2019 IEEE Symposium Series on Computational Intelligence (SSCI)
December 6-9 2019, Xiamen, China

Multi-Agent Reinforcement Learning Based on Clustering in Two-Player Games

Weifan Li, Yuanheng Zhuand Dongbin Zhao
State Key Laboratory of Management and Control for Complex Systems
Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
Email: liweifan2018@ia.ac.cn; yuanheng.zhu@ia.ac.cn; dongbin.zhao@ia.ac.cn

***Abstract*—Non-stationary environment is general in real environment, including adversarial environment and multi-agent problem. Multi-agent environment is a typical non-stationary environment. Each agent of the shared environment must learn a efficient interaction for maximizing the expected reward. Independent reinforcement learning (InRL) is the simplest form in which each agent treats other agents as part of environment. In this paper, we present Max-Mean-Learning-Win-or-Learn-Fast (MML-WoLF), which is an independent on-policy learning algorithm based on reinforcement clustering. A variational auto-encoder method based on reinforcement learning is proposed to extract features for unsupervised clustering. Based on clustering results, MML-WoLF uses statistics and the dominated factor to calculate the values of the states that belong to a certain category. The agent policy is iteratively updated by the value. We apply our algorithm to multi-agent problems including matrix-game, grid world, and continuous world game. The clustering results are able to show the strategies distribution under the agent's current policy. The experiment results suggest that our method significantly improves average performance over other independent learning algorithms in multi-agent problems.**

Keywords*—*reinforcement learning; unsupervised clustering; matrix game; multi-agent.

I. Introduction

In Multi-Agent Reinforcement Learning (MARL), each agent expects to learn a mixed strategy that is able to maximize its return in the repeated games. Due to the change and exploration of agent strategies, the multi-agent environment is non-stationary. One kind of MARL algorithms is independent RL (InRL), where each agent treats other agents' policies as part of its environment [1] [2] [3]. The agent needs to have ability of generalization to learn the equilibrium strategy without any other agents' information. However, this setting of local environments is non-stationary and non-Markovian since other agents may keep changing their policies [4]. It further leads to the overfitting of agent policy with respect to other's policies [5]. Some MARL algorithms address the challenges of multi-agent problem by adding opponent information to avoid imperfect information [6] [7] [8] [9]. While other algorithms use a centralised critic to estimate advantage function or baselines such as counterfactual [10], VDN [11] and Q-MIX [12]. The main idea is to modify the action-state value by comparing with others. In a non-stationary environment, one agent performance can be illustrated by the advantage function. Sampling while keeping the agent's policy fixed, the algorithm calculates the contribution of the current state by comparing the return with other returns at other states. According to the contribution value, the agent increases the probability of state with the maximum contribution [13] [14]. However, in a model-free environment, the relationship between states is hard to define. States under the same strategy should be treated as one category, while states of different strategies should be compared with each other.

This work was supported in part by the National Key Research and Development Program of China under Grants 2018AAA0101005 and 2018AAA0102404, and was also supported in part by Noahs Ark Lab, Huawei Technologies.

There are several reinforcement learning algorithms use the cluster to simplify the state space and maximize the probability of categories that obtain the maximum return [15] [16]. However, they focus on clustering state into discrete categories by the return from a stationary environment. If the environment is non-stationary such as multi-agent environment, it's hard to classify states by their value, because values are affected by the policy of other agents. In other words, there is an optimal distribution and classification of the state to maximize the expected return. Another problem is that mixed strategies are often difficult to interpret, and the distribution of states by a mixed strategy is difficult to be correlated to the agent's strategy. There are several methods to explain the strategy from latent code clustering results [17] [18]. These algorithms mainly focus on explaining the relationship between the latent variables of neural network and states, or the distribution of behavior and latent variables. Besides, there are few methods to obtain the specific number or distribution of strategies in a training agent.

In this paper, a new algorithm called Max-Mean-Learning-Win-or-Learn-Fast (MML-WoLF) is proposed based on cluster and statistics to solve the two-player game. MML-WoLF has a fixed probability distribution of sampling policy. The unsupervised clustering algorithm optimizes the distribution of state to maximize the expected return. The policy uses an actor-critic structure, in which the *actor* is trained along the gradient estimated by the *critic*. Based on the results of the clustering algorithm, the maximum return of the state will be compared with the mean return of other states to obtain the state contribution. The contribution of the state can be back-propagated to tune the policy. The clustering result is able to

Authorized licensed use limited to: INSTITUTE OF AUTOMATION CAS. Downloaded on June 27,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.

show the approximate number of strategies.

Our contributions can be summarized as follows:

- A RL algorithm based on clustering is proposed (Section III-A). The algorithm produces clusters that potentially indicate the number of strategies.
- We design a novel InRL algorithm for multi-agent problems based on clustering and iteratively eliminated strictly dominated strategy (IESDS) to improve the efficiency of learning (Section III-B).
- Our average performance exceeds other multi-agent InRL algorithms in matrix-game and multi-step game (Section IV).

II. Background and Related Work

A. Markov Decision Process and Reinforcement Learning

A Markov Decision Process (MDP) can be represented by four elements and denoted as a tuple as $< S, A, R, T >$: state space S, action space A, expected payoff R and transition function T, where $R : S \times A \rightarrow \Re$ is the payoff function. $R(s, a)$ is the expected payoff for taking action a in state s and $T(s, a, s')$ is the probability of reaching state s' when given action a is taken at state s. At each step t of MDP, the agent chooses an action according to a probability distribution which affected by a policy (aka. strategy) over available actions. To improve agent's performance, agent have to maximize its cumulative return $G_t = \sum_{t=t'}^{T} \gamma^t R_t$, where t is the step and $\gamma \in [0, 1)$ is a discount factor. A common objective is to iteratively estimate the action-value function $Q_i(s, a) = \mathbb{E}^{\pi}[G_t|S_t = s, A_t = a]$, the value function $V_\pi(s_t) = \sum_{a\in A}\pi(s_t, a)Q(s_t, a)$ and the advantage function $A_\pi(s_t, a_t) = Q_\pi(s_t, a) - V_\pi(s_t)$. The parameter θ_π of agent's policy is updated by gradient $g = \mathbb{E}[\sum_{t=0}^{T} A_\pi(s_t, a_t)\nabla_{\theta_\pi} \log \pi(a_t|s_t)]$. The actor is trained by a gradient that depends on a critic, which usually estimates a value function. In particular, R_t is replaced by advantage function $A_\pi(s_t, a_t)$ to reduce variance. Another option is to replace with the Temporal Difference (TD) error $r_t + \gamma V(s_{t+1}) - V(s)$, which is an unbiased estimate of $A_\pi(s_t, a_t)$.

Proximal Policy Optimization (PPO) [19] is an on-policy reinforcement learning algorithm which limits the update step size by replacing advantage function with $\min(\rho_t A(x_t, a_t; \theta_{old}), clip(\rho_t, 1 - \varepsilon, 1 + \varepsilon)A(x_t, a_t; \theta_{old}))$ where ρ_t is a probability ratio $\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, and $\varepsilon > 0$ is a hyper-parameter that determines the threshold.

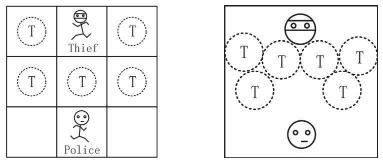
B. Multi-agent Extensive-form Games

The process of including multiple agents in a sequential interaction is called an extensive-form games. In imperfect information games, each agent samples an action according to a probability distribution over available actions based on its local observation states. For convenience, the collection of all possible strategies is $\Pi = (\pi_1, \cdots, \pi_n)$ and π_{-i} represents all strategies in Π except π_i. Any strategy of agent i that achieves worst payoff performance against fixed strategy profile π_{-i} is a strictly dominated strategy which should be iteratively eliminated. A best response is any strategy that achieves optimal performance against fixed strategy profile π_{-i}. If a sub-optimal strategy approximates best response by no more than ζ, the sub-optimal is called ζ best response. An equilibrium in a repeated game is called Nash equilibrium if no agent can gain from deviating. A strategy set is a Nash equilibrium if the strategy of all the agent is the best response to the other strategies. Similarly, an approximate of ζ Nash equilibrium is a set of ζ best response. Fig. 1(a) shows examples of single-step matrix games, the Zero-Sum game, the Shapley's Game and Chasing Game. The Chasing Game has one Nash equilibrium $(0.5, 0, 0.5)$ with a dominated strategy that needs to be iteratively eliminated. Choosing the middle is a strictly dominated strategy for the thief. When the thief eliminates choosing middle, choosing the middle is a strictly dominated strategy for the police. We upgrade the matrix game to a multi-step extensive-form game as shown in Figs. 1(b) and 1(c). The payoff matrix determines the reward of the locations marked with T.

Zero Sum — Shapley's Game — Chasing Game

$$\begin{bmatrix} 0,0 & 0,0 & 0,0 \\ 0,0 & 1,-1 & -1,1 \\ 0,0 & -1,1 & 1,-1 \end{bmatrix} \quad \begin{bmatrix} 0,0 & 1,0 & 0,1 \\ 0,1 & 0,0 & 1,0 \\ 1,0 & 0,1 & 0,0 \end{bmatrix} \quad \begin{bmatrix} 1,0 & 0,1 & 0,1 \\ 0,0 & 1,0 & 0,0 \\ 0,1 & 0,1 & 1,0 \end{bmatrix}$$

(a)



(b) (c)

Fig. 1. Example of matrix-game (a) multi-step game on grid world (b) and continuous game (c).

Serval self-play MARL algorithms find equilibrium strategies without the knowledge of other players' actions or policy information [20]. These include GIGA-WoLF [21], PGA-APP [22], WPL [23], WoLF-PHC [24] and EMA-QL [25]. These algorithms use techniques, including minimize-regret, win-or-learning-fast, adaptive learning rates, policy prediction, and fictitious self-play (FSP) [26] to find Nash equilibrium. In order to reduce the cyclical learning dynamics arises in the gradient ascent, the WoLF-PHC introduces the win-or-learn-fast principle. GIGA-WoLF contains two policies, one of which is updated faster than the other to the convergence and has properties of no-regret and convergence. PGA-APP uses policy prediction to augment the basic gradient ascent. With a variable learning rate, WPL ensure the approach of agent to the equilibrium strategy. EMA-QL uses two learning principle with different learning speeds. WPL and EMA-QL are able to converge in Shapley's Game and Biased Game but failed to learn in simpler games with the requirement of iterative eliminating strictly dominated strategies (IESDS). Since these algorithms are derived from tabular Q-learning,

Authorized licensed use limited to: INSTITUTE OF AUTOMATION CAS. Downloaded on June 27,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.

they cannot handle the high-dimensional environments. Other rapidly developed multi-agent algorithms are based on meta-strategy learning, including policy-space response oracles [27], and Neural Fictitious Self-Play (NFSP) [28]. These methods are also based on efficient algorithms to solve the best response.

C. Reinforcement Learning with Clustering

In reinforcement learning with state aggregation, the number of categories is set to separate the space of return to reduce state dimension. There are several state aggregation methods include Bayesian clustering [29], K-means cluster [30], and value-based supervised learning [31]. Depending on the expected reward, the state is classified into a certain category or a probability distribution over categories. This is a discrete estimate of the value function and the state function. Other similar algorithms estimate the transition function to get the probability distribution of the V or Q value, such as C51 [32], IQN [33], APE-X DQN [34].

Density-based-spatial-clustering of applications with noise (DBSCAN) is the most widely used density-based clustering method (DBSCAN) [35]. It is a density-based clustering non-parametric algorithm based on establishing relationships between points within a certain distance threshold ε. The complexity of DBSCAN is quite low because it performs a linear range query on the database and only needs to be run once. Combining the above two points, DBSC is very suitable for the state or strategy clustering in reinforcement learning. DBSCAN classifies sample points $\mathbf{S}'$ as three different types: core points, density-reachable points and outliers for clustering. A core point s_p are directly reachable from at least minPts points within distance ε. A point s_q is directly reachable from s_p if $N_\varepsilon(s_p) = \{s_q \in \mathbf{S}' | dist(s_p, s_q) \geq N_\varepsilon\}$. If there is a path $s_{p+1}, \cdots, s_{p+n}$ to reach point $s_{p+n} = s_q$ from $s_{p+1} = p$, and each s_{p+k+1} is directly reachable from s_{p+k}, the point s_q is a reachable point. All points not reachable from any other points are outliers or noise points.

III. Proposed Algorithm

In this Section , we propose our reinforcement clustering algorithm MML-WoLF, which combines the DBSC and value-based RL. We incorporate reinforcement clustering with the Bellman Equation and describe the basic idea of reinforcement clustering. Then we explain the MML-WoLF method.

A. Reinforcement Clustering

Suppose there is a classifier C that divides the state space into n categories. Each category c_i has a sampling probability $p(c_i)$. The probability of the state s in category i is $q_i(s)$. The value of current state can be described as follow:

$$V^{p,q}(s) = \sum_i \sum_{s'} p(c_i) q_i(s)((R(s,a,s')) + \gamma V^{p,q}(s')) \quad (1)$$

The probability of reaching the state s_k can be described as $P(s_k) = \sum_i p(c_i) q_i(s_k)$. It's obvious that value of state can be increased by adjusting the sampling probability $p(c_i)$ of category, or increasing the expected return of category. Let the sampling probability of category p_i be fixed as $\hat{p}_i$. The probability of an agent taking an action a_k that can reach state s_k can be described as $\pi(a|s) = \sum_i \hat{p}(c_i) q_i(s_k)$. The optimal value of the state can be defined by optimal probability distribution of the state $q_i^*(s_k)$.

$$V^{\hat{p},q^*}(s) = \sum_i \sum_{s'} \hat{p}(c_i) q_i^*(s')((R(s,a,s')) + \gamma V^{\hat{p},q^*}(s')) \quad (2)$$

Suppose the classifier C changes the q from q^0 to q^1. The change in action probability of reaching state s_k is as follow

$$g(s_k) = P^1(s_k) - P^0(s_k) = \sum_i \hat{p}_i (q_i^1(s_k) - q_i^0(s_k)) \quad (3)$$

Suppose there is an optimal state s^*. To increase the value of each category, the probability of reaching low value state s_{low} should be decreased. The updated probability distribution should make $g(s_{low}) \leq 0$ and $g(s^*) \geq 0$. When the state distribution tends to be optimal, the agent strategy converges.

In model-free environment, the agent has no knowledge of the number of categories. Futhermore, the relationship of states is nonlinear and determined by environment. Therefore, we can only use an unsupervised learning algorithm to classify the state without any prior information and parameters. We choose VAE neural network with encoder E, decoder D and latent variable z for measuring the distance between states. The encoder extracts the features of the state and expresses them by latent variable z. The clustering results are obtained by the clustering of latent variables z. By statistics of the clustering results, the current probability of the category p_i can be calculated. The agent's policy is updated to minimize the difference between p_i and $\hat{p}_i$, which consequently impacts the $g(s)$.

The result of clustering depends on the encoder E. To optimize E, we consider the encoder as an agent and optimization process as a MDP. The MDP can be described as a tuple $< s, z, r_e >$, where the state s of the encoder is the observation of the agent, the action is the latent variable z, and the reward r_e is the different between expected return of the next sampling batch and current sampling batch.

B. Max Mean Learning Win or Learn Fast (MML-WoLF)

In our algorithm, the latent variables z are defined as a Gaussian distributions composed of mean μ and variance σ. The encoder is updated by PPO. When the agent's policy needs to be updated, a certain feature variable of the observation is sampled from the gaussian distribution of the encoder output. We introduce DBSC algorithm to unsupervised clustering. DBSC is chosen becasue it is a non-parametric algorithm based on distance between connecting points and certain distance tresholds ε. In a reinforcement learning environment, each state is a true sample of the environment. There is no need to consider the optimization of denoise. The distance between each sample is defined by the encoder. Finally, the complexity is fairly low, and the result of clustering is essentially the same results in each run. The loss function of the encoder is

Authorized licensed use limited to: INSTITUTE OF AUTOMATION CAS. Downloaded on June 27,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.

described as follow:

$$\mathcal{L}_E = \alpha_{\text{vae}}(s_t - D(z_t))^2 - \alpha_{rl} r_e(s_t) + \alpha_{\text{num}} C_{\text{num}} \tag{4}$$

where α_{vae} is the hyper-parameter of the mean square error, and α_{rl} is the hyper-parameter of reinforcement learning, C_{num} is the total number of categories in current clustering results, and α_{num} is the hyper-parameter. The gradient of the MSE function is used to initialize the encoder. In the training process, the gradient of MSE function is optional. The constraint term $\alpha_{\text{num}} C_{\text{num}}$ limits the number of clustering results. If $\mathcal{L}_E$ is zero, the maximum number of categories c^*_{num} can be calculated by $\alpha_{num} - \alpha_{rl} r_e(s_t) = G^{mean}(c^*_{num}) P(c^*_{num})$. If the category is strictly related to the policy, (c^*_{num}) approximates the number of the strategies.

Before updating the agent's policy, we need to define the fixed sampling probability for each category because the result of DBSC has no label information and the number of clusters is changing. To update the encoder and agent, we define a sampling and updating rule to ensure that the sampling rate of high-return categories increases. First, the initial sampling rate is equal to the ratio of total return of the category to the total return of the sampling batch. But this initial sampling rate is not greedy enough to eliminate the sub-optimal categories with expected return greater than zero. Second, we propose the counter-category factor to eliminate the sub-optimal categories. If the maximum return of a category is lower than mean return of another, the category is a dominated category because increasing probability of sampling another category will increase the total expected return. Therefore, we set the value of the dominated category to zero by dominated parameter λ_i.

$$\lambda_i = \begin{cases} 0 & \min(G^{\max}(c_i) - G^{\text{mean}}(\mathbf{c}_{-i})) < 0 \\ 1 & \min(G^{\max}(c_i) - G^{\text{mean}}(\mathbf{c}_{-i})) \geq 0 \end{cases} \tag{5}$$

The $\mathbf{c}_{-i}$ is any other categories except i. The difference between current probability of category and fixed sampling probability can be described as,

$$G^+(c_i) = \lambda_i G^{\text{mean}}(c_i) - P(c_i) \sum_i \lambda_i G^{\text{mean}}(\mathbf{c_i})) \tag{6}$$

where $P(c_i)$ is the current probability of the agent reaching the states in the category i. To prevent the zero value from being denominator, multiply both sides of the equation by the sum of the category values with dominated factor. The maximum value of category and the G^+ are back-propagated to the state s of the previous layer with γ as,

$$\begin{aligned} G(s) &= R(s, a, s') + \gamma G^+(s') \\ G^m(s) &= R(s, a, s') + \gamma G^{\max}(s') \end{aligned} \tag{7}$$

where G^m is temporary maximum return of the state. The temporary maximum return is used to prevent previous state from being eliminated. To ensure convergence, we introduce the WoLF mechanism to reduce the rotation force in self-play. The gradient for updating agent's policy can be described as,

$$g = \sum_{t=0}^{T} max(\alpha G^+(s_t), G^+(s_t)) \nabla_{\theta_\pi} \log \pi(a_t|s_t) \tag{8}$$

We train agent with an actor-critic method. The critic $f^c(s_t, \theta^c)$ minimizes the loss $\mathcal{L}_t(\theta^c) = (G^+(s_t) - f^c(s_t, \theta^c))^2$ and reduces the variance during policy updates.

IV. Experiment Analysis

In this section, we apply the algorithm to the matrix-game, multi-step game, and continuous game. The payoff matrix in multi-step games is shown in the Table I. Column elements represent the reward when the agent is at this location, the first number in the element represents the case where both agents are at the same location, and the second number represents the case where the agents are not at the same location. The observation is the agent's own coordinates. In grid world, the agent can choose up, down, left, right, and stop. In continuous world, the agent can adjust the angle and speed. If the agent tries to move out of the map, the action will be ignored. There are 16 possible reward locations in the continuous problem, and 5 possible rewards in the grid world. To show the effect of iterative elimination, we design experiments with 8 dominated location that could not be completely eliminated due to sampling probability setting. The agents receive rewards at the end of the game. The reason for choosing this environment is that their Nash equilibrium solutions and the number of strategies can be accurately obtained.

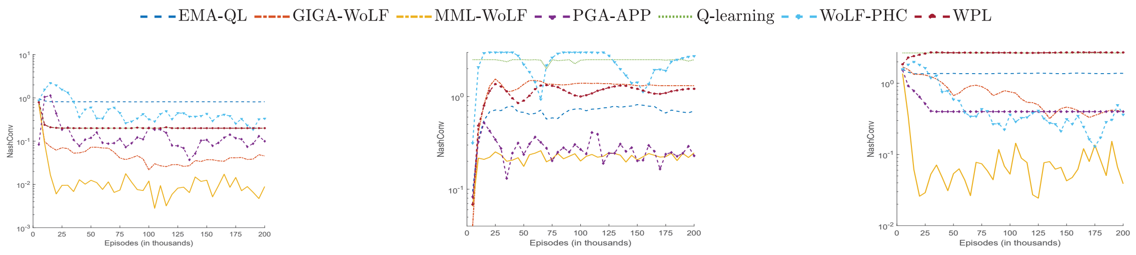
We compare our algorithm with EMA-QL, PGA-APP, GIGA-WoLF and WPL, which also work well in the multi-agent environment. We use the same hyper-parameters for all algorithms. In matrix-game, we set the learning rate $\eta = 0.01$, future reward discount factor $\gamma = 0.9$, the policy learning rate $\eta_\pi = \frac{\eta}{100 + step/2000}$, the wining policy learning rate $\eta_w = \eta_\pi$, and the losing policy learning rate $\eta_l = 2\eta_w$. The agent and cluster's PPO clip ratio chooses $\varepsilon = 0.1$. The cluster's hyper-parameters are set to $\alpha_{\text{num}} = 1e-4$,$\alpha_{rl} = 1$,$\alpha_{\text{vae}} = 1e-2$. The neural network has a hidden layer of 18 nodes, the cluster learning rate and policy learning rate are $\eta = 10^{-4}$. Weights are initialized with the Xavier initializer, and optimized with the RMSprop optimizer. Although the MML-WoLF update requires multiple samples, the number of interactions with environment are the same as other algorithms.

Fig. 2 demonstrates the average cumulative error between the probability of the strategy and the exact Nash equilibrium solution. The result is calculated from the average value of three data with the best performance. Each cumulative error is accumulated by all the agent's error. For each method, we plot the average cumulative statistic error of 1000 samples. The results show that MML-WoLF is superior to the GIGA-WoLF, EMA-QL and WPL in all scenarios. If there is a dominated strategy which needs to be iteratively eliminated in matrix game, EMA-QL will hardly converge to Nash equilibrium because EMA-QL increases the probability of any other actions that also include the dominated action. WPL algorithm is susceptible to exploration of the opponent's policy and eventually converges to the wrong solution. The WoLF-PHC and GIGA-WoLF are able to converge to equilibrium strategy in Chasing Game, but the approximate error of probability

Authorized licensed use limited to: INSTITUTE OF AUTOMATION CAS. Downloaded on June 27,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.

TABLE. I. Example of Multi-step Payoff Matrix

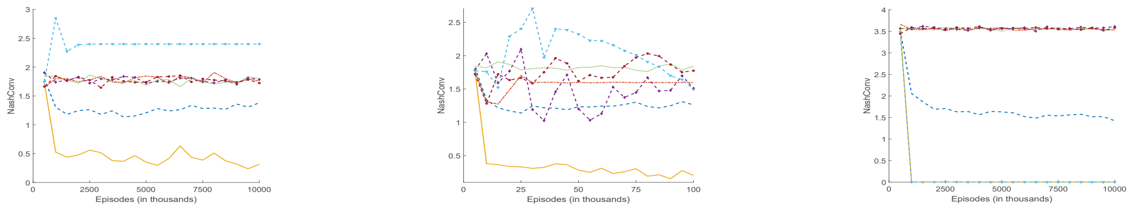
Name	Player1 reward	Player2 reward
Chasing Game	$1,0 \quad \ldots \quad 1,0 \quad 0,0 \quad \ldots \quad 0,0$	$0,1 \quad \ldots \quad 0,1 \quad 0,1 \quad \ldots \quad 0,1$
Cooperative Game	$n,0 \quad \ldots \quad 1,0 \quad 0,0 \quad \ldots \quad 0,0$	$n,0 \quad \ldots \quad 1,0 \quad 0,0 \quad \ldots \quad 0,0$



(a) Zero-Sum

(b) Shapley's Game

(c) Chasing Game



(d) Multi-Step Zero-Sum

(e) Multi-Step Chasing Game

(f) Multi-Step Cooperative Game

Fig. 2. The error between agent decision probability and Nash equilibrium solution in four different scenarios. The algorithms except MML-WoLF prefer reward location near the starting position, which results in the failure of reaching a Nash equilibrium.

TABLE. II. Thief's States Probability Distribution in The Multi-step Chasing Game

Location	1	2	3	4	5	6	7	8	9	Nash Error
EMA-QL	0.277	0.220	0.226	0.071	0.078	0.073	0.021	0.021	0.013	0.756
GIGA-WoLF	0.342	0.476	0.173	0.008	0	0.001	0	0	0	1.236
PGA-APP	0	0.094	0.166	0.329	0.300	0.111	0	0	0	0.645
Q-learning	0.182	0.203	0.177	0.042	0.102	0.101	0.044	0.071	0.078	0.792
WoLF-PHC	0	0.287	0.112	0.101	0.332	0.168	0	0	0	0.838
WPL	0.059	0.256	0.528	0.02	0.043	0.086	0.002	0	0.006	1.202
MML-WoLF	0.211	0.046	0.185	0.180	0.185	0.193	0	0	0	**0.114**
Ground truth	0.2	0	0.2	0.2	0.2	0.2	0	0	0	

is greater than the MML-WoLF under the same number of training steps.

Table II shows the thief's states probability distribution in the multi-step Chasing Game. At the beginning of training, the reward locations 4 and 6 are far from the starting location. Their sampling probability will be diluted by other reward location. Most algorithms fail in converging to the Nash equilibrium in such case. EMA-QL, GIGA-WoLF, WPL prefer to stay in the reward location that is close to the start location instead of going further to get the maximum global expected reward. This leads to an equilibrium solution rather than a global Nash equilibrium in zero-sum game. The MML-WoLF considers the relationship between state probability and the expected return. If the state probability is less than it's reward ratio, MML-WoLF increases its probability by G^+. The counter-categories factor makes sure the agent won't increase the probability of dominated strategy. Fig. 2(f) shows the results of multi-step Cooperative Game, in which the convergence speed of WoLF-PHC is slightly faster than that of MML-WoLF. The main reason is the G^+ of dominated strategy in the MML-WoLF is iteratively decreased with the reduction of both agent's strategy.

Fig. 3 demonstrates the results of the reinforcement clustering and MML-WoLF in continuous environment. Images above show the result of reinforcement clustering without counter-category factor. Images below are the result of MML-WoLF. Fig. 3(a) shows the number of clustering result is 9 when the agent cannot eliminate dominated strategy. Fig. 3(h) shows the average number of the clustering results is 16 in the environment without dominated states. The initial number of clusters is affected by the variance of the VAE output. Both numbers of clusters are close to the number of reward locations. Figs. 3(b) and 3(f) show the error between Nash equilibrium and the strategy. Figs. 3(c) and 3(d) show the distribution of states in the training process of the reinforcement clustering. The coordinates in the figure are not

Authorized licensed use limited to: INSTITUTE OF AUTOMATION CAS. Downloaded on June 27,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.

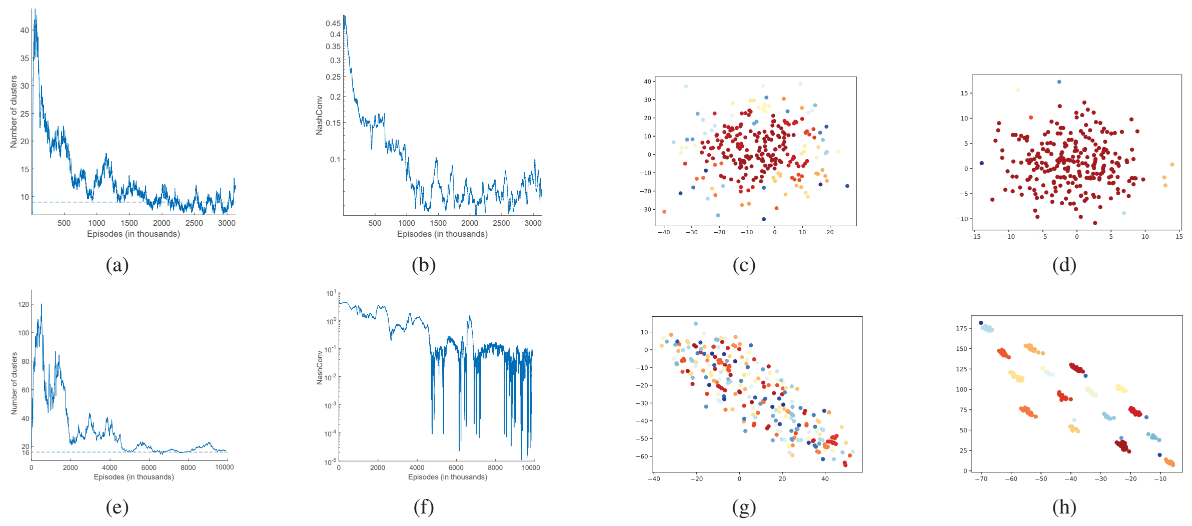


Fig. 3. The results of reinforcement clustering without iterative eliminated: (a) The current number of clusters of states. (b) The error between the probability of states and Nash equilibrium solution. (c) The clustering results at initial. (d) The clustering results after training. The results of MML-WoLF. (e) The current number of clusters of states. (f) The Error between the probability of states and Nash equilibrium solution. (g) The clustering results at initial. (h) The clustering results after training.

related to the actual coordinates. If cluster cannot eliminate the dominated strategy, the distribution of clustering result is bunched together to minimize the probability of sampling dominated strategy. With the improvement of clustering accuracy, the NashConv gradually converges. If the dominated strategy can be eliminated by agent, the clustering result is clear as shown in Fig. 3(h). Because the MML-WoLF can reduce the effect of exploration probability by eliminating the single dominated states, the clustering results can significantly and stably approximate the number of strategies.

V. Conclusion

In this paper, we propose MML-WoLF, a reinforcement learning algorithm based on clustering. We define a sampling rules and make the agent policy satisfy the sampling distribution. Then we train an unsupervised cluster to find the optimal distribution to maximize expected return of the agent. To reduce the effect of the dominated strategy, MML-WoLF uses the counter-category factor to eliminate the dominated categories. MML-WoLF significantly improves the final performance and training speed over other independent multi-agent reinforcement learning algorithms. The results of the clustering approximate the number of the strategies.

References

[1] K. Shao, Y. Zhu, and D. Zhao, "Starcraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 1, pp. 73–84, 2018.

[2] Y. Zhu and D. Zhao, "Vision-based control in the open racing car simulator with deep and reinforcement learning," *Journal of Ambient Intelligence and Humanized Computing*, Sep 2019. [Online]. Available: https://doi.org/10.1007/s12652-019-01503-y

[3] Y. Zhu, D. Zhao, and H. He, "Invariant adaptive dynamic programming for discrete-time optimal control," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2019.

[4] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[5] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, "A study on overfitting in deep reinforcement learning," *arXiv preprint arXiv:1804.06893*, 2018.

[6] A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra, "Learning cooperative visual dialog agents with deep reinforcement learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2951–2960.

[7] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2137–2145.

[8] K. Shao, Y. Zhu, and D. Zhao, "Cooperative reinforcement learning for multiple units combat in starcraft," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2017, pp. 1–6.

[9] Q. Zhang, D. Zhao, and F. L. Lewis, "Model-free reinforcement learning for fully cooperative multi-agent graphical games," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–6.

[10] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[11] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, *et al.*, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 2085–2087.

[12] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," *arXiv preprint arXiv:1803.11485*, 2018.

[13] S. P. Singh, T. Jaakkola, and M. I. Jordan, "Reinforcement learning with soft state aggregation," in *Advances in Neural Information Processing Systems*, 1995, pp. 361–368.

[14] Z. Zhang, D. Wang, D. Zhao, and T. Song, "FMR-GA–A cooperative multi-agent reinforcement learning algorithm based on gradient ascent," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 840–848.

[15] R. Jonschkowski and O. Brock, "Learning state representations with robotic priors," *Autonomous Robots*, vol. 39, no. 3, pp. 407–428, 2015.

[16] J. Asmuth, L. Li, M. L. Littman, A. Nouri, and D. Wingate, "A

Authorized licensed use limited to: INSTITUTE OF AUTOMATION CAS. Downloaded on June 27,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.

bayesian sampling approach to exploration in reinforcement learning," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 19–26.

[17] T. Zahavy, N. Ben-Zrihem, and S. Mannor, "Graying the black box: Understanding DQNs," in *International Conference on Machine Learning*, 2016, pp. 1899–1908.

[18] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," in *Advances in Neural Information Processing Systems*, 2017, pp. 3812–3822.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[20] D. Simões, N. Lau, and L. P. Reis, "Mixed-policy asynchronous deep Q-learning," in *Iberian Robotics conference*. Springer, 2017, pp. 129–140.

[21] M. Bowling, "Convergence and no-regret in multiagent learning," in *Advances in Neural Information Processing Systems*, 2005, pp. 209–216.

[22] C. Zhang and V. Lesser, "Multi-agent learning with policy prediction," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[23] S. Abdallah and V. Lesser, "A multiagent reinforcement learning algorithm with non-linear dynamics," *Journal of Artificial Intelligence Research*, vol. 33, pp. 521–549, 2008.

[24] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, vol. 136, no. 2, pp. 215–250, 2002.

[25] M. D. Awheda and H. M. Schwartz, "Exponential moving average Q-learning algorithm," in *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2013, pp. 31–38.

[26] J. Heinrich, M. Lanctot, and D. Silver, "Fictitious self-play in extensive-form games," in *International Conference on Machine Learning*, 2015, pp. 805–813.

[27] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel, "A unified game-theoretic approach to multiagent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4190–4203.

[28] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," *arXiv preprint arXiv:1603.01121*, 2016.

[29] T. Mandel, Y.-E. Liu, E. Brunskill, and Z. Popovic, "Efficient bayesian clustering for reinforcement learning." in *IJCAI*, 2016, pp. 1830–1838.

[30] S. Mannor, I. Menache, A. Hoze, and U. Klein, "Dynamic abstraction in reinforcement learning via clustering," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 71.

[31] R. Akrour, F. Veiga, J. Peters, and G. Neumann, "Regularizing reinforcement learning with state abstraction," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 534–539.

[32] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 449–458.

[33] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," *arXiv preprint arXiv:1806.06923*, 2018.

[34] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, "Distributed prioritized experience replay," *arXiv preprint arXiv:1803.00933*, 2018.

[35] M. Ester, H.-P. Kriegel, J. Sander, X. Xu*, et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

Authorized licensed use limited to: INSTITUTE OF AUTOMATION CAS. Downloaded on June 27,2023 at 07:36:42 UTC from IEEE Xplore. Restrictions apply.