Intelligence & Robotics

Research Article

Open Access

Opponent modeling with trajectory representation clustering

Yongliang Lv, Yan Zheng, Jianye Hao

College of Intelligence and Computing, Tianjin University, Tianjin 300000, China.

Correspondence to: Yongliang Lv, College of Intelligence and Computing, Tianjin University, No. 135 Yaguan Road, Jinnan District, Tianjin 300000, China. E-mail: Lvyongliang@tju.edu.cn

How to cite this article: Lv Y, Zheng Y, Hao J. Opponent modeling with trajectory representation clustering. *Intell Robot* 2022;2(2):168-79. http://dx.doi.org/10.20517/ir.2022.09

Received: 9 Mar 2022 **First Decision:** 25 Apr 2022 **Revised:** 16 May 2022 **Accepted:** 1 Jun 2022 **Published:** 16 Jun 2022

Academic Editor: Simon X. Yang **Copy Editor:** Fanglin Lan **Production Editor:** Fanglin Lan

Abstract

For a non-stationary opponent in a multi-agent environment, traditional methods model the opponent through its complex information to learn one or more optimal response policies. However, the response policy learned earlier is prone to catastrophic forgetting due to data imbalance in the online-updated replay buffer for non-stationary changes of opponent policies. This paper focuses on how to learn new response policies without forgetting old policies that have been learned when the opponent policy is constantly changing. We extract the representation of opponent policies and make explicit clustering distinctions through the contrastive learning autoencoder. With the idea of balancing the replay buffer, we maintain continuous learning of the trajectory data of various opponent policies that have appeared to avoid policy forgetting. Finally, we demonstrate the effectiveness of the method under a classical opponent modeling environment (soccer) and show the clustering effect of different opponent policies.

Keywords: Non-stationary, opponent modeling, contrastive learning, trajectory representation, data balance

1. INTRODUCTION

In the field of multi-agent reinforcement learning (MARL)[1–3], the non-stationary problem[4,5] caused by policy changes of other agents has always been challenging. Since the policy and behavior of other agents are generally unknown when the policies of other agents change, the environment is no longer considermed to be a stationary arkov decision process (MDP), and it cannot be solved by simply using a single-agent reinforcement learning algorithm[6–8]. A common class of ideas is to introduce additional information to aid training by modeling other agents i.e. opponent modeling[4,9].

Opponent modeling is a common idea in the MARL domain, which has many works of different points of view, such as explicitly representing the opponent's policies through neural networks to train some optimal response policies[10–12] or implicitly learning the opponent policy's representation to assist training[13–16]. Since the goal of the agent under our control is to maximize its local reward, other agents are viewed collectively as an opponent, although "opponent" does not always imply a fully competitive environment. However, existing opponent modeling methods, whether explicitly or implicitly, set the opponent to use a fixed policy or switch between fixed policies, which is not suitable for most real-world situations. Therefore, we further set the opponent policy in the form of a probability distribution, so as to learn a general policy that can deal with all kinds of opponents, which requires additional consideration of policy forgetting.

Specifically, when the opponent policy changes, the data in the replay buffer[17] are constantly replaced by the interactive trajectory with the new opponent policy so that the agent's response policy converges to deal with the new opponent policy. However, at the same time, the agent may forget the response policy it has learned before because of the loss of previous interaction data; therefore, it still needs to re-learn when some opponent policies appear again, which greatly reduces the response efficiency.

We believe that the main reason for this type of policy forgetting problem is that there are not enough trajectories of interactions with various opponent policies saved in the replay buffer. Thus, this paper uses the idea of data balancing[18,19] to ensure the diversity of trajectories interacting with various opponent policies in the replay buffer as much as possible. Data balancing is widely used in continuous learning[20] to solve catastrophic forgetting problems. In contrast, in most continuous learning settings, task IDs are given to distinguish between different tasks, but we do not know the types of opponent policies. Thus, to distinguish various trajectories, we self-supervise extracted policy representations from interactive trajectories by contrastive learning[21–24] and clustering at the representational level. Our proposed method, trajectory representation clustering (TRC), can be combined with any existing reinforcement learning (RL) algorithm, to avoid policy forgetting in non-stationary multi-agent environments.

The contributions of this paper can be summarized as follows: (1) Interaction trajectories are self-supervised encoded through a contrastive learning algorithm so that different opponent policies can be more accurately represented and distinguished in the representation space. No additional information is required except the opponent observation; (2) From the perspective of balancing data types, we artificially retain the types of data that account for a small proportion in the replay buffer to avoid catastrophic policy forgetting.

The rest of this paper is organized as follows. The related work on opponent modeling and contrastive learning is discussed in Section 2. Section 3 details the used network architecture, loss function, and algorithm flow. Then, some experiments based on the classic environment of soccer are presented to verify the performance of our method in Section 4. Finally, the conclusions and future work are introduced in Section 5.

2. RELATED WORK

2.1. Opponent modeling

Opponent modeling stems from a naive motivation that infers the opponent's policy and behavior through the information about the opponent to obtain a higher reward for itself. Early opponent modeling work[25,26] mainly focused on simple game scenarios where the opponent policy is fixed. With the development of deep reinforcement learning, scholars have begun to apply the idea of opponent modeling in more complex environments and settings. The following introduces the opponent modeling work in recent years in terms of explicit modeling and implicit modeling.

2.1.1. Implicit opponent modeling

Implicit opponent modeling generally refers to extracting representations from opponent information to aid training. He *et al.* first used the opponent's observation and agent's observation as merged input in a deep network to train the agent end-to-end. They also pointed out that information such as the opponent's policy type can be used to assist the training of RL[13]. Subsequently, Hong *et al.* additionally used the information of opponent action, fitted the opponent policy through the neural network, and then multiplied the output of the hidden layer of the opponent's policy network with the output of the hidden layer of the Q network to calculate the Q value[14]. Considering that the opponent may also have learning behaviors, Foerster *et al.* maximized the agent's reward by estimating the parameters of the opponent policy network based on the idea of recurrent reasoning[16]. Raileanu *et al.* considered the parameters of the opponent policy network from another perspective and used the agent policy to make decisions based on the opponent observation, so as to infer the opponent's goal and achieve better performance[15]. Due to the different assumptions about the opponent, the effects of different algorithms are also difficult to compare.

2.1.2. Explicit opponent modeling

Explicit opponent modeling generally refers to explicitly modeling opponent policies, dividing opponent types, and detecting and responding online during the interaction process. Rosman *et al.* first proposed Bayes policy reuse (BPR) to be used in multi-task learning, maintaining a belief for each task through Bayesian formula, judging the task type, and choosing the optimal response policy for unknown tasks[27]. Since then, Hernandez-Leal *et al.* extended the environment to a multi-agent system, used MDP to model opponents, and added a detection mechanism for unknown opponent policies[10] . In the face of more complex environments, Zheng *et al.* used neural networks to model opponents and the rectified belief model (RBM) to make opponent detection more accurate and rapid, as well as policy distillation technology to reduce the scale of the network[11]. On this basis, Yang *et al.* introduced the theory of mind[28] to defeat opponents with higher-level decision-making methods for opponents who also use opponent modeling method[12].

2.2. Contrastive learning

Contrastive learning, as the most popular self-supervised learning algorithm in recent years, is different from generative encoding algorithms. Contrastive learning focuses on learning common features between similar instances and distinguishing differences between non-similar instances. van den Oord *et al.* first proposed InfoNCE loss, which encodes time-series data. By separating positive and negative samples, it can extract data-specific representations[21]. Based on similar ideas, He *et al.* achieved high performance in the field of image classification, by improving the similarity between the query vector and its corresponding key vector while reducing the similarity with the key vector of other images[23]. From the perspective of data augmentation, Chen *et al.* performed random cropping, inversion, grayscale, and other transformations on the image and extracted the invariant representation behind the image through contrastive learning[22]. The subsequent series of works[29–31] continued with a series of improvements, and the performance on some tasks is close to that of supervised learning algorithms.

From the above works, we can see that most of the previous opponent modeling work is to additionally input representations into neural networks for policy training. This paper provides another perspective on training a general policy to respond to various opponents by balancing the data in the replay buffer interacting with different opponent policies. Through the powerful representation extraction ability of contrastive learning, we distinguish various opponent policies at the representation level. It is worth noting that we only additionally use opponent observations, which is a looser setting compared to other work in multi-agent settings.

3. METHOD

3.1. Problem formulation

We describe the problem as a partially-observable stochastic game (POSG)[32] composed of a finite set $\mathcal{I} = \{1, 2, \ldots, N\}$, a state space $\mathcal{S}$, the joint action space $\mathcal{A} = \mathcal{A}^1 \times \ldots \times \mathcal{A}^N$, the joint observation space $\mathcal{O} = \mathcal{O}^1 \times \ldots \times \mathcal{O}^N$, a transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ denoting the transition probabilities between two states when given a joint action, and a reward function $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ for each agent $i \in \mathcal{I}$. Since we only focus on the performance of the agent under our control, we denote this agent by 1 and other agents are denoted by -1 with joint observation o^{-1} and joint action a^{-1}.

We design a set of K fixed policies for agent -1 $\Pi = \left\{\pi^{-1,k} \mid k = 1, \ldots, K\right\}$, which can be rule-based (artificial) or network (pre-trained). Assume that an opponent policy π^w is a probability distribution on Π, $w \in \Delta(\Pi)$, which means at the beginning of each episode, the agent -1 samples a policy from Π with probability distribution w and executes this policy throughout the episode. Different from the setting of only switching between fixed policies in previous work, we allow more complex opponent policy changes, making the setting looser and more general.

Ideally, our goal is to find a general response policy π_θ parameterized by θ, which can maximize the reward of agent 1 against each opponent policy π^w. However, considering the reality, we maximize the minimum reward of π_θ against π^w:

$$\max_{\theta} \min_{w \in \Delta(\Pi)} \mathbb{E}_{\pi_\theta, \pi^w} \left[\sum_{t=0}^{H-1} \gamma^t r_t^1 \right] \tag{1}$$

where r_t^1 is the reward of agent 1 at step t after performing the action a_t^1 determined by π_θ, H is the horizon of episode, and $\gamma \in (0, 1)$ is a discount factor. It is also important to note that at any time the agent 1 knows neither the policy type k of opponent nor the distribution w. This gives us the motivation to infer the type of opponent policy.

3.2. Representation extraction module

Different from previous opponent modeling methods that model opponent policies, we use a contrastive learning approach to self-supervised distinguish trajectories against different opponent policies, so that we only use the opponent's observations. We denote trajectory as $\tau = \{o_t^1, o_t^{-1}\}_{t=0}^{t=H-1}$ where o_t^1 and o_t^{-1} are the observations of agent 1 and agent -1 at step t, respectively. Given a set of trajectories $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_M\}$, the representation of each trajectory is self-supervised extracted by the CPC[21] algorithm.

Figure 1 shows the architecture of the contrastive predictive coding algorithm. First, we encode the observations by a multi-layer perceptron (MLP) to get a sequence of latent representation z_t:

$$z_t = f_{\mathrm{MLP}}(o_t^1, o_t^{-1}) \tag{2}$$

Then, use a gated recurrent unit (GRU) to extract the context information for the first t steps:

$$c_t = f_{\mathrm{GRU}}(z_{:t}) \tag{3}$$

In addition, we also need to define the similarity function f_k. To unify the dimensions, we use a bilinear product function:

$$f_k(z_{t+k}, c_t) = z_{t+k}^T W_k c_t \tag{4}$$

where $k = 1, \ldots, H - t - 1$ and W_k is different for each k. For given set of trajectory $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_M\}$, c_t^i and z_{t+k}^i are calculated from τ_i. Since representations extracted from the same trajectory have similarities, we

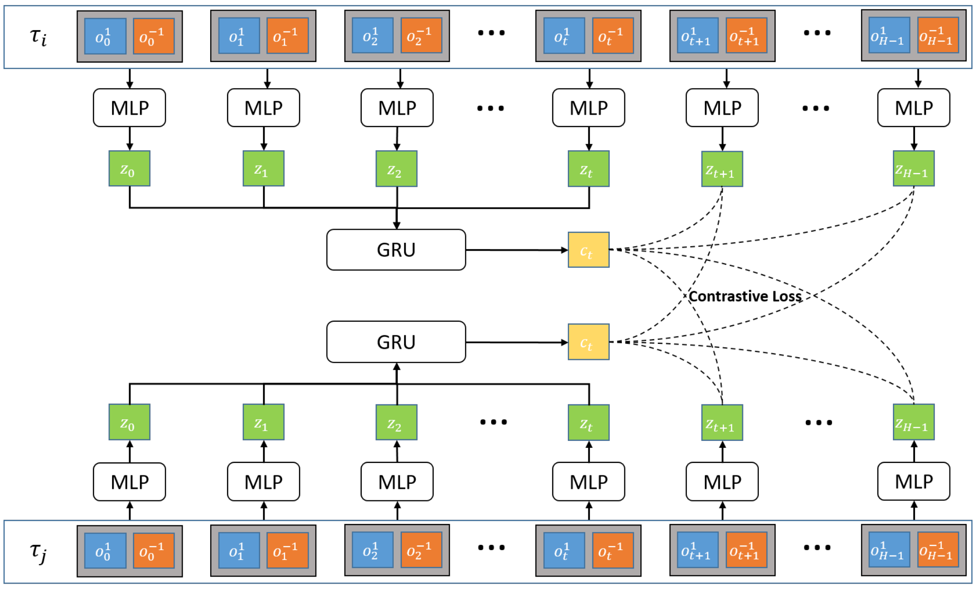

Figure 1. Overview of contrastive predictive coding (CPC), a representation extraction algorithm by contrasting positive and negative samples.The context c_t and subsequent state embeddings $\{z_{t+1}, z_{t+2}, \ldots, z_{H-1}\}$ are regarded as positive samples when they come from the same trajectory; otherwise, they are regarded as negative samples. By increasing the similarity between positive samples and reducing the similarity between negative samples, we obtain trajectory representations to distinguish different opponent policies.

maximize $f_k(z_{t+k}^i, c_t^j)$ when $i = j$ and minimize $f_k(z_{t+k}^i, c_t^j)$ when $i \neq j$. The InfoNCE loss is:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{M(H-t-1)} \sum_{k=1}^{H-t-1} \sum_{i=1}^{M} \left[\log \frac{exp(f_k(z_{t+k}^i, c_t^i))}{\sum_{j=1}^{M} exp(f_k(z_{t+k}^j, c_t^i))} \right] \tag{5}$$

where t is random sampling within a suitable range, M is the size of the trajectory set (batch size), and H is the horizon. Optimizing this loss will extract a unique representation of each trajectory that is different from others.

As described above, we self-supervise the extraction of policy representations from trajectories through contrastive learning, which can discriminate different opponent policies in representation space. Especially the contrast between positive and negative samples makes the representation highlight the differences between trajectories, which is beneficial for subsequent clustering operations.

3.3. Experience replay module

In Section 3.2, we introduce how to extract the representations of opponent policies from the trajectories that interact with opponents. Different from the previous approaches of directly using representations to assist training, we focus on another aspect, that is, the impact of non-stationary opponents on the experience replay. Experience replay is a commonly used method in reinforcement learning whose purpose is to improve the sample efficiency. When the replay buffer is full, the data are usually processed in a first-in, first-out (FIFO) manner. When the opponent uses a fixed policy, the environment can be treated as a deterministic MDP, and FIFO is feasible. When the opponent is non-stationary, the replay buffer will pass through data that interacts with different types of opponent policies. The decrease in the proportion of certain types of data will affect the effectiveness against such an opponent, and the loss of old data may lead to the forgetting of learned strategies. Therefore, we design new data in and out, a mechanism to keep as many types of trajectory data as possible in the replay buffer.

We cluster the trajectory data in the replay buffer in the representation space, and, for the representation, $z^i_{:H}$ and $z^j_{:H}$ of the trajectories τ_i and τ_j, we use the average Euclidean distance to measure the distance between them:

$$d_{i,j} = \frac{1}{H} \sum_{h=0}^{H-1} ||z^i_h, z^j_h|| \tag{6}$$

For all trajectories in the replay buffer, we can calculate the representation distance matrix D by Equation (6). Additionally, the truncation method can be used for trajectory representations of different lengths, or the dynamic time warping (DTW) can be used instead of the Euclidean distance.

Since the number of opponent policies is unknown, some clustering methods such as K-means are not suitable for use. We use agglomerative clustering to distinguish trajectory representations in the replay buffer, which is implemented in the standard algorithm library scikit-learn, and the clustering threshold is set as the average distance of all trajectory representations. Then, the labels of the trajectories that interact with the opponents are obtained in a self-supervised manner.

To balance the proportion of different types of data in the replay buffer, we no longer pop the oldest data when the replay buffer is full, but pop the oldest data from the largest class based on the clustering results. This ensures the dynamic balance of various types of data to a certain extent. Even if a certain type of opponent policy has a very low probability of appearing in a period, the data interacting with it can maintain a certain proportion in the replay buffer, thereby avoiding policy forgetting. However, this approach will lead to some useless old data existing in the replay buffer for a long time, reducing the training effect of reinforcement learning. We introduce a probability threshold ρ, where the replay buffer pops the oldest data from the largest class with the probability of ρ and pops the oldest data from the entire replay buffer with the probability of $1 - \rho$. This allows the data that hinder training to be popped. In this paper, we set $\rho = 0.9$.

3.4. Combine with reinforcement learning

This section describes the overall algorithm flow in combination with the classic reinforcement learning algorithm soft actor–critic (SAC) whose optimization goal is:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot \mid \mathbf{s}_t\right)\right) \right] \tag{7}$$

where $\mathcal{H}\left(\pi\left(\cdot \mid \mathbf{s}_t\right)\right)$ is the additional policy entropy added to encourage exploration and α is the temperature parameter determines the relative importance of the entropy term. However, our method can be combined with any off-policy reinforcement learning algorithm.

Since the training speed of representation learning is much faster than that of reinforcement learning, we set the training frequency F_c for it to balance their learning rate. In addition, this also considers that the flow of data in the replay buffer in the short term will not change the data distribution in it. Considering that the introduction of the clustering requires a large amount of extra computation, and the data that newly entered replay buffer will not be popped in the short term, we update the labels of trajectory representations by clustering every F_l episode.

The complete algorithm is described in Algorithm 1. The training of representation learning and reinforcement learning process alternately. Since the FIFO rule is still followed in the class, our method will not have too much influence on the training of reinforcement learning; at the same time, the diversity of the data in the replay buffer is guaranteed as much as possible, so that the policy forgetting caused by the non-stationary of the opponent policy is avoided.

Algorithm 1 SAC with TRC.

```
Require: Initialize SAC parameter vector $\theta$, CPC parameter vector $\varphi$, total episode $T$, episode horizon $H$,
    batch size $M$, CPC training frequency $F_c$, labels update frequency $F_l$, and threshold $\rho$.
1:  for episode $i = 0 \dots T-1$ do
2:      opponent choose policy $\pi^{-1}$
3:      for step $t = 0 \dots H-1$ do
4:          $a_t^1 \sim \pi_\theta \left(a_t^1 \mid o_t^1, o_t^{-1}\right)$
5:          $a_t^{-1} \sim \pi^{-1} \left(a_t^{-1} \mid o_t^1, o_t^{-1}\right)$
6:          $o_{t+1}^1, o_{t+1}^{-1} \sim p \left(o_{t+1}^1, o_{t+1}^{-1}, \mid o_t^1, o_t^{-1}, a_t^1, a_t^{-1}\right)$
7:          $\tau_i \leftarrow \tau_i \cup \{(o_t^1, o_t^{-1}, a_t^1, r(o_t^1, o_t^{-1}, a_t^1, a_t^{-1}), o_{t+1}^1, o_{t+1}^{-1})\}$
8:      end for
9:      $D \leftarrow D \cup \tau_i$
10:     if $i \bmod F_c == 0$ then
11:         Sample trajectory batch $\mathcal{T}$ from $D$
12:         Update $\varphi$ by Equation (5)
13:     end if
14:     if $|D| == M$ then
15:         if random sample a probability value greater than $\rho$ then
16:             Pop the oldest trajectory from $D$
17:         else
18:             if $i \bmod F_l == 0$ then
19:                 Compute $z_{:H} = f_\varphi(\tau)$ for each $\tau$ in $D$
20:                 Compute distance matrix of trajectory representations by Equation (6)
21:                 Cluster trajectory representations by agglomerative clustering
22:             end if
23:             Pop the oldest trajectory from the largest class
24:         end if
25:     end if
26:     Update $\theta$ by SAC algorithm.
27: end for
```

4. RESULTS

We evaluate our approach in a more complex soccer environment and compare the average returns during RL training against three baselines. We also discuss the impact of the proportion of data in the replay buffer on reinforcement learning training and the improvement of our approach to the diversity of trajectories in the replay buffer. In addition, we analyze representational clustering by t-distributed stochastic neighbor embedding (t-SNE) to analyze the properties of different adversary policies at the representational level.

4.1. Game description

Soccer is a classic competitive environment that has been used by many opponent modeling approaches[11,13] to verify their performance. We extend the rules based on the classic soccer environment and design more complex rule-based opponent policies based on this. As shown in Figure 2, the environment is a 15×15 grid world, and there are two goals on each end line. At the beginning of the episode, the two agents are in the center of their respective end lines with 0 energy, and one random agent holds the ball. Each agent has 13 optional actions, moving to any of the 12 grid points within a two-grid range around itself or staying in place, but moving 2 grids requires 2 energy. The agent with the ball recovers 0.5 energy per step, while the agent without the ball recovers 1 energy per step, and the upper limit of energy is 2. When both agents are about to enter the same grid, they stop in place and exchange the ball possession. When the agent dribbles the ball

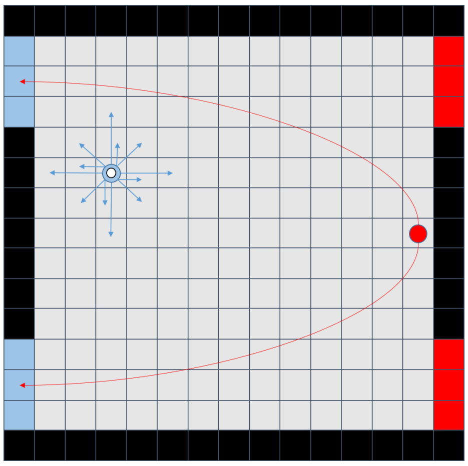

Figure 2. The configuration of soccer. The goal of each agent is to drive the ball into the opponent's goal.

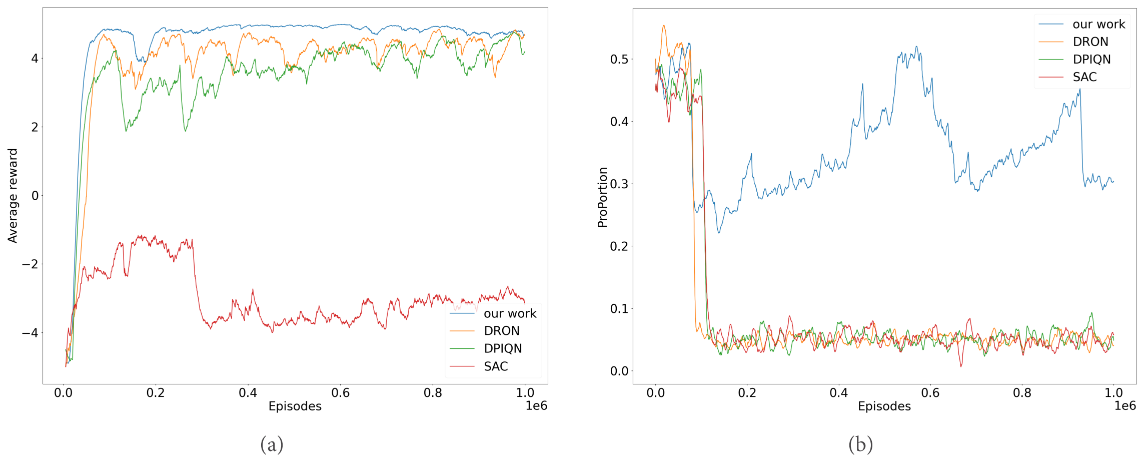

Figure 3. (a) The average reward curve of interacting with opponent policy π_1^{-1}; and (b) the proportion change curve of opponent π_1^{-1} trajectory in replay buffer.

into the opponent's goal, it gets a +5 reward, while the opponent gets a -5 reward, and then ends this episode. If the interaction exceeds 50 steps, the episode will also be terminated and each agent will get 0 rewards. The position, energy, and ball possession are fed back to the agent as observation.

The opponent policies are designed to be random policies based on given rules, which makes it more complex. Specifically, we design two base opponent policies π_1^{-1} and π_2^{-1} with different styles. π_1^{-1}: Keep away from the opponent while attacking the upper goal when holding the ball, and get close to the opponent when not holding the ball. π_2^{-1}: Keep away from the opponent while attacking the lower goal when holding the ball, and defend near its end line when not holding the ball. As described in Section 3.1, we define $w \in [0, 1]$ as a class of opponent policies, and, at the beginning of the episode, the opponent chooses a policy from $\{\pi_1^{-1}, \pi_2^{-1}\}$ with probability distribution $\{w, 1-w\}$.

4.2. Non-stationary opponent

We make the opponent policy switch from $w = 0.5$ to $w = 0.05$ at step 100k to observe the performance of the agent training by different algorithms in a non-stationary environment. Figure 3a shows the comparison of the reward curves of our algorithm and three baselines against opponent policy π_1^{-1}. In these baselines, vanilla

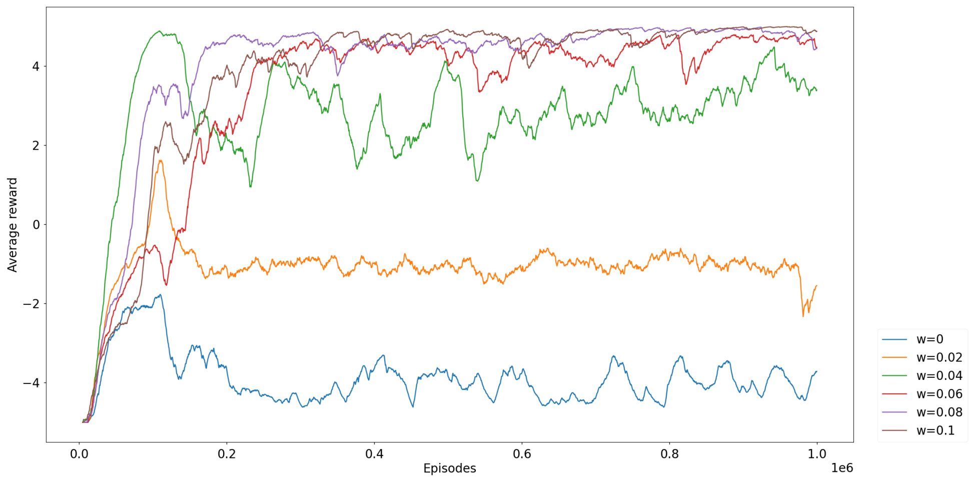

Figure 4. The average reward curve of interacting with opponent policy π_1^{-1} when w change from 0.5 to 0, 0.02, 0.04, 0.06, 0.08, and 0.1.

SAC uses no opponent information and performs the worst. DRON uses the opponent's observation as an additional input, while DPIQN further uses the opponent's actions to obtain the representations of opponent policy to aid training. However, they both perform worse than our work due to a lack of consideration of data balance. Figure 3b shows the change in the proportion of interaction trajectories with opponent strategy π_1^{-1} in the replay buffer. It can be seen that, when the probability of an opponent policy decreases, only our method can maintain a relatively high proportion of the data obtained by interacting with it in the replay buffer. This improves responsiveness to such an opponent policy and avoids forgetting the learned policy.

To explain the impact of data ratio on policy forgetting in more detail, we make the opponent policy change from $w = 0.5$ to $w = 0, 0.02, 0.04, 0.06, 0.08, 0.1$ at step 100k and use SAC for training with the other conditions remaining the same. As shown in Figure 4, when a certain opponent policy appears very infrequently, a small proportional increase in the replay buffer can bring about higher performance improvement, but, for data that already exist in significant proportions, the impact of adjusting the data ratio is minimal. This also explains our motivation to balance the proportion of various data.

4.3. Analysis of clustering

In Section 4.2, we show the performance of the algorithm and analyze the rationale behind data balancing. In this section, from the perspective of policy representation, we analyze the clustering properties of the policy representations obtained by contrastive learning in the representation space. Figure 5 shows the visualization of trajectory encoding after dimensionality reduction by t-SNE. Self-supervised contrastive learning is not very accurate in distinguishing two types of opponent policies. Because policies may have similar parts, a type of policy can also be decomposed into several more refined sub-policies. Self-supervised learning of policy representations only with trajectory information can only be used for coarse clustering. However, our algorithm does not rely on extremely accurate trajectory clustering and strategy identification but balances the proportion of various trajectory data generally. This also makes the algorithm have certain robustness.

5. CONCLUSION

This paper constructs a general sampling algorithm based on data balance for multi-agent non-stationary problems. The trajectory representation of the interaction with the opponent is extracted by comparative learning, and then the representation is distinguished by hierarchical clustering. Finally, the data balance in the replay buffer is realized by changing the order of in and out of the replay buffer. We get better performance against

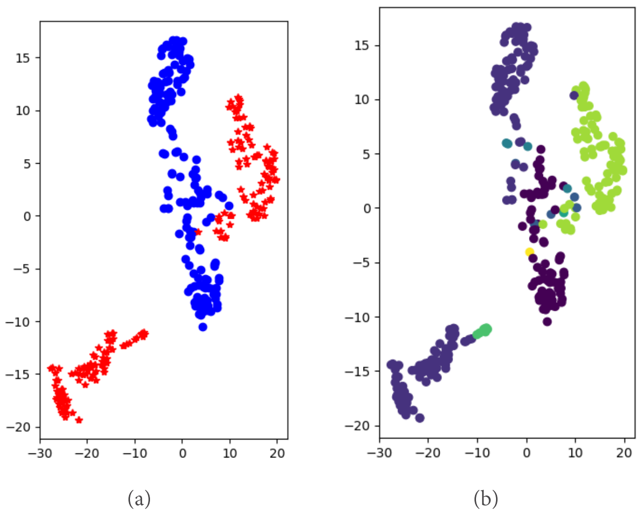

Figure 5. t-SNE projection of the embeddings in the soccer environment: (a) the two colors represent the two base opponent policies π_1^{-1} and π_2^{-1}; and (b) the different colors represent the classes of trajectory representations encoded by the contrastive learning.

a non-stationary opponent. In particular, we only use the observation information of the opponent, and the setting is looser than other opponent modeling algorithms. In the future, we would like to combine multi-task learning algorithms to learn different opponent policies as different tasks and explore more efficient ways to distinguish opponent policies.

DECLARATIONS

Authors' contributions

Designed and run experiments: Lv Y
Made substantial contributions to conception and design of the study: Zheng Y
Provided administrative, technical, and material support: Hao J

Availability of data and materials

Not applicable.

Financial support and sponsorship

None.

Conflicts of interest

All authors declared that they have no conflicts of interest to this work.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

REFERENCES

1. Littman ML. Markov games as a framework for multi-agent reinforcement learning. Machine Learning Proceedings 1994. Elsevier; 1994. pp. 157-63. DOI
2. Busoniu L, Babuska R, De Schutter B. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans Syst, Man, Cybern C* 2008;38:156-72. DOI
3. Vinyals O, Babuschkin I, Czarnecki WM, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 2019;575:350-4. DOI
4. Hernandez-Leal P, Kaisers M, Baarslag T, de Cote EM. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017. DOI
5. Papoudakis G, Christianos F, Rahman A, Albrecht SV. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*, 2019. DOI
6. Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. DOI
7. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. DOI
8. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning, PMLR, 2018. p. 1861-70.
9. Hernandez-leal P, Kartal B, Taylor ME. A survey and critique of multiagent deep reinforcement learning. *Auton Agent Multi-Agent Syst* 2019;33:750-97. DOI
10. Hernandez-Leal P, Taylor ME, Rosman B, et al. Identifying and tracking switching, non-stationary opponents: A bayesian approach. In: Workshops at the Thirtieth AAAI Conference on Artificial Intelligence, 2016.
11. Zheng Y, Meng Z, Hao J, et al. A deep bayesian policy reuse approach against non-stationary agents. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018. p. 962–972.
12. Yang T, Hao J, Meng Z, et al. Towards efficient detection and optimal response against sophisticated opponents. In: Twenty-Eighth International Joint Conference on Artificial Intelligence, 2019.
13. He H, Boyd-Graber J, Kwok K, Daumé III H. Opponent modeling in deep reinforcement learning. In: International Conference on Machine Learning, PMLR, 2016. p. 1804-13. DOI
14. Hong ZW, Su SY, Shann TY, Chang YH, Lee CY. A deep policy inference q-network for multi-agent systems. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, 2018. p. 1388-.96.
15. Raileanu R, Denton E, Szlam A, Fergus R. Modeling others using oneself in multi-agent reinforcement learning. In: International Conference on Machine Learning, PMLR, 2018. p. 4257-66.
16. Foerster J, Chen RY, Al-Shedivat M, Whiteson S, Abbeel P, Mordatch I. Learning with opponent-learning awareness. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, 2018. p. 122-30.
17. Lin LJ. Reinforcement learning for robots using neural networks. Carnegie Mellon University, 1992. Available from: https://dl.acm.org/doi/book/10.5555/168871 [Last accessed on 7 Jun 2022]
18. Chaudhry A, Rohrbach M, Elhoseiny M, et al. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019. DOI
19. Chrysakis A, Moens MF. Online continual learning from imbalanced data. In: International Conference on Machine Learning, PMLR, 2020. p. 1952-61.
20. Khetarpal K, Riemer M, Rish I, Precup D. Towards continual reinforcement learning: a review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020. DOI
21. van den Oord A, Li Y, Vinyals O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. DOI
22. Chen T, Kornblith S, Norouzi M, Hinton G. A simple framework for contrastive learning of visual representations. In: International conference on machine learning, PMLR, 2020. p. 1597–1607.
23. He K, Fan H, Wu Y, Xie S, Girshick R. Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. p. 9729-38.
24. Laskin M, Srinivas A, Abbeel P. Curl: Contrastive unsupervised representations for reinforcement learning. In: International Conference on Machine Learning, PMLR, 2020. p. 5639-50.
25. Koopmans T. Activity analysis of production and allocation. 1951. Available from: https://cowles.yale.edu/sites/default/files/files/pub/mon/m13-all.pdf [Last accessed on 7 Jun 2022]
26. Carmel D, Markovitch S. Model-based learning of interaction strategies in multi-agent systems. *Journal of Experimental & Theoretical Artificial Intelligence*1998;10:309-32. DOI
27. Rosman B, Hawasly M, Ramamoorthy S. Bayesian policy reuse. *Mach Learn* 2016;104:99-127. DOI
28. de Weerd H, Verbrugge R, Verheij B. How much does it help to know what she knows you know? an agent-based simulation study. *Artificial Intelligence* 2013;199-200:67-92. DOI
29. Chen X, Fan H, Girshick R, He K. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. DOI
30. Chen T, Kornblith S, Swersky K, Norouzi M, Hinton GE. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems* 2020;33:22243-55.
31. Grill JB, Strub F, Altché F, et al. Bootstrap your own latent - a new approach to self-supervised learning. *Advances in Neural Information*

Processing Systems 2020;33:21271-84.
32. Hansen EA, Bernstein DS, Zilberstein S. Dynamic programming for partially observable stochastic games. *AAAI* 2004;4:709-15. Available from: https://www.aaai.org/Papers/AAAI/2004/AAAI04-112.pdf [Last accessed on 7 Jun 2022]