而计算支付矩阵。支付矩阵 (payoff-matrix) 包含了备选行动方案、状态信息和效用值的矩阵。备选行动方案  $(a_1, a_2, \dots, a_n)$  是行动空间,行为之间两两独立。所有行为的集合  $A$  称为决策空间,状态  $(s_1, s_2, \dots, s_m)$  是环境带来的状态空间,往往是按照概率随机的,用概率  $(P_1, P_2, \dots, P_m)$  表示,所有状态的集合  $S$  称为状态空间,对于行动方案  $a_i$  在状态  $s_j$  下的收益值为  $(a_i, e_j)$ 。其集合  $Q = \{q_{11}, q_{12}, \dots, Q_{1m}, \dots, q_{n1}, q_{n2}, \dots, Q_{nm}\}$  的效用值域为  $U = \{u_{11}, \dots, u_{nm}\}$ ,其中  $u_{ij} = u(q_{ij})$ 。

在求解纳什均衡的时候,通常最优纯策略解不存在,此时的纳什均衡是一个最优混合策略解。假设  $A, B$  进行对抗,  $B$  以概率  $x_i (i = 1, 2, \dots, m)$  策略集  $S_2$  中选取纯策略  $b_i (i = 1, 2, \dots, m)$ ,则  $B$  的混合策略集为  $S_2^* = \{X = (x_1, x_2, \dots, x_m) | x_i \geq 0\}$ ,同理可得  $A$  的概率为  $y_j (j = 1, 2, \dots, n)$ ,对于  $\mathbf{X} \in S_2^*, \mathbf{Y} \in S_1^*$ ,  $B$  的收益函数为  $E(X, Y) = XAY^T = \sum_{i=1} \sum_j x_i x_i y_j$ , 乙方收益为  $E(X^*, Y^*) = X^* A(Y^*)^T$ ,纳什均衡可以通过迭代法求得。

## 2.2 Actor-Critic 架构

强化学习的思路主要有两类，一类是基于值 (Value-based) 的方法，另一类是基于策略 (Policy-Based) 的方法<sup>[15]</sup>，基于值的方法主要是 Q-learning 系列的算法<sup>[16]</sup>，基于策略的主要是策略梯度算法 (Policy Gradient)<sup>[9]</sup>。在本问题和类似的智能体学习和博弈问题中，行为动作空间的扩大会让 Q 表无限增长，难以表示和计算；而策略梯度本身是解决连续性动作问题的，在单步情况下会出现困难。因此如果把行为决策通过策略梯度函数来进行，而其每一步产生的价值通过 value-based 的函数来判断并进行单步更新，既解决了 Policy Gradient 的单步更新，又使得价值函数不会受到动作空间的限制。两种算法思路的相互补充构成了如今最成熟的强化学习架构，即 Actor-Critic<sup>[14]</sup>。

Actor-Critic 如图 2.1 所示，Actor 模块主要维护的是 Policy 部分的策略函数，Critic 维护的是 Value 部分的价值函数，智能体在做出行为后，系统反馈给 Critic 模块此时的奖励 Reward。获得奖励后，与 Value 函数计算的均方误差构成 TD-error，因此 Critic 模块以最小化 TD-error 为目标优化，而 Actor 的策略函数更新自身的分布来获取更高的奖励。整个过程和运动比赛极其相似，像体操运动员的表演和裁判员的打分以后，两者相辅相成互相优化，最终使得智能体具备获取更高 Reward 的能力。

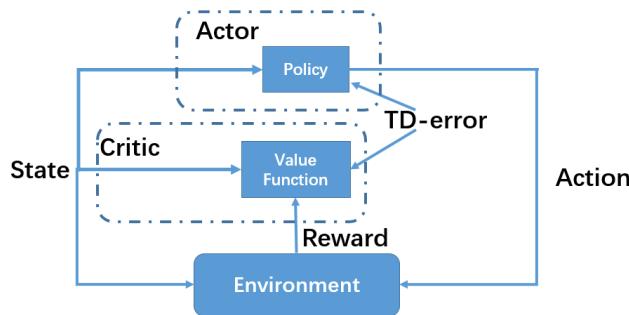


图 2.1: Actor-Critic 框架工作示意图

Actor-Critic 架构的算法中，除了解决复杂问题用的分层强化学习，在本问题的规模和算力的有限情况下最有效的算法是多智能体确定性策略梯度算法 (MADDPG)<sup>[9]</sup> 和近端策略优化 (PPO)<sup>[10]</sup>。MADDPG 算法旨在通过集中训练和分布执行的离线策略思想，既能处理好离散空间，也能处理好连续空间的问题；而

PPO 算法采用重要性采样和阈值化截断的 on-policy 策略思想，能解决策略梯度类算法难以控制更新步长的问题。

区别于传统 Q-learning 基于值函数的思路，策略梯度思路在 1999 年被 Sutton 提出<sup>[17]</sup>。其目的是为了解决连续性控制问题，或 Action 空间巨大的问题，通过一个概率密度函数  $\pi_\theta(s_t|\theta^\pi)$  表示每一步预测的最优策略，采样到的行为  $a_t \sim \pi_\theta(s_t|\theta^\pi)$ 。由于表示的是一些行为的分布，所以还是随机采样的结果。而且每次都要对梯度在整个 Action 空间进行积分会消耗大量的算力。为了解决这个问题，Silver 等人在 2014 年提出了确定性策略梯度<sup>[18]</sup>，通过一个函数  $\mu$  直接计算  $a_t = \mu(s_t|\theta^\mu)$ ，并证明了其确定性解的存在。Deepmind 在 2016 年提出通过 CNN(卷积神经网络) 计算  $\mu$  和  $Q$  函数，利用深度学习来解决策略梯度问题，这也是 AlphaGo 应用的思路和技术<sup>[2]</sup>。

DDPG(深度确定性策略梯度算法) 是一个 off-policy 的算法，如图 2.2 所示，具有一个采样的网络和一个更新的目标网络，为了保证采样过程中的有效经验不丢失，采用了经验回放机制 (Experience Replay)，将  $TD - error$  过大的状态转移元组存到缓存中，并按照一定的频率根据训练的批大小 minibatch 抽取输入到 Actor 网络和 Critic 网络。这样做可以破坏前后状态的连续性，并充分利用经验的内容，加快网络的收敛速度和效果。

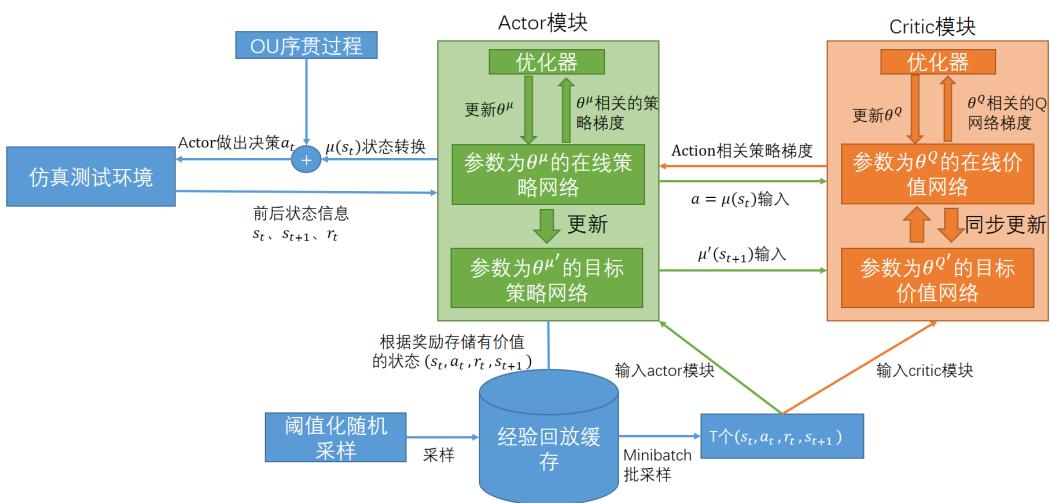


图 2.2: DDPG 的 off-policy 架构示意图

## 2.3 近端策略优化

近端策略优化 (PPO) 是 OpenAI 在 2017 年提出的算法<sup>[10]</sup>, 其根据前身 TRPO(置信域策略优化)<sup>[19]</sup>而来。PPO 算法是本文依赖的一个基线算法, 其思想是利用一阶近似来简化 TRPO 的步骤。传统的策略梯度算法中, 如果更新步长 (学习率  $\alpha$ ) 太小会使得智能体学习策略的速度很慢, 而如果太大则容易学到坏策略或局部最优策略<sup>[1]</sup>。在提升学习率  $\alpha$  的过程中, 一旦学习到坏策略, 会使得算法陷入恶性循环, 因此策略梯度的步长需要调试确定且工作量大。PPO 算法的前身 TRPO 引入了一个置信域的概念, 在较大步长的学习率下设定限制条件, 仅在新旧策略的差异的 KL 散度小于阈值的情况下最大化目标函数值。

但是 TRPO 的计算开销很大, PPO 在此基础上通过一阶梯度优化提高性能。如图图 2.3 所示, 通过旧策略  $\theta_{\text{old}}$  在当前状态  $s_t$  做出行为  $a_t$  的情况下, 更新策略的概率比为

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}.$$

为了使得  $r_t(\theta)$  在一个阈值  $\epsilon$  内, 对于使得  $r_t(\theta)$  不在  $[1 - \epsilon, 1 + \epsilon]$  范围内的更新进行截断。当该样本的优势估计值  $\hat{A}_t$  为正, 则该策略需要被鼓励, 通过最大化  $r_t(\theta)$  来最大化  $\pi_\theta(a_t|s_t)$ , 为了满足截断条件超过  $1 + \epsilon$  的部分直接取  $1 + \epsilon$ 。而当样本的优势估计值为负的时候, 则要压制这种策略, 即最小化  $r_t(\theta)$ , 为了满足截断条件少于  $1 - \epsilon$  的部分也会取  $1 - \epsilon$ 。总结来说, PPO 在优化的方向上会限制其过度的优化, 在反优化的方向上会尽量限制其变化, 保证算法整体上稳定性和策略的优劣性。

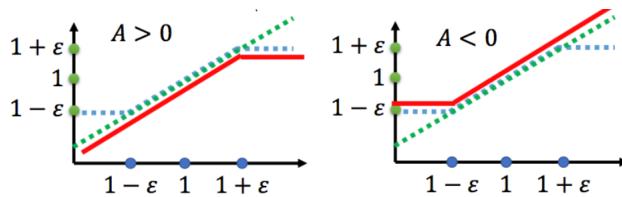


图 2.3: PPO 基于参数截断的实现<sup>[10]</sup>

PPO 算法的官方实现仅仅考虑了通用情况, 因此在本文具体应用在问题的时候需要在工程细节上改进, 为了提高其致胜程度和奖励值。PPO 算法基于 on-policy 的优化思想, 因此无法使用经验回放, 在面对合作对抗任务时设计复杂,

但是其对于竞争智能体的训练思路简洁，实现快速，可以用于验证博弈问题的可解性，并作为衡量改进 MADDPG 算法性能的一个基准。

## 2.4 优先级经验回放

经验回放模块是 DQN、DDPG 等算法优化的重要模块，其利用的是采样网络的训练来优化目标网络，破坏前后状态的相关性从而优化样本分布。而传统的 DDPG 算法直接按照批处理的大小直接从缓存中取数据，忽视了需要回放的状态本身，因此基于优先级的经验回放被提出，也是本文主要贡献的出发点。

在经验回放缓存中，往往存储着前后状态、行为、奖励、折扣系数等重要状态转移信息，这些信息本身并不等价，因此需要进行优先级排序。经验回放的目的之一是重复利用没有充分学习的策略，即 Critic 模块预测的价值和实际奖励偏差过大的，即  $TD - error$  的值可以用来作为优先级排序的依据。设计存储这样具备优先级的信息需要高效的数据结构，例如堆序的优先队列，可以用  $O(1)$  复杂度的弹出操作， $O(\log(k))$  的更新复杂度。在后续发布的其他版本的优先级经验回放论文中<sup>[20]</sup>，提出了一种对于抽样更加高效的经验回放机制，引入了如图 2.4 结构的存储结构。SumTree 本质上也是一个堆序的结构，父节点的值等于子结点的值的和。抽样的时候，把  $p$  的总和除以 batch 的大小，划分成 batch 个区间来进行随机采样，所得到的权值序列分布更加均匀，在提升优先级重要程度的同时保证了效率。

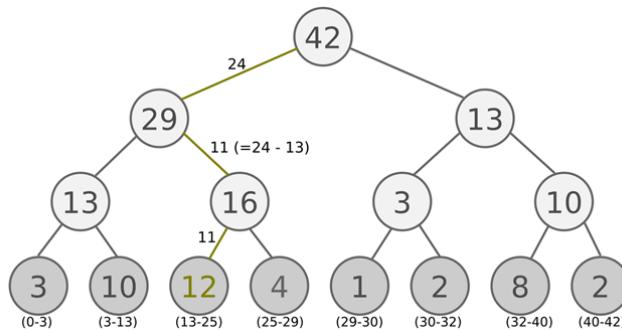


图 2.4: SumTree 结构示意图<sup>[20]</sup>

在经验回放上的相关工作主要围绕 Critic 模块的损失进行优先级排序，忽视了奖励本身代表的行为含义。换言之，其他对于经验回放的改进只强调了对于重

要状态的探索，没有区分重要状态对奖励和胜率的影响。此外，仅仅依靠状态的搜索来判断对手的理性程度是难以实现的。因此，本文旨在通过重要策略和状态的分类，使得 MADDPG 算法获得更快的收敛速度和更高的致胜程度，并提供判断对手理性程度的依据。

## 2.5 多体强化理论

多体强化理论涉及到智能体之间的交互，这也是单智能体的策略在多智能体上很难发挥作用的原因，因为环境和其他智能体的行为都会对决策产生影响。由于智能体在执行自身决策之前，不知道环境的全局信息，因此 DeepMind 提出了 MADDPG 算法<sup>[9]</sup>。其通过推测其他智能体的行为，引入一个  $\hat{\mu}_{\phi_i^j}$  代表与另一个智能体  $j$  的真实策略  $\mu_j$  有关的近似值，加入一个熵正则式来进行学习。对于  $K$  个智能体，其性能和鲁棒性都是非常优异的。

MADDPG 中对智能体强化协作的实现基于对其他智能体的预测，而本文对于经验回放的设计可以应用在所有智能体的经验共享上，使得 MADDPG 的实现思路更加简单，不区分合作和对抗性，以获得更快的收敛速度和相当的致胜程度。

## 2.6 本章小结

本章围绕多智能体相关的博弈理论和强化学习理论进行解释，尤其是和本文相关的工作如纳什均衡的得出，经验回放的相关工作和多体协作的理论，分析各个方法的优势和改进点，为后面章节的主要工作和实验提供依据。

### 3 多智能体博弈主要工作

#### 3.1 总体思路

本文的重点是优化主流的强化学习算法并分析其与博弈的关系，从单智能体的训练到多智能体的集成。如图 3.1 所示，本文采用从问题-模型-问题的思路，首先在多智能体人机博弈相关的问题中选取一个有代表性和实际意义的人机枪战博弈问题，通过博弈论原理和强化学习两种方式进行问题求解，再将算法应用在其他问题上。

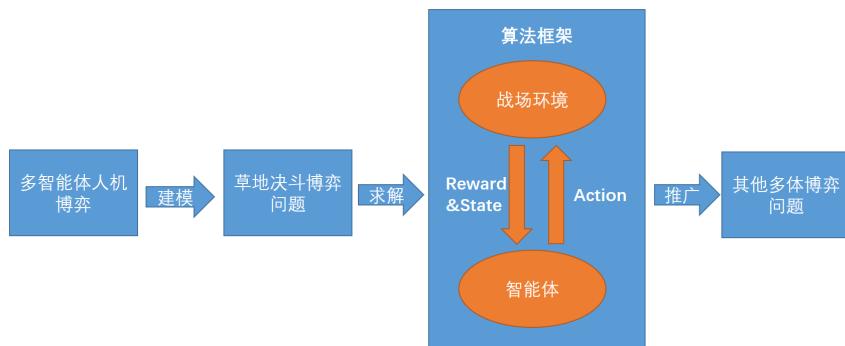


图 3.1: 工作思路示意图

#### 3.2 问题建模

在多种游戏战场设定中，二维方格战场是一种最常见的形式，大型游戏的建模也离不开二维坐标的位置信息。角色控制类游戏的智能体都具备两类及以上的行为，并在战场环境中通过决策执行行为，获得战场提供的奖励并最终获得输赢结果。

本文通过一个二维的草地战场，构建一个红蓝双方士兵的射击博弈。红蓝双方各有 2 个士兵，士兵装备了不同性能的武器，由于在草地环境下，双方匍匐前进互相无法发现，需要通过侦查通讯设备探测敌方。每个士兵配备有枪，侦察用的声波探测器和监听器，以及干扰敌方侦察用的声波干扰器。每方两个士兵同时侦测到敌方则可以准确定位，一个士兵无法准确定位敌方，所以率先击杀敌方的一方即获得游戏胜利。如图 3.2 是一个  $10 \times 10$  的草地战场红蓝双方对抗的示意

图。值得注意的是，两个士兵可以在同一个格子中，此时双方各类行为的效用率都是 1，即完全有效。在简化的版本中，我们如果只考虑射击，那么就假设双方都能发现敌方；如果我们考虑加上侦查的 1V1 环境，则每方只有一个士兵，根据侦察干扰设备的效用率来确定对对手的有效侦察效率。

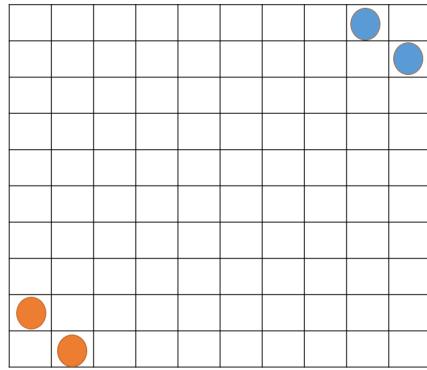


图 3.2: 10×10 的红蓝双方战场对抗示意图

### 3.3 博弈求解

首先，我们根据博弈论的原理计算对抗双方在不同战场设定情况下的支付矩阵，根据支付矩阵的值来求得纳什均衡解，再进行近似纳什均衡策略的设计。求解博弈关系，分成只涉及到射击行为，和涉及到侦察和射击两类行为的情况，分别求解支付矩阵和均衡解。

#### 3.3.1 智能体行为

本问题中双方智能体的行为集  $A$  包含  $A_1, A_2, A_3, A_4$  代表移动的集合，包括上下左右和原地不动 (STILL) 五种决策； $A_2$  代表射击的集合，可以选择开枪或不开； $A_3$  代表侦察和干扰的集合，声波探测器和干扰器都有开关的选项，因此有  $2^2 = 4$  种决策，若用二进制串表示侦察和干扰的决策，0 代表关，1 代表开，第一位代表探测器，第二位代表干扰器，则二进制串 01 代表的是关探测器和开干扰器。行为集  $A$  的具体内容如表 3.1 所示：

表 3.1: 智能体的行为集具体内容

行为类别	备择方案
$A_1$ (移动)	UP,DOWN,LEFT,RIGHT,STILL
$A_2$ (射击)	YES,NO
$A_3$ (侦察和干扰)	00, 01, 10, 11

### 3.3.2 射击博弈

首先考虑二维坐标下双方射击的博弈。双方武器性能不同，但是射击的距离变远都会降低命中率，双方位于同一格的极近情况下都是一击必中，而无穷远处可以认为命中率为 0。为了使得双方的武器命中率在距离不同时存在对抗关系，需要设计两个函数来映射距离和命中率的关系。

为了设计两个不同的概率函数，这里一种采用线性变化的函数，另一种选用神经网络中常见的激活函数 Sigmoid 函数的变种。Sigmoid 函数  $S(x) = \frac{1}{1+e^{-x}}$  如图 3.3 所示，在  $(-\infty, +\infty)$  上为连续光滑可导的函数，并且广泛应用在人工智能领域。<sup>[21]</sup>

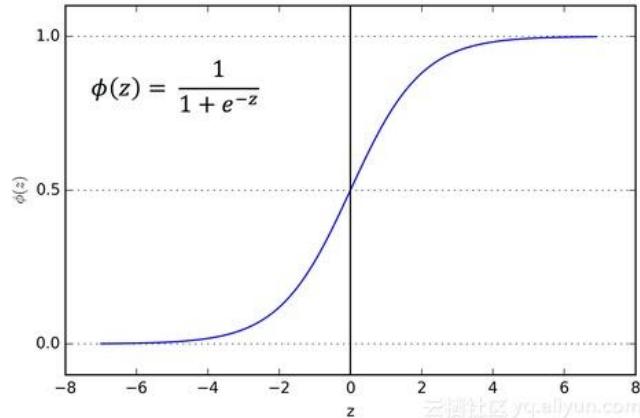


图 3.3: Sigmoid 函数的曲线图

Sigmoid 函数的定义域在  $(-\infty, +\infty)$ ，设最大距离为  $max$ ，只需要通过一个函数  $G(x)$  将  $(0, max)$  映射到  $(-\infty, +\infty)$  即可。因为当 Sigmoid 函数取 10 以上就能达到趋近 0 和 1，所以  $G(x)$  可表示为斜率是  $k$  的线性函数  $G(x) = k * (x - \frac{max}{2})$ ，

这样不会破坏 Sigmoid 函数的性质。

此时修改后的 Sigmoid 函数可以从  $(0, \max)$  直接映射到  $(1, 0)$ ，并且保留了 Sigmoid 函数的性质，通过参数  $\alpha$  和  $k$  可以微调函数的函数值，具体数学表达式 3-1 如下：

$$S(x) = \frac{1}{1 + \alpha * e^{-G(x)}} \quad (3-1)$$

假设战场是一个  $n \times n$  的二维方格世界，则两个智能体所在坐标的距离为两点的欧式距离  $D = \sqrt{x^2 + y^2}$  的值， $x$  和  $y$  分别是横纵方向的距离差，取值都是  $0, 1, \dots, n - 1$ ，则  $D$  的取值本来有  $\sum_{i=1}^n k = \frac{n(n+1)}{2}$ ，但是存在  $a^2 + b^2 = c^2 + d^2$  的情况（如  $3^2 + 4^2 = 5^2 + 0^2$ ），因此要对得到的距离进行去重，得到  $M$  种不同的距离代表不同的射击决策。在  $n = 10$  的时候，最远距离是  $9\sqrt{2}$ ，去掉重复的距离后有 51 种不同的距离，利用线性函数和不同参数的 Sigmoid 型函数计算命中率，此时如图 3.4 所示绘制了该情况下的三种武器命中率图。

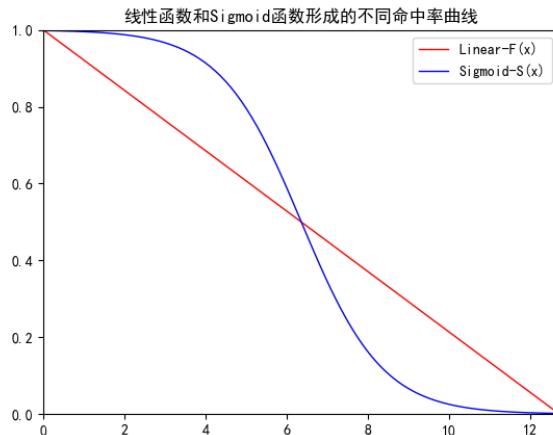


图 3.4:  $n=10$  情况下红蓝双方装备的两种武器命中率的函数图

已知共有  $M$  个不同的距离，将这些距离排序后可得降序序列

$$D_1, D_2, D_3, \dots, D_M$$

( $D_1$  代表最远的距离， $D_M$  一般为贴身的距离 0)

由于博弈双方都只有一颗子弹，如果对方开枪且未击中，那么我方只需要到最近的距离保证一枪必中即可。设定红方在  $D_i$  下进行射击的决策是 RSi，此时

命中率为  $R_i^S$ ; 蓝方在  $D_j$  下进行射击的决策为 BSj, 此时命中率为  $B_j^S$ , 则双方在射击博弈上的支付矩阵可以根据公式 3-2 计算, 对矩阵元进行分块, 分成对角线和上下三角分别处理。

$$P[i][j] = R_i^S - B_j^S \quad (3-2)$$

支付矩阵 Payoff 的对角线上元素表示双方在同样的距离直接决斗, 直接使用红方命中率减去蓝方命中率就是对应的支付值, 此时  $P[i][i] = R_i^S - B_i^S$ ; 支付矩阵下三角的元素表示红方选择在蓝方后面射击(因为 i 越大距离越近, 而  $i > j$ , 距离更近), 如果对方命中则阵亡, 如果对方不命中则可以等到近距离必中, 于是全部和蓝方的命中率相关,  $P[i][j] = 1 - 2 * B_j^S$ ; 同理支付矩阵的上三角元素表示红方选择在蓝方之前攻击, 如果对方命中则获胜, 否则自己会被对面一击必杀, 所以全部和红方命中率相关,  $P[i][j] = 2 * R_i^S - 1$ 。

在  $n=10$  的时候, 使用上图 3.4 中的命中率函数, 可以将矩阵计算出来, 是一个  $51*51$  的矩阵, 如表 3.2 所示的表为 payoff 的局部, 以  $P[2][3]$  为例, 红方第二远就选择开枪, 此时其命中率接近于 0, 而如果其不中则必死, 于是概率为  $1 - 2 * B_j^S$ 。

表 3.2:  $n=10$  时 payoff 矩阵的局部值

	BS1	BS2	BS3	...	BS49	BS50	BS51
RS1	0	-1	-1		-1	-1	-1
RS2	1	-0.0505	-0.9932		-0.9932	-0.9932	-0.9932
RS3	1	0.8922	-0.0977		-0.9871	-0.9871	-0.9871
...							
RS49	1	0.8922	0.7916		0.1041	0.9859	0.9859
RS50	1	0.8922	0.7916		-0.7778	0.0739	0.9907
RS51	1	0.8922	0.7916		-0.7778	-0.8429	0

### 3.3.3 偷察干扰博弈

在射击博弈的基础上, 我们考虑偷察博弈对射击定位的影响。根据战场设定, 双方只能知道对方的大致位置, 需要利用偷察设备的决策, 按照一定的概率

定位对手。假设双方的探测器 P，干扰器 J 和监听器 L 是否生效用三个二进制位表示，0 为不生效，1 为生效。当红方的探测器生效且蓝方干扰器无效，或红方监听器生效而且对方在探测或干扰的时候，红方可以侦察到蓝方的位置并射击。如表 3.3 所示，R000 代表红方的三个设备全部失效，B111 代表蓝方三个设备全部生效，以 (R101, B010) 为例，红方探测器和监听器生效，探测器被蓝方干扰，而蓝方干扰器触发监听器，因此红方最终发现蓝方，蓝方的逻辑表同理。

	$B_{000}$	$B_{001}$	$B_{010}$	$B_{011}$	$B_{100}$	$B_{101}$	$B_{110}$	$B_{111}$
$R_{000}$	FALSE							
$R_{001}$	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
$R_{010}$	FALSE							
$R_{011}$	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
$R_{100}$	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE
$R_{101}$	TRUE							
$R_{110}$	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE
$R_{111}$	TRUE							

表 3.3：双方侦察博弈过程中红方是否能侦察到蓝方的逻辑表

在逻辑关系基础上，我们加入三个设备的触发概率  $r$ ，假定三个设备的概率相同，都和对抗双方距离满足  $r(d) = \frac{1}{1+\beta*d}$ ，即效用率和双方的距离成反比， $\beta$  用于调节分布情况。用于监听器 L 是一直开着的，所以在决策的时候主要用到的是探测器 P 和干扰器 J 的开关，用两位二进制表示，0 为关，1 为开。双方可以根据探测器和监听器发现对方，探测器在对方干扰器无效的情况下起作用记为事件  $P$ ，监听器在敌方开了探测器或干扰器的情况下起作用，记为事件  $L$ 。由于事件  $P$  和事件  $L$  独立，则  $P \vee L = \overline{P} \wedge \overline{L}$ ，设双方的探测器、干扰器和监听器效用率分为  $p, j$  和  $l$ ，于是我们可以根据对抗关系列出如表 3.4 所示的概率统计表，其中每个大列对应的左边代表红方发现蓝方的概率，右边代表蓝方发现红方的概率。以  $(R11, B10)$  的蓝方为例，蓝方考虑探测器没有被红方干扰或者自己的监听器生效，则只要排除事件  $P_1$  和事件  $L_{11}$  即可。

	B00		B01		B10		B11	
R00	0	0	$l$	0	$l$	$p$	$\overline{L_{11}}$	$p$
R01	0	$l$	$l$	$l$	$l$	$\overline{P_1 L_{01}}$	$\overline{L_{11}}$	$\overline{P_1 L_{01}}$
R10	$p$	$l$	$\overline{P_1 L_{01}}$	$l$	$\overline{P_0 L_{10}}$	$\overline{P_0 L_{10}}$	$\overline{P_1 L_{11}}$	$\overline{P_0 L_{10}}$
R11	$p$	$\overline{L_{11}}$	$\overline{P_1 L_{11}}$	$\overline{L_{11}}$	$\overline{P_0 L_{10}}$	$\overline{P_1 L_{11}}$	$\overline{P_1 L_{11}}$	$\overline{P_1 L_{11}}$

对于事件 P, 敌方对干扰器的使用有 2 种情况

1. 记事件  $P_0$  为敌方不用干扰器

$$P_0 = 1 - p$$

2. 记事件  $P_1$  为敌方使用干扰器

$$P_1 = 1 - p * (1 - j)$$

对于事件 L, 敌方对探测器和干扰器的使用有 4 种情况

1. 记事件  $L_{00}$  为敌方两种设备都不使用

此时我方用监听器未能发现敌方的概率  $P(L_{00}) = 1$

2. 记事件  $L_{01}$  为敌方使用干扰器

此时我方用监听器未能发现敌方的概率为  $P(L_{01}) = 1 - l$

3. 记事件  $L_{10}$  为敌方使用探测器

此时我方用监听器未能发现敌方的概率为  $P(L_{10}) = 1 - l$

4. 记事件  $L_{11}$  为敌方两种设备都用

此时我方用监听器未能发现敌方的概率为  $P(L_{11}) = 1 - (1 - l)^2$

表 3.4: 双方在给定距离下侦察到敌方的概率统计表

求解完侦察与双方距离的关系, 考虑侦察对于射击结果的影响。在  $M$  个不同的距离上, 双方采用同一效用率的三类侦察设备, 则有  $M$  个  $4 \times 4$  的距离对应的侦察效用矩阵, 因此原先  $51 \times 51$  的矩阵博弈转变为  $204 \times 204$ , 但我们仍然可以把该矩阵分成  $51 \times 51$  个区域, 每个区域代表同样的射击决策, 区别是探测器和干扰器的使用情况。此时的射击效用率为有效侦察率和射击命中率的乘积。设红方在距离  $D_i$  下对探测器干扰器的使用为  $\{00RSi, 01RSi, 10RSi, 11RSi\}$ , 蓝方在距离  $D_j$  下对探测器干扰器的使用为  $\{00BSj, 01BSj, 10BSj, 11BSj\}$ , 红方在  $4 \times 4$  矩阵下发现蓝方的概率为  $R[x][y]_i$ , 蓝方在  $4 \times 4$  矩阵下发现红方的概率为

$B[x][y]_j$  ( $x, y$  代表侦察决策的二进制具体值)，则双方在侦查干扰影响下博弈的支付矩阵可以根据式 3-3 计算，具体对矩阵分块，判断双方的开枪顺序和侦察决策来具体计算矩阵值。

$$P[i][j][x][y] = R[x][y]_i * R_i^S - B[x][y]_j * B_j^S \quad (3-3)$$

支付矩阵 payoff 在  $i = j$ ，即对角线上的区域，此时是双方在同一条件下开枪，和双方的射击命中率和有效侦查率都相关，即在原先的命中率上乘上侦察的影响系数，此时  $P[i][i][x][y] = R[x][y]_i * R_i^S - B[x][y]_i * B_i^S$ ，和具体的侦察决策相关。支付矩阵下三角的元素表示红方选择在蓝方射击后再射击，此时只考虑蓝方的侦察率和射击命中率， $P[i][j][x][y] = 1 - 2 * B[x][y]_j * B_i^S$ ；同理，矩阵的上三角完全和红方的有效侦察率和射击命中率相关，此时  $P[i][j][x][y] = 2 * R[x][y]_i * R_i^S - 1$ 。通过这样求解，我们可以得到考虑射击和侦察干扰对射击影响的支付关系，进而求解其纳什均衡。

## 3.4 算法设计

### 3.4.1 数学模型

在解决这个问题，即用强化学习方法去训练有多种行为和策略集合的智能体的时候，实际上是求解马尔可夫决策过程 (Markov Decision Process) 中的状态转移函数<sup>[11]</sup>。而由于本问题中，无论是侦查和干扰，还是射击的过程，都有随机概率的因素，因此行为的结果、环境的认知和敌方的状态都是不确定的，属于部分可见的马尔可夫游戏 (partially observable Markov Games)<sup>[11]</sup>。

在一个有  $N$  个智能体的马尔可夫游戏中，其状态集合被定义为  $S$  来描述这  $N$  个智能体可能的状态配置信息，一个行为集  $\mathcal{A}_1, \dots, \mathcal{A}_N$  表示智能体的行为空间，一个观察集  $\mathcal{O}_1, \dots, \mathcal{O}_N$  表示智能体所能感受到的环境空间。为了根据智能体观察到的信息和自身的行为空间来选择自己的行为，每个智能体  $i$  使用一个随机策略  $\pi_{\theta_i} : \mathcal{O}_i \times \mathcal{A}_i \mapsto [0, 1]$ ，这个随机策略是根据状态转移函数  $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \dots \times \mathcal{A}_N \mapsto \mathcal{S}$  来产生下一个状态。每个智能体  $i$  根据自身状态和行为获取奖励的函数是  $r_i : \mathcal{S} \times \mathcal{A}_i \mapsto \mathbb{R}$ ，并且在当前状态下获得自身对环境的观察  $\mathbf{o}_i : \mathcal{S} \mapsto \mathcal{O}_i$ 。初始状态的分布  $\rho : \mathcal{S} \mapsto [0, 1]$  可以是一个完全随机的分布，也

可以加入经验信息或一个预训练的分布。每个智能体的目标就是在环境中获得最大的奖励， $R_i = \sum_{t=0}^T \gamma^t r_i^t$ 。其中  $\gamma$  是折扣因子， $T$  是总的游戏时间跨度，这个奖励在单智能体的时候属于智能体自身的奖励，但是多智能体的时候可以加上团队的奖励或智能体间的共同奖励。在本问题中，智能体的行为集  $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$  分别代表移动、射击和侦察干扰三类行为，观察集  $\mathcal{O}_1, \mathcal{O}_2$  分别代表自身侦察的结果和友方智能体观察的结果，以敌方智能体的列表信息呈现。

### 3.4.2 算法分析

建模在部分可见的马尔可夫游戏问题上，具体到解决枪战博弈问题的时候，对算法需要解决的问题有如下三点认识：

(1) 动作空间有一定规模

行为空间和观察空间组成的 state-action 决策空间虽然离散，但是在二维坐标情况下双方可能的坐标位置有 100 个，决策每一时刻有 8 种，整个 state-action 空间难以用离散化的表形式去表示，使用传统 Q-learning 的 Q 表开销很大并且效果不好，因此适合用深度 Q 网络去预测。

(2) 动作之间的分布差异大

在智能体进行移动的过程中，需要选取一个情况来射击，对于移动和射击两种行为本身来说分布非常不均匀，因此得到的奖励差别自然很大，但是偶然经验能否得到有效学习是一个非常值得关注的问题，也和本文对于经验的利用密切相关。另外，在 1V1 情况下，智能体  $i$  仅仅需要满足自身的最大奖励函数值  $r_i$  来学习自己的状态转移函数  $T$ ；而多智能体对抗的时候，还需要加上团队奖励或合作双方的共同奖励来学习。

(3) 环境存在噪声

本问题虽然看上去很像棋类游戏，可以用类似 AlphaGo 的搜索和模拟的情况去求解，但是奖励 Reward 的噪声很大，比如一方选择在很近的地方开枪，却碰巧没有命中，这在策略搜索的方法中对于更新叶子结点的权值是很有迷惑性的。或者说，采用类似蒙特卡洛树搜索 (MCTS) 或遗传算法等方式都需要足够多次数的模拟，这非常消耗算力和存储的信息。

回到方格世界的特征，动作空间虽然超出了传统 Q 表的范围，但是还在可编码范围内，因此可以用策略梯度的方法去求解。传统的策略梯度方法以及传统

的经验回放机制对于有噪声的环境和奖励会有糟糕的表现，因此本文的重点就是在策略梯度的基础上进行经验回放机制和学习机制的优化。

### 3.4.3 改进 PPO 算法

由于博弈求解的策略集不能直接用来训练，所以需要使用一个简洁的算法快速得到一个基线算法，而本文重点研究的 MADDPG 算法涉及的 off-policy 架构的复杂度更高，不适宜作为最初的基线，因此选择基于 on-policy 的策略 PPO<sup>[10]</sup>。PPO 也是基于 Actor-Critic 的架构，但是从论文中移植过来还要进行必要的改进和优化。仿照 off-policy 的架构，PPO 算法只需要训练同一个目标网络，所以不能使用经验回放，如图 3.5 所示，该框架为了简化经验操作，采用一个相近的策略  $\mu'$  来对于重要的经验进行直接采样会输入到 Critic 模块中，不会保存下来，在结构和性能上都易于实现。

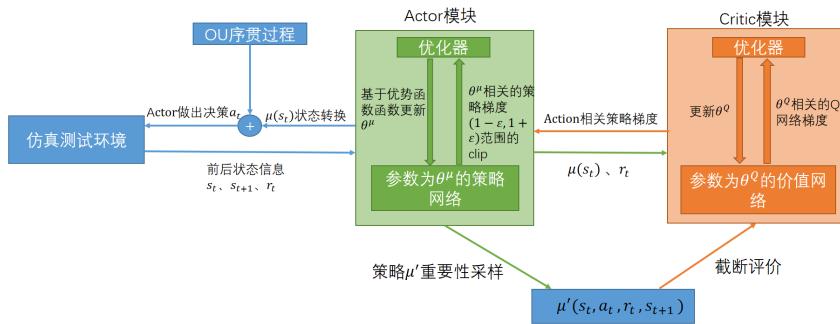


图 3.5: 改进 PPO 算法的工作流程图

针对本问题，对 PPO 做如下的改进：

#### (1) 网络正交初始化

本问题是一个决策序列，因此可能存在梯度爆炸或梯度消失的问题。在训练和执行的最初阶段，需要对算法参数进行初始化。如果仅仅用全 0 来初始化模型的参数  $w_i$ ，在反向传播时候的方差  $z = \sum_{i=1}^n w_i x_i$  都为 0，模型无法收敛。为了解决这个问题，Xavier 利用一个和模型参数规模  $n$  相关的均匀分布来对应  $k$  层模型<sup>[22]</sup>：

$$w \sim U \left[ -\frac{\sqrt{6}}{\sqrt{n_k + n_{k+1}}}, \frac{\sqrt{6}}{\sqrt{n_k + n_{k+1}}} \right].$$

考虑到本问题是智能体的决策问题，信息更加序列化，而 Xavier 初始化更适用 CNN 等网络，因此采用适用于序列信息的正交初始化。在我们的决策过程中，参数矩阵通过序列化的模型反复相乘，很容易造成经过大量计算后梯度爆炸或消失。正交初始化采用一个特征值绝对值为 1 的正交矩阵，那么矩阵的结果不会在数次矩阵运算后爆炸或消失。因此在初始化参数矩阵的时候，选用奇异值分解 (SVD) 的方式，对于随机生成的矩阵  $A \in R^{n \times n}$ ，分解形式为  $A = U\Sigma V^T$ ，其中  $U$ 、 $V$  分别是两组正交基， $\Sigma$  是按照奇异值缩放倍数的变换矩阵，则  $U$ 、 $V$  两者都可以作为初始化的正交参数矩阵，根据实际需要的矩阵维度决定。

采用最简单的三层 MLP 形式初始化参数矩阵，如图 3.6 所示，网络三层分别用 SVD 分解出的正交矩阵进行初始化，使得训练过程更加稳定，在这种长决策序列问题中不出现梯度爆炸和消失。

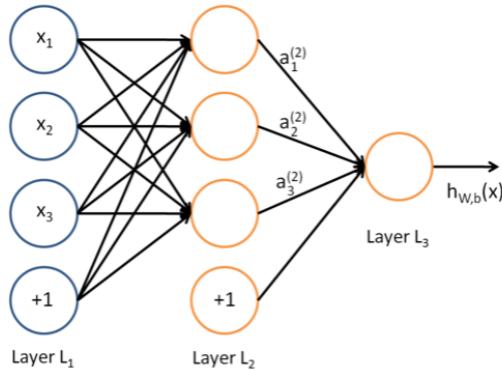


图 3.6: Actor 和 Critic 模块的简单三层 MLP 网络

## (2) OU 序贯过程

根据序贯相关的模型，引入序贯过程的随机噪声，其数学模型满足如下随机微分方程：

$$dx_t = \theta(\mu - x_t) dt + \sigma dW_t$$

其中  $\theta > 0, \mu, \sigma > 0$  均为参数， $W_t$  为布朗运动相关的维纳过程。这是在 DDPG 类算法中常用的一个过程，其长期的演化会收敛到其均值，因此对于训练有数据增强的作用，尤其是在噪声的环境下。

## (3) 学习率和 clip 参数退火

PPO 的核心操作是设定策略梯度更新的步长在  $[1 - \epsilon, 1 + \epsilon]$ ，使得保证较大的更新步长且不更新过度，但是随着训练过程的推进，网络逐渐收敛的过程中需要不断降低学习率，与此同时 clip 的参数  $\epsilon$  也要降低，使得更新不断减少。在学习率退火方面，Adam(Adaptive Moment Estimation) 是一种性能出色有助于快速收敛的优化器<sup>[5]</sup>，尤其是在这类复杂的任务中。其数学模型  $\Delta\theta_t = -\frac{\hat{m}_t}{\sqrt{\hat{n}_t + \epsilon}} * \eta$  使用了对梯度的一阶和二阶的矩估计，对内存需求小。虽然 PPO 的核心思路和 SGD 类似，但是为了快速获取一个基线算法，Adam 是最佳的选择。对于学习率和控制策略梯度更新的参数  $\epsilon$ ，同时使用 Adam 退火的收敛效果会更好。

### 3.4.4 经验回放机制

在 off-policy 和 on-policy 的策略中，都涉及到过去经验的使用，其中 off-policy 的策略是采用经验回放的缓存防止训练停止后经验丢失，而重要性采样是针对同一个网络对重要的样本和经验进行集中训练，这样可以减少前后状态的时序相关性并增加经验的利用。在这部分的设计中，和传统方法进行固定大小 buffer 并随机采样的策略不同，本文按照标准对经验进行划分，并以此判断对手的理性程度以快速适应对方策略。基于单个状态转移的奖励具体值，可以划分为收益策略和损失策略，在将经验输入到经验回放缓存的过程中按照收益大小保持经验缓存的降序。

在枪战游戏中，开枪是极其重要的操作，这类行为状态会造成极大的单回合奖励，即使随着时间的衰减也依然能发挥很大的作用。在本决斗游戏中，这类瞬间往往决定了整局游戏的胜负和走势，己方和敌方的奖励有巨大的差距，这类经验最需要被回放进行重点训练，另外获得奖励更少的一方在这一瞬间的决策可以认为是造成了损失，所以称为损失策略；而获得高额奖励的一方的决策成为收益策略。在面对非理性选手的时候，这类高额奖励的最大化策略更加有利于获取更高的利益，而面对纯理性选手，如纳什均衡的时候，损失策略的回放可以抑制这类经验的再现，保证充分的稳定性从而不重蹈覆辙。

对于某一回合下，对抗双方智能体  $a_m$  和  $a_n$ ，在某一时刻  $t$ ，获得的奖励  $r_m - r_n > \epsilon_2$ ，奖励差异过大，则认为是重要的经验，其中  $a_m$  的行为带来了收益， $a_n$  的行为带来了损失，所以两个智能体都将  $(s_t^m, a_t^m, r_t^m, s_{t+1}^m)$  存入到收益经验池， $(s_t^n, a_t^n, r_t^n, s_{t+1}^n)$  存入损失经验池。收益经验池考虑回放的因素有两个，

一个是 Critic 模块判断的  $loss$ , 即  $TD - error$  的值, 以及该记录带来的奖励值  $r$ , 所以收益池的优先级计算通过基于  $\text{Priority} = \alpha * TD - error + r$  的堆序优先队列。在实践中发现损失策略的奖励值分布都很小, 经常是很多简单的移动操作, 所以采用哈希表实现, 通过  $key - value$  式样的存储统计这类状态对应的平均奖励, 计算公式为  $avg = \frac{\sum_{i=1}^k r_n}{k}$ 。如图 3.7 所示, 收益池和损失池分别用二叉堆和哈希表实现, 在默认情况下保证双方各一半的回放概率。但是在损失池中命中率很低, 则对手较为理性, 则需要充分回放损失池的内容防止失误; 而如果损失池命中率很高, 则对手有非理性因素, 则需要充分回放收益池最大化自己的收益。设经验回放的 minibatch 大小为  $K$ , 则收益池和损失池各回放  $\alpha K$  和  $\beta K$ , 保证  $\alpha + \beta = 1$ , 并根据对对手行为的损失查找来动态调整。

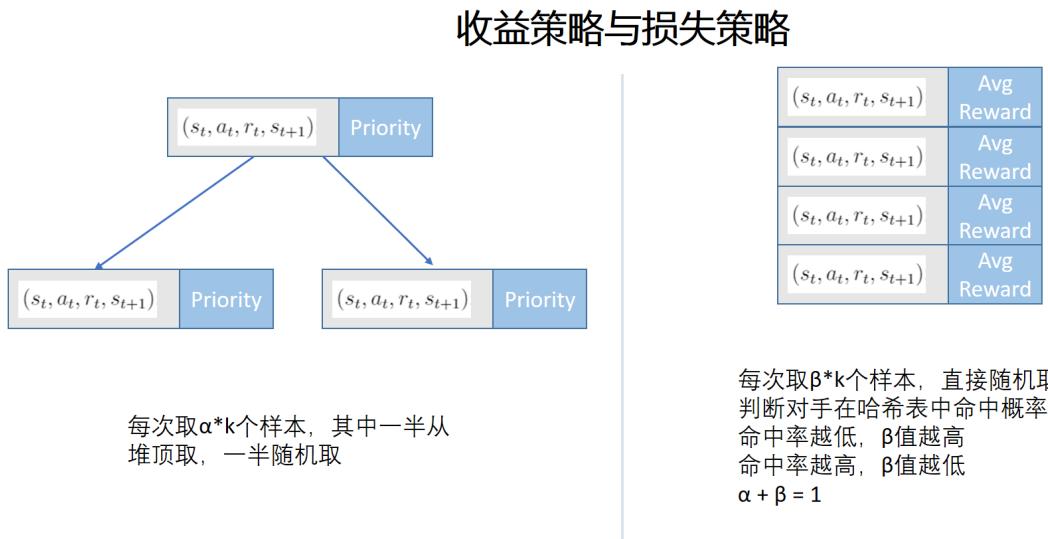


图 3.7: 收益策略和损失策略的说明图

经验回放模块的主要操作就是包括初始化、插入、查找和采样, 如算法 1 所示。其中收益策略池为二叉堆, 采样  $k$  个的复杂度为  $O(k)$ , 调整的复杂度为  $O(k \log(h))$ 。为了保证采样的分布, 不能仅仅从堆顶取, 而是从堆顶弹出一半, 剩下的通过二叉堆结构在两边随机采样, 保证适度学习重要经验。而哈希表的性能更好, 但是对编码空间有一定的要求, 对于本问题的坐标信息、距离信息等状态, 仍然能保证较高的效率。对于损失池的采样, 直接进行随机采样, 不会造成优先级的困扰。在后续的实验中, 对于该哈希结构面对随机选手和理性选手需要做命中测试, 以进一步确认该结构在判断对手理性程度的有效性。

**Algorithm 1** 基于  $TD - error$  和奖励的经验回放**Require:**  $(s_t, a_t, r_t, s_{t+1})$ : 状态转移元组**Ensure:** 主要操作

- 1: 初始化: 新状态设  $TD - error$  为默认最大, 保证每个经验回放一次
- 2: 插入: 收益策略根据  $Priority = \alpha * TD - error + r$  插入优先队列, 损失策略哈希表插入
- 3: 判断: 查找损失策略哈希表, 低命中率则为理性对手, 高命中率则为非理性对手。
- 4: 采样: 两个  $Buffer$  通常各取一半, 判断对手为理性则增加损失  $Buffer$  的权重, 对手为非理性则增加收益  $Buffer$  的权重

### 3.4.5 改进 MADDPG 算法

DDPG 类算法对于经验回放的应用是极其重要的一环, 建立在改进过的经验缓存后, 其他内容不需要改变, 只需要把对于经验回放的采样策略进行修改。结合刚才解释的机制, 现在改进 MADDPG 算法的流程如图 3.8, 主要的改进就在经验回放的设计上。

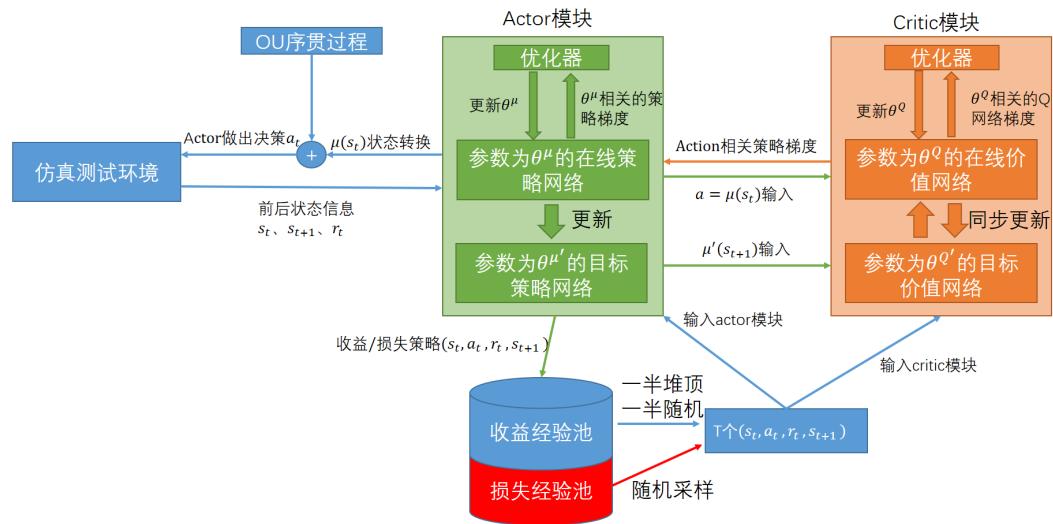


图 3.8: 改进 MADDPG 算法的流程框架

在多智能体的情况下, 对抗的不仅仅是双方, 可能有  $N$  个智能体, 因此对

于每组有进行交互的智能体做以上的经验回放操作，其伪代码如2所示。其复杂度随着智能体数目的增加会指数级增长，但是在讨论有限个多智能体的时候，仍然是一种很有效率的算法。

---

**Algorithm 2** 使用奖励经验回放训练 MADDPG 算法
 

---

```

1: for episode = 1 to M do
2:   初始化一个智能体行为的随机过程  $\mathcal{N}$ ;
3:   从环境 Env 获得初始状态;
4:   for t = 1 to MaxEpisodeTime do
5:     对于每个智能体  $i$ , 决策行为  $a_i = \mu_{\theta_i}(o_i) + \mathcal{N}_t$ ;
6:     执行  $N$  个智能体的行为序列  $a = (a_1, \dots, a_N)$  并获取奖励  $r$  和全局状
   态  $s'$ ;
7:     Critic 模块计算  $TD - error$ , 根据阈值  $\epsilon_1$  判断是否需要存入 Buffer
8:     对于任意两对交互的智能体  $a_m$  和  $a_n$  且  $r_m - r_n > \epsilon_2$ 
9:     存储  $(s_t^m, a_t^m, r_t^m, s_{t+1}^m)$  到收益经验池,  $(s_t^n, a_t^n, r_t^n, s_{t+1}^n)$  到损失经验池
10:    for agent = 1 to M do
11:      从两个经验池分别随机采样  $S/2$  个样本,  $(\mathbf{s}^j, a^j, r^j, \mathbf{s}'^j)$  输入到
       Actor 和 Critic 模块
12:      最小化 Critic 网络的损失  $\mathcal{L}(\theta_i) = \frac{1}{S} \sum_j Mse$ 
13:      采样的策略梯度更新 Actor 模块
14:    end for
15:    为每个智能体  $i$  更新目标网络,  $\theta'_i \leftarrow \tau \theta_i + (1 - \tau) \theta'_i$ 
16:  end for
17: end for
  
```

---

### 3.4.6 多体协作

在多体协作方面，没有采用 MADDPG 的思路，因为要充分利用设计的经验回放池。同对抗智能体类似的情况是，同队的智能体之间可以共享高奖励值的经验，因此同队的智能体之间共享收益策略和损失策略。所以这和2中描述的一样，对于所有有交互的智能体，包括敌对的智能体和所有友军智能体，都进行一

次更新。这个实现相比于 MADDPG 中对于其他智能体行为的猜测，原理上更加简单，充分利用了设计的经验回放机制，因此可以用同一组网络来训练所有的合作型和对抗型智能体，符合集中训练分布执行的 CTDE 思想。

### 3.4.7 奖励设计

奖励值是环境对于智能体状态和行为的反馈，目的就是对于奖励的预测和最大化。奖励机制的设计尤为重要，一些看似合理的奖励设计实际上给算法带来了收敛困难和策略单一的局限性。例如，在 NIPS2018 的多智能体论文中讲述的多体推大球问题，通过给予正向推球高额的奖励可以快速收敛，但是为了抑制反向推的操作加上一个  $1/10$  倍正向奖励的惩罚，算法就出现了难以收敛并陷入局部最优<sup>[10]</sup>。因此，本问题设计奖励机制的时候要兼顾算法的收敛性和多样性。

参考了众多论文中的奖励函数设计，采取了鼓励式的奖励方式，只对某些影响战局的失误给予小惩罚，使得最终的总奖励不会为负，这样的设计对于收敛会更加迅速，和对于人类的激励促进作用是同理的。具体如表 3.5 的设计，在网络计算的过程中，还做了归一化操作，防止出现奖励分布过于不均匀。

有子弹存活一轮	+0.01	侦察到敌方	+1
无子弹存活一轮	+0.05	被侦察到	-1
攻击并击杀	+5	共同侦察到	+3
攻击并未击杀	+1	队友侦察到自己未侦察	-0.5
被击杀	-1		

表 3.5: 奖励机制的设计

## 3.5 平台架构

在多智能体强化学习的领域的诸多基础算法和标准都在著名的平台 gym 上进行了测试，并且 OpenAI 实现了主流算法的 baseline 版本托管在 github 上并接受来自全世界的挑战和比较。为了使得应用在本文题上的算法可以和其他 baseline 有对比的空间，遵照 gym 平台环境和智能体的交互方式设计，尤其是接口上。实现 gym 类似的环境-行为接口，可以直接适应 baseline 并做出对比，不需要重写

其他对比的算法，而且自己实现的算法也能直接在 gym 上测试。

gym 平台的交互方式通过传递 action 信息，调用 env 接口就可以获得一个四元组信息，包括训练和执行最重要的 observation,reward,done 和 info，observation 进一步转化为智能体的 state 信息，reward 代表环境对 action 反馈的奖励，done 代表游戏是否终结，info 代表环境反馈的重要 log 信息。通过搭建命令接口可以测试指定轮数的信息，通过图形界面便于观察智能体特定的行为是否学习到，如图 3.9 是搭建的可视化界面示意图，左边部分是智能体对抗的界面，右边是参数面板。

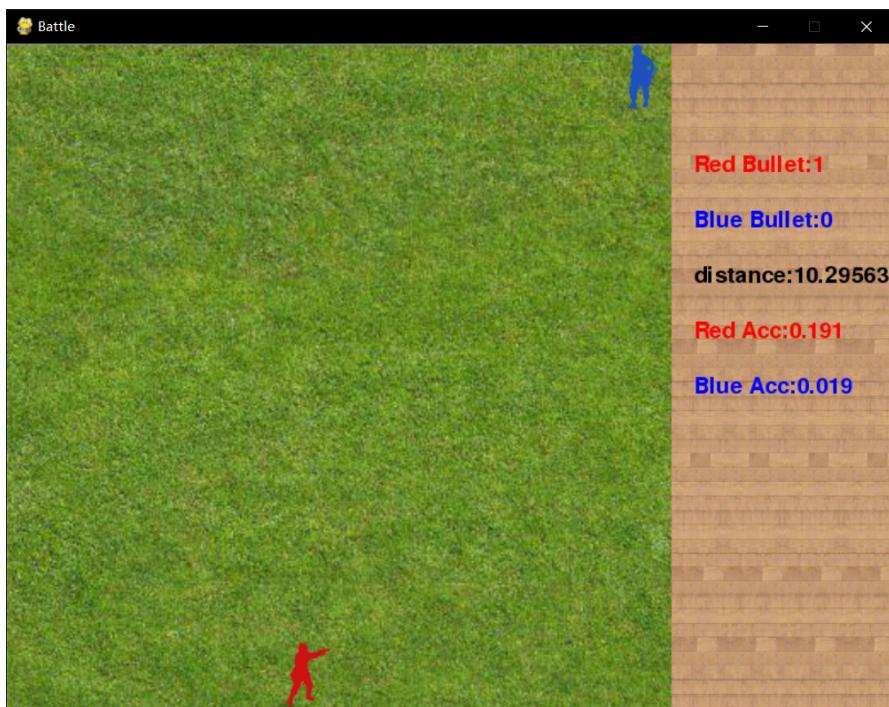


图 3.9：图形版调试工具界面

### 3.6 本章小结

本章围绕多智能体博弈的主要工作，根据博弈的求解和算法的设计，具体阐述了从问题到建模再到问题的工作思路，对于数学推导和两种算法的细节都逐条进行了分析，对比了相关工作的思路，并提出了实验的平台、方法和对比对象等内容。

## 4 实验与分析

本节根据算法改进的目的和贡献点的要求设计测试样例，从不同角度对比算法的有效程度，按照实验目标、实验方法和结果分析的顺序推进。

检验算法有效性的指标包括收敛速度、奖励值和致胜程度。在结果表示上，所有数据表中的收敛速度都是达到收敛时训练的 `eposide` 周期数，所有的统计量都以收敛之后的数据为准。实验环境的具体说明如表 4.1 所示。

硬件参数	服务器:AWS EC2 c5 内存:8GB CPU:3.0 GHz Intel Xeon Platinum
软件环境	操作系统:CentOS7+Docker 语言环境:Python3.6.4+Pytorch0.4.0
测试参数	随机种子:random.seed(10) 训练参数:2000 局/eposide

表 4.1: 实验环境的具体说明

### 4.1 实验目标

- (1) 检验改进 PPO 算法在收敛速度、奖励值和致胜程度上是否对 PPO1 baseline 有提升。
- (2) 检验改进 MADDPG 算法相比改进 PPO 和原始 MADDPG 算法在收敛速度、奖励值和致胜程度上的提升。
- (3) 检验改进 PPO 和改进 MADDPG 对抗近似纳什均衡的结果。
- (4) 检验改进 MADDPG 算法在环境中理性行为的学习程度。
- (5) 检验改进经验回放的性能。

### 4.2 实验方法

- (1) 在训练过程中，记录训练时模型的 Reward 值绘制收敛曲线比较收敛速度，保存模型后进行双方公平换边的对战记录胜平负的场次，对胜率进行显著性 t 检验。

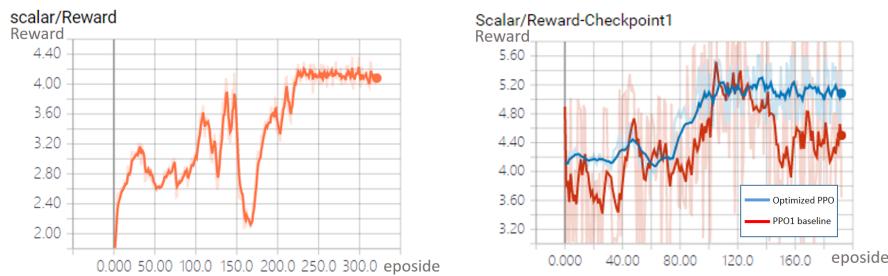
- (2) 单体强化通过智能体的行为日志判断智能体是否具备追杀和逃脱的能力，在人机枪战决斗中，判断在对方无子弹时能否追杀，在自己无子弹时能否逃脱。
- (3) 检查损失池的命中率来检测改进经验回放的性能。

## 4.3 结果分析

### 4.3.1 单体强化

#### (1) 改进 PPO 的验证

在未知纳什均衡的前提下，需要为算法寻找一个强力的基线算法作为训练对象和对比对象。因此，如图(a)所示，首先利用改进 PPO 算法进行自我博弈至第一次收敛，保存此时收敛的模型。为了避免自我博弈陷入的局部最优，因此使得一方的模型固定不再学习，将改进 PPO 算法和 PPO1 baseline 算法分别和第一阶段收敛的模型进行对抗，如图(b)所示，使得 Reward 进一步上升并收敛。



(a) 第一阶段改进 PPO 自我博弈的 Reward 变化曲线 (b) 第二阶段改进 PPO 和 PPO1 baseline 和第一阶段的断点模型对抗的 Reward 变化曲线

图 4.1: 两阶段训练改进 PPO 的 Reward 变化曲线

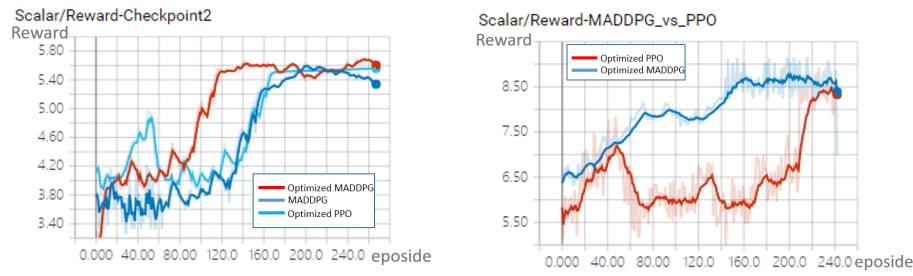
和第一阶段改进 PPO 自我博弈收敛的结果相比，用改进 PPO 和 PPO1 baseline 分别和断点 1 训练的过程中，如表 4.2 所示，奖励进一步优化到了 5.09 和 4.65，与第一阶段模型相比的胜率也分别达到了 66.2% 和 63.1%。在改进 PPO 和 PPO1 的收敛过程中，改进 PPO1 收敛速度更快，奖励方差更小，但是在对抗胜率上经过 t 检验和 PPO1 baseline 没有显著性的差别。

	收敛速度	奖励均值	奖励方差	与断点 1 对抗胜率均值	互相对抗胜率均值
断点 1		4.15	0.01	50.0%	
改进 PPO	110	5.09	0.01	66.2%	50.9%
PPO1 baseline	200	4.65	0.65	63.1%	49.1%

表 4.2: 改进 PPO1 和 PPO1 baseline 在对抗断点 1 模型时的结果统计表

## (2) 改进 MADDPG 的验证

验证完改进 PPO 的有效性后，验证改进 MADDPG 的表现情况，和 MADDPG 原始版和改进 PPO 作为对比。如图(a)是三个算法在射击环境下的奖励变化曲线，如图(b)是改进 MADDPG 和改进 PPO 算法在侦察-射击环境下的奖励变化曲线。



(a) 改进 MADDPG、MADDPG 和 PPO 在射击环境下自我博弈的 Reward 变化曲线  
(b) 改进 MADDPG 和 PPO 在射击和侦察环境下自我博弈的 Reward 变化曲线

图 4.2: MADDPG 和 PPO 在两个环境下的 Reward 变化曲线

如表 4.3 所示，改进 MADDPG 在奖励均值和奖励方差上虽然都有一定的优势，但是经过 t 检验后差异都不显著。而在收敛速度和稳定性上，改进 MADDPG 在两个任务上和其他基线相比都有明显的优势，可以更快更加稳定地学习到策略。

	收敛速度	奖励均值	奖励方差	与改进 PPO 胜率均值	t 检验
<b>射击环境</b>					
改进 MADDPG	<b>140</b>	5.56	0.01	51.9%	不显著
MADDPG	200	5.38	0.09	49.2%	不显著
改进 PPO	164	5.52	0.01	50.0%	不显著
<b>侦察射击环境</b>					
改进 MADDPG	<b>160</b>	8.59	0.01	51.5%	不显著
改进 PPO	220	8.48	0.02	50.0%	不显著

表 4.3: 在两个环境下 MADDPG 类算法和 PPO 算法的自我博弈结果表

### (3) 和近似纳什均衡对抗的验证

通过本文第三节工作中对于该博弈游戏纳什均衡的求解，求解出了基于距离的枪战决策支付矩阵。实际操作过程中，智能体基于坐标的移动是难以用纳什均衡求解的，因此近似的纳什均衡策略是让智能体进行随机移动，判定距离在纳什均衡解的时候进行射击。由于是二维坐标，并不是每个距离在每次移动模拟中可到达，因此该近似纳什均衡策略与真实的完全理性策略有一定的距离。而在使用强化学习训练和执行智能体的时候，智能体的决策基于移动和射击等，所以会学到一些移动的策略，相较于近似纳什均衡解客观上存在一定的优势。

使用基于距离的近似纳什均衡解验证改进 PPO 和改进 MADDPG 的有效性，通过和近似纳什均衡的 20 次 100 场对抗得到表 4.4 的结果图，虽然胜率均值超过了 50%，但是只有改进 MADDPG 的胜率经过 t 检验较为显著，对于近似的纳什均衡解有一定的胜率优势。

	胜率均值	t 检验结果
改进 MADDPG	<b>58.8%</b>	<b>0.01</b> , 显著
改进 PPO	52.2%	0.08, 不显著

表 4.4: 改进 MADDPG 和改进 PPO 在对抗近似纳什均衡的表现

### 4.3.2 多体强化

验证多智能体强化的时候，奖励分为单体奖励和团队奖励，这也和最后团队的胜利有一定的关联。通过在 2V2 枪战博弈上的测试，验证改进 MADDPG、MADDPG 和 DDPG 的奖励和胜率信息，具体如表表 4.5 所示，在单体奖励和 1V1 胜率上，改进的 MADDPG 有一定的优势，尤其是收敛速度较其他两种算法快很多。但是在涉及到合作的 2V2 环境的时候，MADDPG 原始基于对其他智能体预测行为的方式更加合理，在胜率和团队奖励上有明显优势。总体来说，改进的 MADDPG 对于单体和竞争型智能体的训练有一定的优势，但是在合作性上仍然有所欠缺。值得一提的是，改进的 MADDPG 在团队奖励的收敛上速度很快，但是收敛的奖励值明显低于 MADDPG，因此在对抗时候的胜率也落到下风。

奖励	单体奖励收敛	单体奖励均值	团队奖励收敛	团队奖励均值
改进 MADDPG	<b>286</b>	<b>11.15</b>	144	5.23
MADDPG	372	10.82	291	7.84
DDPG	500 未收敛	9.86	500 未收敛	3.23
对抗胜率	1V1 均值	1V1 t 检验	2V2 均值	2V2 t 检验
改进 MADDPG vs MADDPG	<b>52.5%</b>	不显著	44.4%	显著
改进 MADDPG vs DDPG	63.0%	显著	73.7%	显著
改进 MADDPG vs DDPG	61.6%	显著	76.9%	显著

表 4.5: 多体强化中改进 MADDPG、MADDPG 和 DDPG 在奖励和胜率上的结果表

### 4.3.3 性能测试

在改进 MADDPG 的实现中，损失经验池的设计影响到对方理性程度的认识，所以需要针对不同对手测试其哈希结果的性能。这里利用一个完全随机的选手和纳什均衡的选手进行对比实验，检验哈希表的命中率，结果如表 4.6 所示。完全随机的选手的决策落在损失经验池中概率明显更高，也获得了更多的奖励

损失，而理性对手几乎不会出现在损失池中。根据系统保存的行为序列信息，理性选手出现在损失池中的行为只是在某些状态下选择了停止被对手以低概率击毙，从而使得信息混入了损失池。

	损失池命中率	命中平均奖励
随机智能体	16.3%	-0.04
近似纳什均衡	0.4%	-0.01

表 4.6: 损失经验池的有效性测试结果

#### 4.3.4 行为测试

行为测试围绕某些情况下的特定策略，这是通过奖励机制使得智能体在环境中自发获取的一种策略分布，此检验对于智能体的行为评价多了一个维度。我们分为 1V1 枪战场景下的追杀和逃脱，和 2V2 足球游戏的合作来说明这一结果。1V1 枪战场景为自主编写的 GUI(用户图形界面)，2V2 足球场景是利用了 Unity 官方开发的智能体工具<sup>[23]</sup>。

在 1V1 对抗中，选用一个随机对手为红方，本文训练的模型为蓝方，如图(a)所示，蓝方在打完子弹后选择远离红方，进而拖延时间到最后平局。如图(b)所示，红方打完子弹后，蓝方一直追到红方眼前确保射击，体现了智能体追杀的策略实现。

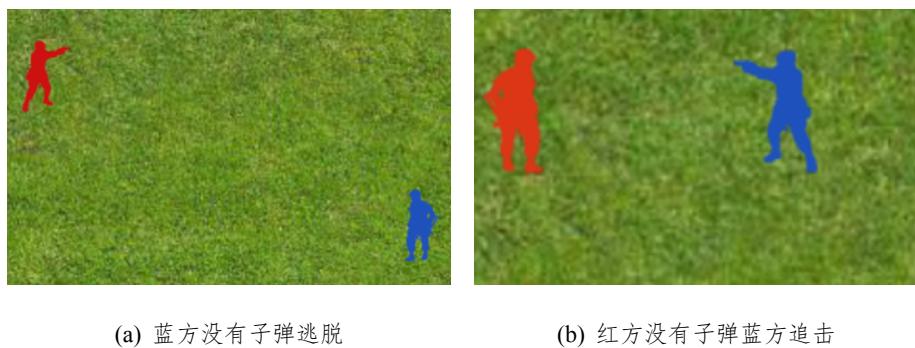
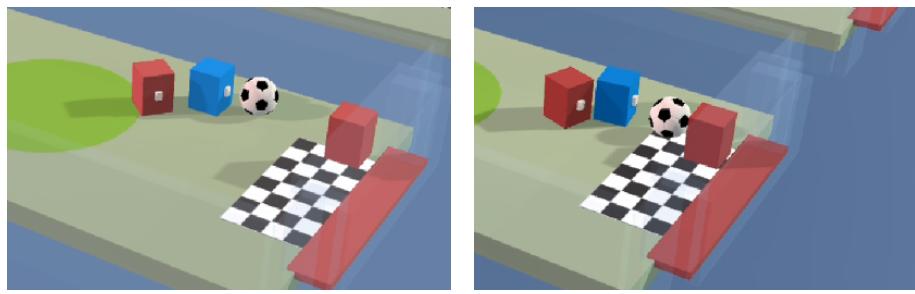


图 4.3: 1V1 枪战的策略可视化

在 2V2 对抗中，使用 2V2 的足球游戏进行验证。如图(a)所示，蓝方球员开

始是和一名球员对抗，在图(b)的情况下红方的守门员协助，共同堵截蓝方球员。通过算法的训练，使得双方智能体使用同一个模型达到了对手间对抗和队友间合作的效果。



(a) 蓝方球员带球进攻，红方球员纠缠

(b) 红方球员合作堵截蓝方球员

图 4.4: 2V2 足球游戏的合作对抗可视化

#### 4.4 本章小结

本章根据前文博弈问题的建模和强化学习算法的设计，具体设计实验测试用例和对比点，对实验数据进行定量展示和分析，为全文的结论提供数据的支撑。

## 5 结论

本文围绕多智能体人机博弈问题，从一个存在纳什均衡的枪战博弈出发，设计了两种算法进行智能体训练。改进 PPO 算法主要应用在 1V1 的枪战博弈问题，相较于 baseline 提升了性能；改进 MADDPG 算法同时能应用在 1V1 和 2V2 的博弈问题上，相较于原始 MADDPG 和改进 PPO 提升了竞争型智能体的性能。

改进 PPO 算法通过优化 OpenAI 的 PPO1-baseline<sup>[10]</sup> 的实现，验证了枪战博弈问题用强化学习的可解性，提升了奖励值和致胜程度（如表 4.2 所示），为验证本文改进 MADDPG 算法的核心工作提供一个对比的基线。

改进 MADDPG 算法是本文的工作核心，可用于竞争和合作对抗的多智能体博弈问题中。改进 MADDPG 算法通过修改 MADDPG 算法的经验回放机制，划分收益策略和损失策略，实现了对抗间的相互学习，并提供了判断对手理性程度的依据。在实验验证中，改进 MADDPG 算法相比于 MADDPG 和改进 PPO，拥有更快地收敛速度、奖励值和致胜程度（如表 4.3 所示），并取得了与近似纳什均衡相当的致胜程度（如表 4.4 所示）。在进一步拓展到 2V2 合作对抗中，改进 MADDPG 算法对竞争型智能体的训练获得了比 MADDPG 算法更好的效果，但是在合作型智能体上还有待加强（如表 4.5 所示）。

在本工作基础上，可以进行更多围绕博弈理论相关的工作，也可以在算法上加入时序相关或分布式的结构进行训练和预测，如果能在其他的环境上有很好的迁移能力，整个算法会有更好的通用性。

## 6 参考文献

- [1] KILINC O, MONTANA G. Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations[J]. 2018.
- [2] NOWÉ A, BRYS T. A Gentle Introduction to Reinforcement Learning[M]. [S.l.]: [s.n.], 2016.
- [3] K T, A N. Evolutionary game theory and multi-agent reinforcement learning[M]. [S.l.]: [s.n.], 2005.
- [4] KAISERS M, BLOEMBERGEN D, TUYLS K. Multi-agent Learning and the Reinforcement Gradient[M]. [S.l.]: [s.n.], 2011.
- [5] ADAM S, BUSONIU L, BABUSKA R. Experience Replay for Real-Time Reinforcement Learning Control[J]. IEEE Transactions on Systems Man & Cybernetics Part C, 2012, 42(2): 201–212.
- [6] HESSEL M, MODAYIL J, van HASSELT H, et al. Rainbow: Combining Improvements in Deep Reinforcement Learning[J]. CoRR, 2017, abs/1710.02298.
- [7] Actor-Critic. [J]. Encyclopedia of the Sciences of Learning, 2012.
- [8] PENG B, LI X, GAO J, et al. Adversarial Advantage Actor-Critic Model for Task-Completion Dialogue Policy Learning[J]. 2017.
- [9] LOWER R, YI W, TAMAR A, et al. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments[J]. 2017.
- [10] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal Policy Optimization Algorithms[J]. 2017.
- [11] LITTMAN M L. Markov games as a framework for multi-agent reinforcement learning[J]. Machine Learning Proceedings, 1994: 157–163.
- [12] SINGH S P, KEARNS M J, MANSOUR Y. Nash Convergence of Gradient Dynamics in General-Sum Games[J]. 2013.

- 
- [13] JIANG J, LU Z. Learning Attentional Communication for Multi-Agent Cooperation[J/OL]. CoRR, 2018, abs/1805.07733arXiv: 1805 . 07733. <http://arxiv.org/abs/1805.07733>.
  - [14] LESLIE D S, COLLINS E J. Generalised weakened fictitious play[J]. Games & Economic Behavior, 2006, 56(2): 285–298.
  - [15] YU F R, HE Y. Reinforcement Learning and Deep Reinforcement Learning[M]. [S.l.]: [s.n.], 2019.
  - [16] LITTMAN M L. Friend-or-Foe Q-learning in General-Sum Games[C]// Eighteenth International Conference on Machine Learning. [S.l.]: [s.n.], 2001.
  - [17] SUTTON R S, MCALLESTER D, SINGH S, et al. Policy Gradient Methods for Reinforcement Learning with Function Approximation[J]. Submitted to Advances in Neural Information Processing Systems, 1999, 12: 1057–1063.
  - [18] SILVER D, SCHRITTWIESER J, SIMONYAN K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676): 354–359.
  - [19] SCHULMAN J, LEVINE S, MORITZ P, et al. Trust Region Policy Optimization[J]. Computer Science, 2015: 1889–1897.
  - [20] DAN H, QUAN J, BIDDEN D, et al. Distributed Prioritized Experience Replay[J]. 2018.
  - [21] HAN J, MORAGA C. The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning.[C]// International Workshop on Artificial Neural Networks: from Natural to Artificial Neural Computation. [S.l.]: [s.n.], 1995.
  - [22] GLOROT X. Understanding the difficulty of training deep feedforward neural networks[J/OL]. Proc. AISTATS, 2010, 2010: 249–256. <https://ci.nii.ac.jp/naid/20000618707/en/>.
  - [23] JULIANI A, BERGES V P, VCKAY E, et al. Unity: A General Platform for Intelligent Agents[J]. 2018.



## 附录



## 作者简历

姓名: 谢文韬 性别: 男 民族: 汉 出生年月: 1996-10 籍贯: 江苏南通市

2012.09-2015.07 江苏省南通中学

2015.09-2019.07 浙江大学攻读学士学位

2017,2018 浙江大学计算机学院社会实践奖学金、浙江大学学生会优秀部委  
中控杯机器人大赛二等奖

参加项目:

发表的学术论文:



# 本科生毕业论文（设计）任务书

一、题目：多智能体人机博弈攻防布局

二、指导教师对毕业论文（设计）的进度安排及任务要求：

## 进度安排：

2018.7.1-2018.8.30: 前期调研工作，论文搜集、阅读和复现。

2019.1.1-2019.2.28: 确定选题，选取枪战博弈主题，进行问题的数学建模

2019.3.1-2019.3.31: 核心强化学习算法的选取和改进，工程和算法的实现

2019.4.1-2019.4.30: 改进算法的训练和实验验证，数据的收集整理，着重验证 1V1 博弈环境的效果。

2019.5.1-2019.5.27: 从 1V1 到 2V2 的扩展，数据的整理分析，进行整体工作的整理和论文的撰写。

## 任务要求：

求解博弈问题的数学模型，设计近似纳什均衡策略。在现有多体强化学习算法的基础上进行改进，验证算法性能的提升，与近似纳什均衡进行对抗验证。

起讫日期 2019 年 1 月 1 日至 2019 年 5 月 27 日

指导教师（签名）\_\_\_\_\_ 职称 \_\_\_\_\_

三、系或研究所审核意见：

负责人（签名）\_\_\_\_\_

年 月 日



## 本科生毕业论文（设计）考核

### 一、指导教师对毕业论文（设计）的评语：

本文通过枪战博弈游戏作为例子，改进了两个基于策略梯度的多智能体强化学习算法，并进行了初步实验。是对多智能体博弈与强化学习结合的有益探索。

指导教师（签名）\_\_\_\_\_

年 月 日

### 二、答辩小组对毕业论文（设计）的答辩评语及总评成绩：

成绩比例	文献综述 (10%)	开题报告 (15%)	外文翻译 (5%)	毕业论文质量 及答辩 (70%)	总评成绩
分值	8	13	4		

负责人（签名）\_\_\_\_\_

年 月 日



## **第二部分**

**毕业论文开题报告**



# 浙 大 学

## 本 科 生 毕 业 论 文

### 文献综述和开题报告



学生姓名 谢文韬

学生学号 3150103626

指导教师 潘纲、王志坚

年级与专业 2015级计算机科学与技术

所在学院 计算机科学与技术



## **一、题目：多智能体人机博弈攻防布局**

## **二、指导教师对文献综述、开题报告、外文翻译的具体要求：**

严格按照学校毕业论文开题的要求，具体要求如下：

- (1) 文献综述通过对研究领域的广泛调研，选取合适的综述和前沿进展作为参考文献，结合自己的理解进行阐述，注意引用规范。
- (2) 开题报告部分结合前期工作，对于毕设相关的博弈基础、强化学习理论和平台设计详细介绍技术路线和当前工作，保证内容的真实和准确，引用必须注明。
- (3) 外文翻译须在自己仔细研读的基础上进行，选取文献避免重复劳动，并且必须保证对文献的理解，不可机器翻译。
- (4) 整体开题报告的格式和内容合乎规范，保证简洁和准确。

指导教师（签名）\_\_\_\_\_

年 月 日



## 一、文献综述

### 1 背景介绍

在机器学习领域，强化学习是一个重要分支，强调的是如何在环境中进行感知并采取相应的行动，使得自己的预期收益最大化，最早来源于心理学中的行为主义理论。这是一个有广泛应用意义的思想，在博弈论、控制论、运筹学和统计学等都有重要的应用。在运筹控制领域，强化学习也被称为“近似动态规划”(approximate dynamic programming, ADP)<sup>[1]</sup>。在博弈论中，强化学习作为一种解释有限状态和条件下的平衡问题的工具<sup>[2]</sup>。而在计算机领域，强化学习作为机器学习的一种，用于智能体的构建和认知的模拟。

#### 1.1 研究意义

主流的观念认为强化学习的应用主要在游戏人工智能领域，构建强大的智能体算法来超越人类玩家，但这只是一个狭义的概念。游戏只是强化学习的一个有效载体，作为出发点和落脚点，通过游戏环境的模拟为其他应用提供算法优化和方案迁移的思路<sup>[3]</sup>。从强化学习的本质上看，它最符合人从无知到经验到有知的过程，其思维过程是智能体形成认知的必要准备，进而解开与思维、经验和决策相关的众多领域的问题。

强化学习研究的核心问题在于智能体和环境交互的过程中如何利益最大化，其包含一个连续的过程。如下图所示，智能体从环境中获取自己的状态信息(State)，智能体从自身的 behavior 集合中选取一个或多个行为(Action)施加在环境上，环境给予反馈(通常是 Reward)<sup>[4]</sup>。这样一个过程可以在智能体生命周期内往复运行，智能体在不断和环境交互和得到反馈的时候更新自己选取行为的策略(Policy)函数，进而形成对环境的认知和对某一任务的概念<sup>[5]</sup>。

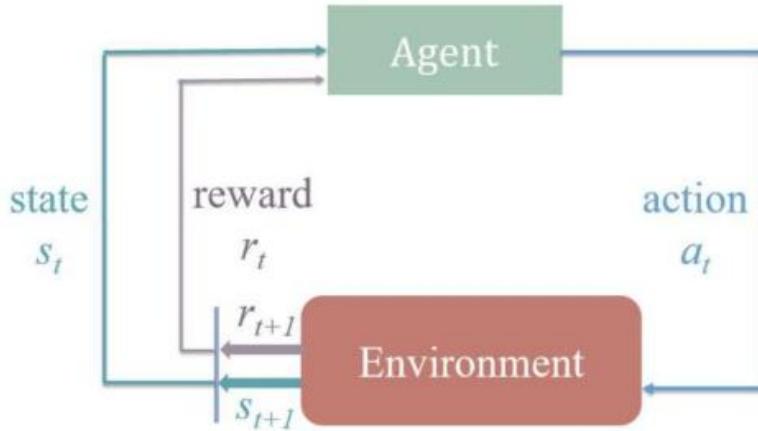


图 1.1: 强化学习智能体和环境交互的示意图

在深度学习领域，最重要的特征是深度神经网络在高维的源数据（如图像，音频，文本等）中提取其有价值的低维特征，从而解决维度灾难的问题。深度强化学习使得原本传统强化学习难以处理的问题能得以解决<sup>[6]</sup>。

通过加深对环境和任务的理解，强化学习可以协助人类完成复杂的任务，也可以作为一种有力的理论探索工具。强化学习最直观的应用就是在电子游戏领域，游戏本身也是玩家通过自身决策行为和游戏环境交互的过程，只是将玩家换成了智能体。人类通过研究强化学习算法，不仅能让智能体在游戏的虚拟环境中训练出强大的适应能力，而且可以将思想和方法迁移到日常生活应用中。在虚拟环境和现实环境中能自主认知的智能体或智能机器人，在室内服务、资源勘探、无人驾驶和竞技对抗等众多领域服务。其思想在其他诸如计算机视觉（Computer Vision）、自然语言处理（Natural Language Processing）和量化分析（Quantitive Analysis）中有重要应用，在数理学科尤其是运筹学、统计学和博弈论中作为有力的实践工具<sup>[7]</sup>。

## 1.2 研究背景和发展脉络

强化学习研究的 State-Action-Reward 架构，实际上就是数学中的马尔可夫决策过程（Markov Decision Processes, MDPs）<sup>[4]</sup>，即一个给定当前状态和过去所有状态的情况下一个随机过程，当前的状态决定未来状态的条件概率分布。其数

学形式如下：

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \dots, S_t] \quad (1-1)$$

马尔可夫奖赏过程 (Markov Reward Process, MRP) 是带有奖赏值的马尔可夫过程，我们可以用一个四元组去表示  $\langle S, P, R, \gamma \rangle$ 。其中  $S$  为有限的状态集合， $P$  为状态转移矩阵， $P_{ss'} = P[S_{t+1} = s' | S_t = s]$ ;  $R$  为奖赏函数， $\gamma$  为折扣因子 (discount factor)，其中  $\gamma \in [0, 1]$ ，在  $t$  时刻的奖赏函数为： $G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ 。而马尔可夫决策过程 (Markov Decision Process, MDP) 是带有决策的 MRP<sup>[4]</sup>，由一个五元组构成  $\langle S, A, P, R, \gamma \rangle$ 。在这个马尔可夫过程上，我们采用的策略 (Policy) 是给定状态下的动作概率分布，即  $\pi(a|s) = P[A_t = a | S_t = s]$ ，在此基础上寻求最优策略的过程就是强化学习解决可抽象成马尔可夫过程的实际问题的模型。

在日常生活中，除了游戏是马尔可夫决策过程，强化学习的应用都可以抽象出这样的数学模型。例如在无人驾驶的场景下，智能体 Agent 即为 AI 控制系统，环境则是给定的道路地图，状态信息包括汽车自身的坐标、速度等信息，如果汽车超速、碰撞或移出地图，会受到惩罚 (Penalty)；而如果汽车在某个特定的 state 下，例如突然出现的行人，采取了避让的 action，则理应得到奖励 (Reward)，在不断训练中则可以提高汽车的驾驶水平。近年来兴起的机器人平台 ROS(Robot Operating System) 也是基于强化学习的马尔可夫过程设计的仿真系统<sup>[8]</sup>，用于实现对智能机器人的行为控制；在视觉任务上，基于视频的动作预测使用强化学习突破了视觉方法的瓶颈；在语音合成和对话领域，策略梯度的方法使得机器对话更具连贯性和交互性；在参数优化和博弈论领域，强化学习可以在资源配置上发挥作用，甚至可以管理谷歌公司的服务集群来节省能耗<sup>[7]</sup>。

纵观强化学习的发展历史，是计算机科学、数理统计学和经济学的共同发展过程。20世纪50年代纳什提出纳什均衡理论<sup>[9]</sup>，为经济学和博弈论提供了坚实的理论基础，和同时期1956年Bellman提出的动态规划算法为强化学习的诞生奠定基础<sup>[10]</sup>。基于马尔可夫决策过程，1992年Watkins提出了Q-learning算法，将传统机器学习的方法与博弈相结合，直到今日Q-learning依然是一大主流流派，其变种被用于解决绝大部分的问题，包括2015年提出的DQN(Deep Q Network，深度Q网络)<sup>[11]</sup>。另一大分支是20世纪50年代提出的虚拟玩家 (Fictitious Play)<sup>[12]</sup> 为20世纪末的策略梯度 (Policy Gradient)<sup>[13]</sup> 算法提供了依据。另一个时间节点就是

2010 年以后深度学习的兴起，强化学习方法都进化为了深度强化学习 (DRL)，主要有 2015 年谷歌提出的 DQN 家族，和基于 DPG(Deterministic Policy Gradient)<sup>[14]</sup> 的 Actor-Critic(行动者-评判者) 机制<sup>[15]</sup>。总体来说，强化学习从传统走向深度，在算法和思想上都在迅速迭代和发展。

## 2 国内外研究现状

### 2.1 研究方向及进展

#### 2.1.1 强化学习

从 2010 年起，深度学习的发展将强化学习带入了一个新的时期，普通强化学习中，状态和动作空间是离散且维数不高尚可解决，高维连续的空间使用传统 Q-learning 的 Q 表不现实，因此就把 Q 表变成一个函数拟合问题。谷歌在 2015 年提出 DQN，利用 CNN(Convolutional Neural Network, 卷积神经网络) 使用 reward 构造标签解决 reward 的稀疏问题，通过借鉴经验池 (experience replay) 的方法来解决相关性及非静态分布<sup>[16]</sup>。其架构如图所示，之后大量的工作都在优化 DQN 的结果，如 DDQN(双向 Q 网络)、DRQN(反向 Q 网络) 和 DQN 集大成者 Rainbow<sup>[17]</sup> 等。

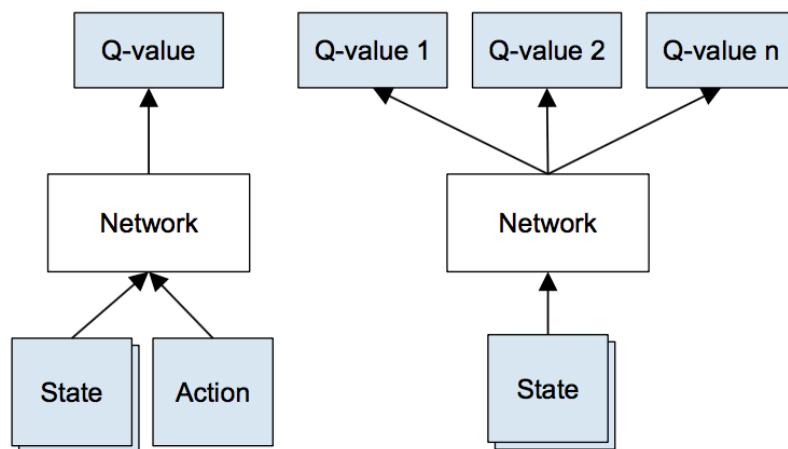


图 2.1: DQN(右与 Q-learning(左) 的区别

策略梯度的方法也在强化学习中占领半壁江山，其从早期的 IGA 算法发展

到基于策略梯度的 Actor-Critic 机制<sup>[15]</sup>。Actor-Critic 机制是指算法分为两部分，一部分是由策略梯度函数构成的 Actor，另一部分是价值网络构成的 critic 来评判 Actor 行为的有效性。就像体操运动一样，一个运动员上去表演，裁判打分来给他评价让其更新其动作。Actor-Critic 机制是基于策略梯度的主流架构，OpenAI 随后相继推出了 A2C(Advantage Actor-Critic)<sup>[18]</sup> 和 A3C(Asynchronous Advantage Actor-Critic)，A2C 在 AC 基础上增加了基线，而 A3C 是 A2C 的异步并行版本，在最新的论文中，A3C 依旧是一个强大的基线算法，其性能和致胜程度在很多的平台上都无可匹敌<sup>[19]</sup>。如下图是 A3C 的架构，由多个分布式 A2C 构成。OpenAI 最新推出 PPO 算法，为了解决强化学习的有效性、简便性和能耗的问题，被认为是一种其他强化学习的简便替代方案，在复杂环境 dota2 游戏中取得了成功<sup>[20]</sup>。

### Asynchronous

Source of image:

<https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9>

1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models

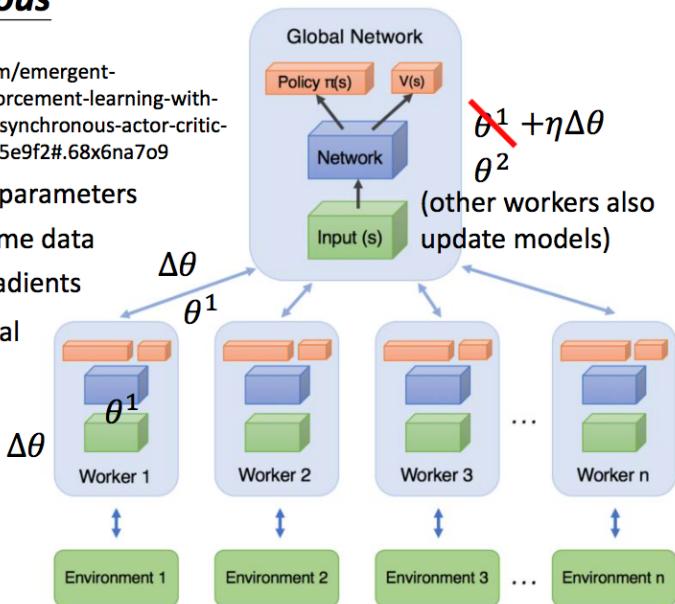


图 2.2: A3C 的架构图

## 2.1.2 多智能体

从智能体角度出发，强化学习分为单智能体和多智能体两方面问题，前沿技术在单智能体上已经提供了非常有效的解决方案，但是针对多智能体一直难以突破，问题的重点在于单智能体和多智能体的协同与独立之间的矛盾<sup>[2]</sup>。智能体在没有先验信息的情况下，默认不知道其他智能体和自己的关系是竞争还是合作，因此需要在竞争合作并存的环境下构建认识的能力。

多智能体算法的基本假设是对手是否完全理性，基于对手完全理性的时候要保证自己的最低收益，因此有 Littman 1994 年提出的 Minimax-Q<sup>[21]</sup> 和 2001 年提出的 Friend-or-Foe Q<sup>[22]</sup>。而基于贪心的最大化收益算法有 2001 年 Singh 提出的 IGA 系列。这些算法在近年深度学习的推进下被改良成了多智能体深度强化学习算法 (MADRL)<sup>[23]</sup>，最具代表性的是基于 DPG 算法的 DDPG(Deep Deterministic Policy Gradient)，加上博弈理论的多智能体部分成为了 MADDPG<sup>[24]</sup>，MADDPG 和它的改良版成为解决复杂智能体问题的前沿算法。

多智能体算法的思路已经从传统的强化学习发展到融入多学科的理论和知识，比如在多智能体之间的通信上的工作，多智能体种群的演化理论和多智能体与物理学中场的结合等。

在部分案例中，智能体互相交互并成功完成了重要的任务。直接通过单智能体深度强化学习的方法应用在学习多智能体策略上是不能适应的，原因有很多<sup>[25]</sup>。首先，从每个单个的智能体角度出发，环境的表现不仅仅依赖单个智能体自身的行为而是所有其他智能体的共同行为，因此表现出一个高度的无序性。这种严重的无序性违反了马尔可夫的假设并且防止了经验回访不假思索地应用，经验回放在稳定训练和改善样本效率上至关重要。此外，每个智能体个体是很难完整和精确地代表整个环境的。典型地，一个智能体仅仅可以获得基于部分环境事实的自身观察。决定哪个智能体应该被基于观察的奖励也是麻烦的。

单独训练每个智能体，因此忽视掉无序的环境，是一个最简单的可能策略。独立的 Q-学习 (IQ-L) 使用这种方法优化了传统的 Q-学习，并且被报道了一些关键的成功即使有均衡性的问题。Tampuu 在 2017 年扩充了 IQ-L 通过深度 Q 学习在一个竞争的多智能体设定下打乒乓球，并且 Tesauro 在 2003 年通过允许智能体的策略依赖对其他智能体策略的估计来解决无序的问题。集中训练分散执行 (CTDE) 的方式已经被广泛采纳用来在训练多智能体系统中解决无序问题。CTDE 使得每个智能体的观察和行为得到提升，并且更好的估计训练时的行为和价值函数。因为每个智能体的策略仅仅依赖他们自己训练时的观察，智能体可以决定在执行的时候分散执行哪个行为。最近，Lowe 在 2017 年把这种方法和 DDPG 算法结合了起来来解决多智能体的微环境，并且 Foerster 在 2018 年使用类似的方法在一个和事实想法的基线上解决了奖励分配的问题<sup>[25]</sup>。

### 2.1.3 仿真平台

强化学习的仿真平台根据不同的应用有很多，也没有一个统一的标准平台。基于博弈论经济学角度的研究有 NetLogo，基于战场攻防有 AFSIM(美国军方使用)，基于机器人控制有 ROS。在一般的强化学习研究中，OpenAI 出品的 Gym 是一个优秀的平台，他集成了数百上千的掌机游戏(如图)，支撑了大量科研的验证工作。而暴雪官方的 pysc2 和 Valve 的 dota2 接口也为训练 ai 的提供了环境支持，虽然每类游戏运行在不同的仿真平台，但是各个运营商都提供了数据接口，为强化学习的发展做出了重大的贡献。



图 2.3: 支持上千游戏的 OpenAI Gym 仿真平台

## 2.2 存在问题

强化学习的发展和机器学习理论是同步的，都在深度学习兴起后得到了突破，但是依然遇到了很多瓶颈。这些瓶颈主要体现在学术界和工业界的不同步上，互相都有前置和滞后的关系。例如，在虚拟环境上的强化学习取得了实验突破后，工业界没有直接能投入使用的产品，或者要等到遥远的未来。另一方面，工业界为学术界提供了巨大的算力支持，但是强化学习的性能问题依然需要结合数理知识和博弈论的支持。AlphaGo 系列，尤其是 AlphaGo Zero，如图可以在 40 天左右就无人能敌，但是其依赖的模拟对局数达到了千万级<sup>[26]</sup>，和人类棋手存在巨大差距，AI 的学习速度和性能问题依然得到学术界的重点关注，例如 AAAI2019 的最佳论文就是在强化学习与蒙特卡洛树搜索的理论证明和优化<sup>[27]</sup>。

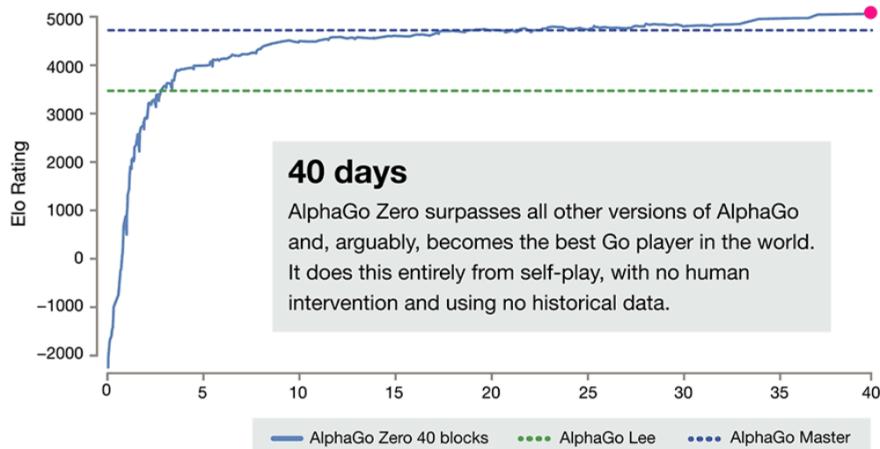


图 2.4: AlphaGo Zero 的训练展示

多智能体强化学习领域存在众多亟待解决的问题，包括智能体间协同与独立的矛盾，复杂干扰环境下的可靠通信机制，大规模智能体的种群管理和演化问题都是学术界的重要议题。

### 3 研究展望

相比于机器学习其他的领域，强化学习和生产生活结合紧密程度有所欠缺。比如，日常使用的计算机视觉技术已经覆盖了生活的方方面面。但是强化学习发挥作用的领域意义重大，强化学习探索与开发的理念对于人类未来的科学的研究有指导意义和实际应用价值<sup>[2]</sup>。

针对前沿强化学习研究存在的问题，未来强化学习需要在性能提升、理论结合和实践落地方面开展工作。迄今为止的实验和应用依然依赖巨大的计算开销，需要数理和博弈论的研究来推动算法的优化。理论界在多智能体的强化学习上依然要提出更有效的算法来促进智能体间的通信和协同<sup>[25]</sup>；实践方面，一方面要构建更完善的多智能体仿真平台来适应真实场景的需求，另一方面要开展真实环境测试工作。结合产品而言，强化学习要在无人驾驶和智能机器人等领域发挥作用，真正造福人类。

## 4 参考文献

- [1] 金玉净. 近似动态规划在资源配置中的应用研究[D]. 苏州大学, 2014.
- [2] 强化学习在多智能体系统任务分配中的应用. [J]. 工业信息学研究室, 2004.
- [3] K T, A N. Evolutionary game theory and multi-agent reinforcement learning[M]. [S.l.]: [s.n.], 2005.
- [4] LEVIN E, PIERACCINI R, ECKERT W. Using Markov decision process for learning dialogue strategies.[C]// IEEE International Conference on Acoustics. [S.l.]: [s.n.], 2002.
- [5] SUTTON R S, BARTO A G. Reinforcement Learning: An Introduction[J]. IEEE Transactions on Neural Networks, 1998, 9(5): 1054–1054.
- [6] YU F R, HE Y. Reinforcement Learning and Deep Reinforcement Learning[M]. [S.l.]: [s.n.], 2019.
- [7] NOWÉ A, BRYS T. A Gentle Introduction to Reinforcement Learning[M]. [S.l.]: [s.n.], 2016.
- [8] KOUBAA A. Robot Operating System (ROS): The Complete Reference (Volume 1)[M]. [S.l.]: [s.n.], 2016.
- [9] NASH J. Non-Cooperative Games[J]. Annals of Mathematics, 1951, 54(2): 286–295.
- [10] BELLMAN R. On a routing problem[J]. Quarterly of Applied Mathematics, 1958, 16(1): 87–90.
- [11] MNIIH V, KAVUKCUOGLU K, SILVER D, et al. Playing Atari with Deep Reinforcement Learning[J]. CoRR, 2013, abs/1312.5602.
- [12] YOUNG H P. The Evolution of Conventions[J]. Econometrica, 1993, 61(1): 57–84.
- [13] BAXTER J, BARTLETT P L, WEAVER L. Experiments with Infinite-Horizon, Policy-Gradient Estimation[J]. J Artificial Intelligence Research, 2001, 15(1): 351–381.

- [14] CAI Q, PAN L, TANG P. Deterministic Policy Gradients With General State Transitions[J]. 2018.
- [15] Actor-Critic. [J]. Encyclopedia of the Sciences of Learning, 2012.
- [16] ADAM S, BUSONIU L, BABUSKA R. Experience Replay for Real-Time Reinforcement Learning Control[J]. IEEE Transactions on Systems Man & Cybernetics Part C, 2012, 42(2): 201–212.
- [17] HESSEL M, MODAYIL J, van HASSELT H, et al. Rainbow: Combining Improvements in Deep Reinforcement Learning[J]. CoRR, 2017, abs/1710.02298.
- [18] PENG B, LI X, GAO J, et al. Adversarial Advantage Actor-Critic Model for Task-Completion Dialogue Policy Learning[J]. 2017.
- [19] BAAZIZADEH M, FROSIO I, TYREE S, et al. Reinforcement Learning through Asynchronous Advantage Actor-Critic on a GPU[J]. 2016.
- [20] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal Policy Optimization Algorithms[J]. 2017.
- [21] LITTMAN M L. Markov games as a framework for multi-agent reinforcement learning[J]. Machine Learning Proceedings, 1994: 157–163.
- [22] LITTMAN M L. Friend-or-Foe Q-learning in General-Sum Games[C]// Eighteenth International Conference on Machine Learning. [S.l.]: [s.n.], 2001.
- [23] KUMAR S, SHAH P, HAKKANI-TUR D, et al. Federated Control with Hierarchical Multi-Agent Deep Reinforcement Learning[J]. 2017.
- [24] LOWER R, YI W, TAMAR A, et al. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments[J]. 2017.
- [25] KILINC O, MONTANA G. Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations[J]. 2018.
- [26] 胡晓峰, 贺筱媛. AlphaGo 的突破与兵棋推演的挑战[J]. 科技导报, 2017, 35(21): 49–60.

- [27] EFRONI Y, DALAL G, SCHERRER B, et al. How to Combine Tree-Search Methods in Reinforcement Learning[J]. 2018.
- [28] SILVER D, SCHRITTWIESER J, SIMONYAN K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676): 354–359.
- [29] BROWN N, SANDHOLM T. Safe and Nested Subgame Solving for Imperfect-Information Games[J]. 2017.
- [30] VINYALS O, EWALDS T, BARTUNOV S, et al. StarCraft II: A New Challenge for Reinforcement Learning[J]. 2017.
- [31] 李鸿, 吴嗣亮, 杨春山. 对策论在雷达反干扰作战中的应用[J]. 现代雷达, 2008, 30(2): 15–17.



## 二、开题报告

### 1 问题提出的背景

#### 1.1 课题背景

近年来，计算机科学界掀起了人工智能 (AI) 的浪潮，得益于计算集群和神经网络硬件部署技术的发展，传统的机器学习理论和人工智能算法有了验证的平台，也提供了更多落地成为产品的可能。人工智能领域的技术涵盖计算机视觉 (Computer Vision)，自然语言处理 (Natural Language Processing) 和游戏智能体研究 (Game Agent) 等<sup>[2]</sup>。以 AlphaGo 系列为为代表的智能体多次击败人类棋手，让人感受到人工智能强大力量的同时，也让人们思考人工智能未来的发展方向和潜在的风险问题。

早在 20 世纪 90 年代，IBM 公司的计算集群“深蓝”系列就在国际象棋项目上击败了人类世界冠军卡斯帕罗夫，人们在赛前普遍对国际象棋大师充满信心，但是终究在 AI 的面前败下阵来。重达 1270 公斤，有 32 个“大脑”(微处理器)，每秒可以计算 2 亿步的计算集群代表了当时棋类人工智能的最高水平。21 世纪以来，游戏 AI 依然不断发展，基于传统的蒙特卡洛搜索技术和经典强化学习算法实现了一批优秀的游戏 AI，2010 年之前棋牌类游戏平台的顶级 AI 都达到了一般大师级的水准。

真正掀起游戏智能体研究热潮的是 2016 年 3 月击败人类九段棋手李世石的 AlphaGo<sup>[26]</sup>，各大科研机构争先建立研究强化学习与游戏 AI 的小组。国际方面，特斯拉建立的 OpenAI，谷歌建立的谷歌大脑 (DeepMind) 在该领域各占半壁江山；国内方面，大型互联网公司腾讯、阿里和网易等都建立了强化学习研究小组，顶尖高校和中科院都在一些知名的大型游戏上取得了突破性进展，达到与国际水平媲美的程度。研究项目从传统的棋牌类游戏，已经发展到了大型掌机游戏如任天堂系列、顶级即时战略游戏 (RTS, Real-Time Strategy) 如星际争霸 II、顶级角色扮演类游戏 (RPG, Role-Play Game) 和多人合作对抗型游戏如 Dota2<sup>[17]</sup>。

20 世纪 50 年代约翰·福布斯·纳什 (John Forbes Nash JR) 证明了博弈平衡

点纳什均衡的存在<sup>[9]</sup>，为博弈理论提供了坚实的基础。博弈理论中有很多经典问题，如囚徒困境、旅行者博弈和智猪博弈等，为现代经济学发展贡献了许多重要的议题。现代博弈论经常和强化学习理论结合，用来解决互联网竞价、市场拍卖和政治外交等议题。当然，许多多人对局的游戏和棋牌都存在博弈论的思想，所以游戏 AI 也是博弈论的重要用武之地，近年来将博弈用在解决德州扑克的问题甚至得到了高度的政治重视。

博弈论作为强化学习研究的重要理论基础，在游戏 AI 算法的设计中起到重要作用。近年来的研究趋势也集中在如何使得算法更快地逼近纳什均衡，如何判断对手的理性程度，如何将博弈的理论应用在智能体的训练和决策过程中去。主流的强化学习算法的设计仍然更多依赖搜索和模拟，不仅有巨大的开销和计算资源占用，而且其决策行为在博弈上的证明和解释都有欠缺<sup>[27]</sup>。博弈论和强化学习的学科交叉，在顶级人工智能会议中备受青睐，其对于构建智能体的思维和决策能力有重要理论意义和应用价值。不仅仅是在游戏上构建出强大的电脑玩家，而是训练出智能体的思维能力和理性决策能力。无论是经济学领域的竞价拍卖，还是设计更具思维能力的智能机器人，都有重大的应用前景。

## 1.2 同类研究现状

近年来，游戏 AI 取得了长足的进步和发展，得益于大型互联网公司和各国对其的重视和巨大投入。研究的问题重点在基于博弈的棋牌游戏如围棋和德州扑克，基于 Atari 游戏（雅塔利游戏，任天堂公司出品的 80,90 年代的简单掌机游戏集合）的游戏类研究，基于即时战略的游戏星际争霸 II（主要由 DeepMind 主导）和基于多人合作多抗游戏 Dota2（主要由 OpenAI 主导）等。

自 2016 年以后，AlphaGo 系列的 AI 棋手称霸围棋区，包括 AlphaGo 的进阶版 AlphaGo Master 在棋类平台上无限连胜击败所有对手。而全新版本的自我博弈从 0 学习的 AlphaGo Zero 经过 3 天就击败了李世石的对手 AlphaGo Lee，21 天超越 Master，40 天登顶世界第一<sup>[28]</sup>。同为 2017 年，卡耐基梅隆大学的作品德州扑克大师 Libratus（“冷扑”）在无限德州扑克比赛中战胜人类顶尖选手，在不完美信息博弈中取得重大突破从而获得 NIPS2017 最佳论文<sup>[29]</sup>。

在 Atari 游戏方面，OpenAI 构建了一个开源训练平台 Gym，集成了上千个

Atari 游戏，为玩家快速验证想法，设计属于自己的 AI 提供平台，同时也在该平台上验证了许多兼顾性能和致胜程度的算法。经过人工智能顶会的论文调研，一半以上的会议论文会在 Gym 上验证算法的性能以证明其有效性和泛化能力。

即时战略类游戏星际争霸 II 上，暴雪公司公开了训练星际争霸 II 主游戏和相关小游戏的测试接口 pysc2，DeepMind 围绕其进行了大量的研究工作，与此同时南京大学团队和腾讯团队也在分层强化学习领域取得突破并在星际争霸相关环境下得到了验证。各机构和职业玩家的星际争霸 II 对抗也在不断进行，在限制 apm(Action Per minute, 每分钟行为的数目) 的情况下，依然和人类选手互有胜负<sup>[30]</sup>。

多人合作对抗问题 Dota2 之前未有公司涉猎，而 OpenAI 团队在 2018 年推出了击败人类平均 6000 分职业选手团队的版本(人类 top1%)，并陆续击败了大量人类玩家，他们将与 2019 年 4 月 13 日挑战 2018 年冠军战队 OG。多智能体的合作，前所未有的复杂环境，OpenAI 提出的近端策略优化 (PPO) 成为了复杂任务中的主流算法<sup>[20]</sup>。

除了上述主流研究方向外，关于强化学习理论的研究主要集中在蒙特卡洛树搜索 (MCTS) 的理论研究、博弈理论和虚拟环境机器人训练上。其中强化学习理论的研究收获了 AAAI2019 的最佳论文和最佳提名<sup>[27]</sup>，基于博弈论的近似求解、搜索优化和算法原理都是研究的方向趋势。在复杂的游戏平台上利用强化学习训练一个足够强大的 AI 需要足够多的计算资源，特别是基于图像视觉信息的输入问题则依赖 GPU(图形处理器, Graphics Processing Unit) 集群的训练。主流的强化学习算法依旧依赖大量的试错和经验重放<sup>[16]</sup>，智能体需要在虚拟环境中反复适应和迭代来进化，其对于纳什均衡的收敛性和近似性依旧是重大价值的研究议题。

在游戏 AI 领域，除了 Dota2 项目，大部分都是单智能体或多个智能体集成在同一单位上的问题，多智能体强化学习的理论都源自经典的多体理论，而与博弈论紧密相关的种群理论和平均场理论也相继提出，对于解决智能体之间共享策略和管理问题起到了关键的作用，给多智能体强化学习的训练策略和算法设计提供了思路和理论基础。

同类型的游戏 AI 研究大多数集中在神经网络结构的修改和某个特定游戏的优化，很少从博弈理论出发去证明算法的有效性。2017 年卡耐基梅隆大学的作

品”冷扑”是一个将博弈应用在强化学习中的极佳例子，从博弈和强化学习两方面解释了德州扑克的对抗过程<sup>[29]</sup>。早期的论文中有对于博弈的研究，但是缺乏在游戏场景中的分析和验证。

### 1.3 项目来源及概括

在各种游戏的博弈过程中，简单游戏的全过程和复杂游戏的子过程，存在确切的纳什均衡解。例如之前玩家众多的游戏《绝地求生》中，决赛圈玩家会聚集在一个草丛，玩家都会选择在草丛中匍匐前进，此时如果一方站立可能会发现敌方，也有暴露的风险；同时，远距离开火能占得先机，但是命中率低还会暴露自身位置。如果把对应的函数关系求出，就可以获得双方策略集合的收益关系矩阵，进而分析把握纳什均衡点。同样类似的还有 RPG 类游戏对决过程，关于决策使用技能的对抗关系，都是子博弈的过程，通过几个对抗型子博弈的叠加，可以视为多个不同任务的智能体的合作。如果每个对象有多个，他们之间或存在对抗关系，或是合作关系，则整个战场就可以建模，构建一个多智能体研究平台，研究多个子博弈和多智能体的强化学习。

在强化学习中，策略空间和智能体移动的范围决定了问题的复杂度，多数基于搜索的强化学习算法依赖大量的尝试去更新策略，但是如果把构成可解博弈的子问题分离出来<sup>[29]</sup>，这部分可解问题就会极大优化搜索空间。研究不同强化学习算法在可解纳什均衡上的致胜程度和收敛特征，对于纳什均衡解无法计算或存在多解的时候有借鉴和推广的价值。通过随机数的 action 反馈概率，使得智能体从环境中的信息认知有一个折扣因子 /*gamma*，也可以以此训练智能体对于环境的探索 (exploit) 和信息的甄别 (discrimination)。

项目的来源主要有 OpenAI 的 MADDPG(Multi-Agent Deep Deterministic Policy Gradient) 算法<sup>[24]</sup> 和 PPO 算法 (Proximal Policy Optimization)<sup>[20]</sup>，它们在验证算法在强化学习算法间的性能上做了有力的证明，但是没有和纳什均衡结合起来进行解释和分析，而早期的多智能体算法也缺乏在游戏中的验证。

本项目从一个实际的游戏对抗场景入手，构建一个存在移动，攻击和侦察三大类决策行为的问题，研究多智能体强化学习算法的致胜程度和收敛特性。参考传统强化学习与博弈论的 Minimax-Q<sup>[21]</sup> 和 Friend-or-Foe Q<sup>[22]</sup> 的合作对抗

关系，优化深度强化学习算法 MADDPG(Multi-Agent Deep Deterministic Policy Gradient)，在自己搭建的平台中优化算法和模型，并在其他有类似结构的游戏 中应用算法并测试致胜程度，如 OpenAI-Gym 的类似游戏，pysc2 中的地图，以及 实验室仿真的第一人称射击游戏等。

## 2 本研究的意义和目的

### 2.1 研究意义

通过强化学习与博弈论的结合，验证多智能体强化学习算法的有效性，进而对算法进行评估和优化，提高在纳什均衡不可解情况下的算法致胜程度，应用在多个游戏和平台上。

本项目旨在多智能体平台上结合强化学习和博弈论，构建一个测试平台并验证改进算法，将优化子结构应用在多个平台上，提升强化学习的性能和致胜程度，进而应用在其他马尔可夫决策问题的优化上。

### 2.2 研究目标

- (1) 构建一个博弈可解的验证平台
- (2) 改进 MADDPG 和 PPO 算法，验证与纳什均衡对抗的致胜程度
- (3) 移植到其他平台，如 OpenAI-Gym，Pysc2 或实验室的第一人称枪战游戏
- (4) 后期可以在毕业论文基础上延伸形成会议级别的论文

## 3 项目的主要内容和技术路线

### 3.1 主要内容

- (1) 从博弈论角度出发，构建基于移动、攻击、侦察的行为对抗关系并求纳什均衡，生成对应策略的基线对手。
- (2) 根据 AFSIM、MAgent 和 Gym 等平台的 Agent 设计思路构建平台环境
- (3) 实现 MADDPG 和 PPO 的优化版本，在平台上测试修改后，移植到其他平台，如 OpenAI-Gym，Pysc2，实验室枪战环境

(4) 评价标准: 与纳什均衡对抗致胜程度、训练速度、收敛速度等

## 3.2 技术路线

### 3.2.1 总路线

总路线如图 3.1 所示围绕三大模块，分别是博弈论基础、平台架构和强化学习。首先，通过博弈论基础建立数学模型，接着根据模型来搭建测试平台，最后将强化学习算法应用该平台上，强化学习模块和博弈论基础密切相关。

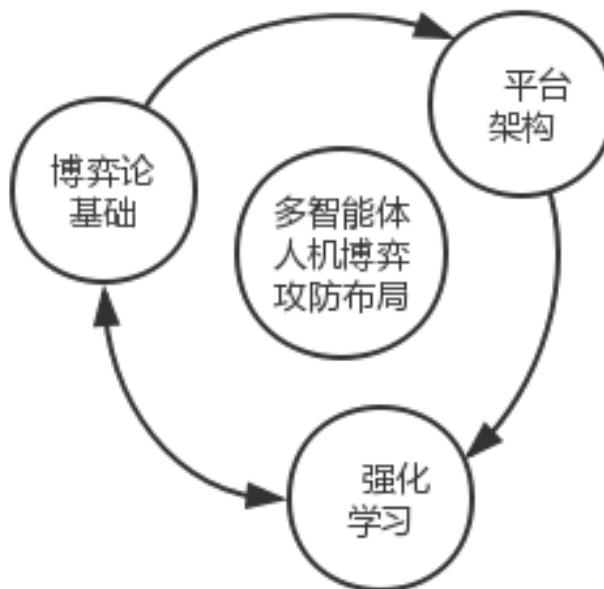


图 3.1: 项目技术总路线

### 3.2.2 博弈论基础

本模块如图 3.2 所示，主要通过现实问题的数学建模，根据概率密度函数和设计的对抗关系构造支付矩阵 (payoff)，进而计算纳什均衡。

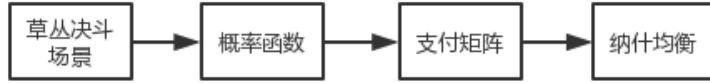


图 3.2: 博弈论基础技术路线

根据建模出的行为空间，计算每个 action 映射到系统结果的函数，从而计算支付矩阵。支付矩阵 (payoff-matrix) 包含了备选行动方案、状态信息和效用值的矩阵。备选行动方案  $(a_1, a_2, \dots, a_n)$  是行动空间，行为之间两两独立。所有行为的集合  $A$  称为决策空间，状态  $(e_1, e_2, \dots, e_m)$  是环境带来的状态空间，往往是按照概率随机的，用概率  $(P_1, P_2, \dots, P_m)$  表示，所有状态的集合  $E$  称为状态空间，对应于行动方案  $a_i$  在状态  $e$  下的收益值为  $(a_i, e_j)$ 。其集合  $Q = \{q_{11}, q_{12}, \dots, Q_{1m}, \dots, q_{n1}, q_{n2}, \dots, Q_{nm}\}$  的效用值域为  $U = \{u_{11}, \dots, u_{nm}\}$ ，其中  $u_{ij} = u(q_{ij})$ 。

在求解纳什均衡的时候，通常最优纯策略解不存在，此时的纳什均衡是一个最优混合策略解。假设  $\mathbf{A}, \mathbf{B}$  连根对抗， $\mathbf{B}$  以概率  $x_i (i = 1, 2, \dots, m)$  策略集  $S_2$  中选取纯策略  $b_i (i = 1, 2, \dots, m)$ ，则  $\mathbf{B}$  的混合策略集为  $S_2^* = \{X = (x_1, x_2, \dots, x_m) | x_i \geq 0\}$ ，同理可得  $\mathbf{A}$  的概率为  $y_j (j = 1, 2, \dots, n)$ ，对于  $\mathbf{X} \in S_2^*, \mathbf{Y} \in S_1^*$ ， $\mathbf{B}$  的收益函数为  $E(X, Y) = XAY^T = \sum_{i=1}^m \sum_{j=1}^n x_i y_j q_{ij}$ ，乙方收益为  $E(X^*, Y^*) = X^*A(Y^*)^T$ ，纳什均衡可以通过迭代法求得。

### 3.2.3 强化学习基础

马尔可夫决策过程 (Markov Decision Process, MDP) 可以用一个元组  $(S, A, P, R, \gamma)$  来表示， $S$  是决策过程中的状态集合， $A$  是决策过程中的行为集合， $P$  是状态之间的条件转移概率， $R$  是某一行为达到某一状态的回报， $\gamma$  是折扣因子，由于转移概率和 Markov 随机过程不同，所以转移表达式为：

$$p_{ss}^a = p(\hat{s}|s, a) = p(S_{t+1} = \hat{s}|S_t = s, A_t = a) \quad (3-1)$$

在 MDP 中，可以用  $\pi(a|s) = p(A_t = a|S_t = s)$  来表示决策，采用一个折扣

因子  $\gamma$  来减小后面的 Reward 对当前状态的影响，所以累计奖励为：

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (3-2)$$

状态-动作函数为  $q_{\pi}(s, a) = E_{\pi} [\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a]$ ，联立得知状态-行为函数的贝尔曼方程为  $q_{\pi}(s, a) = E_{\pi} [R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$ 。

于是最优值函数为  $V^*(s) = \max_{\pi} E \left[ \sum_{t=0}^H \gamma^t R(S_t, A_t, S_{t+1}) | \pi, s_0 = s \right]$ 。

基于上述 MDP 理论，构建一张 Q 表， $Q(s, a)$  为某一时刻  $s$  采取动作  $a$  能收获的期望，则按照贝尔曼方程，Q 表的更新方法为：

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[ r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right] \quad (3-3)$$

传统的 Q-learning 可以解决相当部分的强化学习问题，但是涉及到高维的搜索空间和多智能体情况就需要改进算法。

### 3.2.4 多智能体强化学习

一个随机博弈和一个多智能体强化学习过程存在众多一致性。其实这两点不能完全等价，随机博弈中假定每个状态的奖励矩阵 (payoff) 是已知的，不必进行学习。而多智能体强化学习则是通过与环境的不断交互，通过更新的函数来学习每个状态的奖励值函数，通常是 Q 函数，再通过这些奖励值函数来学习得到最优纳什策略。通常情况下，模型的转移概率与奖励函数直接相关，因此需要利用到 Q-learning 中的方法来不断逼近状态值函数，或是动作-状态值函数。在多智能体强化学习算法中，两个主要的技术指标为合理性与收敛性。

合理性 (rationality) 是指在对手一直采用同一策略的情况下，当前智能体能够学习并收敛到一个相对于对手策略的最优解。收敛性是指其他智能体也使用某种强化学习算法时，当前智能体通过训练可以收敛到一个稳定的策略。通常情况下，收敛性针对系统中的所有的智能体使用相同的学习算法。

考虑到博弈对手的特点，算法分为均衡学习 (Equilibrium Learner) 和最大回报学习 (Best Response Learner)。均衡学习考虑的问题在于假设队友同样是理性思维，或者通过博弈或强化学习训练出的模型，则要确保在最坏的情况下获得最稳定的收益。而若对手是一个非完全理性的选手，如普通人类对手，策略则重于

获取最高的收益，是一种贪心的行为，也往往可以获得最佳的效果，而不是如均衡学习中的求稳方式。

均衡学习主要算法有三种 Q 算法，分别是 MiniMax Q(1994 Littman)<sup>[21]</sup>、Nash Q (1998 Hu, 2003 Hu) 和 Friend-or-Foe Q (2001 Littman)；最大回报学习分为 Fictitious Play (1951 Brown)、Joint Action Learner (1998 Claus)、Infinitesimal Gradient Ascent (IGA) (2000 Singh) 和 WoLF-IGA(2001 Bowling)。总的来说，均衡学习可以保证最坏情况下的 payoff，但是部分算法可能存在多个纳什解，在对手采用非理性策略时容易获得少的收益，使得纳什均衡点并不是一个好的策略。而最大回报学习则可以有效利用对手的非理性行为获得更高的 payoff，同时也可以应对 payoff 矩阵动态变化的情况，但是利用敌人次优解的行为也容易被对手加以利用。

Minimax-Q 算法应用于两个玩家的零和随机博弈中。Minimax-Q 中 Minimax 指的是使用上一篇文章中的 minimax 方法构建线性规划来求解每个特定状态 s 的阶段博弈的纳什均衡策略。Q 指的是借用 Q-learning 中的 TD 方法来迭代学习状态值函数或动作-状态值函数。在两玩家零和随机博弈中，给定一个状态 s，则定义第 i 个智能体的状态值函数为：

$$V_i^*(s) = \max_{\pi_i(s)} \min_{a_{-i} \in A_{-i}} \sum_{a_i \in A_i} Q_i^*(s, a_i, a_{-i}) \pi_i(s, a_i), i = 1, 2 \quad (3-4)$$

在执行 Q 学习的过程中，Q 表的表项  $Q(s_t, a_t)$  表示  $(state_t, action_t)$  的 value。更新函数通过学习率  $\alpha$  进行更新： $Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + (\alpha r_t + V(s_{t+1}))$ 。

在基于 Minimax 的基础上，FFQ(Friend-or-Foe Q) 通过区别对待对方智能体是合作或是对抗关系来采取不同的策略函数，具体为如果敌方为合作关系：

$$\text{Nash}_1(s, Q_1, Q_2) = \max_{a_1 \in A_1, a_2 \in A_2} Q_1[s, a_1, a_2] \quad (3-5)$$

如果为对抗关系，则策略函数为：

$$\text{Nash}_1(s, Q_1, Q_2) = \max_{\pi \in \Pi(A_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi(a_1) Q_1[s, a_1, a_2] \quad (3-6)$$

基于以上的传统多智能体强化学习的思路，可以在深度强化学习中进行算法的改进，而需要改进的算法是基于策略梯度的算法 MADDPG(Multi-Agent Deep Deterministic Policy Gradient)。其架构如图 3.3 所示：

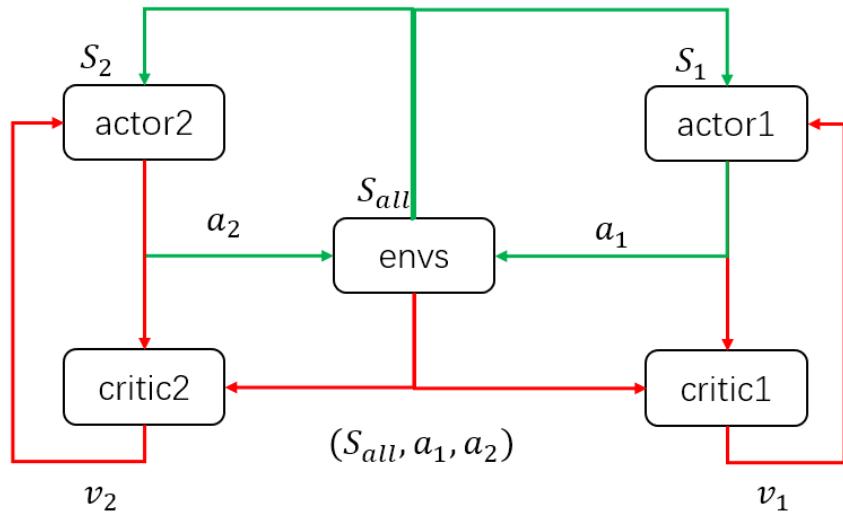


图 3.3: MADDPG 的架构图

当策略在训练对的时候，actor 需要和环境进行交互，通过绿色的循环，获取在环境中的状态  $S$ ，采用的具体动作  $a$ ，训练时 critic 把  $(S, a)$  输入，输出价值  $v$  来评估当前动作的收益，使得 actor 获得反馈从而改进自己的策略，图中是两个智能体的 AC 架构，总体思想上和 DDPG 还是一致的，这里就可以在其中添加博弈论的方式使得两个 Agent 存在合作和竞争关系。

### 3.2.5 平台架构

主流的多智能体仿真平台根据不同的作用分类，有军事仿真用的 AFSIM，有研究博弈论的 NetLogo，也有为研究大数据量智能体的 AAAI2018 论文 MAgent，可以从中获取构建一个多智能体平台的要素。

AFSIM(The Advanced Framework for Simulation, Integration and Modeling, 仿真，集成和模拟的高级框架) 平台是众多智能体仿真平台中实用性较强的，是一个面向对象、基于 C++ 的集成开发环境，用于构建军事级别的资源配置分析和仿真。AFSIM 平台的开发对象包括模拟武器的运动学过程、感知系统、电力伺服系统、通信网络、高级追踪系统等战场管理对象。AFSIM 平台基于早期的 AFNES(Analytic Framework for Network-Enabled Systems 平台)，AFNES 平台在 2013 年 2 月开源。该平台主要由三大模块组成，分别是提供重要接口和功能的框架部分、集成开发环境和可视化模块，为用户提供模拟的顶层控制，特别是对

于时间和事件的管理，还包括对众多用户定义的行动单位、感知器、武器和处理器的行为控制。

AFSIM 的框架架构如下图 1，组件主要包括(1)仿真对象的类定义，包括行动单位、感知器等等;(2)事件和时间的控制，实体数据产生的消息;(3)支持系统的标准数学库，例如存储数据和对象的标准容器;(4)导入标准地理数据的模块，方便导入标准化的地理情况数据，用于真实战场的仿真;(5)支持脚本化的对象管理，形式上简化 API 的复杂程度;(6)通信网络的模拟构建，提供例如网络节点、路由器、多通道协议和消息队列等等；(7)电力伺服系统仿真，包括各类线路噪声和拥塞控制方面的情况;(8)基于信息流和任务的模拟，构建系统和玩家之间的桥梁;(9)支持批处理和实时系统的不同情况;(10)用户的应用程序接口。除了各个组件的核心部分之外，还支持 RIPA(Reactive Intergrated Planning Architecture) 的智能体算法，支持各种量化的任务等等内容。

MAgent 是一个命令行的工具，比 AFSIM 轻量级很多，MAgent 运行的环境设定是一个方块的世界，由智能体和墙组成，多智能体的控制方式为成群控制，成群的智能体拥有同样的总体属性和控制算法。智能体的主要属性有高度、宽度、速度和生命值等等，在类似的游戏规则下可以适当扩充，而这些是必要的属性。多智能体的决策行为包括移动、攻击和转变，通过设定行为的范围可以进一步划分为多个子动作，作为决策的元过程。在进行强化学习的训练过程中，奖励机制可以由测试者进行自行指定，而在进行 DQN 网络构建等过程中的奖励函数也由测试者自行进行调试。在本项目中，主要采用的基线算法是 DQN、DRQN 和 A2C 的 Tensorflow 和 Mxnet 实现，并在 MAgent 的环境下测试出 DQN 具有最佳的运行效果。由于核心的 train 代码需要在 GTX1080Ti 上运行约一天左右的时间，当前尚未具备硬件测试条件，所以当前仅仅是运行了测试可视化的脚本模块。

脚本展示的战场游戏是一个在 2018 年开源的多智能体模拟平台，两个军队，各拥有 64 个同质的多智能体，通过互相协作来消灭敌人获得更高的奖励，采用的行动分为移动和攻击。默认的奖励机制设定如表 1 的战场信息列表，可以根据用户自己的需求修改脚本中的各个参数值。而系统设定的 AI 学习策略 MF-Q 和 MF-AC 分别在两只军队上使用，场景设定是经典多智能体问题的战场 mean field，算法分别采用 Q 函数和 Actor-Critic 机制。Q 函数是通过构建深度 Q 网络

实现，而 Actor-Critic 维护了一个随机决策的 Actor 函数和对决策进行打分评判的 Critic 函数，两种不同的更新状态机制。在实际测试结果中，MF-Q 和 MF-AC 的表现要明显优于其他的策略，而 MF-Q 甚至是达到了 0.8 以上的胜率。

通过这些仿真平台的分析，可以得出仿真平台的几大要素是战场、智能体、策略和可视化的部分。大致框架如下图 3.4，战场部分包括规则、资源和玩家信息，智能体则维护自身的决策空间、状态和奖励信息，可视化部分负责渲染整个战场，而算法部分可以计算一些参数，比如纳什均衡矩阵和 Q 表等，并实现例如 MADDPG 的算法。

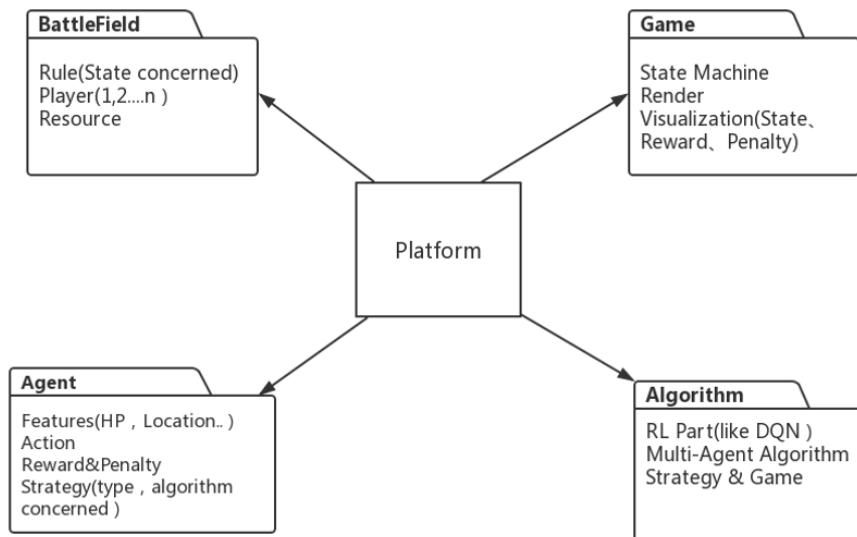


图 3.4: 平台构建相关的设计

### 3.3 可行性分析

#### 3.3.1 理论可行性

本论文研究的主题是构建一个敌我博弈的环境，理论基础建立在对抗双方的纳什均衡上，使得算法有一个评估的标准，纳什均衡既是对整个战场敌我状态的认识，也是算法布局的重要依据。如果战场一方采用混合纳什均衡的策略布局，己方就可以以此为基线来训练自己的算法，验证强化学习算法的性能和致胜程度，也可以训练出对敌方策略布局是否完全理性的认识。根据传统多智能体强化学习的理论，可以在和对方的博弈过程中判断对方策略的理性程度，从而

做出自己的攻防布局。在敌方完全非理性的情况下，可以采用贪心的方法获得最大的收益。而在敌方按照理性策略情况下，要保证自己的收益，采取一种“求稳”的策略。强化学习的算法需要有纳什均衡存在的支撑，来获得一个良好的训练对象和测试基线，并且获得对方理性程度的认识。<sup>[22]</sup>

首先，以一个线性决斗的场景为例阐释纳什均衡的存在。草丛中有两个部队匍匐而行，如图 3.5 展示的两个玩家进行攻防对抗，1V1 的情况下部队里的枪手各自在两个端点上，双方相向行进，在远距离 (Far)、中距离 (Mid) 和近距离 (Near) 上各有一次开火机会，双方射击精确度各不相同。<sup>[31]</sup>

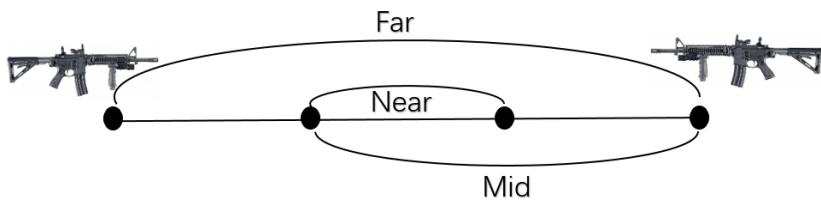


图 3.5: 双方玩家对抗博弈的示意图

游戏中双方士兵的射击命中率和距离的映射关系不同，而双方的决斗是一个零和博弈过程，原理上通过映射关系可以求出他们的支付矩阵 (payoff matrix)。在原理上根据双方对抗的 payoff 矩阵，可以求得 A 和 B 双方的纳什均衡点，在此均衡点双方不能仅通过改变自己的策略获利。如果我们把这里的一维问题推广到二维坐标系，只要拟合出不同的变化函数，按照  $p_m \propto r^{-2}$  或  $p_m \propto r^{-1}$ 。此时双方的状态从一维的点集上升到二维  $(x, y)$  坐标，此时的  $p_m$  根据 A,B 距离  $d = \sqrt{x^2 + y^2}$ ，根据散点拟合即可，如下图 3.6 是对 10\*10 坐标下的拟合情况，可以分割为 10 个单位距离，也可以当成连续性的行进问题。

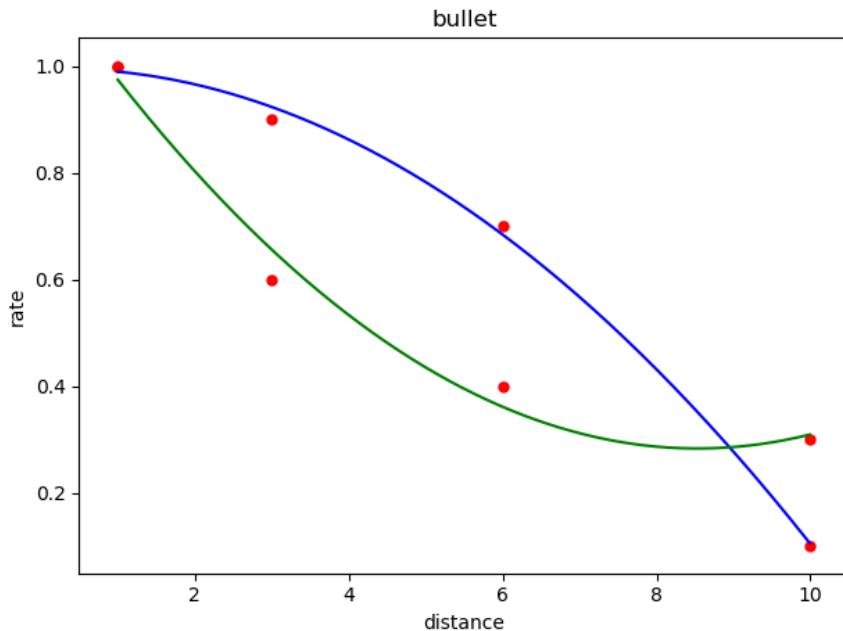


图 3.6: A,B 两种导弹随距离的函数变化

在射击的基础上，为了在草丛中定位敌方，我们加入声波探测器 (Probe)、声波干扰器 (Jam) 和监听器 (Listener) 的对抗。按照真实战场的对抗关系，探测器 P 可以侦测敌方，干扰器 J 可以干扰雷达，而无监听器 L 可以监听敌方探测器和干扰器的声波并且不会暴露自身，这三者因此也形成了博弈对抗关系。假设监听器 L 可以一直开着，那么双方的探测器和干扰器都可以开或关，则双方各自有四种状态，形成一个  $4 \times 4$  的支付矩阵。支付矩阵的具体内容和求解，以及纳什均衡的求解，也是下阶段需要实现的基础目标。

给定一个时间点  $t$ ，两个士兵的坐标已知，则可以计算出两者之间的距离。距离信息可以转化为子弹命中率和侦察设备效用率的具体值。下阶段通过求解两个子过程的支付矩阵，将其结合成整体效用率的支付矩阵，从而求得系统的纳什均衡。由于行为空间是离散型，不存在高维复杂信息，纳什均衡的求解使用 matlab 等计算工具是可解的。若计算出纳什均衡是纯策略，则直接给出确定解，若得到的是混合策略，则构造相应的分布来模拟混合纳什均衡。

### 3.3.2 技术可行性

通过前期在多智能体强化学习上的理论积累，掌握了基于经典强化学习思想的 MiniMax-Q 和 Friend-or-Foe-Q，基于 FictitiousPlay 等 Markov<sup>[21]</sup> 决策经典算法，以及基于策略梯度的深度方法 MADDPG<sup>[25]</sup> 等，这些算法的伪代码中的 state 和 action 都很清晰，特别是深度 Q-learning 中的 Q 表和 payoff 矩阵之间更容易建立联系。经典的方法可以为深度方法提供博弈论的依据，比如在策略网络中加入基于智能体理性的判断和对抗合作判断都是有助于训练的。近年来最为主流的 Actor-Critic 架构也能将纳什均衡相关的指标加入评判，无论是智能体的理性程度，还是和其他智能体的策略共享参数。

通过博弈论分析的理论基础，首先可以将算法和纳什均衡对抗并训练，比较算法对于给定策略的收敛程度，定量分析迭代次数，最终和纳什均衡的致胜程度也是重要指标。纳什均衡的得出还可以判断对手的理性与否，从而调整策略时遵循最大最小原则或是贪心原则，使战场上的攻防向着自己有利的方向发展。通过传统强化学习的 Minimax 思想和竞争合作思想，优化深度强化学习中的策略梯度，在技术上是可行的。

### 3.3.3 环境与条件可行性

为了构建针对上述问题情境的仿真平台，实验室已有一个完整开发的枪战对抗模拟平台。系统主要基于一个 Agent 的模拟平台，为其加入声波探测和干扰对抗攻防的部分。通过阅读大型军用仿真平台 AFSIM 的文档和 AAAI2018 论文 MAgent 的源码，已经初步掌握了构建多智能体仿真系统的要素、原则和方法，配合实验室的仿真平台，进行多智能体强化学习的算法实验。

系统围绕战场环境和攻防双方展开，攻防双方的活动载体是玩家本身，玩家控制的士兵也作为 Agent 单位，士兵可以通过行走进行在系统中的运动，各自配备侦测和干扰设备。战场环境主要为 Agent 提供信息，维护所有 Agent 的列表，进行全局判定，作为各项奖惩和事件的管理依据。士兵作为场上的 Agent，由行走，攻击和侦察三类行为空间组成。Agent 根据算法标签控制其调用的算法，并接受战场环境给予的分数，作为决策算法的依据。算法部分包括战场参数计算和士兵调用的算法。战场上根据对抗关系的矩阵可以算出混合纳什均衡，作

为决策算法的依据，或者将其策略空间参数的混合比作为一个检验算法的依据。如图 3.7 为平台的初步界面，接口已经调通，可以进行模拟，但是界面仍然需要重构和完善。

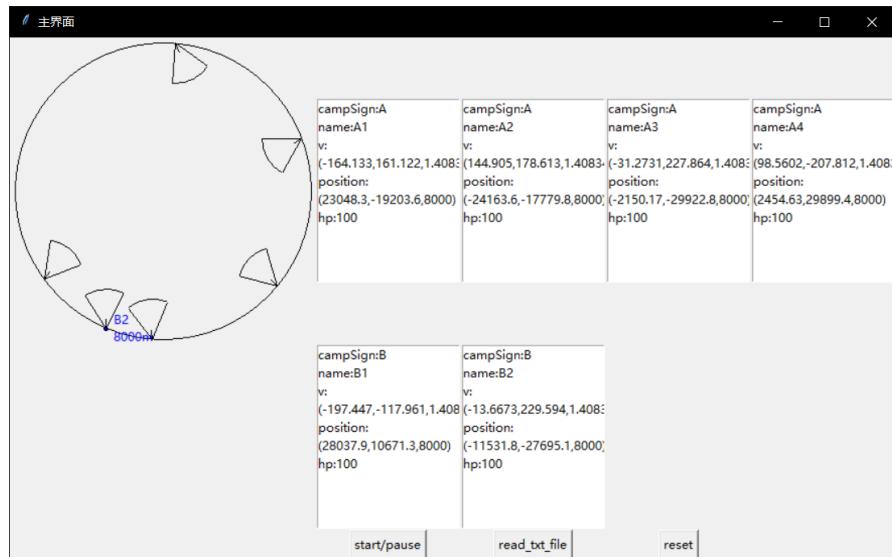


图 3.7: 对抗仿真平台初步示意图

示意图上有一个圆形的战场，以及战场上六个士兵的状态信息，这些可以随机初始化，也可以按照文件的方式读写，在最初验证理论模型的时候，只需要把士兵在战场上任意矢量方向移动调成固定角度速度，探测器和干扰器固定角度即可模拟，后期即使把离散化的行为空间推广到连续性也可以在这个平台上进行验证。

除了本平台外，知名的强化学习平台 OpenAI-gym 中也有大量的小游戏存在和纳什均衡密切相关的子结构，可以将算法快速通过 gym 的工具进行验证，得到更加充分地实验论证。pysc2 中的研究星际争霸小游戏小地图的任务也可以验证算法的优化程度，而实验室自研的另一款人类第一人称射击游戏则可以以人类玩家为基准。从和模型密切相关的飞行器仿真，到集成的小游戏平台和大型游戏，基于博弈论的强化学习都有很好的验证环境和基准。

相比于大型游戏如星际争霸 II 和 Dota2 的 5V5，小游戏的训练和验证不那么依赖图像视觉信息，所以对于计算资源尤其是 GPU 的需求度不高，同时通过设计合理的分布式训练架构也可以提高并行性和训练效率。实验室提供的计算资源足够支撑智能体的训练，自己的电脑也可以承担许多测试的任务。

## 4 研究计划进度安排及预期目标

### 4.1 进度安排

时间节点	安排内容
4.1~4.15	根据前期准备的需求实现测试平台，并开始训练算法
4.16~4.30	算法细节修改和测试，获取评估数据
5.1~5.15	算法在其他平台的移植和测试
5.16~5.31	毕业论文整理和撰写，平台优化和准备答辩
6.1 以后	已有基础的拓展，争取形成会议级别的论文

### 4.2 预期目标

1. 构建完整的战场系统，实现从训练到测试的全过程，提供图形化界面 (GUI) 和命令行测试，界面如图 4.1 所示。
2. 结合传统的 MiniMax-Q 和 Friend-or-Foe-Q 思想，改进 MADDPG 和 PPO 算法，实现混合纳什均衡的 Baseline 训练对象，测试算法的性能，收敛性和致胜程度。按照算法的收敛速度和致胜程度进行对比分析，绘制成图表。
3. 测试算法在 OpenAI-gym 等其他环境的迁移能力，证明算法的有效性
4. 按照如的框架展示成果，包括战场信息的动画和参数面板，最后绘制致胜的曲线和纳什均衡、Q 表等重要图表，参数表如图 4.2 所示。



图 4.1: 战场界面及展示示意图

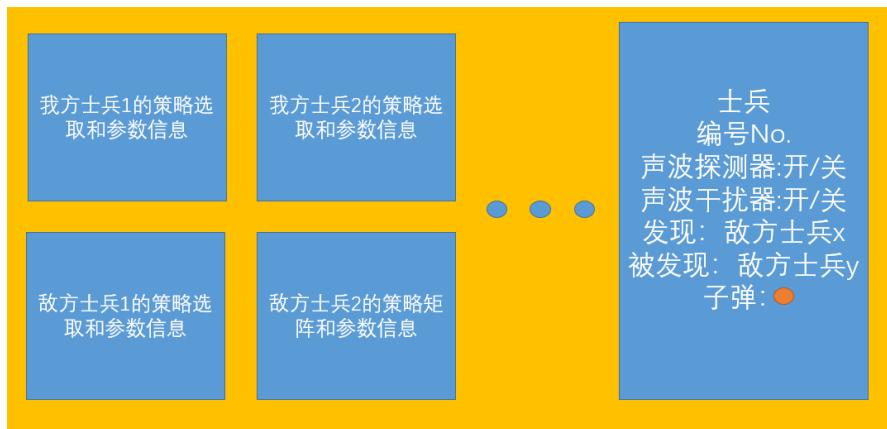


图 4.2: 参数面板展示示意图

## 5 参考文献

- [1] 金玉净. 近似动态规划在资源配置中的应用研究[D]. 苏州大学, 2014.
- [2] 强化学习在多智能体系统任务分配中的应用. [J]. 工业信息学研究室, 2004.
- [3] K T, A N. Evolutionary game theory and multi-agent reinforcement learning[M]. [S.l.]: [s.n.], 2005.
- [4] LEVIN E, PIERACCINI R, ECKERT W. Using Markov decision process for learning dialogue strategies.[C]// IEEE International Conference on Acoustics. [S.l.]: [s.n.], 2002.
- [5] SUTTON R S, BARTO A G. Reinforcement Learning: An Introduction[J]. IEEE Transactions on Neural Networks, 1998, 9(5): 1054–1054.
- [6] YU F R, HE Y. Reinforcement Learning and Deep Reinforcement Learning[M]. [S.l.]: [s.n.], 2019.
- [7] NOWÉ A, BRYS T. A Gentle Introduction to Reinforcement Learning[M]. [S.l.]: [s.n.], 2016.
- [8] KOUBAA A. Robot Operating System (ROS): The Complete Reference (Volume 1)[M]. [S.l.]: [s.n.], 2016.
- [9] NASH J. Non-Cooperative Games[J]. Annals of Mathematics, 1951, 54(2): 286–295.
- [10] BELLMAN R. On a routing problem[J]. Quarterly of Applied Mathematics, 1958, 16(1): 87–90.
- [11] MNIIH V, KAVUKCUOGLU K, SILVER D, et al. Playing Atari with Deep Reinforcement Learning[J]. CoRR, 2013, abs/1312.5602.
- [12] YOUNG H P. The Evolution of Conventions[J]. Econometrica, 1993, 61(1): 57–84.
- [13] BAXTER J, BARTLETT P L, WEAVER L. Experiments with Infinite-Horizon, Policy-Gradient Estimation[J]. J Artificial Intelligence Research, 2001, 15(1): 351–381.

- [14] CAI Q, PAN L, TANG P. Deterministic Policy Gradients With General State Transitions[J]. 2018.
- [15] Actor-Critic. [J]. Encyclopedia of the Sciences of Learning, 2012.
- [16] ADAM S, BUSONIU L, BABUSKA R. Experience Replay for Real-Time Reinforcement Learning Control[J]. IEEE Transactions on Systems Man & Cybernetics Part C, 2012, 42(2): 201–212.
- [17] HESSEL M, MODAYIL J, van HASSELT H, et al. Rainbow: Combining Improvements in Deep Reinforcement Learning[J]. CoRR, 2017, abs/1710.02298.
- [18] PENG B, LI X, GAO J, et al. Adversarial Advantage Actor-Critic Model for Task-Completion Dialogue Policy Learning[J]. 2017.
- [19] BAAZIZADEH M, FROSIO I, TYREE S, et al. Reinforcement Learning through Asynchronous Advantage Actor-Critic on a GPU[J]. 2016.
- [20] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal Policy Optimization Algorithms[J]. 2017.
- [21] LITTMAN M L. Markov games as a framework for multi-agent reinforcement learning[J]. Machine Learning Proceedings, 1994: 157–163.
- [22] LITTMAN M L. Friend-or-Foe Q-learning in General-Sum Games[C]// Eighteenth International Conference on Machine Learning. [S.l.]: [s.n.], 2001.
- [23] KUMAR S, SHAH P, HAKKANI-TUR D, et al. Federated Control with Hierarchical Multi-Agent Deep Reinforcement Learning[J]. 2017.
- [24] LOWER R, YI W, TAMAR A, et al. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments[J]. 2017.
- [25] KILINC O, MONTANA G. Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations[J]. 2018.
- [26] 胡晓峰, 贺筱媛. AlphaGo 的突破与兵棋推演的挑战[J]. 科技导报, 2017, 35(21): 49–60.

- [27] EFRONI Y, DALAL G, SCHERRER B, et al. How to Combine Tree-Search Methods in Reinforcement Learning[J]. 2018.
- [28] SILVER D, SCHRITTWIESER J, SIMONYAN K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676): 354–359.
- [29] BROWN N, SANDHOLM T. Safe and Nested Subgame Solving for Imperfect-Information Games[J]. 2017.
- [30] VINYALS O, EWALDS T, BARTUNOV S, et al. StarCraft II: A New Challenge for Reinforcement Learning[J]. 2017.
- [31] 李鸿, 吴嗣亮, 杨春山. 对策论在雷达反干扰作战中的应用[J]. 现代雷达, 2008, 30(2): 15–17.



### 三、外文翻译

## 摘要

多智能体强化学习的系统旨在为交互的智能体提供互相学习和适应的能力。在很多真实场景的应用中，智能体仅仅能获得世界的局部感知。在这里我们考虑智能体的观察存在严重的噪声，因此只和环境的真实状态有一点微弱的关联。在这样的情形下，学习到一个优化的策略变得尤为具有挑战性，甚至在不实际的情况下智能体的策略会建立在所有其他智能体的观察上。为了克服这些困难，我们提出了一种通讯媒介强化的多智能体的深度确定性策略梯度算法 (MADDPG-M)，实现了一个两层次，连续并发的学习机制。一个智能体的策略依赖他自身的观察以及通过媒介直接共享的其他智能体的观察。在任何给定的时间点，一个智能体必须确定它自身的观察是否足够分享给其他智能体。然而，我们的环境并没有给智能体关于该通讯行为是否有益的直接反馈，而是通讯机制也必须通过经验积累同步进行学习。我们的实验结果分析了我们的算法在六个高度无序和逐渐提高复杂度的环境中有好的表现，并且和基线算法相比可以提供持续的表现提高。

## 1 引言

强化学习 (RL) 关注的是使得智能体学习如何通过采取序列化的行为在一个给定的复杂环境来最大化积累奖励的问题，并且依赖马尔可夫决策过程 (MDP)(Sutton et al., 1998)。这个决策者，或者说智能体，遵循一个定义了在每个环境状态下需要采取哪种行为的策略。在近年来，深度强化学习 (DRL) 通过基于近似函数的深度神经网络来优化强化学习方法，已经被证明在许多应用中达到了人类级别的表现，主要是在游戏环境下，需要一个独立的智能体 (Mnih et al., 2015; Silver et al., 2016)。另一方面，许多实际的应用可以被建模成多智能体系统，比如自动驾驶 (Dresner 和 Stone, 2008)，资源配置 (Jun et al., 2011; Colson et al., 2011)，流量控制 (Stranjak et al., 2008)，轨迹规划 (Bento et al., 2013)，网络包路由 (Di Caro et

al., 1998)。在这些案例中，智能体互相交互并成功完成了重要的任务。直接通过单智能体深度强化学习的方法应用在学习多智能体策略上是不能适应的，原因有很多。首先，从每个单个的智能体角度出发，环境的表现不仅仅依赖单个智能体自身的行为而是所有其他智能体的共同行为，因此表现出一个高度的无序性。这种严重的无序性违反了马尔可夫的假设并且防止了经验回访不假思索地应用，经验回放在稳定训练和改善样本效率上至关重要。此外，每个智能体个体是很难完整和精确地代表整个环境的。典型地，一个智能体仅仅可以获得基于部分环境事实的自身观察。决定哪个智能体应该被基于观察的奖励也是麻烦的。

单独训练每个智能体，因此忽视掉无序的环境，是一个最简单的可能策略。独立的 Q-学习 (IQL)(Tan, 1993) 使用这种方法优化了传统的 Q-学习，并且被报道了一些关键的成功即使有均衡性的问题。Tampuu 在 2017 年扩充了 IQL 通过深度 Q 学习在一个竞争的多智能体设定下打乒乓球，并且 Tesauro 在 2003 年通过允许智能体的策略依赖对其他智能体策略的估计来解决无序的问题。集中训练分散执行 (CTDE) 的方式已经被广泛采纳用来在训练多智能体系统中解决无序问题 (Foerster et al., 2016)。CTDE 使得每个智能体的观察和行为得到提升，并且更好的估计训练时的行为和价值函数。因为每个智能体的策略仅仅依赖他们自己训练时的观察，智能体可以决定在执行的时候分散执行哪个行为。最近，Lowe 在 2017 年把这种方法和 DDPG 算法结合了起来 (Lillicrap et al., 2015) 来解决多智能体的微环境，并且 Foerster 在 2018 年使用类似的方法在一个和事实想法的基线上解决了奖励分配的问题。

在这篇文章中我们考虑了一个特别有挑战性的多智能体学习问题，所有智能体都在部分信息下工作，并关切他们收到的观察都和真实情况相关程度低。这是在真实应用中提出的相似设定，比如在合作和自动驾驶中，当一个智能体在任何时刻观察世界，获得了一定数量的感官信息，可能带有很高程度的不确定性，并且有可能是错的。在这种条件下学习合作解决重要人物变得不可行除非有合适的信息适应机制使得只有准确的观察才能被允许传递并提醒策略。理性上来说，当一个观察是准确的，将其与其他智能体分享会帮助形成一个更好的世界观并且更多好的决定，而不加区分地分享所有信息会因为高程度的噪声有害。为了保持这种设定的现实性，我们假设智能体不被直接告诉他们的观察是准确或是噪声，他们需要通过经验去发掘这一点。

我们提出了一种通过通讯媒介强化的多智能体深度确定性策略梯度算法来解决这些需求。在训练过程中，每个智能体的策略不仅仅依赖自身，还依赖其他智能体直接共享的观察。每个智能体自发学习它当前的自身观察是否对于最大化未来预期的奖励有贡献，因此值得在给定的时间分享给其他智能体，共同学习重要的任务。扩展性的实验结果分析了 MADDPG-M 在高度无序环境下的良好表现，甚至当智能体在同一时间段需要连续改变相关观察的情况。因为该执行的时候是用分散的方式，算法在每个时间片的复杂度在智能体的数目上是线性的。为了评估 MADDPG-M 的表现，我们设计并测试了许多环境作为原先合作探索问题 (Lowe et al., 2017) 的扩充。每个智能体是一个 2D 对象旨在达到一个不同的地标的同时避免和其他智能体的相撞。我们考虑了两种设定，在第一个中，一个单独的优秀智能体可以看到所有地标的真实位置，而其他智能体收到错误的信息。在第二个设定下，智能体只有对于其他智能体有益的信息，并且必须被重定到正确的接受处。我们比较了 MADDPG-M 和所有环境中可能存在的基线算法并讨论了它们的相对优势和可能的未来改进。

## 2 相关工作

多智能体深度确定性策略梯度 (MADDPG) (Lowe et al., 2017) 采纳了 CTDE 的方法并且在 DDPG 的基础上建立了多智能体的方法来解决合作对抗混合的环境。直接的通讯是不允许的。相反，集中训练帮助智能体学习了协同的行为。CTDE 的样式在多智能体微环境和星际争霸单元微管理中证明有良好表现 (Foerster et al., 2018)。进一步的工作已经集中在使得智能体之间的通信更有效上。一大批潜在的方法通过引入不同的使得梯度可以在智能体间传递的通道使得通信有效。比如，在差异智能体间学习 (DIAL) (Foerster et al., 2016)，智能体通过它们的行为进行离散化通信。DIAL 使用了基于深度循环 Q 网络和权重共享的 Q 学习 (DRQN) (Hausknecht 和 Stone, 2015)。这个算法在智能体间端到端可训练因为它在智能体之间通过消息传递梯度的能力，需要随着智能体数目平方增长的通信媒介。这个方法被用来解决例如交换仅需 1 个比特信息的沟通就足够的问题。和 DIAL 类似，通讯神经网络 (CommNet) (Sukhbaatar et al., 2016) 使用一个智能体间参数共享的策略并且定义了一个差异信道。每个智能体可以通过共享的信道去携带所

有智能体间的平均信息。CommNet 使用了一个大的单个网络给所有智能体，这个网络不是简单可扩展的。多智能体双向协同网络 (BiCNet) (Peng et al., 2017) 采取了一个基于循环神经网络的内存来构建一个智能体间的信道，并用一个集中控制的策略来定义真实的条件。其他作者也在语多智能体系统的语言上进行了研究 (Lazaridou et al., 2016; Mordatch 和 Abbeel, 2018; Lazaridou et al., 2018)。在这些工作上，环境典型地位环境的行为作出反馈。和现存的工作不同，我们认为直接知道通信行为的奖励反馈是不存在的。我们的问题需要一个通信机制和本地智能体策略的层级管理，这些必须被持续学习。此外，我们认为观察既可能是错的也可能是随机的，所以必须有一个合适的通信机制才能学习到优化的策略。

### 3 研究背景

#### 3.1 部分可见的马尔可夫游戏

部分可见的马尔可夫游戏 (POMGs) (Littman, 1994) 是由  $N$  个部分可见智能体的智能体的多智能体延伸并且被标记了一个真实状态集合  $\mathcal{S}$ ，一个行为集合  $\mathcal{A} = \{\mathcal{A}_1, \dots, \mathcal{A}_N\}$ ，一个状态转移函数  $\mathcal{T}$ ，一个奖励函数  $\mathcal{R}$ ，一个自身观察函数  $\mathcal{Q} = \{\mathcal{Q}_1, \dots, \mathcal{Q}_N\}$ ，一个自身观察集合  $\mathcal{O} = \{\mathcal{O}_1, \dots, \mathcal{O}_N\}$  以及一个折扣系数  $\gamma \in [0, 1]$ 。一个 POMG 被一个元组定义， $G = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Q}, \mathcal{O}, \gamma, N \rangle$ 。智能体没有完整的环境真实状态的集合  $s \in S$ ，而是获得一个相比于真实状态的部分认识，例如  $o_i = \mathcal{Q}_i(s) : \mathcal{S} \rightarrow \mathcal{O}_i$ 。每个智能体根据一个确定和随机的策略参数  $\theta_i$  来选择行为并且控制它自己的认知，比如  $a_i = \mu_{\theta_i}(o_i) : \mathcal{O}_i \rightarrow \mathcal{A}_i$  或  $\pi_{\theta_i}(a_i | o_i) : \mathcal{O}_i \times \mathcal{A}_i \rightarrow [0, 1]$ ，并且从行为的结果中获得一个奖励，例如  $r_i = \mathcal{R}(s, a_i) : \mathcal{S} \times \mathcal{A}_i \rightarrow \mathbb{R}$ 。环境进入下一个状态  $s' \in \mathcal{S}$  根据控制所有智能体行为的状态转移函数，例如  $s' = \mathcal{T}(s, a_1, \dots, a_N) : \mathcal{S} \times \mathcal{A}_1 \times \dots \times \mathcal{A}_N \rightarrow \mathcal{S}$ ，每个智能体都希望最大化它的期望的反馈， $\mathbb{E}[R_i] = E\left[\sum_{t=0}^T \gamma^t r_i^t\right]$  并且  $r_i^t$  是第  $i$  个智能体在时间  $t$  的奖励， $T$  是时间界限。

### 3.2 确定性策略梯度算法

策略梯度 (PG) 算法是基于更新策略参数向量  $\theta$  在方向  $\nabla_{\theta} J(\theta)$  最大化对象函数,  $J(\theta) = \mathbb{E}[R]$ 。当前的策略  $\pi_{\theta}$  被一个使用一系列行动空间上可能性测试, 例如  $\pi_{\theta} : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$  确定性策略梯度 (DPG)(Silver et al., 2014) 扩展了策略梯度架构, 通过采取策略函数将状态映射到行为, 例如  $\mu_{\theta} : \mathcal{S} \rightarrow \mathcal{A}$ 。梯度被用来优化对象 objective  $J(\theta)$ :

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{s \sim \mathcal{D}} \left[ \nabla_{\theta} \mu_{\theta}(a|s) \nabla_a Q^{\mu}(s, a) \Big|_{a=\mu_{\theta}(s)} \right] \quad (3-1)$$

其中  $D$  是一个经验重放的缓存并且  $Q^{\mu}(s, a)$  是一个对应的行为价值函数。在 DPG 上, 策略梯度采用了在状态空间上的期望, 引入了数据有效性的优势。另一方面, 策略梯度同样依赖于  $\nabla_a Q^{\mu}(s, a)$ , DPG 需要一个连续的策略  $\mu$ 。深度确定性策略梯度 (DDPG) (Lillicrap et al., 2015) 在 DPG 上建立, 采用了深度神经网络来近似  $\mu$  和  $Q^{\mu}$ 。相较于 DQN(Mnih et al., 2015), 经验重放的缓冲  $\mathcal{D}$  和目标网络  $\mu'$ ,  $Q^{\mu}$  帮助稳定了学习。

### 3.3 多智能体深度确定性策略梯度

多智能体深度确定性策略梯度 (MADDPG) 从 DDPG 扩充到多智能体的设定, 通过采取 CTDE 的方法。DDPG 对于延伸非常适合因为  $\mu$  和  $Q^{\mu}$  可以被依赖在外部信息上。为了防止不稳定的情况, MADDPG 使用所有智能体的行为和认知在行为价值函数中,  $Q_i^{\mu}(o_1, a_1, \dots, o_N, a_N)$ , 另一方面, 因为一个智能体的策略只根据自身的认知来控制,  $a_i = \mu_{\theta_i}(o_i)$ , 智能体可以在执行的是后用一个分开的规则。这个连续性策略  $\mu_{\theta_i}$  的梯度 (这里之后被命名为  $\mu_i$ ), 对于参数  $\theta_i$ :

$$\nabla_{\theta_i} J(\mu_i) = \mathbb{E}_{o, a \sim \mathcal{D}} \left[ \nabla_{\theta_i} \mu_i(a_i | o_i) \nabla_{a_i} Q_i^{\mu}(o_1, a_1, \dots, o_N, a_N) \Big|_{a_i=\mu_i(o_i)} \right] \quad (3-2)$$

对于第  $i$  个智能体, 它的集中  $Q_i^{\mu}$  函数被更新来最小化它基于暂时区分的损失函数:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{o, a, r, o'} \left[ (Q_i^{\mu}(o_1, a_1, \dots, o_N, a_N) - y)^2 \right] \quad (3-3)$$

其中  $y = r_i + \gamma Q_i^{\mu'}(o'_1, a'_1, \dots, o'_N, a'_N) \Big|_{a'_j=\mu'_j(o'_j)}$ 。这里  $\mu'_i$  是目标策略, 参数  $\theta'_i$  暂时用  $\theta_i$  更新, 并且  $\mathcal{D}$  是由元组  $(o, a, r, o')$  组成的经验回放缓存, 每个元素是一个大小为  $N$  的集合, 例如:  $o = \{o_1, \dots, o_N\}$

### 3.4 内部驱动的强化学习和分层 DQN

内部驱动的学习已经在 RL 理论上被深入研究了 (Singh et al., 2004; Schmidhuber, 1991); 然而, 如何设计一个好的内部奖励函数依然是个开放性的问题, 存在的和不同的内部奖励相关的技巧密切相关。总的来说, 一个本质的奖励被认为是一个代表被访问状态的探索奖励。换句话说, 本质奖励鼓励智能体在外部奖励的状态空间进行搜索, 这可以是一个状态访问计数的递减函数 (Bellemare et al., 2016; Ostrovski et al., 2017)。

2016 年 Kulkarni 介绍了一种相关的本质奖励来训练一个两级的分层 DQN 模型, 在这个模型中, 在顶层, 智能体学习了策略  $\pi_g$  来选择一个本质的目标  $g \in \mathcal{G}$ , 例如  $\pi_g = P(g|s)$ 。这本质目标被用来在底层作用, 这里智能体学习  $\pi_a$  来决定行为, 比如  $\pi_a = P(a|s, g)$ 。在这个设定下, 顶层和底层的策略分别被外部和内部的奖励

## 4 基于通讯媒介的 MADDPG

### 4.1 问题推导和提出的方法

我们考虑部分认知的马尔可夫游戏, 并且假设大多数智能体接受的认知是极度干扰或者和真实状态联系很少的, 使得学习优化策略不可行。在所有的  $N$  个认识中考虑每个策略, 例如  $a_i = \mu_i(o_1, \dots, o_N)$ , 这对于大部分的  $O_i$  和对应的  $s$  没有关联, 比如, 他们提供了一个对于第  $i$  个智能体的当前真实状态有一个很差的表现。为了解决这个挑战, 我们让每个智能体的策略依赖他自己的认知以及其他智能体直接共享。因为智能体不能再相关信息和噪声中进行区分, 所以决策是否分享他们的认知的能力必须通过经验进行学习。更加正式的, 我们引入了两层的策略集,  $\nu = \{\nu_1, \dots, \nu_N\}$  和  $\mu = \{\mu_1, \dots, \mu_N\}$ , 这些通过一个通信的媒介  $m = \{m_1, \dots, m_N\}$  整合在一起, 这里  $m_i$  表示第  $i$  个智能体共享的信息。在  $\mu$  里的行为策略决定了智能体采取的和环境交互的行为, 这里  $\mathbf{U}$  里的交流策略空置了决定和媒介共享的信息的行为。在这个顶层, 每个智能体选择一个通信的行为  $c_i$ , 在它自身认知的控制下, 例如,  $c_i = \nu_i(o_i)$ 。

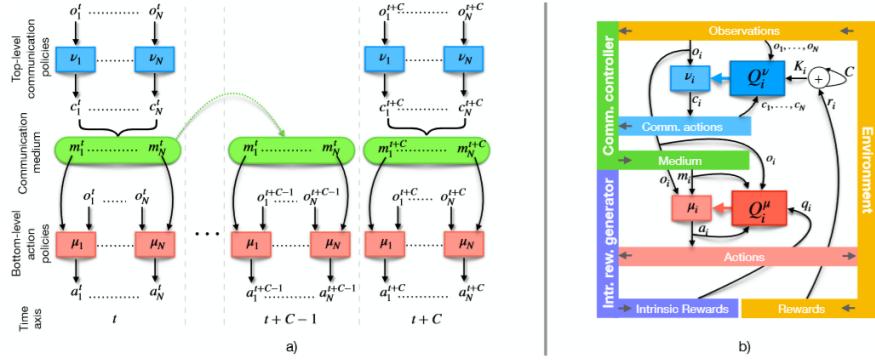


图 4.1: 对于 MADDPG-M 的综述。在 (a) 图中,  $N$  个智能体在两层策略集中, 通过一个通信媒介相连。在训练的时候, 我们在一个很短的时间间隔里运行通信策略, 比如, 在每  $C$  步的行为决策力, 使用固定的通信媒介来决定环境行为。在 (b) 图里, 通信的策略通过一个 CTDE 的方法来整合奖励的和, 这些奖励是环境在这  $C$  步里收集的, 而在通信媒介方面我们通过一个内部的奖励来分别学习行为策略。

我们考虑这两种可能的通信机制: 广播 (一对多) 和单播 (一对一)。在广播的情况下, 每个通信行为都是一个矢量, 此时  $c_j \in \mathbb{R} | 0 \leq c_j \leq 1$  并且最大通信行为的智能体认知就会被送到所有的智能体那里, 在这个情况下  $m$  被定义为:

$$m = \left\{ m_i = o_k \quad \forall i \in \{1, \dots, N\} \quad \text{where } k = \arg \max_j (c_1, \dots, c_j, \dots, c_N) \right\} \quad (4-1)$$

在单播的情况下, 通信的行为是一个  $N$  维的向量, 例如  $c_{j,:} \in \mathbb{R}^{1 \times N} | 0 \leq c_{j,i} \leq 1$ , 这里  $c_{j,i}$  被认为是对于第  $j$  个智能体愿意分享它的认识给第  $i$  个智能体的乐意程度因此对于智能体最乐意的和第  $i$  个智能体, 比如, 当被任命到  $m_i$ 。在这种情况下,  $m$  被定义为:

$$m = \left\{ m_i = o_k \quad \text{where } k = \arg \max_j (c_{1,i}, \dots, c_{j,i}, \dots, c_{N,i}) \right\} \quad (4-2)$$

在底层, 探索被分享的信息, 每个智能体决定她的环境行为, 例如,  $a_i = \mu_i(o_i, m_i)$ 。

## 4.2 学习算法

这两类策略,  $\mathbf{U}$  和  $\mu$ , 被配对并且被并行训练。为了解决这个问题, 我们使用了两种不同级别的缓存来收集训练时环境的转移; 换句话来说,  $\mathbf{U}$  和  $\mu$  在不

相同的时间范围内运行，通信的行为在每  $C$  步的时候进行。在这个阶段，媒介的状态被固定，比如， $m^t = m^{t+1} = \dots = m^{t+C-1}$ ，并且环境没有直接奖励好的通信方法，没有明显的优化通信策略的方法。相反，我们使用了外部奖励的聚合在这  $C$  步，比如， $K = \sum_{t'=t}^{t+C} r^{t'}$ ，在每一步的时候，我们也生成内部的奖励，来应对环境的行为并使用优化因子  $\mu$ 。

在强化学习研究中，内部奖励在探索的时候是被应用最多的。相反，在这个工作中，内部奖励被用来使得智能体学习环境的动态信息甚至在顶层的通信决定并不哟话的时候。在任务中我们考虑，环境的外部奖励代表了智能体和他们真正目标之间的距离甚至当智能体并没有真正发现它们的目标。在智能体不能看到真正的目标，他们不能通过学习来到达它们因为它们的认知和接受的奖励变得不相关。另一方面，我们引入的内部奖励代表了智能体和在媒介中的目标，不管是否这些是真实的或是干扰的。通过这个，在底层智能体学习如何到达媒介共享的目标。在顶层，使用累计的外部奖励，这些指代了认知更好地代表了真正的目标并且应该在媒介中进行共享，我们的发展受到 Kulkarni 在 2016 年工作的启发，那里的内部奖励被用来在单智能体系统中来协助探索。这里我们引入内部目标的概念，被保持到智能体到达它们为止，以及通信媒介  $m$  的概念。

我们保持了两个独立的经验回放缓存， $\mathcal{D}_\nu$  和  $\mathcal{D}_\mu$ ，其中  $\mathcal{D}_\nu$  由元组  $(o, c, K, o'')$  构成，这里  $O''$  代表了第  $C$  个认知在  $o$  之后，例如， $o'' = o^{t+C}$ ，并且提供了用来更新通信策略  $\nu$  的样本。另一方面， $\mathcal{D}_\mu$  由元组  $(o, m, a, q, o')$  组成，这里  $O'$  代表了  $o$  以后的下一个认知，比如， $O' = o^{t+1}$ ，提供了更新行为策略  $\mu$  的样本，我们引入 actor-critic 策略梯度模型对于  $\nu$  和  $\mu$ ，并且训练通信策略  $\nu$  通过一个 CTDE 地方法。对于一个智能体，策略梯度总是相对于  $\theta_{\nu,i}$  被写作：

$$\nabla_{\theta_{\nu,i}} J(\nu_i) = \mathbb{E}_{o,c \sim \mathcal{D}_\nu} \left[ \nabla_{\theta_{\nu,i}} \nu_i(c_i | o_i) \nabla_{c_i} Q_i^\nu(o_1, c_1, \dots, o_N, c_N) \Big|_{c_i=\nu_i(o_i)} \right] \quad (4-3)$$

对应的集中行为价值函数  $Q_i^\nu$  用来更新最小化如下的损失函数，基于暂时的差异：

$$\mathcal{L}(\theta_{\nu,i}) = \mathbb{E}_{o,c,K,o''} [(Q_i^\nu(o_1, c_1, \dots, o_N, c_N) - y)^2] \quad (4-4)$$

这里的  $y = K_i + \gamma Q_i^{\nu'}(o_1'', c_1'', \dots, o_N'', c_N'')$  并且  $\nu'_i$  是目标策略，参数  $\theta'_{\nu,i}$  暂时通过  $\theta_{\nu,i}$  来更新。这样的的层的行为梯度  $\mu$  不仅超过了自身的认知，而

且超过了  $m$ 。这个相对于参数  $\theta_{\mu,i}$  的策略梯度变成了：

$$\nabla_{\theta_{\mu,i}} J(\boldsymbol{\mu}_i) = \mathbb{E}_{o,m,a \sim \mathcal{D}_\mu} \left[ \nabla_{\theta_{\mu,i}} \boldsymbol{\mu}_i(a_i|o_i, m_i) \nabla_{a_i} Q_i^\mu(o_i, m_i, a_i)|_{a_i=\boldsymbol{\mu}_i(o_i, m_i)} \right] \quad (4-5)$$

和通信的策略不同，行为的策略被用一个分散的方式训练，对应的第  $i$  个智能体行为价值函数  $Q_i^\mu$  通过最小化如下的损失函数来更新，基于暂时的差异：

$$\mathcal{L}(\theta_{\mu,i}) = \mathbb{E}_{o,m,a,q,o'} \left[ (Q_i^\mu(o_i, m_i, a_i) - y)^2 \right]$$

这里的  $y = q_i + \gamma Q_i^\mu(o'_i, m_i, a'_i)|_{a'_i=\boldsymbol{\mu}'_i(o'_i, m_i)}$  和  $\boldsymbol{\mu}'_i$  是目标策略，其参数为  $\theta'_{\mu,i}$ ，会在这期间被  $\theta_{\mu,i}$  更新。对于图 1 提出的方法的分析和算法的伪代码会在附录 A 中提供。

## 5 实验

### 5.1 环境

在这个部分我们介绍合作导航这个问题的六个不同的变量，从多智能体微环境角度。在它原先的版本里， $N$  个智能体需要到达  $N$  个目的地并且避免互相相撞。每个智能体观察其他  $N-1$  个智能体的相对位置和  $N$  个目的地。智能体通过到目的地的距离来相对奖励。和大部分的现实多只 I 能提用例不一样，每个智能体能获得几乎真实环境状态的完整信息。像这样，独立训练的智能体可以达到那些集中训练的效果（附录中有支撑的证据）。因此，在它原先的版本，完全没有智能体间交流的需求。我们现在对环境的修改更加接近发现现实应用的复杂程度。我们根据解决任务所需要的通讯种类把方案分成了两个组。

在第一组，只有一个智能体，有天赋的智能体，可以考到目的地的正确位置。这个特殊的智能体既可以在同一个训练的时间段保持同一个也可以在不同时期改变，或者在同一时期内改变。其他的智能体都收到不准确的目的地位置。关键的是，没有智能体知道他们的地位，因此他们都必须通过与环境的交互去学习他们的认知是否帮助整体策略的改进，并且值得被分享给其他人。这种学习的任务通过在任何给定时间是否需要自发通过广播通讯机制传递给其他智能体自己的认知来实现。只有环境的间接反馈，即奖励函数才可以告诉他们当前的方法提高了他们的策略水平。这些人物包括三个不同递增复杂度的变量，依赖于如何定义带天赋的智能体，在修改的安利中，带天赋的智能体保持在训练时期一致。比

如，正确的目的地总是被同一个智能体观察到；在改变的情况下，带天赋的智能体在每个阶段会改变，发现正确的目的地的能力被随机赋予场上的一个智能体并且用一个旗子在他们的认知空间中。动态的情况下，最接近环境中心的智能体在每个时间点成为了带天赋的智能体，通过彼此间的相对距离，智能体需要直接理解哪个是最靠近中心的。

第二组环境被设定每个目的地已经被安排给了一个特定的智能体，一个智能体需要占据他分配的目的地来收集奖励。在这种方案下，每个智能体只能正确认识到一目的地的位置，而其他被误导，智能体不知道他们的真正地位（不知道哪个地标），并且必须通过经验学习如何直接分享策略信息以此来最大化期望的奖励。在这个设定下，一个智能体可以决定去发送它的认知，在任何给定的时间，给哪一个剩余  $N-1$  个智能体发送通过一个不广播的机制，也可以在(5)式中看到。在这个群体中我们也看到了三个不同的复杂度递增的变量在于正确的基于改变的认知。在训练的时候固定，在阶段之间改变并且在每个阶段动态根据智能体到环境中心的距离。在这 6 个方案中，经管外部的奖励代表了到真实地标的距离，内部奖励是在媒介中共享的到地标的距离，不考虑是否是真实的地标。

## 5.2 与基线算法的比较

最初，我们通过广播的方式来测试第一组。图 3-a 展示了 MADDPG-M 的学习曲线并且所有在交换广播的方案上的基线算法。在这种情况下，只有 DDPG 和 MADDPG 没有学习到正确的行为，在给定的不可以共享信息的条件下这是符合预期的结果。这样的差表现强化了我们的观点，学习一个对应的行为通过集中式训练在某些情形下是不适用的，这种级别的表现提供了一个我们实验的下限。在时间中，当使用了这些基线以后，我们观察到智能体仅仅向着环境的中心移动了，甚至带天赋的智能体不能学到一个理性的行为，因为奖励的信号变得噪声化因为智能体随意的行为。在另一方面，DDPG-OC 实现的方案解释了，当  $m$  没有被正确控制，智能体在分散训练的情况下甚至依然达到了要求。有趣的是，即使在一个简单的固定的案例达到了满意的程度，Meta-agent 的表现依然明显下降，因为我们环境的复杂度增加了，并且这个算法没有解决动态的情况。

相反，MADDPG-M 允许智能体自发地学习重要的通讯方式以及优化的行为

策略，并且在我们的环境中和 DDPG-OC 表现地很像。为了专门行动和专门通信的表现，图 3-b 展示了收集到的内部奖励以及通信准确率的表现，在 MADDPG 相对于其他 DDPG-OC 的优化实现中的动态广播方案。在训练的开始阶段，即使通信策略没有被足够优化，MADDPG-M 的智能体依然可以学到动态环境以及预期内部奖励的行为。优化的环境行为提供了更好的反馈来优化通信的行为相比于 DDPG-OC。同时，在环境变得更加复杂的时候通信变得更加困难。但是，甚至多数的广播方案中有复杂的设定，MADDPG-M 的智能体选择了优化通信率达到了 0.8882，足够来完成任务。非常相似的结论也在研究非广播式的方案中得出了。因为增加的复杂度，智能体在所有的基线中都比他们的对手收到了相对更少的奖励，相对于广播的方案。MADDPG-M 可以到达一个接近 DDPG-OC 上限的效果，值得一提的观测到的变量在不广播的方案中比广播的方案中相对更高，由于更高的通信需求和任务复杂度（每个智能体需要移动到环境的对面，会导致冲突）。这也会解释为什么 DDPG-OC 有更高的方差在使用优化通信的时候。有趣的是，在动态非广播的方案中，MADDPG-M 的智能体只能找到总体最优的通信方式 0.2898. 然而，因为该独立的通信行为总共有 0.6423 是准确的，他们可以尝试去实现这个任务，更多的内容可以在附录中看到证明。

## 6 结论

在这篇论文中我们研究了一个多智能体强化学习的算法用来解决在部分极度噪声的认知问题。我们考虑了问题的两个例子，一个是大部分智能体接受到错误的信息，以及智能体需要分辨随机分配给智能体的信息。在两个例子中，我们都证明了学习在什么时候发送什么给环境是一个必要的技巧需要来发展出一个相对的行为来实现重要的任务。我们提出的 MADDPG-M 的算法使得连续的优化策略的学习和重要任务。有效地，智能体学习了分享逐步提高奖励的方法。最重要的技术贡献是通信依赖的层级译码。智能体学习到了两种和通信媒介的策略。为了训练这些策略，我们使用了不同级别的抽象条件和探索内部奖励根据状态。在我们的研究中，我们考虑了分享单独认知足够完成任务的方案。当智能体需要同时获得其他智能体认知的时候会变得更加复杂。另外，和分享原来高维且还有冗余信息的观察不同，它可以学到一个更有用的表示方法。

## 7 参考文献

[1] Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1471-1479, 2016.

[2] José Bento, Nate Derbinsky, Javier Alonso-Mora, and Jonathan S. Yedidia. A message-passing algorithm for multi-agent trajectory planning. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pp. 521-529, 2013.

[3] C.M. Colson, M.H. Nehrir, and R.W. Gunderson. Multi-agent microgrid power management. *IFAC Proceedings Volumes*, 44(1):3678 - 3683, 2011. ISSN 1474-6670. doi: <https://doi.org/10.3182/20110828-6-IT-1002.01188>. 18th IFAC World Congress.

[4] Gianni Di Caro, Marco Dorigo, et al. An adaptive multi-agent routing algorithm inspired by ants behavior. In *Proceedings of PART98-5th Annual Australasian Conference on Parallel and RealTime Systems*, pp. 261-272, 1998.

[5] Kurt M. Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *J. Artif. Intell. Res.*, 31:591–656, 2008. doi: 10.1613/jair.2502.

[6] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2137-2145, 2016.

[7] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

[8] Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR*, abs/1507.06527, 2015.

[9] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with

gumbel-softmax.*CoRR*,abs/1611.01144, 2016. URL <http://arxiv.org/abs/1611.01144>.

[10]Zeng Jun, Liu Junfeng, Wu Jie, and H.W. Ngan. A multi-agent solution to energy management in hybrid renewable energy generation system.*Renewable Energy*, 36(5):1352–1363, 2011. ISSN 0960-1481. doi: <https://doi.org/10.1016/j.renene.2010.11.032>.

[11]Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3675–3683, 2016.

[12]Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language.*CoRR*,abs/1612.07182, 2016.

[13]Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. Emergence of linguistic communication from referential games with symbolic and pixel input. *arXiv preprint arXiv:1804.03984*, 2018.

[14]Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR* abs/1612.07182, 2016.

[15]Long Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992. doi: 10.1007/BF00992699.

[16]Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, July 10-13, 1994*, pp. 157–163, 1994.



## 四、外文原文

---

# Multi-agent Deep Reinforcement Learning with Extremely Noisy Observations

---

**Ozsel Kilinc**  
WMG

University of Warwick  
Coventry, UK CV4 7AL  
[ozsel.kilinc@warwick.ac.uk](mailto:ozsel.kilinc@warwick.ac.uk)

**Giovanni Montana**  
WMG

University of Warwick  
Coventry, UK CV4 7AL  
[g.montana@warwick.ac.uk](mailto:g.montana@warwick.ac.uk)

## Abstract

Multi-agent reinforcement learning systems aim to provide interacting agents with the ability to collaboratively learn and adapt to the behaviour of other agents. In many real-world applications, the agents can only acquire a partial view of the world. Here we consider a setting whereby most agents' observations are also extremely noisy, hence only weakly correlated to the true state of the environment. Under these circumstances, learning an optimal policy becomes particularly challenging, even in the unrealistic case that an agent's policy can be made conditional upon all other agents' observations. To overcome these difficulties, we propose a multi-agent deep deterministic policy gradient algorithm enhanced by a communication medium (MADDPG-M), which implements a two-level, concurrent learning mechanism. An agent's policy depends on its own private observations as well as those explicitly shared by others through a communication medium. At any given point in time, an agent must decide whether its private observations are sufficiently informative to be shared with others. However, our environments provide no explicit feedback informing an agent whether a communication action is beneficial, rather the communication policies must also be learned through experience concurrently to the main policies. Our experimental results demonstrate that the algorithm performs well in six highly non-stationary environments of progressively higher complexity, and offers substantial performance gains compared to the baselines.

## 1 Introduction

Reinforcement Learning (RL) is concerned with enabling agents to learn how to accomplish a task by taking sequential actions in a given stochastic environment so as to maximise some notion of cumulative reward, and relies on Markov Decision Processes (MDP) (Sutton et al., 1998). The decision maker, or agent, follows a policy defining which actions should be chosen under each environmental state. In recent years, Deep Reinforcement Learning (DRL), which leverages RL approaches using Deep Neural Network based function approximators, has been proved to achieve human-level performance in a number of applications, mostly gaming environments, requiring an individual agent (Mnih et al., 2015; Silver et al., 2016). Many real-world applications, on the other hand, can be modelled as multi-agent systems, e.g. autonomous driving (Dresner & Stone, 2008), energy management (Jun et al., 2011; Colson et al., 2011), fleet control (Stranjak et al., 2008), trajectory planning (Bento et al., 2013), and network packet routing (Di Caro et al., 1998). In such cases, the agents interact with each other to successfully accomplish the underlying task. Straightforward applications of single-agent DRL methodologies for learning a multi-agent policy are not well suited for a number of reasons. Firstly, from the point of view of an individual agent, the environment behaves in a highly non-stationary manner as it now depends not only on the agent's own actions, but also on the joint action of all other agents (Lowe et al., 2017). The severe non-stationarity

Deep Reinforcement Learning Workshop, NIPS 2018, Montréal, Canada.

violates the Markov assumption and prevents the naive use of experience replay (Lin, 1992), which is important to stabilise training and improve sample efficiency. Furthermore, only rarely each individual agent has a complete and accurate representation of the entire environment. Typically, an agent receives its own private observations providing only a partial view of the true state of the world. Determining which agent should be credited for the observed reward is also non-trivial (Foerster et al., 2018).

Training each agent independently, thus effectively ignoring the non-stationarity, is the simplest possible approach. Independent Q-Learning (IQL) (Tan, 1993) leverages traditional Q-learning in this fashion, and some empirical success has been reported (Maignon et al., 2012) despite convergence issues. Tampuu et al. (2017) has extended IQL for Deep Q-learning to play Pong in a competitive multi-agent setting, and Tesauro (2003) has addressed the non-stationarity problem by allowing each agent's policy to depend upon the estimates of the other agents' policies. The *Centralised Training Decentralised Execution* (CTDE) paradigm has been widely adopted to overcome the non-stationarity problem when training multi-agent systems (Foerster et al., 2016). CTDE enables to leverage the observations of each agent, as well as their actions, to better estimate the action-value functions during training. As the policy of each agent only depends on its own private observations during training, the agents are able to decide which actions to take in a decentralised manner during execution. Recently, Lowe et al. (2017) have combined this paradigm with a Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al., 2015) to solve Multi-agent Particle Environments, and Foerster et al. (2018) have used a similar approach integrated with a counter-factual baseline to address the credit-assignment problem.

In this article we consider a particularly challenging multi-agent learning problem whereby all agents operate under partial information, and the observations they receive are weakly correlated to the true state of the environment. Similar settings arise in real-world applications, e.g. in cooperative and autonomous driving, when an agent's view of the world at any given time, obtained through a number of sensors, may carry a high degree of uncertainty and can occasionally be wrong. Learning to collaboratively solve the underlying task under these conditions becomes unfeasible unless an appropriate information filtering mechanism is in place allowing only the accurate observations to be shared across agents and inform their policies. The rationale is that, when an observation is accurate, sharing it with others will progressively contribute to form a better view of the world and make more educated decisions, whereas indiscriminately sharing of all the information can be detrimental due to the high level of noise. To keep the setting realistic, we do not assume that the agents are explicitly told whether their private observations are accurate or noisy, rather they need to discover this through experience.

We propose a multi-agent deep deterministic policy gradient algorithm enhanced by a communication medium (MADDPG-M) to address these requirements. During training, each agent's policy depends on its own observations as well as those explicitly shared by other agents; every agent simultaneously learns whether its current private observations contribute to maximising future expected rewards, and therefore are worth sharing with others at a given time, whilst also collaboratively learning the underlying task. Extensive experimental results demonstrate that MADDPG-M performs well in highly non-stationary environments, even when the agents acquiring relevant observations continuously change within an episode. As the execution operates in a decentralised manner, the algorithm complexity per time-step is linear in the number of agents. In order to assess the performance of MADDPG-M, we have designed and tested a number of environments as extensions of the original *Cooperative Navigation* problem (Lowe et al., 2017). Each agent is a 2D object aiming to reach a different landmark while avoiding collisions with other agents. We consider two types of settings. In the first one, a single "gifted" agent can see the true location of all the landmarks, whilst the other agents receive their wrong whereabouts. In the second one, the agents may have information that is only valuable to other agents, and has to be redirected to the right recipient. We compare MADDPG-M against existing baselines on all the environments and discuss its relative merits and potential future improvements.

## 2 Related Work

Multi-agent Deep Deterministic Policy Gradient (MADDPG) (Lowe et al., 2017) adopts the CTDE paradigm and builds a multi-agent approach upon DDPG (Lillicrap et al., 2015) to solve mixed cooperative-competitive environments. No explicit communication is allowed. Instead, centralised

training helps agents learn a coordinated behaviour. The CTDE paradigm has been shown to work well on Multi-agent Particle Environments (Lowe et al., 2017) and *StarCraft unit micromanagement* (Foerster et al., 2018). Extensive efforts have also been spent towards enabling communication amongst agents. The large majority of existing methods enable communication by introducing a differential channel through which the gradients can be sent across agents. For instance, in Differentiable Inter-agent Learning (DIAL) (Foerster et al., 2016), the agents communicate in a discrete manner through their actions. DIAL uses Q-learning based Deep Recurrent Q-Networks (DRQN) (Hausknecht & Stone, 2015) with weight sharing. The algorithm is end-to-end trainable across agents due to its ability to pass gradients from agent to agent over the messages, which requires a communication medium scaling quadratically with the number of agents. This approach has been used to solve problems such as switch riddle where communicating over 1-bit messages is sufficient. Analogously to DIAL, Communication Neural Net (CommNet) (Sukhbaatar et al., 2016) uses a parameter-sharing strategy across agents and defines a differentiable communication channel between them. Every agent has access to this shared channel carrying the average of the messages of all agents. CommNet uses a large single network for all the agents, which may not be easily scalable. Multi-agent Bidirectionally-Coordinated Network (BiCNet) (Peng et al., 2017) adopts a recurrent neural network-based memory to form a communication channel among agents and uses a centralised control policy conditioning on the true state. Other authors have also studied the emergence of language in multi-agent systems (Lazaridou et al., 2016; Mordatch & Abbeel, 2018; Lazaridou et al., 2018). In these works, the environment typically provides explicit feedback about the communication actions. Unlike existing studies, we do not assume the existence of explicit rewards guiding the communication actions. Our problem therefore requires the hierarchical arrangement of communication policies and local agent policies that act on the environment, which must be learned concurrently. Furthermore, we consider settings where the observations are either wrong or randomly allocated across agents so that, without an appropriate communication strategy, no optimal policies can be learned.

### 3 Background

#### 3.1 Partially Observable Markov Games

Partially observable Markov Games (POMGs) (Littman, 1994) are multi-agent extensions of MDPs consisting of  $N$  agents with partial observations and characterised by a set of true states  $\mathcal{S}$ , a collection of action sets  $\mathcal{A} = \{\mathcal{A}_1, \dots, \mathcal{A}_N\}$ , a state transition function  $\mathcal{T}$ , a reward function  $\mathcal{R}$ , a collection of private observation functions  $\mathcal{Q} = \{\mathcal{Q}_1, \dots, \mathcal{Q}_N\}$ , a collection of private observations  $\mathcal{O} = \{\mathcal{O}_1, \dots, \mathcal{O}_N\}$  and a discount factor  $\gamma \in [0, 1]$ . A POMG is then defined by a tuple,  $G = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Q}, \mathcal{O}, \gamma, N \rangle$ . The agents do not have full access to the true state of the environment  $s \in \mathcal{S}$ , instead each receives a private partial observation correlated with the true state, i.e.  $o_i = \mathcal{Q}_i(s) : \mathcal{S} \rightarrow \mathcal{O}_i$ . Each agent chooses an action according to a (either deterministic or stochastic) policy parameterised by  $\theta_i$  and conditioned on its own private observation, i.e.  $a_i = \mu_{\theta_i}(o_i) : \mathcal{O}_i \rightarrow \mathcal{A}_i$  or  $\pi_{\theta_i}(a_i | o_i) : \mathcal{O}_i \times \mathcal{A}_i \rightarrow [0, 1]$ , and obtains a reward as a result of this action, i.e.  $r_i = \mathcal{R}(s, a_i) : \mathcal{S} \times \mathcal{A}_i \rightarrow \mathbb{R}$ . The environment then moves into the next state  $s' \in \mathcal{S}$  according to the state transition function conditioned on actions of all agents, i.e.  $s' = \mathcal{T}(s, a_1, \dots, a_N) : \mathcal{S} \times \mathcal{A}_1 \times \dots \times \mathcal{A}_N \rightarrow \mathcal{S}$ . Each agent aims to maximise its own total expected return,  $\mathbb{E}[R_i] = \mathbb{E}[\sum_{t=0}^T \gamma^t r_i^t]$  where  $r_i^t$  is the collected reward by the  $i^{\text{th}}$  agent at time  $t$  and  $T$  is the time horizon.

#### 3.2 Deterministic Policy Gradient Algorithms

Policy Gradient (PG) algorithms are based on the idea that updating the policy's parameter vector  $\theta$  in the direction of  $\nabla_{\theta} J(\theta)$  maximises the objective function,  $J(\theta) = \mathbb{E}[R]$ . The current policy  $\pi_{\theta}$  is specified by a stochastic function using a set of probability measures on the action space, i.e.  $\pi_{\theta} : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$ . Deterministic Policy Gradient (DPG) (Silver et al., 2014) extends the policy gradient framework by adopting a policy function that deterministically maps states to actions, i.e.  $\mu_{\theta} : \mathcal{S} \rightarrow \mathcal{A}$ . The gradient used to optimise the objective  $J(\theta)$  is

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{s \sim \mathcal{D}} [\nabla_{\theta} \mu_{\theta}(a | s) \nabla_a Q^{\mu}(s, a) |_{a=\mu_{\theta}(s)}] \quad (1)$$

where  $\mathcal{D}$  is an experience replay buffer and  $Q^{\mu}(s, a)$  is the corresponding action-value function. In DPG, the policy gradient takes the expectation only over the state space, which introduces data

efficiency advantages. On the other hand, as the policy gradient also relies on  $\nabla_a Q^\mu(s, a)$ , DPG requires a continuous policy  $\mu$ . Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015) builds upon DPG by adopting Deep Neural Networks to approximate  $\mu$  and  $Q^\mu$ . Analogously to DQN (Mnih et al., 2015), the experience replay buffer  $\mathcal{D}$  and target networks  $\mu'$ ,  $Q^{\mu'}$  help stabilise the learning.

### 3.3 Multi-agent Deep Deterministic Policy Gradient

Multi-agent Deep Deterministic Policy Gradient (MADDPG) extends DDPG to multi-agent settings by adopting the CTDE paradigm. DDPG is well-suited for such an extension as both  $\mu$  and  $Q^\mu$  can be made dependent upon external information. In order to prevent non-stationarity, MADDPG uses the actions and observations of all agents in the action-value functions,  $Q_i^\mu(o_1, a_1, \dots, o_N, a_N)$ . On the other hand, as the policy of an agent is only conditioned upon its own private observations,  $a_i = \mu_{\theta_i}(o_i)$ , the agents can act in a decentralised manner during execution. The gradient of the continuous policy  $\mu_{\theta_i}$  (hereafter abbreviated as  $\mu_i$ ) with respect to parameters  $\theta_i$  is

$$\nabla_{\theta_i} J(\mu_i) = \mathbb{E}_{o, a \sim \mathcal{D}} [\nabla_{\theta_i} \mu_i(a_i | o_i) \nabla_{a_i} Q_i^\mu(o_1, a_1, \dots, o_N, a_N) |_{a_i = \mu_i(o_i)}], \quad (2)$$

for  $i^{\text{th}}$  agent. Its centralised  $Q_i^\mu$  function is updated to minimise a loss based on temporal-difference

$$\mathcal{L}(\theta_i) = \mathbb{E}_{o, a, r, o'} [(Q_i^\mu(o_1, a_1, \dots, o_N, a_N) - y)^2], \quad (3)$$

where  $y = r_i + \gamma Q_i^{\mu'}(o'_1, a'_1, \dots, o'_N, a'_N) |_{a'_j = \mu'_j(o'_j)}$ . Here,  $\mu'_i$  is the target policy whose parameters  $\theta'_i$  are periodically updated with  $\theta_i$  and  $\mathcal{D}$  is the experience replay buffer consisting of the tuples  $(o, a, r, o')$ , where each element is a set of size  $N$ , i.e.  $o = \{o_1, \dots, o_N\}$ .

### 3.4 Intrinsically Motivated RL and Hierarchical-DQN

Intrinsically motivated learning has been well-studied in the RL literature (Singh et al., 2004; Schmidhuber, 1991); nevertheless, how to design a good intrinsic reward function is still an open question. Existing techniques relate to different notions of intrinsic reward. In general, an intrinsic reward can be considered an exploration bonus representing the novelty of the visited state. In other words, the intrinsic rewards encourage the agent to explore the state space while the extrinsic rewards collected from the environment provide task related feedback. For example, in the simplest setting, an intrinsic reward can be a decreasing function of state visitation counts (Bellemare et al., 2016; Ostrovski et al., 2017).

Kulkarni et al. (2016) introduce a notion of relational intrinsic rewards in order to train a two-level hierarchical-DQN model. In this model, at the top-level, the agent learns a policy  $\pi_g$  to select an intrinsic goal  $g \in \mathcal{G}$ , i.e.  $\pi_g = P(g|s)$ . This intrinsic goal is then used at the bottom-level whereby the agent learns a policy  $\pi_a$  for its actions, i.e.  $\pi_a = P(a|s, g)$ . In this setting, the top-level and the bottom-level policies are driven by the extrinsic and intrinsic rewards, respectively.

## 4 Multi-agent DDPG with a Communication Medium

### 4.1 Problem formulation and proposed approach

We consider partially observable Markov Games, and assume that the observations received by most agents are extremely noisy and weakly correlated to the true state, which makes learning optimal policies unfeasible. Conditioning each policy on all  $N$  private observations, i.e.  $a_i = \mu_i(o_1, \dots, o_N)$ , is not helpful given that a large majority of  $o_i$ 's are uncorrelated to the corresponding  $s$ , i.e. they provide a poor representation of the current true state for the  $i^{\text{th}}$  agent. To address this challenge, we let every agent's policy depend on its private observations as well as those explicitly and selectively shared by other agents. As agents cannot discriminate between relevant and noisy information on their own, the ability to decide whether to share their own observations with others must also be acquired through experience. More formally, we introduce two hierarchically arranged sets of policies,  $\nu = \{\nu_1, \dots, \nu_N\}$  and  $\mu = \{\mu_1, \dots, \mu_N\}$ , that are coupled through a communication medium  $m = \{m_1, \dots, m_N\}$ , where  $m_i$  denotes the information shared to the  $i^{\text{th}}$  agent. The *action policies* in  $\mu$  determine the actions agents take to interact with the environment, whereas the *communication*

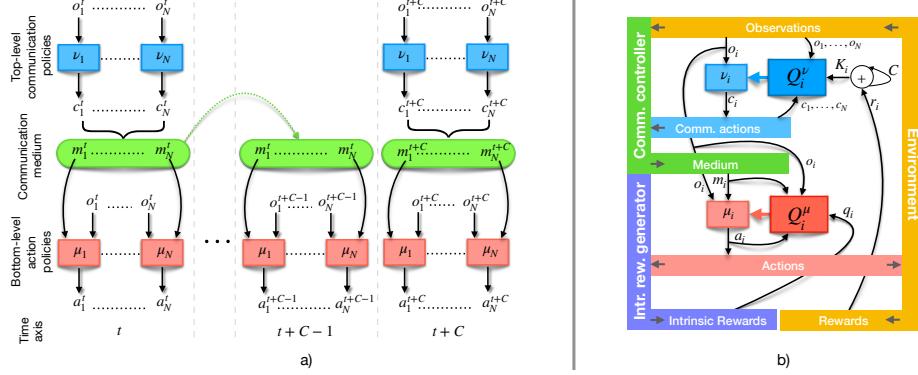


Figure 1: Overview of MADDPG-M. In (a), the  $N$  agents learn two hierarchically arranged sets of policies, which are connected through a communication medium. During training, we run communication policies at a slower time-scale, i.e. once in every  $C$  steps of the action policies, and determine the environmental actions using fixed communication medium over the  $C$  steps. In (b), the communication policies are learned within a CTDE paradigm using the cumulative sum of the rewards collected from the environment for these  $C$  steps, while we learn action policies in decentralised way using intrinsic rewards estimated with respect to the communication medium.

*policies* in  $\nu$  control the communication actions determining the information shared in the medium. At the top-level, each agent chooses a communication action  $c_i$  through its communication policy  $\nu_i$  conditioned only on its own private observation, i.e.  $c_i = \nu_i(o_i)$ .

We consider two possible types of communication mechanisms: *broadcasting* (one-to-all) and *unicasting* (one-to-one). In the broadcasting case, each communication action is a scalar,  $c_j \in \mathbb{R} \mid 0 \leq c_j \leq 1$ , and the observation of the agent with the largest communication action is sent to all other agents; in this case  $m$  is defined as

$$m = \left\{ m_i = o_k \quad \forall i \in \{1, \dots, N\} \quad \text{where } k = \arg \max_j (c_1, \dots, c_j, \dots, c_N) \right\} \quad (4)$$

In the unicasting case, the communication action is an  $N$ -dimensional vector, i.e.  $c_{j,:} \in \mathbb{R}^{1 \times N} \mid 0 \leq c_{j,i} \leq 1$  where  $c_{j,i}$  can be interpreted as a measure of the  $j^{\text{th}}$  agent's "willingness" to share its private observation with the  $i^{\text{th}}$  agent such that the observation of the agent with the greatest willingness is shared with  $i^{\text{th}}$  agent, i.e. assigned to  $m_i$ . In this case,  $m$  is defined as

$$m = \left\{ m_i = o_k \quad \text{where } k = \arg \max_j (c_{1,i}, \dots, c_{j,i}, \dots, c_{N,i}) \right\} \quad (5)$$

At the bottom-level, exploiting the information that has been shared, each agent determines its environmental action, i.e.  $a_i = \mu_i(o_i, m_i)$ .

#### 4.2 Learning algorithm

The two sets of policies,  $\nu$  and  $\mu$ , are coupled and must be learned concurrently. To address this issue, we use two different levels of temporal abstraction to collect transitions from the environment during training; that is,  $\nu$  and  $\mu$  are run at different time-scales. The communication actions are performed once in every  $C$  steps. During this period, the state of medium is kept fixed, i.e.  $m^t = m^{t+1} = \dots = m^{t+C-1}$ , and the environmental actions  $a$  are obtained using the fixed medium. Given that the environment does not explicitly reward good communication strategies, there is no obvious way to optimise the communication policies. Instead, we use the cumulative sum of the extrinsic rewards collected from the environment for these  $C$  steps, i.e.  $K = \sum_{t'=t}^{t+C} r^{t'}$ . At each time step, we also generate an *intrinsic reward*,  $q$ , in response to the environmental actions and use it to optimise  $\mu$ .

In the RL literature, the notion of intrinsic rewards is mostly used for exploration purposes. Instead, in this work, intrinsic rewards are introduced to enable the agents to learn the environmental dynamics even when the communication decisions coming from the top-level are not optimal. In the tasks we

consider, the extrinsic rewards of the environment measure the distance between the agents and their true targets even when the agents do not truly observe these targets. When the agents cannot see the true targets, they cannot learn to reach them because their observations and the received rewards become uncorrelated. On the other hand, the intrinsic rewards we introduce represent the distance between the agents and the targets that appear in the medium, regardless of whether they are the noisy ones or the true ones. By doing this, at the bottom-level the agents learn how to reach the targets shared in the medium. At the top-level, using accumulated extrinsic rewards, they collectively infer which observations better represent the true targets and should be shared through the medium. Our developments are inspired by Kulkarni et al. (2016) where the intrinsic rewards are used to aid exploration in a single agent system. Here we introduce an analogy between their concept of intrinsic goals, which are held fixed until they are reached by the agent, and the concept of the communication medium  $m$ .

We keep two separate experience relay buffers,  $\mathcal{D}_\nu$  and  $\mathcal{D}_\mu$ . The experience replay buffer  $\mathcal{D}_\nu$  consists of the tuples  $(o, c, K, o'')$ , where  $o''$  denotes the  $C^{\text{th}}$  observation after  $o$ , i.e.  $o'' = o^{t+C}$ , and provides the samples to be used to update the communication policies  $\nu$ . On the other hand, the other experience replay buffer,  $\mathcal{D}_\mu$ , consists of the tuples  $(o, m, a, q, o')$  where  $o'$  denotes the next observation after  $o$ , i.e.  $o' = o^{t+1}$ , and provides the samples to be used to update the action policies  $\mu$ . We employ actor-critic policy gradient based methods for both  $\nu$  and  $\mu$ , and train the communication policies  $\nu$  within a CTDE paradigm. For an agent, the policy gradient with respect to parameters  $\theta_{\nu,i}$  is written as

$$\nabla_{\theta_{\nu,i}} J(\nu_i) = \mathbb{E}_{o,c \sim \mathcal{D}_\nu} [\nabla_{\theta_{\nu,i}} \nu_i(c_i|o_i) \nabla_{c_i} Q_i^\nu(o_1, c_1, \dots, o_N, c_N)|_{c_i=\nu_i(o_i)}]. \quad (6)$$

The corresponding centralised action-value function  $Q_i^\nu$  is updated to minimise the following loss based on temporal-difference

$$\mathcal{L}(\theta_{\nu,i}) = \mathbb{E}_{o,c,K,o''} [(Q_i^\nu(o_1, c_1, \dots, o_N, c_N) - y)^2], \quad (7)$$

where  $y = K_i + \gamma Q_i^{\nu'}(o''_1, c''_1, \dots, o''_N, c''_N)|_{c''_j=\nu'_j(o''_j)}$  and  $\nu'$  is the target policy whose parameters  $\theta'_{\nu,i}$  are periodically updated with  $\theta_{\nu,i}$ . The action policies  $\mu$  at the bottom-level generalise not only over the private observations, but also over  $m$ . The policy gradient with respect to parameters  $\theta_{\mu,i}$  then becomes

$$\nabla_{\theta_{\mu,i}} J(\mu_i) = \mathbb{E}_{o,m,a \sim \mathcal{D}_\mu} [\nabla_{\theta_{\mu,i}} \mu_i(a_i|o_i, m_i) \nabla_{a_i} Q_i^\mu(o_i, m_i, a_i)|_{a_i=\mu_i(o_i, m_i)}], \quad (8)$$

Unlike the communication policies, the action policies are trained in a decentralised manner. The corresponding action-value function  $Q_i^\mu$  of the  $i^{\text{th}}$  agent is updated to minimise the following loss based on temporal-differences

$$\mathcal{L}(\theta_{\mu,i}) = \mathbb{E}_{o,m,a,q,o'} [(Q_i^\mu(o_i, m_i, a_i) - y)^2], \quad (9)$$

where  $y = q_i + \gamma Q_i^{\mu'}(o'_i, m_i, a'_i)|_{a'_i=\mu'_i(o'_i, m_i)}$  and  $\mu'$  is the target policy whose parameters  $\theta'_{\mu,i}$  are periodically updated with  $\theta_{\mu,i}$ . An illustration of the proposed approach can be found in Figure 1, and the pseudo-code describing the learning algorithm can be found in the Appendix.

## 5 Experiments

### 5.1 Environments

In this section we introduce six different variations of the *Cooperative Navigation* problem from the Multi-agent Particle Environment (Lowe et al., 2017). In its original version,  $N$  agents need to learn to reach  $N$  landmarks while avoiding collisions with each other. Each agent observes the relative positions of the other  $N - 1$  agents and the  $N$  landmarks. The agents are collectively rewarded based on their distance to the landmarks. Unlike most real-world multi-agent use cases, each agent's private observation provides an almost complete representation of the true state of the environment. As such, independently trained agents can reach performance levels comparable to those achievable through centralised learning (see the Appendix for supporting evidence). Hence, in its original version, there is no real need for inter-agent communication. We now describe our modifications of this environment that more closely capture the complexities of real-world applications. We classify the scenarios into two groups according to the type of the communication strategy required to solve the task.

In the first group, only one of the agents - the *gifted agent* - can observe the true position of the landmarks. This special agent can either remain the same throughout the whole learning period or vary across episodes, and even within an episode. All other agents besides the gifted one receive inaccurate information about the landmarks' positions. Crucially, no agent is aware of their status (i.e. whether they are gifted or not), rather they all need to learn through interactions with the environment whether their own observations truly contribute to the improvement of the overall policies, and should therefore be shared with others. This learning task is accomplished by deciding at any given time whether to pass their observations onto all other agents simultaneously through a broadcasting communication mechanism; see also Eq. (4). Only indirect feedback from the environment through the reward function can inform the agents as to whether their current communication strategy improves their policies. This group of tasks include three different variants of increasing complexity depending on how the gifted agent is defined: in the *fixed* case, the gifted agent stays the same throughout the training phase, i.e. the true landmarks are always observed by the same agent; in the *alternating* case, the gifted agent may change at each episode, i.e. the ability to observe the true landmarks is randomly assigned to one of the agents at the beginning of each episode and represented by a *flag* in their observation space; in the *dynamic* case, the agent closest to the centre of the environment becomes the gifted one within each episode. Through the relative distances between each other, agents need to understand implicitly which one of them is closest to the centre.

The second group of environments is characterised by the fact that each landmark has been pre-assigned to a particular agent, and an agent needs to occupy its allocated landmark to collect good rewards. In these scenarios, each agent correctly observes only the location of one landmark and the other landmarks are wrongly perceived. The agents are again unaware of their true status (i.e. they do not know which one of the landmarks is true, and dedicated to whom), and must learn through experience how to strategically share information so as to maximise the expected rewards. In these settings an agent can decide to send its observation, at any given time, to which one(s) of the  $N - 1$  remaining agents through a unicasting mechanism; see also Eq. (5). Within this group we also have three different variants of increasing complexity depending on how frequently the correct observation dependencies change; *fixed* throughout the training, *alternating* across episodes and *dynamic* within each episode according to the agents' distances to the centre of the environment. In all 6 scenarios, while the extrinsic rewards of the environment represent the distance to the true landmark locations to be occupied, the intrinsic rewards we generate represent the distance to the landmark locations shared in the medium, regardless of whether or not they are actually the true landmarks.

## 5.2 Comparison with Baselines

We evaluate MADDPG-M against four actor-critic based baselines - DDPG, MADDPG, Meta-agent and DDPG-OC - on the six environments introduced in the previous section. In all experiments we use three agents, i.e.  $N = 3$ . In DDPG, agents are trained and executed in decentralised manner, i.e.  $Q_i^\mu(o_i, a_i)$  and  $\mu_i(o_i)$ . In MADDPG, agents are trained in centralised manner, but the actions are executed in decentralised manner as each agent's policy is conditioned only on its own observations, i.e.  $Q_i^\mu(o_1, a_1, \dots, o_N, a_N)$  and  $\mu_i(o_i)$ . A Meta-agent has access to all the observations, across all agents, during both training and execution, i.e.  $Q_i^\mu(o_1, a_1, \dots, o_N, a_N)$  and  $\mu_i(o_1, \dots, o_N)$ . Although this approach becomes impractical with a large number of agents, it is included here to demonstrate the performance gains that can be achieved by strategically communicating only the

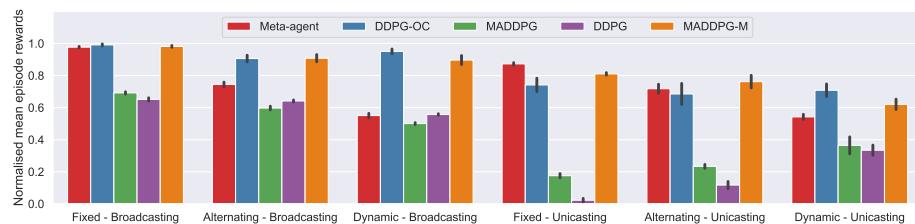


Figure 2: Comparison between MADDPG-M and 4 baselines for all 6 scenarios. Each bar cluster shows the 0-1 normalised mean episode rewards when trained using the corresponding approach, where a higher score is better for the agent. Full results are given in the Appendix.

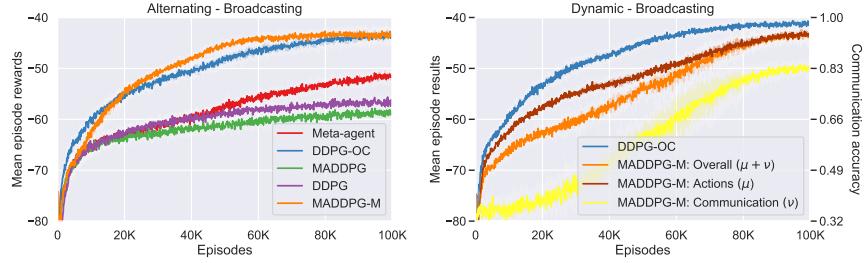


Figure 3: a) On *Alternating - Broadcasting*, the reward of MADDPG-M against baseline approaches after 100,000 episodes. b) On *Dynamic - Broadcasting*, the rewards of MADDPG-M against DDPG-OC, the accuracy curve of the communication actions to observe the individual performance of  $\nu$ , and the collected intrinsic rewards to observe the individual performance of  $\mu$ .

relevant information compared to a naive solution where all the observations are shared, including the noisy ones. The DDPG-OC (DDPG with Optimal Communication) baseline is related to DDPG, but uses a hard-coded communication pattern, i.e.  $m$  is assigned optimally using *a priori* knowledge about the underlying communication requirement. The agents are trained and executed in a decentralised manner, but exploiting the communication, i.e.  $Q_i^{\bar{\mu}}(o_i, m_i, a_i)$  and  $\mu_i(o_i, m_i)$ . As the state of the medium is hard-coded, only the action policies need to be learned in this case. This approach is included to study what level of performance is achievable when communicating optimally, and learning only the main policies. Following Lowe et al. (2017), we use a two-layer ReLU Multi Layer Perceptron (MLP) with 64 units per layer to parameterise the policies, and a similar approximator using 128 units per layer to parameterise their action-value functions. We train them until they all converge. After training, we run an additional 1,000 episodes to collect performance metrics: *collected rewards* for all baselines, in addition to *collected intrinsic rewards* and *communication accuracies* only for MADDPG-M. We repeat this process five times per baseline, and per scenario, and report the averages. Figure 2 summarises our empirical finding in terms of normalised mean episode rewards (the higher the better). Additional numerical details are provided in Appendix.

Initially, we examine the first group of scenarios consisting of tasks that can be solved using a *broadcasting* communication mechanism. Figure 3-a shows learning curves for MADDPG-M and all baselines on the *alternating-broadcasting* scenario in terms of rewards collected from the environment. In these cases, both DDPG and MADDPG fail to learn the correct behaviour; this was an expected outcome given that both methods do not allow for the observations to be shared. Their poor performances reinforce the idea that learning a coordinated behaviour through centralised training may not be sufficient in certain situations. These levels of performance provide a lower bound in our experiments. In practice, when using these baselines, we observed that the agents simply move towards the middle of the environment; even the gifted agent cannot learn a rational behaviour as the reward signal becomes noisy due to arbitrary actions taken by the agents. On the other hand, the performance achieved by DDPG-OC demonstrate that, when  $m$  is correctly controlled, all the scenarios can be accomplished even when the agents are trained in a decentralised manner. Interestingly, despite reaching a satisfactory level on the simplest *fixed* case, the performance of the Meta-agent decreases dramatically as the complexity of our environments increases, and this algorithm completely fails to solve the *dynamic* case.

Conversely, MADDPG-M allows the agents to simultaneously learn the underlying communication scheme as well as the optimal action policies, and ultimately perform quite similarly to DDPG-OC in all our environments. In order to assess the action-specific (due to  $\mu$ ) and communication-specific (due to  $\nu$ ) performances, Figure 3-b presents the collected intrinsic rewards as well as the accuracy of the communication actions performed by MADDPG-M with respect to those optimally implemented in DDPG-OC on *dynamic-broadcasting* scenario. In the initial phases of training, although the communication policy is not yet sufficiently optimised, the MADDPG-M agents are nevertheless able to begin learning the environment dynamics and the expected actions through the intrinsic rewards. Improved environmental actions subsequently provide better feedback yielding improved communications actions, and so on. Ultimately, MADDPG-M agents perform comparably to DDPG-OC. Again, the communication accuracy decreases as the environments become more difficult. However, even in the most complex setting amongst the *broadcasting* scenarios, MADDPG-M agents

choose the optimal communication actions 88.82% of time, which is sufficient to accomplish the task. Very similar conclusions can be drawn when studying the *unicasting* scenarios. Due to the increased complexity, agents across all baselines tend to collect less rewards than their counterparts in the *broadcasting* scenarios. MADDPG-M can achieve a performance similar to the empirical upper-bound provided by DDPG-OC. It is worth noting that the observed variability in the *unicasting* scenarios is higher compared to the *broadcasting* scenarios due to the increased communication requirements as well as the task complexity (e.g. each agent needs to move to the opposite side of the environment, which results in more collisions). This may explain why DDPG-OC has higher variance despite using optimal communication. Interestingly, in *dynamic-unicasting* scenario, MADDPG-M agents can only find the overall optimal communication pattern 28.98% of time. However, as the individual communication actions are accurate for 64.23% of time, they can manage to accomplish the task. Further results as well as implementation details (including hyperparameters) can be found in the Appendix.

## 6 Conclusions

In this paper we have studied a multi-agent reinforcement learning problem characterised by partial and extremely noisy observations. We have considered two instances of this problem: a situation where most agents receive wrong observations, and a situation where the observations required to characterise the environment are randomly distributed across agents. In both cases, we demonstrate that learning what and when to communicate is an essential skill that needs to be acquired in order to develop a collaborative behaviour and accomplish the underlying task. Our proposed MADDPG-M algorithm enables concurrent learning of an optimal communication policy and the underlying task. Effectively, the agents learn an information sharing strategy that progressively increases the collective rewards. The key technical contribution consists of hierarchical interpretation of the communication-action dependency. Agents learn two policies that are connected through a communication medium. To train these policies concurrently, we use different levels of temporal abstraction and also exploit intrinsically generated rewards according to the state of the medium. In our studies, we have considered scenarios where sharing a single observation at a time is sufficient to accomplish the task. There might be more complex cases where an agent needs to reach the observations of multiple agents at the same time. Moreover, rather than sharing raw observations, which may be high-dimensional and possibly contain redundant information (e.g. pixel data), it may be conceivable to learn a more useful representation.

## References

- Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1471–1479, 2016.
- José Bento, Nate Derbinsky, Javier Alonso-Mora, and Jonathan S. Yedidia. A message-passing algorithm for multi-agent trajectory planning. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pp. 521–529, 2013.
- C.M. Colson, M.H. Nehrir, and R.W. Gunderson. Multi-agent microgrid power management. *IFAC Proceedings Volumes*, 44(1):3678 – 3683, 2011. ISSN 1474-6670. doi: <https://doi.org/10.3182/20110828-6-IT-1002.01188>. 18th IFAC World Congress.
- Gianni Di Caro, Marco Dorigo, et al. An adaptive multi-agent routing algorithm inspired by ants behavior. In *Proceedings of PART98-5th Annual Australasian Conference on Parallel and Real-Time Systems*, pp. 261–272, 1998.
- Kurt M. Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *J. Artif. Intell. Res.*, 31:591–656, 2008. doi: 10.1613/jair.2502.
- Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2137–2145, 2016.

- Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.
- Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR*, abs/1507.06527, 2015.
- Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144, 2016. URL <http://arxiv.org/abs/1611.01144>.
- Zeng Jun, Liu Junfeng, Wu Jie, and H.W. Ngan. A multi-agent solution to energy management in hybrid renewable energy generation system. *Renewable Energy*, 36(5):1352 – 1363, 2011. ISSN 0960-1481. doi: <https://doi.org/10.1016/j.renene.2010.11.032>.
- Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3675–3683, 2016.
- Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *CoRR*, abs/1612.07182, 2016.
- Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. Emergence of linguistic communication from referential games with symbolic and pixel input. *arXiv preprint arXiv:1804.03984*, 2018.
- Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.
- Long Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992. doi: 10.1007/BF00992699.
- Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, July 10-13, 1994*, pp. 157–163, 1994.
- Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 6382–6393, 2017.
- Laëtitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *Knowledge Eng. Review*, 27(1):1–31, 2012. doi: 10.1017/S0269888912000057.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236.
- Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.
- Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 2721–2730, 2017.
- Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.
- Jürgen Schmidhuber. Curious model-building control systems. In *Neural Networks, 1991. 1991 IEEE International Joint Conference on*, pp. 1458–1463. IEEE, 1991.
- David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 387–395, 2014.

- David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961.
- Satinder P. Singh, Andrew G. Barto, and Nuttapong Chentanez. Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pp. 1281–1288, 2004.
- Armin Stranjak, Partha Sarathi Dutta, Mark Ebden, Alex Rogers, and Perukrishnen Vytelingum. A multi-agent simulation system for prediction and scheduling of aero engine overhaul. In *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, May 12-16, 2008, Industry and Applications Track Proceedings*, pp. 81–88, 2008. doi: 10.1145/1402795.1402811.
- Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2244–2252, 2016.
- Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.
- Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PLOS ONE*, 12(4):1–15, 04 2017. doi: 10.1371/journal.pone.0172395.
- Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.
- Gerald Tesauro. Extending q-learning to general adaptive multi-agent systems. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pp. 871–878, 2003.
- George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

## A Appendices

### A.1 MADDPG-M pseudo code

For completeness, we provide the pseudo-code for MADDPG-M in Algorithm 1.

---

#### Algorithm 1: Learning algorithm for MADDPG-M

---

```

Initialise parameters of  $\mu$ ,  $\nu$  and  $Q^\mu$ ,  $Q^\nu$ 
Initialise replay buffers  $\mathcal{D}_\mu$  and  $\mathcal{D}_\nu$ 
Initialise random processes  $\mathcal{N}_\mu$  and  $\mathcal{N}_\nu$  for exploration
for  $episode \leftarrow 1$  to  $num\_episodes$  do
    Reset the environment and receive initial observations  $o = \{o_1, \dots, o_N\}$ 
    Reset temporal abstraction counter  $count \leftarrow 0$ 
    for  $t \leftarrow 1$  to  $num\_steps$  do
        if  $count$  is 0 then
            for  $i \leftarrow 1$  to  $N$  do
                Select comm. action  $c_i = \nu_i(o_i) + \mathcal{N}_\nu^t$  w.r.t. the current comm. policy and exploration
                Obtain the state of the medium  $m = f_m(o, c)$  where  $f_m$  is either Eq. (4) or Eq. (5)
                Keep observations  $o_{init} \leftarrow o$ 
                Reset accumulated reward  $K \leftarrow 0$ 
            for  $i \leftarrow 1$  to  $N$  do
                Select action  $a_i = \mu_i(o_i, m_i) + \mathcal{N}_\mu^t$  w.r.t. the current action policy and exploration
            Execute actions  $a = (a_1, \dots, a_N)$ , get rewards  $r = (r_1, \dots, r_N)$  and next obs.  $o' = (o'_1, \dots, o'_N)$ 
            Accumulate rewards  $K \leftarrow K + r$ 
            Get intrinsic rewards  $q = f_q(o, a, o', m)$ , where  $q = (q_1, \dots, q_N)$ 
            Store  $(o, m, a, q, o')$  in replay buffer  $\mathcal{D}_\mu$ 
        if  $count$  is  $C - 1$  then
            Store  $(o_{init}, c, K, o')$  in replay buffer  $\mathcal{D}_\nu$ 
            Reset temporal abstraction counter  $count \leftarrow 0$ 
        else
            Increment temporal abstraction counter  $count \leftarrow count + 1$ 
         $o \leftarrow o'$ 
        for  $i \leftarrow 1$  to  $N$  do
            Sample a random mini-batch of  $\mathcal{S}$  samples  $(o^k, m^k, a^k, q^k, o'^k)$  from  $\mathcal{D}_\mu$ 
            Set  $y^k = q^k + \gamma Q_i^\mu(o_i^k, m_i^k, a_i^k)|_{a_i'=\mu_i'(o_i^k, m_i^k)}$ 
            Update action critic by minimising the loss:
             $\mathcal{L}(\theta_{\mu,i}) = \frac{1}{S} \sum_k (Q_i^\mu(o_i^k, m_i^k, a_i^k) - y^k)^2$ 
            Update action actor using the sampled policy gradient:
             $\nabla_{\theta_{\mu,i}} J(\mu_i) \approx \frac{1}{S} \sum_k \nabla_{\theta_{\mu,i}} \mu_i(o_i^k, m_i^k) \nabla_{a_i} Q_i^\mu(o_i^k, m_i^k, a_i)|_{a_i=\mu_i(o_i^k, m_i^k)}$ 
            Sample a random mini-batch of  $\mathcal{S}$  samples  $(o^k, c^k, K^k, o'^k)$  from  $\mathcal{D}_\nu$ 
            Set  $y^k = K^k + \gamma Q_i^\nu(o_i^k, c_i^k, \dots, o_N^k, c_N^k)|_{c_j'=\nu_j'(o_j^k)}$ 
            Update communication critic by minimising the loss:
             $\mathcal{L}(\theta_{\nu,i}) = \frac{1}{S} \sum_k (Q_i^\nu(o_1^k, c_1^k, \dots, o_N^k, c_N^k) - y^k)^2$ 
            Update communication actor using the sampled policy gradient:
             $\nabla_{\theta_{\nu,i}} J(\nu_i) \approx \frac{1}{S} \sum_k \nabla_{\theta_{\nu,i}} \nu_i(o_i^k) \nabla_{c_i} Q_i^\nu(o_1^k, c_1^k, \dots, o_N^k, c_N^k)|_{c_i=\nu_i(o_i^k)}$ 
        Update target network parameters for each agent  $i$ :
         $\theta'_{\nu,i} \leftarrow \tau \theta_{\nu,i} + (1 - \tau) \theta'_{\nu,i}$ 
         $\theta'_{\mu,i} \leftarrow \tau \theta_{\mu,i} + (1 - \tau) \theta'_{\mu,i}$ 

```

---

### A.2 Further experimental details

In all of our experiments, we use the Adam optimiser with a learning rate of 0.01 and  $\tau = 0.01$  for updating the target networks. The size of the replay buffer is  $10^6$  and we update the network parameters after every 100 samples added to the replay buffer. We use a batch size of 1024. During training we set  $C = 5$ , but to get performance metrics for execution we set it back to  $C = 1$ . For the exploration noise, following (Lillicrap et al., 2015), we use an Ornstein-Uhlenbeck process (Uhlenbeck & Ornstein, 1930) with  $\theta = 0.15$  and  $\sigma = 0.2$ . For all environments, we run the

algorithms for 100,000 episodes with 25 steps each. We only change  $\gamma$  across different scenarios. We use  $\gamma = 0.8$  for MADDPG-M in *fixed-broadcasting*, *alternating-unicasting* and *dynamic-unicasting* scenarios. We all use  $\gamma = 0.85$  for all models in all scenarios, except these three cases. The learning curves for all model in all scenarios are provided in Figure A.4 and Figure A.5. The learning curves of three baseline models, Meta-agent, MADDPG, DDPG, in the original *Cooperative Navigation* scenario of Multi-agent Particle Environments are given in Figure A.6 in order to show that there is no real need for inter-agent communication in the original *Cooperative Navigation* scenario as DDPG performs similarly to MADDPG and Meta-agent.

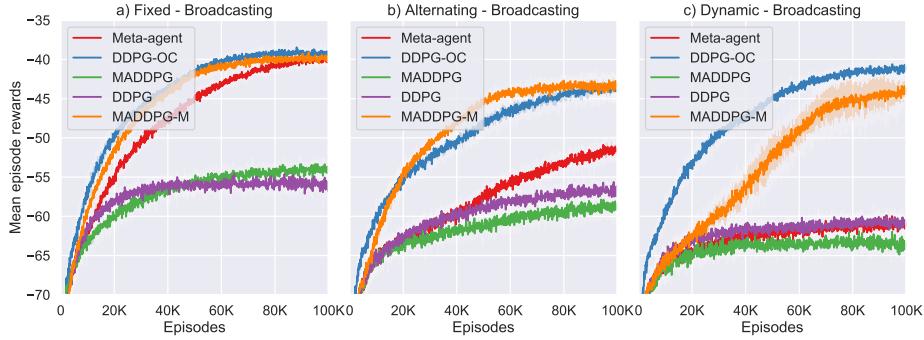


Figure A.4: Learning curves for all model in *broadcasting* scenarios over 100,000 episodes. Results of baseline models are also provided for comparison.

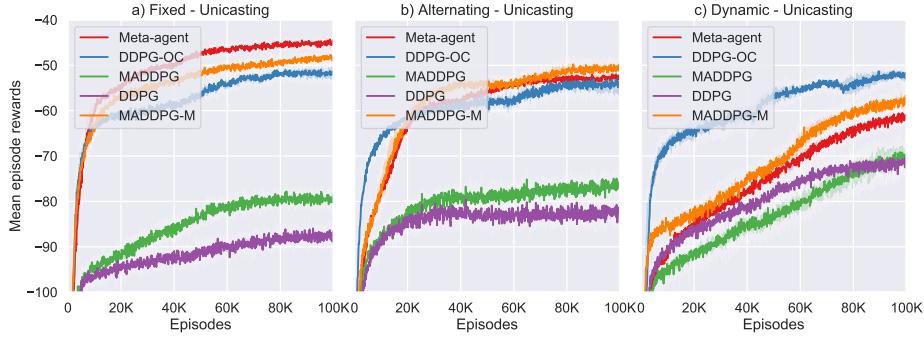


Figure A.5: Learning curves for all model in *unicasting* scenarios over 100,000 episodes. Results of baseline models are also provided for comparison.

Table A.1: Mean (standard deviations) episode rewards for all baselines in all 6 scenarios.

	Fixed - Broadcasting	Alternating - Broadcasting	Dynamic - Broadcasting	Fixed - Unicasting	Alternating - Unicasting	Dynamic - Unicasting
Meta-agent	-39.95( $\pm 4.50$ )	-51.42( $\pm 7.70$ )	-60.98( $\pm 8.82$ )	-45.09( $\pm 6.58$ )	-52.73( $\pm 6.73$ )	-61.39( $\pm 12.22$ )
Hard-coded	-39.26( $\pm 4.45$ )	-43.44( $\pm 5.92$ )	-41.25( $\pm 5.24$ )	-51.57( $\pm 7.03$ )	-54.34( $\pm 10.61$ )	-53.20( $\pm 8.54$ )
MADDPG	-54.00( $\pm 7.43$ )	-58.67( $\pm 8.90$ )	-63.44( $\pm 9.88$ )	-79.47( $\pm 12.99$ )	-76.60( $\pm 17.72$ )	-70.15( $\pm 14.07$ )
DDPG	-56.00( $\pm 8.96$ )	-56.50( $\pm 8.51$ )	-60.66( $\pm 8.68$ )	-87.02( $\pm 14.21$ )	-82.35( $\pm 19.50$ )	-71.63( $\pm 14.41$ )
Our approach	-39.73( $\pm 5.09$ )	-43.34( $\pm 7.29$ )	-43.91( $\pm 7.75$ )	-48.16( $\pm 7.02$ )	-50.55( $\pm 8.50$ )	-57.53( $\pm 8.50$ )

### A.2.1 Effects of hyperparameters

Figure A.7-a illustrates the learning curves for MADDPG-M with different  $C$  values in *dynamic-broadcasting* scenario. All curves are obtained using  $\gamma = 0.7$ . One can observe that choosing  $C > 2$  improves the training; however, further changes do not affect the performance significantly.

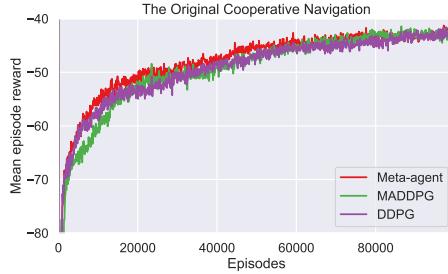


Figure A.6: Learning curves for Meta-agent, MADDPG, DDPG in the original *Cooperative Navigation* scenario over 100,000 episodes. Please note that DDPG performs similarly to MADDPG and Meta-agent.

Table A.2: Mean (standard deviation) communication accuracies for MADDPG-M in all 6 scenarios. *All* presents the overall accuracy (if all  $N$  dimensions of the medium is set correctly) and *any* presents the individual accuracy (if any one of  $N$  dimensions of the medium is set correctly). These two values are the same in the *broadcasting* scenarios.

	Fixed - Broadcasting	Alternating - Broadcasting	Dynamic - Broadcasting	Fixed - Unicasting	Alternating - Unicasting	Dynamic - Unicasting
All	99.98%( $\pm 0.02$ )	99.54%( $\pm 0.56$ )	88.82%( $\pm 5.91$ )	96.89%( $\pm 2.79$ )	47.43%( $\pm 0.66$ )	28.98%( $\pm 5.82$ )
Any	-	-	-	96.96%( $\pm 0.08$ )	49.30%( $\pm 1.43$ )	64.23%( $\pm 2.56$ )

### A.2.2 Effects of fixing the medium during training

We make an analogy between the concept of intrinsic goals introduced by Kulkarni et al. (2016), which are held fixed until they are reached by the agent, and the concept of the communication medium  $m$  in this work. Nevertheless, one might also make an analogy between these intrinsic goals and the communication actions  $c$ , and propose to fix only  $c$  while updating  $m$  according to these fixed communication actions. However in this case, the state of the medium would change at each time-step with the environmental actions of the agents that are granted to change it. Therefore, this would introduce a non-stationary reference for other agents as they condition their actions on the medium, and would contribute to the overall non-stationarity of the multi-agent setting. Figure A.7-b empirically shows that fixing the medium along with the communication actions help the learning algorithm find a better policy. During training, fixing the medium for  $C$  steps is crucial for the performance as it provides a stationary reference to the agents to learn their action policies conditioning on the medium. However, it is worth noting that updating the medium in a slower time-scale during training does not cause a sparse communication during execution. Instead, as we set  $C = 1$  after the training, agents can communicate at every time step during execution.

### A.2.3 Effects of using discrete communication actions

Communication actions we consider in this paper can also be implemented as discrete actions through a Gumbel-Softmax estimator (Jang et al., 2016). However, we have observed similar performances with both cases, i.e. -43.91 with continuous communication actions vs. -44.54 with discrete communication actions, and decided to keep the continuous actions as they appear more generalisable and potentially amenable to further extensions.

## A.3 Environment details

Figure A.8 illustrates the scenarios considered in this paper. The details of the experimental results are shown in Tables A.1 and A.2. In all environments, each agent observes its own position and velocity, and also observes the true relative positions of other agents. The observation schemes of the landmark locations vary across the scenarios. Environmental actions of agents consist of a vector of 4 continuous 0 – 1 valued scalars. Each scalar corresponds to a direction and its magnitude determines the velocity of the agent in that direction.

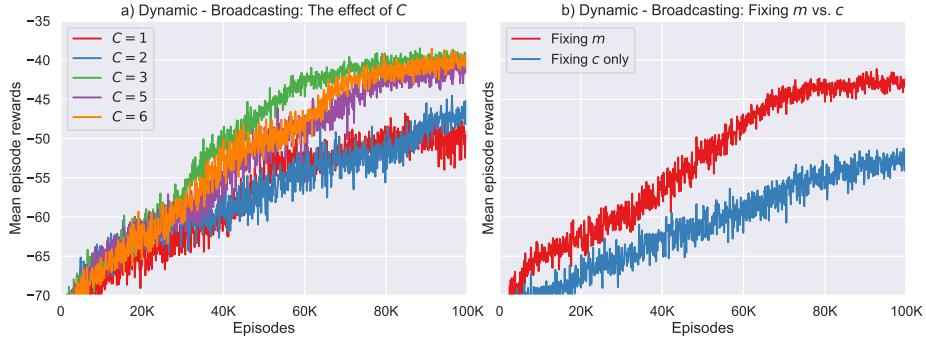


Figure A.7: a) Learning curves for MADDPG-M with different  $C$  values in *dynamic-broadcasting* scenario. b) Learning curves for MADDPG-M i) when  $m$  is fixed for  $C$  steps ii) when  $c$  is fixed and  $m$  is being updated according to the fixed  $c$  for  $C$  steps.

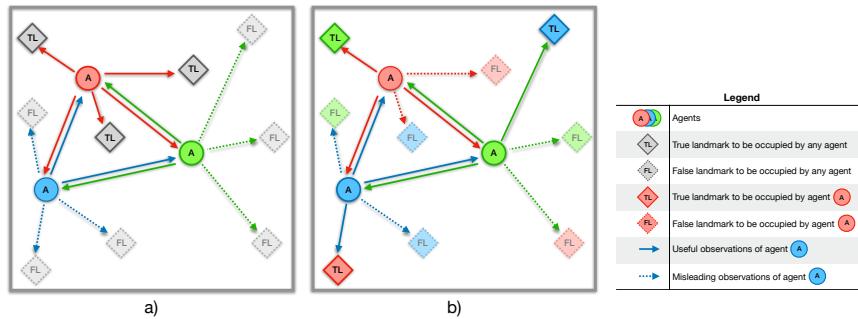


Figure A.8: An illustration of our environments: a) Within the first group, the *gifted* agent (red circle) is able to correctly observe all three landmarks (grey squares); the other agents (blue and green circles) receive the wrong landmarks' locations. b) Within the second group, each landmark is designated to a particular agent, and the agents get rewarded only when reaching their allocated landmarks. The *gift* is equally but partially distributed across the agents, and hence each agent can only correctly observe one of the landmarks (either its own or another agent's), but otherwise receives the wrong whereabouts of the remaining ones. In this case: the red agent can see the landmark assigned to the green agent, the green agent can see the landmark assigned to the blue agent, and the blue agent can see the landmark assigned to the red agent.



## 浙江大学本科生毕业论文（设计）编写规则

为规范我校本科生毕业论文(设计)编写格式,根据《学位论文编写规则》(GB/T 7713.1 - 2006),结合我校实际情况,制定本本科生毕业论文(设计)编写规则。

### 1 本科生毕业论文(设计)工作文档

本科生毕业论文(设计)工作文档分两大部分。

第一部分(论文(设计)材料)编排顺序依次是前置部分、主体部分、结尾部分、《浙江大学本科生毕业论文(设计)任务书》、《浙江大学本科生毕业论文(设计)考核表》。

第二部分(开题材料)编排顺序依次是文献综述和开题报告封面、指导教师对文献综述和开题报告具体要求、目录、文献综述、开题报告、外文翻译和外文原文、《浙江大学本科生文献综述和开题报告考核表》。

本科生毕业论文(设计)工作文档的纸质版,可作为院(系)教学资料存档保存,参照上述编排顺序,打印装订成册,其中封面、题名页、承诺书、致谢、摘要、《浙江大学本科生毕业论文(设计)任务书》、《浙江大学本科生毕业论文(设计)考核表》、指导教师对文献综述和开题报告具体要求、《浙江大学本科生文献综述和开题报告考核表》应单面打印,其它部分内容应双面打印;主体部分各章之间应分页。论文检测报告、浙江大学本科生毕业论文(设计)专家评阅意见、浙江大学本科生毕业论文(设计)现场答辩记录表等文档可作为附件单面打印装订在最后面。

本科生毕业论文(设计)工作文档的电子版,其内容中应不包含指导性、评价性及成绩考核等内容,如:《浙江大学本科生毕业论文(设计)任务书》、《浙江大学本科生毕业论文(设计)考核表》、指导教师对文献综述和开题报告具体要求、《浙江大学本科生文献综述和开题报告考核表》、论文检测报告、浙江大学本科生毕业论文(设计)专家评阅意见、浙江大学本科生毕业论文(设计)现场答辩记录表等。

#### 1.1 前置部分

- (1) 封面
- (2) 题名页(可根据需要)
- (3) 承诺书
- (4) 勘误页(可根据需要)
- (5) 致谢
- (6) 摘要页

- (7) 序言或前言(可根据需要)
- (8) 目次页
- (9) 图和附表清单 (可根据需要)
- (10) 符号、标志、缩略词、首字母缩写、计量单位、术语 等的注释表 (可根据需要)

## 1.2 主体部分

- (1) 引言 (绪论)
- (2) 正文
- (3) 结论

## 1.3 结尾部分

- (1) 参考文献
- (2) 附录(可根据需要)
- (3) 分类索引、关键词索引(可根据需要)
- (4) 作者简历

# 2 编写规范与要求

## 2.1 语种要求

论文撰写语种，对于国际学生，参照 2017 年教育部、外交部、公安部联发的第 42 号令《学校招收和培养国际学生管理办法》执行；非国际学生遵照国家相关法律法规执行。

## 2.2 前置部分

### 2.2.1 封面

**作者学号：**全日制学生需要填写学号。

**论文题目：**应准确概括整个论文的核心内容，简明扼要，一般不能超过 25 个汉字，英文题目翻译应简短准确，一般不应超过 150 个字母，必要时可以加副标题。

### 2.2.2 承诺书

见浙江大学本科生毕业论文（设计）承诺书。

### 2.2.3 致谢

致谢对象限于对课题工作、毕业论文（设计）完成等方面有较重要帮助的人员。

### 2.2.4 摘要

包括中文摘要和英文摘要两部分。摘要应具有独立性和自含性，即不阅读论文全文就能获得必要信息。摘要的内容应包含与论文等同量的主要信息，供读者确定有无必要阅读

全文，也可供二次文献采用。摘要应说明研究目的、方法、结果和结论等，重点是结果和结论。不宜使用图、表、化学结构式、非公知公用的符号和术语。中文摘要的字数一般为300-600字以内，英文摘要实词在300个左右。英文摘要应与中文摘要内容相对应。摘要最后另起一行，列出3-8个关键词。关键词应体现论文特色，具有语义性，在论文中有明确的出处，并应尽量采用《汉语主题词表》或各专业主题词表提供的规范词。

### 2.2.5 序言或前言

毕业论文（设计）的序言或前言，一般是作者对本篇论文基本特征的简介，如说明研究工作缘起、背景、主旨、目的、意义、编写体例，以及资助、支持、协作经过等。这些内容也可在正文引言中说明。

### 2.2.6 目次页

论文中内容标题的集合。

### 2.2.7 图和附表清单

论文中如图表较多，可以分别列出清单置于目次页之后。图的清单应有序号、图题和页码。表的清单应有序号、表题和页码。

### 2.2.8 符号、标志、缩略词、首字母缩写、计量单位、术语等的注释表。

## 2.3 主体部分

包括引言（绪论）、正文和结论。

### 2.3.1 一般要求

#### 2.3.1.1 引言（绪论）

应包括论文的研究目的、流程和方法等。论文研究领域的历史回顾、文献回溯、理论分析等内容，应独立成章，用足够的文字叙述。

#### 2.3.1.2 正文

主体部分由于涉及不同的学科，在选题、研究方法、结果表达方式等有很大的差异，不能作统一的规定。但是，必须实事求是、客观真切、准备完备、合乎逻辑、层次分明、简练可读。

**图：**图包括曲线图、构造图、示意图、框图、流程图、记录图、地图、照片等。图应具有“自明性”；图宜有图题，即名称，置于图的编号后。图的编号和图题置于图下方；照片图要求主题和主要显示部分的轮廓鲜明，便于制版。如用放大缩小的复制品，必须清晰，反差适中。照片上应有表示目的物尺寸的标度。

**表：**表应具有“自明性”。表宜有表题，即表的名称，置于表的编号之后。表的编号和表题应置于表上方。表的编排，一般是由左至右横读，数据依序竖读。

## 毕业论文（设计）文献综述和开题报告考核

导师对开题报告、外文翻译和文献综述的评语及成绩评定：

此项研究为探索性问题、难题。该开题报告可通过。希望在后期抓紧努力，完成强化多智能体博弈强化学习方法的训练，争取有明确的进展，同时希望在数学表示完整性上能有加强。

成绩比例	文献综述 占 (10%)	开题报告 占 (15%)	外文翻译 占 (5%)
分值	8	14	4

导师签名 \_\_\_\_\_

年 月 日

学院盲审专家对开题报告、外文翻译和文献综述的评语及成绩评定：

论文选取研究强化学习与博弈论的结合来改进智能体强化学习算法，并构建算法验证平台，选题具有创新性，并有一定难度。文献综述表明作者对于强化学习、博弈论，以及多智能体、仿真平台等都有了较清楚的理解。研究目标和任务清楚，开题报告和译文行文通顺，技术路线调研充分。同意开始下一阶段的工作。

成绩比例	文献综述 占 (10%)	开题报告 占 (15%)	外文翻译 占 (5%)
分值	8	13	4

开题报告审核负责人（签名/签章）\_\_\_\_\_

年 月 日