Learning to Cooperate with Unseen Agents Through Meta-Reinforcement Learning

Extended Abstract

Rujikorn Charakorn, Poramate Manoonpong, Nat Dilokthanakul
Vidyasirimedhi Institute of Science and Technology (VISTEC)
{rujikorn.c_s19,poramate.m,natd_pro}@vistec.ac.th

ABSTRACT

Ad hoc teamwork problem describes situations where an agent has to cooperate with previously unseen agents to achieve a common goal. For an agent to be successful in these scenarios, it has to have cooperative skills. One could implement cooperative skills into an agent by using domain knowledge (e.g., goals, roles, and protocols) to design the agent's behaviours. However, in complex domains, domain knowledge might not be available. Therefore, it is interesting to explore how to directly learn cooperative skills from data. In this work, we apply meta-reinforcement learning (meta-RL) formulation in the context of ad hoc teamwork problem. Our experiments show that such a method could produce cooperative agents in two cooperative environments with different cooperative circumstances.

KEYWORDS

Ad-hoc Teamwork; Cooperation; Generalisation; Meta-Learning; Deep Reinforcement Learning

ACM Reference Format:
Rujikorn Charakorn, Poramate Manoonpong, Nat Dilokthanakul. 2021. Learning to Cooperate with Unseen Agents Through Meta-Reinforcement Learning: Extended Abstract. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 2 pages.

1 INTRODUCTION

In this work, we posit that a cooperative agent needs the ability to *generalise* its policy to the dynamics of unseen agents and this can be achieved with meta-reinforcement learning. Meta-learning paradigm considers learning an adaptive behaviour using data. At test time, the agent can utilise newly collected samples to shape its behaviour in an unseen environment. Intuitively, this method could work well in the ad hoc teamwork problem, where we would like our agent to adapt its behaviour such that it cooperates smoothly with an unseen partner. This approach will be useful when working with complex environments, where domain knowledge might not be available.

2 METHODS

2.1 Training setup

In this work, we consider two-player cooperative tasks, where our agent will have to cooperate with a set of agents from a pool, $\mathcal{P}$, of partner agents. Similar to Wang et al. [4] and Duan et al. [1], our meta-RL agent is implemented with an RNN and trained with a distribution of partners $\mathcal{P}$.

2.2 Environments

2.2.1 Lever Coordination Game (LC Game). We adapt the lever coordination game from Hu et al. [2]. This game consists of five levers; two players are playing at a time. At each timestep, each of the agents will select one of the levers. A positive reward of 1.0 is achieved if both of them selects the same lever. However, one of the agents is a pre-programmed agent, which keeps repeating a pattern of three actions. For example, one of the partner agents will keep choosing lever 1, 4, 5, 1, 4, 5 and, then, keep repeating these actions.

2.2.2 Discrete Speaker-Listener Game (DSL Game). We adapt the speaker-listener game from Lowe et al. [3]. The game consists of a speaker, a listener and five landmarks. At each timestep, the reward of 1.0 will be randomly assigned to one of the landmarks. The speaker can see where the reward is and has to speak to the listener by emitting a one-hot vector to the listener. The listener then chooses one of the landmarks as a target and receive a positive reward if it selects the correct target. Similar to the LC game, the speaker is a pre-programmed partner agent. Each pre-programmed agent has a unique one-to-one mapping between landmarks and the five-dimensional one-hot vectors.

3 EXPERIMENTS

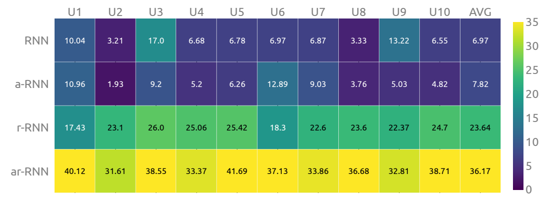
3.1 Emergence of Cooperative skills

We test meta-RL agents in the LC game and the DSL game with four model variations, which have different input features to the RNNs. The models are:

- RNN: Baseline recurrent neural network (RNN) with the standard observation.
- a-RNN: Recurrent neural network with the standard observation and the previous action of the agent a_{t-1}.
- r-RNN: Recurrent neural network with the standard observation and the reward r_{t-1}.
- ar-RNN: Recurrent neural network with the standard observation, the previous action a_{t-1} and the reward r_{t-1}.

The results from Fig. 1 show that recurrent architectures with the previous reward as an additional input (r-RNN and ar-RNN) outperform the ones that do not have the feature as input (RNN and a-RNN). Specifically, when matched with unseen agents, ar-RNN has the highest average score of 36.2 in LC game and 34.4 in DSL game. An agent with the complete knowledge of the partner's

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.
© 2021 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.



	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	AVG
RNN	10.04	3.21	17.0	6.68	6.78	6.97	6.87	3.33	13.22	6.55	6.97
a-RNN	10.96	1.93	9.2	5.2	6.26	12.89	9.03	3.76	5.03	4.82	7.82
r-RNN	17.43	23.1	26.0	25.06	25.42	18.3	22.6	23.6	22.37	24.7	23.64
ar-RNN	40.12	31.61	38.55	33.37	41.69	37.13	33.86	36.68	32.81	38.71	36.17

Figure 1: Test-time score when matched with unseen agents in the LC game. Each cell in the heatmap shows an average score over five training seeds for a pair of a trained agent and an unseen agent. ar-RNN and r-RNN have high test-time scores while a-RNN and r-RNN do not. The results are similar in the DSL game.

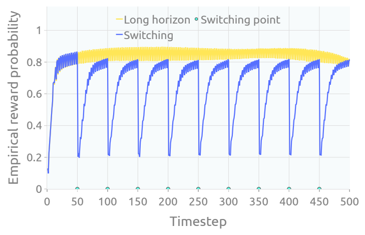


Figure 2: Continual Adaptation in the LC game. We test whether the meta-RL agent can adapt to multiple partners within the same trajectory.

pattern would theoretically get the score of 50.0. However, RNN and a-RNN have a lower average score than a random policy that would have an average score of 10.0.

These experiments indicate that ar-RNN and r-RNN can cooperate effectively while a-RNN and RNN cannot. We interpret these results as follow:

- The reward signal r_{t-1} is necessary for the emergence of cooperative skills. This is because the agent needs to know whether or not its current strategy is suited to the current partner.
- The previous action input a_{t-1} helps the agent to cooperate quicker when used in combination with the reward signal r_{t-1} because this feature can be used by the RNN to correlate the action with the reward. This makes it easier for the RNN to identify what is the correct action during adaptation.

3.2 Continual Adaptation

When an agent is deployed into the real world, test-time scenarios might differ from the ones that are used during the training. In this section, we examine the ability of a meta-RL agent to extrapolate under unexpected situations including working under longer horizon and partner switching.

3.2.1 Longer horizon. In this experiment, the agent is tested in trajectories with horizon length of 500 in the LC game and 2,000 in the DSL game. This is much longer compared to the horizon length of 50 that the agent is trained with. We find that meta-RL agent is robust when it performs to longer horizon length in both environments. The performance is stable throughout the entire trajectory.

3.2.2 Partner Switching. Working with only one partner throughout an entire trajectory might not be realistic when considering real-world applications where behaviour or partner switching could occur over the course of the task. Here, we investigate meta-RL ability to adapt under this circumstance *without explicitly trained or designed* for this situation.

In this experiment, the partner agent is changed periodically. The periods are to be 50 timesteps in both games and also 500 timesteps in the DSL game to highlight the adaptation speed. The results from the LC game are shown in Fig. 2. As can be seen from the results, the meta-RL agent can adapt flexibly even though it has been trained to adapt with only one partner per episode. We see different adaptation behaviours when the agent is already adapted to one partner. Specifically, it adapts faster to the first partner compared to later partners in both environments. We think this is because the agent is optimised to adapt to only one partner in an episode.

3.3 Limitations

So far, the experimental results have shown that meta-RL can produce ad hoc agents with preferable attributes. In this experiment, we want to find the limitations of the agents and pitfalls that one needs to avoid when using this training method.

3.3.1 Quantity. First, we study the impact of the number of training partners. We consider the number of training partners from the set of {5,10,15,20}. We observe that the ad hoc teamwork performance and cooperation get better as we increase the number of training partners. This indicates that the quantity of training partners has a direct impact on the adaptation of the agent.

3.3.2 Diversity. Next, we study the impact of diversity of the training partners. Instead of randomly selecting training partners from the pool of all possible agents, we select the training partners such that they only come from a specific part of a behaviour space. This selection process skews the partner selection process such that some lever sequences or mappings will appear more often, while some are not presented during the training process at all. This is similar to out-of-distribution testing in supervised learning.

The training score of meta-RL agent does not deteriorate when trained under this skewed distribution. However, we notice that the test-time performance reduced significantly in both environments. This suggests that lack of diversity is detrimental to the generalisation of the meta-RL agent causing the agent to be less adaptive to unseen agents during test-time.

REFERENCES

[1] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779* (2016).
[2] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. 2020. "Other-Play" for Zero-Shot Coordination. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 4399–4410.
[3] Ryan Lowe, Yi I 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*. 6379–6390.
[4] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. 2016. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763* (2016).