控制与决策
Control and Decision

部分可观测下基于RGMAAC算法的多智能体协同

王子豪, 张严心, 黄志清, 殷辰堃

引用本文:
王子豪,张严心,黄志清,殷辰. 部分可观测下基于RGMAAC算法的多智能体协同[J]. 控制与决策, 2023, 38(5): 1267−1277.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2022.0422

您可能感兴趣的其他文章

Articles you may be interested in

基于多智能体强化学习的无人艇协同围捕方法

Research on cooperative hunting method of unmanned surface vehicle based on multi−agent reinforcement learning

控制与决策. 2023, 38(5): 1438−1447 https://doi.org/10.13195/j.kzyjc.2022.0564

基于多智能体深度强化学习的船舶协同避碰策略

Ship cooperative collision avoidance strategy based on multi−agent deep reinforcement learning

控制与决策. 2023, 38(5): 1395−1402 https://doi.org/10.13195/j.kzyjc.2022.1159

多智能体深度强化学习及其可扩展性与可迁移性研究综述

A survey on scalability and transferability of multi−agent deep reinforcement learning

控制与决策. 2022, 37(12): 3083−3102 https://doi.org/10.13195/j.kzyjc.2022.0044

基于过滤机制筛选信息的多智能体策略方法

Research on multi−agent strategy based on filtering mechanism to filter information

控制与决策. 2022, 37(6): 1643−1648 https://doi.org/10.13195/j.kzyjc.2020.1139

基于深度强化学习的多配送中心车辆路径规划

Deep reinforcement learning for multi−depot vehicle routing problem

控制与决策. 2022, 37(8): 2101−2109 https://doi.org/10.13195/j.kzyjc.2021.1381

部分可观测下基于RGMAAC算法的多智能体协同

王子豪[1], 张严心[1†], 黄志清[2], 殷辰堃[1]

(1. 北京交通大学 电子信息工程学院，北京 100091；2. 北京工业大学 信息学部，北京 100124)

摘 要: 多智能体深度强化学习(MADRL)将深度强化学习的思想和算法应用到多智能体系统的学习和控制中, 是开发具有群智能体的多智能体系统的重要方法. 现有的MADRL研究主要基于环境完全可观测或通信资源不受限的假设展开算法设计,然而部分可观测性是多智能体系统实际应用中客观存在的问题,例如智能体的观测范围通常是有限的,可观测的范围外不包括完整的环境信息,从而对多智能体间协同造成困难. 鉴于此,针对实际场景中的部分可观测问题,基于集中式训练分布式执行的范式,将深度强化学习算法Actor-Critic扩展到多智能体系统,并增加智能体间的通信信道和门控机制,提出recurrent gated multi-agent Actor-Critic算法(RGMAAC). 智能体可以基于历史动作观测记忆序列进行高效的通信交流,最终利用局部观测、历史观测记忆序列以及通过通信信道显式地由其他智能体共享的观察进行行为决策;同时,基于多智能体粒子环境设计多智能体同步且快速到达目标点任务,并分别设计 2 种奖励值函数和任务场景. 实验结果表明,当任务场景中明确出现部分可观测问题时, RGMAAC算法训练后的智能体具有很好的表现,在稳定性方面优于基线算法.

关键词: 多智能体；深度强化学习；部分可观测；多智能体深度确定性策略梯度；智能体间通信

中图分类号: TP181 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2022.0422

引用格式: 王子豪,张严心,黄志清,等. 部分可观测下基于 RGMAAC 算法的多智能体协同[J]. 控制与决策, 2023, 38(5): 1267-1277.

Multi-agent collaboration based on RGMAAC algorithm under partial observability

WANG Zi-hao[1], *ZHANG Yan-xin*[1†], *HUANG Zhi-qing*[2], *YIN Chen-kun*[1]

(1. School of Electronic and Information Engineering，Beijing Jiaotong University，Beijing 100091，China；2. Department of Information Science， Beijing University of Technology， Beijing 100124， China)

Abstract: Multi-agent deep reinforcement learning (MADRL) applies the ideas and algorithms of deep reinforcement learning to the learning and control of multi-agent systems, which is an important method to develop multi-agent systems with swarm agents. Existing MADRL studies mainly design algorithms based on the assumption that the environment is completely observable or communication resources are not limited. However, partial observability is an objective problem in the practical application of multi-agent systems. For example, the observation range of agentsis is usually limited, and the complete environmental information is not included outside the observable range, which makes it difficult for multi-agent collaboration. Aiming at the problem of partial observability in real scenes, based on the paradigm of centralized training and distributed execution, this paper extends the deep reinforcement learning algorithm Actor-Critic to multi-agent systems and adds communication channels and gating mechanisms between agents, finally proposes a recurrent gated multi-agent Actor-Critic (RGMAAC) algorithm. Agents can communicate efficiently based on the historical action observation sequence, and finally use the local observation, the historical observation sequence and observations shared by other agents through communication channels to make behavior decisions. Meanwhile, based on the multi-agent particle environment, the multi-agent task of synchronous and fast arrival is designed, and two reward value functions and task scenarios are designed respectively. The experimental results show that the trained agent with the RGMAAC algorithm performs well and is superior to the baseline algorithm in terms of stability when some observable problems clearly appear in the task scenario.

Keywords: multi-agent；deep reinforcement learning；partial observable；MADDPG；communication between agents

收稿日期: 2022-03-18；**录用日期:** 2023-02-28.

责任编委: 杨涛.

†通讯作者. E-mail: yxzhang@bjtu.edu.cn.

0 引 言

深度强化学习(DRL)作为一种可以解决复杂问题的有效手段,极大地推动了人工智能和自动化技术的发展[1]. 将DRL扩展到多智能体系统决策问题,对于构建能够与其他系统以及人类进行有效交互的人工智能系统至关重要[2],同时也符合人工智能的发展目标,即从感知到决策、从单智能体到多智能体系统的发展过程. 许多现实中的复杂问题,如自动驾驶汽车协调[3]、网络路由[4]和多机器人群控制问题[5]等都可以被描述为多智能体间存在合作或竞争的任务场景,DRL为训练鲁棒智能体提供了一个强大的框架.

多智能体深度强化学习(MADRL)研究目前面临3个核心挑战:一是环境的非平稳性,当多个智能体同时与环境交互学习策略时,对于单个智能体而言,交互的环境是在变化的,每个智能体需要同时考虑其他智能体策略变化对于自身策略的影响;二是状态空间和动作空间随着智能体个数增加呈指数级增加,因此如果直接将单智能体中成熟的算法应用于多智能体系统中,则与环境交互学习的速度将会非常缓慢,甚至智能体的策略难以收敛;三是环境的部分可观测性,目前很多MADRL研究基于不切实际的假设,比如环境的完全可观察性,每个智能体孤立地与环境交互学习,又或是对于通信网络资源无限制的访问,这些都不存在于现实的多智能体任务场景中. 例如,对于系统中的单个智能体而言,并不能观测到所有智能体的状态和策略,或由于传感器的带宽以及精度存在限制时,部分可观测性问题便会出现,这使得多智能体强化学习可以被建模为分布式部分可观测马尔可夫决策过程(DEC-POMDP)[6].

针对环境的非平稳性和状态动作空间爆炸的问题,目前主流的MADRL算法都采用集中式训练分布式执行(CTDE)的范式解决这个两个棘手的问题,如MADDPG[7]、DOP[8]、Mi-DDPG[9]等将策略梯度算法与CTDE相结合,能够有效解决多智能体粒子环境中多种任务场景. 其中多智能体深度确定性策略梯度算法MADDPG可以应用于智能体间合作、竞争以及二者同时存在的混合场景,同时可以估计其他智能体的策略以及采用策略集合优化的思想,增强了算法的鲁棒性,常作为MADRL领域的基线算法. QPLEX[10]、QMIX[11]和VDAC[12]等将值函数算法与CTDE相结合,能够有效解决星际争霸中大规模的复杂任务. 针对多智能体系统存在的部分可观测问题,结合智能体间通信的MADRL算法,如CommNet[13]、TarMAC[14]和DGN[15]等,在智能体间建立显示的信道,可互相传递各自的环境信息以缓解部分可观测的问题. 然而,这类基于通信的MADRL方法并未考虑在实际场景中,智能体并不能无限制地访问通信网络,即通信资源是有限的. 例如在1回合100个时间步的多智能体路径规划任务中,智能体间不能在每个时间步都互相交流各自的信息,对于单个智能体而言,可观测的范围也是有限的,因此最终难以互相协作完成目标任务.

本文提出多智能体深度强化学习算法RGMAAC. 不同于MADDPG和Mi-DDPG,RGMAAC打破了环境完全可观测的假设,即每个智能体不能观测到其他智能体的任何信息,仅能根据自身局部观测和通信消息进行决策,更加符合现实场景中的情况. 与CommNet相比,RGMAAC在基于部分可观测限制的同时,考虑到实际场景中智能体间有限通信资源的限制,每个智能体仅能在20%的回合时间步中进行互相通信交流,因此智能体需要学会如何高效地通信,包括何时以及与哪些智能体通信.

1 基于CTDE的多智能体深度强化学习

1.1 多智能体系统

单智能体强化学习建模为马尔科夫决策过程(MDP),而多智能体强化学习(MARL)遵循随机博弈过程,因为多智能体系统中的单个智能体奖励函数的计算不仅仅由自身的策略决定,还与博弈中其他智能体的策略有关,因此随机博弈的解决方案不同于MDP. 常用马尔科夫博弈描述多智能体系统,图1给出了多智能体强化学习的基本框架. 以元组$(N, S, A_1, \ldots, A_N, T, r_1, \ldots, r_N, \gamma)$进行描述. 其中:$N$为智能体的数目,$S$为联合状态空间,$A_i$为第$i$个智能体的动作空间,$T$为联合状态转移概率,$r_i : S \times A_1, \ldots, A_N \longrightarrow R$为第$i$个智能体的奖励回报,$\gamma$为折扣因子.

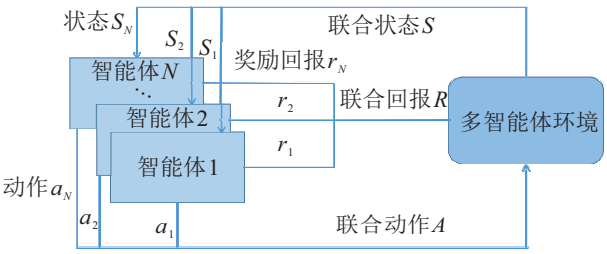


图1 多智能体强化学习框架

相比于单智能体系统,多智能体系统的特点可总结如下:

1) 单智能体的状态转移仅与自身策略选择的动作相关,而多智能体系统中的状态转移由联合动作共同决定.

2) 在多智能体系统中,每个智能体的奖励回报不仅与自身策略选择的动作相关,还受联合动作决定,因此每个智能体各获得的回报都可能存在差异,且均有各自的状态价值函数. 在多智能体系统中,智能体i的状态价值函数V_i为智能体在联合策略$\vec{\pi}$下的累计期望奖励,即当智能体采用策略$\vec{\pi}$时累计回报服从一个分布,在状态s处的期望值定义为状态值函数,有

$$V^i_{\pi^i,\pi^{-i}}(s)=E^{\vec{\pi}}\left\{\sum_{t=0}^{\infty}\gamma^t r^t(s^t,\vec{a})\Big|_{a^i\sim\pi^i(\cdot|s)}\right\}. \quad (1)$$

其中:$\vec{a}=(a_1,a_2,\ldots,a_N)$为联合动作,$\vec{\pi}=(\pi_1,\pi_2,\ldots,\pi_N)$为智能体的联合策略,$i$为除智能体$i$之外的$N-1$个智能体,$a^i\sim\pi^i(\cdot|s)$表示每个智能体的动作由各智能体的策略基于环境状态决策得来. γ的大小体现了对未来奖励和当前奖励的重要程度,当γ为0时表示智能体只考虑当前奖励,当γ为1时表示未来每一时刻的奖励与当前奖励同等重要. 智能体状态价值函数的定义同样体现了多智能体系统的特点,即状态价值函数由联合动作共同决定,多智能体系统的值函数同样是对联合策略进行评价.

1.2　DDPG

深度确定性策略梯度算法(DDPG)是用于连续高维动作空间学习的一种流行的无模型强化学习算法,在多智能体强化学习中可以作为一类算法设计的基础. DDPG基于Actor-Critic结构训练学习确定性的策略$\pi:S\longrightarrow A$和状态动作值函数$\pi:S\times A\longrightarrow R$,Critic网络对Actor网络的决策动作进行评估. Actor和Critic网络分别由参数为θ和ϕ的深度神经网络参数化表示,均有独立的副本,即为目标网络,目标网络的参数被周期性地延迟更新,以保持稳定性,DDPG网络结构如图2所示. 为了平衡探索与利用,在策略网络中加入OU随机噪声,智能体在与环境进行一次交互后将四元组存储到经验池中.

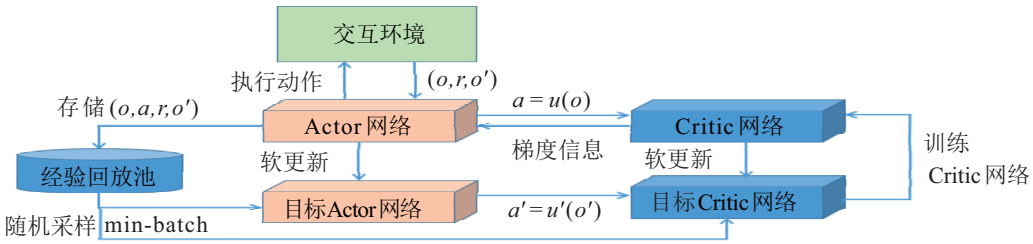


图 2　DDPG训练及网络结构

在训练阶段,从经验池中随机抽取经验用于训练网络,DDPG通过梯度下降法更新Actor网络,即

$$\nabla_\theta J(\theta)=E_{o\sim D}[\nabla_\theta\mu(a\ o)\times\nabla_a Q(o,a)|_{a=\mu(o)}], \quad (2)$$

其中θ为确定性策略网络u的网络参数. Actor网络利用Critic网络对决策动作评估,通过策略梯度的方法进行更新.

DDPG通过最小化贝尔曼误差更新Critic网络,即

$$\begin{cases}L(\phi)=E_{(o,a,o',r)\sim D}[(Q(o,a)-y)],\\ y=r+\gamma Q'(o',a')|_{a'=\mu'(o')}.\end{cases} \quad (3)$$

其中:Critic网络Q和目标Critic网络Q'分别由ϕ和ϕ'参数化表示,目标Actor网络u'由θ'参数化表示. 目标Critic网络Q'的输入包括智能体下一时刻的观测o'和由目标Actor网络根据下一时刻观测o'做出的决策动作a'.

目标Actor网络和目标Critic网络参数的更新采用软更新的方式,τ为软更新超参数,τ越小更新越“软”. 虽然每次更新参数变化的范围受限,但是可以使训练稳定易于收敛,即

$$\theta'=\tau\theta+(1-\tau)\theta', \quad (4)$$

$$\phi'=\tau\phi+(1-\tau)\phi'. \quad (5)$$

1.3　集中式训练分布式执行

单多智能体强化学习发展至今,不同的算法按照训练范式可分为完全集中式、完全分布式和集中式训练分布式执行(CTDE)三种训练范式. 其中:完全集中式的缺点是算法的扩展性较差,所有智能体的策略由一个集中式联合的策略决定;完全分布式的缺点是难以解决多智能体系统中存在的环境非平稳性的问题,同时也不能解决智能体间的信度分配问题;而CTDE是目前最常用也是效果最好的训练范式.

集中式训练,是指在采用强化学习算法的智能体在与动态环境交互训练的过程中,采用联合状态价值动作值函数对策略进行评估,相比于完全分布式训练使用的$q_i(s_i,a_i)$,集中式$Q_i(s_1,\ldots,s_n,a_1,\ldots,a_n)$考虑了联合状态和联合动作,同时也可以加入其他额外信息帮助值函数对联合策略进行更优地评估. 在训练阶段考虑更多的信息,虽然会降低与环境交互学习的速度,但是对于算法最终收敛的效果有非常明显的

改善,且集中式训练方式还可以解决多智能体强化学习的非平稳性问题.

分布式执行,是指在完成强化学习的交互训练过程后,智能体具备了在面临不同状态做出最优决策的能力. 同样地,在多智能体系统中,经过集中式训练后,智能体具备做出最优决策的能力. 考虑到实际场景中对于观测的限制,智能体仅需根据局部观测做出决策.

综上,集中式训练分布式执行是在考虑强化学习算法训练过程和实际场景限制下的一种最佳的训练范式,不但能在训练过程中尽可能多地考虑到环境与智能体的状态,在执行阶段也仅需智能体的局部观测即可进行决策,符合实际场景. 因此,本文采用基于CTDE的范式将Actor-Critic算法扩展到多智能体系统.

1.4 MADDPG

MADDPG是基于CTDE范式最经典的Actor-Critic (AC)算法,假设智能体交互学习的环境中存在N个智能体,联合策略为$\pi = (\pi(\theta_1),\dots,\pi(\theta_N))$,每个智能体网络均采用单智能体深度强化学习算法DDPG,由Actor、Critic、目标Actor和目标Critic网络组成,结构如图3所示.

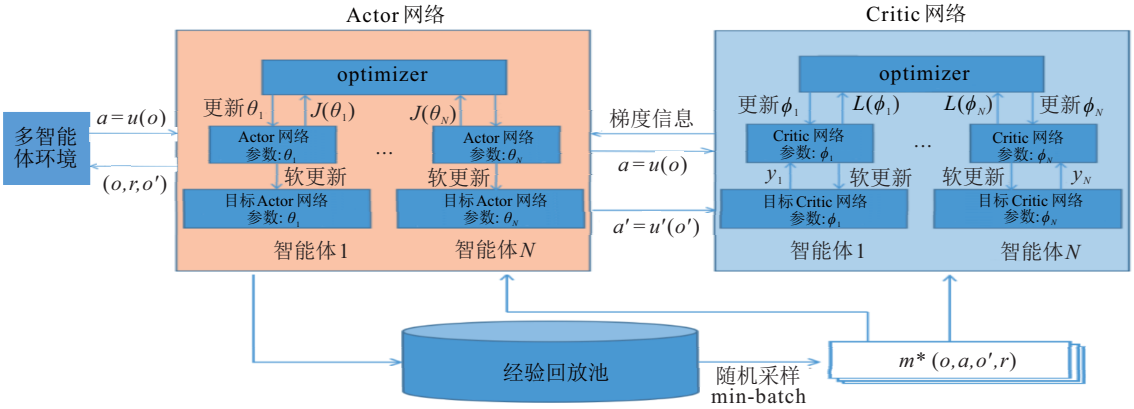


图 3 MADDPG训练及网络结构

为解决环境非平稳性和状态动作空间爆炸的问题,MADDPG采用CTDE范式实现寻找最优联合策略的目标,因此MADDPG对经验池存储数据的格式进行改进. 经验池中的一条经验由元组$(\vec{o},\vec{a},\vec{o}',r)$组成. 其中:$\vec{o} = (o_1,o_2,\dots,o_N)$为$t$时刻所有智能体自身观察的集合,$\vec{a} = (a_1,a_2,\dots,a_N)$表示所有智能体的动作集合. MADDPG选择将AC算法与CTDE范式相结合,是因为智能体的策略网络即Actor网络的输入在训练和实际执行时应该一致,无法将额外的信息直接结合在策略的输入中,所以要在训练阶段将全局观测和动作输入到集中式的Critic网络中. 如图4所示,在MADDPG的训练阶段,每个智能体Q函数的输入(即Critic网络的输入)包括所有智能体的观测和动作集合. 在训练完毕后的执行阶段,智能体只需要根据自己部分观测输入到Actor网络即可做出最优决策,巧妙地利用了AC算法本身的特点,将AC算法与CTDE相结合.

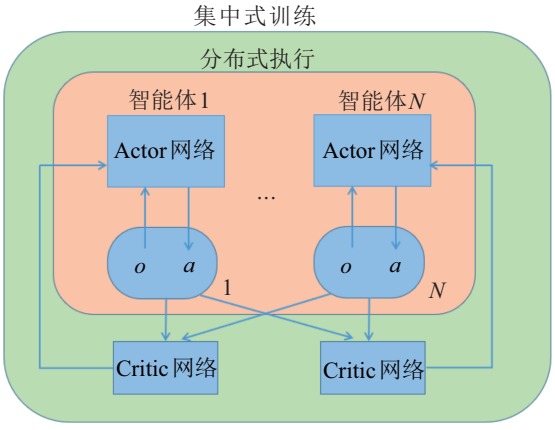


图 4 MADDPG算法模型

MADDPG通过梯度下降法更新单个智能体Actor网络,有

$$\nabla_{\theta_i} J(\theta_i) = E_{o\sim D}[\nabla_{\theta_i}\mu_i(a_i\ o_i)\nabla_{a_i} Q_i(\vec{o},\vec{a})|_{a=\mu(o)}], \tag{6}$$

其中Q_i为集中式的动作-价值函数,输入是所有智能体的动作$\vec{a}$和状态观测信息$\vec{o}$,输出是智能体的Q值. 通过最小化loss值更新Critic网络,借鉴DDPG中目标网络和时序差分的方法,有

$$\begin{cases} L(\phi_i) = E_{(o,a,o',r)\sim D}[(Q_i(\vec{o},\vec{a}) - y)], \\ y = r_i + \gamma Q_i'(\vec{o}',\vec{a}')|_{a'=\mu'(o')}. \end{cases} \tag{7}$$

其中:Q_i'为目标Critic网络;$\mu' = [\mu_1',\cdots,\mu_n']$为具有滞后更新参数$\theta_i'$的目标Actor网络,与DDPG目标网络的软更新方式保持一致. 由于每个智能体都拥有一个考虑联合观测和动作集合的Critic网络,智能体的奖励函数能够根据实际场景设计成不同的结构,

MADDPG可以应用在合作、竞争和混合的场景中.

式(6)、(7)与式(2)、(3)的区别在于,MADDPG的Critic网络和目标Critic网络的输入是所有智能体的观测和动作集合,而DDPG中的输入只有单个智能体的观测和动作.

1.5　Dec-POMDP

MADDPG在训练阶段假设可以观测到所有智能体,并且智能体间不存在通信信道. 然而,在大部分现实场景中,智能体在与环境交互学习时无法观测到完整的环境信息,这在单智能体强化学习中被描述为部分可观测马尔可夫决策过程(POMDP). 同样地,多智能体强化学习中的部分可观测问题可以被建模为分布式部分可观测马尔可夫决策过程(Dec-POMDP)[16],由元组 $(N, S, A, T, \Omega, O, R, \gamma)$ 构成. 其中: N 为智能体的数量, S 为全局状态集合, A 为全局动作集合, T 为状态转移概率函数, O 为所有智能体部分观测集合, Ω 为部分观测函数, R 为奖励值函数, γ 为折扣因子. 在每个时间步 t, 智能体根据局部观测 o_t^i 和策略 $\pi^i(o_t^i;\theta^i)$ 进行决策, θ^i 是智能体 i 的策略网络参数. 状态 S_t 转移到状态 S_{t+1} 由联合动作决定,即 $T(S_{t+1};S_t,a_t)$. 每个智能体的学习目标是最大化累计折扣奖励 $E\sum_{t=0}^{\infty} r_t\gamma^t$.

2　基于CTDE的RGMAAC网络模型设计

2.1　DRGMAAC网络模型设计

集中式训练分布式执行的范式虽然通过中心化的训练能够克服一部分环境不平稳的问题,但是由于在执行阶段每个智能体在部分可观测的条件下并不能够观测完整的环境信息,智能体之间的协同仍然存在困难. 为解决多智能体协同决策中存在的部分可观测问题,本文提出一种可以在部分可观测条件下,训练智能体利用有限通信资源进行高效通信交流的算法模型. 在MADDPG的Actor网络中引入通信门控模块,使智能体学会在不同的训练时间步选择是否与其他智能体通信,以此最大化单步通信的收益,同时将GRU(门控循环单元,一种循环神经网络)与MADDPG中的Actor和Critic网络相结合. 针对GRU隐状态如何初始化的问题,采用隐状态存储策略并对经验回放池中存储经验的数据结构进行设计,使智能体不仅根据局部观测决策,而且学会记住之前的信息,利用循环神经网络的记忆单元使智能体基于历史状态动作序列决策. 借鉴结合智能体间通信的MADRL算法,在智能体间加入显示的通信信道,使智能体可以选择在某些时间步与其他智能体进行通信,通信消息 m 可以补充到智能体的局部观测. 另一方面,通过智能体之间的信息传递能够实现“局部的中心化”,从而进一步缓解环境的非平稳性,促进智能体之间的协作. 最终,提出recurrent gated multi-agent Actor Critic (RGMAAC)算法模型如图5所示. 为了清晰地表述集中式训练和分布式执行的过程,将训练和执行拆解为两部分进行展示.

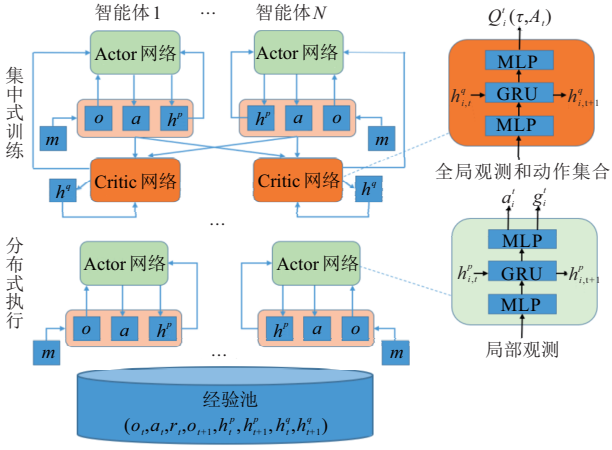


图 5　RGMAAC模型结构

图5上半部分为集中式训练阶段,下半部分为分布式执行阶段,即CTDE的核心思想在RGMAAC中具体体现. m 为智能体之间相互通信交流传递的信息,可作为对于智能体局部观测的补充,帮助智能体更好地进行决策,促进智能体之间的协作,通信的具体数据形式将在实验环节根据任务场景具体介绍. 对于GRU网络与其隐状态而言,在MADDPG的Actor和Critic网络中引入记忆单元,即将网络的第2层设计为GRU(图5右侧). 在具体实现过程中考虑到多智能体系统和强化学习特点,将交互数据四元组和智能体与环境交互前后的记忆一起存到经验池,GRU网络的隐状态即为对历史观测序列的记忆编码信息. 因此每条经验由元组 $(o_t, a_t, r_t, o_{t+1}, h_t^p, h_{t+1}^p, h_t^q, h_{t+1}^q)$ 组成,这样做的好处是,从经验池抽样数据训练时,使得网络利用的GRU隐状态与环境交互时的记忆一致,有助于逼近真实的状态并使GRU网络能够有效地利用历史信息,从而缓解多智能体系统存在的部分可观测问题. 其中: o_t 和 o_{t+1} 分别为所有智能体与环境交互前后的部分观测集合, a_t 为所有智能体的动作集合, h_t 为所有智能体在时间步 t 与环境交互前的记忆, h_{t+1} 为与环境交互时策略网络决策动作之后的记忆,上标 p 为策略网络,上标 q 指评价网络. 这样做的目的是,当从经验池抽样数据训练时,保证智能体训练使用的记忆与环境交互时的记忆一致,有助于更好地逼近真实状态,从而

更好地解决部分可观测和有限通信资源的限制.

相比于MADDPG, 在Critic网络加入GRU模块后, 智能体可以基于t时刻的部分观测集合与之前的观测序列记忆数据对决策动作评估, 同时对经验池存储的数据进行调整后, 可以帮助Critic网络更加准确地基于观测和记忆对决策动作进行评估. Recurrent-Critic网络的更新计算公式由式(7)更新为

$$\begin{cases} L(\phi_i) = E_D[(Q_i(\vec{o}_t, \vec{a}_t, h_{i,t}^q) - y_i)^2], \\ y_i = r_i + \gamma Q_i'(\vec{o}_{t+1}, \vec{a}_{t+1}, h_{i,t+1}^q)|_{a=u'(o_{t+1}, h_{t+1}^p)}. \end{cases} \tag{8}$$

状态动作值函数Q_i的输入增加了$h_{i,t}^q$, 目标状态动作值函数Q_i'的输入增加了$h_{i,t+1}^q$, 使得智能体充分利用历史经验对Actor网络的决策动作逼近真实评价. Actor网络的输入增加了$h_{i,t}^q$, 即$a_{i,t} \sim u(o_{i,t})$转变为$a_{i,t} \sim u(o_{i,t}, h_{i,t}^p)$, GRU和经验池的设计使得智能体能够更加准确地基于历史观测动作序列和当前局部观测进行决策.

在Recurrent-Actor网络中引入通信门控模块, 智能体不仅输出自身的下一步决策动作, 同时根据历史观测动作序列决定是否与其他智能体进行通信交流, 即智能体的通信决策动作g_i^t, 有

$$g_i^t = f^g(h_i^{t-1}), \tag{9}$$

$$h_i^t = GRU(e(o_i^{t-1}) + c_i^{t-1}, h_i^{t-1}). \tag{10}$$

Actor网络中门控模块f^g的输入为上一时刻GRU网络的隐状态记忆, 输出智能体选择是否通信的动作. 二进制动作g_i^t表示智能体在时间步t是否选择与其他智能通信, GRU隐状态h_i^t的更新由经过第1层全连接层网络编码后的o_i^{t-1}、其他智能体的通信消息c_i^{t-1}和上一时刻的隐状态h_i^{t-1}共同决定. 时刻t至时刻$t+1$的通信及隐藏记忆状态变化如图6所示.

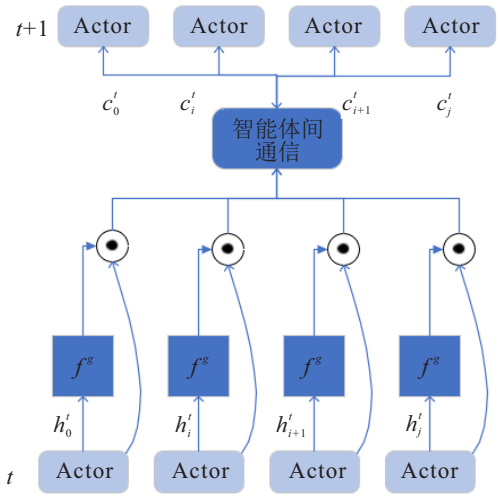


图 6 Recurrent Gated Actor模型结构

此时, 通过梯度下降法更新Recurrent Gated Actor网络的计算公式, 由式(6)更新如下:

$$\nabla_{\theta_i} J(\theta_i) = E_D[\nabla_{\theta_i} \mu_i(a_{i,t}|o_{i,t}, h_{i,t}^p) \times \nabla_{a_i} Q_i(\vec{o}_t, \vec{a}, h_{i,t}^q)|_{a=\mu(o_t, h_t^p)}]. \tag{11}$$

Actor网络的输入增加了$h_{i,t}^p$, GRU网络和经验池的设计使得智能体能够基于历史观测动作序列$\tau_i = (a_i^0, o_i^1, \ldots, a_i^{t-1}, o_i^t)$进行决策, 输出智能体的决策动作以及是否选择通信的二进制动作. 相比于MADDPG, 智能体可以基于时刻的部分观测集合与之前的观测序列数据进行决策, 同时对经验池存储的数据进行调整后, 可以帮助Actor网络更加准确地基于观测和记忆进行决策. RGMAAC算法步骤如算法1所示.

算法1 RGMAAC.

step 1: 初始化Actor网络$(\pi_\theta^1, \pi_\theta^2, \ldots, \pi_\theta^N)$、Critic网络$(Q_\phi^1, Q_\phi^2, \ldots Q_\phi^N)$.

step 2: 初始化目标Actor网络$\pi_{\bar{\theta}}^i \leftarrow \pi_\theta^i$, 目标Critic网络$Q_{\bar{\theta}}^i \leftarrow Q_\theta^i$.

step 3: 初始化经验回放池RB.

step 4: for 1 to max-episodes do:

step 5: 收到每个智能体的初始观测$o_{i,t}$.

step 6: for 1 to max-episode length do

step 7: 收集每个智能体Actor和Critic网络的隐状态: $h_{i,t}^p, h_{i,t}^q$.

step 8: 每个智能体的Actor网络输出采样一个动作: $a_{i,t}$.

step 9: 收集Actor网络的隐状态$h_{i,t+1}^p$.

step 10: 计算$Q_\theta^i = (o_t, a_t, h_{i,t}^q)$并收集计算$Q$值后的Critic网络隐状态$h_{i,t+1}^q$.

step 11: 执行联合动作$a = (a_1, \ldots, a_N)$, 接收环境的奖励值r并转移到下一个环境状态.

step 12: 将$(o_t, a_t, o_{t+1}, r_t, h_t^p, h_{t+1}^p, h_t^q, h_{t+1}^q)$存入经验回放池.

step 13: for agent $i = 1$ to N do:

step 14: 采用随机抽取的方式从RB中抽取min-batch大小的数据.

step 15: 基于式(11)更新Actor网络.

step 16: 基于式(8)更新Critic网络.

step 17: 软更新目标网络参数

$$\bar{\phi} \leftarrow (1-\tau)\bar{\phi} + \phi,$$

$$\bar{\theta} \leftarrow (1-\tau)\theta + \bar{\theta}.$$

step 18: end for

step 19: end for

step 20: end for

2.2　探讨GRU作用的两种基线算法

为了探究在部分可观测和有限通信资源限制下, 智能体基于历史状态动作序列决策和训练对于算法性能的影响, 即GRU网络在算法设计中的实际作用, 本文分别在Actor和Critic网络单独引入GRU并对经验池中存储的经验格式进行相应的调整, 后续实验章节中将以此作为基准算法与MADDPG和所提出的RGMAAC进行比较.

2.2.1　R-Actor

Recurrent on Actor (R-Actor)网络结构如图7所示, 只在智能体的Actor网络中引入GRU, 经验回放池存储的一条经验为 $(o_t, a_t, r_t, o_{t+1}, h_t^p, h_{t+1}^p)$. 通过梯度下降法更新Recurrent-Actor网络, 有

$$\nabla_{\theta_i} J(\theta_i) = E_{(o_t, h_t^p)\sim D}[\nabla_{\theta_i}\mu_i(a_{i,t}|o_{i,t}, h_{i,t}^p)\times \nabla_{a_i} Q_i(\vec{o}_t, \vec{a})|_{a=\mu(o_t, h_t^p)}]. \tag{12}$$

Critic网络基于下式进行更新:

$$L(\phi_i) = E_D[(Q_i(\vec{o}_t, \vec{a}_t) - y_i)^2],$$
$$y_i = r_i + \gamma Q_i'(\vec{o}_{t+1}, \vec{a}_{t+1})|_{a=u'(o_{t+1}, h_{t+1}^p)}. \tag{13}$$

相比于式(8)和(11), 智能体Critic网络和目标Critic网络的输入去掉了GRU的隐状态, 仅有智能体Actor网络和目标Actor网络能够根据自己的部分观测和记忆信息进行决策.

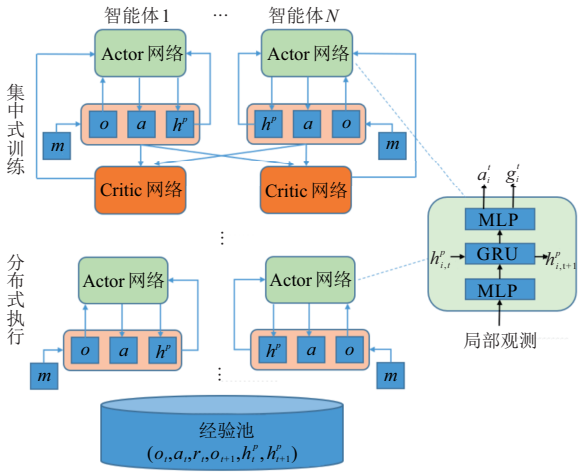


图 7　循环演员网络

2.2.2　R-Critic

Recurrent on Critic(R-Critic)网络结构如图8所示, 只在算法中的Critic网络加入GRU, 经验池存储一条经验为 $(o_t, a_t, r_t, o_{t+1}, h_t^q, h_{t+1}^q)$. 通过梯度下降法更新Actor网络, 有

$$\nabla_{\theta_i} J(\mu_i) = E_D[\nabla_{\theta_i}\mu_i(a_{i,t}|o_{i,t})\times \nabla_{a_i} Q_i^{\mu}(\vec{o}, \vec{a}, h_{i,t}^q)|_{a_j, t=\mu_j(o_{j,t})}]. \tag{14}$$

Recurrent-Critic网络基于下式进行更新:

$$\begin{cases} L(\phi_i) = E_D[(Q_i(\vec{o}_t, \vec{a}_t, h_{i,t}^q) - y_i)^2], \\ y_i = r_i + \gamma Q_i'(\vec{o}_{t+1}, \vec{a}_{t+1}, h_{i,t+1}^q)|_{a=u'(o_{t+1})}. \end{cases} \tag{15}$$

即相比于式(8)和(11), 智能体Actor网络和目标Actor网络的输入去掉了GRU的隐状态, 仅有智能体Critic网络和目标Critic网络能够根据自己的部分观测和历史经验对状态动作对进行评估.

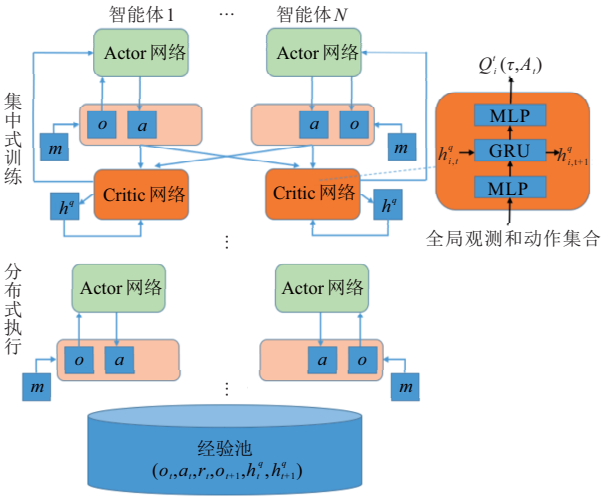


图 8　循环评论家网络

3　实验分析

3.1　实验基本设置

软件环境ubuntu16.04+Tensorflow+gym, 硬件为英伟达GeForce GTX 2080+64G内存, 基于多智能体粒子环境(multi-agent particle envs, MPE)设计多智能体同步且快速到达目标点的任务场景. MPE由Open AI团队设计并开源, 是多智能体深度强化学习领域最重要的开源测试环境之一. 本文利用开源的API设计更加有针对性的任务场景进行算法验证, 在后续实验场景介绍中详细介绍了智能体任务以及奖励函数设计等细节. 需要特别指出的是, 基于MPE设计的多智能体环境均是一个具有连续动作空间和离散时间的二维平面世界, 并不需要采用纯图像作为智能体的观测信息, 环境直接对有限数量的多智能体和地标进行坐标以及速度实时采集, 因此在神经网络模型中不需要加入卷积神经网络来提出图像特征, 可以让算法设计专注于强化学习算法和多智能体系统.

在2.1节和2.2节提出的模型中, Actor和Critic网络均由64个神经元的三层神经网络组成, 加入GRU的模型中, 第1层和最后1层为激活函数是relu的全连接层. 实验中, 设置学习率为0.01, 强化学习折扣因子为0.95, 经验池大小为100 000. 每个回合共100个时间步, 单次训练共30 000回合, 每次从经验池抽取

256批次大小的存储数据,所有实验均采用5个不同的随机种子以提高实验结果的可靠性.

不同于深度学习,强化学习的数据均由智能体与环境交互产生,因此衡量算法性能的标准主要有: 1) 在训练相同回合的基础上,比较智能体奖励值的大小,奖励值越大表明智能体学到的策略越优; 2) 比较不同训练随机种子的智能体奖励值曲线方差,方差越小表明智能体的表现越稳定,算法的稳定性越优; 3) 训练完毕后,比较智能体的具体表现,任务的完成度越高智能体的实际表现越佳,算法的性能越好. 在后续实验中,比较基线算法与RGMAAC在部分可观测限制下的算法性能.

3.2 多智能体同步且快速到达目标点任务

在二维平面中有2个智能体和1个目标点,地图中央存在一个体积比智能体大3倍的黑色障碍物,2个智能体从不同出发点开始移动,每回合开始时智能体和目标点的位置将随机变化. 2个绿色智能体的任务是同步且快速地到达黑色目标点,在移动过程中需要避免相互碰撞,同时还需要在向目标点移动的过程中避开比自己体积更大的黑色障碍物. 每个智能体可以获悉目标点和障碍物的位置坐标,但是仅能在预先设定的有限观测半径范围内获取到另一个智能体的位置信息,观测半径等于自身大小半径.

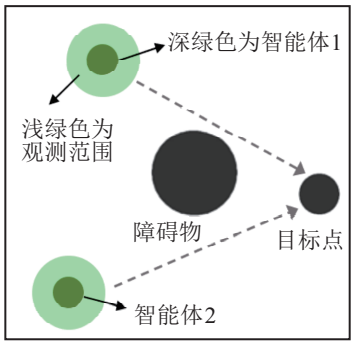


图 9 多智能体同步且快速到达目标任务示意图

每个智能体的完全状态观测 O_{full} 由 $(\text{Self}_x, \text{Self}_y, \text{Goal}_x, \text{Goal}_y, \text{Other}_x, \text{Other}_y, \text{Obs}_x, \text{Obs}_y)$ 组成, Self 为智能体自身位置坐标, Goal 为目标点坐标, Other 为其他智能体位置坐标, Obs 为障碍物位置坐标. 然而,在现实场景中智能体并不能在任何时刻都观测到其他所有智能体的位置坐标,因此在设计算法模型时,需要考虑多智能体系统存在的部分可观测问题. 结合智能体间通信的方法,智能体部分状态观测 O_{partial} 由 $(\text{Self}_x, \text{Self}_y, \text{Goal}_x, \text{Goal}_y, \text{msg}_x, \text{msg}_y, \text{Obs}_x, \text{Obs}_y)$ 组成, msg 为另一个智能体传递的位置坐标消息,只有当一个智能体决定通信时,其他智能体才能获悉其位置信息. 考虑到通信资源及带宽在实际场景中是有限的,因此后续实验中每个训练回合有100个时间步,其中20个时间步智能体可以选择将自身位置信息传递给另一个智能体. 将智能体的动作划分为环境动作和通信动作,环境行动是指对环境动态和智能体获得的奖励产生直接影响的行动,通信动作不会影响环境动态(除了其所传递的消息被其他智能体接受). 在有限通信资源的限制下,智能体需要学会记住最后一次传递的位置信息、最后通信的时间步以及选择在某一时间步选择通信如何随着时间的推移影响通信资源预算.

此任务场景根据智能体间奖励函数关系可归为合作型任务,因此每个智能体的奖励函数相同,联合奖励设计如下:

$$R_1 = -\sum_i d(\text{Self}_i, \text{Goal}) + C + D - \sum_{\text{pairs}(i,j)} |d(\text{Self}_i, \text{Goal}) - d(\text{Other}_j, \text{Goal})|. \tag{16}$$

其中

$$C = \begin{cases} -1, & \text{智能体间碰撞;} \\ 0, & \text{智能体间未碰撞.} \end{cases}$$

$$D = \begin{cases} -2, & \text{智能体与障碍物碰撞;} \\ 0, & \text{智能体未与障碍物碰撞.} \end{cases}$$

R_1 第1项为 R_{distance},是每个智能体与目标点距离的累加和的负数;第2项 C 为智能体间在移动过程中的碰撞奖励;第3项 D 为智能体在移动过程中与障碍物的碰撞奖励,因为在环境设置中,一旦与障碍物发生碰撞,智能体的速度便会降低至0,因此当与障碍物发生碰撞时奖励值为负. 第4项为智能体与目标点距离差值累加和的负数,奖励值函数这样设计的目的是要引导智能体最终学会如何快速到达目标点的过程中,还要与另两个智能体保持同步到达,距离目标点较近的智能体需要在目标点附近等待另两个智能体,因此智能体必须掌握其他智能体的位置信息,但也不需要在一回合的每个时间步都互相通信,因此智能体需要学会选择何时以及与哪些智能体通信.

为了在实验中对比不同奖励值函数设计对于智能体行为的影响和算法的实际效果,设计奖励值为 R_2 的对照实验,即去掉式(16)第4项,智能体只需要各自以最快的速度到达目标点,即智能体仅需根据自己的局部观测快速到达目标点,并不需要掌握其他智能体的位置信息,有

$$R_2 = -\sum_i d(\text{Self}_i, \text{Goal}) + C + D. \tag{17}$$

综上,为了通过实验对比算法的有效性,设计两种多智能体协同任务场景:避障同步快速到达目标任务,智能体的奖励值函数为式(16),因此智能体需要观测到其他智能体的位置信息才能与其他智能体同步到达目标点;避障快速到达目标任务,智能体的奖励值函数为式(17),因此智能体只需各自快速到达目标点,并不需要观测到其他智能体的位置. 通过不同的任务设置,可以探究当任务场景中存在部分可观测的限制时,智能体经过不同算法训练后的实际表现.

3.3　实验结果与分析

在部分可观测的条件下分别采用RGMAAC和3种基线模型,经过30 000回合训练,采用R_1或R_2为奖励函数的智能体奖励曲线如图10和图11所示.

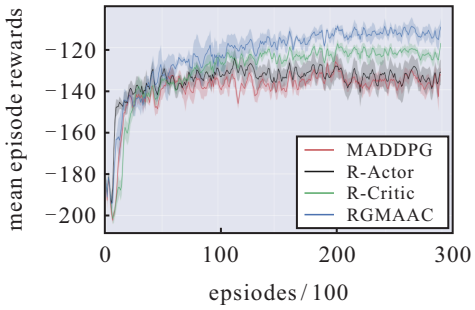


图 10　部分可观测下同步且快速到达任务中的奖励变化趋势

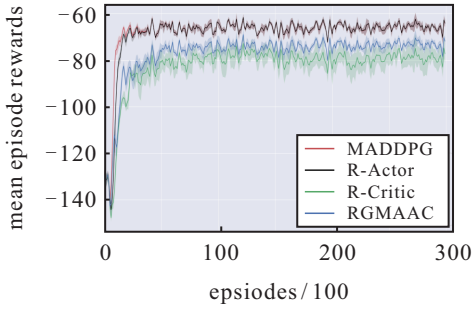


图 11　部分可观测下快速到达任务中的奖励变化趋势

由图10可见,在部分可观测的限制下,当智能体需要同步且快速到达目标点时,红色曲线明显低于绿色和蓝色曲线,表明在Critic网络或在AC网络中都融合GRU可以明显缓解部分可观测的问题. R-Actor训练的智能体表现逊于R-Critic算法,表明发挥主要作用的是在Critic网络中加入的记忆单元,这一点从Actor-Critic本身的结构能够解释,Actor网络的更新基于Critic网络对Actor网络选择的动作进行的评估,仅仅基于部分可观测状态的Critic网络并不能捕捉完全的环境动态信息,导致Actor网络最终收敛到一个不佳的策略. 上述实验结果表明,仅让Critic网络基于状态动作序列决策的效果(即Q网络为$Q_i^{\mu}(\vec{o},\vec{a},h_{i,t}^{q})$)优于仅让Actor网络基于状态动作序列决策的效果(即$\mu_i(a_{i,t}|o_{i,t},h_{i,t}^{p})$),且在RGMAAC中,Critic网络是考虑所有智能体动作和观测集合的,让Critic基于历史状态动作序列决策,可明显缓解部分可观测的限制. 由图11的实验结果可见,即使在部分可观察的限制下,智能体在基线模型下的表现也没有受到影响,因此对于快速到达目标点的任务智能体不需要观测到其他智能体的位置信息.

为了进一步比较算法在不同环境下的性能,在完全可观察的条件下分别采用RGMAAC和3种基线模型经过30 000回合训练,采用R_1或R_2为奖励函数的智能体奖励曲线如图12和图13所示.

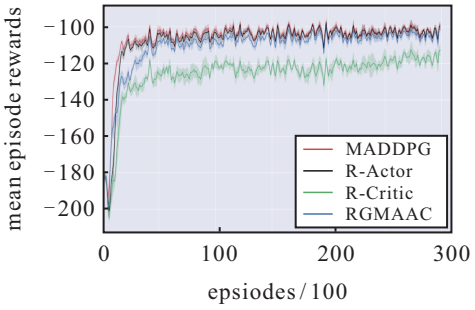


图 12　完全可观测下同步且快速到达任务中的奖励变化趋势

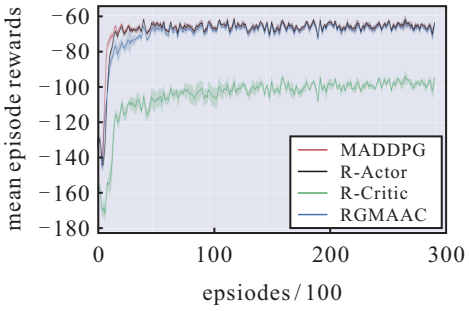


图 13　完全可观测下快速到达任务中的奖励变化趋势

两种任务场景中,图12和图13的实验结果表明,在完全可观测的条件下,GRU的记忆机制和通信门控机制并没有发挥作用,加入GRU网络后奖励值曲线反而下降了,收敛速度也明显慢于基线模型,这表明对于具体任务而言,如果不存在明确的部分可观测问题,则加入记忆机制并在经验回放池对隐状态进行存储反而会有副作用.

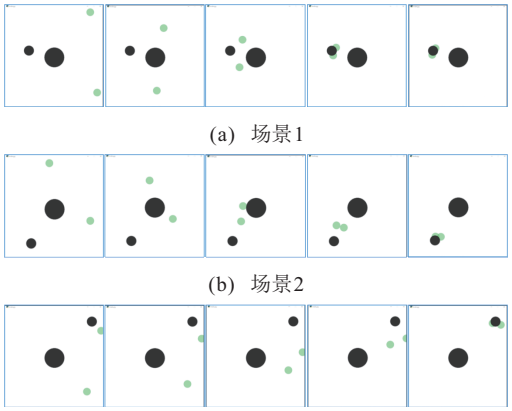
为了进一步通过实验对比经过RGMAAC和3种基线算法训练30 000回合后智能体的实际表现,在测试阶段从另外1 000回合中收集样本评估智能体学习到的策略,然后使用测试阶段的样本统计不同的性能指标,如表1所示. 其中:平均回合奖励是对智能体的实际表现进行量化,平均碰撞次数为智能体在移动过程中与其他智能体和障碍物的碰撞次数,同步到达成功率为在测试阶段智能体能够在一回合的100时间步之内同步到达目标点的成功率. 如表1所示,智

能体的实际表现与图10智能体奖励值展现的结果保持一致,明显地,RGMAAC在测试阶段的平均回合奖励、平均碰撞次数以及同步到达成功率3个指标均优于基线算法.

表1 部分可观测下同步且快速到达任务测试表现

实验算法	平均回合奖励	平均碰撞次数	同步成功率/%
MADDPG	−140.2	6.52	24.6
R-Actor	−130.4	6.52	31.9
R-Critic	−120.7	8.47	77.3
RGMAAC	−111.4	4.26	88.5

通过分析多智能体的实际表现验证算法的性能,在算法训练完30 000回合后的测试阶段,将一个回合内的决策行为进行渲染. 图14分别展示了不同的3个回合中智能体的位置移动变化情况. 在每一个场景中,最左侧的子图描述回合开始智能体和目标点的初始位置,最右侧子图描述为回合结束时两个智能体对目标点的到达情况,即通过分析在一个回合不同时期的智能体实际表现和任务完成情况更加直观地分析采用RGMAAC算法训练智能体的效果.



(a) 场景1

(b) 场景2

(c) 场景3

图14 部分可观测下同步且快速到达任务中智能体决策过程

为描述方便,统一将3个场景中与目标点初始距离较近的智能体记作智能体1,与目标点初始距离较远的智能体记作智能体2. 图15～图17分别对应这3个场景中两个绿色智能体与目标点距离的变化情况.

在场景1中,回合开始时两个智能体的初始化位置与目标点距离相差不大,最终两个智能体在避免碰撞的前提下同步到达目标点. 在场景2中,智能体1的初始化位置与目标点距离较近,随后智能体1和智能体2都在向目标点移动,当智能体1即将到达目标点时,由于智能体2与目标点相距较远,智能体1为了保持同步到达采取绕目标点移动的等待策略,等智能体2移动到目标点附近二者才一起同步抵达目标点. 在场景3中,两个智能体的初始化位置与目标点的距离有较大差距,智能体1的初始化位置就在目标点附近,为了保持同步,智能体1先朝着远离目标点的方向移动,等智能体2靠近目标点后两个智能体才一起同步地抵达目标点. 以上行为决策表明智能体学会了相互配合的协同策略,尽管智能体与目标点的初始距离不尽相同,但最终都能相互配合,以协同策略最终成功地完成同步且快速到达目标点的任务.

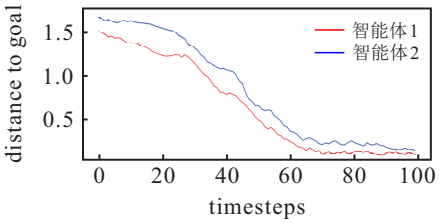


图15 场景1中智能体与目标点距离的变化趋势

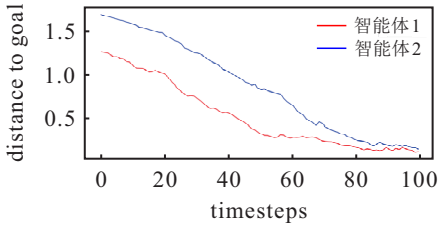


图16 场景2中智能体与目标点距离的变化趋势

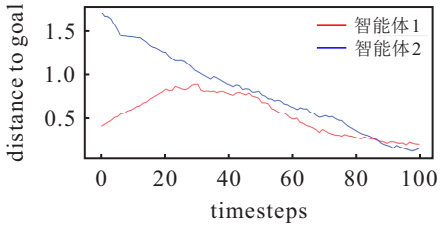


图17 场景3中智能体与目标点距离的变化趋势

综上,上述实验展示了RGMAAC和3种基线模型在完全可观测和局部可观测下的性能对比. 由此得出,在完全可观测下,4种算法训练后的智能体表现相当,MADDPG效果更佳. 而在部分观测和有限通信资源限制下,所提出的RGMAAC算法性能明显比基线模型性能更好. 这是因为对于局部观测环境而言,通过引入GRU可以让Q函数估计更加准确并可补充智能体的局部观测,门控机制使智能体学会如何通信交流;反之,通过对Actor网络引入GRU网络,全局Critic无法对过往数据进行利用,因此对Q函数的估计偏差较大.

4 结 论

为解决在实际场景中,多智能体系统存在的部分可观测问题,本文采用集中式训练分布式执行的范式,将深度强化学习算法Actor-Critic扩展到多智能体系统,并增加智能体间的通信信道和门控机制,提出多智能体深度强化学习算法RGMAAC. 同时,设计了部分可观测且需要高度协同的多智能体同步且快

速到达目标点任务和两种奖励值函数. 实验结果表明, RGMAAC的算法性能优于基线模型, 经训练后的智能体在不同的初始场景下都能学会相互配合的协同策略, 最终成功地完成同步到达目标点的任务. 基于此思想, 如何将RGMAAC在更大规模和更复杂的混合型任务场景中进行实验, 并优化智能体间通信, 进一步提高算法性能, 是下一步研究的重点.

参考文献(References)

[1] 梁星星, 冯旸赫, 马扬, 等. 多Agent深度强化学习综述[J]. 自动化学报, 2020, 46(12): 2537-2557.
(Liang X X, Feng Y H, Ma Y, et al. Deep multi-agent reinforcement learning: A survey[J]. Acta Automatica Sinica, 2020, 46(12): 2537-2557.)

[2] 孙长银, 穆朝絮. 多智能体深度强化学习的若干关键科学问题[J]. 自动化学报, 2020, 46(7): 1301-1312.
(Sun C Y, Mu C X. Important scientific problems of multi-agent deep reinforcement learning[J]. Acta Automatica Sinica, 2020, 46(7): 1301-1312.)

[3] Cao Y C, Yu W W, Ren W, et al. An overview of recent progress in the study of distributed multi-agent coordination[J]. IEEE Transactions on Industrial Informatics, 2013, 9(1): 427-438.

[4] Ye D, Zhang M, Yang Y. A multi-agent frame work for packet routing in wireless sensor networks[J]. Sensors, 2015, 15(5): 10026-10047.

[5] Hüttenrauch M, Sosic A, Neumann G. Guided deep reinforcement learning for swarm systems[J/OL]. 2017, arXiv: 1709.06011.

[6] Oliehoek F A, Amato C. Infinite-horizon dec-POMDPs[C]. A Concise Introduction to Decentralized POMDPs. Cham: Springer, 2016: 69-77.

[7] Lowe R, Wu Y, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments[C]. Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, 2017: 6382-6393.

[8] Wang Y H, Han B N, Wang T H, et al. DOP: Off-policy multi-agent decomposed policy gradients[J/OL]. 2021, arXiv: 2007.12322.

[9] 陈亮, 梁宸, 张景异, 等. Actor-Critic框架下一种基于改进DDPG的多智能体强化学习算法[J]. 控制与决策, 2021, 36(1): 75-82.
(Chen L, Liang C, Zhang J Y, et al. A multi-agent reinforcement learning algorithm based on improved DDPG in actor-critic framework[J]. Control and Decision, 2021, 36(1): 75-82.)

[10] Wang J H, Ren Z Z, Liu T, et al. QPLEX: Duplex dueling multi-agent Q-learning[J/OL]. 2021, arXiv: 2008.01062.

[11] Rashid T, Samvelyan M, de Witt C S, et al. Monotonic value function factorisation for deep multi-agent reinforcement learning[J/OL]. 2020, arXiv: 2003.08839.

[12] Su J Y, Adams S, Beling P. Value-decomposition multi-agent actor-critics[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2021, 35(13): 11352-11360.

[13] Sukhbaatar S, Szlam A, Fergus R. Learning multiagent communication with backpropagation[C]. Proceedings of the 30th International Conference on Neural Information Processing Systems. Barcelona, 2016: 2252-2260.

[14] Das A, Gervet T, Romoff J, et al. Tarmac: Targeted multi-agent communication[C]. International Conference on Machine Learning. Piscataway: IEEE, 2019: 1538-1546.

[15] Jiang J C, Dun C, Huang T J, et al. Graph convolutional reinforcement learning[J/OL]. 2018, arXiv: 1810.09202.

[16] Gmytrasiewicz P J, Doshi P. A framework for sequential planning in multi-agent settings[J]. Journal of Artificial Intelligence Research, 2005, 24: 49-79.

[17] Yu C, Velu A, Vinitsky E, et al. The surprising effectiveness of PPO in cooperative, multi-agent games[J/OL]. 2021, arXiv: 2103.01955.

[18] Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward[C]. Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. New York: ACM, 2018: 2085-2087.

[19] Son K, Kim D, Kang W J, et al. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning[C]. International Conference on Machine Learning. Piscataway: IEEE, 2019: 5887-5896.

[20] Pesce E, Montana G. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication[J]. Machine Language, 2020, 109(9/10): 1727-1747.

作者简介

王子豪 (1997–), 男, 硕士生, 从事多智能体深度强化学习的研究, E-mail: 1669169299@qq.com;

张严心 (1976–), 女, 副教授, 博士, 从事复杂大系统的智能控制、无人驾驶中的智能控制、复杂交通网络控制等研究, E-mail: yxzhang@bjtu.edu.cn;

黄志清 (1971–), 男, 副教授, 博士, 从事无人驾驶智能决策控制、车联网及区块链等研究, E-mail: zqhuang@bjtu.edu.cn;

殷辰堃 (1981–), 男, 副教授, 博士, 从事迭代学习控制、数据驱动控制、智能交通系统等研究, E-mail: chkyin@bjtu.edu.cn.

(责任编辑: 郑晓蕾)