# MA-TREX: Mutli-agent Trajectory-Ranked Reward Extrapolation via Inverse Reinforcement Learning

Sili Huang[1,2], Bo Yang[1,2(✉)], Hechang Chen[2,3], Haiyin Piao[4], Zhixiao Sun[5], and Yi Chang[2,3]

[1] College of Computer Science and Technology, Jilin University, Changchun, China
ybo@jlu.edu.cn
[2] Key Laboratory of Symbolic Computation and Knowledge Engineering, Ministry of Education, Beijing, China
[3] College of Artificial Intelligence, Jilin University, Changchun, China
[4] School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China
[5] Unmanned System Research Institute, Northwestern Polytechnical University, Xi'an, China

**Abstract.** Trajectory-ranked reward extrapolation (T-REX) provides a general framework to infer users' intentions from sub-optimal demonstrations. However, it becomes inflexible when encountering multi-agent scenarios, due to its high complexity caused by rational behaviors, e.g., cooperation and communication. In this paper, we propose a novel Multi-Agent Trajectory-ranked Reward EXtrapolation framework (MA-TREX), which adopts inverse reinforcement learning to infer demonstrators' cooperative intention in the environment with high-dimensional state-action space. Specifically, to reduce the dependence on demonstrators, the MA-TREX uses self-generated demonstrations to iteratively extrapolate the reward function. Moreover, a knowledge transfer method is adopted in the iteration process, by which the self-generated data required subsequently is only one third of the initial demonstrations. Experimental results on several multi-agent collaborative tasks demonstrate that the MA-TREX can effectively surpass the demonstrators and obtain the same level reward as the ground truth quickly and stably.

**Keywords:** Mutli-agent system · Inverse reinforcement learning · Reward extrapolation · Iterative extrapolation · Knowledge transfer

## 1 Introduction

Existing multi-agent reinforcement learning can effectively deal with multi-agent tasks with reasonable reward design [14]. However, in many complex scenarios, it

This work was supported in part by National Natural Science Foundation of China under grants 61876069, 61572226, 61902145, Jilin Province Key Scientific and Technological Research and Development project under grants 20180201067GX and 20180201044GX.


G. Li et al. (Eds.): KSEM 2020, LNAI 12275, pp. 3–14, 2020.
https://doi.org/10.1007/978-3-030-55393-7_1

is difficult for experts to design reasonable rewards and goals, and agents cannot learn the behaviors people expect [2,16]. If the agent cannot obtain the reward signal, inverse reinforcement learning can find a reasonable reward function from demonstrations, which are provided by the demonstrator [1]. It has been confirmed that in a complex multi-agent environment when the agent can obtain the high-performance expert trajectory, the reward function highly related to the basic facts can be restored [1]. Unfortunately, various complex tasks cannot provide high-quality expert demonstrations [18,21], and the problem is more serious in the multi-agent field.

If a demonstrator is sub-optimal and can inform their intentions, the agent can use these intents to learn performance beyond the demonstrator [5,20]. But most of the existing inverse reinforcement learning algorithms cannot do this, and usually look for reward functions that make the demonstration look close to the best [8,9,17,22]. Therefore, when the demonstrator is sub-optimal, IRL will also lead to sub-optimal behavior such as behavior cloning [19]. Imitation learning method [3] directly imitates behavior without reward inference, which also has the same disadvantage. Brown proposed an algorithm learned from the sub-optimal demonstrator [5], but it is only effective for single-agent problems, and reward inference is limited to the demonstrator. Different from the single agent, multi-agent problems usually use Nash equilibrium [11] as the optimal solution, which makes the algorithm more demanding on the demonstrator and more difficult for reward inference.

In view of this, inspired by the trajectory-ranked reward extrapolation (T-REX) algorithm [5], we propose a novel multi-agent trajectory-ranked reward extrapolation (MA-TREX) framework, and give an iterative form of reward extrapolation using self-generated demonstrations. Specifically, through the ranked team trajectories, the reward function learns to allocate higher team rewards for better trajectories based on the global state, and guides the agent to achieve performance beyond the demonstrator. In order to break through the demonstrators' restrictions on reward reasoning, collect new trajectories generated during the agents' learning process, and add ranking labels as a new training set. The new reward function uses the new ranked demonstrations to reason about higher returns, and is then used to train agents with higher performance. In the learning process of the new reward function, a knowledge transfer method is adopted, which takes only a small amount of demonstrations to complete the learning after inheriting the parameters of the previous round of reward function. Our contributions can be summarized as following:

- A novel multi-agent trajectory-ranked reward extrapolation (MA-TREX) framework is proposed. To the best of our knowledge, this is the first framework for MA-IRL, which only uses a few ranked sub-optimal demonstrations to infer the users' intentions in multi-agent tasks.
- Learning from the trajectory generated during the agent training process further reduces the dependence on the demonstrator, and the reward function learning from generated trajectories can achieve the same level reward as the ground-truth quickly and stably.

– By combining the idea of knowledge transfer in the iterative process, the self-generated trajectories required to learn the reward function subsequently is only one-third of the initial trajectories, thereby reducing the cost of adding preference labels to pairwise trajectories.
– The effectiveness of our proposed MA-TREX is validated by using several simulated particle environments in that simulated particle environments are representative and most of the cutting-edge MA-IRL algorithms are validated based on them.

## 2 Preliminaries

In this section, we introduce Markov game concepts and existing algorithms involved in the experiment, and give definitions of commonly used symbols.

### 2.1 Markov Games

Markov games [13] are generalizations of Markov decision processes to the case of $N$ interacting agents, which can be represented as a tuple $(N, S, A, P, \eta, r)$. In a Markov game with $N$ agents, where $S$ represents the global state and $\{A_i\}_{i=1}^{N}$ represents the set of actions taken by agents, $P : S \times A_1 \times ... \times A_n$ is the state transition probability of the environment. At time $t$, the agents are in the state $s^t$, chooses the action $(a_1...a_N)$, and the probability of the state transitioning to $s^{t+1}$ is $P\left(s^{t+1}|s^t, a_1, ..., a_N\right)$. The agent can get a reward through the function $r_i : S \times A_1 \times ... \times A_N \rightarrow R$. $\eta$ represents the distribution of the initial environmental state. We use $\pi$ without subscript $i$ to represent the agent's joint policy, $a_i$ represents the action of agent $i$, and $a_{-i}$ represents the set of actions of all agents except for $i$. The goal of each agent $i$ is to maximize their expected returns $E_\pi\left[\sum_{t=1}^{T} \gamma^t r_{i,t}\right]$, where $\gamma$ is the discount factor and $r_{i,t}$ is the reward obtained by agent $i$ at step $t$ in the future.

### 2.2 Trajectory-Ranked Reward Extrapolation

Suppose agent cannot obtain the ground-truth reward signal $r$, but there are some demonstrations $D$ provided by demonstrator. $D$ is the set of trajectories $\{\tau_i\}_{i=1}^{m}$, which is obtained by sampling after expert $\pi_E$ interacts in the environment. Unlike traditional inverse reinforcement learning, when the demonstrator is sub-optimal, but experts can rank these trajectories without using ground-truth rewards, the goal of trajectory-ranked reward extrapolation (TREX) is to infer the users potential intention through the ranked demonstrations. Utilizing this intention allows agents to learn policies beyond the demonstrator.

More specifically, given a sequence of $m$ ranked trajectories $\tau_t$ for $t = 1...m$, where $\tau_i \prec \tau_j$ if $i < j$. The goal of TREX is to predict the cumulative return $J(\tau)$ of the trajectory, and classify the pairwise trajectories $(\tau_i, \tau_j)$ in order to

learn the potential optimization goals of experts. The objective function of the classifier is defined in the form of cross entropy:

$$L(\theta) = -\sum_{\tau_i \prec \tau_j} \log \frac{exp \sum_{s \in \tau_j} r_\theta(s)}{exp \sum_{s \in \tau_i} r_\theta(s) + exp \sum_{s \in \tau_j} r_\theta(s)} \tag{1}$$

where $r_\theta$ is the evaluation of the state s by the reward function.

## 3 Methodology

In this section, we first describe our MA-TREX algorithm, which is a multi-agent version of TREX. Then, we will introduce the iterative form MA-TREX, which is an improved version of MA-TREX.

### 3.1 Multi-agent Trajectory Ranked Reward Extrapolation

Similar to the TREX assumption, we use expert knowledge to rank demonstrations without ground-truth rewards [4,15]. MA-TREX infers the cooperation intention of the demonstrator based on the ranking. As is shown in Fig. 1, given $T$ demonstrations, from the worst to the best $(\tau_{11}, ..., \tau_{1N}), ..., (\tau_{T1}, ..., \tau_{TN})$. MA-TREX has two main steps: (1) joint reward inference and (2) policy optimization.

Given the ranked demonstrations, the MA-TREX uses a neural network to predict the team return $r_\theta(S)$ for the global state $S : (s_1, s_2, ..., s_N)$, and performs reward inference such that $\sum_{S \in (\tau_{i1}, ..., \tau_{iN})} r_\theta(S) < \sum_{S \in (\tau_{j1}, ..., \tau_{jN})} r_\theta(S)$, when $(\tau_{i1}, ..., \tau_{iN}) \prec (\tau_{j1}, ..., \tau_{jN})$. The reward function $r_\theta$ can be trained with ranked demonstrations using the generalized loss function:

$$\begin{aligned} L(\theta) = E_{(\tau_{i1}, ..., \tau_{iN}), (\tau_{j1}, ..., \tau_{jN}) \sim \pi} [\xi (P(J_\theta(\tau_{i1}, ..., \tau_{iN}) < J_\theta(\tau_{j1}, ..., \tau_{jN}))), \\ (\tau_{i1}, ..., \tau_{iN}) \prec (\tau_{j1}, ..., \tau_{jN})] \end{aligned} \tag{2}$$

where $\pi$ represents the joint distribution of the team demonstration, $\prec$ represents the preference relationship between the pairwise trajectories, $\xi$ corresponds to the binary classification loss function, and $J_\theta$ is the cumulative return to the team trajectory $\tau$ calculated using the reward function.

Specifically, we use cross entropy as the classification loss function. The cumulative return $J_\theta$ is used to calculate the softmax normalized probability distribution $P$. We can derive the pairwise trajectories classification probability and loss function:

$$P(J_\theta(\tau_{i\tau}) < J_\theta(\tau_{j\tau})) \approx \frac{exp \sum_{S \in \tau_{j\tau}} r_\theta(S)}{exp \sum_{S \in \tau_{i\tau}} r_\theta(S) + exp \sum_{S \in \tau_{j\tau}} r_\theta(S)} \tag{3}$$

$$L(\theta) = -\sum_{\tau_{i\tau} \prec \tau_{j\tau}} \log \frac{exp \sum_{S \in \tau_{j\tau}} r_\theta(S)}{exp \sum_{S \in \tau_{i\tau}} r_\theta(S) + exp \sum_{S \in \tau_{j\tau}} r_\theta(S)} \tag{4}$$

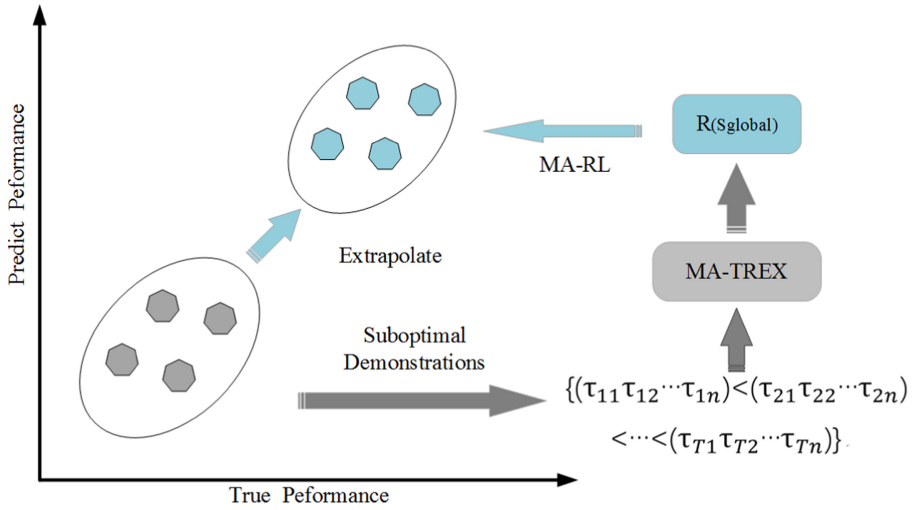


**Fig. 1.** The MA-TREX obtains some ranked demonstrations and learns a joint reward function from these rankings. Through multi-agent reinforcement learning, the learned joint reward function can be used to train joint strategies better than demonstrator.

where $\tau_{i\tau} = (\tau_{i1}, ..., \tau_{iN})$. Through the above loss function, a classifier can be trained, and the classifier calculates which trajectory is better based on the cumulative return of the team. This form of loss function follows from the classic Bradley-Terry and Luce-Shephard models of preferences [4,15] and has been shown to be effective for training neural networks from preferences [6,12]. To increase the number of training samples, we use data augmentation to obtain pairwise preferences from partial trajectories, which can reduce the cost of generating demonstrations. The specific scheme is to randomly select pairwise team trajectories from demonstrations and extract partial state sets, respectively. By predicting the return of the state, the cumulative return of the trajectory is calculated as the logit value in the cross entropy.

Based on the above method, the MA-TREX can obtain the team's cumulative return $r_\theta(S)$ from the demonstrations. We use multi-agent reinforcement learning to train the joint policy $\pi$ through $r_\theta(S)$. The optimization goal of agent $i$ is:

$$J(\pi_i) = E[\sum_{t=0}^{\infty} \gamma^t r_\theta(S) | \pi_i, \pi_{-i}] \tag{5}$$

where the reward function in the formula is not a ground-truth reward, but the return value predicted by the neural network for the global state $S$. Using the predicted reward function, agents with better performance than experts can be obtained.

### 3.2 MA-TREX Iterative Optimization

In multi-agent tasks, our algorithm can extrapolate rewards from sub-optimal demonstrator and train agents with better performance. As with the initial assumption, we can collect the trajectory during the training process and add a preference label to generate a new training set, and then use the above method to train a new reward function.

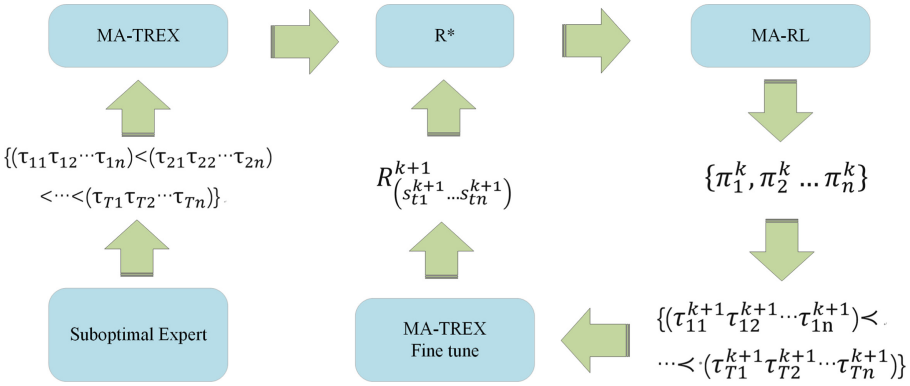


**Fig. 2.** An iterative form MA-TREX. After the first round of reward function learning, the new demonstrations generated during each multi-agent reinforcement learning process is combined with the fine tune method to train a new reward function.

The iterative training process is shown in Fig. 2. Unlike the initial iteration, the training uses demonstrations that are not provided by human experts, but are generated independently by the model. Humans only need to provide ranking labels for new demonstrations. In addition, although the demonstrations used in each iteration are different, the tasks are the same, so we combined the idea of knowledge transfer in meta-learning. Similar to the application of fine-tune technology in image classification, the problem of pairwise trajectories classification can be viewed as two steps: the first step is to extract the features of the global state, and the second step is to evaluate the features. Intuitively, the second step should be re-learned, so that in subsequent iterations, the new reward function inherits the first half of the previous reward function network. The advantage of using parameter inheritance is that subsequent training of the new reward function only needs to generate one-third of the initial demonstrations, and the reward can be extrapolated again.

For the new demonstrations, we still use cross entropy as the loss function, and calculate the softmax normalized probability distribution $p$ by predicting the cumulative return of the new trajectory. We can derive the classification probability and loss function in iterative form:

$$P(J_{\theta_k}(\tau_{i\tau}^{k-1}) < J_{\theta_k}(\tau_{j\tau}^{k-1}k)) \approx \frac{exp\sum_{S\in\tau_{j\tau}^{k-1}} r_{\theta_k}(S)}{exp\sum_{S\in\tau_{i\tau}^{k-1}} r_{\theta_k}(S) + exp\sum_{S\in\tau_{j\tau}^{k-1}} r_{\theta_k}(S)} \quad (6)$$

$$L(\theta_k) = -\sum_{\tau_{i\tau}^{k-1} \prec \tau_{j\tau}^{k-1}} \log \frac{exp \sum_{S \in \tau_{j\tau}^{k-1}} r_{\theta_k}(S)}{exp \sum_{S \in \tau_{i\tau}^{k-1}} r_{\theta_k}(S) + exp \sum_{S \in \tau_{j\tau}^{k-1}} r_{\theta_k}(S)} \quad (7)$$

where the demonstration for the k-th network is generated by the training policy at the k-1th iteration. When $k = 1$, the formula is the same as the MA-TREX without iteration.

Each iteration of the MA-TREX can obtain a new reward function $r_{\theta_k}(S)$. Combining multi-agent reinforcement learning, we fuse the reward function learned in multiple iterations to train new joint policy $\pi_k$. The iterative multi-agent reinforcement learning objective function is:

$$J(\pi_{k,i}) = E[\sum_{t=0}^{\infty} \sum_{j=1}^{k} \gamma^t w_j r_{\theta_j}(S) | \pi_{k,i}, \pi_{k,-i}] \quad (8)$$

where $w_j$ represents the weight of the reward function $r_{\theta_j}$ in the fusion reward function, $k$ represents the $k-th$ iteration. In the experiment, specify that the latest round of reward function has a weight of 0.5, because the demonstrations used in the new round of iterative training is usually better. We summarize the MA-TREX iterative training process in Algorithm 1.

## 4 Experiments

In this section, we evaluate our MA-TREX algorithm on a series of simulated particle environments. Specifically, consider the following scenarios: 1) Cooperative navigation, three agents need to reach three target points by cooperating with each other while maintaining no collision; 2) Cooperative communication, two agents, a speaker and a listener, navigate to the target location by cooperating with each other; 3) Cooperative reference, similar to cooperative communication, but each agent acts as both speaker and listener.

### 4.1 Experiment Demonstrations

To generate expert trajectories, we use ground-truth rewards and the standard multi-agent deep deterministic policy gradient (MADDPG) algorithm to train sub-optimal demonstrator models. In order to investigate the reward extrapolation ability of the MA-TREX under different performance demonstrations, three demonstrator models with different performances were trained for different tasks. Specifically, for each task, we train 500 steps, 1000 steps, and 1500 steps, and collect 1500 pairwise trajectories, respectively. In iterative optimization, new policies are trained using predicted rewards, and 500 pairwise trajectories are collected from the training process.

**Algorithm 1:** MA-TREX Iterative Optimization Algorithm.

**Input:** Expert policy $\pi^1 = \pi^E$;Ranked sub-optimal Expert trajectories $D_{\pi^1} = \{\tau_j^1\}$; Markov game as a black box with parameters $(N, S, A, P, \eta)$; Maximum number of iterations $K$; The set of joint policies $\{\pi^k\}_{k=1}^K$ ; The set of reward functions $\{r_{\theta_k}\}_{k=1}^K$

**Output:** The set of reward functions $\{r_{\theta_k}\}_{k=2}^K$; The last joint policy $\pi^K$

Initialize Joint policies $\{\pi^k\}_{k=1}^K$ ; Reward functions $\{r_{\theta_k}\}_{k=1}^K$.

**for** $k = 2, ..., K$ **do**

  Inherit the first half network of $\theta_{k-1}$ to $\theta_k$.

  Add preference labels to new trajectories $\{\tau_j^{k-1}\}$.

  Sample pairwise trajectories with preference labels:

  $(\tau_{i1}^{k-1}, \tau_{i2}^{k-1}, ..., \tau_{iN}^{k-1}) \prec (\tau_{j1}^{k-1}, \tau_{j2}^{k-1}, ..., \tau_{jN}^{k-1}) \sim D_{\pi_{k-1}}$

  Update $\theta_k$ to optimization objective function in EP. 7:

  $L(\theta_k) = -\sum_{\tau_{i\tau}^{k-1} \prec \tau_{j\tau}^{k-1}} \log \frac{exp \sum_{S \in \tau_{j\tau}^{k-1}} r_{\theta_k}(S)}{exp \sum_{S \in \tau_{i\tau}^{k-1}} r_{\theta_k}(S) + exp \sum_{S \in \tau_{j\tau}^{k-1}} r_{\theta_k}(S)}$

  **for** $i = 1, 2, ..., N$ **do**

    Update $\pi_{k,i}$ with fusion reward to increase the objective in EP. 8:

    $J(\pi_{k,i}) = E[\sum_{t=0}^{\infty} \sum_{j=1}^{k} \gamma^t w_j r_{\theta_j}(S) | \pi_{k,i}, \pi_{k,-i}]$

  **end**

**end**

**Return** $\{r_{\theta k}\}_{k=1}^K$; $\pi^K$.

### 4.2 Experiment Setup

We use 1500 random pairwise trajectories with preference labels based on trajectory ranking instead of ground-truth rewards to train the first reward network. In the iterative training phase of the new reward function, the fine tune technology is used to inherit the first half of the parameters of the previous reward network, and only 500 new random trajectories are used for training.

In order to evaluate the quality of predicted rewards, the policy is trained through the multi-agent deep deterministic policy gradient (MADDPG) algorithm to maximize the reward function. For the iterative training process, with the latest reward accounting for half of the fusion reward standard, the learning rate is fixed at 0.01 and the batch size is fixed at 100, to ensure that the non-reward function external factors have minimal impact on performance. After training is completed, the ground-truth reward signal is utilized to evaluate the performance of the joint policy.

### 4.3 Result and Analysis

We compared against two multi-agent inverse reinforcement learning (MA-IRL) algorithms that have achieved remarkable results in the task: multi-agent generative adversarial imitation learning (MA-GAIL) [21] and multi-agent adversarial inverse reinforcement learning (MA-AIRL) [18]. MA-GAIL and MA-AIRL are

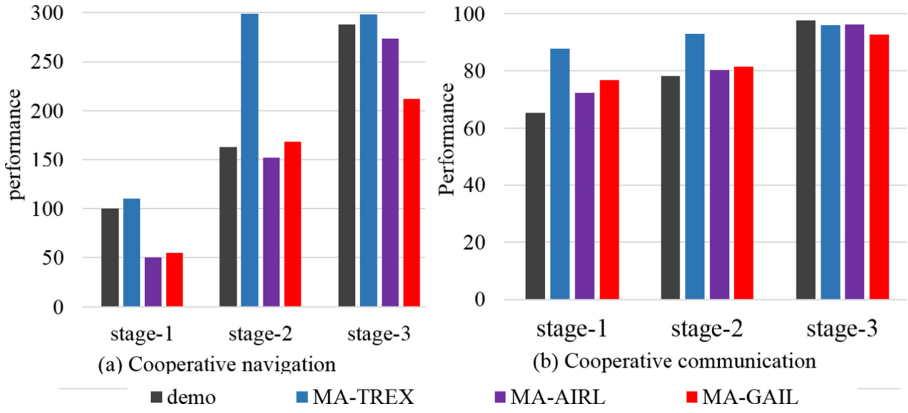


**Fig. 3.** Performance comparison of the sub-optimal expert trajectory in three stages of two collaborative tasks. Performance is obtained by calculating the average value of ground-truth rewards for 500 tasks.

the multi-agent versions of the famous algorithms generative adversarial imitation learning [10] and adversarial inverse reinforcement learning [7], which very representative in multi-agent inverse reinforcement learning.

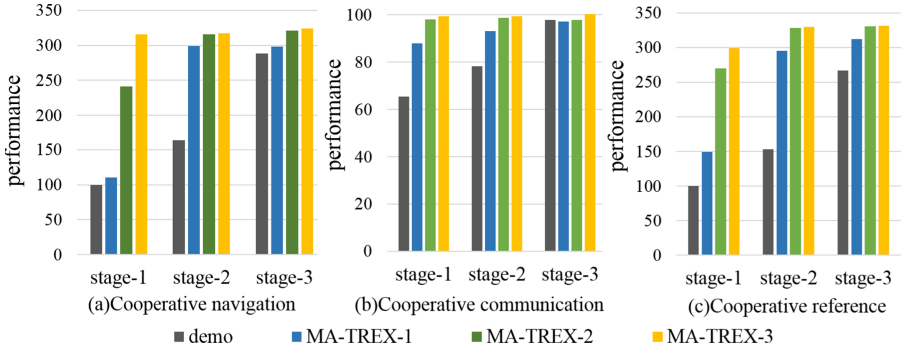


**Fig. 4.** Performance comparison of the MA-TREX performing multiple iteration training in three collaborative tasks. For all phase tasks, our MA-TREX algorithm basically achieves very high performance within 3 iterations of learning.

Without using iterative optimization, we tested the learning ability of the sub-optimal demonstrator in three different stages. As is shown in Fig. 3, in the second stage of Cooperative navigation and the first and second stages of Cooperative navigation, the performance of the strategy learned by the MA-TREX is significantly better than that of the demonstrator. Since the demonstrator is close to optimal in the third stage, there is no significant improvement in performance. In the first stage of Cooperative navigation, provided trajectories

are so poor that all algorithms are not effective, but our algorithm still has obvious advantages. MA-GAIL and MA-AIRL usually cannot achieve significantly higher performance than demonstrators, because they are just imitating demonstrations rather than inferring the intent of demonstrations. Experimental results show that in a multi-agent environment, the MA-TREX can effectively use preference information to surpass the performance of demonstrator.

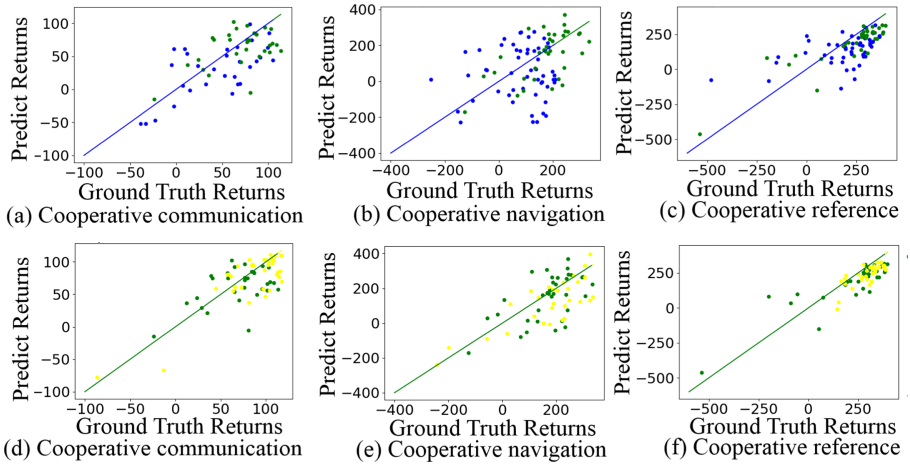


**Fig. 5.** The extrapolation rewards plot of the MA-TREX under three iterations in 3 collaborative tasks. The blue, green, and yellow points correspond to the trajectories of the first to third extrapolations, respectively. Solid line corresponds to the performance range of the trajectory. The $x$ axis is ground-truth returns, and the $y$-axis is predicted return. (Color figure online)

To verify that the reward function has the ability to learn using self-generated demonstrations, we compared the performance of the MA-TREX after multiple iterations of learning. In addition, in order to prove the rationality of the reward function combined with knowledge transfer, the new iteration only generates one-third of the initial demonstrations. As is shown in Fig. 4, in the first stage of Cooperative navigation and Cooperative reference, although the performance of the MA-TREX after the first reward extrapolation has improved, it is still far from the level of ground-truth reward. From the above experimental results, conclusions can be drawn as follows: 1) the ability to infer rewards is limited by the demonstrator; 2) the MA-TREX achieves high performance in all stages after iterative training through self-generated demonstrations; 3) the MA-TREX can effectively inherit the knowledge it has learned through iterative training and is no longer limited to the initial demonstrator.

In order to investigate the ability of the MA-TREX to extrapolate the trajectory of experts, we compared the ground-truth returns and predicted returns of the trajectory from the demonstration generated in the iteration. Figure 5 shows

the demonstrations generated by the MA-TREX at different iterations. It can be seen from the figure that with the iterative learning of the reward function, the positive correlation between the predicted reward and the ground truth reward gradually increases, which corresponds to the previous performance comparison experiment. For example, in Fig. 5 (b), the reward function after the second iteration (green dots) has a significant improvement in the positive correlation with the ground-truth reward compared to the first iteration (blue dots). It is consistent with the phenomenon of greatly improving performance occurring in Fig. 4 (a). In summary, the experimental results show that the reward function is learning in a more reasonable direction, and the performance gradually approaches the ground-truth reward level with iteration.

## 5 Conclusion

In this paper, we present a novel reward learning framework, MA-TREX, which uses sub-optimal ranked demonstrations to extrapolate agent intentions in multi-agent tasks. After combining the reward function with multi-agent deep reinforcement learning, it achieves better performance than the demonstrator, and it is also superior to the MA-AIRL and MA-GAIL methods. Furthermore, combining the knowledge transfer idea and using the model's self-generated demonstrations, the iterative optimization form of the MA-TREX is realized. And the reward function can reach the same level as the ground truth within three iterations by using self-generated demonstrations. In the future, one direction is to complete subsequent iterative learning without adding new labels.

## References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the Twenty-first International Conference, Machine Learning, Canada, vol. 69 (2004)
2. Amodei, D., Olah, C., Steinhardt, J., Christiano, P.F., Schulman, J., Mané, D.: Concrete problems in AI safety. CoRR abs/1606.06565 (2016)
3. Argall, B.D., Chernova, S., Veloso, M.M., Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst. **57**(5), 469–483 (2009)
4. Bradley, R.A., Terry, M.E.: Rank analysis of incomplete block designs: I. The method of paired comparisons. Biometrika **39**(3–4), 324–345 (1952)
5. Brown, D.S., Goo, W., Nagarajan, P., Niekum, S.: Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In: Proceedings of the 36th International Conference on Machine Learning, USA, vol. 97, pp. 783–792 (2019)
6. Christiano, P.F., Leike, J., Brown, T.B., Martic, M., Legg, S., Amodei, D.: Deep reinforcement learning from human preferences. In: Advances in Neural Information Processing Systems 2017, USA, pp. 4299–4307 (2017)
7. Finn, C., Christiano, P.F., Abbeel, P., Levine, S.: A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. CoRR abs/1611.03852 (2016)

8. Finn, C., Levine, S., Abbeel, P.: Guided cost learning: deep inverse optimal control via policy optimization. In: Proceedings of the 33nd International Conference on Machine Learning, USA, vol. 48, pp. 49–58 (2016)
9. Henderson, P., Chang, W., Bacon, P., Meger, D., Pineau, J., Precup, D.: OptionGAN: learning joint reward-policy options using generative adversarial inverse reinforcement learning. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, USA, pp. 3199–3206 (2018)
10. Ho, J., Ermon, S.: Generative adversarial imitation learning. In: Advances in Neural Information Processing Systems 2016, Spain, pp. 4565–4573 (2016)
11. Hu, J., Wellman, M.P.: Multiagent reinforcement learning: theoretical framework and an algorithm. In: Proceedings of the Fifteenth International Conference on Machine Learning USA, pp. 242–250 (1998)
12. Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., Amodei, D.: Reward learning from human preferences and demonstrations in Atari. In: Advances in Neural Information Processing Systems 2018, Canada, pp. 8022–8034 (2018)
13. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning (1994)
14. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in Neural Information Processing Systems 2017, USA, pp. 6379–6390 (2017)
15. Luce, D.R.: Individual choice behavior: a theoretical analysis. J. Am. Stat. Assoc. **115**(293), 1–15 (2005)
16. Ng, A.Y., Harada, D., Russell, S.J.: Policy invariance under reward transformations: theory and application to reward shaping. In: Proceedings of the Sixteenth International Conference on Machine Learning, Slovenia, pp. 278–287 (1999)
17. Ramachandran, D., Amir, E.: Bayesian inverse reinforcement learning. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, India, pp. 2586–2591 (2007)
18. Song, J., Ren, H., Sadigh, D., Ermon, S.: Multi-agent generative adversarial imitation learning. In: Advances in Neural Information Processing Systems 2018, Canada, pp. 7472–7483 (2018)
19. Torabi, F., Warnell, G., Stone, P.: Behavioral cloning from observation. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, Sweden, pp. 4950–4957 (2018)
20. Wang, S., Hu, X., Yu, P.S., Li, Z.: MMRate: inferring multi-aspect diffusion networks with multi-pattern cascades. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, USA, pp. 1246–1255 (2014)
21. Yu, L., Song, J., Ermon, S.: Multi-agent adversarial inverse reinforcement learning. In: Proceedings of the 36th International Conference on Machine Learning, California, vol. 97, pp. 7194–7201 (2019)
22. Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, USA, pp. 1433–1438 (2008)