The Non-Obedient Agent (Course Track)

Zihan Yu
Student ID: 2019011194 (Undergraduate)
Department of Electronic Engineering
Tsinghua University
yuzh19@tsinghua.org.cn

Ruize Zhang
Student ID: 2019011189 (Undergraduate)
Department of Electronic Engineering
Tsinghua University
zhangrz19@mails.tsinghua.cn

Liu Cao
Student ID: 2020011234 (Undergraduate)
Department of Electronic Engineering
Tsinghua University
l-cao20@mails.tsinghua.cn

Chenyu Wang
Student ID: 2019013206 (Undergraduate)
Department of Electronic Engineering
Tsinghua University
chenyu-w19@mails.tsinghua.cn

Abstract

Imaging two layers (i.e. agents) in a multi-agent system, the upper one can see the whole map but may miss some important details; while the lower one can only see its neighborhood situation but has no global knowledge. These two agents cannot share their observations directly due to some reason such as a restricted information channel, but can only exchange limited information through a command from upper to lower and an action from lower to upper. How can these two agents cooperate to accomplish a search task? We solve this problem with an adversarial reward design, where the upper agent manages to make the lower one obey itself; while the lower agent is rebellious and tries not to obey the upper one. Experiments demonstrate the effectiveness of our design: Without the upper agent, the lower one can only reach the goal through an ineffective route; while with the upper agent's help, the lower one can take the shortest route to the goal.

1 Introduction

Consider such a scenario: A robot dog is sent to a firing building to conduct a rescue mission, while a command center is directing it from outside. The command center has a floor plan of the building, the current location of the robot dog obtained from GPS or so on, and the exact location of the trapped people at which the robot dog should arrive. With a lot of computing power that robot dogs don't have, the command center can plan a rescue route for it, so the robot dog only needs to obey the instructions sent by the command center to complete the task - Wait, there is something wrong with the route provided by the command center, the road is collapsed due to fire and creating an impassable hole. At this moment, the robot dog won't be obedient, but choose a movement direction that is different from the instruction provided by the command center, and the command center can detect these anomalies through the robot dog's disobedience, and re-planning a new rescue route.

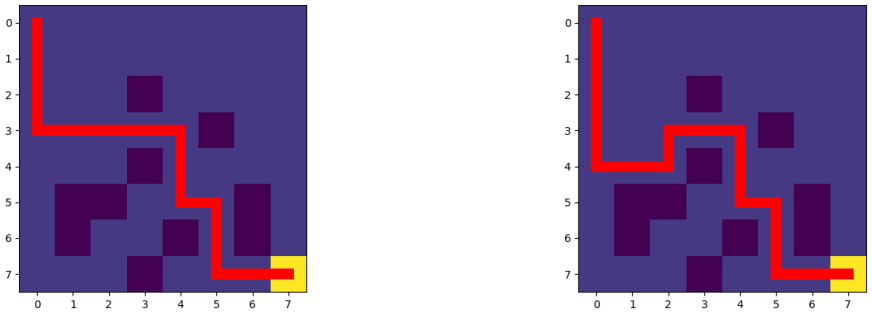
That's not a problem made up out of thin air. In a centralized system, the upper commander can observe the global but maybe out-of-date information; while the lower soldier can observe the firsthand but restrict-to-neighboured information, and due to some reasons, such as the complex electromagnetic environment or limited capacity of the channel that limits the data transmission rate between these two agents, they cannot share their observations directly and conduct a decision based on it - The lower agent can only get something with limited information, such as an action assigned by the upper one; while the upper agent can only observe the action chosen by the lower one and further

Preprint. Under review.

guess what it observed in its neighborhood situation. If fact, this situation is widely observed in many scenarios, such as UAC (Unmanned Aerial Vehicle) cluster[1] and battlefield communication[2].

In this work, we focus on such a problem: Imaging two layers (i.e. agents) in a multi-agent system, the upper one can see the whole map but may miss some important details; while the lower one can only see its neighborhood situation but has no global knowledge. How can these two agents cooperate through a restricted information channel to accomplish a search task? Specifically, the upper agent can observe the lower one's current position and the destination, and assign an action for the lower one. However, there are some randomly generated holes in the scenario that the upper one is unaware of, which can be observed by the lower agent only when it comes near to them.

This problem is non-trivial due to two challenges: 1) The lower agent cannot obey the upper one all the time since its command may be harmful (e.g., guide it into a hole), but justify whether the upper one's command is trusty and choose an appropriate action when the upper one is unreliable. 2) The upper agent must learn from the lower agent's non-obedience and plan an effective path for the lower one. To solve these challenges, we design their reward in a specific way: Apart from the reward that can be gotten from reaching the goal, the upper agent can get additional reward when the lower agent chooses an action that obeys (i.e. equal to) its command, while the lower agent can get additional reward when it chooses an action that does not obey (i.e. unequal to) the upper one's command. Through this adversarial reward design, we encourage the upper one to plan an effective and efficient route for the lower, and encourage the lower to be rebellious when the upper one's command seems useless or even harmful. Experiments demonstrate the effectiveness of our design: Without the upper agent, the lower one can only reach the goal through an ineffective route; while with the upper agent's help, the lower one can take the shortest route to the goal, as is shown in Figure 1.



(a) with upper agent

(b) without upper agent

Figure 1: Routes (red) with and without the upper agent. The yellow cells are goals and the dark cells are holes.

2 Method

2.1 Environment

We use FrozenLake environment from Gymnasium in our project. In this environment, the lake is described as a grid world, most of which is frozen, but there are a few holes where the ice has melted. If the agent step into one of the holes, it will fall into the freezing cold water. There is a target position in the grid. If the agent reaches the target position, it will get a reward. In our project, the map size is set to be 8*8.

In addition, the environment can be set to be slippery, which means the agent is more likely to head in the corresponding direction, but it also has the chance of heading each side apart from the corresponding direction.

The reward schedule of the environment is described as follows:

- Reach goal: $r_e = +5$

- Reach hole: $r_e = 0$
- Reach frozen: $r_e = 0$

2.2 Agent

There are two kinds of agents in our project: namely the **upper agent** and the **lower agent**. The **upper agent** has a broader but rougher view of the environment, while the **lower agent** has a more narrow but more precise view of the environment. Only the **lower agent** makes actions to interact with the environment. They cooperate with each other to help the **lower agent** avoid the holes and reach the target position.

2.2.1 Upper Agent

Observation space The Observation shape of the **upper agent** is (4,). The **upper agent** can observe the location(x and y) of the **lower agent** and the location(x and y) of the target position, i.e:

$$\mathbf{O}_u = [x_l, y_l, x_{goal}, y_{goal}] \tag{1}$$

However, it doesn't know where the holes are.

Action space The action shape of the **upper agent** is (1,) in the range $\{0, 3\}$ indicating the **upper agent** gives the targeting direction to the **lower agent** as an advice. Caution that the action of the **upper agent** will not interact with the environment directly.

Reward The reward function of the **upper agent** is divided into two parts. The first part is the environment's reward. The second part is whether the advice given to the **lower agent** is helpful. To balance those two parts, we add a weight to the second part.

$$r_u = r_e + w \times f(a_u == a_l), \tag{2}$$

where $f(\cdot)$ denotes the metrics over the similarity between two actions.

2.2.2 Lower Agent

Observation space The Observation shape of the **lower agent** is (9 + action.space,). The **lower agent** can observe the 3*3 area around it, which means that it knows whether there are holes or the target nearby. Moreover, it can gets the advice from the **upper agent**, namely the action of **upper agent**, i.e:

$$\mathbf{O}_l = [square.flatten, action.space] \tag{3}$$

For instance,

Action space The action shape of the **lower agent** is (1,) in the range $\{0, 3\}$ indicating which direction to move for the **lower agent**.

Reward The reward function of the **lower agent** is the environment reward.

$$r_l = r_e - w \times f(a_u! = a_l) \tag{4}$$

3 Experiment Results

3.1 Probabilistic Experiments

In this part, we take the network output of the probability over different actions.

For the upper agent, the objective at time t is respect to the reward r_e gotten from the environment and a discount w of the KL divergence over two action spaces, i.e:

$$r_u = r_e + w \times \mathbf{KL}_{a_u||a_l} \tag{5}$$

For the lower agent, the objective at time t is the naive reward r_e plus the **non-obedience**:

$$r_l = r_e - w \times \mathbf{KL}_{a_u||a_l} \tag{6}$$

3.1.1 Results

3.1.2 Ablation study

3.2 Deterministic experiments

In this part, we change the action output to be deterministic, instead of a distribution over the action space, and use the characteristic function on different agents' actions to represent the obedience, i.e:

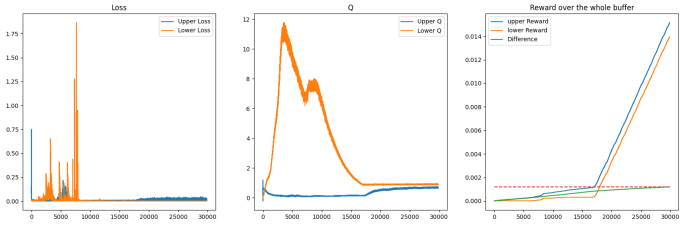
$$r_u = r_e + w \times \mathbf{I}_{a_u == a_l} \tag{7}$$

For the lower agent, the objective at time t is the naive reward r_e:

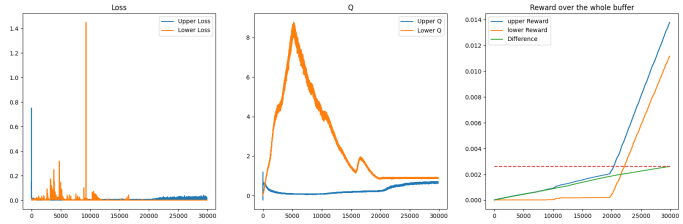
$$r_l = r_e \tag{8}$$

3.2.1 Results

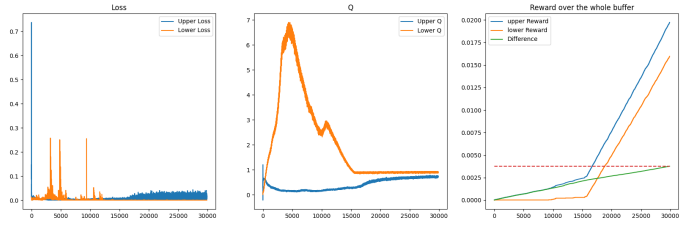
After tuning the weight controlled in the range below 0.05, we ensure the success of finding the goal and have gotten some results in Figure 2. We can see that the Q values for both agents show the same trend over the training steps. At a training step near 17000, the Q value of lower agent levels off, which represents the convergence of its training process, meanwhile the Q value of upper agent begins to increase to a value slightly lower than that of lower agent, somehow to our surprise. This means that the training process of the two agents is carried out in different stages, and the upper agent learns to find a way to goal due to the better performance of lower agent.



(a) $weight = 0.01$



(b) $weight = 0.02$



(c) $weight = 0.03$

Figure 2: Results over three weights calculated on the upper agent reward

3.2.2 Ablation Study

With the question about the order in which the Q values of two agents tend to be stabilized, we directly remove the upper agent and use only the local observation of the lower agent to take action,

get reward, and update the parameters of networks. Based on the observation in Figure 3, we can see the trend of the Q value are as the same as the variation in Figure 2. This reinforces the fact that under the deterministic policy, lower agent's policy guide the upper to improve, which exhibits the **non-obedience** to some extent.

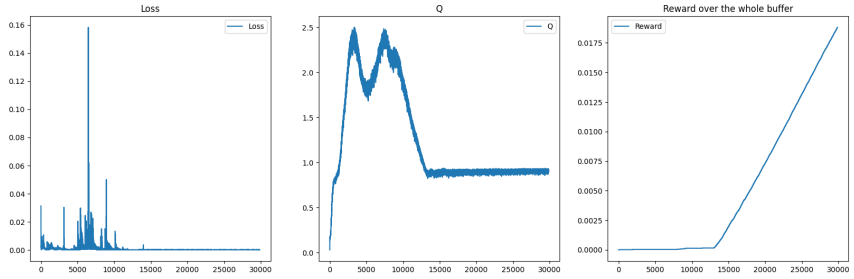


Figure 3: Results without upper agent

However, when we go deeper into the routes generated by lower agents with and without the upper agent, the policies are not the same. Upper agent somehow helps the lower one take a more direct route to find the goal.

As Figure 1 shows, when without upper agent, lower agent prefers to arrive at a more distinguished state, for example it prefers to go down at (0, 3) instead of go right because the observations over the state (0, 3) and (1, 3) are similar. Considering the environment that upper agent exists, lower agent can take the upper action as a label to distinguish different states, instead of taking the upper action as advice or guidance. Therefore it chooses to go right at the same state (0, 3).

4 Related Work

- MAAC[3]: An Multi-agent RL algorithm proposed by OpenAI, which serves as a pioneer in this field.
- MAPPO[4]: A promising multi-agent verson of PPO proposed by Yu et al., which may help we understand this problem better.
- HDRL[5]: A hierarchical architecture proposed by Kulkarni et al. to deal with the sparse and delayed reward signals. This hierarchical architecture is kinds of similar to our focused idea.

5 Conclusion

In this work, focusing on a non-obedient agent problem, we propose an adversarial reward design to make the upper agent manage to make the lower one obey itself, and make the lower agent rebellious and try not to obey the upper one. Through this design, the upper one learns to plan an effective and efficient route for the lower, and the lower violate the upper's command when the environment is disagree with the command. Experiments demonstrate the effectiveness of this design.

There is some direction that can be explored further in the future, including but not limited to:

- Can we explicitly know how the lower agent's non-obedience influence the upper one? For example, can the upper one infer the location of holes that make the lower violate its command?
- In this work, the lower agent is memoryless, i.e. it can only observe its neighbor state at the current time. However, if it will be helpful to make the lower agent remember the environment that it has passed through?
- In our group's previous discussions, we planned the network structure of the lower agent input observation and output action along with belief, which is in the range (0, 1). However, trouble was encountered when considering how to bring such a defined belief level into the

reward calculation and the gradient backpropagation. After today's lecture about extrinsic and intrinsic rewards, we can treat the belief as a symbol of intrinsic reward in the future.

References

[1] Yang J, You X, Wu G, et al. Application of reinforcement learning in UAV cluster task scheduling[J]. Future generation computer systems, 2019, 95: 140-148.

[2] Adamy D. EW 103: Tactical battlefield communications electronic warfare[M]. Artech House, 2008.

[3] Lowe, Ryan, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments." arXiv, March 14, 2020. https://doi.org/10.48550/arXiv.1706.02275.

[4] Yu, Chao, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. "The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games." arXiv, November 4, 2022. https://doi.org/10.48550/arXiv.2103.01955.

[5] Kulkarni, Tejas D., Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. "Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation." arXiv, May 31, 2016. http://arxiv.org/abs/1604.06057.