SparseMAAC: Sparse Attention for Multi-agent Reinforcement Learning

Wenhao Li, Bo Jin(✉), and Xiangfeng Wang

Shanghai Key Lab for Trustworthy Computing,
School of Computer Science and Software Engineering,
East China Normal University, Shanghai, China
{bjin,xfwang}@sei.ecnu.edu.cn

Abstract. In multi-agent scenario, each agent needs to aware other agents' information as well as the environment to improve the performance of reinforcement learning methods. However, as the increasing of the agent number, this procedure becomes significantly complicated and ambitious in order to prominently improve efficiency. We introduce the sparse attention mechanism into multi-agent reinforcement learning framework and propose a novel Multi-Agent Sparse Attention Actor Critic (SparseMAAC) algorithm. Our algorithm framework enables the ability to efficiently select and focus on those critical impact agents in early training stages, while eliminates data noise simultaneously. The experimental results show that the proposed SparseMAAC algorithm not only exceeds those baseline algorithms in the reward performance, but also is superior to them significantly in the convergence speed.

Keywords: Multi-agent deep reinforcement learning · Sparse attention mechanism · Actor-attention-critic

1 Introduction

1.1 Multi-agent Deep Reinforcement Learning

Reinforcement learning is one of the most popular research area in academia and industry. Its goal is to maximize the cumulative feedback while the agent interacts with the environment, as a result the agent can guarantee an optimal strategy. The biggest limitation of traditional reinforcement learning is the demand to manually design features to model the state of environment, which significantly increases the difficulty of extending to sophisticated task. With the development of deep learning, representational learning in complex tasks has become achievable, and deep reinforcement learning has recently achieved remarkable results in games [19], robotics [5], and autonomous driving [2].

Our work focuses on the multi-agent reinforcement learning (MARL) [1]. Actually in many practical problems, there are multiple agents that need to controlled more intelligently, e.g., multi-robot control [13], multiplayer games

G. Li et al. (Eds.): DASFAA 2019, LNCS 11448, pp. 96–110, 2019.
https://doi.org/10.1007/978-3-030-18590-9_7

[20] and etc. These agents always affect each other while completing their own targets at the same time, so that the single agent reinforcement learning algorithms can not be directly applied to multi-agent scenario [22]. The process of state transition in multi-agent system can be principally described as: At a certain time step, the environment receives the joint actions of all agents, further moves to the next state with reward returned from each agent. Multi-agent problems can be divided into two categories according to the relationship between agents, i.e., collaboration problem and competition problem [1]. In the competition problem, the agents of different teams are independent of each other, that is, they have their own reward functions. The goal of each team is to only maximize the cumulative rewards belong to themselves. On the contrary, in collaboration problem, the goal of all agents is no longer to maximize the cumulative rewards they receive individually, but to maximize whole return of all agents in groups. Furthermore in general, all agents share the same value function (objective) in the collaboration problem.

1.2 Attention Mechanism in Multi-agent Reinforcement Learning

The attention mechanism automatically extract the semantic information with respect to each task prior information through an end-to-end manner. This prominent advantage of attention mechanism has been greatly extracted in recent years and it has many successful applications in the field of computer vision [27], natural language processing [11,25] and even reinforcement learning [6,7,18].

The Multiple Actor-Attention-Critic algorithm (MAAC) [6] is a typical attention mechanism based multi-agent reinforcement learning method. MAAC learns the multi-agent system through one centralized critic and many separated decentralized actors. With the purpose to solve the limitations of both traditional value function method and strategy gradient method on multi-agent problems, MAAC borrowed the basic idea of Multi-Agent Deep Deterministic Policy Gradient method (MADDPG) [10], which allows introducing extra agent information in the training process. Further in details, the critic is augmented with extra information about the policies of other agents. As the increasing of agents number, the information of each agent which needs to process or communicate in order to make decision is significantly growing. If each agent considers the behaviors of all other agents, the decision-making learning procedure becomes extremely more difficult. For a large-scale multi-agent system, each agent's decision is not affected by all other agents. Furthermore, it will inevitably make the useful signal submerged in the background noise once all other agents are equally taken into account. The MAAC algorithm introduced the attention mechanism to address this problem, by sharing an attention mechanism which selects relevant information for each agent at every timestep.

However, MAAC still take advantages of all the agents although the utilization level is considered to increase accuracy and efficiency. In practical applications, the tasks performed by different agents maybe quite different. Therefore,

in the process of learning, each agent needs to select the related agents while filter out the independent ones. This selection scheme should not only focus on the degree judgement but more importantly on agent picking. In order to solve this problem, we combine the sparse selection technique with classical MAAC algorithm and propose a multi-agent sparse attention actor critic algorithm (Sparse-MAAC) for multi-agent reinforcement learning problem. Our main contributions are summarized as follows:

1. We introduce the sparse attention mechanism into the multi-agent reinforcement learning combined with MAAC algorithm, which enables our algorithm to quickly filter out the useful parts from the received information in the complex environment and eliminate the noise data. This allows the algorithm to have a faster convergence or the ability to jump out of a local optimal solution. Through the guaranteed sparse attention weights, the established system can be more interpretable.
2. Based on the related work [16], we introduce the hyperparameters of the control algorithm sparsity of sparsemax [12], as a result prior environment information can be easily introduced to our algorithm framework in order to acclimate varied complexity environments.
3. To better verify the effects of sparse attention, we design a more complex collaborative environment called *Grouped Cooperative Treasure Collection* (GCTC), and conduct related and significative experiments performance.

The rest of the paper is organized as follows. In Sects. 2 and 3, we discuss related work and backgrounds, followed by a detailed description of our approach in Sect. 4. We report experimental results in Sect. 5 and conclude our paper in Sect. 6.

2 Related Work

2.1 Sparse Attention Mechanism

The attention mechanism has been widely applied in deep learning in recent years because of its automatically output driven learning performance. The attention mechanism was first successfully applied in [27] which automatically generates a description that matches the content of the pictures. The images are extracted by convolutional neural network (CNN) and further combined with the implicit state of long short-term memory (LSTM). A multi-layer perceptron is introduced to calculate the similarity, and the softmax function is used to compute the attention weights. The authors also put forward the concept of hard attention and soft attention which are significantly different.

Then [11] extended the attention mechanism into the machine translation application. By considering the partial relationship between output word and input words, they proposed both global attention and local attention schemes. The attention mechanism is regularly and prosperously combined with recurrent neural networks (RNN). However because of the recurrent nature of RNN, it

is not conducive to parallel computing, so that usually the training procedure always be time-consuming. [25] uses a self-attention mechanism instead of a circular neural network, while both scaled dot-product attention and multi-head attention scchemes are introduced.

The softmax function are usually introduced to calculate the attention weights, which is dense without any possibility to select agents. Although both [27] and [11] attempt to introduce sparsity by adding special structures, like hard attention or local attention, they ignore introducing sparse constraints on activation function. Reference [12] introduced sparsity into the attention mechanism by seeking a sparse activation function for the first time. The core idea is to take advantage of the fact that the Euclidean projection of any input vector to a simplex is sparse. Reference [16] found that it also falls into the above situation by calculating the max function subgradient. By considering the discontinuity of this projection, the authors introduced a strongly convex regularizer on the dual problem of the max operator with the purpose to guarantee efficient training. The sparsemax algorithm in [12] can be included into the algorithm framework of [16] In our paper, we will utilize the generalized γ-sparsemax algorithm in [16].

2.2 Attention Mechanism in Reinforcement Learning

Attention mechanism is also popularly applied to reinforcement learning methods. Reference [17] used reinforcement learning to conduct brain activity research and introduced attention mechanism to screen input signals in order to accelerate the training process. Reference [18] considered the memory structure in the reinforcement learning algorithm to preserve the knowledge learned by the agent. In decision-making procedure, the attention mechanism is introduced to select relevant information from the memory with the purepose to assist decision-making. A soft attention mechanism combined with the Deep Q-Network (DQN) [14] model is proposed in [15] to highlight task-relevant locations of input frames.

In the field of multi-agent reinforcement learning, [6] combines the attention mechanism with the actor-critic algorithm to train a centralized critic that automatically selects relevant information. [7] also proposed an attention-based actor-critic algorithm. Their main concerns are on learning attention models for sharing information between policies. Our work in this paper is basiclly based on [6], but firstly and originally designs a sparse attentional mechanism for multi-agent reinforcement learning, which strengthens the algorithm stability for complicated and noisy environments and guarantee better interpretability.

3 Preliminaries and Backgrounds

3.1 Markov Decision Process and Markov Game

Reinforcement learning models the process in which an agent learns in constant interaction with the environment as a Markov decision process (MDP) [21]. An MDP is defined by a quintuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$ where $\mathcal{S}$ and $\mathcal{A}$ represent

a limited state space and a limited action space respectively. The probability transfer matrix $\mathcal{P}$ is a three-dimensional tensor, where each element represents the probability that the agent will move to the next state s' after executing action a in state s. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ defines the instant feedback made by the environment after the agent executes the action a in the state s and this feedback can be stochastic. Finally, $\gamma \in [0, 1]$ represents the discount factor that defines the trade-off between the immediate feedback and the accumulated future feedbacks during the agent learning process

Since our work focuses on multi-agent situation, at the beginning we present a formal definition of this problem. A Markov game $\langle \mathcal{S}, \mathcal{N}, \mathcal{A}, \mathcal{R}_{\mathcal{N}}, \mathcal{P}, \gamma \rangle$ denotes an extension of the Markov decision process in multiple agent scenarios [9], while $\mathcal{N}$ represents the number of agents. The key differences with traditional MDP exist on the reward functions, i.e., $\mathcal{R}_{\mathcal{N}}$, while the probability transition matrix $\mathcal{P}$ and the policy π both depend on the actions of all agents in the environment.

3.2 Deep Q-Network

DQN is the most famous deep reinforcement learning method in recent years, which learns the optimal action value function corresponding to the optimal policy by minimizing the mean square bellman error (MSBE):

$$\mathcal{L}(\theta, \mathcal{D}) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[Q(s, a; \theta) - Q_{\text{targ}} \right]^2, \tag{1}$$

where $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$ and $r \in \mathcal{R}$ denote the state, action and reward respectively. $Q(s, a; \theta)$ denotes the action value function which is parameterized with θ. $Q_{\text{targ}} = r(s, a) + \gamma \max_{a'} \bar{Q}(s', a')$. Since DQN approximates the action value function Q with neural network, its convergence can not be guaranteed theoretically until now. In order to ensure the stability of the training procedure, traditionally the target function $\bar{Q}$ and the experience replay buffer $\mathcal{D}$ are introduced, where the parameters of $\bar{Q}$ are periodically updated with the behavior function Q's parameters.

3.3 Vanilla Policy Gradient and Actor Critic

The policy gradient-type method is another typical reinforcement learning algorithm framework. Its core idea is to learn policy directly rather than indirectly establish through learning value functions. More specifically, policy gradient method directly adjust parameters θ of the policy in order to maximize the objective function $J(\theta) = \mathbb{E}_{\tau} [R_{\tau}]$ by gradient ascent scheme with $\nabla_{\theta} J(\theta)$. In details the gradient takes the follow formulation [24]:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{s_t \sim p^{\pi}, a_t \sim \pi_{\theta}} \left[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t}^{T} r(s_t, a_t) \right]. \tag{2}$$

where $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$ denote the state and action at time t, $p^{\pi} = P(s_t | s_{t-1}, a_{t-1}) \times \pi(a_{t-1} | s_{t-1})$ and $\pi_{\theta} = \pi(a_t | s_t)$. The resulting method with

the above formula is called vanilla policy gradient method (or REINFORCE method [26]). However, this method used a real reward in order to directly estimate the gradient, as a result the variance would be very large because of error accumulation. Lots of variant algorithms are proposed with the purpose to solve this shortcoming. Their core idea is to estimate the cumulative rewards with another function. This function is usually called *critic*, while the resulting algorithm framework is called *actor-critic* algorithm [23]. The advantage function $A(s,a) = Q(s,a) - V(s)$ ($V(s)$ denotes the value function at state s) is a popularly used critic function, which measures the pros and cons of one action relative to the average performance.

3.4 Soft Actor Critic

Soft Actor Critic (SAC) [4] is an algorithm of training stochastic policy through off-policy way. The entropy regularization plays an important role in the algorithm framework. The objective of SAC is to maximize a tradeoff between cumulative rewards together with the entropy of the policy distribution. Each agent is encouraged to pay more attention to exploration in the early stage of training while also avoids converging to a local optimal solution. In SAC algorithm framework, the goal becomes to maximize the following objective function, i.e.,

$$\pi^{\star} = \arg\max_{\pi} \mathbf{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t(r(s_t,a_t) + \alpha H(\pi(\cdot|s_t)))\right], \tag{3}$$

where $s_t \in \mathcal{S}, a_t \in \mathcal{A}$ denote the state and action at time t, $\gamma \in (0,1]$ denote the discount factor and $\alpha > 0$ is the trade-off coefficient. Then state value function and action value function are calculated as

$$V_{\pi}(s) = \mathbf{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t(r(s_t,a_t) + \alpha H(\pi(\cdot|s)))\big|s_0 = s\right], \tag{4}$$

$$Q_{\pi}(s,a) = \mathbf{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t) + \alpha\sum_{t=1}^{\infty}\gamma^t H(\pi(\cdot|s_t))\big|s_0 = s, a_0 = a\right]. \tag{5}$$

With these definitions, the corresponding MSBE's target term and policy gradient respectively become

$$Q_{\text{targ}}(s_t,a_t) = r(s_t,a_t) + \mathbf{E}_{s_{t+1}\sim p^{\pi},a_{t+1}\sim\pi_{\theta}}\left[\bar{Q}(s_{t+1},a_{t+1}) - \alpha\log(\pi_{\theta}(a_{t+1}|s_{t+1}))\right],$$

$$\nabla_{\theta}J(\theta) = \mathbf{E}_{s_t\sim p^{\pi},a_t\sim\pi_{\theta}}\left[\nabla_{\theta}\log\pi_{\theta}(a_t|s_t)(\alpha\log\pi_{\theta}(a_t|s_t) - Q(s_t,a_t) + b(s_t))\right],$$

where $b(s)$ is a state-dependent baseline for variance reduction [3]. Under the giving two formulations, the SAC algorithm alternately iteratively updates the *actor* and *critic* like the traditional *actor-critic* algorithm.

4 Algorithm Framework: SparseMAAC

In this section, we introduce our proposed **Multi-Agent Sparse Attention Actor Critic** (SparseMAAC) algorithm. As discussed above, the main motivation for our algorithm is the partial relationship between agents in a practical multi-agent system. Especially only a limited number of other agents have impacts on each agent's own decisions. Excessive consideration of other agents that have less influence will not only reduce the sampling efficiency, but also cause the convergence to a "worse" local optimal solution. The sparse attention mechanism introduced to solve these problems also leads to strong interpretability.

In the following, we introduce our sparse attention mechanism together with its variants, and then discuss our proposed SparseMAAC algorithm in details.

4.1 Sparsemax

The softmax function is commonly used for attention mechanism designing. Suppose the input of the softmax function is a K-dimensional vector $\boldsymbol{z} \in \mathbb{R}^K$, we can guarantee the K-dimensional output vector $\boldsymbol{p} \in \mathbb{R}^K$ which satisfies $\|\boldsymbol{p}\|_1 = 1$. It is obviously that $\boldsymbol{p}$ belongs to a $K-1$ dimension simplex set $\triangle^{K-1}$. The function $f : \mathbb{R}^K \to \triangle^{K-1}$ denotes the mapping to calculate attention weights.

Further in order to guarantee sparse attention weights, [12] proposed the sparsemax attention mechanism by introducing the projection operation to the target simplex set, i.e.,

$$\text{sparsemax}(\boldsymbol{z}) = \arg \min_{\boldsymbol{p} \in \triangle^{K-1}} \|\boldsymbol{p} - \boldsymbol{z}\|_2^2. \tag{6}$$

The optimal solution of problem (6) can be calculated in closed form [12], which greatly simplifies the process of extending sparsemax attention mechanism to multi-agent reinforcement learning algorithms.

4.2 γ-Sparsemax

Although sparse attention weights are obtained through the sparsemax mechanism, but it cannot handle the degree of sparseness. For multi-agent reinforcement learning algorithms, we can properly take advantages of the prior information on the degree of mutual influence between the agents, which can significantly accelerate the training process.

We introduce the general γ-sparsemax attention mechanism proposed in [16] to our multi-agent sparse attention actor critic algorithm framework. The corresponding optimization problem is defined as

$$\max(\boldsymbol{z}) = \max_{i \in \{1,2,\cdots,K\}} (z_i) = \sup_{\boldsymbol{p} \in \triangle^{K-1}} (\boldsymbol{p}^T \boldsymbol{z}), \tag{7}$$

where the subgradient $\partial \max(\boldsymbol{z}) = \{\boldsymbol{e}, e_i = 1, e_{-i} = 0 \big| i \in \arg\max_{i \in \{1,2,\cdots,K\}} z_i\}$. The subgradient $\partial \max(\boldsymbol{z})$ maps $\boldsymbol{z}$ to the $K-1$ dimension simplex set. However

this mapping can not be directly used due to its discontinuity, so that a strongly convex regularizer can be introduced. We can define the new mapping $\Pi_\Omega : \mathbb{R}^K \to \triangle^{K-1}$ by solving the following optimization problem:

$$\Pi_\Omega(\boldsymbol{z}) = \arg\max_{\boldsymbol{p}\in\triangle^{K-1}} \boldsymbol{p}^T\boldsymbol{z} - \gamma\Omega(\boldsymbol{p}), \tag{8}$$

where $\Omega(\boldsymbol{p})$ denotes a strongly convex function. When $\Omega(\boldsymbol{p}) = \frac{1}{2}\|\boldsymbol{p}\|_2^2$, the γ-sparsemax attention mechanism can be calculated by

$$\gamma\text{-sparsemax}(\boldsymbol{z}) = \Pi_\Omega(\boldsymbol{z}) = \arg\max_{\boldsymbol{p}\in\triangle^{K-1}} \left\|\boldsymbol{p} - \frac{\boldsymbol{z}}{\gamma}\right\|_2^2. \tag{9}$$

The coefficient γ can be considered as the sparsity control hyperparameter. Smaller γ leads to more sparsity attention weights. This should be the first work to introduce self-adjusted sparsity machanism into multi-agent reinforcement learning algorithm, while a priori information about the degree of influence between agents can be introduced to guidance adjusting γ.

4.3 Multi-agent Sparse Attention Actor Critic

The MAAC algorithm modifies the framework of MADDPG [10] by replacing the deep deterministic policy gradient (DDPG) [8] with the soft actor critic (SAC). The Q-value function of each agent not only depends on its own state and actions, but also the strategies of all other agents which are utilized as additional information, i.e.,

$$Q_i^\pi(\boldsymbol{s}, \boldsymbol{a}) = Q_i(\boldsymbol{s}, a_1, \cdots, a_n)|_{a_i=\pi_i(s_i)}. \tag{10}$$

For the agent i, all other agents have the same influence on each agent's decision-making procedure. The attention mechanism for constructing special structured Q-valued function plays the main role in MAAC, which indicates that different weights are used for different agents in the training procedure.

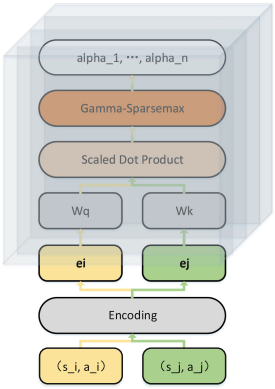
The attention distribution are usually continuous in MAAC, which means that every agent keeps attention on all other agents. However, in many practical applications, the agent's attention has some selective manner. For example, in soccer game, the striker doesn't need to pay attention to his own goal keeper while attacking. Therefore, the attention mechanism needs to extended with sparsity. In this paper, to introduce the sparse attention tactics, we propose the multi-agent sparse attention actor critic (SparseMAAC) algorithm. The main procedure is presented in Algorithm 1, and the key idea of SparseMAAC is shown in Fig. 1(a).

To calculate the attention weight of the remaining agents to the agent i, the attention module receives the observations $\boldsymbol{s} = \{s_1, s_2, \cdots, s_n\}$ and actions $\boldsymbol{a} = \{a_1, a_2, \cdots, a_n\}$ for all agent. Then all the data is divided into two parts, which belong to the agent i and not belong to the agent i, and the latter is denoted as $-i$, and j is used to index in $-i$.

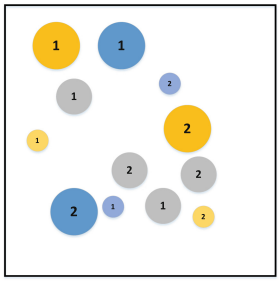
Since the observation of the agent in reinforcement learning often contains a lot of noise, we, like the MAAC algorithm, first encode the agent's observation-action pair, then pass a bilinear mapping, and the final scaled output is used as input of the γ-sparsemax algorithm:

$$\alpha_j = \gamma\text{-sparsemax}\left(\frac{(W_k E_{-i})^T (W_q e_i)}{\sqrt{d}}\right)_j, \tag{11}$$

where d is the dimension of the encoding of tuple (s, a), $W_k, W_q \in \mathbb{R}^{n\times d}$, $E_{-i} \in \mathbb{R}^{d\times(n-1)}$. The j-th column $e_j \in \mathbb{R}^{d\times 1}$ of E_{-i} is the encoding of tuple (s_j, a_j) and $e_i \in \mathbb{R}^{d\times 1}$ is the encoding of tuple (s_i, a_i). We also have used the multiple attention heads to explicitly cluster the coding on the semantic level, so that the learned attention weight is more representative.



(a)



(b)

Fig. 1. (a) Multiple sparse attention head in SparseMAAC. (b) Grouped Cooperative Treasure Collection. The number in the circle represents group.

5 Experiments

5.1 Environments

We evaluation the performance of the proposed algorithm for two aspects. Firstly, we demonstrate the sparse attention can improve the final performance of the algorithm and make the results more interpretable in a multi-agent environment. Second, the affects of different attention sparsity are analyzed.

In the multi-agent literature, Cooperative Treasure Collection and Rover-Tower are usually used for evaluation But, only the latter one is specifically designed to verify the effectiveness of the attention mechanism. Therefore, we choose two environment in our experiments, i.e. Rover-Tower and a modified Cooperative Treasure Collection.

Algorithm 1. SparseMAAC

Input: initialize policy parameters θ_i, target policy parameters $\bar{\theta}_i$ centralized Q-function paramaters ϕ and centralized target Q-function paramaters $\bar{\phi}$ for every agent, initialize replay buffer $\mathcal{D}$ and E parallel environments
for $m = 1$ to n_{episodes} **do**
  Reset all environments and get $s_{i,1}^m$ for each agent i in each environment
  for $t = 1$ to $\min(T_{\text{done}}, T_{\text{max\_per\_episode}})$ **do**
    Select action $a_{i,t}^m \sim \pi_{\theta_i}(\cdot|s_{i,t}^m)$ for each agent i in each environment
    Each agent i excutes action $a_{i,t}^m$ in each environment
    Each agent i observes $s_{i,t+1}^m$ and get $r_{i,t+1}^m$ in each environment
    Store transition $(\boldsymbol{s}_t^m, \boldsymbol{a}_t^m, \boldsymbol{r}_{t+1}^m, \boldsymbol{s}_{t+1}^m)$ for all environments in $\mathcal{D}$
    Randomly sample a batch of transion $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_{t+1}, \boldsymbol{s}_{t+1})$ from $\mathcal{D}$
    for each agent i **do**
      Calculate Q_{targ}^i by 3.4
      Calculate critic loss $\mathcal{L}_i(\phi) = \frac{1}{m}\sum_{j=1}^m (Q_{\text{targ}}^i - Q^i(\boldsymbol{s}_j, a_j^1, \cdots, a_j^N))^2$
    Calculate overall critic loss $\mathcal{L}(\phi) = \frac{1}{N}\sum_{i=1}^N \mathcal{L}_i(\phi)$ and update critic
    for each agent i **do**
      Update actor using sampled policy gradient 3.4
    Update target critic network parameters $\bar{\phi}$
    Update target policy network parameters $\bar{\theta}_i$ and for each agent i
Output: Trained policy parameters θ_i for each agent i

Rover-Tower. The configuration is the same as the experiment in MAAC [6], where each two agents are teamed up and only one agent in each team can move For the agent which can move, the immovable agent in other teams has little influence on its policy. Therefore, this environment can also be used to verify the effectiveness of sparse attention.

Grouped Cooperative Treasure Collection. We design a new collaborative environment show as Fig. 1(b), which consists of two four-agent groups. For each group (1 or 2), there are two deposits (biggest circles), two collectors (gray circles) and two treasures (smallest circles). Each group can be seen as a mini version of the original environment. All entities are moveable except for the treasures. The collectors' task is to collect the treasure and place it in the deposit of the same color. And the task of deposits is to store as many treasures as possible. The agent can not only observe the agent from the same group, but also see different groups. To verify the effectiveness of sparse attention, the tasks of different groups are independent. Therefore, the optimal strategy should be with sparse attention. The complexity of the environment is the same as the original environment.

5.2 Settings

To verify the effectiveness of sparse attention, we compare the proposed SparseMAAC algorithm with the MAAC. For fairness, we keep all the hyperparameters

in the original paper to obtain its best performance. Besides, in order to demonstrate the influence of different sparsity, we further compare the sparsemax with different sparsity ($\gamma = 1, 0.1, 0.01$) in our SparseMAAC, as well as the softmax in MAAC.

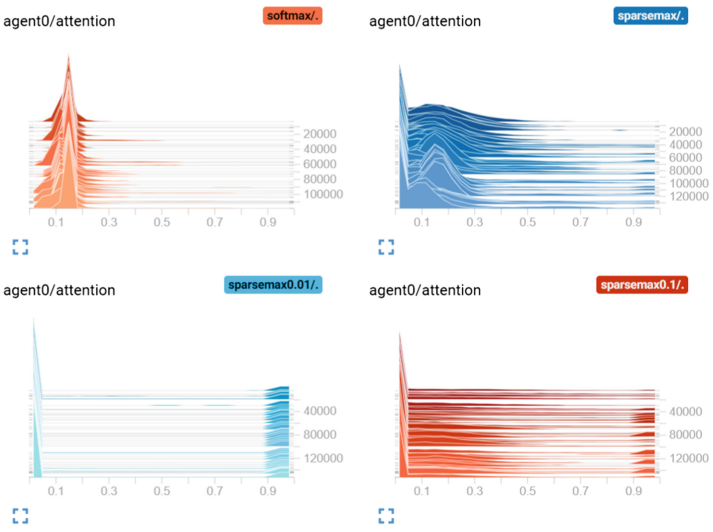
5.3 Results and Analysis

Firstly, we demonstrate that the proposed γ-sparsemax function can produce the sparse attention weight. Figure 2 shows the attention weight of agent-0 in Rover-Tower and Grouped Cooperative Treasure Collection by MAAC (softmax) and SparseMAAC (sparsemax, $\gamma = 0.01, 0.1, 1$). The x axis (0–1) stands for the value of attention, the y axis (0–120000) stands for the iteration number, and the height stands for the attention of agent-0 on other agents. It can be seen that, the output of MAAC (softmax) is all greater than 0 and is concentrated around a certain value (about 0.15 in Fig. 2(a)). So, the agents in MAAC have similar attention for every other agents. But, the attention value of SparseMAAC (sparsemax) are partially 0. The smaller γ is, the more sparse the attention distribution is. Therefore, the agents can learn a sparse policy via SparseMAAC, and treat other agents selectively.

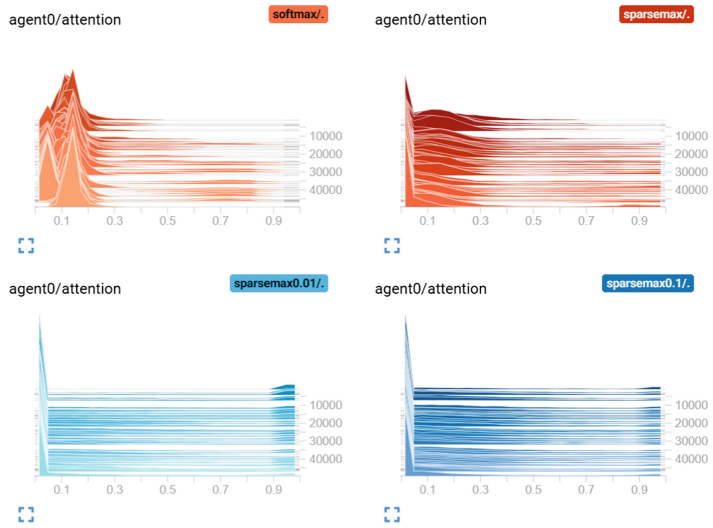
Impact of Sparse Attention. Figure 3(a) shows the performance of the proposed SparseMAAC algorithm with MAAC in the Rover-Tower environment by the mean episode rewards. Obviously, the proposed SparseMAAC algorithm is significantly superior to MAAC, not only in terms of mean rewards, but also the convergence speed. The phenomenon proofs that the introduced attention sparsity can help the agent quickly filter out useful information in the early stage of training. Thus, it speeds up the training process with a better results.

Figure 3(b) shows the performance of the proposed SparseMAAC algorithm with MAAC in the Grouped Cooperative Treasure environment by the mean episode rewards. Because each agent can observe all other agents, the performances of compared algorithms are nearly the same. But the convergence speed of the proposed SparseMAAC is still faster than the MAAC algorithm. SparseMAAC can select useful agents with the sparse attention, so it achieve similar mean episode reward with shorter training time and fewer computations.

Impact of the Sparsity. In this part, we delve into the effectiveness of different sparsity on the performance of the proposed algorithm. We choose the Rover-Tower environment for discussion, where the agents are paired in pairs. For an agent, there is only one agent which has the greatest impact on it. Figure 4 visualizes the attention results of the paired relationship in Rover-Tower. We first train all the algorithms for 50k epochs, and then let each agent randomly sample an observation from the environment and calculate the corresponding attention weight. MAAC and SparseMAAC both adopt the multi-head attention mechanism, and 4 attention heads are totally used in our experiments. The first column in Fig. 4 stands for the ground truth, the 2–5

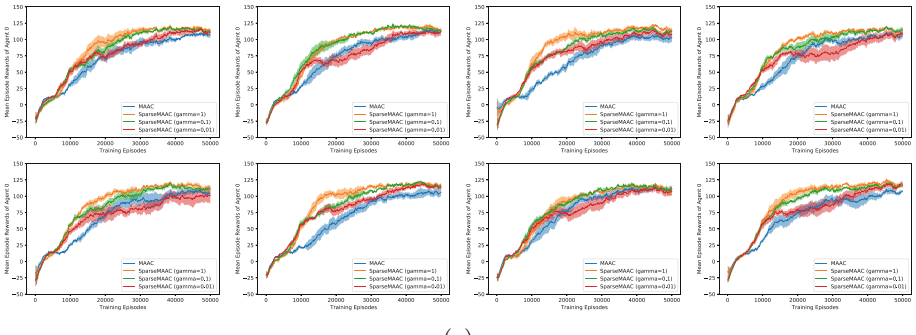


(a)

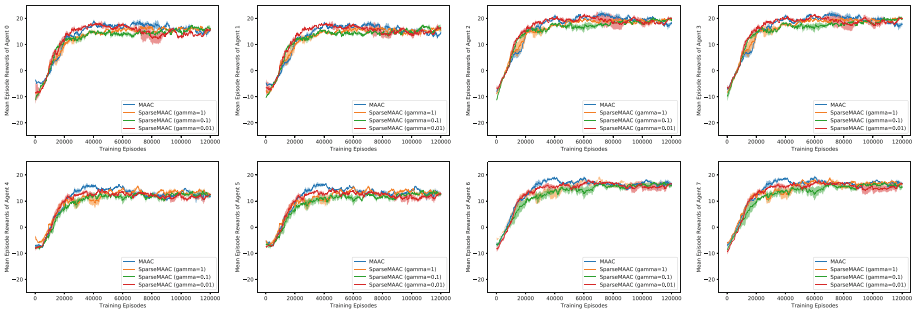


(b)

Fig. 2. (a) The distribution of the attention weights of the 7(=8−1) agents outputted in the Grouped Cooperative Treasure Collection environment with the number of training iterations. (b) The distribution of the attention weights of the 7(=8−1) agents outputted in the Rover-Tower environment with the number of training iterations.



(a)



(b)

Fig. 3. In the legend, softmax represents the MAAC algorithm, and γ-sparsemax represents the SparseMAAC algorithm with different γ values. (a) Mean episode rewards (of 5 runs) of each agent on Grouped Cooperative Treasure Collection environment. (b) Mean episode rewards (of 3 runs) of each agent on Rover-Tower environment. The shaded part represents the standard deviation.

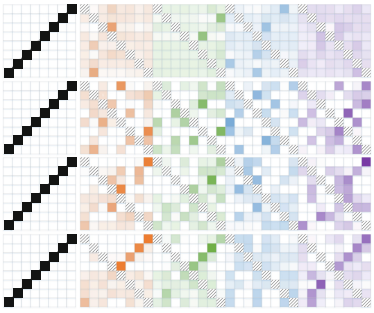


Fig. 4. Attention thermodynamic map in *Rover-Tower* environment. Attention weight is calculated by randomly sampling an observation. The darker the color, the greater the attention weight. Since each agent does not calculate the attention with itself, the main diagonal of the squares in color columns is filled with twill. (Color figure online)

columns stand for different head attention, and the 1–4 row stand for MAAC (softmax), SparseMAAC($\gamma = 0.01, 0.1, 1$).

From Fig. 4, it can be seen that the smaller the γ value, the more the number of white grids. That that the obtained attention weight is more sparse. This also coincides with the results from Fig. 2. We can see that our SparseMAAC algorithm has the trend to concentrate the true diagonal elements. It is most noticeable when the γ is 1. It is worth noting that the third attention head differs greatly from the other three modes in the fourth column. This is because different attention heads learn different level of environment, due to the advantages of the multi-head attention mechanism. In our experiments, the third attention head represents the overall information more than other heads.

6 Conclusion

In order to adjust the agent's selective attention and reduce the complexity of relationship between agents, we introduce the sparse attention mechanism into multi-agent reinforcement learning via a sparsemax function, and propose the Sparse Multiple Attention Actor Critic (SparseMAAC) algorithm. The proposed SparseMAAC can learn a selective policy with sparse attention, and detect helpful agent on the early stage of training procedure. This guarantees our SparseMAAC with a better optimization routine and lower complexity. We also introduce an adjustable scheme via a hyperparameter to control the sparseness of attention. The proposed algorithm under different sparsity level is evaluated in two different environments with MAAC. The results demonstrate that SparseMAAC can achieve sparse attention distribution effectively. The experiment also show that our SparseMAAC can not only exceed or be equal to the MAAC algorithm, but also converge significantly faster than MAAC.

Acknowledgment. This work is supported by the National Natural Science Foundation of China (Grant No. 61702188, No. U1609220, No. U1509219 and No. 61672231).

References

1. Busoniu, L., Babuska, R., De-Schutter, B.: A comprehensive survey of multi-agent reinforcement learning. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **38**(2), 156–172 (2008)
2. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: learning affordance for direct perception in autonomous driving. In: IEEE International Conference on Computer Vision, pp. 2722–2730 (2015)
3. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: 32nd AAAI Conference on Artificial Intelligence (2018)
4. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290 (2018)

5. Haarnoja, T., et al.: Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905 (2018)
6. Iqbal, S., Sha, F.: Actor-attention-critic for multi-agent reinforcement learning. arXiv preprint arXiv:1810.02912 (2018)
7. Jiang, J., Lu, Z.: Learning attentional communication for multi-agent cooperation. arXiv preprint arXiv:1805.07733 (2018)
8. Lillicrap, T., et al.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
9. Littman, M.: Markov games as a framework for multi-agent reinforcement learning. In: International Conference on Machine Learning, pp. 157–163 (1994)
10. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in Neural Information Processing Systems, pp. 6379–6390 (2017)
11. Luong, M.T., Pham, H., Manning, C.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
12. Martins, A., Astudillo, R.: From softmax to sparsemax: a sparse model of attention and multi-label classification. In: International Conference on Machine Learning, pp. 1614–1623 (2016)
13. Matignon, L., Jeanpierre, L., Mouaddib, A.I.: Coordinated multi-robot exploration under communication constraints using decentralized Markov decision processes. In: 26th AAAI Conference on Artificial Intelligence (2012)
14. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
15. Mousavi, S., Schukat, M., Howley, E., Borji, A., Mozayani, N.: Learning to predict where to look in interactive environments using deep recurrent Q-learning. arXiv preprint arXiv:1612.05753 (2016)
16. Niculae, V., Blondel, M.: A regularized framework for sparse and structured neural attention. In: Advances in Neural Information Processing Systems, pp. 3338–3348 (2017)
17. Niv, Y., Daniel, R., Geana, A., Gershman, S., Leong, Y., Radulescu, A., Wilson, R.: Reinforcement learning in multidimensional environments relies on attention mechanisms. J. Neurosci. **35**(21), 8145–8157 (2015)
18. Oh, J., Chockalingam, V., Singh, S., Lee, H.: Control of memory, active perception, and action in minecraft. arXiv preprint arXiv:1605.09128 (2016)
19. OpenAI: Openai Five (2018). https://blog.openai.com/openai-five/
20. Peng, P., et al.: Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. arXiv preprint arXiv:1703.10069 (2017)
21. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, Hoboken (2014)
22. Shoham, Y., Powers, R., Grenager, T.: If multi-agent learning is the answer, what is the question? Artif. Intell. **171**(7), 365–377 (2007)
23. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction (2018)
24. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems, pp. 1057–1063 (2000)
25. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
26. Williams, R.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**(3–4), 229–256 (1992)
27. Xu, K., et al.: Show, attend and tell: Neural image caption generation with visual attention. In: International Conference on Machine Learning, pp. 2048–2057 (2015)