# 一、前言

Cooperative Multi-Agent Reinforcement Learning 最近是一个比较热的研究点，涌现出了许多新奇的算法，主流算法主要分为$Communication$和 Centralized Training Decentralized Execution($\boldsymbol{CTDE}$)两种。本文将从基本的 MARL 问题出发，尽可能简明扼要地对 CTDE 中的各种 Value Decomposition 方法（VDN、QMIX、QTRAN、QPD）进行介绍。

# 二、MARL 中的难点

## 1. 部分可观察

当 agent 和环境进行交互时，agent 无法看到和环境的全局状态$\boldsymbol{s}$，只能观察到自己视野范围内的局部信息$o$。

图 1：部分可观察

## 2. 不稳定性

在 multi-agent 环境中，由于 agents 之间相互影响，因此 agent $i$ 在观察$o_i$下执行动作$u_i$后得的$r_i$与$o_i'$是由所有 agents 的行为造成的，即$r_i = R_i(\boldsymbol{s}, \boldsymbol{u}), o_i' = T_i(\boldsymbol{s}, \boldsymbol{u})$。那么此时对于 agent $i$ 而言，即使他在观察$o_i$下一直都执行动作$u_i$，但是由于$\boldsymbol{s}$未知且其他 agent 的策略在不断变化，此时 agent $i$ 得到的$r_i$与$o_i'$可能是

不同的（比如之前的$(o_i,u_i)$得到的奖励为 1，但是下一次由于队友的动作发生了变化，奖励又变成了-1）。这就是 MARL 中的不稳定性，即 reward 与 transition 存在不稳定性，从而导致 agent $i$ 的值函数$Q_i(o_i,u_i)$的更新就会十分不稳定。

到这里我们可以总结一下，造成不稳定性的原因主要有两点：

- 部分可观察的场景使得$o_i$下对应的$\boldsymbol{s}$有很多
- 其他 agent 也在学习，策略不断在变化，选择的动作也在不断变化

因此，同样的$(o_i,u_i)$可能对应着许多不同的$(\boldsymbol{s},\boldsymbol{u})$，从而导致其获得的反馈不稳定。此时可能有人会问，既然部分可观察造成的也是不稳定性，为什么还要单独拎出来说呢？这一点我们在下一节会提到。

# 三、为什么要进行值函数分解

其实简单来说，MARL 中的难点就是智能体 $i$ 只能站在自己的角度去观察去决策，无法站在全局的角度去观察并决策，从而无法学到全局最优策略。因此为了解决这个问题，大家提出使用$Centralized\ Training\ Decentralized\ Excution(CTDE)$的方法，将条件限制放松，允许 agents 在训练的时候可以访问全局信息，从而站在全局的角度去训练。但是即使能站在全局的角度去训练，你要训练出一个什么形式的策略才行呢？

直观的答案就是去训练一个全局的$Q_{total}(\boldsymbol{s},\boldsymbol{u})$，它考虑了全局信息，可以直接克服 MARL 中的不稳定性。但是要**注意**，即使你训练出了$Q_{total}(\boldsymbol{s},\boldsymbol{u})$又能怎么样呢，部分可观察导致 agents 在执行的过程中是无法得到$\boldsymbol{s}$的，也就是说你拥有$Q_{total}(\boldsymbol{s},\boldsymbol{u})$却无法使用它。说到这里就可以回答上一节提出的问题，部分可观察除了会造成不稳定性，还会导致我们无法直接使用$Q_{total}(\boldsymbol{s},\boldsymbol{u})$，这一点是极为致

命的。

此时问题已经很明显了，仅仅使用 agent 的$Q_i(o_i,u_i)$进行决策存在不稳定性，而只有$Q_{total}(\boldsymbol{s},\boldsymbol{u})$才能站在全局的角度进行学习去解决不稳定性，但它得到了又没办法直接用，因此就出现了一系列值函数分解的方式来解决这个问题。

# 四、VDN

上面说到只有$Q_{total}(\boldsymbol{s},\boldsymbol{u})$才能站在全局的角度进行学习去，从而解决不稳定性，但是部分可观察的限制使得$Q_{total}(\boldsymbol{s},\boldsymbol{u})$无法使用。因此 VDN 开创性地提出使用$Q_i(o_i,u_i)$对$Q_{total}(\boldsymbol{s},\boldsymbol{u})$进行**分解**而不是直接去学习，具体的分解方式为

$$Q_{total}(\boldsymbol{s},\boldsymbol{u})=\sum_{i=1}^{N}Q_i(o_i,u_i) \tag{1}$$

其中$N$是环境中 agents 的数量。近似得到$Q_{total}(\boldsymbol{s},\boldsymbol{u})$之后，VDN 使用 DQN 的更新方式，通过全局奖励$r$来更新$Q_{total}(\boldsymbol{s},\boldsymbol{u})$，其 loss 函数表示为

$$L(\theta)=\frac{1}{M}\sum_{j=1}^{M}\left(y_j-Q_{total}(\boldsymbol{s},\boldsymbol{u})\right)^2 \tag{2}$$

其中$Q_i^-$是 batch size，$y_j=r_j+\gamma\underset{\boldsymbol{u}}{\operatorname{argmax}}\,Q_{total}^-(\boldsymbol{s}',\boldsymbol{u})$，$Q_{total}^-=\sum_{i=1}^{N}Q_i^-(o_i,u_i)$，$Q_i^-$是 target_net。

我们知道$Q_i$是经过神经网络得到的，它是一个 tensor，那么所有的$Q_i$加起来得到的$Q_{total}$也是一个 tensor，因此通过 TD-error 来更新$Q_{total}$，**梯度会经过$\boldsymbol{Q_{total}}$反向传递给每个$\boldsymbol{Q_i(o_i,u_i)}$从而去更新它们**，这样就可以站在全局的角度去更新$Q_i(o_i,u_i)$。

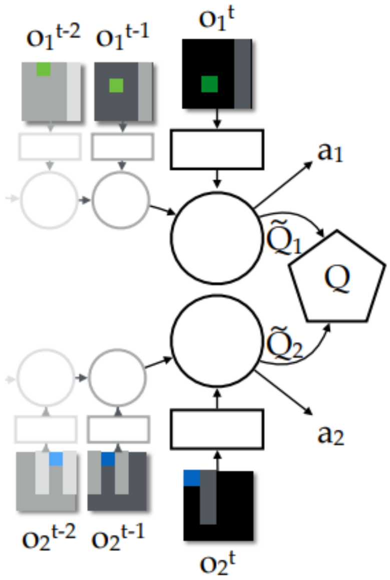


图 2：VDN 结构

# 五、QMIX

## 1. 重新认识值函数分解

我们知道 VDN 是使用$\sum_{i=1}^{N}Q_i(o_i,u_i)$来计算$Q_{total}(\boldsymbol{s},\boldsymbol{u})$，那么其实在我看来与其将这种方法叫做值函数分解，不如叫做**值函数近似**更好理解，因为它是通过$\sum_{i=1}^{N}Q_i(o_i,u_i)$来得到一个全局的$Q_{total}(\boldsymbol{s},\boldsymbol{u})$，从而让其尽可能去近似真实的$Q_{total}^{*}(\boldsymbol{s},\boldsymbol{u})$，这个真实的$Q_{total}^{*}$我们是不知道的，即使知道了我们也会因为部分可观察的限制无法使用。因此只要$Q_{total}$越接近$Q_{total}^{*}$，那么$Q_i$的更新就越顺滑。所以如何让$Q_{total}$尽可能逼近$Q_{total}^{*}$呢？这就需要通过调整$Q_{total}$的计算方式，通过$Q_i$去得到一个更加接近$Q_{total}^{*}$的$Q_{total}$，QMIX 就是基于这个想法而产生的。

**注意下文我们将一律使用值函数近似的称呼。**

## 2. VDN 的缺点

我们可以发现 VDN 是将所有的$Q_i(o_i,u_i)$加起来去近似$Q_{total}(\boldsymbol{s},\boldsymbol{u})$，即认为$Q_{total}$和$Q_i$之间的关系是求和，通过**累加和**来近似$Q_{total}$。但是我们知道求和是一

种很简单的关系，它不能表示复杂的$Q_{total}$，如果实际上$Q_{total}$和$Q_i$的关系很复杂，那么 VDN 就没用了。此外，既然我们已经是集中式训练了，那么为什么不尽可能地利用集中式训练这个优势呢？而 VDN 忽略了学习期间可用的任何额外状态信息。

## 3. QMIX 的思想

由于 VDN 不能表示复杂的$Q_{total}(\boldsymbol{s},\boldsymbol{u})$，因此 QMIX 首先提出使用神经网络$f$去近似$Q_{total}(\boldsymbol{s},\boldsymbol{u})$，因为神经网络是具有强大的表示能力的。此外，由于 VDN 没有尽可能利用集中式训练的优势，忽略了学习期间可用的任何额外状态信息，因此 QMIX 在近似$Q_{total}(\boldsymbol{s},\boldsymbol{u})$时额外使用了全局状态$\boldsymbol{s}$，这样就可以基于全局状态$\boldsymbol{s}$进行训练，而不是像 VDN 那样仅仅拿$Q_1,\dots,Q_N$去训练。QMIX 的 Loss 函数和 VDN 一样，还是使用 DQN 那一套的 TD-error 来训练。

这样做还存在一个问题，虽然我们希望通过神经网络$f$去学习$Q_{total}$与$[Q_1,\dots,Q_N]$之间的关系表示，但是这并不代表$f$可以随便学。要注意我们的目的是学习到好的$Q_i(o_i,u_i)$，$f$只是为了更好地近似$Q_{total}$。如果$f$是一个很差的关系表示，那么近似出的$Q_{total}$已经不对了，更别说去更新$Q_i$。因此 QMIX 限制$f$中的参全部非负，从而确保满足条件

$$\frac{\partial Q_{total}}{\partial Q_i} \geq 0, \forall i \tag{3}$$

该条件可以让$Q_{total}$与$Q_i$之间的关系满足单调性，从而确保

在单调性保证时，要使得Qtotal最大，只需每个Qi最大。

$$\underset{\boldsymbol{u}}{\operatorname{argmax}}\, Q_{total}(\boldsymbol{s},\boldsymbol{u}) = \left[\underset{u_1}{\operatorname{argmax}}\, Q_1(o_1,u_1), \dots\dots, \underset{u_N}{\operatorname{argmax}}\, Q_1(o_N,u_N)\right] \tag{4}$$

上式含义是只要确保$Q_{total}(\boldsymbol{s},\boldsymbol{u})$上的$argmax$得到的联合动作$\boldsymbol{u}$与每个$Q_i$上$argmax$得到的$[u_1,u_2,\dots,u_N]$相同即可，那么每个 agent 选择的**局部最优动作**$\underset{u_i}{\operatorname{argmax}}\, Q_i(o_i,u_i)$恰好就是**全局最优动作**$\underset{\boldsymbol{u}}{\operatorname{argmax}}\, Q_{total}(\boldsymbol{s},\boldsymbol{u})$的一部分。

我们可以这么理解：对于神经网络$f$，你可以自己去学习，然后学习如何通过$[Q_1,\dots,Q_N]$近似得到$Q_{total}$，但是你不能乱近似，需要守住$\frac{\partial Q_{total}}{\partial Q_i} \geq 0$这个底线，只有守住了底线才能保证**局部最优动作**就是**全局最优动作**。

### 4. 使用 hypernetworks 去利用全局状态$s$

前面还说到 QMIX 在近似$Q_{total}(\boldsymbol{s},\boldsymbol{u})$时额外使用了全局状态$\boldsymbol{s}$，这样就可以基于全局状态$\boldsymbol{s}$进行训练。但是如果直接将$\boldsymbol{s}$和$[Q_1,\dots,Q_N]$一起输入到神经网络$f$去得到$Q_{total}$，由于我们前面限制了$f$中的参数是非负的，那么$Q_{total}$和$\boldsymbol{s}$的关系也随之变成了$\frac{\partial Q_{total}}{\partial s} \geq 0$，但这会对$Q_{total}$和$\boldsymbol{s}$的关系进行不必要的限制，因为我们只希望**局部最优动作**就是**全局最优动作**，$Q_{total}$和$\boldsymbol{s}$的关系是无所谓什么形式的。

因此 QMIX 使用了 hypernetworks 方法，即专门用一个神经网络$g(\boldsymbol{s})$，输入$\boldsymbol{s}$后得到$f$的参数$w,b$，然后将$[Q_1,\dots,Q_N]$输入到$f$中输出$Q_{total}$，为了保证单调性令$w \geq 0$。此外，使用$g$去生成$f$的参数能够让$g$聚焦$\boldsymbol{s}$，从而根据$\boldsymbol{s}$更好地生成当前状态下$Q_{total}$与$Q_i$之间的关系，不同的$\boldsymbol{s}$下$Q_{total}$与$Q_i$之间的关系可能是不同的。

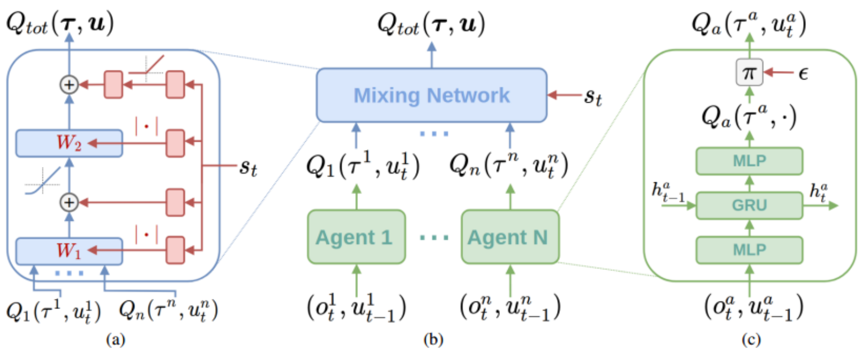


图 3：QMIX 结构

# 六、QTRAN

## 1. VDN 和 QMIX 的缺点

我们知道 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任务，因为可能很多任务中$Q_{total}$与$Q_i$的关系不一定是累加或者单调，从而使得 VDN 和 QMIX 近似得到的$Q_{total}$与真实的$Q_{total}^*$相差很远，那么这样的更新是无用的。

## 2. QTRAN 的提出

QTRAN 聚焦于释放累加性和单调性的限制，去分解所有可分解的任务。其的思想在于只要保证个体最优动作$\bar{u}$和联合最优动作$u^*$是相同的，即

$$\underset{\boldsymbol{u}}{\operatorname{argmax}} \, Q_{total}(\boldsymbol{s}, \boldsymbol{u}) = \left[\underset{u_i}{\operatorname{argmax}} \, Q_i(o_i, u_i)\right]_{i=1}^{N} \tag{5}$$

论文定义只要满足公式(5)，那么就称$[Q_i]_{i=1}^N$对$Q_{total}$满足 IGM(Individual-Global-Max)。因此论文认为只要$[Q_i]_{i=1}^N$对$Q_{total}$满足 IGM，那么$[Q_i]_{i=1}^N$与$Q_{total}$的具体关系我们是不需要考虑的。

接下来开始讲 QTRAN 的具体做法，由于 QTRAN 中的公式与证明比较难懂，因此本文尽可能通俗地对其进行介绍。 此外，由于本文的核心部分在于定理的提出，因此对于 QTRAN-alt 的提出本文就不多做介绍了，因为其实和 QTRAN-base 大同小异。

## 3. 做法

### 1) 直接学习一个全局的$Q_{total}(\boldsymbol{s}, \boldsymbol{u})$

QTRAN 认为既然 VDN 和 QMIX 是通过累加或者单调近似得到的$Q_{total}$，那么$Q_{total}$就很有可能与真实的$Q_{total}^*$相差很远，那我不如直接去学习一个真实的$Q_{total}^*$。因此 QTRAN 先**直接去学习（注意这里不是近似了）**一个全局的$Q_{total}(\boldsymbol{s}, \boldsymbol{u})$，这样首先可以保证我的$Q_{total}$和真实的$Q_{total}^*$非常接近。

其实广义上我一般不区分学习和近似，但这里“近似”指的是用某个可能并不真实的形式对某个值进行分解

但是问题来了，前面我们说到，即使得到了$Q_{total}(\boldsymbol{s}, \boldsymbol{u})$由于部分可观察的限

制，我们是无法使用$Q_{total}(\boldsymbol{s},\boldsymbol{u})$进行决策的，现在该怎么办呢？QTRAN 认为即使我无法使用$Q_{total}(\boldsymbol{s},\boldsymbol{u})$进行决策，但是我可以使用$Q_{total}(\boldsymbol{s},\boldsymbol{u})$来更新$Q_i(o_i,u_i)$，只要我能建立起$Q_{total}$与$Q_i$之间的关系，成功使用真实的$Q_{total}(\boldsymbol{s},\boldsymbol{u})$来更新$Q_i(o_i,u_i)$，那么决策的时候使用$Q_i(o_i,u_i)$就可以了。**因此接下来的问题就变为如何建立$\boldsymbol{Q_{total}}$与$\boldsymbol{Q_i}$之间的关系。**

## 2) 建立起$Q_{total}(\boldsymbol{s},\boldsymbol{u})$与$Q_i(o_i,u_i)$之间的关系

QTRAN 首先通过累加和$\sum_{i=1}^{N}Q_i(o_i,u_i)$来近似$Q_{total}(\boldsymbol{s},\boldsymbol{u})$，但是由于累加和得到的$Q'_{total}(\boldsymbol{s},\boldsymbol{u})$可能和我们学到的接近真实的$Q_{total}(\boldsymbol{s},\boldsymbol{u})$差很多，因此又引入了一个$V(\boldsymbol{s})$来弥补$Q'_{total}$与$Q_{total}$之间的差距，即使用$\sum_{i=1}^{N}Q_i(o_i,u_i)+V(\boldsymbol{s})$来近似$Q_{total}$。但是$V(\boldsymbol{s})$如何得到的？我们又如何通过$Q_{total}$去更新$Q_i$呢？

QTRAN 提出了一个让$[Q_i]_{i=1}^{N}$对$Q_{total}$满足 IGM 的充分条件，首先给出该条件：

---

**定理 1：**对于一个$\boldsymbol{s}$与$Q_{total}(\boldsymbol{s},\boldsymbol{u})$，令$\bar{u}_i$表示**局部最优动作**$\arg\max_{u_i} Q_i(o_i,u_i)$，$\bar{\boldsymbol{u}}=[\bar{u}_i]_{i=1}^{N}$，$\boldsymbol{Q}=[Q_i]_{i=1}^{N}$，$\boldsymbol{u}=[u_i]_{i=1}^{N}$表示**实际选择的动作**。如果满足

$$\sum_{i=1}^{N}Q_i(o_i,u_i)-Q_{total}(\boldsymbol{s},\boldsymbol{u})+V(\boldsymbol{s})=\begin{cases}0 & \boldsymbol{u}=\bar{\boldsymbol{u}} \quad (4a)\\ \geq 0 & \boldsymbol{u}\neq\bar{\boldsymbol{u}} \quad (4b)\end{cases} \tag{5}$$

其中

$$V(\boldsymbol{s})=\max_{\boldsymbol{u}}Q_{total}(\boldsymbol{s},\boldsymbol{u})-\sum_{i=1}^{N}Q_i(o_i,\bar{u}_i)$$

则$[Q_i]_{i=1}^{N}$对$Q_{total}(\boldsymbol{s},\boldsymbol{u})$满足 IGM。

---

**这个条件是全文最难懂的地方，但也是最核心的内容。**论文的证明很繁琐，

因此我们换一个角度理解，站在作者的角度去思考作者为什么会提出这个条件：

a) 首先，我们的目标是希望让$[Q_i]_{i=1}^{N}$与$Q_{total}$满足

$$\underset{\boldsymbol{u}}{\operatorname{argmax}} Q_{total}(\boldsymbol{s},\boldsymbol{u}) = \left[\underset{u_i}{\operatorname{argmax}} Q_i(o_i,u_i)\right]_{i=1}^{N} \tag{6}$$

b) 由于$\bar{u}_i = \arg\max_{u_i} Q_i(o_i,u_i)$是局部最优动作，那么我们只需要让

$$Q_{total}(\boldsymbol{s},\overline{\boldsymbol{u}}) \geq Q_{total}(\boldsymbol{s},\boldsymbol{u}), \boldsymbol{u} \neq \overline{\boldsymbol{u}} \tag{7}$$

即可，也就是让局部最优动作组成的$\overline{\boldsymbol{u}}$比其他所有$\boldsymbol{u}$对应的$Q_{total}$都大，那么此时$\overline{\boldsymbol{u}}$就是全局最优，自然就满足了上述条件。因此现在的问题就转变为红色部分。

c) 作者想通过$V(\boldsymbol{s})$去弥补$Q'_{total}(\boldsymbol{s},\boldsymbol{u}) = \sum_{i=1}^{N} Q_i(o_i,u_i)$与学习到的真实的$Q_{total}(\boldsymbol{s},\boldsymbol{u})$之间的差距，但是由于$V(\boldsymbol{s})$是一个标量，只能弥补$\boldsymbol{s}$下一个动作$\boldsymbol{u}$上$Q'_{total}(\boldsymbol{s},\boldsymbol{u})$与$Q_{total}(\boldsymbol{s},\boldsymbol{u})$的差距，该去弥补谁呢？因此 QTRAN 让$V(\boldsymbol{s})$去弥补$\boldsymbol{u} = \overline{\boldsymbol{u}}$时$Q'_{total}$与$Q_{total}$的差距，即

$$\sum_{i=1}^{N} Q_i(o_i,\bar{u}_i) + V(\boldsymbol{s}) = Q_{total}(\boldsymbol{s},\overline{\boldsymbol{u}}), \boldsymbol{u} = \overline{\boldsymbol{u}} \tag{$\boldsymbol{a}$}$$

也就是说，当$\boldsymbol{u} = \overline{\boldsymbol{u}}$时$\sum_{i=1}^{N} Q_i(o_i,\bar{u}_i) + V(\boldsymbol{s})$恰好能够正确近似出真实的$Q_{total}(\boldsymbol{s},\overline{\boldsymbol{u}})$，这就是条件$(a)$的来由。

d) 我们现在再回过头来看(7)中的条件，此时 c)通过条件$(a)$已经帮我们正确地近似了$Q_{total}(\boldsymbol{s},\overline{\boldsymbol{u}})$，那么我们可以把(7)转化为

$$\sum_{i=1}^{N} Q_i(o_i,\bar{u}_i) + V(\boldsymbol{s}) \geq Q_{total}(\boldsymbol{s},\boldsymbol{u}), \boldsymbol{u} \neq \overline{\boldsymbol{u}} \tag{8}$$

此时问题就变为如何满足条件(8)。

e) 由于$\bar{u}_i = \arg\max_{u_i} Q_i(o_i,u_i)$是局部最优动作，那么我们可以得到

$$\sum_{i=1}^{N} Q_i(o_i,\bar{u}_i) + V(\boldsymbol{s}) \geq \sum_{i=1}^{N} Q_i(o_i,u_i) + V(\boldsymbol{s}), \boldsymbol{u} \neq \overline{\boldsymbol{u}} \tag{9}$$

此时(9)是客观事实，我们需要创造条件满足(8)，那么我们只需要让

$$\sum_{i=1}^{N} Q_i(o_i, a_i) + V(\boldsymbol{s}) \geq Q_{total}(\boldsymbol{s}, \boldsymbol{u}), \boldsymbol{u} \neq \overline{\boldsymbol{u}} \qquad (\boldsymbol{b})$$

通过(**9**)与($\boldsymbol{b}$)我们就可以得到(8)，这样我们就可以按照(8) → (7) → (6)地顺序让(6)成立，从而使得$[Q_i]_{i=1}^N$对$Q_{total}(\boldsymbol{s}, \boldsymbol{u})$满足 IGM。这就是条件$(b)$的由来。

f) 此外，对于定理 1 中的

$$V(\boldsymbol{s}) = \max_{\boldsymbol{u}} Q_{total}(\boldsymbol{s}, \boldsymbol{u}) - \sum_{i=1}^{N} Q_i(o_i, \bar{u}_i)$$

它是由$(a)$、$(b)$都成立之后推导出来的，而不是我们需要创造的前提条件。只有$(a)$、$(b)$是我们需要创造的前提条件。

### 3) 使用$Q_{total}(\boldsymbol{s}, \boldsymbol{u})$去更新$Q_i(o_i, u_i)$

QTRAN 的 Loss 函数为

$$L_{td} = \left(Q_{total}(\boldsymbol{s}, \boldsymbol{u}) - y^{dqn}(r, \boldsymbol{s}')\right)^2$$

$$L_{opt} = \left(Q'_{total}(\boldsymbol{s}, \overline{\boldsymbol{u}}) - \hat{Q}_{total}(\boldsymbol{s}, \overline{\boldsymbol{u}}) + V(\boldsymbol{s})\right)^2$$

$$L_{nopt} = \left(\min\left[Q'_{total}(\boldsymbol{s}, \boldsymbol{u}) - \hat{Q}_{total}(\boldsymbol{s}, \boldsymbol{u}) + V(\boldsymbol{s}), 0\right]\right)^2$$

其中$y^{dqn}(r, \boldsymbol{s}') = r + \gamma Q_{total}(\boldsymbol{s}', \overline{\boldsymbol{u}}')$，$\overline{\boldsymbol{u}}' = \left[\arg\max_{u_i} Q_i(s_i', u_i; \boldsymbol{\theta}^-)\right]_{i=1}^N$。

对数上述的三个 Loss 函数，$L_{td}$使用标准的 TD-error 来更新我们要学习的全局$Q_{total}(\boldsymbol{s}, \boldsymbol{u})$，$L_{opt}$、$L_{nopt}$是为了通过$Q_{total}(\boldsymbol{s}, \boldsymbol{u})$来指导$Q_i(o_i, u_i)$与$V(\boldsymbol{s})$的更新，其中$L_{opt}$对应条件$(a)$，$L_{nopt}$对应条件$(b)$，$\hat{Q}_{total}$表示将$Q_{total}$固定，也就是说$L_{opt}$、$L_{nopt}$不更新$Q_{total}$，因为此时是需要通过$Q_{total}(\boldsymbol{s}, \boldsymbol{u})$来指导$Q_i(o_i, u_i)$与$V(\boldsymbol{s})$的更新。

# 七、总结

最后想对值函数分解这一部分的工作再说明一下。VDN 与 QMIX 之所以通过累加和或者神经网络去近似$Q_{total}(\boldsymbol{s},\boldsymbol{u})$，不是因为$Q_{total}(\boldsymbol{s},\boldsymbol{u})$不好得到，而是因为$Q_{total}(\boldsymbol{s},\boldsymbol{u})$即使得到了，由于 Decentralized Execution 中的部分可观察的限制，$Q_{total}(\boldsymbol{s},\boldsymbol{u})$无法被使用。因此 VDN 与 QMIX 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$Q_i$。而 QTRAN 呢则是直接学习一个$Q_{total}$，同时创造了两个条件来约束$Q_{total}$和$Q_i$之间的关系，从而通过该关系去更新$Q_i$。

其实后续还有很多优秀的值函数分解工作，比如今年 ICML20 的 QPD，也是先学习$Q_{total}$来建立$Q_{total}$和$Q_i$之间的关系；还有张崇洁老师组里正在投稿的 DOP 等等，以后如果有时间会把这些优秀的文章也补充进来。