Adaptive Learning Rates for Multi-Agent Reinforcement Learning

Jiechuan Jiang
Peking University
Beijing, China
jiechuan.jiang@pku.edu.cn

Zongqing Lu*
Peking University
Beijing, China
zongqing.lu@pku.edu.cn

ABSTRACT

In multi-agent reinforcement learning (MARL), the learning rates of actors and critic are mostly hand-tuned and fixed. This not only requires heavy tuning but more importantly limits the learning. With adaptive learning rates according to gradient patterns, some optimizers have been proposed for general optimizations, which however do not take into consideration the characteristics of MARL. In this paper, we propose AdaMa to bring adaptive learning rates to cooperative MARL. AdaMa evaluates the contribution of actors' updates to the improvement of Q-value and adaptively updates the learning rates of actors to the direction of maximally improving the Q-value. AdaMa could also dynamically balance the learning rates between the critic and actors according to their varying effects on the learning. Moreover, AdaMa can incorporate the second-order approximation to capture the contribution of pairwise actors' updates and thus more accurately updates the learning rates of actors. Empirically, we show that AdaMa could accelerate learning and improve performance in a variety of multi-agent scenarios. More importantly, AdaMa does not require heavy hyperparameter tuning and thus significantly reduces the training cost. The visualizations of learning rates during training clearly explain how and why AdaMa works.

KEYWORDS

Reinforcement learning; Multi-agent reinforcement learning; Adaptive learning rates

ACM Reference Format:
Jiechuan Jiang and Zongqing Lu. 2023. Adaptive Learning Rates for Multi-Agent Reinforcement Learning. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 8 pages.

1 INTRODUCTION

Recently, multi-agent reinforcement learning (MARL) has been applied to decentralized cooperative systems, *e.g.*, autonomous driving [1], smart grid control [19], and traffic signal control [17]. Many MARL methods [5, 6, 10, 13, 14] have been proposed for multi-agent cooperation, which follow the paradigm of centralized training and decentralized execution. In many of these methods, a centralized critic learns the joint Q-function using the information of all agents, and the decentralized actors are updated towards maximizing the Q-value based on local observation.

However, in these methods, the actors are usually assigned the same learning rates, which is not optimal for maximizing the Q-value. This is because some agents might be more critical than others to improving the Q-value and thus should have higher learning rates. On the other hand, the learning rates of actors and critic are often hand-tuned and fixed, and hence require heavy tuning. More importantly, over the course of training, the effect of actors and critic on the learning varies, so the fixed learning rates will not always be the best at every learning stage. The artificial schedules, *e.g.*, time-based decay and step decay, are pre-defined and require expert knowledge about the model and problem. Some optimizers, *e.g.*, AdaGrad [4] and AdaDelta [20], could adjust the learning rate adaptively, but they adjust the learning rate according to only the gradient pattern and are proposed for general optimization problems, without specifically considering the convergence of Q-value and the improvement of actor policies in the learning process of multi-agent situations.

In this paper, we propose AdaMa for adaptive learning rates in cooperative MARL. AdaMa dynamically evaluates the contribution of actors and critic to the optimization and adaptively updates the learning rates based on their quantitative contributions. First, we examine the gain of Q-value contributed by the update of each actor. We derive the direction along which the Q-value improves the most. Thus, we can update the vector of learning rates of all actors towards the direction of maximizing the Q-value, which leads to diverse learning rates that explicitly capture the contributions of actors. Second, we consider the critic and actors are updated simultaneously. If the critic's update causes a large change of Q-value, we should give a high learning rate to the critic since it is leading the learning. However, the optimization of actors, which relies on the critic, would struggle with the fast-moving target. Thus, the learning rates of actors should be reduced accordingly. On the other hand, if the critic has reached a plateau, increasing the learning rates of actors could quickly improve the actors, which further generates new experiences to boost the critic's learning. These two processes alternate during training, promoting overall learning. Further, by incorporating the second-order approximation, we additionally capture the pairwise interaction between actors' updates so as to more accurately update the learning rates of actors towards maximizing the improvement of Q-value.

AdaMa is a general method and could be applied to many multi-agent actor-critic methods. We evaluate AdaMa in popular multi-agent cooperation scenarios in multi-agent particle environment [10] and multi-agent mujoco [3]. Empirical results demonstrate that dynamically regulating the learning rates of actors and critic according to the contributions to the change of Q-value could accelerate the learning and improve the performance, which can be further

*Corresponding Author

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

enhanced by additionally considering the effect of pairwise actors' updates. AdaMa does not require heavy hyperparameter tuning, which significantly reduces the training cost. The visualizations of learning rates during training clearly explain how and why AdaMa works.

2 RELATED WORK

MARL. We consider the formulation of decentralized partially observable Markov decision process (Dec-POMDP) [12]. There are N agents interacting with the environment. At each timestep t, each agent i receives a local observation o_t^i of the state, takes an individual action a_t^i, and gets a shared reward r_t. The environment transitions to the next state taking the joint action $\vec{a}_t$. The agents aim to maximize the expected return $\mathbb{E}\sum_{t=0}^{T}\gamma^t r_t$, where γ is a discount factor and T is the episode time horizon. Many methods [5, 6, 10, 13, 14] have been proposed for Dec-POMDP, which adopt centralized learning and decentralized execution (CTDE). In many of these methods, a centralized critic learns a joint Q-function by minimizing the TD-error. In training, the critic is allowed to use the information of all agents. The actors, which only have access to local information, learn to maximize the Q-value learned by the critic. In execution, the critic is abandoned and the actors act in a decentralized manner.

Adaptive Learning Rate. Learning rate schedules aim to reduce the learning rate during training according to a pre-defined schedule, including time-based decay, step decay, and exponential decay. The schedules have to be defined in advance and depend heavily on the type of model and problem, which requires much expert knowledge. Some optimizers, such as AdaGrad [4], AdaDelta [20], and RMSprop [15], provide adaptive learning rate to ease manual tuning. AdaGrad performs larger updates for more sparse parameters and smaller updates for less sparse parameters, and other methods are derived from AdaGrad. However, these methods only deal with the gradient pattern for general optimization problems, without specifically considering the convergence of Q-value and the improvement of actor policies. Hence they offer no specialized way to boost multi-agent learning. WoLF [2] provides variable learning rates for stochastic games, but it needs to solve the equilibrium strategy, which is not practical in complex multi-agent environments.

Meta Gradients for Hyperparameters. Some meta-learning methods employ hyperparameter gradients to tune the hyperparameter automatically. Reverse-mode differentiation of hyperparameters has been utilized to optimize step sizes, momentum schedules, weight initialization distributions, parameterized regularization schemes, and neural network architectures [11]. Meta RL [18] computes the meta-gradient to update the discount factor and bootstrapping parameter in reinforcement learning. OL-AUX [9] uses the meta-gradient to automate the weights of auxiliary tasks. LIRPG [21] learns the intrinsic reward by meta-gradient of maximizing the extrinsic reward, avoiding the challenging reward design. These methods are beneficial for improving the performance, and more importantly, reduce the cost of hyperparameter tuning and the complexity of artificial design. The proposed AdaMa can also be viewed as a meta-gradient method for adaptive learning rates in MARL.

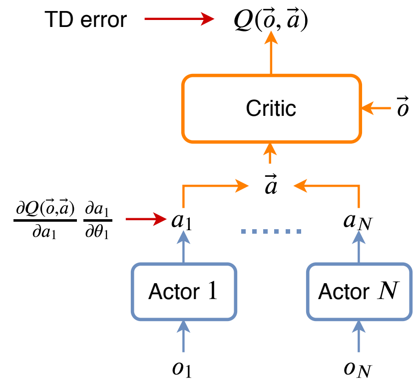


Figure 1: MADDPG

3 METHOD

In this section, we first introduce the single-critic version of MADDPG [10], on which we instantiate AdaMa. Then, we use the Taylor approximation to evaluate the contributions of the critic and actors' updates to the change of Q-value. Based on the derived quantitative contributions, we dynamically adjust the direction of the vector of actors' learning rates and balance the learning rates between the critic and actors. Further, we incorporate higher-order approximation to estimate the contributions more accurately. AdaMa is a general method and could be applied to many multi-agent actor-critic methods, e.g., COMA [5], MAAC [6], and DOP [16].

3.1 Single-Critic MADDPG

In mixed cooperation and competition, each MADDPG agent learns an actor π_i and a critic for the local reward. However, since the agents share the reward in Dec-POMDP, we only maintain a single shared critic, which takes the observation vector $\vec{o}$ and the action vector $\vec{a}$ and outputs the Q-value, as illustrated in Figure 1. The critic parameterized by ϕ is trained by minimizing the TD-error δ

$$\mathbb{E}_{(\vec{o},\vec{a},r,\vec{o}')\sim\mathcal{D}}\left[(Q(\vec{o},\vec{a})-y)^2\right],$$
$$\text{where } y = r+\gamma Q^-(\vec{o}',\vec{\pi_i^-}(o_i')).$$

Q^- is the target critic, π_i^- is the target actor, and $\mathcal{D}$ is replay buffer. Each actor π_i (parameterized by θ_i) is updated to maximize the learned Q-value by gradient ascent. The gradient with respect to θ_i is

$$\frac{\partial Q(\vec{o},\vec{a})}{\partial a_i}\frac{\partial a_i}{\partial \theta_i}.$$

We denote the learning rates of each actor i and the critic as l_{a_i} and l_c respectively.

3.2 Adaptive $\vec{l_a}$ Direction

First, suppose that the critic is trained and frozen, and we only update the actors. By expanding the Q-function, we can estimate the gain of Q-value contributed by actors' updates by the Taylor approximation:

$$\Delta Q = Q(\vec{o},\vec{a}+\Delta\vec{a}) - Q(\vec{o},\vec{a})$$

$$\approx Q(\vec{o},\vec{a}) + \sum_{i=1}^{N} \Delta a_i \frac{\partial Q(\vec{o},\vec{a})}{\partial a_i}^T - Q(\vec{o},\vec{a})$$
$$= \sum_{i=1}^{N} [\pi_i(\theta_i + l_{a_i}\frac{\partial Q(\vec{o},\vec{a})}{\partial \theta_i}) - \pi_i(\theta_i)] \frac{\partial Q(\vec{o},\vec{a})}{\partial a_i}^T$$
$$\approx \sum_{i=1}^{N} l_{a_i} \frac{\partial Q(\vec{o},\vec{a})}{\partial \theta_i} \frac{\partial a_i}{\partial \theta_i}^T \frac{\partial Q(\vec{o},\vec{a})}{\partial a_i}^T$$
$$= \sum_{i=1}^{N} l_{a_i} \frac{\partial Q(\vec{o},\vec{a})}{\partial \theta_i} \frac{\partial Q(\vec{o},\vec{a})}{\partial \theta_i}^T$$
$$= \vec{l_a} \cdot \frac{\partial Q}{\partial \theta} \frac{\vec{\partial Q}}{\partial \theta}^T .$$

Assuming the magnitude of the learning rate vector $\|\vec{l_a}\|$ is a fixed small constant η, the largest ΔQ is obtained when the direction of $\vec{l_a}$ is consistent with the direction of vector $\frac{\partial Q}{\partial \theta}\frac{\vec{\partial Q}}{\partial \theta}^T$. Thus, we can softly update $\vec{l_a}$ to the direction of $\frac{\partial Q}{\partial \theta}\frac{\vec{\partial Q}}{\partial \theta}^T$ to improve the Q-value:

$$\vec{l_a} = \alpha \vec{l_a} + (1-\alpha)\eta \frac{\partial Q}{\partial \theta}\frac{\vec{\partial Q}}{\partial \theta}^T / \| \frac{\partial Q}{\partial \theta}\frac{\vec{\partial Q}}{\partial \theta}^T \|$$
$$\vec{l_a} = \vec{l_a}\frac{\eta}{\|\vec{l_a}\|}, \tag{1}$$

where the second line normalizes the magnitude of $\vec{l_a}$ to η, and α is a parameter that controls the soft update. From another perspective, the update rule (1) can be seen as updating $\vec{l_a}$ by gradient ascent to increase the Q-value the most, since $\frac{\partial \Delta Q}{\partial \vec{l_a}} = \frac{\partial Q}{\partial \theta}\frac{\vec{\partial Q}}{\partial \theta}^T$.

3.3 Adaptive l_c and $\|\vec{l_a}\|$

In the previous section, we assume that the critic is frozen. However, in MADDPG and other MARL methods, the critic and actors are trained simultaneously. Therefore, we investigate the change of Q-value by additionally considering the critic's update:

$$\Delta Q = Q(\phi + \Delta\phi, \vec{o}, \vec{a} + \Delta\vec{a}) - Q(\phi, \vec{o}, \vec{a})$$
$$\approx Q(\phi,\vec{o},\vec{a}) + \sum_{i=1}^{N} \Delta a_i \frac{\partial Q(\phi,\vec{o},\vec{a})}{\partial a_i}^T$$
$$+ \Delta\phi \frac{\partial Q(\phi,\vec{o},\vec{a})}{\partial \phi}^T - Q(\phi,\vec{o},\vec{a})$$
$$\approx \vec{l_a} \cdot \frac{\partial Q}{\partial \theta}\frac{\vec{\partial Q}}{\partial \theta}^T - l_c \frac{\partial \delta}{\partial \phi}\frac{\partial Q}{\partial \phi}^T .$$

We can see that ΔQ is contributed by the updates of both the critic and actors. In principle, the critic's learning is prioritized since the actor's learning is determined by the improved critic. When the critic's update causes a large change of the Q-value, the critic is leading the learning, and we should assign it a high learning rate. However, the optimization of actors, which relies on the current critic, would struggle with the fast-moving target. Therefore, the actors' learning rates should be reduced. On the other hand, when the critic has reached a plateau, increasing the actors' learning rates could quickly optimize the actors, which further injects new experiences into the replay buffer to boost the critic's learning, thus promoting the overall learning. The contributions of actors' updates are always nonnegative, but the critic's update might either increase or decrease the Q-value. Therefore we use the absolute value $|\frac{\partial \delta}{\partial \phi}\frac{\partial Q}{\partial \phi}^T|$ to evaluate the contribution of critic to the change of Q-value. Based on the principles above, we adaptively adjust l_c and $\|\vec{l_a}\|$ by the update rules:

$$l_c = \alpha l_c + (1-\alpha) l \cdot \text{clip}(|\frac{\partial \delta}{\partial \phi}\frac{\partial Q}{\partial \phi}^T|/m, \epsilon, 1-\epsilon)$$
$$\eta = l - l_c. \tag{2}$$

The hyperparameters α, m, l, and ϵ have intuitive interpretations and are easy to tune. α controls the soft update and m controls the target value of l_c. The clip function and the small constant ϵ prevent the learning rate from being too large or too small. For α, l, and ϵ, we recommend using the default settings. And we provide an easy empirical approach for selecting m in Experiments. Therefore, AdaMa does not require heavy hyperparameter tuning. AdaMa works as follows: first update l_c and get η using (2), then regulate the direction and magnitude of $\vec{l_a}$ according to (1).

As pointed out by [7], the actor should have a lower learning rate than the critic, and a high learning rate of actor leads to a performance breakdown. Also, empirically, in DDPG [8] the critic's learning rate is set to 10 times higher than the actor's learning rate. However, we believe such a setting only partially addresses the problem. During training, if the learning rates of actors are always low, actors learn slowly and thus the learning is limited. Therefore, AdaMa keeps high l_c when the critic is updating fast, but decreases l_c and increases $\|\vec{l_a}\|$ when the learning of critic reaches a plateau, which could avoid the fast-moving target and speed up the overall learning.

3.4 Second-Order Approximation

Under the first-order Taylor approximation, the actor i's contribution to ΔQ is only related to the change of a_i, without capturing the joint effect with other agents' updates. However, when there are strong correlations between the agents, the increase of the Q-value cannot be sufficiently estimated as the sum of individual contributions of each actor' update, which instead is a result of the joint update. To estimate the actors' contributions more precisely, we extend AdaMa to the second-order Taylor approximation to take pairwise agents' updates into account:

$$\Delta Q = Q(\vec{o}, \vec{a} + \Delta\vec{a}) - Q(\vec{o},\vec{a})$$
$$\approx \sum_{i=1}^{N} \Delta a_i \frac{\partial Q(\vec{o},\vec{a})}{\partial a_i}^T + \frac{1}{2}\sum_{i,j=1}^{N} \Delta a_i \frac{\partial^2 Q(\vec{o},\vec{a})}{\partial a_i \partial a_j} \Delta a_j^T$$
$$\approx \sum_{i=1}^{N} l_{a_i} \frac{\partial Q(\vec{o},\vec{a})}{\partial \theta_i}\frac{\partial Q(\vec{o},\vec{a})}{\partial \theta_i}^T$$
$$+ \frac{1}{2}\sum_{i,j=1}^{N} l_{a_i} l_{a_j} \frac{\partial Q(\vec{o},\vec{a})}{\partial \theta_i}\frac{\partial a_i}{\partial \theta_i}^T \frac{\partial^2 Q(\vec{o},\vec{a})}{\partial a_i \partial a_j}\frac{\partial a_j}{\partial \theta_j}\frac{\partial Q(\vec{o},\vec{a})}{\partial \theta_j}^T .$$

As the actors are updated by the first-order gradient, we still estimate $\Delta\vec{a}$ utilizing the first-order approximation and compute the second-order ΔQ on the first-order $\Delta\vec{a}$. Then, the gradient

Algorithm 1 AdaMa on MADDPG

Initialize critic network ϕ, actor networks θ_i, target networks, and the replay buffer $\mathcal{D}$.
Initialize the learning rates l_c and $\vec{l_a}$.
for episode = $1, \ldots, \mathcal{M}$ **do**
  for $t = 1, \ldots, \mathcal{T}$ **do**
    Select action $a_t^i = \pi_i(o_t^i) + \mathcal{N}_t^i$ for each agent i
    Execute action a_t^i, obtain reward r_t, and get new observation o_{t+1}^i for each agent i
    Store transition $(\vec{o}_t, \vec{a}_t, r_t, \vec{o}_{t+1})$ in $\mathcal{D}$
  end for
  Sample a random minibatch of transitions from $\mathcal{D}$
  Adjust l_c and $\|\vec{l_a}\|$ by (2).
  Adjust $\vec{l_a}$ by (1) (first order) or (3) (second order).
  Update the critic ϕ by $\phi = \phi - l_c \frac{\partial \delta}{\partial \phi}$.
  Update the actor θ_i by $\theta_i = \theta_i + l_{a_i} \frac{\partial Q(\vec{o}, \vec{a})}{\partial a_i} \frac{\partial a_i}{\partial \theta_i}$ for each agent.
  Update the target networks.
end for

of l_{a_i} is $\frac{\partial \Delta Q}{\partial l_{a_i}} = \frac{\partial Q}{\partial \theta_i} \frac{\partial Q}{\partial \theta_i}^T + \frac{1}{2} \sum_{j=1}^{N} l_{a_j} \frac{\partial Q}{\partial \theta_i} \frac{\partial a_i}{\partial \theta_i}^T \frac{\partial^2 Q}{\partial a_i \partial a_j} \frac{\partial a_j}{\partial \theta_j} \frac{\partial Q}{\partial \theta_j}^T + \frac{1}{2} \sum_{j=1}^{N} l_{a_j} \frac{\partial Q}{\partial \theta_j} \frac{\partial a_j}{\partial \theta_j}^T \frac{\partial^2 Q}{\partial a_j \partial a_i} \frac{\partial a_i}{\partial \theta_i} \frac{\partial Q}{\partial \theta_i}^T$. Similarly, $\vec{l_a}$ can be updated as:

$$\begin{aligned} \vec{l_a} &= \alpha \vec{l_a} + (1-\alpha)\eta \frac{\partial \Delta Q}{\partial \vec{l_a}} / \| \frac{\partial \Delta Q}{\partial \vec{l_a}} \|, \\ \vec{l_a} &= \vec{l_a} \frac{\eta}{\|\vec{l_a}\|}. \end{aligned} \tag{3}$$

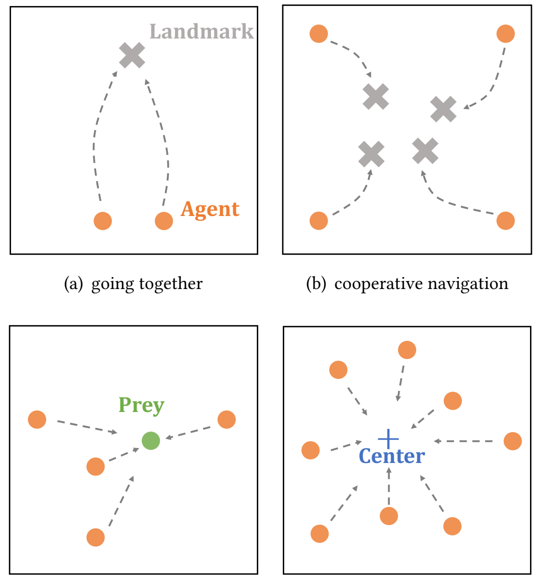
For completeness, we summarize the training of AdaMa on MADDPG in Algorithm 1.

4 EXPERIMENTS

4.1 Settings

We validate AdaMa in four cooperation scenarios based on MPE [10] (MIT License) with continuous observation space and continuous action space, which are illustrated in Figure 2.

- *Going Together.* In the scenario, there are 2 agents and 1 landmark. The reward is $-0.5(d_i + d_j) - d_{ij}$, where d_i is the distance from agent i to the landmark, and d_{ij} is the distance between the two agents. The agents have to go to the landmark together, avoiding moving away from each other.
- *Cooperative Navigation.* In the scenario, there are 4 agents and 4 corresponding landmarks. The reward is $-\max_i(d_i)$, where d_i is the distance from agent i to the landmark i. The slowest agent determines the reward in this scenario.
- *Predator-Prey.* In the scenario, 4 slower agents learn to chase a faster rule-based prey. Each time one of the agents collides with the prey, the agents get a reward +1.
- *Clustering.* In the scenario, 8 agents learn to cluster together. The reward is $-\sum d_i$, where d_i is the distance from agent i to the center of agents' positions. Since the center is changing along with the agents' movements, there are strong interactions between agents.



(a) going together (b) cooperative navigation

(c) predator-prey (d) clustering

Figure 2: Illustration of experimental scenarios.

In these scenarios, agents observe the relative positions of other agents, landmarks, and other items, and take two-dimensional actions $\in [-1, 1]$ as physical velocity. The reward functions are strongly related to all agents, and a small change in one agent's policy would greatly influence the cumulative reward. Thus, the optimization is sensitive to the learning rates.

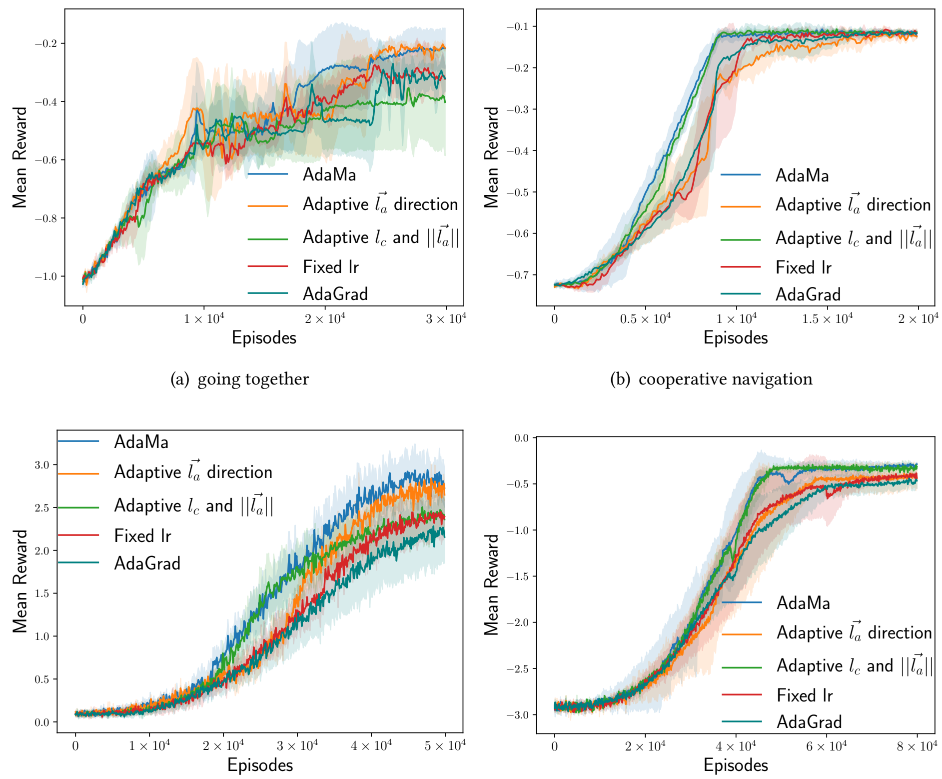
To investigate the effectiveness of `AdaMa` and for ablation, we evaluate the following methods:

- `AdaMa` adjusts l_c and $\|\vec{l_a}\|$ using (2), and $\vec{l_a}$ according to (1).
- `Fixed lr` uses grid search to find the optimal combination of l_c and $\|\vec{l_a}\|$ from 0.01 to 0.001 with step 0.001. The learning rate of each agent is set to $\|\vec{l_a}\|/\sqrt{N}$.
- `Adaptive` $\vec{l_a}$ `direction` sets l_c and $\|\vec{l_a}\|$ as that in `Fixed lr` and only adjusts the direction of $\vec{l_a}$ using (1). Additionally, `Adaptive` $\vec{l_a}$ `direction (2nd)` uses the update rule (3) for the second-order approximation.
- `Adaptive` l_c `and` $\|\vec{l_a}\|$ adjusts l_c and $\|\vec{l_a}\|$ using (2) and sets $l_{a_i} = \|\vec{l_a}\|/\sqrt{N}$.
- `AdaGrad` is an adaptive learning rate optimizer that performs larger updates for more sparse parameters and smaller updates for less sparse parameters. The initial learning rates are sets as that in `Fixed lr`.

Except `AdaGrad`, all other methods use SGD optimizer without momentum. We trained all the models for five runs with different random seeds. All the learning curves are plotted using mean and standard deviation.

4.2 Performance of Adaptive $\vec{l_a}$ Direction

As shown in Figure 3(a) and 3(c), `Adaptive` $\vec{l_a}$ `direction` converges to a higher reward than `Fixed lr` that treats each agent as



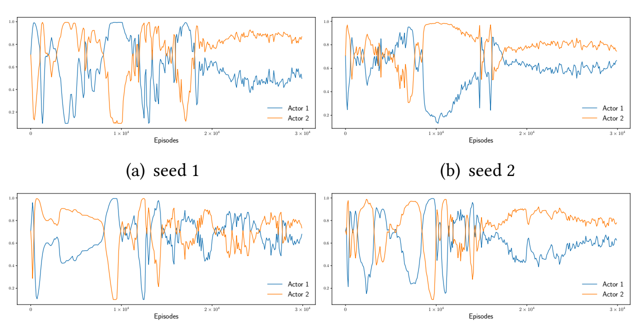
(a) going together

(b) cooperative navigation

(c) predator-prey

(d) clustering

Figure 3: Learning curves in the four scenarios.



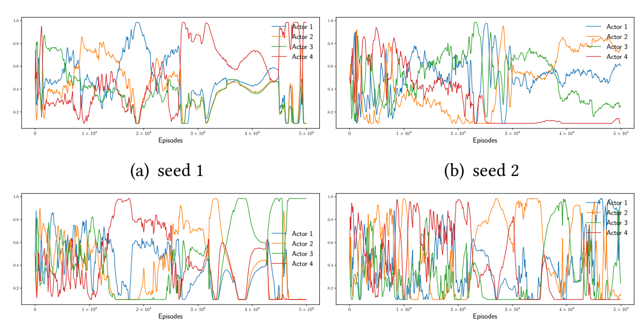
(a) seed 1

(b) seed 2

(c) seed 3

(d) seed 4

Figure 4: Normalized actors' learning rates during the training in going together.



(a) seed 1

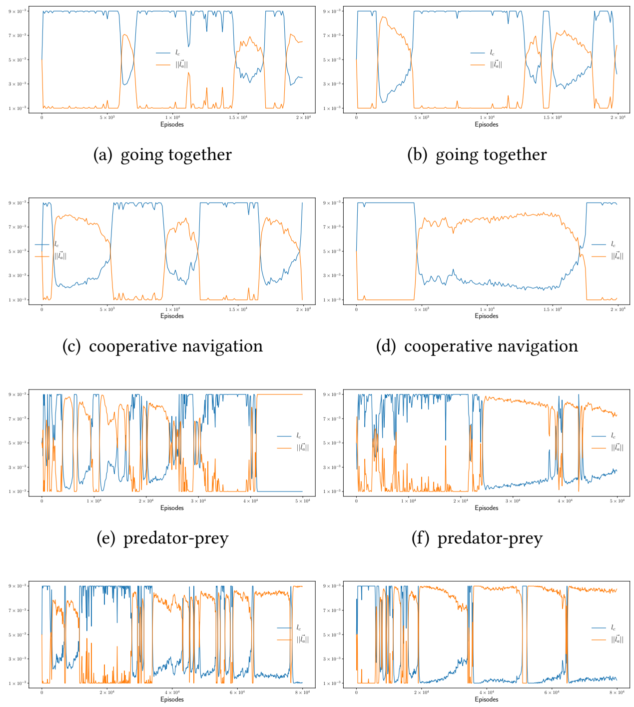
(b) seed 2

(c) seed 3

(d) seed 4

Figure 5: Normalized actors' learning rates during the training in predator-prey

equally important. To make an explicit explanation, we visualize the normalized actors' learning rates $\vec{l_a}/||\vec{l_a}||$ in Figure 4 and Figure 5. In going together and predator-prey, the actors' learning rates fluctuate dynamically and alternately as depicted in Figure 4(a) and 5(a). An actor has a much higher learning rate than other actors in different periods, meaning that the actor is critical to the learning. The direction of $\vec{l_a}$ is adaptive to the changing contributions during the learning, assigning higher learning rates to the actors that make more contributions to ΔQ. In clustering, the center is determined by all agents' positions, and the actors' updates make similar contributions to ΔQ, leading to similar learning rates for the actors. That is the reason `Adaptive` $\vec{l_a}$ `direction` is not beneficial in this scenario. Moreover, in single-critic MADDPG, the gradient of an actor depends on the current policies of other actors. If other actors are updating at a similar rate, the update of this actor will become unstable, since the changes of others' policies are invisible



(a) going together (b) going together

(c) cooperative navigation (d) cooperative navigation

(e) predator-prey (f) predator-prey

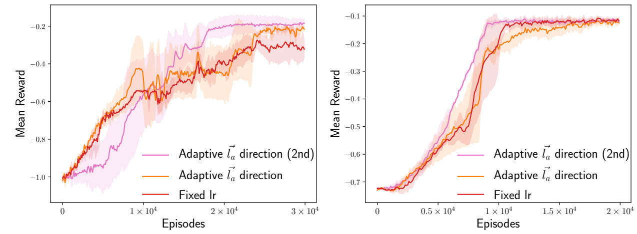
(g) clustering (h) clustering

Figure 6: l_c and $||\vec{l_a}||$ during the training.

and unpredictable. In our method, the agents critical to increasing the Q-value learn fast while other agents have low learning rates, which partly attenuates the instability.

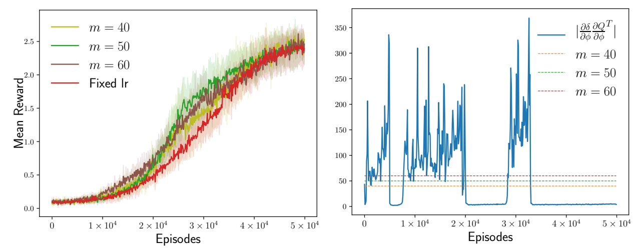
4.3 Performance of Adaptive l_c and $||\vec{l_a}||$

As illustrated in Figure 3(b), 3(c), and 3(d), `Adaptive` l_c `and` $||\vec{l_a}||$ learns faster than `Fixed lr`. To interpret the results, we plot l_c and $||\vec{l_a}||$ during the training in Figure 6 and find that l_c and $||\vec{l_a}||$ rise and fall alternately and periodically. When the update of the critic impacts greatly on ΔQ, *e.g.*, at the beginning with large TD-error, the fast-moving Q-value, which is the optimization target of actors, might cause a performance breakdown if the actors are also learning fast. In this situation, our method could adaptively speed up the learning of the critic and slow down the learning of actors for stability. After a while, the TD-error becomes small and makes the critic reach a plateau. According to the update rules (2), the learning of actors is accelerated whilst the learning rate of the critic falls, which keeps the target of actors stable and thus avoids the breakdown. The fast-improving actors generate new experiences, which change the distribution in the replay buffer and increase the TD-error. As a consequence, the learning rate of the critic rises again. Therefore, the learning rates of the critic and actors fluctuate alternately, promoting the overall learning continuously. In going together, the alternate fluctuation is not obvious, so `Adaptive` l_c `and` $||\vec{l_a}||$ performs worse than `Fixed lr` with grid search.



(a) going together (b) cooperative navigation

Figure 7: Learning curves with the second-order approximation.



(a) learing curves with different m (b) curve of $|\frac{\partial \delta}{\partial \phi}\frac{\partial Q}{\partial \phi}^T|$

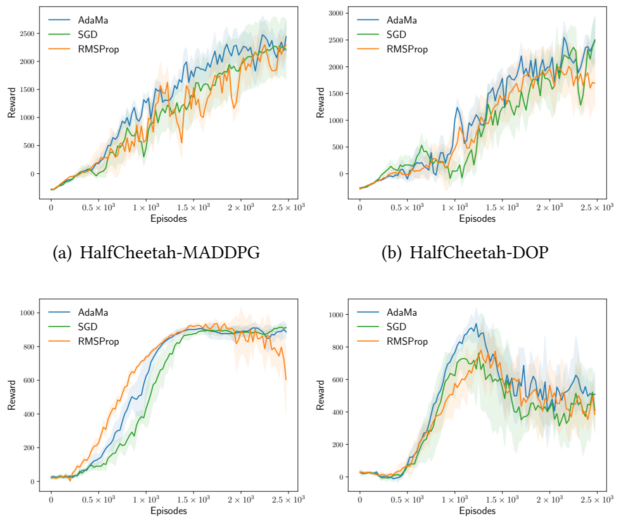
Figure 8: Learning curves and visualizations with different m in Predator-Prey.

Combined with the two adaptive mechanisms, `AdaMa` learns faster and converges to a higher reward than all other baselines in Figure 3(c). In other scenarios, `AdaMa` produces similar results to the mechanism that brings the main improvement. Since `Fixed lr` has to search 100 combinations, **the cost is prohibitive**. `AdaMa` could achieve better performance with very little tuning, which significantly reduces the training cost. Despite adaptively adjusting the learning rates, `AdaGrad` does not show competitive performance, since it only focuses on the gradient pattern, without considering the convergence of Q-value and the improvement of policies.

4.4 Performance of Second-Order Approximation

In Figure 3, the performance gain of `Adaptive` $\vec{l_a}$ `direction` is limited, which we think is attributed to that the first-order approximation is relatively rough when an actor's update affects other actors' updates. We apply the second-order approximation to `Adaptive` $\vec{l_a}$ `direction (2nd)`. As shown in Figure 7, the second-order approximation that captures the pairwise effect of agents' updates on ΔQ obtains a more accurate update on the learning rates, which eventually leads to better performance.

4.4.1 Hyperparameter Tuning. The hyperparameter m controls the target value of the critic's learning rate. If m is too large or too small, the learning rate will reach the boundary value ϵl or $(1-\epsilon)l$, which destroys the adaptability and hampers the learning process. An empirical approach for tuning is setting m to be the



(a) HalfCheetah-MADDPG (b) HalfCheetah-DOP

(c) Ant-MADDPG (d) Ant-DOP

Figure 9: Learning curves of AdaMa on multi-agent mujoco.

mean $|\frac{\partial \delta}{\partial \phi}\frac{\partial Q}{\partial \phi}^T|$ of the first K updates in a trial run. In predator-prey, we test $m = 40, 50, 60$, among which 50 is the rounding mean value of 100 updates, and plot the results in Figure 8(a). The three settings show similar performance, revealing our method is robust to the hyperparameter m. Having noticed that l_c and $||\vec{l_a}||$ change violently in Figure 6, we visualize $|\frac{\partial \delta}{\partial \phi}\frac{\partial Q}{\partial \phi}^T|$ during the training in Figure 8(b) to interpret the robustness. Since most of the time $|\frac{\partial \delta}{\partial \phi}\frac{\partial Q}{\partial \phi}^T|$ is higher than 60 or lower than 40, similar learning rate patterns is observed when m is between 40 and 60, which verifies that there is high fault tolerance in m. Although `Adaptive` l_c and $||\vec{l_a}||$ converges to a similar reward with `Fixed lr`, the former learns faster and is much easier to tune.

4.5 Multi-Agent Mujoco

We also evaluate AdaMa in two multi-agent mujoco [3] tasks, which are 2-agent HalfCheetah and 4-agent Ant. Each agent independently controls one or some joints of the robot and could obtain the state and reward of the robot, which are defined in the original tasks. We instantiate `AdaMa` on MADDPG and the deterministic version of DOP [16], and compare with `SGD` and `RMSprop`. As shown in Figure 9, `AdaMa` learns faster than `SGD` in HalfCheetah and Ant. Although `RMSprop` with MADDPG convergences faster than `AdaMa` with MADDPG in Ant, which is because of the large update rate of `RMSprop` in the early period, the performance of `RMSprop` drops in the latter learning. The experimental results show that `AdaMa` is compatible with different multi-agent actor-critic methods and improves them. Different from the homogeneous agents in MPE scenarios, the agents in multi-agent mujoco are heterogeneous, which means that the agents' contributions to the gain of Q-value might be very diverse. For example, the hind leg is more important than the front leg in HalfCheetah. `AdaMa` could capture the different contributions and gives higher learning rates to the agents that contribute more to the gain of Q-value to improve the Q-value faster.

Table 1: Hyperparameters in MPE

Hyperparameter	GT	CN	PP	CL
horizon (T)	10	6	20	10
discount (γ)	0.96	0.9	0.97	0.95
replay buffer size	5×10^5	5×10^5	1×10^6	1×10^6
l_c (grid search)	8×10^{-3}	9×10^{-3}	7×10^{-3}	8×10^{-3}
$\|\|\vec{l_a}\|\|$ (grid search)	3×10^{-3}	2×10^{-3}	2×10^{-3}	1×10^{-3}
batch size	1024			
MLP units	(64, 64)			
MLP activation	ReLU			
m	10	5	50	80
α	0.99			
l	1×10^{-2}			
ϵ	0.1			

Table 2: Hyperparameters in Mujoco

Hyperparameter	HalfCheetah	Ant
discount (γ)	0.96	0.96
l_c (MADDPG, DOP)	1×10^{-3} , 1×10^{-4}	
l_{a_i} (MADDPG, DOP)	1×10^{-3}, 1×10^{-4}	
replay buffer size	1×10^6	
horizon (T)	1000	
batch size	1024	
MLP units	(64, 64)	
MLP activation	ReLU	
m (MADDPG, DOP)	100, 1000	50, 100
α	0.99	
l (MADDPG, DOP)	1×10^{-3}, 1×10^{-4}	
ϵ	0.05	

4.6 Detailed Settings and Hyperparameters

The experimental settings and hyperparameters of MPE tasks are summarized in Table 1. Initially, we set $l_c = ||\vec{l_a}||$ and $l_{a_i} = ||\vec{l_a}||/\sqrt{N}$ in AdaMa. For exploration, we add random noise to the action $(1-\varepsilon)a_i + \varepsilon\eta$, where the uniform distribution $\eta \in [-1, 1]$. We anneal ε linearly from 1.0 to 0.1 over 10^4 episodes and keep it constant for the rest of the learning. We update the model every episode and update the target networks every 20 episodes.

The experimental settings and hyperparameters of Mujoco tasks are summarized in Table 2. For exploration, we add random noise to the action $(1-\varepsilon)a_i + \varepsilon\eta$, where the uniform distribution $\eta \in [-1, 1]$. We anneal ε linearly from 1.0 to 0.05 over 10^3 episodes and keep it constant for the rest of the learning. We update the model every episode and softly update the target networks with the update ratio 0.98.

The experiments are carried out on Intel i7-8700 CPU and NVIDIA GTX 1080Ti GPU.

5 CONCLUSION

In this paper, we proposed AdaMa for adaptive learning rates in multi-agent reinforcement learning. AdaMa adaptively updates the vector of learning rates of actors to the direction of maximally improving the Q-value. It also dynamically balances the learning rates between the critic and actors during learning. Moreover, AdaMa can incorporate the higher-order approximation to more accurately update the learning rates of actors. Empirically, we show that AdaMa could accelerate learning and improve performance in a variety of multi-agent scenarios. AdaMa is a general method and could be applied to many multi-agent actor-critic methods. AdaMa does not require heavy hyperparameter tuning and thus significantly reduces the training cost. In future work, we hope to explore methods for adaptive learning rates in decentralized multi-agent learning.

ACKNOWLEDGMENTS

This work was supported by NSFC under grant 62250068.

REFERENCES

[1] Sushrut Bhalla, Sriram Ganapathi Subramanian, and Mark Crowley. 2020. Deep Multi Agent Reinforcement Learning for Autonomous Driving. In *Canadian Conference on Artificial Intelligence (Canadian AI)*.

[2] Michael Bowling and Manuela Veloso. 2002. Multiagent learning using a variable learning rate. *Artificial Intelligence* 136, 2 (2002), 215–250.

[3] Christian Schroeder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2020. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *arXiv preprint arXiv:2003.06709* (2020).

[4] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* 12, 7 (2011).

[5] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[6] Shariq Iqbal and Fei Sha. 2019. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

[7] Roman Liessner, Jakob Schmitt, Ansgar Dietermann, and Bernard Bäker. 2019. Hyperparameter Optimization for Deep Reinforcement Learning in Vehicle Energy Management. In *International Conference on Agents and Artificial Intelligence (ICAART)*.

[8] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning.. In *International Conference on Learning Representations (ICLR)*.

[9] Xingyu Lin, Harjatin Baweja, George Kantor, and David Held. 2019. Adaptive Auxiliary Task Weighting for Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[10] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[11] Dougal Maclaurin, David Duvenaud, and Ryan Adams. 2015. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning (ICML)*.

[12] Frans A Oliehoek, Christopher Amato, et al. 2016. *A concise introduction to decentralized POMDPs*. Vol. 1. Springer.

[13] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*.

[14] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

[15] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[16] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. 2021. DOP: Off-Policy Multi-Agent Decomposed Policy Gradients. (2021).

[17] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. 2019. Colight: Learning network-level cooperation for traffic signal control. In *International Conference on Information and Knowledge Management (CIKM)*.

[18] Zhongwen Xu, Hado P van Hasselt, and David Silver. 2018. Meta-gradient reinforcement learning. In *Advances in neural information processing systems (NeurIPS)*.

[19] Yaodong Yang, Jianye Hao, Mingyang Sun, Zan Wang, Changjie Fan, and Goran Strbac. 2018. Recurrent deep multiagent Q-learning for autonomous brokers in smart grid. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[20] Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

[21] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. 2018. On learning intrinsic rewards for policy gradient methods. In *Advances in Neural Information Processing Systems (NeurIPS)*.