同济大学

**TONGJI UNIVERSITY**

# 深度学习期中作业报告

| | |
|---|---|
| 作业名称 | QMIX |
| 学　　院 | 电子与信息工程学院 |
| 专　　业 | 数据科学与大数据技术 |
| 学生姓名 | 朱沙桐　张铭锐 |
| 学　　号 | 2151131　2152032 |
| 日　　期 | 2024 年 4 月 22 日 |

# 目 录

1 背景 …… 1
1.1 什么是 Dec-POMDP …… 1
1.1.1 马尔科夫性质与马尔科夫过程——MP …… 1
1.1.2 马尔科夫决策过程——MDP …… 1
1.1.3 部分可观测马尔科夫决策过程——POMDP …… 1
1.1.4 Dec-POMDP …… 1
1.1.5 从 Dec-POMDP 到 MARL …… 2
1.2 CTDE 的概念 …… 2
1.2.1 综述（两个极端） …… 3
1.2.2 中心式训练 …… 3
1.2.3 分布式执行 …… 3
2 相关工作（前置知识） …… 4
2.1 Deep $Q$-Learning …… 4
2.2 价值函数分解 …… 4
2.3 Value Decomposition Networks …… 4
3 QMIX 理论分析与实现 …… 5
3.1 原理 …… 5
3.2 网络 …… 5
3.3 智能体网络（Agent Networks） …… 5
3.4 混合网络（Mixing Network） …… 6
3.5 超网络（Hypernetworks） …… 6
3.6 训练 …… 7
3.7 分析（PPT 不写缺点，就写优点） …… 7
4 代码仓库与贡献 …… 9

装 订 线

# 1 背景

本次分享的论文 QMIX 是多智能体强化学习 MARL 中的经典之作。

谈到强化学习，我们自然会想到马尔科夫决策过程 MDP。在实际中，智能体往往不能全部了解全部的 State，所以引入 POMDP。而 MARL 中，涉及多智能体，引入 Dec-POMDP。

## 1.1 什么是 Dec-POMDP

本小节主要介绍基本概念

### 1.1.1 马尔科夫性质与马尔科夫过程——MP

马尔科夫性质（Markov Property）是指系统的下一状态仅依赖于当前状态，而与之前的历史状态无关。

基于这一性质的随机过程被称为马尔科夫过程（Markov Process, MP），通常表示为元组 $(S, P)$，其中 $S$ 是状态空间，$P$ 是状态转移概率矩阵。马尔科夫过程的一个典型例子是随机游走：在每一步，系统的状态转移仅依赖于当前位置，而与之前的路径无关。该特性简化了过程的分析与预测，因为只需考虑当前的状态即可。

### 1.1.2 马尔科夫决策过程——MDP

在马尔科夫过程的基础上，马尔科夫决策过程（Markov Decision Process, MDP）引入了决策者的行为选择。MDP 由元组 $(S, A, P, R)$ 定义，其中 $A$ 为行动空间，$R$ 为奖励函数。与简单的马尔科夫过程不同，MDP 考虑了每个状态下可以执行的行动以及执行特定行动后的奖励。例如，在一个迷宫寻路问题中，每个位置（状态）不仅需要决定下一步的移动方向（行动），还要考虑因行动得到的即时奖励或惩罚。决策者的目标是找到一策略 $\pi$，使得从任何初始状态出发，长期累积奖励最大化。

### 1.1.3 部分可观测马尔科夫决策过程——POMDP

部分可观测马尔科夫决策过程（Partially Observable Markov Decision Process, POMDP）进一步发展了 MDP 的模型，允许决策者不能完全观测到系统的真实状态。POMDP 由元组 $(S, A, P, R, O, \Omega)$ 定义，其中 $O$ 是观测函数，$\Omega$ 是观测空间。在 POMDP 中，决策者根据其观测到的信息，需要维护一个信念状态（状态的概率分布），并基于这一信念状态来做出决策。例如，在自动驾驶汽车中，传感器可能无法完全准确地捕捉周围环境的信息，因此需要在部分可观测的信息上进行决策制定。

### 1.1.4 Dec-POMDP

一个合作的多智能体任务可以被建模为一个去中心化的部分可观测马尔可夫决策过程（Dec-POMDP），由元组 $< S, A, O, \Omega, P, R, n, \gamma >$ 定义。这与传统的部分可观测马尔可夫决策过程（POMDP）相比，最大的区别在于，Dec-POMDP 考虑了多智能体的协作，而不是单一智能体的决策。这种模

型的复杂性显著增加，因为它必须处理联合行动和策略空间的增长。

在 Dec-POMDP 模型中，$s \in S$ 是全局状态。每个智能体 $i$，在执行与其他 $n-1$ 个智能体的联合行动 $a = \{a_1, ..., a_n\}$ 后，根据状态转移函数 $P(s'|s, a)$ 从 $s$ 转移到下一个时间步骤的状态 $s'$，并且所有智能体获得一个共享奖励 $r = R(s, a, s')$。在部分可观测的情况下，每个智能体只能根据观测函数 $O(o_i|s)$ 获得部分环境的观测 $o_i \in \Omega$。

关键的概念在于每个智能体具有一个个体策略 $\pi_i(a_i|\tau_i)$，其中 $\tau_i$ 是行动-观测历史，$\tau = \{\tau_1, ..., \tau_n\}$。每个策略 $\pi_i$ 依赖于观测历史，而不是完整的状态信息，这体现了部分可观测性的核心挑战。解决 Dec-POMDP 的目标是找到一个最优的联合策略 $\pi = \{\pi_1, ..., \pi_n\}$，以最大化联合价值函数 $V^{\pi}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$，其中 $\gamma$ 是未来奖励的折扣因子。

由于多智能体的存在，每个智能体的策略不仅要处理自身的部分可见性，还需要协调与其他智能体的策略，这使得策略设计和优化成为一个非常复杂的问题。此外，由于信息的不对称性和每个智能体观测的局限性，整体系统的动态预测与控制变得更为困难。

#### 1.1.5 从 Dec-POMDP 到 MARL

去中心化的部分可观测马尔科夫模是研究不确定性情况下多主体协同决策的重要模型。由于其求解难度是 NEXP-complete，迄今为止尚没有有效的算法能求出其最优解，但是可以用强化学习来近似求解。

从去中心化的部分可观测马尔科夫决策过程（Dec-POMDP）到多智能体强化学习（Multi-Agent Reinforcement Learning, MARL）的转换体现了从理论模型到实际应用的演进。在 MARL 中，每个智能体通过与环境交互来学习最优策略，这一过程通常涉及到从试错中学习以适应复杂的、动态变化的环境。

MARL 扩展了传统的单智能体强化学习框架，使其能够处理多个智能体在共享环境中的交互。每个智能体都尝试通过选择行为来最大化其累积奖励，同时还必须考虑其他智能体的行为对环境的影响。处理 Dec-POMDP 模型，MARL 的核心挑战在于每个智能体的决策仅基于其部分观测的环境状态，而这些状态又受到其他智能体行为的影响。

### 1.2 CTDE 的概念

本小节，主要介绍论文的训练与执行方式。训练，就是与环境交互来学习策略；执行，就是利用策略来实现目标。

在多智能体系统的强化学习领域，”中心式训练，分布式执行”（CTDE）的方法逐渐成为一种主流的学习范式。

#### 1.2.1 综述（两个极端）

在多智能体系统中，强化学习算法的设计通常需要在独立操作和集中协调之间寻找平衡。其中，Independent Q-Learning（IQL）和 Counterfactual Multi-Agent Policy Gradients（COMA）是两种极端的代表性算法，它们展示了这种平衡的不同取向。

IQL 是一种非常直接的方法，每个智能体都独立地学习和更新其自己的动作价值函数，这意味着每个代理评估其动作的价值而不考虑其他智能体的存在或策略。这种方法的优点在于其简单性，使得实现和理解都较为容易。然而，IQL 的主要限制在于它不支持智能体之间的有效协调。由于每个智能体都在独立操作，整个环境对于单个智能体而言是非固定的，这可能导致学习过程难以收敛，尤其是在需要智能体合作时。

另一方面，COMA 采用了一种更为集中的方法，通过学习一个全局的状态-动作价值函数来指导每个智能体的策略。在通常的演员-评论家框架下，每个智能体的策略（演员）由一个集中式的评论家通过评估这些策略来进行指导。这种方法尽管在理论上可以提高智能体间的协调能力，但也带来了显著的局限性。首先，COMA 通常要求基于策略的学习，这意味着需要不断地获取新数据来更新策略，从而可能导致样本效率低下。其次，随着代理数量的增加，维护和训练一个全面的集中式评论家变得非常复杂和计算量大，尤其是当涉及到评估所有可能的代理组合和动作时。

在此背景下，提出 CTDE。

#### 1.2.2 中心式训练

利用集中式学习环境在训练阶段共享信息，以便于在实际执行时能够进行有效的分布式操作。

集中式学习环境提供了额外的优势，一个是能够访问通常对智能体隐藏的更全面的状态信息 S，一个是能够在智能体间自由进行通信，这些在实际的去中心化执行环境中可能是不可行的。

#### 1.2.3 分布式执行

然而，集中式训练也面临着挑战，尤其是当智能体的数量增加时，联合动作空间会指数级增长，这使得学习一个全局的动作值函数变得极为复杂。这种函数需要决定整个系统的状态以及所有智能体的联合动作，随着智能体数量的增多，学习难度急剧增加。

为了解决这些问题，分布式执行策略被采用，在这种策略下，每个智能体在执行时都依赖于自己的策略，不需要整个系统的全局信息。

这种方法不仅保证了系统的灵活性和可扩展性，还减轻了因智能体数量增加而带来的联合行动空间扩张的问题。通过这种去中心化的执行方式，智能体只依赖于自身的行动和观察历史，而无需依赖其他所有智能体的状态和行动，从而简化了系统的管理和维护，提高了执行效率。

# 2 相关工作（前置知识）

## 2.1 Deep $Q$-Learning

深度 $Q$ 学习使用深度神经网络（Deep Neural Network, ($\theta$ 参数化）来表示动作价值函数。深度 $Q$ 网络（Deep Q-Networks, DQN）利用一个 回放缓冲区（replay memory）存储状态-动作-奖励-新状态四元组 $\langle s, u, r, s' \rangle$，其中在状态 $s$ 中采取动作 $u$ 后，观察到新状态 $s'$ 并获得奖励 $r$。通过从回放缓冲区随机抽取大小为 $b$ 的转储，网络通过最小化 时间差分误差（TD error）的平方来学习 $\theta$ 参数。

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[ \left( y_i^{\text{DQN}} - Q(s, u; \theta) \right)^2 \right], \tag{2.1}$$

其中，$y^{\text{DQN}} = r + \gamma \max_{u'} Q(s', u'; \theta^-)$。$\theta^-$ 是一个目标网络（target network）的参数，它们定期从 $\theta$ 复制而来，并在一段时间内保持不变。

（QMIX 的训练采用类似的方式）

## 2.2 价值函数分解

价值函数分解（或价值分解）是在 CTDE 范式中最常用的方法之一。QMIX 就是 MARL-值分解方法下的曾经的 SOTA。

在训练期间提供全局状态信息，以获得更准确的联合行动价值函数 $Q_{tot}(\tau, a)$，而个体行动价值函数 $Q_i(\tau_i, a_i)$ 仅接收其自己的观测，以实现完全去中心化。$Q_{tot}$ 通过一个混合函数 $f_{mix}$ 分解成 $Q_i$:

$$Q_{tot} = f_{mix}(Q_1(o_1, a_1), ..., Q_n(o_n, a_n)) \tag{2.2}$$

## 2.3 Value Decomposition Networks

在这样的架构下，先前的价值分解网络（Value Decomposition Networks，VDNs）将 $Q_{tot}$ 表示为每个个体价值函数 $Q_a(\tau^a, u^a; \theta^a)$ 的和，每个函数仅依赖于单个 agent $a$ 的动作-观察历史:

$$Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \sum_{i=1}^{n} Q_i(\tau^i, u^i; \theta^i). \tag{2.3}$$

严格来说，每个 $Q_a$ 是一个效用函数（utility function，即效用函数）而不是价值函数，因为它本身并不估计期望回报。然而，为了术语简洁，我们称 $Q_{tot}$ 和 $Q_a$ 为价值函数。

VDN 的损失函数等同于 公式（2.1），其中 $Q$ 被替换为 $Q_{tot}$。这种表示的优点是，一个去中心化的策略可以从每个代理根据其 $Q_a$ 进行贪婪动作选择时自然产生。

# 3 QMIX 理论分析与实现

在这个部分，我们开始介绍 QMIX。它介于 IQL 和 COMA 的两个极端之间，但能够表示更丰富的动作价值函数。

## 3.1 原理

方法的关键在于，为了能够提取与集中式策略完全一致的分散策略，完全分解 VDN（Value Decomposition Network）并不是必需的。相反，为了保持一致性，我们只需要确保对 $Q_{tot}$ 进行全局 argmax 操作的结果，与对每个 $Q_a$ 进行单独 argmax 操作的结果相同：

$$\operatorname{argmax}_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \operatorname{argmax}_{u^1} Q_1(\tau^1, u^1) \\ \vdots \\ \operatorname{argmax}_{u^n} Q_n(\tau^n, u^n) \end{pmatrix}. \tag{3.1}$$

这使得每个代理 $a$ 仅通过根据其 $Q_a$ 选择贪心行动，就能参与到去中心化的执行中。作为副作用，如果满足公式（3.1），那么执行 off-policy 学习更新时所需的 $Q_{tot}$ 的 argmax 计算变得简单易行。

（个体-全局最大（IGM）条件）

VDN 的表示足以满足公式（3.1）。然而，QMIX 是基于这样一个观察：这种表示可以推广到满足公式（3.1）的更广泛的单调函数家族中，这些函数既充分又非必要。单调性可以通过对 $Q_{tot}$ 与每个 $Q_a$ 之间的关系施加约束来实现：

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \ \forall a \in A. \tag{3.2}$$

## 3.2 网络

为了实现公式（3.2），QMIX 使用了一个由智能体网络（agent networks）、混合网络（mixing network）和一组超网络（hypernetworks）组成的架构。图3.1展示了整体架构。

## 3.3 智能体网络（Agent Networks）

对于每个智能体 $a$，有一个代表其个体价值函数 $Q_a(\tau^a, u^a)$ 的智能体网络。我们将智能体网络表示为 DRQN，它们在每个时间步接收当前个体观察 $o_t^a$ 和上一步动作 $u_{t-1}^a$ 作为输入，如图3.1c 所示。

这部分与其他多智能体强化学习方法类似，智能体网络独立地处理各自的观测数据。

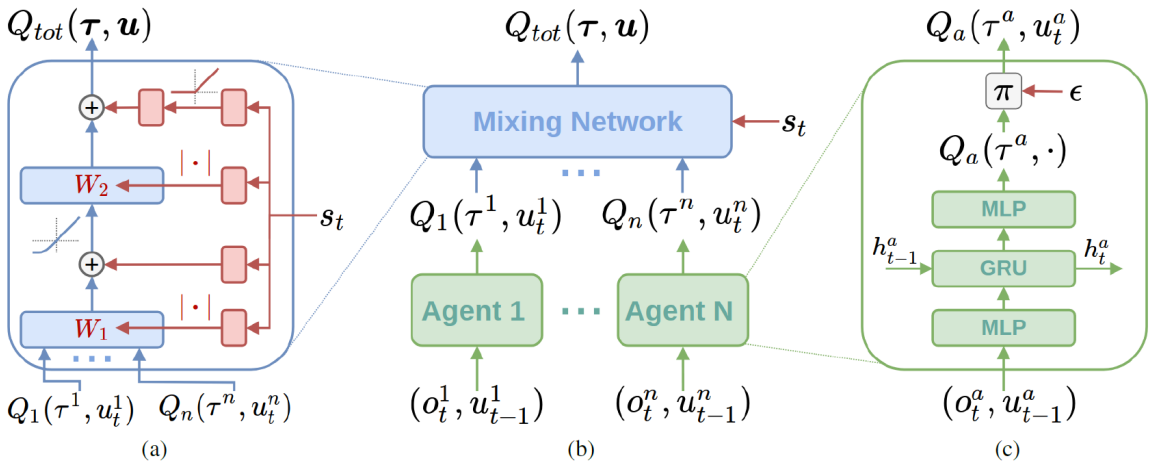


*Figure 2.* (a) Mixing network structure. In red are the hypernetworks that produce the weights and biases for mixing network layers shown in blue. (b) The overall QMIX architecture. (c) Agent network structure. Best viewed in colour.

图 3.1 QMIX 智能体网络

### 3.4 混合网络（Mixing Network）

混合网络是一个前馈神经网络，它接受智能体网络的输出作为输入，并单调地混合它们，产生 $Q_{tot}$ 的值，如图3.1a 所示。为了实现公式（3.2）中的单调性约束，混合网络的权重（但不是偏置）被限制为非负。

不同于 VDN 的简单线性组合，QMIX 的混合网络能够以更复杂的非线性方式来结合这些输出，从而允许全局价值函数 Qtot 捕捉智能体间更复杂的相互作用。由于混合网络是单调函数的通用函数逼近器，因此它能表示任何可以分解为代理个体价值函数的非线性单调组合的价值函数。

### 3.5 超网络（Hypernetworks）

混合网络的权重由单独的超网络生成。每个超网络接收状态 $s$ 作为输入，并生成混合网络一层的权重。每个超网络由一个单层线性层和绝对激活函数组成，以确保混合网络的权重为非负。超网络的输出是一个向量，然后被重塑为适当大小的矩阵。偏置是通过类似的方式生成的，但不被限制为非负。最终的偏置由一个具有 ReLU 非线性的两层超网络产生。图3.1a 展示了混合网络和超网络。

**A. 处理额外状态信息**

状态由超网络使用，而不是直接传递给混合网络，因为 $Q_{tot}$ 被允许以非单调的方式依赖于额外的状态信息。因此，直接将 $s$ 的某种函数与每个智能体的价值一起通过单调网络可能会过于约束。相反，使用超网络使得能够根据 $s$ 以任意方式条件化混合网络的权重，从而尽可能灵活地将整个状态 $s$ 整合到联合动作价值估计中。

### 3.6 训练

QMIX 被端到端训练，以最小化以下损失：

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[ \left( y_i^{tot} - Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, s; \theta) \right)^2 \right], \tag{3.3}$$

其中，$b$ 是从回放缓冲区中采样的转换批次大小，$y^{tot} = r + \gamma \max_{\mathbf{u}'} Q_{tot}(\boldsymbol{\tau}', \mathbf{u}', s'; \theta^-)$，$\theta^-$ 是 DQN 中目标网络的参数。公式（3.3）类似于标准 DQN 损失公式（2.1）。由于公式（3.1）成立，我们可以线性时间（而非在最坏情况下指数级增长）内最大化 $Q_{tot}$。

### 3.7 分析（PPT 不写缺点，就写优点）

QMIX 所表示的价值函数类别包括在完全可观测情况下，可以分解为各个智能体个体价值函数的非线性单调组合的价值函数。这扩展了 VDN 可以表示的线性单调价值函数。然而，公式（3.2）中的约束阻止 QMIX 表示非以这种方式分解的价值函数。

直观来说，如果一个智能体的最佳行动取决于同一时间步上其他智能体的行动，那么这样的价值函数将无法适当分解，因此 QMIX 无法完美表示。然而，QMIX 可以比 VDN 更准确地近似这样的价值函数。

**算法 1** QMIX 算法

初始化网络参数 $\theta$ 用于 Q 函数和 $\phi$ 用于混合网络
用 $\theta$ 和 $\phi$ 初始化目标网络参数 $\theta^-$ 和 $\phi^-$
初始化重放缓冲区 $\mathcal{B}$
**for** 每个剧集 **do**
    初始化环境并获取初始状态 $s$
    **while** 未结束 **do**
        **for** 每个智能体 $i$ **do**
            以 $\epsilon$ 的概率选择随机动作 $a_i$，否则选择 $a_i = \arg\max_a Q_i(s, a; \theta)$
        **end for**
        执行动作 $\mathbf{a} = (a_1, ..., a_n)$，观察奖励 $r$ 和新状态 $s'$
        在重放缓冲区 $\mathcal{B}$ 中存储转移 $(s, \mathbf{a}, r, s')$
        从 $\mathcal{B}$ 中随机抽取一个小批量转移 $(s, \mathbf{a}, r, s')$
        **for** 每个智能体 $i$ **do**
            计算 $Q_i(s, a_i; \theta)$
        **end for**
        计算总 Q-值 $Q_{\text{tot}}(s, \mathbf{a}; \theta, \phi) = f_{\text{mix}}(Q_1, ..., Q_n; \phi)$

计算目标 Q-值 $y = r + \gamma Q_{\text{tot}}(s', \arg\max_{\mathbf{a}'} Q_{\text{tot}}(s', \mathbf{a}'; \theta, \phi); \theta^-, \phi^-)$

通过最小化损失更新 $\theta$ 和 $\phi$: $\mathcal{L}(\theta, \phi) = (y - Q_{\text{tot}}(s, \mathbf{a}; \theta, \phi))^2$

定期更新目标网络: $\theta^- \leftarrow \theta, \phi^- \leftarrow \phi$

**end while**

**end for**

装 订 线

## 4 代码仓库与贡献

repo 地址：https://github.com/Zhu-Shatong/QMIX_pytorch

A. 2151131 **朱沙桐**

论文理论阅读与推导

代码复现

环境配置与实验

报告撰写

(50%)

B. 2152032 **张铭锐**

综述与背景

代码复现

PPT 制作

讲稿撰写

(50%)