MARL 值分解算法补充材料

符号表

> 注意：在本文中，统一用上标表示智能体索引 i ，下标表示时间 t ，有时会省略时间下标 t 。

名称	符号
智能体 i 在时刻 t 的局部观察	o_t^i
时刻 t 所有智能体的全局状态	$\boldsymbol{s_t} = (o_t^1, ..., o_t^n)$
智能体 i 在时刻 t 的动作	u_t^i
时刻 t 所有智能体的联合动作	$\boldsymbol{u}_t = (u_t^1, ..., u_t^n)$
智能体 i 的动作-观察联合历史	$\tau^i = (u_0^i, o_1^i, \cdots, u_{t-1}^i, o_t^i)$
所有智能体的全局（联合）动作-观察联合历史	$\boldsymbol{\tau} = (\tau^1, \cdots, \tau^n)$
智能体 i 的局部动作值函数	$Q^i(\cdot)$
全局（联合）动作值函数	$Q^{total}(\cdot)$
智能体 i 的局部值函数	$V^i(\cdot)$
全局（联合）值函数	$V^{total}(\cdot)$
单个智能体 i 的优势函数	$A^i(\cdot)$
全局（联合）优势函数	$A^{total}(\cdot)$

理论框架

Centralized Training Decentralized Execution（CTDE） 框架

采用强化学习来解决多智能体协作问题，最基础的训练框架有两种，即**中心化 （Fully Centralized）**和**去中心化 （Fully Decentralized）** 的训练框架。本小节将从分析两者的定义和问题入手，最后引出 CTDE 训练框架的核心思想和设计。

去中心化

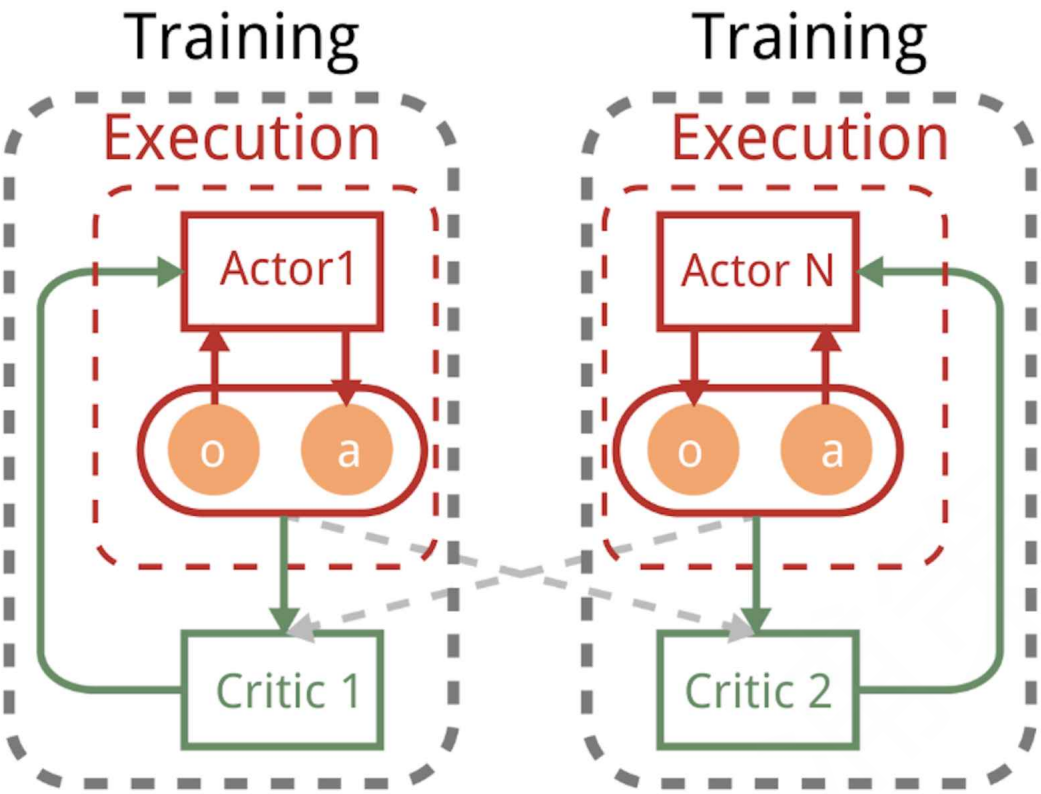
- **定义**：这种框架是单智能体算法的简单拓展，每个智能体的训练独立于其他智能体，彼此之间没有信息共享和交流，即简单地将单智能体算法分别应用在多个智能体上。一个经典的例子就是 IQL [5]（independent Q-learning），具体来说，每个智能体单独学习一个值网络 $Q^i(o^i, u^i)$，将局部观测作为输入，得到每个局部动作的 Q 值，然后通过 $u^i = argmax_{u^i} Q^i(o^i, u^i)$ 选择要采取的动作。
- **缺陷**：去中心化的训练框架所面临最大的问题就是**环境非平稳**。在多智能体环境中，由于智能体之间的相互影响，智能体 i 在观察 o^i 下执行动作 u^i 后得的 r^i 与 $o^{i'}$ 是由**所有**智能体的行为造成的，即 $r^i = R^i(\boldsymbol{s}, \boldsymbol{u})$，$o^{i'} = T^i(\boldsymbol{s}, \boldsymbol{u})$。所以对于单个智能体 i 而言，即使在观察 o^i 下执行同样的动作 u^i，但是由于 $\boldsymbol{s}$ 未知且其他智能体的策略在不断变化，因此执行动作和状态转移得到的 r^i 与 $o^{i'}$ 可能是不同的。这就是多智能体协作中的不稳定性，即 reward 与 transition 存在不稳定性，影响了强化学习算法遵循的标准马尔可夫假设，进而导致智能体 i 的值函数 $Q^i(o^i, u^i)$ 的学习就会存在各种问题，无法简单复用单智能体算法。
- **例子**：去中心化范式下最典型的算法是 IQL 。IQL 算法中将其余智能体直接看作环境的一部分，对于每个智能体来说，就看作是在解决一个单智能体任务，每个智能体执行一个 DQN 算法。由于上文中提到的环境非稳态问题，这种方法很难保证收敛性，智能体会很容易陷入无止境的探索中。

中心化

- **定义**：这种框架会汇集所有智能体的信息去训练一个中央控制器，通过中央控制器来输出所有智能体的决策，例如训练一个全局的动作价值函数 $Q^{total}(\boldsymbol{s}, \boldsymbol{u})$ 或是全局的策略函数来实现这一目标。中心化的训练框架可以缓解环境非平稳性问题，通过整合所有信息来学习一个所有智能体的联合策略，即输入是所有智能体的联合观测，输出所有智能体的联合动作。
- **问题**：当智能体数量增大时，中心化训练框架就会面临输入和输出空间规模指数级增长的问题，因此难以适应较大规模的多智能体系统。

针对上述问题，多智能体协作领域的研究者结合上述 2 种范式的优点，提出了一种**中心化训练但去中心化执行的框架（Centralized Training Decentralized Execution，CTDE）**，代表算法如 MADDPG [12] 与 QMIX [9]，通过整合全局信息来学习一个所有智能体的联合策略，但同时保证推理时是去中心化执行。

- 中心化训练：训练的时候存在一个额外的中央控制器，它会利用所有智能体的动作、状态以及奖励等信息。基于**全局观测**进行训练，从而帮助各个智能体训练其自身的价值网络或策略网络。
- 去中心化执行：执行时不再使用中央控制器，每个智能体只是基于自己的**局部观察**，利用自己的价值网络或策略网络做决策，但由于中心化训练过程的约束，执行时各个智能体仍能保持优秀的协作，从而在一定程度上同时克服环境不平稳和智能体规模变大时中心化执行开销大的问题。



（图1：中心化训练去中心化执行的框架（Centralized Training Decentralized Execution）概述图）

Individual-Global-Max（IGM） 条件

不过，CTDE 训练框架虽然理论上兼顾了训练和执行两方面的优势，但想要高效实现基于多智能体 Q-learning 的 CTDE 框架，其中的一个关键点是：各个智能体根据 individual Q functions 选择的局部贪心动作构成的联合动作等价于根据 global/joint Q function（ Q_{joint} ）选择出的全局贪心动作。这被称为 IGM（Individual-Global-Max）条件，形式化定义如下：

$$\text{argmax}_u Q^{tot}(\tau, u) = \begin{pmatrix} \text{argmax}_{u^1} Q^1(\tau^1, u^1) \\ \vdots \\ \text{argmax}_{u^n} Q^n(\tau^n, u^n) \end{pmatrix}$$

IGM 条件为 CTDE 框架的实际应用提供了两个优势：

- 确保了中心化训练（学习联合 Q 函数）和去中心化执行（使用单个 Q 函数）的策略一致性
- 在此基础上，训练时能够使用联合 Q 函数的 TD-learning 目标进行可扩展的集中优化，而在执行时又可以从单个智能体 Q 函数组合中导出联合贪婪动作。

不过，IGM 条件并不总是成立，它只是多智能体最优协作决策的一个必要条件，实际中存在部分多智能体协作问题，并不满足这样的形式定义。

实践算法

更具体地，基于 CTDE 框架的一类经典方法就是价值函数分解类算法（Value-Decomposition），这类方法尝试寻找合理的建模方式，从而精确地**表示和分解每个独立智能体的 Q^i 与联合 Q 值 Q^{total} 之间的关系**。在训练阶段，通过中央控制器接收全局状态信息 $\boldsymbol{s}$ 并指导每个智能体的训练，尝试解决单个智能体不具全局决策能力的缺点。而在执行阶段，单个智能体根据自己的价值函数独立执行自己的动作。值分解类算法拥有非常自然的设计动机，但是，如果不能准确表示 Q^i 和 Q^{total} 之间的关系，那么这种分解的作用就会大打折扣，从而影响最终的模型性能。为了寻找最佳的分解实践方案，研究者们相继提出了 VDN、QMIX、QPLEX 等典型算法，接下来的小节将介绍它们的设计动机和优缺点。

VDN: Value-Decomposition Networks For Cooperative Multi-Agent Learning [7]

值分解基本思想

仅仅使用单个智能体的 $Q^i(o^i, u^i)$ 进行决策存在不稳定性，只有使用 $Q^{total}(\boldsymbol{s}, \boldsymbol{u})$ 才能站在全局的角度进行优化，从而解决不稳定性的问题，但是，全局价值函数 Q^{total} 无法直接用于去中心化的执行阶段，因此就出现了一系列值函数分解的方式来解决这个问题。VDN 是第一个值分解类型算法，它主要研究多智能体强化学习协作问题 （cooperative multi-agent reinforcement learning） 问题，并进一步分析了传统中心化和去中心化框架存在的以两大类问题：

- 懒智能体 （lazy agent）：在传统的合作型多智能体环境中，团队有一个共同的 reward，每个智能体没有自己单独的 reward。当完全**中心化**地训练时（即将所有的智能体结合作为一个大的智能体进行训练，其中观测空间、动作空间都是全局的），这样一个整体的 reward 就不能很好的体现出每个智能体的“贡献大小”，当有一个优秀的智能体时，可能这个智能体完成了绝大部分工作，其他智能体就会“不作为”，也就是 VDN 论文中所说的“**lazy agent**”。
- 虚假奖励 （spurious reward signals）：在完全**去中心化**训练的情景（例如 IQL），由于环境是非稳态的，理论上的收敛性是难以保证，从而优化过程变得非常不稳定，同时，每个智能体的 reward 也无法判断是由智能体自己还是由队友产生的，因此可能会产生的错误的优化信号，即论文中的“spurious reward”。

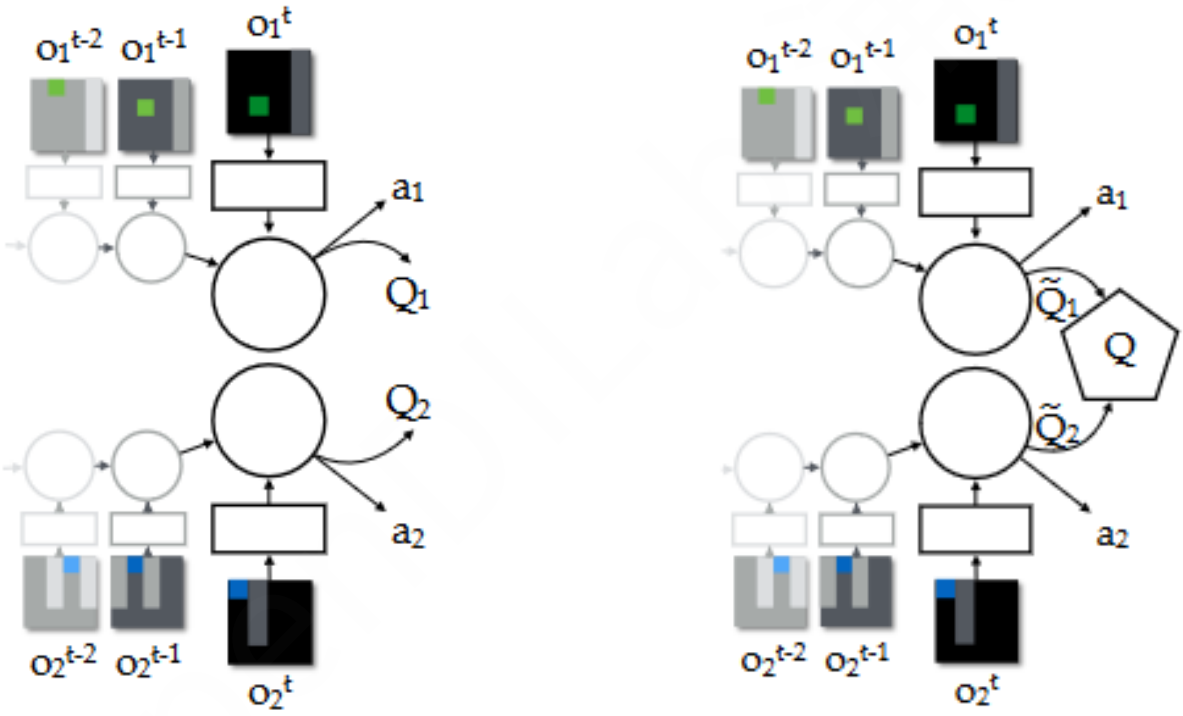
> **Credit Assignment** （信用分配）
>
> 信用分配通常分为两种类型： 自下而上类型和自上而下类型。
>
> - 自上而下类型：这种类型通常指我们只能拿到一个团队的最终得分，而无法获得每一个智能体的独立得分，因此我们需要把团队回报（Team Reward）合理的分配给每一个独立的 Agent（Individual Reward），这个过程通常也叫 “独立回报分配” （Individual Reward Assign）。上述例子就属于这种类型，典型的代表算法为 COMA 算法。
> - 自下而上类型：另外一种类型恰恰相反，指当我们只能获得每个智能体的独立回报（Individual）时，如何使得整个团队的团队得分（Team Reward）最大化。VDN、QMIX 算法解决的是上述第二种类型的问题，即在获得各 Agent 的独立回报的情况下，如何使得整个团队的团队收益最大化问题。

如果每个智能体都会获得自己对团队的贡献具体贡献值（奖励），以此为反馈信号优化各自的目标函数，就能够解决上述问题。基于这样的动机，VDN 提出了“值函数分解”的研究思路，将团队整体的值函数分解成 N 个子值函数的加法组合，分别作为各智能体执行动作的依据。

假设 $Q^{total}(\boldsymbol{s},\ \boldsymbol{u})$ 是多智能体团队的整体 Q 函数， d 是智能体个数， o^i 是智能体 i 的历史序列信息， u^i 是其动作。该 Q 函数的输入集中了所有智能体的观测和动作，可通过团队奖励 r 来迭代拟合。为了得到各个智能体的值函数，VDN 设计了直接相加求和的方式：

$$Q^{total}(\boldsymbol{s},\ \boldsymbol{u}) \approx \sum_{i=1}^{d} \tilde{Q}^i(o^i, u^i)$$

该假设表明团队的 Q 函数可以通过求和的方式近似分解成 d 个子 Q 函数，分别对应 d 个不同的智能体，且每个子 Q 函数的输入为该对应智能体的**局部观测序列和动作**，互相不受影响，如下右图所示：



（图 2：（左）完全去中心化的训练框架，单个智能体根据局部观察 o_i^t 得到各自的 Q 值 Q_i。（右）值分解框架，单个智能体各自的局部观察 o_i^t 得到各自的 Q 值 $\tilde{Q}_i$，并对这些 Q 值进行汇总得到一个总体的 Q 值。注意：图中的符号下标为智能体索引，上标为时间步。）

这样，每个智能体就有了自己的值函数，且明显这样简单的加和关系满足 IGM 条件：

$$\text{argmax}_{\boldsymbol{u}} Q^{total}(\boldsymbol{s}, \boldsymbol{u}) = \begin{pmatrix} \text{argmax}_{u^1} Q^1(o^1, u^1) \\ \vdots \\ \text{argmax}_{u^n} Q^n(o^n, u^n) \end{pmatrix}$$

因此，在执行阶段它们就可以根据自己的局部值函数来进行决策了：

$$u^i = \arg\max_{u^{i'}} \tilde{Q}^i(o^i, u^{i'})$$

近似得到 $Q^{total}(\boldsymbol{s},\ \boldsymbol{u})$ 之后，VDN 使用类似单智能体 DQN 的更新方式，通过全局奖励 r 来更新 $Q^{total}(\boldsymbol{s},\ \boldsymbol{u})$，其损失函数表示为：

$$L(\theta) = \frac{1}{M}\sum_{j=1}^{M}(y_j - Q^{total}(\boldsymbol{s},\ \boldsymbol{u}))$$

$$y_j = r_j + \gamma \arg\max_{\mathbf{u}} \bar{Q}^{total}(\boldsymbol{s}',\ \boldsymbol{u})$$

$$\bar{Q}^{total}(\mathbf{s},\ \mathbf{u}) = \sum_{i=1}^{d}\bar{Q}^i(o^i, u^i)$$

其中 M 为 batch size， $\bar{Q}^i$ 为 target net，用于抑制价值函数的过估计问题。

QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning [9]

VDN 的问题

值分解的基本思想是把全局 Q 函数近似分解成 d 个子 Q 函数，分解的方式代表了我们预设的每个智能体对总体奖励的关系。VDN 认为 $Q^{total}(\mathbf{s},\ \mathbf{u})$ 和 $Q^i(o^i, u^i)$ 之间的关系可以用求和表示，通过**累加和**来近似 $Q^{total}(\mathbf{s},\ \mathbf{u})$。但是求和是一种很简单的关系，它不能表示复杂的组合关系，如果实际问题中 $Q^{total}(\mathbf{s},\ \mathbf{u})$ 和 $Q^i(o^i, u^i)$ 的**关系很复杂**，那么 VDN 就无法建模这种类型的协作。更重要的是，VDN 在中心化训练时并**没有用到全局状态 $\mathbf{s}$**，并没有很好地利用中心化训练的优势。

QMIX 的思想

QMIX 是基于 VDN 的一种拓展，由于 VDN 只是将每个智能体的局部动作值函数求和相加得到联合动作值函数，虽然满足联合值函数与局部值函数单调性相同的可以进行分布化策略的条件，但是如上面所讨论，其没有在学习时利用全局状态信息以及没有采用非线性方式对单智能体局部值函数进行整合，使得 VDN 算法还有很大的提升空间。

QMIX 的主要思想就是利用神经网络 f 去学习 Q^{total} 与 $[Q^1, \ldots, Q^N]$ 之间的复杂关系，并在训练学习过程中利用全局状态信息来提高算法性能，建模更复杂的协作关系。

- 神经网络近似

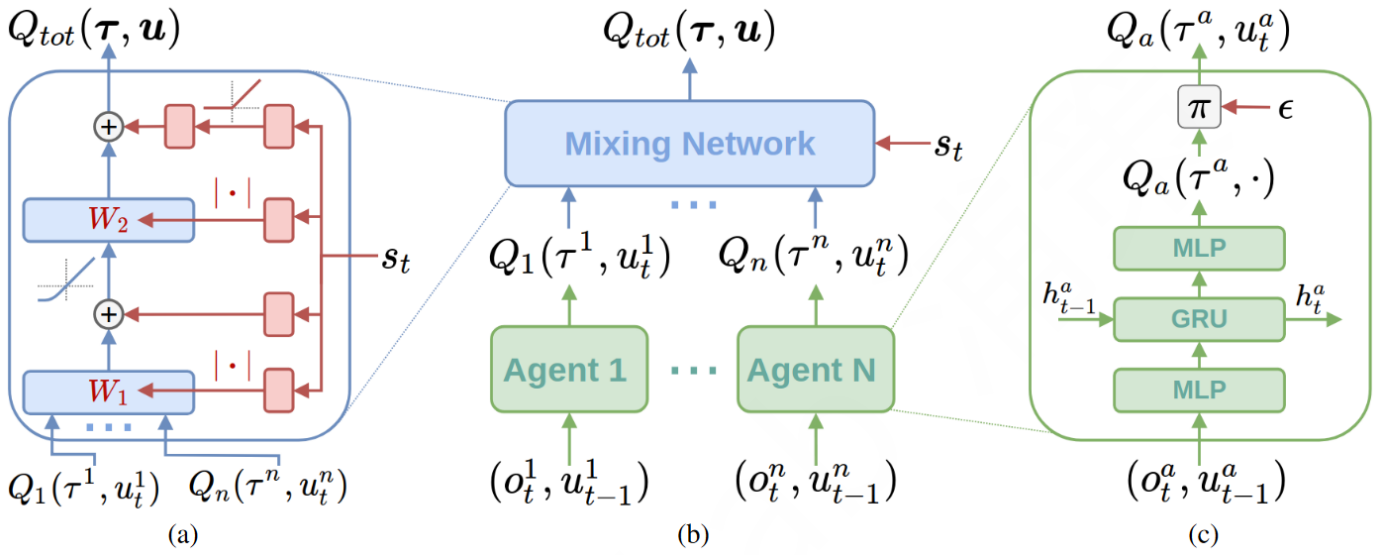
  在 QMIX 中，神经网络 f 学到的 Q^{total} 与 $[Q^1, \ldots, Q^N]$ 之间关系仍然要满足 IGM 条件，即对联合动作值函数取 $argmax$ 等价于对每个局部动作值函数取 $argmax$ ，其单调性相同。QMIX 在 VDN 的基础上推广到更大范围的单调函数族， Q^{total} 和 Q^i 满足如下单调性约束则满足 IGM 条件：

  $$\frac{\partial Q^{total}}{\partial Q^i} \geq 0, \forall i \in \{1, 2, \cdots, n\}$$

  具体实现中，QMIX 通过限制组合函数 f 中的参数全部非负来保证满足此条件。

- 利用全局状态信息

QMIX 在近似 Q^{total} 时额外使用了全局状态 $\mathbf{s}$ ，这样就可以充分利用全局状态 $\mathbf{s}$ 进行训练。但是为了保证单调性，我们希望 $[Q^1, \ldots, Q^N]$ 对应的参数都是**非负的**，如果直接朴素地将 $\mathbf{s}$ 和 $[Q^1, \ldots, Q^N]$ 一起输入到神经网络 f 去得到 Q^{total} ，这样做会隐含限制 Q^{total} 和 $\mathbf{s}$ 的关系为 $\frac{\partial Q^{total}}{\partial \mathbf{s}} \geq 0$ ，这和 IGM 条件，而且全局状态和单个智能体价值函数之间的组合会较为复杂难以优化。所以 QMIX 额外设计了混合网络 （Mixing Network） 来实现这一机制，使用全局信息来产生组合单智能体价值函数的相关参数，其具体结构如下所示：



（图 3： （a） 混合网络结构。红色为超网络 （hypernetworks） ，输入为全局状态 s_t ，输出为混合网络蓝色部分中使用的权值 （weight） 及偏移量 （bias）。（ b ） QMIX 算法总体网络架构。（ c ） 单个智能体网络结构，使用建模时序信息的 GRU 结合 MLP 构成最终的模型。）

因为 QMIX 中使用 GRU 来建模时序信息，即是一个含有 RNN 的 DQN（DRQN） ，所以这里额外补充一些符号定义：

令 $\boldsymbol{\tau} = (\tau^1, \cdots, \tau^n)$ 表示联合动作-观测历史轨迹，其中 $\tau^i = (u_0^i, o_1^i, \cdots, u_{t-1}^i, o_t^i)$ 为某个智能体的动作-观测历史轨迹。 $\boldsymbol{u}_t = (u_t^1, \cdots, u_t^n)$ 表示联合动作。 Q^{total} 为联合动作值函数，$Q^i(\tau^i, u^i; \theta_i)$ 为智能体 i 的局部动作值函数，局部值函数只依赖于每个智能体的局部观测。

- `agent networks` ： 如图 2 （c） 表示每个智能体采用一个 DRQN 来拟合自身的 Q 值函数的到 $Q_a(\tau^a, u_t^a)$ ，DRQN 循环输入当前的观测 o_t^a 以及上一时刻的动作 u_{t-1}^a 得到 Q 值。
- `mixing network` ： 如图 2 （a） 所示， `mixing network` 是一个全连接网络，接收每个智能体的输出 $Q^n(\tau^n, u_t^n)$ 作为输入，输出联合动作值函数 $Q^{total}(\boldsymbol{\tau}, \boldsymbol{u})$ ，对每个子智能体的动作值函数做非线性映射，并且要保证公式 （2） 的单调性约束的话，只需要保证 `mixing network` 的使用的权重非负即可。
- `hypernetworks` ： `hypernetworks` 网络去产生 `mixing network` 的权重，超参数网络输入状态 s_t ，输出 `Mixing` 网络的每一层的超参数向量，采用一个线性网络以及绝对值激活函数来使得输出的权重值非负，对偏移量采用同样的方式生成，但没有非负性的约束。

- QMIX 的训练方式是依然类似标准的 DQN：

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

$$y^{\text{tot}}=r+\gamma\max_{\mathbf{u}'}Q^{total}\left(\tau',\mathbf{u}',s';\theta^{-}\right)$$

其中 b 表示从经验池中采样的 mini-batch 样本数量，θ^{-} 为目标网络 (target network) 参数。

QPLEX: DUPLEX DUELING MULTI-AGENT Q-LEARNING [10]

VDN 将 Q^{total} 分解成所有 Q^i 的累加和，QMIX 将 Q^{total} 分解成所有 Q^i 的组成的单调函数。但是不管是累加性还是单调性，它们都其实严格限制了 Q^{total} 与 Q^i 之间的关系，从而使得它们只能解决一小部分多智能体协作任务，因为复杂的实际问题中 Q^{total} 与 Q^i 的关系不一定是累加或者单调的，这样会使得 VDN 和 QMIX 近似得到的 Q^{total} 与真实的 $Q^{total,*}$ 相差很远，相应的更新方式就会存在缺陷。

由于 **VDN 和 QMIX 都只能表示 IGM 定义空间的一个子集合**，因此，QPLEX尝试去提出一种完整表达 IGM 定义的空间的算法。具体来说，QPLEX 分别对联合 Q 值 Q^{total} 和各个 agent 的 Q 值 Q^i 使用 Dueling structure：$Q=V+A$ 进行分解，将 IGM 条件（一致性）转化为更易于实现的优势函数 A 的取值范围约束，从而衍生出了具有线性分解结构的值函数分解训练算法。

Advantage-based IGM

一致性：指 joint Q 和 local Q 的贪心选择结果一致，其实就是 IGM 条件

> 作者认为一致性仅仅和 A 相关，但是和 V 无关（V 与状态有关）
>
> - 从 dueling DQN 提出的 dueling 分解结构 Q=V+A 的角度来看，这种一致性应仅约束依赖于动作的优势项 A，而不受状态值函数 V 的约束。

QPLEX 将 Dueling structure 和 IGM 结合，提出了 Advantage-based IGM，并证明了二者是等价的。

- Joint Dueling: $Q^{tot}(\boldsymbol{\tau},\boldsymbol{u})=V^{tot}(\boldsymbol{\tau})+A^{tot}(\boldsymbol{\tau},\boldsymbol{u})\ \ and\ \ V^{tot}(\boldsymbol{\tau})=\max_{\boldsymbol{u}'}Q^{tot}(\boldsymbol{\tau},\boldsymbol{u}')$
- Individual Dueling: $Q^i(\tau^i,u^i)=V^i(\tau^i)+A^i(\tau^i,u^i)\ \ and\ \ V^i(\tau^i)=\max_{u_i'}Q^{tot}(\tau^i,u^{i'})$

通过一系列推导，得出如下条件成立：

$$\forall\boldsymbol{\tau}\in\mathcal{T},\arg\max_{\boldsymbol{u}\in\mathcal{A}}A^{\text{tot}}(\boldsymbol{\tau},\boldsymbol{u})=\left(\arg\max_{u^1\in\mathcal{A}}A^1\left(\tau^1,u^1\right),\ldots,\arg\max_{u^n\in\mathcal{A}}A^n\left(\tau^n,u^n\right)\right)$$

所以 $[Q^i]_{i=1}^n$ 满足 Q^{tot} 的 Advantage-based IGM。这个分解比较直观，因为 V 只和与状态有关，与动作无关，影响 Q 的主要是 A 。与常规 IGM 相比，Advantage-based IGM 将原本作用到值函数 Q 的一致性约束，等价变换到优势函数 A 上，且可以通过限制优势函数的值范围直接实现。在此基础上，文章提出上述 Advantage-based IGM 的约束的等价形式：

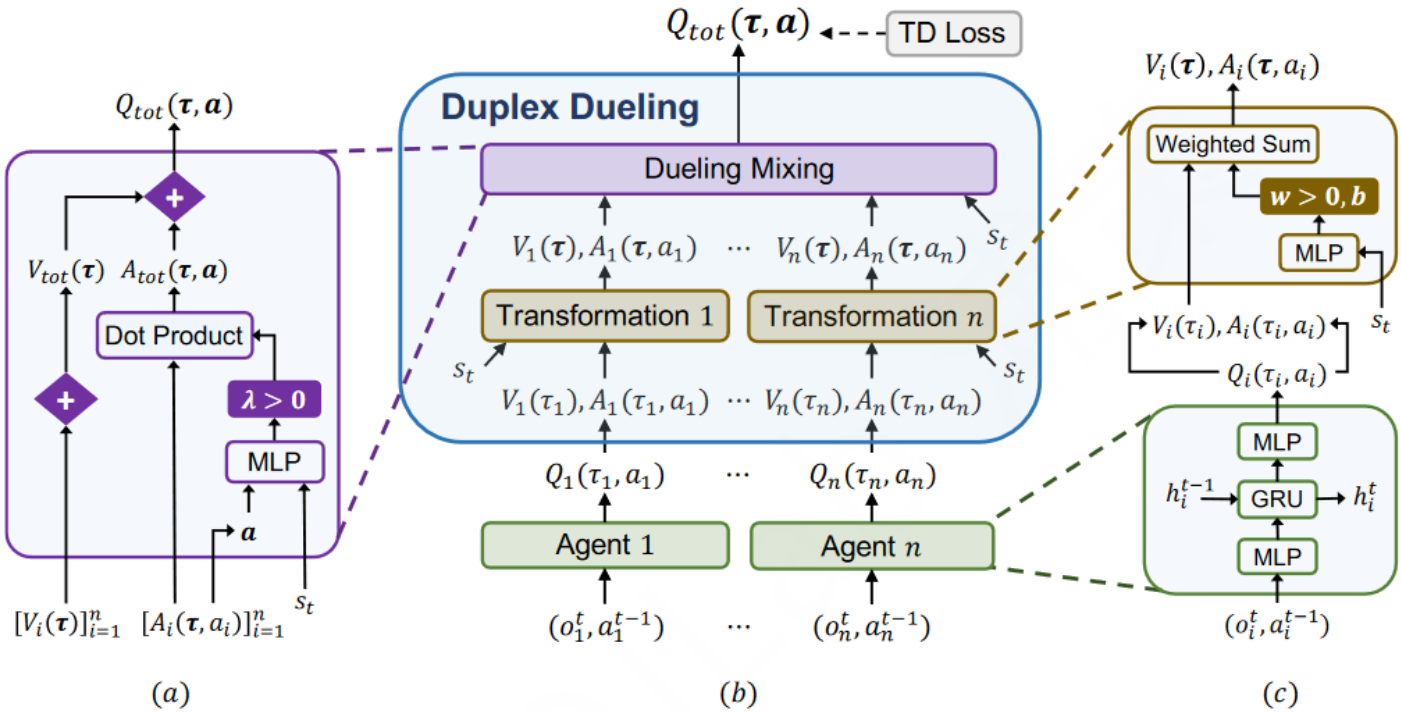
$$\forall\,\tau\in\mathcal{T},\forall\,\boldsymbol{u}^*\in\mathcal{A}^*(\tau),\forall\,\boldsymbol{u}\in\mathcal{A}(\tau)\setminus\mathcal{A}^*,\forall\,i\in\mathcal{N}:$$

$$A^{tot}(\boldsymbol{\tau},\, \boldsymbol{u}^*) \;=\; A^i(\tau^i,\, u^{i,*}) \;=\; 0 \quad and \quad A^{tot}(\boldsymbol{\tau},\, \boldsymbol{u}^*) \;<0 \quad A^i(\tau^i,\, u^{i,*}) \;\leq 0$$
$$where \quad \mathcal{A}^*(\boldsymbol{\tau}) \;=\; \{\boldsymbol{u}\,|\boldsymbol{u} \;\in\; \mathcal{A},\, Q^{tot}(\boldsymbol{\tau},\, \boldsymbol{u}) \;=\; V^{tot}(\boldsymbol{\tau})\}$$

因为 argmax 的约束，对于不是最优动作的智能体 i 来说，其 A 为负。以上公式就完成了从 IGM 到 advantage A 约束的转化。只要满足了上式，QPLEX 就能做到对 IGM 有完全的表达能力。

算法框架

QPLEX 的总体架构如图 4 所示，主要由三部分组成：



（图 4：(a) dueling mixing network 结构。 (b) QPLEX 的整体架构。(c)（底部）Agnet 网络结构和（顶部） 转换网络结构。注意：图中的符号下标为智能体索引，上标为时间，图中的 $\boldsymbol{a}$ 即为文中的 $\boldsymbol{u}$，tot 是 total 的简写。）

- **Agent 网络**

网络结构和输入与 QMIX 一致。

- **Transformation 网络**

将局部的值函数 $V^i(\tau^i), A^i(\tau^i, u^i)$ 与全局信息 s_t 结合。

具体实现方式：通过把 s_t 经过 hyppernetwork 转换称为 $w^i(\boldsymbol{\tau^i})$ 和 $b^i(\boldsymbol{\tau^i})$，然后再把 $V^i(\tau^i)$ 和 $A^i(\tau^i, u^i)$ 经过线性转换：

$$V^i(\boldsymbol{\tau}, u^i) = \omega^i(\boldsymbol{\tau})V^i(\tau^i) + b^i(\boldsymbol{\tau}) \quad and \quad A^i(\boldsymbol{\tau}, u^i) = \omega^i(\boldsymbol{\tau})A^i(\tau^i, u^i) + b^i(\boldsymbol{\tau})$$

获得基于全局观测信息的局部值函数 $V^i(\boldsymbol{\tau}),\, A^i(\boldsymbol{\tau}, a^i)$。作用主要是削弱 Partially observable 的影响，令 $\omega_i > 0$，从而保证局部函数和全局函数之间的单调性 。从而有：

$$Q_i(\boldsymbol{\tau}, u_i) = \omega_i(\boldsymbol{\tau})Q_i(\tau_i, u_i) + b_i(\boldsymbol{\tau})$$

- **Duplex Dueling 网络**

类似于 QMIX 中的 Mixing 网络，通过 $V^i(\boldsymbol{\tau}), A^i(\boldsymbol{\tau}, u^i)$ 获得联合函数 $V^{tot}(\boldsymbol{\tau}), A^{tot}(\boldsymbol{\tau}, u^i)$：

$$V^{tot}(\boldsymbol{\tau}) = \sum_{i=1}^{n} V^i(\boldsymbol{\tau})$$

$$A^{tot}(\boldsymbol{\tau}, \boldsymbol{u}) = \sum_{i=1}^{n} \lambda^i(\boldsymbol{\tau}, \boldsymbol{u}) A^i(\boldsymbol{\tau}, u^i), \lambda^i(\boldsymbol{\tau}, \boldsymbol{u}) > 0$$

其中 $\lambda^i(\boldsymbol{\tau}, \boldsymbol{u}) > 0$ 同样是为了保证 advantage-based IGM 的一致性。只有当 joint action 都是最优值时，$A^{tot}(\boldsymbol{\tau}, \boldsymbol{u^*})$ 和 $A^i(\tau^i, u^{i,*})$ 都是 0，其他所有包含非最优值的 joint action 时，$A^{tot}(\boldsymbol{\tau}, \boldsymbol{u}) < 0$，从而导致 $A^i(\tau^i, u^i) \leq 0$。λ^i 代表对不同的 individual advantage function A^i 的信度分配（credit assignment）。在 λ^i 的确定上，QPLEX 使用了 multi-head attention 机制：

$$\lambda^i(\boldsymbol{\tau}, \boldsymbol{u}) = \sum_{k=1}^{K} \lambda^{i,k}(\boldsymbol{\tau}, \boldsymbol{u}) \phi^{i,k}(\boldsymbol{\tau}) v^k(\boldsymbol{\tau})$$

K 是 head 的数量，$\lambda^{i,k}(\boldsymbol{\tau}, \boldsymbol{u}), \phi^{i,k}(\boldsymbol{\tau})$ 为 attention weights，$v^k(\boldsymbol{\tau}) > 0$ 为每个 head 的 key。最后，Q^{tot} 可以表示为如下形式：

$$Q^{tot}(\boldsymbol{\tau},\ \boldsymbol{u}) \ = \ V^{tot}(\boldsymbol{\tau}) \ + \ A^{tot}(\boldsymbol{\tau},\ \boldsymbol{u}) \ = \ \sum_{i=1}^{n} \ Q^i(\boldsymbol{\tau}, u^i) \ + \ \sum_{i=1}^{n} \ (\lambda^i(\boldsymbol{\tau},\ \boldsymbol{u}) \ - 1 \ ) A^i(\boldsymbol{\tau}, u^i)$$

- **损失函数**

在中心化训练期间，以端到端的方式学习整个网络，依然使用值分解方法常用的损失函数：

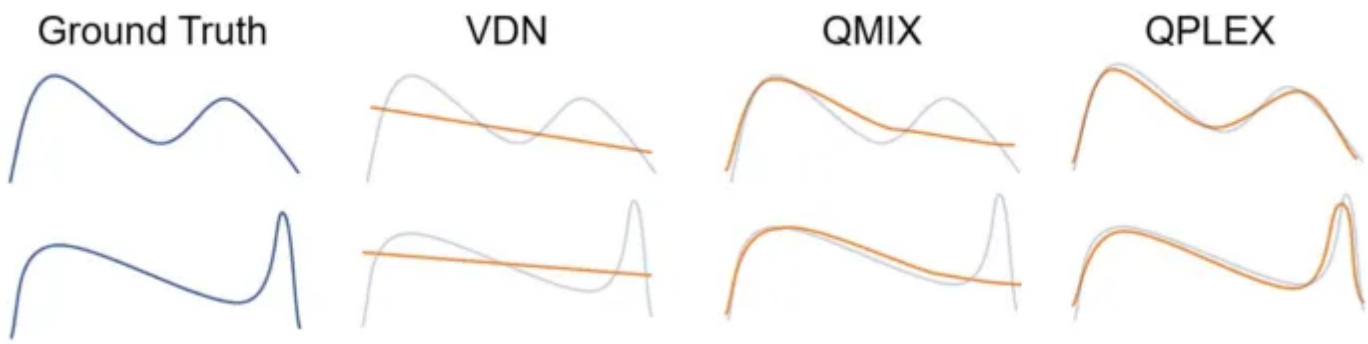
$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[\left(y^{tot} - Q^{tot}(\boldsymbol{\tau}, \boldsymbol{u}, s; \theta) \right)^2 \right]$$

$$y^{\text{tot}} = r + \gamma \max_{\boldsymbol{u}'} Q^{tot}\left(\boldsymbol{\tau}', \boldsymbol{u}'; \theta^{-}\right)$$

在去中心化执行期间，dueling mixing network 将被移除，每个智能体将根据本地动作观察历史，根据其各自的 Q 函数选择动作。

总结

上述值分解算法之所以通过累加和或者神经网络去近似 Q^{total}，并不是因为 Q^{total} 难以计算，而是因为即使得到了准确的 Q^{total}，但由于去中心化执行时，单个智能体只能得到部分可观察的限制，Q^{total} 无法被使用。因此值分解方法另辟蹊径，通过 Q^i 去近似 Q^{total}，然后在更新 Q^{total} 时利用神经网络的反向传递来更新 Q^i。总的来说，值分解的目的就是通过一个合理的，准确的方式将全局状态信息引入到每个智能体的更新梯度中。如果分解的形式越精妙，约束越少，能覆盖的多智能体协作类型越多，那么值分解类算法的最终性能就越好。更直观地，上述算法的表达能力可大致表示为图 5。



（图 5 [11]：图中纵轴为 Q^i，横轴为 Q^{total} （1） 预设的 Q^i 和 Q^{total} 之间的真实关系。（2） VDN 用 Q^i 拟合的 $Q^{total,'}$：由于 VDN 只是加和操作，所以它等价于一个平面。 （3） QMIX 用 Q^i 拟合的 $Q^{total,'}$：QMIX 要求保证单调性，所以 QMIX 是一个单峰的函数 （4） QPLEX 用 Q^i 拟合的 $Q^{total,'}$：QPLEX 和 IGM 的表达空间保持一致，表达多智能体协作的能力最丰富。）

参考文献

[1] 基于通信的多智能体强化学习方法（一）——简介及研究现状，https://zhuanlan.zhihu.com/p/421098367, 2021

[2] 强化学习笔记-MARL 之 Centralized vs Decentralized，https://zhuanlan.zhihu.com/p/331722990, 2020

[3] 多智能体强化学习中的值函数分解——VDN、QMIX、QTRAN，https://zhuanlan.zhihu.com/p/203164554, 2022

[4] 基于值分解的多智能体强化算法回顾， https://zhuanlan.zhihu.com/p/471039860, 2022

[5] Tampuu A, Matiisen T, Kodelja D, et al. Multiagent cooperation and competition with deep reinforcement learning[J]. PloS one, 2017, 12(4). https://arxiv.org/abs/1511.08779

[6] QMIX 算法分析， https://zhuanlan.zhihu.com/p/55003734, 2019

[7] Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning[J]. arXiv preprint arXiv:1706.05296, 2017. http://arxiv.org/abs/1706.05296

[8] IQL、VDN、QMIX、QTRAN 算法详解， https://zhuanlan.zhihu.com/p/386551586, 2021

[9] Rashid T, Samvelyan M, De Witt C S, et al. Monotonic value function factorisation for deep multi-agent reinforcement learning[J]. The Journal of Machine Learning Research, 2020, 21(1): 7234-7284. http://arxiv.org/abs/1803.11485

[10] Wang J, Ren Z, Liu T, et al. Qplex: Duplex dueling multi-agent q-learning[J]. arXiv preprint arXiv:2008.01062, 2020. http://arxiv.org/abs/2008.01062

[11] VDN、QMIX 和 QPLEX， https://zhuanlan.zhihu.com/p/380533160， 2021

[12] Lowe R, Wu Y I, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments[J]. Advances in neural information processing systems, 2017, 30. https://arxiv.org/abs/1706.02275