Adaption of Cooperative Agents to Human Play Through Deep Reinforcement Learning and Transfer Learning

Ryoma Usui s1280123 Supervised by Prof. Maxim Mozgovoy

Abstract

In action games, there are situations where cooperative NPCs do not move as intended. Since people have different play styles, optimal behavior as a cooperative agent varies among players. There is MADDPG method to train multiple agents with interacting each other, and transfer learning method to reuse trained data. The goal is to create supportive agents, which can improve player scores by adapting cooperative agents to the individual's playstyle through learning. Additionally, it's also important to minimize the amount of human play required for learning. To achieve this, MADDPG self-training is executed at first, then do transfer learning involving human play has done. It was a success to learn agent action to improve human play score with few trials of human play compared to learning from zero.

1 Introduction

Reinforcement learning [7] is a type of machine learning that uses a method to maximize the reward from the environment based on the actions of an agent. There are some approaches to use human action for learning, and it is often called Human-In-The-Loop (HITL) learning [10]. The main approach is imitation learning. The purpose of the algorithm is to imitate human play, so that agent can perform high score quickly. There is one agent per human. There are various algorithms such as behavior cloning [9] and inverse reinforcement learning [8]. However, it seems that few studies have examined methods for learning multiple agents based on single human play. These methods do not fit to implement supportive agent in multiple actor games. Therefore, I focused on using HITL to learn multiple agents that adapt to the movements of playing humans. The HITL learning is necessary to train an agent that performs optimally for individuals, but the challenge is that human movements are irregular, which could lead to a long learning time or not able to learn at all. Hence, this study proposes methods for learning even with human unstable policies.

2 Background

These are the main ideas utilized to the implementation.

2.1 Deep Q Network (DQN)

Q-learning and Deep Q Network (DQN) [5] is frequently used in RL domain. Q-learning is a method to save value of a certain action taken in a certain state. This value, Q-value is updated for each action and used to select actions.

DQN estimates action value function by neural network. Experience replay buffer is used to update the network. It is tuple (state s, action a, reward r, next state s'). There are two networks: Q Network and Target Network. Neural networks have parameters of weight of each neuron. The learning aims to maximize reward from environment by adjusting these parameters. Q Network is updated by batch randomly selected from buffers, and Target Network is reflected to Q Network with certain rate. It helps to reduce overestimation.

$\overline{Q^*}$ is target function, and y is target Q-value.
The network parameter θ is updated to minimize this loss function.

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}[(Q^*(s,a \mid \theta) - y)^2],$$
$$\text{where} \quad y = r + \gamma \max_{a'} \overline{Q^*}(s', a')$$

The loss function represents the error between the predicted values output by the model and the actual measured values.

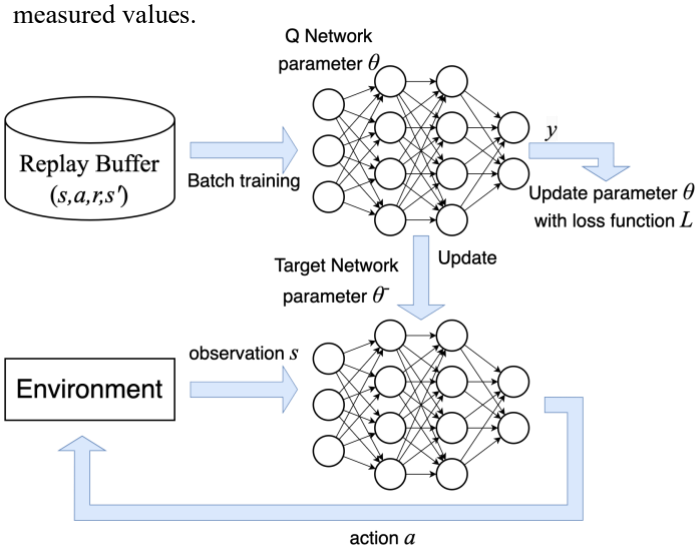

Figure 1. Deep Q Network

2.2 Policy Gradient

Another popular method is Policy Gradient [6]. It aims to maximize the objective function by adjusting the network parameters θ of actor policy π.

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[\nabla_\theta \log \boldsymbol{\pi}_\theta(a \mid s) Q^\pi(s,a)]$$

It is also can be applied to deterministic policy μ.

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\mathcal{D}}[\nabla_\theta \boldsymbol{\mu}_\theta(a \mid s) \nabla_a Q^\mu(s,a)|_{a=\boldsymbol{\mu}_\theta(s)}]$$

Deep Deterministic Policy Gradient (DDPG) is variant of DPG. Deep neural network is used to solve policy μ and critic Q^μ. DDPG also makes target network as DQN.

2.3 Actor-Critic

Actor-Critic [7] is combination of DQN and Policy Gradient idea. It thinks the model separately as the actor, which selects actions, and the critic, which evaluates the value function of the actor and trains it. Generally, this reduces learning instability. In many cases, policy gradient methods are used for actor learning.

2.4 MADDPG

Multi-agent DDPG (MADDPG) [1] is extended DDPG to multi-agent environment. The main idea is Centralized training and Decentralized execution. Agent on his model use only own observation when executing, and critic to output Q-value has extra information for other agents. It can be adaptable to both cooperative and competitive environments.

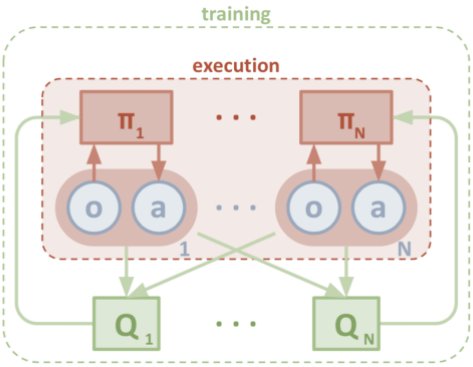

Figure 2. Overview of MADDPG (adopted from [1])

Each network can be updated as follows.

Actor update

The objective function for each agent i to update actor policy.

$$\nabla_{\theta_i} J(\boldsymbol{\mu_i}) = E_{\boldsymbol{x},a\sim\mathcal{D}}\left[\nabla_{\theta_i}\boldsymbol{\mu_i}(a_i \mid o_i)\nabla_{a_i} Q_i^\mu(\boldsymbol{x}, a_1, \ldots, a_N)\big|_{a_i=\boldsymbol{\mu_i}(o_i)}\right]$$

$\mathcal{D}$ is batch of replay buffers.

Critic update

Input for critic is all state information in of other agents and all their actions. x could consist of all agents' observations, for example.

$$\mathcal{L}(\theta_i) = \mathbb{E}_{x,a,r,x'}[(Q_i^\mu(\boldsymbol{x}, a_1, \ldots a_N) - y^2)]$$
$$where \quad y = r_i + \gamma Q_i^{\mu'}(\boldsymbol{x}', a_1', \ldots, a_i, \ldots, a_N')|_{a'=\mu'(o)}$$

2.5 Fine Tuning

Fine tuning is a type of transfer learning, which is the process of reusing a neural network model trained on one task for a similar task. This can significantly reduce the time needed to train the model compared to from scratch. One way to do this is to freeze the parameters in the lower layers and only train the upper layers. Low layers mostly have important weight that should not be changed.

3 Method

These are the steps of the experiment to create efficient cooperative agents using human play.

3.1 Pre-training

Conducting reinforcement learning with human-in-the-loop from the beginning takes too much time. In this method, agents will be trained by themselves with simple MADDPG (Fig. 3, above), and then do HITL training. Actions are randomly selected for the certain steps.

3.2 Human-In-the-loop training

Pre-trained model is used for HITL training (Fig. 3, below) by fine tuning method. Since the goal is adaption for human specific play, single person will be attempted to the training.

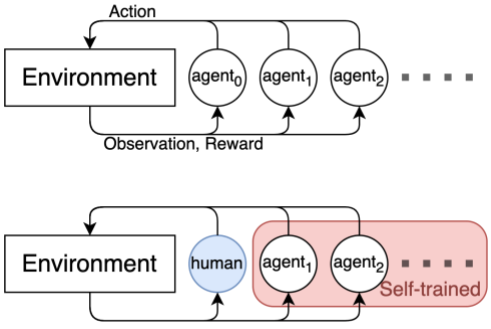

Figure 3. Training process
Above: self-training; below: HITL-training

While this HITL-training, parameters of some lower layers of model network is being frozen.

3.3 Evaluation

It is difficult to measure mean expected reward for models accurately because human intention can be involved in the results. To minimize the evaluator's intention, the two policies to compare will be randomly selected. One is before and one is after human-in-the-loop learning. This should reduce the bias in the player's movements during evaluation. The evaluation should be done by the same person as HITL training.

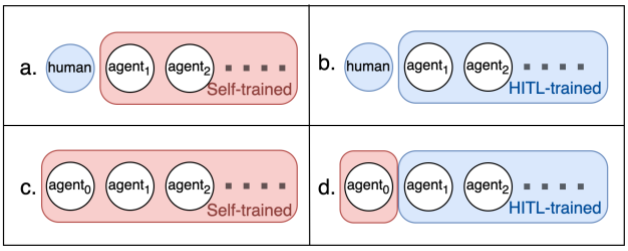

Figure 4. Situations for evaluation

Figure 4 shows models in evaluation. Evaluation a and c use self-trained model, which is not HITL-trained, and b and d use HITL-trained model. a and b is

3.4 Expected Result

There is limit to human reflex, so it is likely that the average reward of plays without humans will be higher. The method will be successful if mean reward of human play in HITL-trained model (Fig. 4b) is higher than in pre-trained model (Fig. 4a). Furthermore, since HITL-trained model should be optimized to human play, performance when using pre-trained agent instead of a human through HITL-trained model (Fig. 4d) is expected to decline compared to using pre-trained model for cooperative agents (Fig. 4c).

4 Experiments

In this section, previously mentioned method is applied to simple multi-agent environment.

4.1 Environment

A python library “PettingZoo” provides multi-agent reinforcement learning environment. The environment "Simple_tag" in PettingZoo is customized for this experiment. In this environment, there are 3 adversary actor and single prey actor. Each time predator and prey collide, all predators get positive rewards, and the prey gets a negative reward. They aim to maximize their rewards. Predator, prey, and an obstacle are displayed with red, green, and gray respectively. A predator with the first index is changed to purple for clarity that it is human player during HITL training.

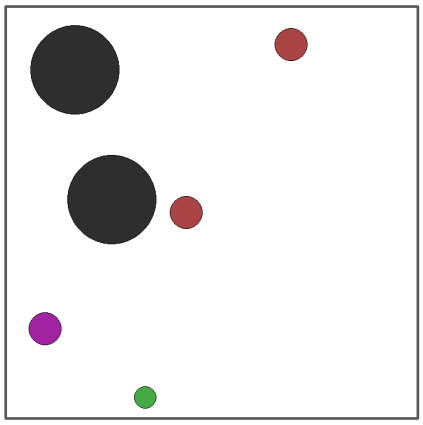

Figure 5 Customized simple_tag environment

Each actor and obstacle will rebound each other. Their action space is discrete; movements up, down, left, and right. Since prey could move faster than predators, predators must cooperate to catch the prey.
Prey also gets negative rewards if it moved too far from the center, so that it won’t run away straight forever.

4.2 Training Flow

Pre-training
The pre-training is done with mostly same model structure and hyperparameters as original MADDPG experiment [1], except that activation function is tanh. This table shows the hyper parameters for the training. tau is update rate from Q network to Target network.

Hyper parameters	Value
Episode number	30000
Episode length	100
Learn interval	100
tau	0.02
gamma	0.95
batch size	1024
Learning rate	0.01

Fig. 6 shows how predator agents in pre-trained model can cooperate to catch prey. In this episode, direction to prey from predator_1 was upper right at t=0. However, the agent decided to move upper left to surround the prey with other predators. If DDPG is

used, actors won't develop to consider other predators.

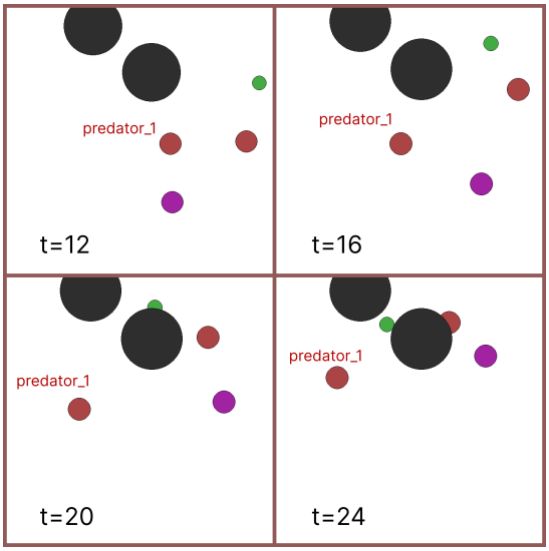

Figure 6 Agent movement after pre-training

Human-In-the-Loop training

After pre-training human-in-the-loop training has done. In this training, actor policy of prey is fixed to make comparation of predators' performances before and after this learning.

This is the parameters for the training. The batch size is smaller than pre-training so that it can learn precise policy.

HITL training parameters	Value
Buffer number	5000
tau	0.5
gamma	0.95
batch size	128
Learning number	100
Learning rate	0.1

100 trials recorded and each episode is set to be 50 steps, so 5000 replay buffers are used for HITL-training.

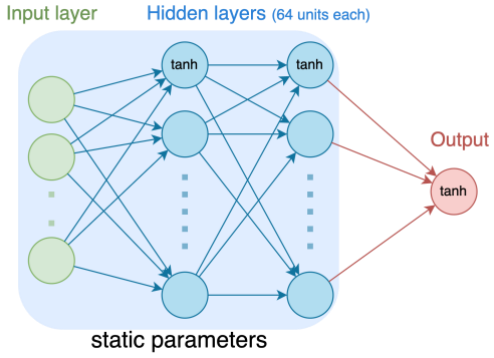

Figure 7. Network of while HITL-training

All actor and critic networks have shown in figure 7. These have 2 dense hidden layers, which receives all inputs from previous layer. They have 64 units for each. Activation function is tanh. While HITL training in this experience, all of weights are frozen except output layer, shown with red node. Without freezing, the model performance gets extremely bad.

5 Result

This table shows sum of rewards of predators through trials.

	Pre-trained model	HITL trained model
Human play (500 trials)	6740	8310
No human (1000 trials)	262210	105680

Reward of predators with pre-trained model and HITL-trained model are compared when predator_0 is a pre-trained agent or a human.

After HITL-training, predator performance decreased by more than 50%. On the other hand, reward with human play improved slightly.

6 Discussion

Due to uncertain evaluation method, it cannot be proved that HITL-trained model is always better than pre-trained model under human play. However, even though the model's performance without human has decreased significantly, human play performance did not decrease. It can be said that it adapted to human play.

7 Conclusion

This research aimed to implement cooperative agents with MADDPG algorithm and fine-tuning. The main part of training is Human-In-The-Loop, and fine-tuning is successful to reduce the trial number for human to play.

8 Future work

In this experiment, the results were not very reliable because there was only one human participant. The result can be changed if play style was different. Also, it took over 3 hours, to evaluate 1000 trials for one human. In future, Human-In-The-Loop training

process can be adapting the frameworks such as COGMENT [2], which enables many humans to participate in the learning process online.

In addition to the Fine-tuning method used this time, I would also like to try other transfer learning method or using inverse reinforcement learning methods such as GAIL [5] to reduce the number of human plays required for learning by imitating human play during the learning process.

Acknowledgement

I would like to thank Prof. Maxim Mozgovoy for advice on designing experiments.

References

[1] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments" in *Advances in neural information processing systems 30*, 2017, pp. 6379-6390.

[2] N. Navidi, F. Chabo, S. Kurandwa, I. Lutigma, V. Robt, G. Szrftgr, and A. Schuh, "Human and Multi-Agent collaboration in a human-MARL teaming framework" in *arXiv e-prints*, 2020, arXiv:2006.07301 [cs.AI]

[3] C. Tan, F. Sun, T. Knong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning" in *Internatial conference on artificial neural networks,* 2018 September, pp. 270-279

[4] J. Ho, S. Ermon, "Generative adversarial imitation learning" in *Advances in neural information processing systems 28,* 2016, pp. 4567-4573

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves,M. Riedmiller, A. K. Fidjeland, G. Ostrovski, "Human-level control through deep reinforcement learning" in *Nature 518*, 2015, pp. 529–533

[6] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation" in *Advances in neural information processing systems*, 2000, pp.1057-1063

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, Cambridge, MIT press, 1998

[8] A. Y. Ng and S. Russell. "Algorithms for inverse reinforcement learning" tn *ICML*, 2000

[9] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration" in *Robotics and autonomous systems, 57(5)*, pp.469–483, 2009

[10] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He. 2021. "A Survey of Human-in-the-loop for Machine Learning" in *arXiv preprint*, 2021, arXiv:2108.00941