

第十四章 合作关系设定下的多智能体强化学习

本章只考虑最简单的设定——完全合作关系——并在这种设定下研究多智能体强化学习 (MARL)。第 14.1 节定义“完全合作关系”下的策略学习。第 14.2 节介绍“完全合作关系”下的多智能体 A2C 方法, 本书称之为 MAC-A2C。第 14.3 节介绍 MARL 的三种常见架构——完全去中心化、完全中心化、中心化训练 + 去中心化决策——并在三种框架下实现 MAC-A2C。

本章与上一章对状态的定义有所区别。在多智能体系统中, 一个智能体未必能观测到全局状态 S 。设第 i 号智能体有一个局部观测, 记作 O^i , 它是 S 的一部分。不妨假设所有的局部观测的总和构成全局状态:

$$S = [O^1, O^2, \dots, O^m],$$

MARL 的文献大多采用这种假设。本章中采用的符号如图 14.1 所示。

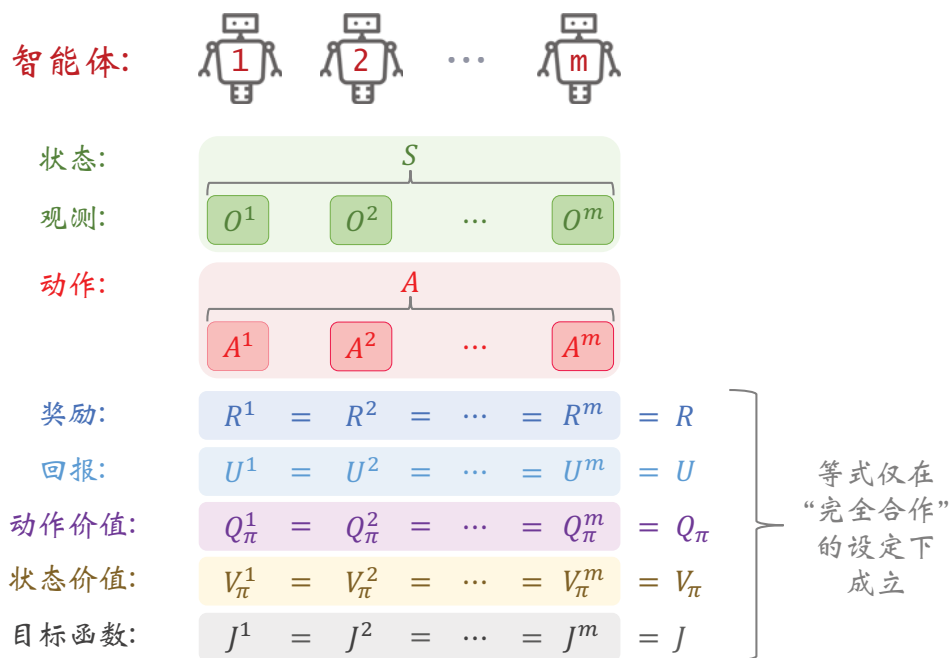


图 14.1: 多智能体强化学习 (MARL) 在“完全合作关系”设定下的符号。

14.1 合作关系设定下的策略学习

MARL 中的**完全合作关系** (Fully-Cooperative) 意思是所有智能体的利益是一致的, 它们具有**相同的奖励**:

$$R^1 = R^2 = \dots = R^m \triangleq R.$$

因此, 所有的智能体都有**相同的回报**:

$$U^1 = U^2 = \dots = U^m \triangleq U.$$

因为价值函数是回报的期望, 所以所有的智能体都有**相同的价值函数**。省略上标 i , 把动作价值函数记作 $Q_\pi(S, A)$, 把状态价值函数记作 $V_\pi(S)$ 。

注意, 价值函数 Q_π 和 V_π 依赖于所有智能体的策略:

$$\pi(A^1 | S; \theta^1), \quad \pi(A^2 | S; \theta^2), \quad \dots, \quad \pi(A^m | S; \theta^m).$$

举个例子, 在某个竞技电游中, 玩家组队打任务; 每完成一个任务, 团队成员 (即智能体) 获得相同的奖励。所以大家的 R, U, Q_π, V_π 全都是一样的。回报的期望——即价值函数 Q_π 与 V_π ——显然与所有成员的策略相关: 只要有一个猪队友 (即策略差) 拖后腿, 就有可能导致任务失败。通常来说, 团队成员有分工合作, 所以每个成员的策略是不同的, 即 $\theta^i \neq \theta^j$ 。

如果做策略学习 (即学习策略网络参数 $\theta^1, \dots, \theta^m$), 那么所有智能体都有一个共同目标函数:

$$J(\theta^1, \dots, \theta^m) = \mathbb{E}_S[V_\pi(S)].$$

所有智能体的目的是是一致的, 即改进自己的策略网络参数 θ^i , 使得目标函数 J 增大。那么策略学习可以写作这样的优化问题:

$$\max_{\theta^1, \dots, \theta^m} J(\theta^1, \dots, \theta^m). \quad (14.1)$$

注意, 只有“完全合作关系”这种设定下, 所有智能体才会有共同的目标函数, 其原因在于 $R^1 = \dots = R^m$ 。对于其它设定——“竞争关系”、“混合关系”、“利己主义”——智能体的目标函数是各不相同的 (见下一章)。

合作关系设定下的策略学习的原理很简单, 即让智能体各自做策略梯度上升, 使得目标函数 J 增长。

$$\text{第 1 号智能体执行:} \quad \theta^1 \leftarrow \theta^1 + \alpha^1 \cdot \nabla_{\theta^1} J(\theta^1, \dots, \theta^m),$$

$$\text{第 2 号智能体执行:} \quad \theta^2 \leftarrow \theta^2 + \alpha^2 \cdot \nabla_{\theta^2} J(\theta^1, \dots, \theta^m),$$

$$\vdots \quad \quad \quad \vdots$$

$$\text{第 } m \text{ 号智能体执行:} \quad \theta^m \leftarrow \theta^m + \alpha^m \cdot \nabla_{\theta^m} J(\theta^1, \dots, \theta^m).$$

公式中的 $\alpha^1, \alpha^2, \dots, \alpha^m$ 是学习率。判断策略学习收敛的标准是目标函数 $J(\theta^1, \dots, \theta^m)$ 不再增长。在实践中, 当平均回报不再增长, 即可终止算法。由于无法直接计算策略梯度 $\nabla_{\theta^i} J$, 我们需要对其做近似。下一节用价值网络近似策略梯度, 从而推导出一种实际可行的策略梯度方法。

《深度强化学习》2021-02-19 尚未校对, 仅供预览。
如发现错误, 请告知作者 shusen.wang@stevens.edu

14.2 合作设定下的多智能体 A2C

第 8 章介绍过 Advantage Actor-Critic (A2C) 方法。本节介绍“完全合作关系”设定下的多智能体 A2C 方法 (Multi-Agent Cooperative A2C), 缩写 MAC-A2C。注意, 本节介绍的方法仅适用于“完全合作关系”, 也就是要求所有智能体有相同的奖励: $R^1 = \dots = R^m$ 。第 14.2.1 小节定义策略网络和价值网络。第 14.2.2 小节描述 MAC-A2C 训练和决策。第 14.2.3 小节讨论 MAC-A2C 实现中的难点。

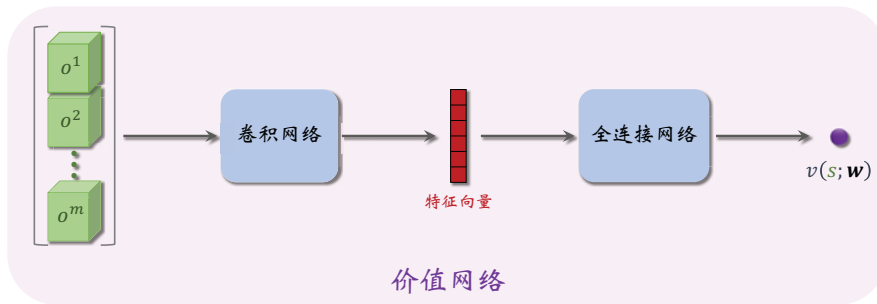


图 14.2: 图中是 MAC-A2C 中的价值网络 $v(s; \mathbf{w})$ 。所有智能体共用这个价值网络。输入是所有智能体的观测: $s = [o^1, \dots, o^m]$ 。输出是价值网络给 s 的评分。

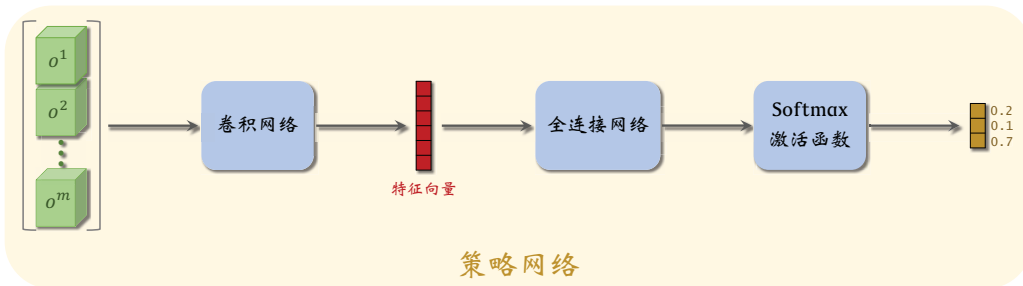


图 14.3: 图中是 MAC-A2C 中第 i 号智能体的策略网络 $\pi(\cdot|s; \theta^i)$ 。所有智能体的策略网络结构都一样, 但是参数 $\theta^1, \dots, \theta^m$ 可能不一样。输入是所有智能体的观测: $s = [o^1, \dots, o^m]$ 。输出是在离散动作空间 \mathcal{A}^i 上的概率分布。

14.2.1 策略网络和价值网络

本章只考虑离散控制问题, 即动作空间 $\mathcal{A}^1, \dots, \mathcal{A}^m$ 都是离散集合。MAC-A2C 使用两类神经网络: 价值网络 v 与策略网络 π ; 见图 14.2、图 14.3。

所有智能体共用一个价值网络, 记作 $v(s; \mathbf{w})$, 它是对状态价值函数 $V_\pi(s)$ 的近似。它把所有观测 $s = [o^1, \dots, o^m]$ 作为输入, 并输出一个实数, 作为对状态 s 的评分。

每个智能体有自己的策略网络。把第 i 号策略网络记作 $\pi(a^i|s; \theta^i)$ 。它的输入是所有智能体的观测 $s = [o^1, \dots, o^m]$ 。它的输出是一个向量, 表示动作空间 \mathcal{A}^i 上的概率分布。比如, 第 i 号智能体的动作空间是 $\mathcal{A}^i = \{\text{左}, \text{右}, \text{上}\}$; 策略网络的输出是

$$\pi(\text{左} | s; \theta^i) = 0.2, \quad \pi(\text{右} | s; \theta^i) = 0.1, \quad \pi(\text{上} | s; \theta^i) = 0.7.$$

《深度学习》2021-02-19 尚未校对, 仅供预览。
如发现错误, 请告知作者 shusen.wang@stevens.edu

第 i 号智能体依据该概率分布抽样得到动作 a^i 。

MAC-A2C 属于 Actor-Critic 方法：策略网络 $\pi(A^i|S; \theta^i)$ 相当于第 i 个运动员，负责做决策；价值网络 $v(S; \mathbf{w})$ 相当于评委，对运动员团队的整体表现予以评价，反馈给整个团队一个分数。

训练价值网络：我们用 TD 算法训练训练价值网络 $v(s; \mathbf{w})$ 。观测到状态 s_t 、 s_{t+1} 和奖励 r_t ，计算 TD 目标：

$$\hat{y}_t = r_t + \gamma \cdot v(s_{t+1}; \mathbf{w}).$$

把 \hat{y}_t 视作常数，更新 \mathbf{w} 使得 $v(s_t; \mathbf{w})$ 接近 \hat{y}_t 。定义损失函数：

$$L(\mathbf{w}) = \frac{1}{2} [v(s_t; \mathbf{w}) - \hat{y}_t]^2.$$

损失函数的梯度等于：

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = \delta_t \cdot \nabla_{\mathbf{w}} v(s_t; \mathbf{w}),$$

其中 $\delta_t = v(s_t; \mathbf{w}) - \hat{y}_t$ 是 TD 误差。做一次梯度下降更新 \mathbf{w} ：

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \delta_t \cdot \nabla_{\mathbf{w}} v(s_t; \mathbf{w}).$$

这样可以减小损失函数，也就是让 $v(s_t; \mathbf{w})$ 接近 \hat{y}_t 。上述 TD 算法与单智能体 A2C 的 TD 算法完全一样。

训练策略网络：完全合作关系设定下的动作价值函数记作 $Q_{\pi}(s, a)$ ，第 i 号智能体的策略网络为 $\pi(A^i | S; \theta^i)$ 。不难证明下面的策略梯度定理（见习题 1）：

定理 14.1. 合作关系 MARL 的策略梯度定理

设基线 b 为不依赖于 $A = [A^1, \dots, A^m]$ 的函数。那么有

$$\nabla_{\theta^i} J(\theta^1, \dots, \theta^m) = \mathbb{E}_{S, A} \left[(Q_{\pi}(S, A) - b) \cdot \nabla_{\theta^i} \ln \pi(A^i | S; \theta^i) \right].$$

期望中的动作 A 的概率质量函数为

$$\pi(A | S; \theta^1, \dots, \theta^m) \triangleq \pi(A^1 | S; \theta^1) \times \dots \times \pi(A^m | S; \theta^m).$$

把基线设置为状态价值： $b = V_{\pi}(s)$ 。定义

$$\mathbf{g}^i(s, a; \theta^i) \triangleq (Q_{\pi}(s, a) - V_{\pi}(s)) \cdot \nabla_{\theta^i} \ln \pi(a^i | s; \theta^i).$$

定理 14.1 说明 $\mathbf{g}^i(s, a; \theta^i)$ 是策略梯度的无偏估计，即

$$\nabla_{\theta^i} J(\theta^1, \dots, \theta^m) = \mathbb{E}_{S, A} [\mathbf{g}^i(S, A; \theta^i)].$$

因此 $\mathbf{g}^i(s, a; \theta^i)$ 可以作为策略梯度的近似。但是我们不知道公式中的 Q_{π} 、 V_{π} ，还需要进一步做近似。根据第 8.3 节 A2C 的推导，我们把 $Q_{\pi}(s_t, a_t)$ 近似成 $r_t + \gamma \cdot v(s_{t+1}; \mathbf{w})$ ，把 $V_{\pi}(s_t)$ 近似成 $v(s_t; \mathbf{w})$ 。那么近似策略梯度 $\mathbf{g}^i(s_t, a_t; \theta^i)$ 可以进一步近似成：

$$\tilde{\mathbf{g}}^i(s_t, a_t; \theta^i) \triangleq \underbrace{\left(r_t + \gamma \cdot v(s_{t+1}; \mathbf{w}) - v(s_t; \mathbf{w}) \right)}_{\text{对 } Q_{\pi}(s_t, a_t) - V_{\pi}(s_t) \text{ 的近似}} \cdot \nabla_{\theta^i} \ln \pi(a_t^i | s_t; \theta^i).$$

观测到状态 s_t 、 s_{t+1} 、动作 a_t^i 、奖励 r_t ，这样更新策略网络参数：

$$\theta^i \leftarrow \theta^i + \beta \cdot \tilde{g}^i(s_t, a_t^i; \theta^i).$$

根据 TD 误差 δ_t 的定义，不难看出 $\tilde{g}^i(s_t, a_t^i; \theta^i) = -\delta_t \cdot \nabla_{\theta^i} \pi(a_t^i | s_t; \theta^i)$ 。因此，上面更新策略网络参数的公式可以写作：

$$\theta^i \leftarrow \theta^i - \beta \cdot \delta_t \cdot \nabla_{\theta^i} \pi(a_t^i | s_t; \theta^i).$$

14.2.2 训练和决策

训练： 实际实现的时候，应当使用目标网络缓解自举造成的偏差。目标网络记作 $v(s; \mathbf{w}^-)$ ，它的结构与 v 相同，但是参数不同。设当前价值网络和目标网络的参数分别是 \mathbf{w}_{now} 和 $\mathbf{w}_{\text{now}}^-$ 。设当前 m 个策略网络的参数分别是 $\theta_{\text{now}}^1, \dots, \theta_{\text{now}}^m$ 。MAC-A2C 重复下面的步骤更新参数：

1. 观测到当前状态 $s_t = [o_t^1, \dots, o_t^m]$ ，让每一个智能体独立做随机抽样：

$$a_t^i \sim \pi(\cdot | s_t; \theta_{\text{now}}^i), \quad \forall i = 1, \dots, m,$$

并执行选中的动作。

2. 从环境中观测到奖励 r_t 与下一时刻状态 $s_{t+1} = [o_{t+1}^1, \dots, o_{t+1}^m]$ 。
3. 让价值网络做预测： $\hat{v}_t = v(s_t; \mathbf{w}_{\text{now}})$ 。
4. 让目标网络做预测： $\hat{v}_{t+1}^- = v(s_{t+1}; \mathbf{w}_{\text{now}}^-)$ 。
5. 计算 TD 目标与 TD 误差：

$$\hat{y}_t^- = r_t + \gamma \cdot \hat{v}_{t+1}^-, \quad \delta_t = \hat{v}_t - \hat{y}_t^-.$$

6. 更新价值网络参数：

$$\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{now}} - \alpha \cdot \delta_t \cdot \nabla_{\mathbf{w}} v(s_t; \mathbf{w}_{\text{now}}).$$

7. 更新目标网络参数：

$$\mathbf{w}_{\text{new}}^- \leftarrow \tau \cdot \mathbf{w}_{\text{new}} + (1 - \tau) \cdot \mathbf{w}_{\text{now}}^-.$$

8. 更新策略网络参数：

$$\theta_{\text{new}}^i \leftarrow \theta_{\text{now}}^i - \beta \cdot \delta_t \cdot \nabla_{\theta^i} \ln \pi(a_t^i | s_t; \theta_{\text{now}}^i), \quad \forall i = 1, \dots, m.$$

MAC-A2C 属于同策略 (On-policy)，不能使用经验回放。

决策： 在完成训练之后，不再需要价值网络 $v(s; \mathbf{w})$ 。每个智能体可以用它自己的策略网络做决策。在时刻 t 观测到全局状态 $s_t = [o_t^1, \dots, o_t^m]$ ，然后做随机抽样得到动作：

$$a_t^i \sim \pi(\cdot | s_t; \theta^i),$$

并执行动作。注意，智能体并不能独立做决策，因为一个智能体的策略网络需要知道其他所有智能体的观测。

14.2.3 实现中的难点

上述 MAC-A2C 的训练和决策貌似简单，然而实现起来却不容易。在 MARL 的常见设定下，第 i 号智能体只知道 o^i ，而观测不到全局状态：

$$s = [o^1, \dots, o^m].$$

这会给决策和训练造成如下的困难：

- 每个智能体有自己的策略网络 $\pi(a^i|s; \theta^i)$ ，可以依靠它做**决策**。但是它的决策需要全局状态 s 。
- 在**训练**的过程中，价值网络 $v(s; w)$ 需要知道全局状态 s 才能计算 TD 误差 δ 与梯度 $\nabla_w v(s; w)$ 。
- 在**训练**的过程中，每个策略网络都需要知道全局状态 s 来计算梯度 $\nabla_{\theta^i} \ln \pi(a^i|s; \theta^i)$ 。

综上所述，如果智能体之间不交换信息，那么智能体既无法做训练，也无法做决策。想要做训练和决策，有两种可行的途径：

- 一种办法是让智能体**共享观测**。这需要通信，每个智能体把自己的 o^i 传输给其他智能体。这样每个智能体都有全局的状态 $s = [o^1, \dots, o^m]$ 。
- 另一种办法是对策略网络和价值函数**做近似**。通常使用 $\pi(a^i|o^i; \theta^i)$ 替代 $\pi(a^i|s; \theta^i)$ 。甚至可以进一步用 $v(o^i; w^i)$ 代替 $v(s; w)$ 。

共享观测的缺点在于通信会让训练和决策的速度变慢。而做近似的缺点在于不完全信息造成训练不收敛、做出错误决策。我们不得不在两种办法之间做出取舍，承受其造成的不良影响。

下一节介绍**中心化** (Centralized) 与**去中心化** (Decentralized) 的实现方法。中心化让智能体共享信息；优点是训练和决策的效果好，缺点是需要通信，造成延时，影响速度。去中心化需要做近似，避免通信；其优点在于速度快，而缺点则是影响训练和决策的质量。

14.3 三种架构

本节介绍 MAC-A2C 的三种实现方法。第 14.3.1 节介绍“**中心化训练 + 中心化决策**” (Centralized Training with Centralized Execution)，它是对 MAC-A2C 的忠实实现，训练和决策都需要通信。第 14.3.2 节介绍“**去中心化训练 + 去中心化决策**” (Decentralized Training with Decentralized Execution)，它对策略网络和价值网络都做近似，以避免训练和决策的通信。第 14.3.3 节介绍“**中心化训练 + 去中心化决策**” (Centralized Training with Decentralized Execution)，它只近似策略网络以避免决策的通信，它的训练需要通信。

图 14.4 对比了三种架构的策略网络和价值网络。用“完全中心化”作出的决策最好，但是速度最慢，在很多问题中不适用。“中心化训练 + 去中心化决策”虽然在训练中需要通信，但是决策的时候不需要通信，可以做到实时决策。“中心化训练 + 去中心化决策”是三种架构中最实用的。

	价值网络	策略网络	训练	决策
中心化训练 + 中心化决策	$v(s; \mathbf{w})$	$\pi(a^i s; \theta^i)$	需要通信	需要通信
去中心化训练 + 去中心化决策	$v(o^i; \mathbf{w}^i)$	$\pi(a^i o^i; \theta^i)$	无需通信	无需通信
中心化训练 + 去中心化决策	$v(s; \mathbf{w})$	$\pi(a^i o^i; \theta^i)$	需要通信	无需通信

图 14.4: 三种架构的对比。

14.3.1 中心化训练 + 中心化决策

本节用完全中心化 (Fully Centralized) 的方式实现 MAC-A2C，没有做任何近似。这种实现的缺点在于通信造成延时，使得训练和决策速度变慢。图 14.5 描述了系统的架构。最上面是中央控制器 (Central Controller)，里面部署了价值网络 $v(s; \mathbf{w})$ 与所有 m 个策略网络

$$\pi(a^1 | \theta^1), \quad \pi(a^2 | \theta^2), \quad \dots, \quad \pi(a^m | \theta^m).$$

训练和决策全部由中央控制器完成。智能体负责与环境交互，执行中央控制器的决策 a^i ，并把观测到的 o^i 汇报给中央控制器。如果智能体观测到奖励 r^i ，也发给中央控制器。

中心化训练：在时刻 t 和 $t+1$ ，中央控制器收集到所有智能体的观测值

$$s_t = [o_t^1, \dots, o_t^m] \quad \text{和} \quad s_{t+1} = [o_{t+1}^1, \dots, o_{t+1}^m].$$

在“完全合作关系”的设定下，所有智能体有相同的奖励：

$$r_t^1 = r_t^2 = \dots = r_t^m \triangleq r_t.$$

r_t 可以是中央控制器直接从环境中观测到的，也可能是所有智能体本地的奖励 \tilde{r}_t^i 的加和：

$$r_t = \tilde{r}_t^1 + \tilde{r}_t^2 + \dots + \tilde{r}_t^m.$$

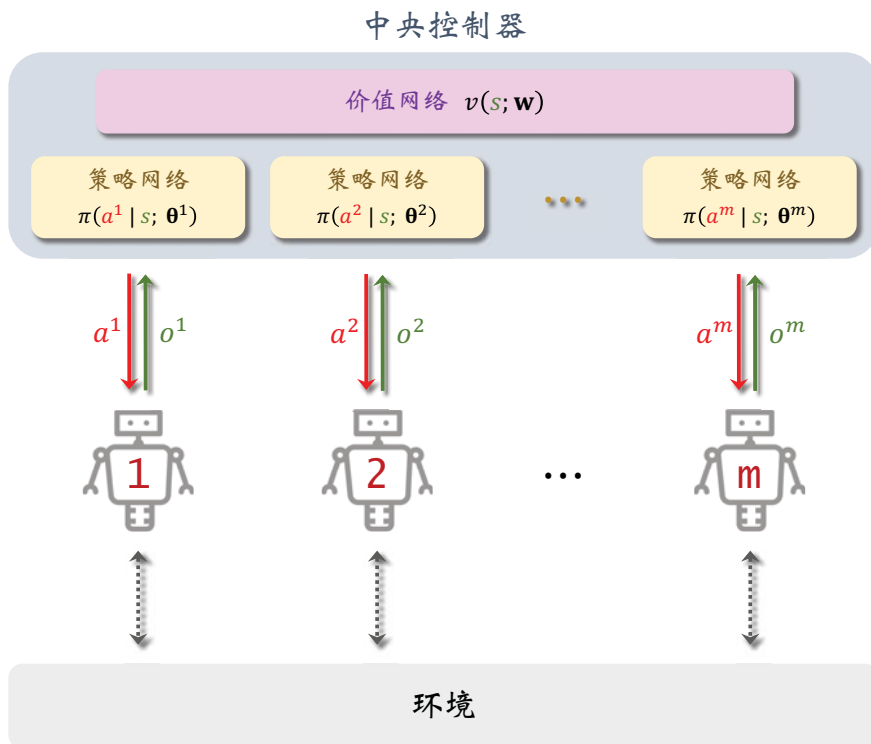


图 14.5: 中心化训练 + 中心化决策的系统架构。

决策是中央控制器上的策略网络做出的，中央控制器因此知道所有的动作：

$$a_t = [a_t^1, \dots, a_t^m].$$

综上所述，中央控制器知道如下信息：

$$s_t, s_{t+1}, a_t, r_t.$$

因此，中央控制器有足够的信息按照第 14.2.2 小节中的算法训练 MAC-A2C，更新价值网络的参数 w 和策略网络的参数 $\theta^1, \dots, \theta^m$ 。

中心化决策： 在 t 时刻，中央控制器收集到所有智能体的观测值 $s_t = [o_t^1, \dots, o_t^m]$ ，然后用中央控制器上部署的策略网络做决策：

$$a_t^i \sim \pi(\cdot | s_t; \theta^i), \quad \forall i = 1, \dots, m.$$

中央控制器把决策 a_t^i 传达给第 i 号智能体，该智能体执行 a_t^i 。综上所述，智能体只需要执行中央下达的决策，而不需要自己“思考”。其原因在于策略函数 π 需要全局的状态 s_t 作为输入，而单个智能体不知道全局状态，没有能力单独做决策。

优缺点： 中心化训练 + 中心化决策的优点在于完全按照 MAC-A2C 的算法实现，没有做任何改动，因此可以确保正确性。基于全局的观测 $s_t = [o_t^1, \dots, o_t^m]$ 做中心化的决策，利用完整的信息，因此作出的决策可以更好。中心化训练和决策的缺点在于延迟 (Latency) 很大，影响训练和决策的速度。在中心化执行的框架下，智能体与中央控制器要做通信。第 i 号智能体要把 o_t^i 传输给中央控制器，而控制器要在收集到所有观测 $[o_t^1, \dots, o_t^m]$ 之

后才会做决策，做出的决策 a_t^i 还得传输给第 i 号智能体。这个过程通常比较慢，使得实时决策不可能做到。机器人、无人车、无人机等应用都需要实时决策，比如在几十毫秒内做出决策；如果出现几百毫秒、甚至几秒的延迟，可能会造成灾难性的后果。

14.3.2 去中心化训练 + 去中心化决策

上一小节的“中心化训练 + 中心化决策”严格按照 MAC-A2C 的算法实现，其缺点在于训练和决策都需要智能体与中央控制器之间通信，造成训练的决策的速度慢。想要避免通信代价，就不得不对策略网络和价值网络做近似。MAC-A2C 中的策略网络

$$\pi(a^1 | s; \theta^1), \quad \pi(a^2 | s; \theta^2), \quad \dots, \quad \pi(a^m | s; \theta^m),$$

和价值网络 $v(s; \mathbf{w})$ 都需要全局的观测 $s = [o^1, \dots, o^m]$ 。“去中心化训练 + 去中心化决策”的基本思想是用局部观测 o^i 代替 s ，把策略网络和价值网络近似成为：

$$\pi(a^i | o^i; \theta^i) \quad \text{和} \quad v(o^i; \mathbf{w}^i).$$

在每个智能体上部署一个策略网络和一个价值网络，它们的参数记作 θ^i 和 \mathbf{w}^i 。智能体之间不共享参数，即 $\theta^i \neq \theta^j$, $\mathbf{w}^i \neq \mathbf{w}^j$ 。这样一来，训练就可以在智能体本地完成，无需中央控制器的参与，无需任何通信。见图 14.5 中的系统架构。

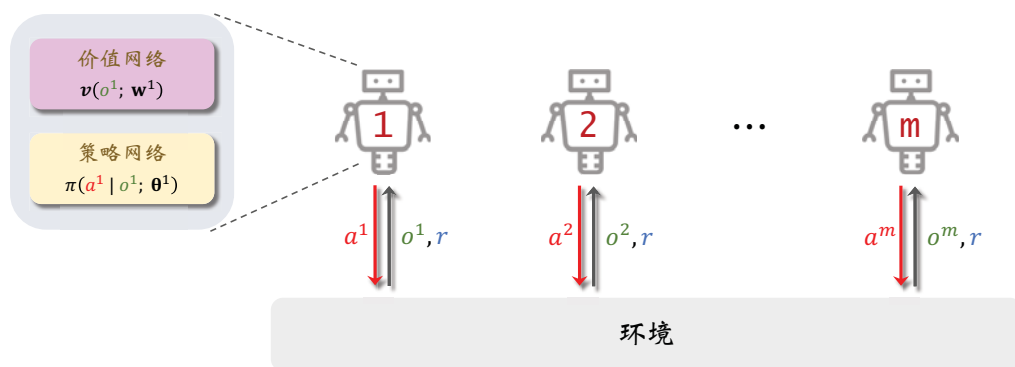


图 14.6: 去中心化训练 + 去中心化决策的系统架构。这种方法也叫做 Independent Actor-Critic。

去中心化训练：假设所有智能体的奖励都是相同的，而且每个智能体都能观测到奖励 r 。每个智能体独立做训练，智能体之间不做通信，不共享观测、动作、参数。这样一来，MAC-A2C 就变成了标准的 A2C，每个智能体独立学习自己的参数 θ^i 与 \mathbf{w}^i 。

实际实现的时候，每个智能体还需要一个目标网络，记作 $v(s; \mathbf{w}^{i-})$ ，它的结构与 $v(s; \mathbf{w}^i)$ 相同，但是参数不同。设第 i 号智能体的策略网络、价值网络、目标网络当前参数分别为 θ_{now}^i 、 $\mathbf{w}_{\text{now}}^i$ 、 $\mathbf{w}_{\text{now}}^{i-}$ 。该智能体重复以下步骤更新参数：

1. 在 t 时刻，智能体 i 观测到 o_t^i ，然后做随机抽样 $a_t^i \sim \pi(\cdot | o_t^i; \theta^i)$ ，并执行选中的动作 a_t^i 。
2. 环境反馈给智能体奖励 r_t 与新的观测 o_{t+1}^i 。
3. 让价值网络做预测： $\hat{v}_t^i = v(o_t^i; \mathbf{w}_{\text{now}}^i)$ 。
4. 让目标网络做预测： $\hat{v}_{t+1}^i = v(o_{t+1}^i; \mathbf{w}_{\text{now}}^{i-})$ 。

5. 计算 TD 目标与 TD 误差:

$$\hat{y}_t^i = r_t + \gamma \cdot \hat{v}_{t+1}^i, \quad \delta_t^i = \hat{v}_t^i - \hat{y}_t^i.$$

6. 更新价值网络参数:

$$\mathbf{w}_{\text{new}}^i \leftarrow \mathbf{w}_{\text{now}}^i - \alpha \cdot \delta_t^i \cdot \nabla_{\mathbf{w}^i} v(o_t^i; \mathbf{w}_{\text{now}}^i).$$

7. 更新目标网络参数:

$$\mathbf{w}_{\text{new}}^{i-} \leftarrow \tau \cdot \mathbf{w}_{\text{new}}^i + (1 - \tau) \cdot \mathbf{w}_{\text{now}}^{i-}.$$

8. 更新策略网络参数:

$$\boldsymbol{\theta}_{\text{new}}^i \leftarrow \boldsymbol{\theta}_{\text{now}}^i - \beta \cdot \delta_t^i \cdot \nabla_{\boldsymbol{\theta}^i} \ln \pi(a_t^i | o_t^i; \boldsymbol{\theta}_{\text{now}}^i), \quad \forall i = 1, \dots, m.$$

注 上述算法不是 MAC-A2C, 而是单智能体的 A2C。去中心化训练的本质就是单智能体强化学习 (SARL), 而非多智能体强化学习 (MARL)。在 MARL 中, 智能体之间会相互影响, 而本节中的“去中心化训练”把智能体视为独立个体, 忽视它们之间的关联, 直接用 SARL 方法独立训练每个智能体。用上述 SARL 的方法解决 MARL 问题, 在实践中效果往往不佳。

去中心化决策: 在完成训练之后, 智能体 i 不再需要其价值网络 $v(o^i; \mathbf{w}^i)$ 。智能体只需要用其本地部署的策略网络 $\pi(a^i | o^i; \boldsymbol{\theta}^i)$ 做决策即可, 决策过程无需通信。去中心化执行的速度很快, 可以做到实时决策。

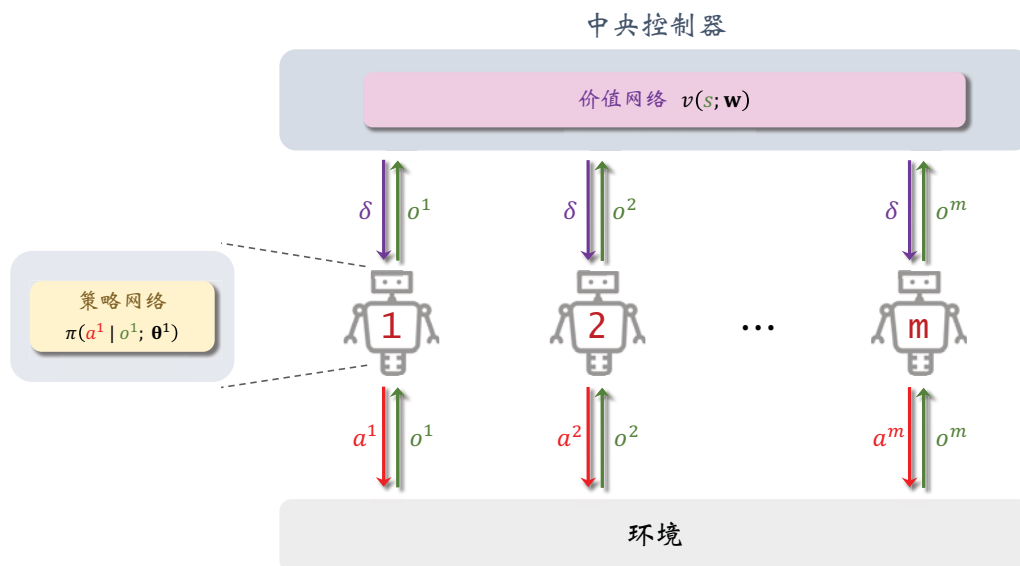


图 14.7: 中心化训练的系统架构。价值网络（以及没画出的目标网络）部署到中央控制器上，策略网络部署到每个智能体上。训练的时候，智能体 i 将观测 o^i 传输到控制器上，控制器将 TD 误差 δ 传回智能体。

14.3.3 中心化训练 + 去中心化决策

前面两节讨论了完全中心化与完全去中心化，两种实现各有优缺点。当前更流行的 MARL 架构是“中心化训练 + 去中心化决策”。训练的时候使用中央控制器，辅助智能体做训练；见图 14.7。训练结束之后，不再需要中央控制器，每个智能体独立根据本地观测 o^i 做决策；见图 14.8。

本小节与“完全中心化”使用相同的价值网络 $v(s; \mathbf{w})$ 及其目标网络 $v(s; \mathbf{w}^-)$ ；本节与“完全去中心化”使用相同的策略网络：

$$\pi(a^1 | o^1; \theta^1), \quad \dots, \quad \pi(a^m | o^m; \theta^m).$$

第 i 号策略网络的输入是局部观测 o^i ，因此可以将其部署到第 i 号智能体上。价值网络 $v(s; \mathbf{w})$ 的输入是全局状态 $s = [o^1, \dots, o^m]$ ，因此需要将其部署到中央控制器上。

中心化训练：训练的过程需要所有 m 个智能体共同参与，共同改进策略网络参数 $\theta^1, \dots, \theta^m$ 与价值网络参数 \mathbf{w} 。设当前 m 个策略网络的参数为 $\theta_{\text{now}}^1, \dots, \theta_{\text{now}}^m$ 。设当前价值网络和目标网络的参数分别是 \mathbf{w}_{now} 和 $\mathbf{w}_{\text{now}}^-$ 。训练的流程如下：

1. 每个智能体 i 与环境交互，获取当前观测 o_t^i ，独立做随机抽样：

$$a_t^i \sim \pi(\cdot | o_t^i; \theta_{\text{now}}^i), \quad \forall i = 1, \dots, m, \quad (14.2)$$

并执行选中的动作。

2. 下一时刻，每个智能体 i 都观测到 o_{t+1}^i 。假设中央控制器可以从环境获取奖励 r_t ，或者向智能体询问奖励 r_t 。
3. 每个智能体 i 向中央控制器传输观测 o_t^i 和 o_{t+1}^i ；中央控制器得到状态

$$s_t = [o_t^1, \dots, o_t^m] \quad \text{和} \quad s_{t+1} = [o_{t+1}^1, \dots, o_{t+1}^m].$$

4. 中央控制器让价值网络做预测： $\hat{v}_t = v(s_t; \mathbf{w}_{\text{now}})$ 。
5. 中央控制器让目标网络做预测： $\hat{v}_{t+1}^- = v(s_{t+1}; \mathbf{w}_{\text{now}}^-)$ 。
6. 中央控制器计算 TD 目标和 TD 误差：

$$\hat{y}_t^- = r_t + \gamma \cdot \hat{v}_{t+1}^-, \quad \delta_t = \hat{v}_t - \hat{y}_t^-,$$

并将 δ_t 广播到所有智能体。

7. 中央控制器更新价值网络参数：

$$\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{now}} - \alpha \cdot \delta_t \cdot \nabla_{\mathbf{w}} v(s_t; \mathbf{w}_{\text{now}}).$$

8. 中央控制器更新目标网络参数：

$$\mathbf{w}_{\text{new}}^- \leftarrow \tau \cdot \mathbf{w}_{\text{new}} + (1 - \tau) \cdot \mathbf{w}_{\text{now}}^-.$$

9. 每个智能体 i 更新策略网络参数：

$$\theta_{\text{new}}^i \leftarrow \theta_{\text{now}}^i - \beta \cdot \delta_t \cdot \nabla_{\theta^i} \ln \pi(a_t^i | o_t^i; \theta_{\text{now}}^i).$$

注 此处的算法并不等价于第 14.2 节的 MAC-A2C。区别在于此处用 $\pi(a^i | o^i; \theta^i)$ 代替 MAC-A2C 中的 $\pi(a^i | s; \theta^i)$ 。

去中心化决策：在完成训练之后，不再需要价值网络 $v(s; \mathbf{w})$ 。智能体只需要用其

本地部署的策略网络 $\pi(a^i|o^i;\theta^i)$ 做决策，决策过程无需通信。去中心化执行的速度很快，可以做到实时决策。

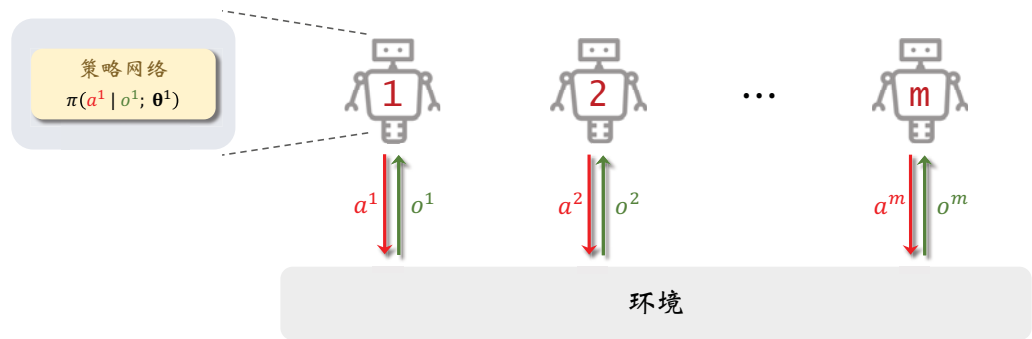


图 14.8: 去中心化决策的系统架构。在完成训练之后，智能体不再做通信，智能体用本地部署的策略网络做决策。

《深度学习》2021-02-19 尚未校对，仅供预览。
如发现错误，请告知作者 shusen.wang@stevens.edu

第十四章 习题

1. 设动作 $A = [A^1, \dots, A^m]$ 的概率质量函数为

$$\pi(A | S; \theta^1, \dots, \theta^m) \triangleq \pi(A^1 | S; \theta^1) \times \dots \times \pi(A^m | S; \theta^m).$$

由第 8 章中带基线的策略梯度定理可得：

$$\nabla_{\theta^i} J(\theta^1, \dots, \theta^m) = \mathbb{E}_{S,A} \left[\left(Q_{\pi}(S, A) - b \right) \cdot \nabla_{\theta^i} \ln \pi(A | S; \theta^1, \dots, \theta^m) \right].$$

公式中动作 A 的概率质量函数为 $\pi(A | S; \theta^1, \dots, \theta^m)$ ，公式中的 b 是任意不依赖于 A 的函数。请用上面两个公式证明下面的公式：

$$\nabla_{\theta^i} J(\theta^1, \dots, \theta^m) = \mathbb{E}_{S,A} \left[\left(Q_{\pi}(S, A) - V_{\pi}(S) \right) \cdot \nabla_{\theta^i} \ln \pi(A^i | S; \theta^i) \right].$$

第十四章 相关文献

完全去中心化的架构早在 1993 年就被提出 [101], 在 2017 年被用在多智能体 DQN 上 [38, 100]。中心化训练 + 去中心化执行 (Centralized Training with Decentralized Execution) 在近年来很流行 [77, 42, 37, 65, 52]。

MAC-A2C 是本书设计出来的简单方法, 用于讲解 MARL 的三种架构; MAC-A2C 这个名字并没有出现在任何文献中。MAC-A2C 本质是带基线的 Actor-Critic, 其中的基线是状态价值

$$V_{\pi}(s) \triangleq \mathbb{E}_A [Q_{\pi}(s, A)],$$

期望是关于动作 $A = [A^1, \dots, A^m]$ 求的。可以把基线换成

$$Q_{\pi}^{-i}(s, a^{-i}) \triangleq \mathbb{E}_{A^i} [Q_{\pi}(s, A^i, a^{-i})],$$

公式中 $a^{-i} = [a^1, \dots, a^{i-1}, a^{i+1}, \dots, a^m]$ 。公式中的期望是关于第 i 号智能体的动作 $A^i \sim \pi(\cdot | o^i, \theta^i)$ 求的。用 $Q_{\pi}^{-i}(s, a^{-i})$ 作为基线, 代替 $V_{\pi}(s)$, 得到的方法叫做 **CO**unterfactual **M**ulti-**A**gent, 缩写 **COMA** [37]。此外, COMA 还在策略网络中使用 RNN; 其原理见第 11 章的解释。COMA 的表现略好于 MAC-A2C, 但是 COMA 的实现很复杂, 不建议读者自己实现。

《深度学习》2021-02-19 尚未校对, 仅供预览。
如发现错误, 请告知作者 shusen.wang@stevens.edu