Multi-Agent Actor-Critic for Mixed

Cooperative-Competitive Environments

文章探索了多智能体领域中使用深度强化学习方法。先分析了传统方法：Q-learning 很难处理内部不稳定的环境，策略梯度受到智能体数量增长的限制。接着，文章展示了一种自适应 actor-critic 方法，这种方法在制定策略的时候会考虑其他智能体的行为策略，并能成功学习到需要多智能体协作的复杂策略。还介绍了一种为每一个智能体都应用全局策略的训练方法，提高了多智能体策略的鲁棒性。在竞争与协作的任务中，这个方法比现有的方法都强大，智能体群体也能够学习到许多种协作策略。

1.介绍

强化学习在某些领域的应用解决了许多挑战性的问题，比如游戏 Atari 2600、围棋、机器人等。在工业上，强化学习也开始崭露头角，比如数据中心冷却这样的大型系统。但是，很多强化学习的应用都是在单个智能体领域，在单智能体系统中，对环境中其他智能体建模或者预测行为都是不必要的。

然而，也有许多重要的应用涉及到了多智能体之间交互，包括突发行为和多智能体共同演化的复杂性。举个例子，多机器人控制、交流和语言的发现、多玩家的游戏、社交困境的分析、多智能体环境下的相关操作。其中的存在的难题：比如分级强化学习的变体也可以看作是一个多智能体系统，每一层级都是一个智能体。另外，多智能体的自我博弈也是挺有用训练方法。能够成功地将强化学习应用于多智能体是构造具有与人类交互和彼此交互的人工智能的关键。

不幸的是，类似 Q-learning 和策略梯度等传统强化学习并不能适用多智能体。一个原因是，训练时的每一个智能体的策略是在不断变化的。另一个原因，环境由于每个独立智能体的 **perspective***(没有理解这个单词的意思)*而变得不稳定（某种程度上，这并不能用智能体本身策略的变化来解释）。这表明学习过程的稳定性阻碍了经验回溯的直接使用，尽管这个方法是深度 Q-learning 算法稳定运行的关键。另一方面，策略梯度算法在多智能体之间协作的时候就变得非常不稳定。还有一种基于策略优化的方法，通过反向传播来学习优化策略，但是这需要一个可微分的环境动态模型并假设智能体之间都有交互。因为对抗训练方法的不稳定性，所以将这些方法应用到竞争环境中时的优化也将是一个挑战。

文中的工作，设想了一个通用多智能体学习算法如下：（1）在每一个时间片内只使用局部信息学习策略（比如他们各自的观察）。（2）不假设一个可微分的动态环境或者智能体之间的交流方式上有任何特殊的结构。（3）选择合适的交互形式，不仅有竞争还有协作。智能体能够在竞争合作的混合环境下如何执行行为是问题的关键。在学习到竞争策略的时候还要能够展示协作行为（比如和人类）。

文章采用了集中训练但分散执行的框架，允许策略在训练时可以使用额外的信息，只是这些信息在测试的时候不使用到。在这种情况下，不宜采用不带环境结构信息假设的 Q-learning，因为 Q 函数在训练和测试的时候通常包含不同的信息。因此，提出了一种基于 actor-critic 策略梯度的简单扩展方法。Critic（评价器）使用了其他智能体的策略信息，而 actor 只使用局部环境信息。在训练完成后，只有局部 actors（执行器）被用来执行动作，在去中心化管理中执行，并且同样地应用在竞争与协作环境中。

由于中心化的评价函数直接使用了其他智能体的策略，文章展示了智能体能够在线估计其他的智能体模型并且在自我决策的时候有效利用了这些估计信息。还介绍了一种提高多智能体策略稳定性的方法，这种方法在训练时使用了全部的策略信息，因此需要大量的竞争者

和协作者策略信息。从结果来看，文章的方法同现有的方法在竞争与协作的场景下相比是有效的，智能体群体能够发现复杂的动作和交流协作策略。

2 相关工作

多智能体中最简单的学习方法就是各个智能体独立学习。其中，Q-learing 表现的并不好。其中的一个问题是训练过程中策略变化导致的环境不稳定进而阻碍了经验回溯算法。也有一些尝试来解决这个问题，在 Q 函数中输入其他智能体的策略，直接在回溯缓存中添加迭代索引，或者使用重要性采样。

智能体之间的交互包括竞争、协作或二者都有，但是很多算法在设计的时候只考虑了一种情况。像优化和 Q 函数滞后更新策略只考虑了协作，这里假设了其他智能体的动作可以用来提高累计奖励。另一种直接实现协作的方法是共享策略参数，但是这需要智能体具有相同的功能。上述的这些算法都不适用竞争环境或者混合环境。

针对本文的问题，[文献 7]提出了一个简单的带有全局评价的策略梯度方法，并在星际争霸的微操任务中验证。改进的地方如下：（1）文献 7 中的方法对所有的智能体使用一个全局评价器，本文的方法为每一个智能体都学习一个全局评价器，这样可以让智能体在竞争场景下具有不同的奖励函数；（2）本文认为智能体具有明确的交互动作；（3）文献 7 中将周期策略 recurrent policies 和反馈评价结合在一起，本文的方法只使用反馈策略（尽管本文的方法也适用于周期策略）；（4）文献 7 学习离散策略，本文可以学习连续策略。

当前的研究热点都集中智能体之间交互的基础协议，这能解决许多问题。然而，这些方法都只适用于特殊、可微的通信方法。

本文的方法要对其他智能体的决策过程进行建模。无论是从强化学习还是认知科学的角度看，对决策过程建模都是很重要的。本文中让该智能体与环境中其他智能体的备选策略进行交互，提高了训练稳定性和训练后智能体的鲁棒性。

3 背景

Markov Games：有 N 个智能体，每个智能体都有状态集 S，可执行状态 A，观察值 O。每个智能体采用随机策略 π 选择动作，然后智能体根据状态转移函数切换到下一个状态。每个智能体都有一个包含状态和智能体动作的奖励函数，这个函数接收每个本体的观察值。初始状态服从分布 ρ，每个智能体的目标是最大化期望奖励$R_i \sum_{t=0}^{T} \gamma^t r^t$，其中 γ 是折扣因子，T 是时间范围。

Q-learning 和 DQN：

策略梯度算法（PG）：主要思路是直接应用策略的参数 θ 以最大化目标 J(θ)，通过求解$\nabla_\theta J(\theta)$，然后再使用之前定义好的 Q 函数。则可以写成：

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

其中$P^{\boldsymbol{\pi}}$是状态分布。PG 算法根据$Q^{\boldsymbol{\pi}}$的不同衍生出了许多变体。PG 算法尤其在多智能体环境下变得非常不稳定，因为一个智能体的奖励通常有很多智能体的动作决定，如果奖励条件只受单个个体约束的话将会很不稳定（优化的过程不考虑其他动作），因此增加了梯度的发散。接下来，展示了一种简易的设置可以使多智能体以指数形式衰减梯度的概率：

考虑 N 个只有两个动作的智能体$\mathrm{P}(a_i = 1) = \theta_i$，其中 $\mathrm{R}(a_1, a_2, \dots, a_N)=1_{a1=\cdots aN}$。假设在一个特定场景下，智能体初始化服从分布$\theta_i = 0.5 \ \forall i$，那么如果使用 PG 算法估计 J，有：

$$\mathrm{P}\left(\langle \widehat{\nabla} J, \nabla J\rangle > 0\right) \propto (0.5)^N$$

其中，$\widehat{\nabla} J$是单个采样的策略梯度估计值，∇J则是真实值。证明见附页。

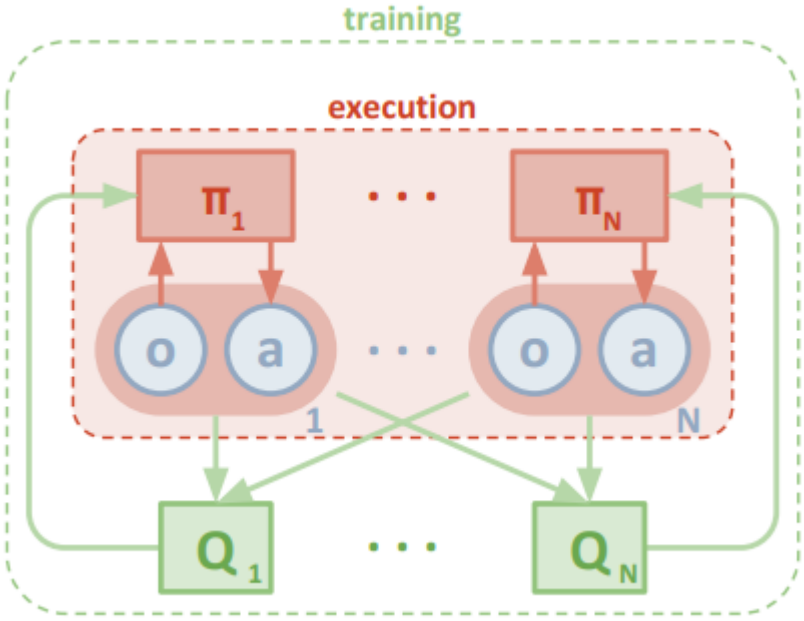
关于基线的使用，比如典型的值函数基线能够减轻发散，由于之前提到的不稳定问题，这在多智能体环境中是不确定的。

DPG 算法，将策略梯度框架扩展到决定策略μ_θ。特殊的，在某些条件下，可以将目标函数$\mathrm{J}(\theta)$写作：

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

因为这个定理依赖于$\nabla_a Q^\mu(s,a)$，需要动作空间 A 是连续的。

DDPG：DPG 的变形，使用深度神经网络近似策略 μ 和评价器Q^μ，DDPG 是一个 off-policy 算法，不断地从经验回溯缓存中采样。DDPG 也使用了 DQN 这样的目标网络。



图一. 全局评价和分散执行示意图

4.方法

4.1 多智能体动作评价

之前讨论了原始的策略梯度方法在多智能体环境中表现很差，在下面的实验中也可以看到。现考虑如下约束：（1）仅使用局部信息更新策略（各自智能体的观察）；（2）不假设环境动态模型是可微分的；（3）不假设智能体交互具有特殊结构的交流方式（不假设可微分交流通道）；满足上述条件就可以生成一个通用多智能体学习算法，适用于竞争和协作的场景。

解决这个问题的办法就是使用全局训练 centralized training 和分散执行 decentralized execution，让策略使用额外的信息来促进训练，只要这些信息在测试的时候不使用。这并不适合 Q-learning，因为 Q 函数通常在训练和测试的时候包含同样的信息。因此，可将 AC 策略梯度方法扩展一下：让评价器考虑其他智能体的策略信息。

具体而言，考虑一个 N 个智能体的游戏，策略参数$\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_N\}$,$\boldsymbol{\pi} = \{\boldsymbol{\pi_1}, \ldots, \boldsymbol{\pi_N}\}$是所有智能体策略的集合，接着可将智能体 i 的期望回报的梯度写作：

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

其中$Q_i^{\boldsymbol{\pi}}(\boldsymbol{X}, a_1, \ldots, a_N)$是全局动作价值函数，它的输入是所有智能体的动作、一些状态信息 X 和智能体 i 的 Q 值输出。在最简单的场景里，X 可以包含所有智能体的观察，$X = (o_1, \ldots, o_N)$，有必要的话还可以包含其他的状态信息。由于每一个$Q_i^{\boldsymbol{\pi}}$都是独立学习的，所以可以适用任意结构的奖励机制，包括竞争环境中冲突奖励 conflicting rewards。

可以将上述的方法扩展到决定性策略 deterministic policies。如果考虑 N 个连续策略$\boldsymbol{\mu}_\theta$，梯度可以写为：

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{X,a\sim D}\left[\nabla_{\theta_i} \quad \boldsymbol{\mu}_i(a_i|o_i) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\boldsymbol{X}, a_1, \ldots, a_N)|a_i = \boldsymbol{\mu}_i(o_i)\right]$$

其中的经验回溯缓存 D 包含元组$(\boldsymbol{X}, \boldsymbol{X}', a_1, \ldots, a_N, r_1, \ldots, r_N)$，记录着所有智能体的经验动作。全局动作值函数$Q_i^{\boldsymbol{\mu}}$可以使用下面的公式更新：

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

其中$\boldsymbol{\mu}' = \left\{\boldsymbol{\mu}_{\theta_1'}, \ldots, \boldsymbol{\mu}_{\theta_N'}\right\}$是带有延迟系数$\theta_i'$的目标策略集合。在实际应用中，带有决定性策略的全局评价工作得非常好。附录中有多智能体深度决定策略梯度 MADDPG 的算法框架。

如果我们知道了被所有智能体采取的动作，比如所有策略的变化，那么环境就是静态的了。这是因为对于任意$\boldsymbol{\pi}_i \neq \boldsymbol{\pi}'_i$都有：

$$\mathrm{P}(s'|s, a_1, \ldots, a_N, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_N) = \mathrm{P}(s'|s, a_1, \ldots, a_N) = \mathrm{P}(s'|s, a_1, \ldots, a_N, \boldsymbol{\pi}'_1, \ldots, \boldsymbol{\pi}'_N)$$

正如大部分传统的强化学习方法那样，都没有明确考虑其他智能体的动作。

需要指出的是获取其他智能体的策略和观察值并不是一个特殊的假设；如果想在仿真中展示复杂的交流行为，所有智能体的策略和观察值都应该能轻易获得。然而，也可以对其他智能体进行观测来学习策略，这样就放松了假设。接下来是更详细的细节：

4.2 其他智能体的策略推断

为了解决“必须了解多智能体策略”这个假设，每个智能体 i 都要对另一个智能体 j 的真实策略进行估计$\hat{\boldsymbol{\mu}}_{\emptyset_i^j}$($\emptyset$是近似参数；下面$\hat{\boldsymbol{\mu}}_i^j$)，这个近似策略是通过最大化智能体 j 的对数概率，并带有熵正则化：

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

其中的 H 是策略分布的熵。对于近似策略，y 可以使用下面的公式

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

其中，$\hat{\boldsymbol{\mu}}_i'^j$表示目标网络对策略$\hat{\boldsymbol{\mu}}_i^j$的近似。对于$\mathcal{L}\left(\emptyset_i^j\right)$函数可以使用完全在线的方式来优化，在更新$Q_i^{\boldsymbol{\mu}}$之前，全局 Q 函数从每个智能体 j 的回溯缓存中最新采样来更新$\emptyset_i^j$。需要注意的是，在上面的方程中，直接输入每个智能体的动作日志概率到 Q，而不是通过采样。

4.3 具有整体策略的智能体

正如之前多次重述的，多智能体强化学习由于智能体策略动态变化而导致的不稳定问题。尤其是在竞争环境下，智能体会通过拟合竞争者的行为来学习 strong 策略。但是这种学习到的策略是不靠谱的，有时会在竞争者改变策略后失效。

为了使多智能体能够学习到更加鲁棒性的策略以应对竞争者的策略变化，提出训练一组 K 个不同的子策略。在每一个时间周期内，每一个智能体随机挑选一个子策略执行。假设策略$\boldsymbol{\mu}_i$是一个 K 个不同子策略的集合，k 表示$\boldsymbol{\mu}_{\theta_i^{(k)}}$，对于智能体 i，最大化整体目标：

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

由于不同的周期内执行不同的子策略，所以需要对智能体 i 的每一个子策略维护一个缓存D_i^k。于是，可以得出带有$\boldsymbol{\mu}_{\theta_i^{(k)}}$也即是$\theta_i^{(k)}$的全局目标的梯度：

$$\nabla_{\theta_i^{(k)}} J_e(\boldsymbol{\mu}_i) = \frac{1}{K}\mathbb{E}_{X,a\sim D_i^k}\left[\nabla_{\theta_i^{(k)}} \boldsymbol{\mu}_i^{(k)}(a_i|o_i) \nabla_{a_i} Q^{\boldsymbol{\mu}_i}(X, a_1, \ldots, a_N)|_{a_i=\boldsymbol{\mu}_i^{(k)}(o_i)}\right]$$

下面是主要的实验部分。

5 实验

5.1 环境

N 个智能体和 L 个地标在一个二维空间内，空间连续，时间离散。智能体可以执行动作并广播给其他智能体。这里没有假设所有智能体执行相同的动作、观察、策略。设定协作（所有的智能体必须最大化同一个共享的奖励值）与竞争（智能体之间会有冲突）环境。有些环境需要智能体之间明确的交流以最大化奖励，而不参与交互的智能体只能执行动作。下面是更详细的细节：

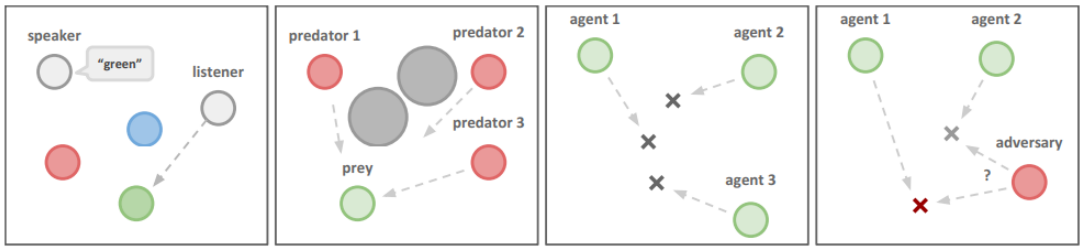


图 2.实验环境描述和具体任务，第一个是协作环境，第二个是捕食模型，第三个是协作导航，第四个是动作欺骗

第一个协作通信：这个任务包含两个智能体，一个发布者--指挥，一个接收者--听从，被放置在一个具有三种颜色地标的环境里。在每一个时间周期内，接收者必须导航到一个指定颜色地标位置，并且获得奖励，这个奖励基于智能体与正确地标的距离。然而，当接收者能够观察到地标颜色和相对位置的时候，它并不知道要去哪个颜色地标。与接受者的设定相反，发布者能观察到正确地标的颜色，并在每一个时间步内向外广播让接收者观察到这个信号。因此，发布者必须学习基于接收者的动作来输出地标颜色。这个问题看起来简单，但是传统的强化学习方法并不能解决的很好。

第二个协作导航：在这个环境下，智能体必须相互协作才能到达一个带有 L 个地标的集合。一个智能体需要观察其他智能体的相关动作和地标，并根据其接近每个地标的程度获得相应的奖励。换句话说，就是智能体必须能够覆盖所有的地标。此外，智能体是有空间体积的，如果彼此发生碰撞，那么将会受到惩罚。最后智能体学会推断出他们必须覆盖的地标，并且在移动的过程中避免碰撞。

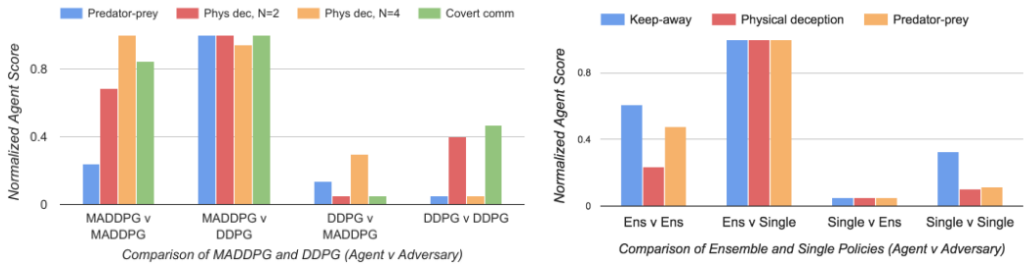
保持距离：这个场景包含 L 个地标和一个目标点地标，N 个协作智能体知道目标点地标是哪个，并根据他们距离目标点地标的远近确定奖励，同时存在 M 个对抗智能体，这些对抗智能体必须阻止其他协作智能体到达目标点。所以，这些对抗者会将智能体推离地标，然后自己临时占领。同时对抗者也能获得相应奖励，奖励是基于他们距离目标点地标的远近。但是，对抗者并不知道正确的地标颜色，所以这就需要从对方智能体的动作中做出推断。

动作欺骗：N 个智能体协作从 N 个地标中选取一个抵达。其奖励是基于任何一个智能体到目标点的最小距离（所以只需要有一个智能体到达目标即可）。然而，也有其他的对手想要抵达目标点，关键的是对手并不知道哪一个是正确目标点。因此，参与协作的机器人将学会如何分散并占领整个环境中的地标来欺骗对手，并根据对手与目标点地标的距离大小受到惩罚。

捕食者模型：在很多经典的 Predator-prey 捕食者模型中，N 个较慢的协作机器人必须追逐较快的猎物，这些猎物在 L 个地标周围随机生成。每当追逐者与猎物碰撞一次，追逐者受到奖励，猎物受到惩罚。智能体观察的范围包括其他智能体的位置和速度，以及地标的位置。

隐藏通信：这是一个对抗的通信环境，发布者（Alice）必须向接收者（Bob）发送消息，接收者必须重建 reconstruct 该消息。然而，对抗者（Eve）也在观察着信道，想要窃取消息并重建出来。如果被成功窃取，Alice 和 Bob 都将会受到惩罚，所以 Alice 必须使用一

个随机的 key 加密她的消息，这个 key 只有她跟 Bob 两个人知道。这似乎更像是密码学的研究范围。

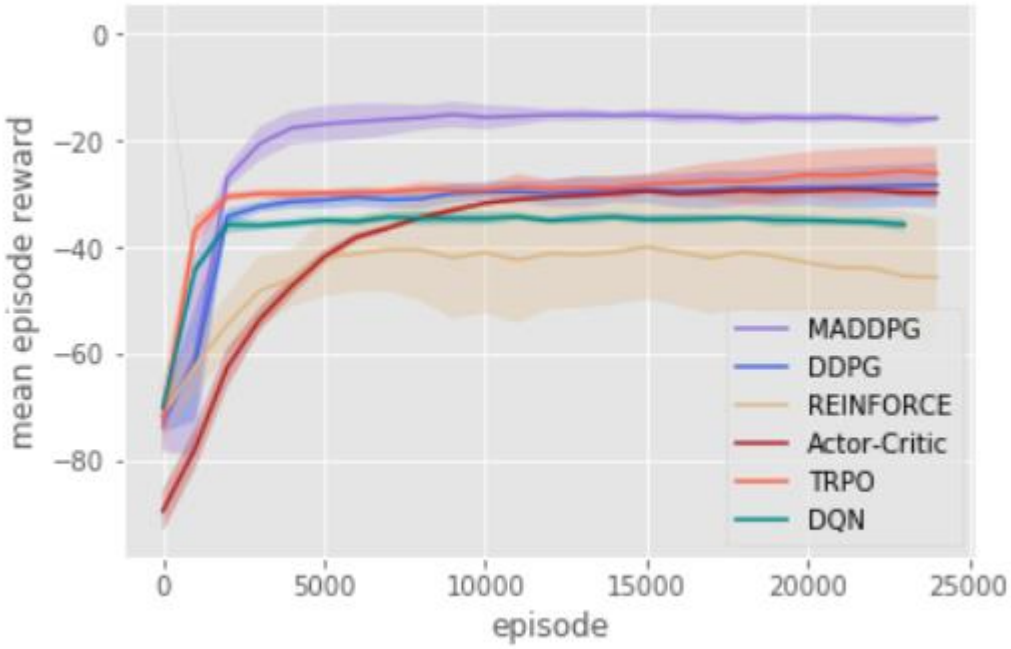


图三. MADDPG（右）与 DDPG（左）对比，MADDPG 使用的是整体策略信息，DDPG 使用的是单个策略的信息。每一簇柱状图都显示了一组竞争策略的得分（分数被正则化在 0-1 区间内），分数越高代表效果越好。在所有的情况下，MADDPG 都比 DDPG 效果好。

5.2 与去中心化的强化学习方法比较

在 5.1 的场景下，实现了 MADDPG(multi-agent deep deterministic policy gradient)，除非特殊说明，都是两层的 ReLU 的多层感知机 MLP(Multi-layer Perceptron)，每层有 64 个单元。为了支持离散通信消息，使用了类别变分自编码器(Gumbel-Softmax)评价器。为了评估在竞争环境下学习到的策略的质量，可以比较 MADDPG 智能体与 DDPG 智能体的实验结果。然后，训练模型直到收敛，并在额外的 1000 个训练周期内用多项指标的平均值来评估模型质量。

第一个实验是协作通信，尽管任务简单（发布者只要简单地学习输出观察值），传统的像 DQN、Actor-Critic、一阶 TRP、和 DDPG 等强化学习方法都失败了，不能学习到正确的行为（通过测量接收者是否靠近目标点地标）。实际上，接收者学会了忽略发布者的消息，只是简单地移动到了所有观察的地标中间。如图四所示，几种方法的对比。



图四. 协作通信场景下，25000 步，智能体的奖励曲线

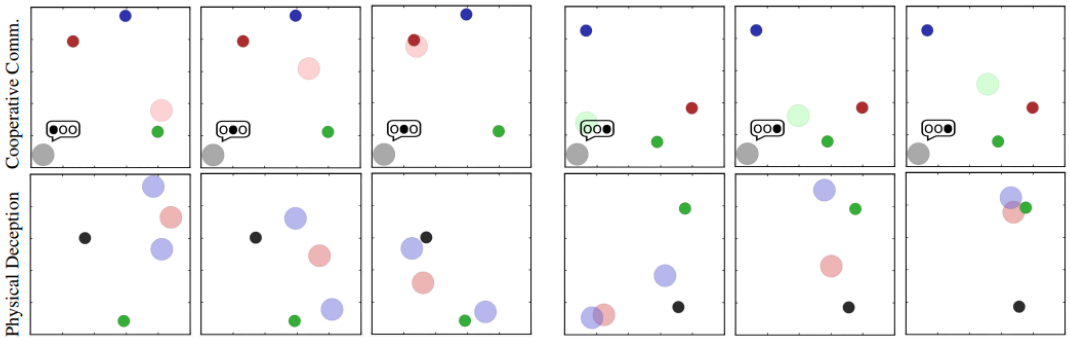
文中推测传统方法失效的主要原因是缺少一个恒定的梯度信号。比如，如果发布者输出

正确而接收者移动的方向错了，那么发布者受到惩罚。这个问题随着时间增长将逐渐恶化：当每一个时间步内，接收者只是简易地重构发布者的观察，或者智能体和地标的初始位置是指定的甚至只在均匀分布的情况下，传统的策略梯度方法才能够学习的。这意味着很多之前的短时场景下多智能体方法不适用复杂任务。

相反，MADDPG 智能体通过全局评价能够更容易地学到协作行为。在协作通信的时候，MADDPG 能够可靠地学习到正确的发布者和接收者策略，在 84%的时间内，接收者可以正确完成导航任务。

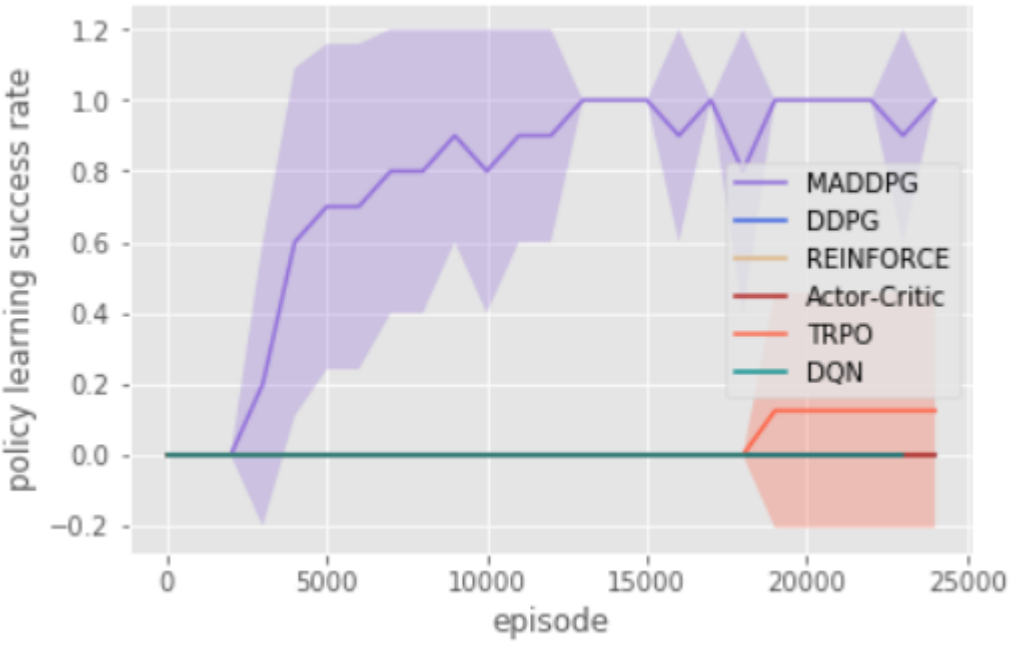
PD 任务的结果是：当协作智能体被 MADDPG 方法训练的时候，在 94%的是时间内，通过覆盖所有的地标成功地欺骗到了对手，其地标数目=2（图五所示）。此外，对手占领地标的概率很低，特别是使用 DDPG 方法的时候只有 16.4%。更明显的是，DDPG 训练的智能体欺骗不了使用 MADDPG 的对手，甚至在地标数为 4 的时候也无法欺骗使用 DDPG 的智能体了。

协作导航和捕食任务有点类似，同样地，MADDPG 比 DDPG 表现好。协作导航的时候，MADDPG 智能体到每一个地标有更小的平均距离，每个周期内的平均碰撞次数也只有 DDPG 的一半。同时，MADDPG 捕食者更容易追上 DDPG 的猎物（每个周期 16.1 次碰撞），而 DDPG 追逐 MADDPG 的时候只有每个周期内 10.3 次碰撞（碰撞越多表示被抓到的次数越多）。



图五. 在协作通信 CC 和动作欺骗 PD 场景下比较 MADDPG 和 DDPG 方法，时间 t=0,5,25。黑色圆点表示地标。在协作通信场景下，灰色的智能体是发布者，接收者的颜色表示目标点地标。在动作欺骗 PD 场景下，蓝色智能体试图欺骗红色的对手，同时努力覆盖绿色地标（目标点）。MADDPG 方法在两个场景下都学习到了正确的行为：CC 场景下，发布者学会了直接向接收者输出目标点地标的颜色；PD 场景下，多个智能体都去覆盖地标以迷惑对手。DDPG 或者其他的强化学习方法在这两个场景下都不适用：在协作通信的时候，发布者总是重复相同的表达，接收者移动到地标中间；在 PD 场景下，一个智能体总是朝着目标移动，而且被对手跟随着，而另一个智能体总是徘徊不定。

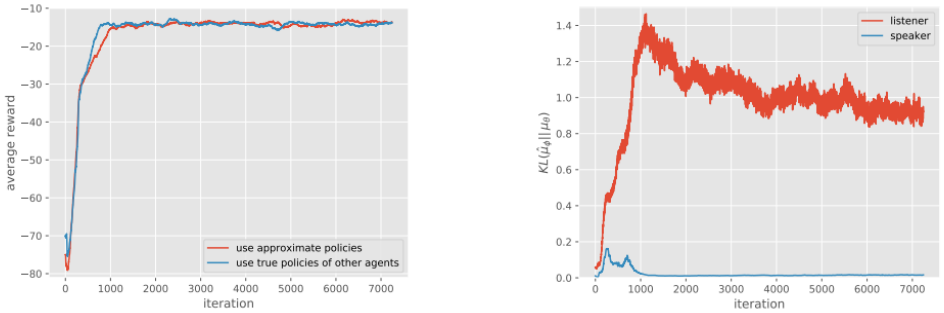
在隐藏通信环境（*或者说加密通信*）下，MADDPG 和 DDPG 训练后的 Bob 比 Eve 在重构 Alice 消息时表现好。然而，MADDPG 训练后的 Bob 还是比 DDPG 好，相对成功率为 52.4%比 25.1%。此外，MADDPG 训练后的 Alice 能够加密她的消息，以至于 Eve 只有随机猜测般的准确率。附录中的学习曲线显示了由环境中竞争因素引起的振荡也是一个亟待解决的问题。需要强调的是，这里没有使用任何密码学的技巧，包括修改 Eve 的损失函数，交替训练智能体和对手，或者使用混合“mix & transform”反馈和卷积架构。



图六. 25000 轮训练之后的协作环境下策略学习成功率

5.3 其他智能体学习策略的效果

在协作通信环境下评估了其他智能体学习策略的效果，超参数不变 λ=0.001。结果如图七，尽管两条曲线没有完全重合（有些地方的差异还蛮大的），使用近似策略进行学习也能达到和真实策略同样的成功率，也没有明显发散。



图七. 通过近似其他智能体的策略在协作环境下学习的效率。左边的图是迭代次数与奖励的曲线，可以看出，MADDPG 智能体很快地学会了使用近似其他智能体策略的方法解决任务；右边是真实策略与近似的策略之间的 KL 分歧(*相对熵，又称 KL 散度(Kullback－Leibler divergence)，是描述两个概率分布 P 和 Q 差异的一种方法*)。

5.4 使用全局策略的训练效果

重点考察竞争环境下使用全部策略的效果，包括“保持距离”（keep-away）、协作导航、捕食者等场景。“保持距离”场景选择 K=3 子策略，协作导航环境和为捕食者选择 K=2。为了提高收敛速度，强行让协作智能体和竞争智能体在每个周期内都具有相同的策略。为了评估这样做的效果，测试全部策略和单个策略在智能体竞争中的表现。结果如图三中的右侧图。可以看出，使用全部策略的智能体表现更好。尤其是第二个柱状图左侧比第三个柱状图高出

了一大截。

6.总结和接下来的方向

文章提出了一个多智能体策略的梯度算法，该算法能基于所有智能体的观察和动作学习一个全局评价方法。直观地看，这个方法比传统强化学习方法在多个竞争协作环境下都要好使。接下来，采用全部的策略来训练智能体可以提高性能，这个方法可适用于任何一个多智能体算法。

文中的方法一个缺陷是 Q 的输入空间随着智能体个数 N 呈线性增长（取决于 X 包含的信息）。实际中，可以通过采用标准的 Q 函数，并只考虑给定智能体附近的智能体。

2018.4.21

下面是一些常用 RL 方法的总结。

DQN *是一个基于价值 value 的方法。换句话说就是通过计算每一个状态动作的价值，然后选择价值最大的动作执行。这是一种间接的做法。*

Q-learning *是一种基于值函数估计的强化学习方法，Policy Gradient 是一种策略搜索强化学习方法。Q-learning 在离散状态空间中理论上可以收敛到最优策略，但收敛速度可能极慢。在使用函数逼近后（例如使用神经网络策略模型）则不一定。Policy Gradient 由于使用梯度方法求解非凸目标，只能收敛到不动点，不能证明收敛到最优策略。*

Policy gradient 策略梯度算法*，接受环境信息 (observation),输出不是 action 的 value, 而是具体的那一个 action。Policy gradient 最大的一个优势是: 输出的这个 action 可以是一个连续的值, 之前我们说到的 value-based 方法输出的都是不连续的值, 然后再选择值最大的 action. 而 policy gradient 可以在一个连续分布上选取 action。*

什么是策略网络 Policy Network？就是一个神经网络，输入是状态，输出直接就是动作（不是 Q 值）。要更新策略网络，或者说要使用梯度下降的方法来更新网络，我们需要有一个目标函数。对于策略网络，目标函数其实是比较容易给定的，也就是

$$L(\theta) = \mathbb{E}\left(\gamma_1 + r\gamma_2 + r^2\gamma_3 + \cdots|\pi(,\theta)\right)$$

所有带衰减 reward 的累加期望，问题就在于如何利用这个目标来更新参数θ呢？就是如何能够计算出损失函数关于参数的梯度（也就是策略梯度）$\nabla_\theta L(\theta)$

*一个 Policy Network，没有 loss，怎么更新？从概率的角度来思考问题。我们有一个策略网络，输入状态，输出动作的概率。**如果某一个动作得到 reward 多，那么我们就使其出现的概率增大，如果某一个动作得到的 reward 少，那么我们就使其出现的概率减小。**一个 reward，result 都依赖于大量的动作才导致的，reward 来评判动作的好坏是不准确的，甚至用 result 来评判也是不准确的。**如果能够构造一个好的动作评判指标，来判断一个动作的好与坏，那么我们就可以通过改变动作的出现概率来优化策略！**假设这个评价指标是*

$$f(s,a)$$

那么我们的 Policy Network 输出的是概率。更常使用 log likelihood:$\log(a|s,\theta)$，可以构造一个损失函数如下：$L(\theta)\sum log\pi(a|s,\theta)f(s,a)$

f(s,a)不仅仅可以作为动作的评价指标，还可以作为目标函数。因此，我们可以利用评价指标 f(s,a)来优化 Policy，同时也是在优化的同时优化了 f(s,a).那么问题就变成对 f(s,a)求关于参数的梯度。

Actor-Critic

Actor-Critic 是结合 Policy 和 Value Function 的产物

在 RL 的基本设置当中，有 agent，environment, action, state, reward 等基本元素。agent 会与 environment 进行互动，而产生轨迹，通过执行动作 action，使得 environment 发生状态的变化，s -> s'；然后 environment 会给 agent 当前 动作选择以 reward（positive or negative）。通过不断的进行这种交互，

使得积累越来越多的 experience，然后更新 policy，构成这个封闭的循环。

在 RL 任务中，我们本质上最终要学习的是策略（Policy）。前者用的是间接方法，即通过学习值函数（value function）或者动作值函数（action-value function）来得到 policy。而后者是直接对 policy 进行建模和学习，因此后者也称为 policy optimization。Policy-based 方法又可分为两大类：gradient-based 方法和 gradient-free 方法。前者也称为 policy gradient（PG）方法。而 policy gradient 方法又可细分为几类，如 finite difference，Monte-Carlo 和 Actor-Critic 等。Actor-Critic（AC）方法其实是 policy-based 和 value-based 方法的结合。因为它本身是一种 PG 方法，同时又结合了 value estimation 方法，所以有些地方将之归为 PG 方法的一种，有些地方把它列为 policy-based 和 value-based 以外的另一种方法，都好理解。在 AC 框架中，actor 负责 policy gradient 学习策略，而 critic 负责 policy evaluation 估计 value function。可以看到，一方面 actor 学习策略，而策略更新依赖 critic 估计的 value function；另一方面 critic 估计 value function，而 value function 又是策略的函数。Policy 和 value function 互为依赖，相互影响，因此需要在训练过程中迭代优化。这种多元优化的迭代思想其实在机器学习中有很多体现。

有了 critic 和 actor 的概念，再看当时其它的方法，就可以分为 critic-only 方法和 actor-only 方法两类。前者基于 value estimation。它广泛应用于各种领域，但有一些缺点使它的应用受到局限。如 1） 难以应用到随机型策略（stochastic policy）和连续的动作空间。2） value function 的微小变化会引起策略变化巨大，从而使训练无法收敛。尤其是引入函数近似（function approximation，FA）后，虽然算法泛化能力提高了，但也引入了 bias，从而使得训练的收敛性更加难以保证。而后者通过将策略参数化，从而直接学习策略。这样做的好处是与前者相比拥有更好的收敛性，以及适用于高维连续动作空间及 stochastic policy。但缺点包括梯度估计 variance 比较高，且容易收敛到非最优解。另外因为每次梯度的估计不依赖以往的估计，意味着无法充分利用老的信息。

***actor**：优化这个 policy，使得其表现的越来越好；*

***critic**：尝试估计 value function，使其更加准确；*

异步优势 actor-critic 算法（Asynchronous advantage actor-critic，即：A3C）。