# Multi-Agent Actor-Critic for Mixed

# Cooperative-Competitive Environments

- 摘要

MA 环境是动态的，我们先分析传统方法在 MA 环境中的难点，Q-learning 在不稳定的环境中受阻；PG 方法在智能体数量增加时有很大的方差。作者提出的一种方法是基于 AC 的，该算法考虑了其他智能体的行为策略，能在需要复杂协作环境中成功学习策，此外使用一种训练方案，利用每个智能体的策略信息使得 MA 环境下的智能体策略更具有鲁棒性。

该方法适用于 MA 协作和竞争环境中。

- 介绍

由于 MA 环境过于复杂，使用 model-base 方法需要复杂建模，也不适用。

因此作者提出一个通用 MA 学习方法：(1)在执行阶段引导学习只使用局部信息，即智能体自己的观测的策略。(2)在智能体的通信方法中，不使用环境动力学或者任何一种结构的可微模型，即是 model-free 方法。(3)不仅应用在合作策略上，也应用在竞争或者带有物理或者通信行为的混合策略中。具有混合策略行为能力的智能体可能更关键；竞争在训练中自然而然，但也希望有合作。

作者使用集中化训练和分散式执行的框架，即训练统一训练，执行每个智能体独自执行不同的动作，这些不同动作的组合形成不同的统一策略，可以是协作可以是竞争。允许使用额外信息减轻训练，但测试时只使用观测信息。

因为 Q-learning 不能处理不同类型的信息和测试时间，所以在不做任何环境假设的情况下，Q-learning 可以运作是不自然的。作者使用 AC 方法的拓展，其中 critic 使用其他智能体策略的额外信息，同时 actor 只接收智能体的局部信息。

因为中心化的 critic 显式地使用其他智能体的策略，所以该方法可以在线地将其他智能体的策略大概模型进行学习，同时有效地在它们学习策略过程中使用。我们还引入了一种方法来提高 MA 策略的稳定性，该方法是用一组策略训练智能体(一组一组策略进行学习)，从而需要与各种协作者和竞争者进行稳健的交互。

- 相关工作

最简单的学习 MA 方法就是独立学习每个智能体。Q-learning 就是其中之一但不够实用，PG 也如此。

交互自然就是合作、竞争或者混合，多数的算法只针对其中一种交互。目前多数合作设置，在假定其他智能体采取了使得 reward 增大的动作情况下，优化或者滞后 Q 函数更新等。另一种非直接的协作方法就是共享策略参数，但该算法要求具有同性质的智能体。这些方法不适用于竞争或者混合情况。

对照来讲，作者提出一种相似的带有中心化 critic 的 PG 方法，同时在星际争霸环境中进行测试。以上方法和作者不同的点有：(1)**以上方法为所有智能体学习一个中心化的 critic，而作者为每个智能体学习一个中心化的 critic，允许每个智能体具有不同 reward 功能，包括竞争场景。**(2)作者考虑具有显式通信的环境。(3)**以上方法使用重复策略和前馈 critic 结合，作者方法使用前馈 critic(虽然该方法也能适用于重复策略**。(4)作者方法学习**连续策略**，而以上方法学习离散策略。

最近的工作重点是学习智能体之间的基础合作通信协议，以解决各种任务。但这些方法一般只适用于专门的、可微的通信通道中。作者的方法显式地使用其他智能体的策略，增加

鲁棒性，建立贝叶斯决策模型也是如此。作者同时也是用一组成功交互的策略对智能体进行训练。

- 背景

**PG**

策略梯度下降也是另一种流行方法，主要观点是调整策略参数使得目标函数 J 最大化，每一步前进 J 的梯度。公式如下：

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^{\boldsymbol{\pi}}, a\sim \boldsymbol{\pi}_\theta}[\nabla_\theta \log \boldsymbol{\pi}_\theta(a|s) Q^{\boldsymbol{\pi}}(s,a)],$$

pΠ表示状态分布，Π θ 是动作分布，PG 理论为许多算法提供了实用理论，人们可以通过学习近似 Q 函数作为 critic 组成许多 AC 方法。由于 MA 环境下，一个智能体的 reward 由许多智能体来决定。简单 PG 方法是只考虑一个智能体行为而不考虑其他智能体行为，但不同的其他智能体行为将得到不同的 reward，因此导致了 PG 方法的高方差。

下面，作者给出了一个简单的设置，其中在正确方向上采取梯度步骤的概率随着智能体数的增加呈指数下降。即随着智能体数量的增加，简单方法在正确方向行动的可能性在减少，同时减少速度呈指数下降。

**Proposition 1.** *Consider $N$ agents with binary actions: $P(a_i = 1) = \theta_i$, where $R(a_1, \ldots, a_N) = \mathbf{1}_{a_1=\cdots=a_N}$. We assume an uninformed scenario, in which agents are initialized to $\theta_i = 0.5\ \forall i$. Then, if we are estimating the gradient of the cost $J$ with policy gradient, we have:*

$$P(\langle \hat{\nabla} J, \nabla J\rangle > 0) \propto (0.5)^N$$

*where $\hat{\nabla} J$ is the policy gradient estimator from a single sample, and $\nabla J$ is the true gradient.*

上述公式左边表达，估计梯度和真实梯度的接近程度，>0 表示非正交同时接近真实梯度，因此左边表达正确方向的概率，它是正比于动作集的均匀分布的指数。

PG 方法使用 baseline，例如用状态函数作为 baseline，可在单智能体问题中有效改善高方差问题，但是在 MA 环境中无法有效解决不稳定问题。

**DPG(Deterministic PG)**

$$\boldsymbol{\mu}_\theta : \mathcal{S} \mapsto \mathcal{A}\ [\textbf{?}\ ].$$

将确定策略方法(Deterministic Policy)加入 PG 方法中，特别地我们在适当条件下，可以将目标函数梯度改为以下形式：

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\mathcal{D}}[\nabla_\theta \boldsymbol{\mu}_\theta(a|s) \nabla_a Q^{\boldsymbol{\mu}}(s,a)|_{a=\boldsymbol{\mu}_\theta(s)}]$$

由于使用策略的梯度，因此要求策略函数连续，等价于动作空间是连续的。

而 DDPG 是一种 DPG 方法的变种，在 DPG 框架下，将策略和 Q 估计的 critic 使用网络进行学习。Off-policy，也是用 target 网络。

- 方法

1. MA - AC

该算法实现的限制条件有：(1)在智能体执行期间所用信息只有智能体的局部可观测信息，即局部观测。(2)该方法是 model-free 的。(3)同时通信也没有固定模型结构，不设定任何形式信道。只要履行上述限制条件，就可以实现不仅在合作上可以显式通信的算法，同时该算法也能在智能体之间只有物理互动的竞争中发挥效果。

作者将额外信息加入训练，而执行中只使用局部观测信息。因为 Q 结构不具有该性质，因此作者使用 AC 结构拓展，使得 critic 使用其他智能体的额外信息进行增强。

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s\sim p^{\boldsymbol{\mu}}, a_i\sim \boldsymbol{\pi}_i}[\nabla_{\theta_i} \log \boldsymbol{\pi}_i(a_i|o_i) Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, ..., a_N)].$$

这里 Qi 是联合动作的 action-value 函数，带有 a1~an 的联合动作，也带有其他智能体的状态信息 X，但输出是对于智能体 i 的 Q value。X 最简单的形式是由所有智能体的观测组成，当然条件允许情况下，也可以增加额外的状态信息。由于对于每个智能体，Q 是独立的，因此可以学习出任意 reward 结构，包括竞争情况下的博弈结构。

也可以使用确定性策略进行扩展，DPG 变种。

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{\mathbf{x},a\sim\mathcal{D}}[\nabla_{\theta_i} \boldsymbol{\mu}_i(a_i|o_i) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, ..., a_N)|_{a_i=\boldsymbol{\mu}_i(o_i)}],$$

其中 D 集合包含有 X，X’，a1~an，r1~rn。而联合动作 Q 函数由以下公式更新：

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}[(Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N) - y)^2], \quad y = r_i + \gamma\, Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', a_1', \ldots, a_N')\big|_{a_j'=\boldsymbol{\mu}_j'(o_j)},$$

以上公式组成的算法叫 MADDPG。MADDPG 背后的一个主要动机是，如果我们知道所有智能体采取的行动，即使政策发生变化，环境也是静止的，因为已知当前状态和所有采取的动作还有策略Π，跳转到 s’的概率是等于已知当前状态和所有采取的动作还有策略Π’，跳转到 s’的概率，不会因为策略的变化而使得概率变化。

如果我们不像大多数传统的 RL 方法那样，显式地以其他智能体的动作为输入条件，情况就不是这样。该算法中是否了解其他智能体的观测并不是必要的。4.2 描述如何使用其他智能体的信息。

2. 其他智能体的推断策略

为了移除准确了解其他智能体策略的行为，对于每一个智能体 i 可以额外维护一个近似 $\hat{\boldsymbol{\mu}}_{\phi_i^j}$ 其中 ϕ 表示估计的策略参数。该近似通过最大化智能体 j 动作的对数概率，带有一个熵 z 正则化器。

$$\mathcal{L}(\phi_i^j) = -\mathbb{E}_{o_j,a_j}\left[\log \hat{\boldsymbol{\mu}}_i^j(a_j|o_j) + \lambda H(\hat{\boldsymbol{\mu}}_i^j)\right], \tag{7}$$

其中 H 为策略分布的熵，通过近似策略，y 可以通过下面公式计算：

$$\hat{y} = r_i + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', \hat{\boldsymbol{\mu}}_i'^1(o_1), \ldots, \boldsymbol{\mu}_i'(o_i), \ldots, \hat{\boldsymbol{\mu}}_i'^N(o_N)),$$

其中以上策略就是通过近似策略估计得到的。观测公式 7 注意到该式子可以完全在线方式进行优化：在更新 Q 之前，可以从 buffer 区中获取每个智能体 j 的最新行为，以执行单个梯度步骤来更新 ϕ 策略估计。而根据公式 8，Q 的更新是直接将及算法的 ϕ 估计策略带入计算，而不是从采样中计算。

3. 策略组合下的智能体

MA 中重复强调的问题就是策略变化导致的环境不稳定性，特别在竞争设定下很明显，该设定下，智能体可能会过度适应竞争对手的行为而学习到一个强有力的策略，该策略不稳定很脆弱，因为强对手智能体一旦改变策略就可能失效。为获得具有鲁棒性的策略，作者提

出一种方法，该方法将 K 种不同子策略集合用于训练。对于每一个片段，作者随机给每一个智能体选择一种特定子策略去执行，假定策略集合 μ 是 K 个不同子策略构成的策略集合，其中每个策略都由标号 k 和 i 的策略 μ ki 表达。对于智能体 i，算法希望最大化目标组合函数：

$$J_e(\boldsymbol{\mu}_i) = \mathbb{E}_{k\sim\text{unif}(1,K),s\sim p^{\boldsymbol{\mu}},a\sim\boldsymbol{\mu}_i^{(k)}}\left[R_i(s,a)\right].$$

由于不同的子策略会执行于不同的片段中，因此作者维护一个回放 buffer，因此，我们可以导出集合目标相对于 θ （K）i 的梯度，如下所示。

$$\nabla_{\theta_i^{(k)}} J_e(\boldsymbol{\mu}_i) = \frac{1}{K}\mathbb{E}_{\mathbf{x},a\sim\mathcal{D}_i^{(k)}}\left[\nabla_{\theta_i^{(k)}}\boldsymbol{\mu}_i^{(k)}(a_i|o_i)\nabla_{a_i}Q^{\boldsymbol{\mu}_i}(\mathbf{x},a_1,\ldots,a_N)\Big|_{a_i=\boldsymbol{\mu}_i^{(k)}(o_i)}\right]. \quad (9)$$

- 算法伪代码

**Multi-Agent Deep Deterministic Policy Gradient Algorithm**

For completeness, we provide the MADDPG algorithm below.

**Algorithm 1:** Multi-Agent Deep Deterministic Policy Gradient for $N$ agents

**for** episode $= 1$ to $M$ **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state $\mathbf{x}$
  **for** $t = 1$ to max-episode-length **do**
    for each agent $i$, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_N)$ and observe reward $r$ and new state $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    **for** agent $i = 1$ to $N$ **do**
      Sample a random minibatch of $S$ samples $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ from $\mathcal{D}$
      Set $y^j = r_i^j + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}'^j, a'_1, \ldots, a'_N)|_{a'_k=\boldsymbol{\mu}'_k(o_k^j)}$
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor using the sampled policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\boldsymbol{\mu}_i(o_i^j)\nabla_{a_i}Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j)\Big|_{a_i=\boldsymbol{\mu}_i(o_i^j)}$$

    **end for**
    Update target network parameters for each agent $i$:

$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$$

  **end for**
**end for**

- 实验
  1. 环境
  2. 与非联合动作强化学习方法进行比较
  3. 其他智能体学习策略的影响
  4. 使用一组策略训练策略的影响

- 总结

作者方法的一个缺点是 Q 的输入空间呈线性增长，依赖于 N 个智能体的观测信息多少。具有模块化的 Q 函数，即有一组 Q 函数将智能体划分，该划分中的 Q 函数只考虑给定智能体的某一邻域中的代理。