Reinforcement Learning for Cooperative Multi-Agent Systems

Liangyawei Kuang
Hong Kong University of Science and Technology
lkuang@connect.ust.hk

Abstract

With the advances of robotics, autonomous vehicles, and other notable reinforcement learning (RL) applications in recent years, multi-agent RL (MARL) has reemerged with the advances in single-agent RL. As an interdisciplinary research area that includes learning, game theory, communication, and optimization, there are many technics come with challenges and restrictions in different MARL settings, MARL-related subareas, and theoretical foundation lackings. The centralized training and decentralized execution (CTDE) paradigm is used to tackle communication problems as it enables agents to share information during training and somehow reduce other concerns on nonstationarity and partial observability. However, with its inner constraints, this paradigm may be not beneficial in more general multi-agent settings with consideration of real-world problems so we need more powerful techniques to address those new issues. In this short paper which mainly focuses on cooperative MARL, I will provide problems background, introduce selective current challenges with related work, and propose potential directions for future solutions.

Background

In this section, I will review the theoretical methods from single-agent RL to MARL, especially methods related to cooperative MARL settings.

Markov Decision Process.
For a reinforcement learning question, we can formulate it as an infinite-horizon discounted Markov Decision Process (MDP). An MDP is defined by a quintuiple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the set of states; $\mathcal{A}$ is the set of actions; $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denotes the set of possibility from a state $s \in \mathcal{S}$ to a state $s' \in \mathcal{S}$, given a action $a \in \mathcal{A}$; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the immediate reward function for agents tranfer from (s, a) to s'; $\gamma \in [0, 1)$ is the discount factor.

At time t, the agent in state s_t executes an action a_t by following the policy $\pi : \pi(a|s)$, which is a mapping from states $\mathcal{S}$ to actions $\mathcal{A}$. The system transit from state s_t to the next state $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$. For MDPs, the goal is to find the optimal policy π to maximize $a_t \sim \pi(\cdot|s_t)$ and the accumulated rewards

$$\mathbb{E}\left[\sum_{t\geq 0} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})\middle| a_t \sim \pi(\cdot|s_t), s_0\right].$$

Accordingly, given policy π, for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$, we could define the *action-value function* (the Q-function), which is starting from $(s_0, a_0) = (s, a)$, as

$$Q_\pi(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})\middle| a_t \sim \pi(\cdot|s_t), a_0 = a, s_0 = s\right],$$

and the *state-value function* (the V-function), starting from $s_0 = s$, as

$$V_\pi(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})\middle| a_t \sim \pi(\cdot|s_t), s_0 = s\right]$$

π^* are referred to the optimal policy as the optimal Q-function and V-function respectively. By virtue of the Markov property, the optimal function could be obtained by iteration based on dynamic programming (DP), which is usually required for the complete knowledge of the model.

Value-Based Methods.
The value-based methods are mainly to find the estimate of the optimal Q-function Q^*_π. One famous value-based algorithm is Monte-Carlo tree search (MCTS), which is used under the imcomplete environment knowledge. In this method, a monte carlo simulation is executed based on a search tree to estimate the optimal value function.

Temporal-Difference (TD) learning method is a combination of MCTS and DP. It learns the estimates partially based on estimates, which is known as *bootstrapping*. As model-free methods, TD methods are implemented more naturally than MCTS and DP in an online and fully incremental way.

Q-learning is one of the most important value-based methods, which is actually an off-policy TD control. The optimal policy can be approximated by taking the greedy action of estimation of the Q-value function $\hat{Q}(s, a)$. The Q-function is updated according to

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha\left[r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s,a)\right]$$

with the loss function

$$\mathcal{L}(s, a, r, s') = (r + \gamma \max_{a'} Q(s', a') - Q(s, a))^2.$$

Policy-Based Methods.
The main idea of the policy-based method is to update the parameter by following the gradient direction, which is known as policy gradient (PG). The closed form of PG is given as

$$\nabla J(\theta) = \mathbb{E}_{a\sim\pi_\theta(\cdot|s), s\sim\eta_{\pi_\theta}(\cdot)}\left[\mathcal{Q}_{\pi_\theta}(s,a) \nabla log \pi_\theta(a|s)\right],$$

where $J(\theta)$ and $\mathcal{Q}_{\pi}$ are the expected reward and Q-function with following policy π_{θ}, respectively, $\pi_{\theta}(\cdot|s)$ is the approximation of $\pi(\cdot|s)$, η_{π_θ} is the measurement of state occupancy, and $\nabla log\pi_{\theta}(a|s)$ is the score of the policy.

Compared with value-based methods, policy-based ones are more powerful with better convergence guarantees with neural networks for function approximation, which is a fashion today with the rise of Deep Learning (DL). And policy-based methods are believed to have the ability to handle bigger discrete or even continuous state-action spaces.

Markov Games.
Markov games (MGs), which are also known as stochastic games, is originally a framework for MDP in multi-agent settings. A markov game is defined by a tuple $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{\mathcal{R}^i\}_{i\in\mathcal{N}}, \gamma)$, where $\mathcal{N} = \{1, \cdots, N\}$ denotes the set of N agents, $\mathcal{S}$ denotes the globally observed state space by the whole system, $\mathcal{A}^i$ denotes the action space of agent i, $\mathcal{P} : \mathcal{S} \times \{\mathcal{A} := \mathcal{A}^1 \times \cdots \times \mathcal{A}^N\} \to \Delta(\mathcal{S})$ denotes the transition probability from any state $s \in \mathcal{S}$ to any state $s' \in \mathcal{S}$ denotes the possibility distribution for mapping from any state $s \in \mathcal{S}$ to any state $s' \in \mathcal{S}$ via any joint action $a \in \mathcal{A}$, $\mathcal{R}^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ denotes the immediate reward received by agent i via a transition from (s, a) to s', $\gamma \in [0, 1)$ denotes the discount factor.

From time t to time $t+1$, agent $i \in \mathcal{N}$ executes action a_t^i, the system will transits from s_t to s_{t+1}, and all agents get immediate reward by $R^i(s_t, a_t, s_{t+1})$. For every individual agent i, the goal is to maximize its own reward in a long finite horizon or infinite horizon by finding the optimal policy $\pi^i : \mathcal{S} \to \Delta(\mathcal{A}^i)$ so that $a_t^i \sim \pi^i(\cdot|s_t)$. The joint policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is $\pi(a|s) := \prod_{i\in\mathcal{N}} \pi^i(a^i|s)$. For any state $s \in \mathcal{S}$ and joint policy π,

$$V^i_{\pi^i,\pi^{-i}}(s) := \mathbb{E}\left[\sum_{t\geq 0} \gamma^t R^i(s_t, a_t, s_{t+1}) \middle| a_t^i \sim \pi^i(\cdot|s_t), s_0 = s\right],$$

where $-i$ denotes the indices of all other agents in $\mathcal{N}$ except agent i. A nash equilibrium (NE) of a *markov game* $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{\mathcal{R}^i\}_{i\in\mathcal{N}}, \gamma)$ is a joint policy $\pi^* = (\pi^{1,*}, \cdots, \pi^{N,*})$ so that for any $s \in \mathcal{S}$, $i \in \mathcal{N}$, and π^*

$$V^i_{\pi^{i,*},\pi^{-i,*}}(s) \geq V^i_{\pi^i,\pi^{-i,*}}(s)$$

The nash equilibrium point π^* is a fixed point so that all agents won't transit to a better point as there is not any incentive to do so. For any agent $i \in \mathcal{N}$, $\pi^{i,*}$ is the best response to $\pi^{-i,*}$. For MARL settings, finding the NE is a standard learning goal and NE always exists for finite-space infinite-horizon discounted MGs (Filar and Vrieze 2012).

Cooperative Settings.
In this short paper, we only consider cooperative settings which means all the agents collaborate with each other to achieve shared goals. In a fully cooperative setting, all agents share one reward function $\mathcal{R}^1 = \mathcal{R}^2 = \cdots = \mathcal{R}^N = \mathcal{R}$. With this model, the Q-function is identical to all agents so that Q-learning updates could be applied by taking the max over the joint action space $a' \in \mathcal{A}$. In a more general cooperative setting, agents have their own reward function and the goal is to optimize the long-term reward for all agents. One common reward model is *team-averager* reward $\bar{R}_(s, a, s') := N^{-1} \cdot \sum_{i\in\mathcal{N}} R^i(s, a, s')$ for any $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$.

Partial Observability.
In multi-agent settings, we can not ignore the influence of the real environment, the noise and limited sensors may prevent the agent from observing the state of the environment (Oliehoek and Amato 2016). However, MGs can only handle the fully observed environment. In this case, *Partially Observable Markov Decision Process* (POMDP) is more suitable to represent such state uncertainty by incorporating observations and their probability of occurrence conditional on the state of the environment (Kaelbling, Littman, and Cassandra 1998). More generally speaking, this partially observed setting can be modeled by a decentralized POMDP (Dec-POMDP), which shares most of the elements, including the reward function and the transition model. Based on MGs, a Dec-POMDP is formally defined by the tuple $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{\mathcal{R}^i\}_{i\in\mathcal{N}}, \mathcal{Z}, \mathcal{O}, \gamma)$, where $\mathcal{Z} := \{\mathcal{Z}^1 \times \cdots \times \mathcal{Z}^N\}$ denotes the joint observations, $\mathcal{O} : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$ denotes the observation probabilities, the others denotation remain the same as MGs. Under this setting. At time t, agent i has its *action-observation* history $\mathcal{H}_i := [\mathcal{O}_{i,1}, \mathcal{A}_{i,1}, \cdots \mathcal{O}_{i,t-1}, \mathcal{A}_{i,t-1}]$, $\mathcal{H}_i \in \mathcal{H}$, and a stochastic policy for agent i is $\pi^i(\mathcal{A}^i|\mathcal{H}^i)$. Given the history $\mathcal{H}$, the goal for each agent is to maximize its expected discounted rewards in a long term.

Challenges

The challenges in *Cooperative Multi-Agent Reinforcement Learning* (CMARL) not only lie in problems with the models based on MDP but also in the different training schemes and the lacking of theoretical foundations. In this section, I will cover several ones in this research area.

Non-Stationarity.
In CMARL, one major issue is that the environment for each agent is non-stationary during the learning process as other agents are learning at the same time. One agent's action influences other agents' reward functions and transition functions (Papoudakis et al. 2019). In single-agent RL, the stationarity Markovian property is assumed so that convergence could be guaranteed. However, in CMARL settings, the agents need to model other agents' behavior and adapt to the *joint behavior*. With the invalidation of stationarity assumption in MARL settings, simply transferring single-agent mathematical tools to MARL is inapplicable.

Independent learning might be useful to face the convergance under non-stationarity by taking other agents as part of the environment, but the practice is difficult as independent learning increases the dimensionality of the Q-function, making it infeasible to learn. One key factor is how to design a communication mechanism for agents with low-dimension learning requirements, and another one is how to maintain a generalized form for all agents with different models so that they could fit their *fingerprint* to the same value after policies have converged (Foerster et al. 2017).

Scalability.
In CMARL settings, each agent needs to make a decision based on *joint action space* which grows rapidly with the

increasing number of agents. The curse of dimensionality in MARL leads to exponential computational complexity which is the combinatorial nature of MARL (Hernandez-Leal, Kartal, and Taylor 2019). With a large number of agents, the convergence is complicated to be analyzed and hard to get. One fact is that MARL theories under two-player zero-sum settings are more extensive and advanced than cooperative settings or mixed settings (general-sum) with more than two agents (Zhang, Yang, and Başar 2021).

Scalability is a key challenge not only in MARL but also in general deep neural networks where function approximation is necessary. Though having successful implementations, MARL still needs more research on the theoretical foundations.

Non-Unique Learning Goals.
In many early works on MARL, the learning goals are vague at times(Shoham, Powers, and Grenager 2003). If we take multi-agent settings as repeated matrix games and assume the algorithms will converge in the end, the rational agents will transit to fixed points defined by NE, via perfect self-reasoning and modeling other agents. However, in CMARL settings, agents are with bounded rationality and limited mutual modeling skills so the convergence to NE may not applicable.

Not only the goals to optimize the curriculum reward, learning how to communicate during coordinating with other agents so that they can cooperate better also draws researchers' attention (Foerster et al. 2016). Maintaining communication during learning costs a lot of computation power, and one natural challenge is how to design efficient communication protocols. Over-fitting on certain agents is another concern in CMARL settings (Lowe et al. 2017).

Credit Assignment Problem.
In fully cooperative settings, agents are required to maximize the shared reward. However, it is difficult to determine which agents and actions contribute to the final shared reward when agents do not have access to the joint action. This associating rewards to agents are known as *credit assignment problem*. The sequential nature of reinforcement learning makes the problem harder to solve as the agent is not only required to measure the influence of single action but also the whole action sequences which lead to the final reward. How to efficiently determine the contribution of agents' jointly-shared reward settings is still a key challenge.

Heterogeneous Settings.
Most of the works in MARL, especially CMARL, are under homogeneous settings, meaning that agents in the network have the same skills and dynamic models. However, in real-world applications, we face more general cases under heterogeneous settings. How different agents cooperate with each other to learn a policy to maximize their return is one problem. If we only consider agents' skills are monotonic, for example, in a two-dimensional grid-world multi-agent pathfinding setting, agents' beginning action space could be defined by a tuple $(\mathcal{A}^1, \cdots, \mathcal{A}^N)$, $\mathcal{A}^i \in \{$up, down, left, right, stall$\}$ where $i \in \{1, \cdots, N\}$. Agents could learn more abilities during train so that we may assume the agent's maximized ability as $\mathcal{A}^i \in \{$up, down, left, right, upper-left, upper-right, lower-left, lower-right, stall$\}$. In this case, the challenges also include the non-stationarity which will be more unstable if we consider facing direction in state space and rotation in action space, scalability issue, and over-fitting problems.

Related Work

For current work in CMARL, one goal is to solve some real-world multi-agent problems with partial observability. In this section, I mainly focus on related work with Dec-POMDPs settings.

Centralized and Decentralized Schemes.
For reinforcement learning, there are two kinds of schemes for training and execution, separately. For training, there are centralized training and decentralized training. The centralized training updates the agent policies via shared information. Distributed training allows agents to have their own history and learn independently so there is no information exchange. For centralized execution, agents make decisions based on the joint action space which is from a centralized training unit. While in distributed execution, agents behave by updating their individual policy.

Centralized Training Centralized Execution.
In *centralized training centralized execution (CTCE)* paradigm, there is a centralized executor $\pi : \mathcal{O} \rightarrow \mathcal{P}(\mathcal{A})$ to model the joint policy. It is more likely to take multi-agent problems as single-agent problems. Natural ideas are to transfer some single-agent methods under this paradigm. While due to the *curse of demensionality*, these simple transformations do not work well. By using deep RL in a curriculum learning scheme, it is shown that policy gradient methods tend to outperform the temporal-difference (TD) method and actor-critic (AC) method (Gupta, Egorov, and Kochenderfer 2017). Another approach is to address the "lazy agent" problem which means agents have less incentive to learn if one of them already learned a good policy. This problem is arisen due to partial observability. It shows that value function factorization with weight sharing and information channel has superior results in MARL with partial observability consideration (Sunehag et al. 2017).

Decentralized Training Decentralized Execution.
In *decentralized training decentralized execution (DTDE)* paradigm, each agent has its own policy $\pi^i : \mathcal{O}^i \rightarrow \mathcal{P}(\mathcal{A}^i)$ where $i \in \{1, \cdots, N\}$. Under this paradigm, one agent makes its decision based on its own local observation and do not share information with other agents so it learns independently. The drawbacks under this scheme are non-stationarity and poor scalability with the number of agents. In tabular worlds, it is shown that independent learners learn slower by independent Q-Learning (IQL) algorithm (Tan 1993). This tabular IQL works very well in small-size problems, but not in the case of function approximation, especially deep neural networks (DNNs). One drawback of this tabular method lies in the need for experience replay (ER) to stabilize the training with DNNs. Even ER helps but it is could be a problem under multi-agent settings due to the non-stationarity. To address this issue, one way is to disable the ER part (Foerster et al. 2016). However, this method does reduce the non-stationarity of the environment but limits the sample efficiency. To tackle this problem, researchers pro-

posed two algorithms to stabilize the ER in IQL-type algorithms (Foerster et al. 2017). One is to calculate the *importance sampling correction* by current policy. The original loss function is:

$$\mathcal{L}(\theta_i) = \sum_{j=1}^{b} [(r_i + \gamma \max_{a'_i} Q(s', a'_i; \theta_i^-) - Q(s_i, a_i; \theta_i))^2]$$

where agent i's policy parameter θ_i is learned by sampling batches of b transitions, θ^- is a target network parameter, which is periodically copied from θ_i and is fixed for iterations. And the importance weighted loss function is:

$$\mathcal{L}(\theta_i) = \sum_{j=1}^{b} \frac{\boldsymbol{\pi}_{-i}^{t_r}(\boldsymbol{a}_{-i}|s)}{\boldsymbol{\pi}_{-i}^{t_i}(\boldsymbol{a}_{-i}|s)} [y_i^{DQN} - Q(s, a_i; \theta_i))^2]$$

where $y_i^{DQN} = r_i + \gamma \max_{a'_i} Q(s', a'_i; \theta_i^-)$, t_r is the replay time, t_i is the time of collection of the i-th sample, $-a$ denoted for joint quantities over agents other than a. The other is called *fingerprint*, which is to augment ER with some parts of other agents' policies by iteration number and the ϵ of the ϵ-greedy algorithm.

Among recent works with DTDE scheme, distributed training has inferior performance compared to policies that are trained with centralized scheme (Gupta, Egorov, and Kochenderfer 2017) and independently learning actor-critic methods have slower speed than using centralized learning (Foerster et al. 2018). There are also several works using DTDE scheme in partially observable domains (Nguyen, Kumar, and Lau 2017; Dobbe, Fridovich-Keil, and Tomlin 2017).

Centralized Training Decentralized Execution.
In recent years, there is more research interest in MARL for Dec-POMDP, and most of the work is using *centralized training decentralized execution (CTDE)* scheme that means each agent i holds an individual policy $\pi^i : \mathcal{O}^i \rightarrow \mathcal{P}(\mathcal{A}^i)$ which is the mapping from local observations to individual actions. While only during training, agents are endowed with shared information. This paradigm had been successfully used in value-based methods at first (Oliehoek, Spaan, and Vlassis 2008). And as of late, it is used among most MARL works, from policy-gradient approaches to value-based methods.

Lowe et al. proposed a model-free MARL algorithm called MADDPG by using one trained central critic in a fully cooperative setting. At time t, each agent i accesses its own local observation o_i^t, local action a_i^t, and local rewardsr_i^t, and trains a DDPG algorithm (Silver et al. 2014) so that the actor $\pi_i(o_i^t; \theta_i)$ with policy weight θ_i get the local observations. The centralized critic is based on the observations and actions of all agents. However, as the policy gradient is determined by expectations over the actions of other agents, the convergence of collaboration is not guaranteed at the end of the training. Foerster et al. proposed COMA which uses a *counterfactual baseline* $b(s^t, a_{-i}^t) = \sum_{\hat{a}_i^t} \pi_i(\hat{a}_i^t|o_i^t) Q(s^t, [\hat{a}_i^t, a_{-i}^t])$ that marginalizes out a single agent's action to obtain the contribution of action via advantage function $A(s, a^t) = Q(s, [a_{-i}^t, a_i]) - b(s^t, a_{-i}^t)$. This method helps with credit assignment problems, but the factorizztion is still a downside as the policy gradient for a given agent is not changed.

Sunehag et al. proposed the VDN algorithm to decompose each agent's value function. VDN measures the influence of each agent on the observed joint reward and assumes that the joint action-value function $Q_{joint} = \sum_{i=1}^{N} Q_i(h_i, a_i, \theta_i)$ where $h_i \in \mathcal{H}$, and $\mathcal{H}$ is the joint observation-action history. As the joint Q-function is just a simple summation of single Q-functions, its results are narrowed in this way. One improvement for this limitation is called QMIX (Rashid et al. 2018) which adds monotonic constraint $\frac{\partial Q_{joint}}{\partial Q_i} \geq 0$ to enforce positive weights on the mixing network so that monotonic improvement is guaranteed. One problem in QMIX is that guaranteeing the monotonicity is more restrictive than enforcing the max factorization so a more general factorization method is needed. Son et al. proposed QTRAN to address this issue by setting an alternative joint action-value Q'_{joint} which is assumed to be factorizable by additive decomposition. They built three networks: individual Q_i, Q_{joint}, and joint regularizer V_{joint}, where $V_{joint} = \max_a Q_{joint}(\boldsymbol{\tau}, \boldsymbol{a}) - \sum_{i=1}^{N} Q_i(\tau_i, \bar{a}_i)$ and $\bar{a}_i = arg\max_{a'_i Q_i(\tau_i, a'_i)}$, to demonstrate three related loss functions. But there are also some drawbacks to QTRAN. First, they add the TD loss function for estimating the true action value, but how to choose the other two hyper-parameters in their full loss function is unclear so it is hard to tell how good or bad are their policy if their loss functions fall to satisfy the estimation condition with the loss function. Second, it is still a method based on DQN but not the policy-gradient methods so it is harder to learn as it is not allowed to use stochasticity.

Proposed Methods

The goal of this proposal is to tackle general challenges with new potential approaches that may lead the future research. The problem setting is large-scale CMARL problems with partial observability and long-term dynamic tasks consideration as we want our methods to be used in real-world problems, for example, autonomous driving.

Attention Mechanism for Communication.
The attentional mechanism is widely used in natural language processing to weigh sequential information and reduce redundancy. This mechanism is also used in CMARL and it is not the first time since researchers use gated recurrent units (GRUs) on actors in QMIX and COMA. Independent agents do not share information with each other, while with communication, they could cooperate. In this case, how to exchange information efficiently and necessarily is the key part of multi-agent cooperation. Current work on communication has to maintain communication all the time while adding attention units on actors in a CTDE scheme enables information sharing only when necessary. Also, for large-scale systems, there are several critics, and attention-based parameter sharing could also be among critics so agents could learn more efficiently in a dynamic environment. An advantage function like COMA is also needed to tackle credit assignment problems.

Graph Networks for Multi-Agent Cooperation.
At the beginning of multi-agent systems research, people used a 2-dimensional grid world to do multi-agent planning. From a data structure perspective, this environment could be

modeled as a graph and each agent is a node. Then agents in a CMARL setting with partial observability can be formulated by updating weighted graphs, and the weight on edge is the influence parameter between neighbors. Also, this weight could have directions as we consider heterogeneous agents, which means they have different abilities. Under this setting, we could use a mean-field fashion to update our graphs so that the curriculum reward could reach optimal. Also, as we use graph networks, we could use convolutional networks to extract latent features so that agents could learn how to cooperate. One benefit of graph networks is that they are more straightforward to represent the influence among agents, weights changing, and parameter sharing, especially considering partial observable dynamic environments.

Transfer-Learning for Generalization.
In large-scale multi-agent systems, it is hard to maintain a single critic for all actors and there are cases where agents have to cooperate with unknown agents in different critics, which is also known as the Ad Hoc challenge. So a more generalized learning method is needed and transfer learning could be one of the solutions. If agents have pre-learned knowledge which is usually the initial parameter in deep neural networks, they could also transfer knowledge and used it for upcoming similar agents. This method is widely used in multi-task RL, and can also be used in large-scale multi-agent systems with similar settings so that agents could adapt quickly to non-stationarity.

Conclusion

In this short paper, I presented the basic knowledge of MARL, current key problems, related work in three famous schemes, and drawbacks of notable methods in the CTDE paradigm. As there are lots of directions for future work, I only focus on CMARL with partially observable environment consideration and proposed three possible methods to deal with current challenges with advanced techniques. Although my methods are still at starting places, they are still worth doing as there is likely a wide place to work on. And for future work, centralized training might still be dominant among other methods for distributed executions.

References

Dobbe, R.; Fridovich-Keil, D.; and Tomlin, C. 2017. Fully decentralized policies for multi-agent systems: An information theoretic approach. *Advances in neural information processing systems*, 30.

Filar, J.; and Vrieze, K. 2012. *Competitive Markov decision processes*. Springer Science & Business Media.

Foerster, J.; Assael, I. A.; De Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Foerster, J.; Nardelli, N.; Farquhar, G.; Afouras, T.; Torr, P. H.; Kohli, P.; and Whiteson, S. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *International conference on machine learning*, 1146–1155. PMLR.

Gupta, J. K.; Egorov, M.; and Kochenderfer, M. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International conference on autonomous agents and multiagent systems*, 66–83. Springer.

Hernandez-Leal, P.; Kartal, B.; and Taylor, M. E. 2019. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6): 750–797.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2): 99–134.

Lowe, R.; Wu, Y. I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30.

Nguyen, D. T.; Kumar, A.; and Lau, H. C. 2017. Policy gradient with value function approximation for collective multiagent planning. *Advances in neural information processing systems*, 30.

Oliehoek, F. A.; and Amato, C. 2016. *A concise introduction to decentralized POMDPs*. Springer.

Oliehoek, F. A.; Spaan, M. T.; and Vlassis, N. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32: 289–353.

Papoudakis, G.; Christianos, F.; Rahman, A.; and Albrecht, S. V. 2019. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*.

Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, 4295–4304. PMLR.

Shoham, Y.; Powers, R.; and Grenager, T. 2003. Multi-agent reinforcement learning: a critical survey. Technical report, Technical report, Stanford University.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *International conference on machine learning*, 387–395. PMLR.

Son, K.; Kim, D.; Kang, W. J.; Hostallero, D. E.; and Yi, Y. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, 5887–5896. PMLR.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.

Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 330–337.

Zhang, K.; Yang, Z.; and Başar, T. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, 321–384.