计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(1):145−151 [doi: 10.15888/j.cnki.csa.008237]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于值分布的多智能体分布式深度强化学习算法①

陈妙云, 王　雷, 盛　捷

(中国科学技术大学 信息科学与技术学院, 合肥 230027)
通信作者: 王　雷, E-mail: wangl@ustc.edu.cn

摘　要: 近年来深度强化学习在一系列顺序决策问题中取得了巨大的成功, 使其为复杂高维的多智能体系统提供有效优化的决策策略成为可能. 然而在复杂的多智能体场景中, 现有的多智能体深度强化学习算法不仅收敛速度慢, 而且算法的稳定性无法保证. 本文提出了基于值分布的多智能体分布式深度确定性策略梯度算法 (multi-agent distributed distributional deep deterministic policy gradient, MA-D4PG), 将值分布的思想引入到多智能体场景中, 保留预期回报完整的分布信息, 使智能体能够获得更加稳定有效的学习信号; 引入多步回报, 提高算法的稳定性; 引入了分布式数据生成框架将经验数据生成和网络更新解耦, 从而可以充分利用计算资源, 加快算法的收敛. 实验证明, 本文提出的算法在多个连续/离散控制的多智能体场景中均具有更好的稳定性和收敛速度, 并且智能体的决策能力也得到了明显的增强.

关键词: 多智能体; 深度强化学习; 值分布; 多步回报; 分布式数据生成

引用格式: 陈妙云,王雷,盛捷.基于值分布的多智能体分布式深度强化学习算法.计算机系统应用,2022,31(1):145–151. http://www.c-s-a.org.cn/1003-3254/8237.html

Multi-agent Distributed Deep Reinforcement Learning Algorithm Based on Value Distribution

CHEN Miao-Yun, WANG Lei, SHENG Jie

(School of Information Science and Technology, University of Science and Technology of China, Hefei 230027, China)

Abstract: In recent years, deep reinforcement learning has achieved great success in many sequential decision-making problems, which makes it possible to provide effective and optimized decision-making strategies for complex and high-dimensional multi-agent systems. However, in complex multi-agent scenarios, the existing multi-agent deep reinforcement learning algorithm has a low continuous convergence speed, and the stability of the algorithm cannot be guaranteed. Herein, we propose a new multi-agent deep reinforcement learning algorithm, which is called multi-agent distributed distributional deep deterministic policy gradient (MA-D4PG). We adapt the idea of value distribution to multi-agent scenarios and retain the complete distribution information of expected return, so that agents can obtain a more stable and effective learning signal. We also introduce a multi-step return to improve the stability of the algorithm. In addition, we use a distributed data generation framework to decouple empirical data generation and network update for the purpose of taking full advantage of computing resources to speed up the convergence. Experiments show that the proposed method has better stability and a higher convergence speed in multiple continuous/discrete controlled multi-agent scenarios and the decision-making ability of agents has also been significantly enhanced.

Key words: multi-agent; deep reinforcement learning; value distribution; multi-step return; distributed data generation

① 基金项目: 中国科学技术大学预研基金 (YZ2101900004)
收稿时间: 2021-03-11; 修改时间: 2021-04-07, 2021-04-16; 采用时间: 2021-04-20; csa 在线出版时间: 2021-12-17

1 引言

近年来, 强化学习在解决一系列顺序决策问题上取得了巨大的成功. Minih 等人[1] 将深度学习和强化学习结合起来, 提出了深度强化学习, 在众多领域都得到了广泛的应用. 比如机器人控制[2], 生命科学[3], 自动驾驶[4], 5G 通信[5], 物联网[6], 车联网[7] 等. 在深度强化学习的基础上众多研究者提出了多项改进来提升算法的收敛速度和稳定性. 然而这些算法的研究和应用大多集中在单智能体领域. 现实世界中的任务和场景通常更加复杂, 涉及到多个智能体之间的合作和竞争. 比如工厂中的协作机器人, 交通控制系统和自动军事系统等. 将单智能体强化学习算法简单地移植到多智能体场景中通常会带来一些问题和挑战[8]. 比如单智能体深度强化学习算法无法表征智能体之间的合作与竞争关系以及智能体之间共同进化导致环境非平稳性的问题. 并且多智能体与复杂场景会带来计算量大幅提升, 算法收敛速度明显下降等挑战.

目前, Self-play[9] 被证明是一种有效的多智能体训练范式, 并成功的应用在了围棋, 象棋以及博弈竞争场景中. Palmer 等人[10] 提出了一种基于宽容的 DQN 算法, 对同伴不合理的行为给予一定的容忍度. Foerster 等人[11] 提出了反事实的思想, 解决了多智能体合作场景中信用分配的问题. 这些算法在多智能体场景中都取得了很好的效果. 但是只能应用在离散动作空间中, 并且 Self-play 还增加了智能体之间动作集完全一致的限制. Lowe 等人[12] 提出了一种集中式训练, 分散式执行的算法 (multi-agent deep deterministic policy gradient, MA-DDPG), 解决了多智能体场景中环境非平稳的问题. 并且该算法可以同时应用在连续和离散控制的多智能体场景中, 使得应用场景更加灵活.

然而对于复杂的多智能体场景, 目前的多智能体深度强化学习算法还有很多问题没有解决. 智能体动作策略的更新质量非常依赖于预期回报的估计, 但是多个智能体与环境交互时, 其内在的随机性会导致预期回报具有不确定性, 现有的方法将预期回报简单地平均为期望值则会丢失完整的分布信息, 导致智能体不能进行有效的学习. 并且现有的方法主要基于单步回报, 智能体只考虑当下的收益而忽略未来的信息, 从而引发算法不稳定. 此外现有方法中经验数据生成往往和网络更新耦合在一起, 导致只能串行处理, 难以有效利用更多的计算资源. 为此, 本文提出了基于值分布的多智能体分布式深度确定性策略梯度算法 (multi-agent distributed distributional deep deterministic policy gradient, MA-D4PG), 引入值分布的思想[13] 对预期回报的分布建模, 避免分布信息丢失, 使得智能体能够获得更加准确的梯度更新, 保证学习的有效性; 引入多步回报使得智能体能够从未来的经验中学习, 增加智能体的预见能力, 提高算法的稳定性; 引入分布式数据生成框架 (APE-X)[14], 将智能体与环境交互生成经验数据部分与网络更新解耦, 使得经验数据生成过程和网络更新能够并行进行, 充分利用更多的计算资源.

在多个不同的连续/离散控制的多智能体场景中本文算法的稳定性和收敛速度都有了极大的提升, 智能体的决策能力也有了明显的增强. 最后本文还做了消融实验展示每个改进各自的贡献以及相互作用.

2 相关工作

2.1 多智能体马尔可夫决策过程

多智能体马尔可夫决策过程是马尔可夫决策过程在多智能体场景下的扩展. 其同样由四元组构成:

$$M = (S, A, Pr(S'|S, A), R) \tag{1}$$

其中, S表示所有智能体的状态集合$\{s_1, s_2, \cdots, s_k\}$, k表示智能体的数量. A表示所有智能体的动作集合$\{A_1, A_2, \cdots, A_k\}$. $Pr(S'|S, A)$表示状态转移函数, $S \times A_1 \times \cdots \times A_k \to P(S')$. 其中$P(S')$定义了所有智能体的下一个状态的概率分布. R表示所有智能体的回报集合$\{r_1, \cdots, r_k\}$. 本文遵循以上符号表示.

对于每一个智能体i, 学习相应的动作选择策略π_i: $o_i \to P(A_i)$. 其中o_i表示智能体i的局部观测, $P(A_i)$表示选择动作的概率分布.

智能体的目标是学习一个动作选择策略以最大化预期回报如下, 其中γ为折现因子.

$$J_i(\pi_i) = E_{A_1 \sim \pi_1, \cdots, A_k \sim \pi_k}\left[\sum_{t=0}^{\infty} \gamma^t r_{i,t}(S_t, A_{1,t}, \cdots, A_{k,t})\right] \tag{2}$$

2.2 多智能体深度确定性策略梯度

深度策略梯度算法 (deep policy gradient, DPG)[15] 是一种非常经典的深度强化学习算法. 该算法使用神经网络模拟动作策略函数, 通过梯度上升法调整网络参数以最大化目标函数, 如式 (3) 所示:

$$J(\theta) = Q(S_t, A_t) \tag{3}$$

其中, $Q(S_t, A_t)$表示状态动作对的预期回报的期望值. 相应的动作策略网络的梯度公式如式 (4) 所示:

$$\nabla J(\theta)=E_{s_t,A_t}[\nabla_\theta \log\pi(A|s_t)Q(s_t,A_t)] \tag{4}$$

其中, π表示智能体的随机动作策略, 输出所有动作的概率分布.

DDPG (deep deterministic policy gradient)[16] 是 DPG 的一种变体, 其中动作策略输出的是确定性动作而不是所有动作的概率分布. 可以应用于连续控制的多智能体场景中, 相应的动作策略网络梯度公式如式 (5):

$$\nabla J(\theta)=E_{s_t,A_t}[\nabla_\theta \mu(A_t|s_t,\theta)\nabla_{A_t}Q(s_t,A_t)] \tag{5}$$

其中, μ表示智能体的确定性动作策略.

而多智能体深度确定性策略梯度算法是 DDPG 在多智能体场景的扩展, 并且与 actor-critic框架相结合. 其中 actor 模拟动作选择策略函数μ, 根据智能体i的局部观测o_i, 输出动作$A_i=\mu_i(o_i|\theta_i)$. critic 模拟动作价值函数Q_i以输出智能体i某一时刻的状态动作对的预期回报的期望值. Q_i是一个集中式的函数, 考虑了所有智能体的状态动作信息, 解决了多智能体场景中智能体共同学习以及相互作用导致的环境非平稳.

在 MA-DDPG 算法智能体i的 actor 网络的梯度公式以及 critic 网络的损失函数分别如式 (6) 和式 (7):

$$\nabla J_i(\theta_i)=E_{S,A}[\nabla_{\theta_i}\mu_i(A_i|s_i,\theta_i)\nabla_{A_i}Q_i(S,A_1,\cdots,A_k)] \tag{6}$$

$$\begin{cases} L_i(w_i)=\mathrm{E}_{S,A,r,S'}[(Q_i^{\pi}(S,A_1,\cdots,A_k|w_i)-y)^2] \\ y=r_i+\gamma\max\limits_{A'}\bar{Q}_i^{\pi}(S',A_1',\cdots,A_k'|\bar{w}_i) \end{cases} \tag{7}$$

3 基于值分布的多智能体分布式深度确定性策略梯度算法

本文引入了值分布的思想对预期回报的分布建模, 提出了一种基于值分布的评论家 (critic) 更新方法; 并在此基础上引入多步回报衡量 critic 的损失; 最后引入了分布式数据生成架构 (APE-X) 将智能体与环境交互生成经验数据的部分与网络更新解耦. 这些将会在本节进行详细描述.

3.1 基于值分布的 critic 更新

基于值的多智能体深度强化学习算法直接对智能体状态动作对的预期回报的期望建模, 不考虑智能体与环境的交互的内在随机性对预期回报的影响. 这种建模方式损失了关于预期回报的完整的分布信息. 本文引入值分布的思想, 强调预期回报的分布信息的重要性, 对预期回报的分布建模, 而非对其期望值建模. 本文将预期回报看作随机变量, 记为$Z_i^{\mu_i}(S_t,A_t)$. 定义如式 (8) 所示:

$$Q_i^{\mu_i}(S_t,A_t)=E_{S_t,A_t,S_{t+1}}[Z_i^{\mu_i}(S_t,A_t)] \tag{8}$$

其中, $Q_i^{\mu_i}(S_t,A_t)$表示智能体i的 critic 网络输出预期回报的期望值.

本文使用参数化的多分类分布对预期回报的分布建模{$N\in\mathbb{N}$, $V_{\min}$, $V_{\max}\in\mathbb{R}$}, 其中N表示分类个数, $V_{\min}$表示最小类别值, $V_{\max}$表示最大类别值. 以此模拟预期回报的近似分布. 其中一系列的类别值分别如式 (9):

$$z_j=V_{\min}+j\cdot\Delta z,\ 0\le j<N,\ \Delta z=\frac{V_{\max}-V_{\min}}{N-1} \tag{9}$$

在 actor-critic 的框架下, 改进的基于值分布思想的 critic 网络输出预期回报的N分类分布$Z_i^{\mu_i}$, 表示各类预期回报值的概率分布. 记$p_{i,j}(S_t,A_t)$为智能体i的 critic 网络所输出的预期回报属于第j类z_j的概率.

$$p_{i,j}(S_t,A_t)=\frac{\mathrm{e}^{w_j(S_t,A_t)}}{\sum_{n=0}^{N-1}\mathrm{e}^{w_n(S_t,A_t)}},\ 0\le j<N-1 \tag{10}$$

其中, $S_t=\{s_{1,t},\cdots,s_{k,t}\}$表示所有智能体的状态信息, $A_t=\{A_{1,t},\cdots,A_{k,t}\}$表示所有智能体的动作信息.

基于值分布的 critic 的损失函数定义如式 (11) 和式 (12):

$$L_i(w_i)=E[d((T\bar{Z}_i^{\bar{\mu}_i})(S_t,A_{1,t},\cdots,A_{k,t}),Z_i^{\mu_i}(S_t,A_{1,t},\cdots,A_{k,t}))] \tag{11}$$

$$(T\bar{Z}_i^{\bar{\mu}_i})(S_t,A_{1,t},\cdots,A_{k,t})=r_{i,t}+\gamma E[\bar{Z}_i^{\bar{\mu}_i}(S_{t+1},A_{1,t+1},\cdots,A_{k,t+1})] \tag{12}$$

其中, $Z_i^{\mu_i}$表示智能体i的 critic 网络输出的预期回报的近似分布 (Z分布), $\bar{Z}_i^{\bar{\mu}_i}$表示智能体i的目标 critic 网络的输出, $(T\bar{Z}_{i,t}^{\bar{\mu}_{i,t}})$表示值分布的贝尔曼算子. 式 (12) 将目标 critic 网络的输出映射成目标Z分布, d表示预期Z分布和目标Z分布之间距离的度量指标, 本文采用交叉熵损失进行度量.

以上所述引入了值分布的思想对 critic 的更新进行改进, 相应的 actor 更新也引入预期回报的分布信息, 本文通过引入Z分布的期望来实现, 则 actor 网络更新的梯度公式如式 (13) 所示:

$$\begin{cases} \nabla J_i(\theta_i)=E_{S_t,A_t}[\nabla_{\theta_i}\mu_i(A_{i,t}|o_{i,t},\theta_i)E[\nabla_{A_{i,t}}Z_i(S_t,A_{1,t},\cdots,A_{k,t})]] \\ A_{i,t}=\mu_i(o_{i,t}|\theta_i) \end{cases} \tag{13}$$

3.2 多步回报

本小节在第 3.1 节所述的基于值分布的 critic 更新

的基础上引入多步回报来计算 critic 的 TD (temporal-difference) 误差. 利用多步回报可以使得智能体从未来的经验中学习, 而不是只考虑当下的收益. 多步回报的思想已经广泛的应用在众多单智能体强化学习算法[17,18] 中. 类似的修正也可以应用在多智能体深度强化学习算法中. 基于多步回报的贝尔曼方程定义如式 (14) 和式 (15) 所示:

$$(T\bar{Z}_i^{\bar{\mu}_i})(S_t,A_{1,t},\cdots,A_{k,t}) = R_{i,t}+\gamma E[\bar{Z}_i^{\bar{\mu}_i}(S_{t+N},A_{1,t+N},\cdots,A_{k,t+N})] \tag{14}$$

$$R_{i,t}=\sum_{t'=t}^{t+N-1}\gamma^{t'-t}r_{i,t'} \tag{15}$$

其中, $R_{i,t}$表示时刻t智能体i的N步回报.

3.3 分布式数据生成和网络更新

针对模型耦合度高无法充分利用计算资源的问题, 本文引入了分布式数据生成的框架 (APE-X) 将多智能体场景中智能体与环境交互生成经验数据和网络更新两部分进行解耦. 负责数据生成的部分称为 Actor, 负责从经验回放池中采样数据进行网络更新的部分称为 Learner. Actor 的网络结构与 Learner 完全一致, 只负责与环境交互得到经验数据存入经验回放池中, 不执行梯度计算进行参数更新, 而是定期复制 Learner 的网络参数. Learner 只负责从经验回放池中采样数据进行网络更新而不负责与环境进行交互. 本文算法并行运行多个独立的 Actor, 分别与环境进行交互得到经验数据 $\{S_{t+1-N},A_{t+1-N},R_{t+1-N},S_{t+1}\}$, 写入同一个经验缓冲池中. Learner 从该经验缓冲池中随机采样 M 批经验数据进行网络更新. 具体的算法框架图如图 1 所示. 其中 $S_{t+1-N}=\{s_{1,t+1-N},\cdots,s_{k,t+1-N}\}$, $A_{t+1-N}=\{A_{1,t+1-N},\cdots,A_{k,t+1-N}\}$, $R_{t+1-N}=\{R_{1,t+1-N},\cdots,R_{k,t+1-N}\}$.

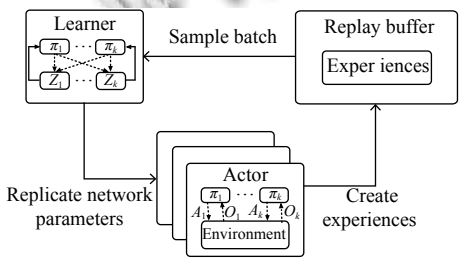


图 1 分布式数据生成框架示意图

3.4 本文算法小结

本文算法的伪代码如算法 1 和算法 2 所示. Learner 中 actor-critic 的网络都采用梯度下降法进行网络参数更新. actor 和 critic 的学习率分别是α_t, β_t, 均采用 Adam 优化器[19], 利用动量和自适应学习率来加快收敛速度. 该算法伪代码中同时包含了 Actor 和 Learner 的伪代码, 多个 Actor 初始化为相同的多智能体环境, 并行与环境交互生成经验数据并将其存入经验回放池中.

算法 1. 经验数据生成-Actor

初始化N值, 智能体数量 K, 经验回放池 B, 初始化软更新频率 TAU
初始化 Actor 网络参数$(\theta_{\text{Actor}}^{1:k},\omega_{\text{Actor}}^{1:k})\leftarrow(\theta_{\text{Learner}}^{1:k},\omega_{\text{Learner}}^{1:k})$
初始化 States, Actions, Rewards, Terminals 为大小为N+1的 deque
For j=1,⋯,$Episode\_num$ **do**
  Clear (States, Actions, Rewards, Terminals)
  $S_0\leftarrow$Env.Reset()
  States.Add(S_0)
  For t=1,⋯,max_step **do**
    $A_{1,t-1},\cdots,A_{k,t-1}\leftarrow\mu_1(s_{1,t-1}),\cdots,\mu_k(s_{k,t-1})$
    $r_{1,t-1},\cdots,r_{k,t-1},S_t,\text{term}_t\leftarrow\text{Env.Step}(A_{t-1}+OU)$
    States.Add(S_t)
    Actions.Add(A_{t-1})
    Rewards.Add($\{r_{1,t-1},\cdots,r_{k,t-1}\}$)
    Terminals.Add(term_t)
    If States.full **then**
      根据式 (15) 计算N步回报R_{t-N}
      B.Add(States[0],Actions[0], R_{t-N} States[N], Terminals[N–1])
      PopLeft(States, Actions, Rewards, Terminals)
    End If
    If term_t **break**
  End For
End For

算法 2. 网络更新-Learner

初始化采样批次 M, 最大更新步数 T
随机初始化网络参数$(\theta_{\text{Learner}}^{1:k},\omega_{\text{Learner}}^{1:k})$
初始化目标网络参数$(\theta_{\text{target}}^{1:k},\omega_{\text{target}}^{1:k})\leftarrow(\theta_{\text{Learner}}^{1:k},\omega_{\text{Learner}}^{1:k})$
For t=1,⋯,T **do**
  If len(B) <M **break**
  从 B 中随机采样 M 批数据
  For i=1,⋯,k **do**
    根据式 (14) 和式 (11) 计算 critic 的损失, 使用梯度下降法进行反向传播更新 critic 网络参数
    根据式 (13) 计算 actor 的梯度, 其中$A_{i,t}=\mu_i(s_{i,t}|\theta_i)$, 更新 actor 网络参数
  End For
  $(\theta_{\text{target}}^{1:k},\omega_{\text{target}}^{1:k})=(1-TAU)\cdot(\theta_{\text{target}}^{1:k},\omega_{\text{target}}^{1:k})+TAU\cdot(\theta_{\text{Learner}}^{1:k},\omega_{\text{Learner}}^{1:k})$
  If t mod actor_update_freq==0 **then**
    Replicate Network Weights to Actors
  End If
End For

4 实验分析

本文设置了 3 个实验场景来测试本文算法 (MA-D4PG) 的性能, 将在本节依次描述. 在实验中设置了一个额外的 Actor. 该 Actor 只利用, 而不进行探索, 方便在没有噪声的情况下测试本文算法的性能. 其他 Actor 则负责探索环境, 获得更多关于环境的信息, 并且与环境交互生成经验数据. Learner 每更新 N 步, 将网络参数复制给 Actor. 本文将 N 值设为 5. 为了加快训练模型以及更好的利用资源, Learner 模型在 GPU 上进行运算, Actor 在 CPU 进行运算. 强化学习算法性能评价的两个关键指标是算法收敛所需的步数以及最终收敛的 Episode Reward 值. 收敛所需的步数越少, 表示算法的收敛速度越快, 最终的收敛的 Episode Reward 值越大, 表示智能体的学习效果越好. 本文将根据这两个指标对本文算法进行评价. 并且本文还做了一些消融实验, 分别删除本文算法中的单项组件, 以确定其特定的贡献.

本文实验中统一使用大小为1×10^6经验重放池存储智能体与环境交互产生的经验数据. 为确保一定的探索性, 在智能体的动作策略中添加 Ornstein-Uhlenbeck 噪声[19]. 并且本文实验中 actor 和 critic 的学习率以及采样批次大小, 软更新频率统一设置如下:

$$\alpha=5\times10^{-5},\beta=5\times10^{-5},M=256,TAU=0.001$$

4.1 实验环境配置

本文算法中实验环境的配置如表 1 所示. 本文采用了 PyTorch 作为深度学习框架, 在 GPU 上使 CUDNN 加速训练模型.

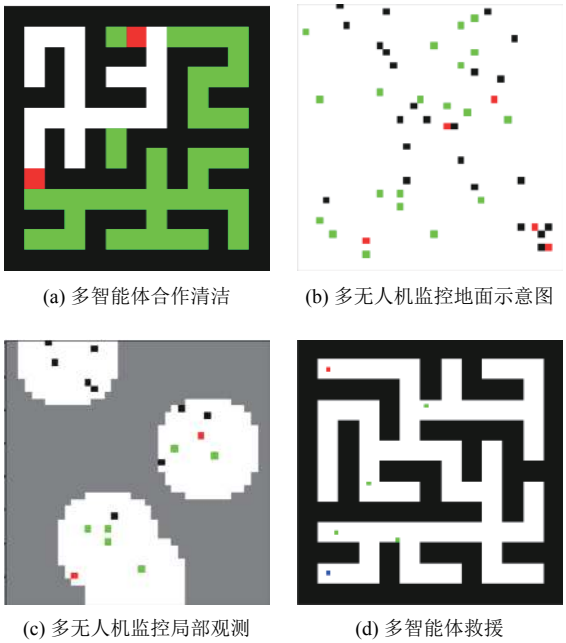
表 1 本文实验环境配置

设备名称	设备信息
CPU	AMD Ryzen 9 5900X,12c,24t
GPU	Nvidia GeForce RTX 3090
操作系统	Ubuntu 18.04 LTS
CUDA版本	11.1.0
CUDNN版本	8.1.0
Python版本	3.6.12
PyTorch版本	1.7.1

4.2 实验场景

本文实验包含了 3 个实验场景, 如图 2 所示. 这 3 个实验场景分别是多智能体合作清洁, 多无人机监控, 多智能体救援. 本节将对这 3 个实验场景进行简单描述.

在多智能体清洁场景中, 两个或者多个智能体将合作打扫一个房间, 实验场景如图 2(a) 所示. 环境中智能体的任务是在规定的时间内尽量扩大清洁面积, 或者是在最短的时间内尽量清洁完所有的区域. 该任务实际上是一个路径规划任务, 智能体之间应该合作规划具有最小重叠区域的路径. 每个智能体的局部观测由周围 8 个方块的状态组成, 一共 8 维. 动作空间是 1 维, 可以朝上下左右 4 个方向移动.



(a) 多智能体合作清洁 (b) 多无人机监控地面示意图

(c) 多无人机监控局部观测 (d) 多智能体救援

图 2 实验场景示意图

多无人机监控实验场景如图 2(b) 所示. 正方形区域表示地面, 空中有多个无人机使用摄像头监控地面, 无人机可以在给定区域自由飞行. 每架无人机都有一个由半径定义的局部观测区域 (如图 2(c) 所示), 无人机只能观测到该区域内的物体. 在该实验场景中, 多个智能体 (无人机) 的目标是尽量减少视觉区域的重叠以及监控更多的行人. 每个智能体的动作空间包含朝上下左右 4 个方向移动以及停留原地 5 个动作.

多智能体救援实验场景如图 2(d) 所示. 方形区域模拟地震后的废墟. 多个智能体从安全点出发, 进入该区域, 找到该区域的人并带回安全点. 该任务的目标是智能体尽可能在更短的时间内将更多的人带回安全点. 每个智能体的局部观测范围是其前方 90 度的扇形区域, 半径为 3 个方块的长度. 智能体的动作空间维度大小为 4, 分别表示沿x轴方向行走$[-0.5,0.5]$, 沿y轴方向行走$[-0.5,0.5]$, 旋转$[-10,10]$, 背起/放下人$\{0,1,2\}$.

3 个实验场景中的超参数如表 2 所示.

表 2 本文实验场景的超参数

实验场景	Actor	多步回报-N	值分布		
			N	$V_{\min}$	$V_{\max}$
多智能体合作清洁	{1,4}	{1,5}	51	−200	400
多无人机监控	{1,4}	{1,5}	51	−600	7000
多智能体救援	{1,4}	{1,5}	51	0	10000

4.3 实验结果分析

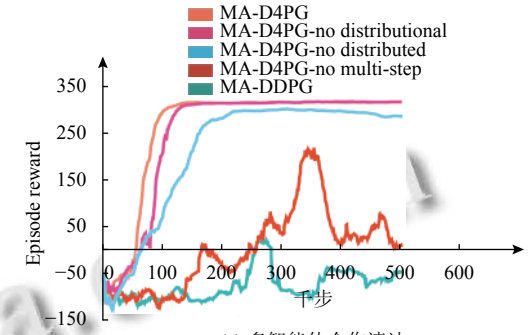
在框架上, 本文较多借鉴了目前性能最好的多智能体深度强化学习算法-多智能体深度确定性策略梯度算法, 因此选择与该算法进行对比实验. 此外还进行了消融实验, 分别确定本文所提每个方法的贡献. 图 3 展示了本文算法 (MA-D4PG) 以及 3 个消融实验和 MA-DDPG 在该实验环境中的性能. 其中 MAD4PG-no distributional 所示曲线表示本文算法去掉第 3.1 节所述改进后的运行效果, MAD4PG-no multistep 所示曲线表示本文算法去掉第 3.2 节所述改进的运行效果, MAD4PG-no distributed 所示曲线表示本文算法去掉第 3.3 节所述改进的运行效果.

从图 3(a) 中可看到在多智能体合作清洁的场景中, 本文算法 MA-D4PG 在收敛速度和最终收敛的 Episode Reward 值这两个指标上都有最好的表现, 并且算法的稳定性更好. 对 MAD4PG-no distributional/MAD4PG-no distributed 这两条曲线进行分析可看出, 第 3.1 节和第 3.3 节所述的改进对算法最后收敛的 Episode Reward值的提升虽不太明显, 但是加快了算法的收敛速度.

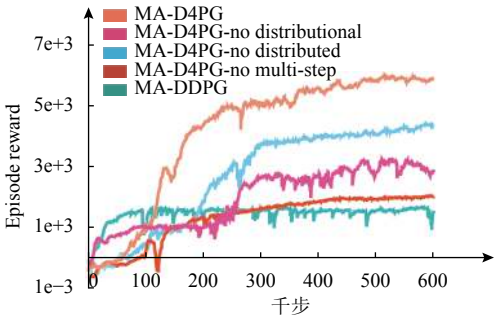
对图 3(b) 进行分析可得, 在多人机监控场景中, 相较 MA-DDPG 算法, 本文算法 (MA-D4PG) 收敛速度较慢, 但是最终收敛的 Episode Reward 值约是 MA-DDPG 的 3.6 倍, 智能体的决策能力得到了极大地增强. 对 MAD4PG-no distributional/MAD4PG-no distributed 分析可得, 第 3.1 节和第 3.3 节所述改进使得最终收敛的 Episode Reward值分别提升了约 97% 和 43%.

对图 3(c) 分析可得, 在多智能体救援场景中, 本文算法 (MA-D4PG) 的收敛速度约是 MA-DDPG 算法的 3.75 倍. 对 MAD4PG-no multistep/MAD4PG-no distributed 两条曲线分析可得, 第 3.2 节和第 3.3 节所述改进使得算法的收敛速度分别提升了约 185% 和 100%.

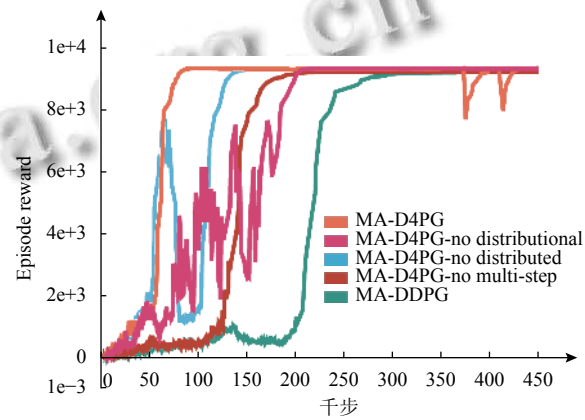
对图 3(a), 图 3(b) 的 MAD4PG-no multistep 曲线联合分析可看出, 去掉多步回报的改进, 算法性能下降最大, 并且去掉该改进会导致算法的稳定性显著下降. 这两个实验场景中智能体只包含一维的离散动作空间, 控制任务相对简单. 从中可看出在离散控制的多智能体场景中, 引入多步回报对算法性能的贡献最大, 而第 3.1 节和第 3.3 节所述改进之间的互补作用较小.



(a) 多智能体合作清洁



(b) 多无人机监控



(c) 多智能体救援

图 3 本文算法性能对比图

对图 3(c) 的 MAD4PG-no distributional 曲线分析可得, 去掉值分布的改进, 算法的收敛速度降低最大, 并且算法的稳定性也有一定的下降. 该实验环境中的控制任务比较复杂, 既包含连续动作, 又包含离散动作, 而前两个实验场景中智能体只包含一维的离散动作空

间. 从中可看出在复杂的控制任务中, 引入值分布对算法性能的贡献最为突出.

5 结论与展望

本文的主要贡献是提出了一种基于值分布的多智能体分布式策略梯度算法, 利用值分布替代期望值, 进一步区分智能体行为的好坏, 而不是简单地将其平均为期望值; 同时引入了多步回报增加智能体的预见能力, 提高算法稳定性; 此外引入了分布式数据生成框架将多智能体场景中智能体与环境交互生成经验数据和网络更新解耦, 加快算法的收敛. 本文算法在 3 个实验场景中都实现了最好的性能. 这 3 项改进都有助于提升本文算法 (MA-D4PG) 的整体性能, 在前两个实验场景中对性能贡献最大的改进是多步回报. 而在复杂的任务场景中, 值分布的贡献变得更加突出. 本文的实验场景都是基于多智能体合作的场景, 接下来的研究工作是在竞争场景甚至既有合作又有竞争的混合场景中研究本文算法的性能.

参考文献

1 Mnih V, Kavukcuoglu K, Silver D, *et al*. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529–533. [doi: 10.1038/nature14236]

2 Andrychowicz M, Baker B, Chociej M, *et al*. Learning dexterous in-hand manipulation. The International Journal of Robotics Research, 2020, 39(1): 3–20. [doi: 10.1177/0278364919887447]

3 Dabney W, Kurth-Nelson Z, Uchida N, *et al*. A distributional code for value in dopamine-based reinforcement learning. Nature, 2020, 577(7792): 671–675. [doi: 10.1038/s41586-019-1924-6]

4 Kiran BR, Sobh I, Talpaert V, *et al*. Deep reinforcement learning for autonomous driving: A survey. arXiv: 2002.00444, 2021.

5 Hua YX, Li RP, Zhao ZF, *et al*. GAN-powered deep distributional reinforcement learning for resource management in network slicing. IEEE Journal on Selected Areas in Communications, 2020, 38(2): 334–349. [doi: 10.1109/JSAC.2019.2959185]

6 Liang W, Huang WH, Long J, *et al*. Deep reinforcement learning for resource protection and real-time detection in IoT environment. IEEE Internet of Things Journal, 2020, 7(7): 6392–6401. [doi: 10.1109/JIOT.2020.2974281]

7 Chen MJ, Wang T, Ota K, *et al*. Intelligent resource allocation management for vehicles network: An A3C learning approach. Computer Communications, 2020, 151: 485–494. [doi: 10.1016/j.comcom.2019.12.054]

8 Nguyen TT, Nguyen ND, Nahavandi S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. IEEE Transactions on Cybernetics, 2020, 50(9): 3826–3839. [doi: 10.1109/TCYB.2020.2977374]

9 Sukhbaatar S, Lin ZM, Kostrikov I, *et al*. Intrinsic motivation and automatic curricula via asymmetric self-play. arXiv: 1703.05407, 2018.

10 Palmer G, Tuyls K, Bloembergen D, *et al*. Lenient multi-agent deep reinforcement learning. arXiv: 1707.04402, 2018.

11 Foerster J, Farquhar G, Afouras T, *et al*. Counterfactual multi-agent policy gradients. Proceedings of the 32nd AAAI Conference on Artificial Intelligence. New Orleans: AAAI, 2018. 2974–2982.

12 Lowe R, Wu Y, Tamar A, *et al*. Multi-agent actor-critic for mixed cooperative-competitive environments. arXiv: 1706.02275, 2020.

13 Bellemare MG, Dabney W, Munos R. A distributional perspective on reinforcement learning. Proceedings of the 34th International Conference on Machine Learning. Sydney: PMLR, 2017. 449–458.

14 Horgan D, Quan J, Budden D, *et al*. Distributed prioritized experience replay. arXiv: 1803.00933, 2018.

15 Silver D, Lever G, Heess N, *et al*. Deterministic policy gradient algorithms. Proceedings of the 31st International Conference on Machine Learning. Beijing: JMLR, 2014. 387–395.

16 Lillicrap TP, Hunt JJ, Pritzel A, *et al*. Continuous control with deep reinforcement learning. arXiv: 1509.02971, 2019.

17 Barth-Maron G, Hoffman MW, Budden D, *et al*. Distributed distributional deterministic policy gradients. arXiv: 1804.08617, 2018.

18 Hessel M, Modayil J, Van Hasselt H, *et al*. Rainbow: Combining improvements in deep reinforcement learning. Proceedings of the 32nd AAAI Conference on Artificial Intelligence. New Orleans: AAAI, 2018. 3215–3222.

19 Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv: 1412.6980, 2017.