Q−MIX

一、算法简述

QMIX 是一个多智能体强化学习算法，具有如下特点： 1. 学习得到分布式策略。 2. 本质是一个值函数逼近算法。 3. 由于对一个联合动作-状态只有一个总奖励值，而不是每个智能体得到一个自己的奖励值，因此只能用于合作环境，而不能用于竞争对抗环境。 4. QMIX 算法采用集中式学习，分布式执行应用的框架。通过集中式的信息学习，得到每个智能体的分布式策略。 5. 训练时借用全局状态信息来提高算法效果。是后文提到的 VDN 方法的改进。 6. 接上一条，QMIX 设计一个神经网络来整合每个智能体的局部值函数而得到联合动作值函数，VDN 是直接求和。 7. 每个智能体的局部值函数只需要自己的局部观测，因此整个系统在执行时是一个分布式的，通过局部值函数，选出累积期望奖励最大的动作执行。 8. 算法使联合动作值函数与每个局部值函数的单调性相同，因此对局部值函数取最大动作也就是使联合动作值函数最大。 9. 算法针对的模型是一个分布式多智能体部分可观马尔可夫决策过程（Dec-POMDP）。

二、预备知识

1. 多智能体强化学习核心问题

在多智能体强化学习中一个关键的问题就是如何学习联合动作值函数，因为该函数的参数会随着智能体数量的增多而成指数增长，如果动作值函数的输入空间过大，则很难拟合出一个合适函数来表示真实的联合动作值函数。另一个问题就是学得了联合动作值函数后，如何通

过联合值函数提取出一个优秀的分布式的策略。这其实是单智能体强化学习拓展到 MARL 的**核心问题**。

2. Dec-POMDP

Dec-POMDP 是将 POMDP 拓展到多智能体系统。每个智能体的局部观测信息 ，动作 ，系统状态为 。其主要新定义了几个概念，简要介绍几个主要的。每个智能体的动作-观测历史可表示为 ，表示从初始状态开始，该智能体的时序动作-观测记录，联合动作-观测历史 表示从初始状态开始，所有智能体的时序动作-观测记录。则每个智能体的分布式策略为 ，其值函数为 都是跟动作-观测历史 有关，而不是跟状态有关了。

3. IQL

IQL（independent Q-learning）就是非常暴力的给每个智能体执行一个 Q-learning 算法，因为共享环境，并且环境随着每个智能体策略、状态发生改变，对每个智能体来说，环境是动态不稳定的，因此这个算法也无法收敛，但是在部分应用中也具有较好的效果。

4. VDN

VDN（value decomposition networks）也是采用对每个智能体的值函数进行整合，得到一个联合动作值函数。令 表示联合动作-观测历史，其中 为动作-观测历史， 表示联合动作。 为联合动作值函数， 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。VDN 采用的方法就是直接相加求和的方式

虽然 $Q_i(\tau_i, a_i)$ 不是用来估计累积期望回报的，但是这里依然叫它为值函数。分布式的策略可以通过对每个 $Q_i(\tau_i, a_i)$ 取 max 得到。

5. DRQN

DRQN 是一个用来处理 POMDP（部分可观马尔可夫决策过程）的一个算法，其采用 LSTM 替换 DQN 卷基层后的一个全连接层，来达到能够记忆历史状态的作用，因此可以在部分可观的情况下提高算法性能。具体讲解可以看强化学习——DRQN 分析详解。由于 QMIX 解决的是多智能体的 POMDP 问题，因此每个智能体采用的是 DRQN 算法。

三、QMIX

上文“多智能体强化学习核心问题”提到的就是 QMIX 解决的最核心问题。其是在 VDN 上的一种拓展，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，但是其没有在学习时利用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 就是采用一个混合网络对单智能体局部值函数进行合并，并在训练学习过程中加入全局状态信息辅助，来提高算法性能。

为了能够沿用 VDN 的优势，利用集中式的学习，得到分布式的策略。主要是因为对联合动作值函数取 $\arg\max$ 等价于对每个局部动作值函数取 $\arg\max$，其单调性相同，如下所示

$$\forall \tau \in \arg\max_{\boldsymbol{a}} Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}) \Leftrightarrow \forall i \in [1, N] \in \arg\max_{a_i} Q_i(\tau_i, a_i)$$

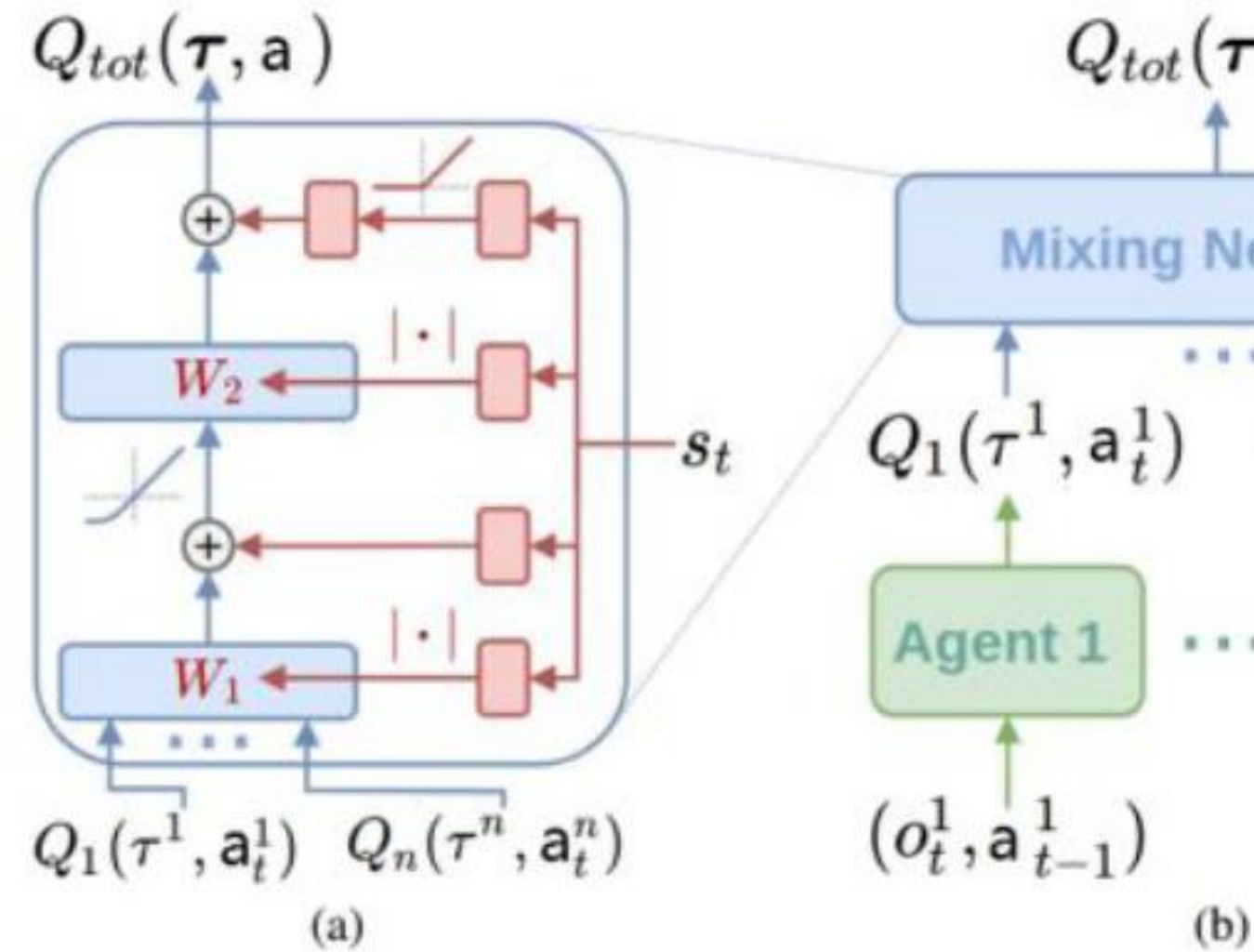
因此分布式策略就是贪心的通过局部 a 获取最优动作。QMIX 将(1)转化为一种单调性约束，

如下所示

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

若满足以上单调性，则(1)成立，为了实现上述约束，QMIX 采用混合网络（mixing network）

来实现，其具体结构如下所示



图(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 $Q_i(\tau_i, a_i; \theta_i)$，DRQN 循环输入当前的观测 o_i 以及上一时刻的动作 a_{i-1} 来得到 Q 值。

图(b)表示混合网络的结构。其输入为每个 DRQN 网络的输出。为了满足上述的单调性约束，混合网络的所有权值都是非负数，对偏移量不做限制，这样就可以确保满足单调性约束。

为了能够更多的利用到系统的状态信息 s，采用一种超网络（hypernetwork），将状态 s 作为输入，输出为混合网络的权值及偏移量。为了保证权值的非负性，采用一个线性网络以及绝对值激活函数保证输出不为负数。对偏移量采用同样方式但没有非负性的约束，混合网络最后一层的偏移量通过两层网络以及 ReLU 激活函数得到非线性映射网络。由于状态信息 s 是通过超网络混合到 Q_{tot} 中的，而不是仅仅作为混合网络的输入项，这样带来的一个好处是，如果作为输入项则 s 的系数均为正，这样则无法充分利用状态信息来提高系统性能，相当于舍弃了一半的信息量。

QMIX 最终的代价函数为

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

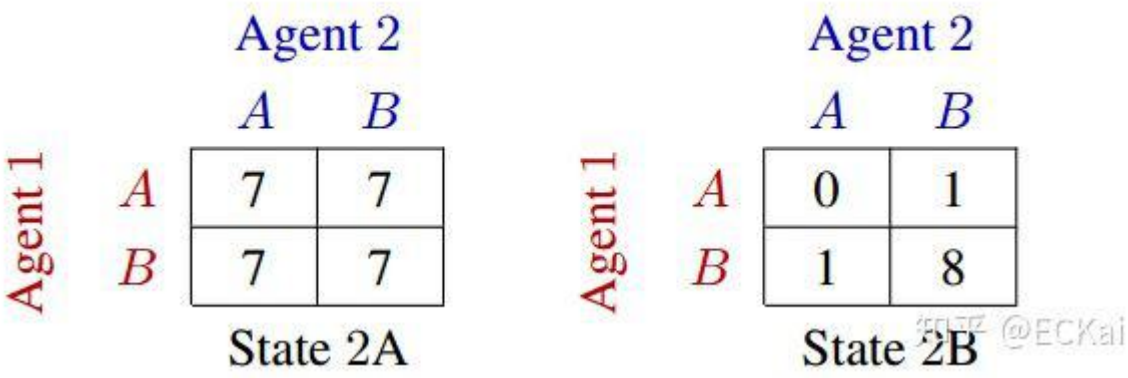
更新用到了传统的 DQN 的思想，其中 b 表示从经验记忆中采样的样本数量，$y_i^{tot}=r+\gamma\max_{a'}Q_{tot}(\tau',a',s';\theta^-)$，$\tau',a',s'$ 表示目标网络。

由于满足上文的单调性约束，对 Q_{tot} 进行 $argmax$ 操作的计算量就不在是随智能体数量呈指数增长了，而是随智能体数量线性增长，极大的提高了算法效率。

四、小示例

原文中给了一个小示例来说明 QMIX 与 VND 的效果差异，虽然 QMIX 也不能完全拟合出真实的联合动作值函数，但是相较于 VDN 已经有了很大的提高。

如下图为一个两步合作矩阵博弈的价值矩阵



在第一阶段，只有智能体 1 的动作能决定第二阶段的状态。在第一阶段，如果智能体 1 采用动作 A，则跳转到上图 2A 状态，如果智能体 1 采用动作 B，则跳转到上图 2B 状态，第二阶段的每个状态的价值矩阵如上两图所示。

现在分别用 VDN 与 QMIX 学习上述矩阵博弈各个状态的值函数矩阵，得到结果如下图所示

(a)

	State 1		State 2A		State 2B	
	A	*B*	*A*	*B*	*A*	*B*
A	6.94	6.94	6.99	7.02	-1.87	2.31
B	6.35	6.36	6.99	7.02	2.33	6.51

(b)

	A	*B*	*A*	*B*	*A*	*B*
A	6.93	6.93	7.00	7.00	0.00	1.00
B	7.92	7.92	7.00	7.00	1.00	8.00

Table 2. Q_{tot} on the two-step game for (a) VDN and (b) QMIX.

(a)为 VDN 拟合结果，(b)为 QMIX 拟合结果。可以从上图，VDN 的结果是智能体 在第一阶段采用动作 ，显然这不是最佳状态，而 QMIX 是智能体 在第一阶段采用动作 ，得到了最大的累积期望奖励。由上可得 QMIX 的逼近能力比 VDN 更强，QMIX 算法的效果更好。

https://zhuanlan.zhihu.com/p/55003734