NEURON AS AN AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Communication methods on multi-agent reinforcement learning (MARL) so far relies on a trusted third party (TTP) which distributes incentive to agents, and hence cannot be applied into a peer-to-peer environment. We propose *Neuron as an Agent* (NaaA) for incentive distribution in MARL without TTP with two key ideas: (i) inter-agent reward distribution and (ii) auction theory. The reason why we introduce auction theory is inter-agent reward distribution is insufficient for optimization. An agent in NaaA maximizes its profit, difference between reward and cost. As a theoretical result, we show that the auction mechanism has an agent autonomously evaluates counterfactual return as other agent's value. NaaA enables us to trade of representation in peer-to-peer and regard a unit in a neural network as an agent ultimately. Finally, we confirm that optimization with the framework of NaaA leads to better performance of RL, with numerical experiments. Specifically, we use a single-agent environment from Open AI gym, and a multi-agent environment from ViZDoom.

1 INTRODUCTION

After intelligence with reinforcement learning beat humans (Tesauro, 1995; Mnih et al., 2015; Silver et al., 2016), reinforcement learning has been expected to be applied into industry such as stock trade, autonomous cars, smart grid and IoT. In a world which realized the industrial application in the future, various types of companies will own their agents to improve their revenue. The situation can be regarded as one that each agent are independently solving problems of partially observed Markov decision process (POMDP).

Although agents in the companies are closed to maximize their own reward, if the agents exchange their own information each other, entire revenue will be improved more than individuals. Using economic metaphor, similarly to a supply chain of diamond from a company which collects raw material, one which processes it to one which sells it, if we can realize representation learning in the each layers as social division, a single company can yield revenue more than one by itself. Thus, this paper aims to realize a society in which stakeholders which can have a conflict of interest trade their own information.

We regard the situation as communication in multi-agent reinforcement learning (MARL), addressed by several existing methods such as R/DIAL (Foerster et al., 2016) and CommNet (Sukhbaatar et al., 2016). CommNet is a state-of-the-art of MARL which considers communication among agents, and the feature is learning among agents with backpropagation.

In the case when we are trying to consider MARL in which different stakeholders make different agents which communicate each other, it needs design of incentive distribution (e.g., monetary payment) and a framework without *trusted third party* (TTP). TTP is a neutral administrator which assumes distribution of reward for all the participants, supposed implicitly by most of existing literatures with regard to MARL (Agogino & Tumer, 2006; Foerster et al., 2016; Sukhbaatar et al., 2016). While TTP is required to be neutral against all the participants, several configuration of peer-to-peer trade such as inter-industry and -country trade cannot prepare TTP. If untrusted third party assumes reward distribution, it can undesirably make reward for partial participants higher than necessity.

To the best of our knowledge, no existing literatures discuss the reward distribution on the configuration above. Since CommNet assumes an environment which distributes a uniform reward to all the agents, in the case distributing limited reward in supply such as money, it causes *Tragedy of the Commons* (Lloyd, 1833) in which contributing agents' reward will reduce due to participant of

free riders. Although there are several MARL methods which distributes reward depends on their contribution such as QUICR (Agogino & Tumer, 2006) and COMA (Sukhbaatar et al., 2016), they suppose the existence of TTP, and hence it cannot be applied into our situation.

Our proposed method, *Neuron as an Agent* (NaaA) extends CommNet to realize incentive distribution in MARL without TTP with two key ideas: (i) inter-agent reward distribution and (ii) auction theory. The reason why we introduce auction theory is inter-agent reward distribution is insufficient for optimization. An agent in NaaA maximizes *profit*, difference between reward which it receives and cost which it redistributes to other agents. If we optimize the framework naively, we obtain a trivial solution that agents make their cost zero to maximize the profit. Then, NaaA employs game design with auction theory to keep cost being smaller than necessity. As a theoretical result, we show that an agent autonomously evaluates *counterfactual return* as other agent's value. Counterfactual return equals to discounted cumulative sum of counterfactual reward (Agogino & Tumer, 2006) which QUICR and COMA distribute. NaaA realizes reward distribution which make it pareto improvement more than inter-agent reward distribution.

NaaA enables us to trade of representation in peer-to-peer and regard a unit in a neural network as an agent ultimately. As NaaA can regard a unit as an agent without loss of generality indeed, this paper uses the setting. We illustrate the concept proposed method in Fig. 1 (TBD).

In the experiment, we use an environment which extends ViZDoom (Kempka et al., 2016), a POMDP environemnt, to MARL. We put two agents in the environment. The one is a cameraman who send information, and the another one is a main player to defeat enemies with a gun. We confirm that the cameraman learns cooperative action to send information in dead angle, behind of main player, and outperform CommNet in score.

The remaining part of this paper is organized as follows. First, we describe the two key ideas: inter-agent reward distribution and auction theory. After introducing related works, we show the experimental result in ViZDoom. Next, we show Adapritve DropConnect as a further application. Then we perform discussion and conclude this paper.

2 Inter-agent Reward Distribution

2.1 Problem Definition

Suppose there are N agents interacting to an environment. Some agents get reward from the environment, and distribute it to other agents as incentive to give precise information. The reward is limited so that if an agent distribute ρ reward, the agents' reward subtracted by ρ such as currency. For the reason, other agents for an agent itself can be interpreted as another environment which give a reward $-\rho$ instead of an observation x. We name the another environment as *internal environment* and name the original environment as an *external environment*.

Our goal is to maximize the discounted cumulative reward which the system will obtain from the external environment.

$$G = \sum_{i=1}^{n} \left[\sum_{t=0}^{T} \gamma^k R_{it}^{\mathrm{ex}} \right], \tag{1}$$

where R_{it}^{ex} is an external reward which i-th agent obtains at t, and $\gamma \in [0, 1]$ is the discount rate and T is the terminal time.

Similarly to CommNet (Sukhbaatar et al., 2016), we assume that the communication protocol between the agent is a continuous quantity such as a vector, and the content can be trained by backpropagation. Hence, we can interpret the multi-agent communication as a huge neural network. Therefore, we can interpret a unit as an agent without loss of generality. This framework can single-agent reinforcement learning as well as multi-agent reinforcement learning.

2.2 Neuron as an Agent

A typical artificial neural network is a directed graph $\mathfrak{G} = (\mathcal{V}, \mathcal{E})$ among the units. $\mathcal{V} = \{v_1, \ldots, v_N\}$ is a set of the units. $\mathcal{E} \subset \mathcal{V}^2$ is a set of edges representing connections between two units. If $(v_i, v_j) \in \mathcal{E}$, then connection $v_i \to v_j$ holds, indicating that v_j observes activation of

v_i. We denote activation of the unit v_i at time t as $x_{it} \in \mathbb{R}$. Additionally, we designate a set of units which unit i connects to as $N_i^{\text{out}} = \{j|(v_i, v_j) \in \mathcal{E}\}$ and a set of units which unit i is connected from as $N_i^{\text{in}} = \{j|(v_j, v_i) \in \mathcal{E}\}$. We denote $N_i = N_i^{\text{in}} \cup N_i^{\text{out}}$.

NaaA interprets v_i as an agent. Therefore, $\mathfrak{G}$ is a multi-agent system. An environment for v_i comprises an environment that the multi-agent system itself touches and a set of the unit to which v_i directly connects: $\{v_i \in \mathcal{V}|i \in N_i\}$. We distinguish both environments by naming the former as an external environment, and by naming the latter as an internal environment. v_i will receive rewards from both environments. We add the following assumption for characteristics of the v_i.

- N1: (Selfishness) The utility which an v_i wants to maximize is its own return (cumulative discounted reward): $G_{it} = \sum_{k=0}^{T} \gamma^k R_{i,t+k}$.
- N2: (Conservation) The summation of internal reward over $\mathcal{V}$ equals to 0. Hence, the summation of a reward by which $\mathcal{V}$ will receive both an internal and external environment R_{it} are equivalent to reward R_t^{ex}, which the entire multi-agent system receives from the external environment.
- N3: (Trade) The v_i receives internal reward ρ_{jit} from $v_j \in \mathcal{V}$ in exchange of activation signal x_i before transferring the signal to the unit. At the same time, ρ_{jit} is subtracted from the reward of v_j.
- N4: (NOOP) v_i has NOOP (no operation), for which the return is $\delta > 0$ as an action. With NOOP, the unit inputs nothing and outputs nothing.

In terms of neuroscience, N1 states that the unit acts as a cell. N2 and N3 state the distribution of NTF. N4 corresponds to apoptosis. NOOP is selected when the expected returns of the other actions are non-positive. In the following, we construct the framework of NaaA from the assumptions.

The social welfare function (total utility of the agents) G^{all} is equivalent to the objective function G. That is,

$$G^{\text{all}} = \sum_{i=1}^{n} G_{it} = \sum_{i=1}^{n} \left[\sum_{k=0}^{T} \gamma^k R_{it} \right] = \sum_{k=0}^{T} \left[\gamma^k \sum_{i=1}^{n} R_{it} \right]. \tag{2}$$

From N2, $\sum_{i=1}^{n} R_{it} = \sum_{i=1}^{n} R_{it}^{\text{ex}}$ holds. Hence, $G^{\text{all}} = G$ holds.

2.3 Cumulative Discounted Profit Maximization Framework

We denote the external reward by which unit v_i receives at time step t as R_{it}^{ex}, where $\sum_{i=1}^{n} R_{it}^{\text{ex}} = R_t^{\text{ex}}$ holds. From N3, reward R_{it}, which v_i receives at t can be written as the following.

$$R_{it} = R_{it}^{\text{ex}} + \sum_{j \in N_i^{\text{out}}} \rho_{jit} - \sum_{j \in N_i^{\text{in}}} \rho_{ijt}. \tag{3}$$

The equation is divided into positive terms and a negative term, we name the former as revenue, and the latter as cost, and denote them respectively as $r_{it} = R_{it}^{\text{ex}} + \sum_{j \in N_i^{\text{out}}} \rho_{jit}$, $c_{it} = \sum_{j \in N_i^{\text{in}}} \rho_{ijt}$. We name R_{it} as profit.

v_i maximizes the cumulative discounted profit G_{it} represented as

$$G_{it} = \sum_{k=0}^{T} \gamma^k R_{i,t+k} = \sum_{k=0}^{T} \gamma^k (r_{i,t+k} - c_{i,t+k}) = r_t - c_t + \gamma G_{i,t+1}. \tag{4}$$

G_{it} is unobserved unless the time is reached at the end of the episodes. Because prediction based on the current value is needed to select the optimal actions, we approximate G_{it} with value function $V_i^{\pi_i}(s_{it}) = \mathbb{E}_{\pi_i}[G_{it} \mid s_{it}]$ where $s_{it} \in \mathcal{S}_{\text{O}}$. In this case, the following equation holds.

$$V_i^{\pi_i}(s_{it}) = r_{it} - c_{it} + \gamma V_i^{\pi_i}(s_{i,t+1}), \tag{5}$$

Therefore, we need only consider maximization of revenue, the value function, and cost minimization. $R_{it} > 0$, i.e., $r_{it} > c_{it}$ indicates that the unit gives the additional value to the obtained data. The unit acts NOOP because $V_i^{\pi_i}(s_{it}) \leq 0 < \delta$ if $R_{it} \leq 0$ for all t because of N4.

3 AUCTION THEORY

We introduce mechanism design because, unlike several existing studies (Sukhbaatar et al., 2016), NaaA assumes that all agents are not cooperative but selfish. If we naively optimize the optimization problem of NaaA, then we obtain the trivial solution that the internal rewards will converge to 0, and that all the units except of the output units become NOOP. This phenomena occurs regardless the network topology $\mathfrak{G}$ as any nodes have no incentive to send payment ρ_{ijt} to other units. Therefore, the multi-agent system should select the action with no information. It is equivalent to taking an action randomly. For that reason, the external reward R_t^{ex} shrinks markedly.

3.1 ENVY-FREE AUCTION

To maximize the overall reward, our objective function, we borrow the idea from the digital goods auction. The auction theory belongs to mechanism design. It is intended to unveil the true price of goods. Digital goods auction is one mechanism from auction theory. It is target to copyable goods without cost, such as digital books and music.

Although several variations of digital goods auctions exist, we use an envy-free auction (Guruswami et al., 2005) because it requires a simple assumption: the same goods have one price simultaneously. In NaaA, it can be represented as the following assumption:

N5: (Law of one price) If $\rho_{j_1,i,t}, \rho_{j_2,i,t} > 0$, then $\rho_{j_1,i,t} = \rho_{j_2,i,t}$.

The assumption above indicates that ρ_{jit} takes either 0 or a positive value depending on i at a same timing t. Therefore, we name the positive side v_i's *price*, and denote as q_{it}.

We present the envy-free auction process at the left of Figure 1. It shows the negotiation process between one unit in sending activation and a group of units that buy the activation. The negotiation performed per time step in RL. We name the unit in sending activation as a seller, and units in buying activation as buyers. First, the buyer bids the unit in bidding price b_{jit} (**1**). Next, the seller decides the optimal price $\hat{q}_{it}$, and performs allocation (**2**). Payment occurs if b_{ijt} exceeds q_{jt}. In this case, $\rho_{jit} = H(b_{jit} - q_{it})q_{it}$ holds where $H(\cdot)$ is a step function. Besides, we define $g_{jit} = H(b_{jit} - q_{it})$ and name it *allocation*. After allocation, the buyers perform payment as $\rho_{jit} = g_{jit}\hat{q}_{it}$ (**3**). Eventually, seller earns The seller only sends activation x_i to the allocated buyers (**4**). A buyer which cannot receive the activation approximates x_i with $\mathbb{E}_\pi[x_i]$.

In the following, we discuss revenue, cost, and value functions based on Eq:(5).

Revenue: The revenue of a unit is given as

$$r_{it} = \sum_{j \in N_i^{\text{out}}} g_{jit} q_{it} + R_i^{\text{ex}} = q_i d_{it} + R_i^{\text{ex}}, \tag{6}$$

where $d_{it} = \sum_{j \in N_i^{\text{out}}} g_{jit}$ is a count of units for which the bidding price for q_{it} is greater than or equal to q_{it}, designated as demand. q_{it} maximizing the equation is designated as the optimal price. It is denoted as $\hat{q}_{it}$. Because R_i^{ex} is independent of q_t, the optimal price $\hat{q}_{it}$ is given as

$$\hat{q}_{it} = \underset{q \in [0,\infty)}{\operatorname{argmax}}\, q d_{it}(q). \tag{7}$$

We present the curve of r_{it} on the right side of Figure 1.

Cost: The cost is an internal reward that the unit should pay to other units. It is represented as shown below.

$$c_{it} = \sum_{j \in N^{\text{in}}} g_{ijt} q_{jt} = \mathbf{g}_{it}^{\mathrm{T}} \mathbf{q}_t, \tag{8}$$

where $\mathbf{g}_{it} = (g_{i1t}, \ldots, g_{iNt})^{\mathrm{T}}$ and $\mathbf{q}_t = (q_{1t}, \ldots, q_{Nt})^{\mathrm{T}}$. Although c_{it} itself is minimized when $b_{ijt} = 0$, this represents a tradeoff with the following value function.

Value Function: The activation x_{it} depends on input from the units in N_i^{in} affecting the bidding price from units in N_i^{out}. If we minimize b_{ijt} and let $b_{ijt} = 0$, then the purchase of activation fails,

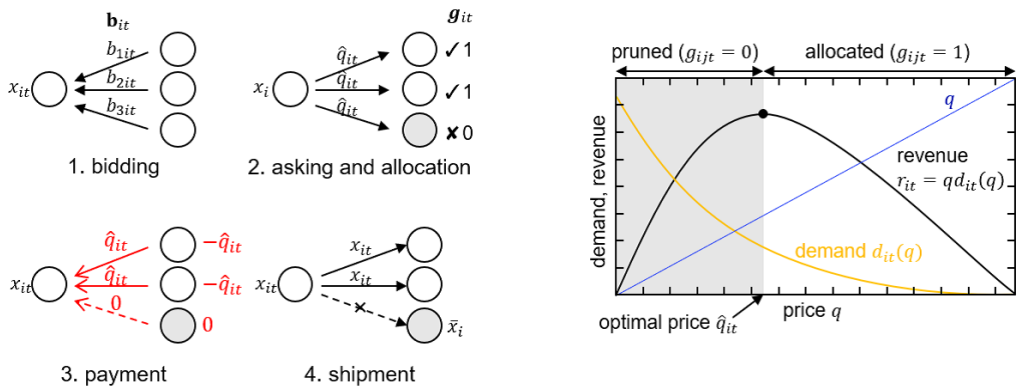

Figure 1: **Left**: The process of trade in an envy-free auction. **Right**: A price determination curve for a unit. Revenue of a unit is a product of monotonically decreasing demand and price. The price maximizing the revenue is the optimal price.

and the reward the unit can obtain from the units to which the unit connects becomes lower in the future.

We consider effects for value functions in the cases when a unit succeeds in purchasing v_j or not. We approximate the value function as a linear function of $\mathbf{g}_{it}$:

$$V_i^{\pi_i}(s_{i,t+1}) \approx \mathbf{o}_{it}^{\mathrm{T}} \mathbf{g}_{it} + V_{i,t+1}^0, \tag{9}$$

where $\mathbf{o}_{it}$ is a parameter implemented as difference between two returns of v_i whether we observe x_i or not. As $\mathbf{o}_{it}$ is equivalent to the cumulative discount value of counterfactual reward (Agogino & Tumer, 2006), we name it *counterfactual return*. V_{it}^0 is a constant independent of $\mathbf{g}_{it}$ and we name it *blind value function* as it is equivalent to value function when v_i takes action without any observation $x_1, \ldots, x_N$.

Therefore, the optimization problem is presented below.

$$\max_{\mathbf{a}} Q_i(s_{it}, \mathbf{a}) = \max_{q} q d_{it}(q) - \min_{\mathbf{b}} \mathbb{E}_{\hat{\mathbf{q}}_t} \left[\mathbf{g}_{it}(\mathbf{b})^{\mathrm{T}} (\hat{\mathbf{q}}_t - \gamma \mathbf{o}_{it}) \right] + \text{const.}, \tag{10}$$

where $\mathbf{a} = (\mathbf{b}, q)$. Note that $\mathbf{g}_{it} = H(\mathbf{b} - \mathbf{q}_t)$. We take the expectation $\mathbb{E}_{\hat{\mathbf{q}}_t}[\cdot]$ because the asked price $\hat{\mathbf{q}}_t$ is unknown for v_i, except for $\hat{q}_{it}$, and $g_{iit} = 0$.

Then, what is bidding price b_{it} to maximize return? The following theorem holds.

Theorem 3.1. *(Truthfulness) the optimal bidding price for maximizing return is* $\hat{\mathbf{b}}_{it} = \gamma \mathbf{o}_{it}$.

See the Appendix for the proof.

That is, the unit should only consider its counterfactual return (!). If $\gamma = 0$, the case is equivalent to a case without auction. Hence, the bidding value raises if each unit consider long-time reward. Consequently, in the mechanism of NaaA, the unit obeys as if performing valuation to the other units, and declares the value truthfully.

Then, the following corollary holds:

Corollary 3.1. *The Nash equilibrium of an envy-free auction* $(\mathbf{b}_{it}, q_{it})$ *is* $(\mathbf{o}_{it}, \operatorname{argmax}_q q d_{it}(q))$.

The remaining problem is how to predict $\mathbf{o}_t$. Although several method can be applied to this problem, we use Q-learning to predict $\mathbf{o}_t$. As $\mathbf{o}_{it}$ is difference of two Qs, we approximate each of Q. Other RL such as SARSA and A3C can be employed. We parametrize the state with a vector $\mathbf{s}_t$ which contains input and weight. ϵ-greedy policy with Q-learning typically suppose that discrete actions So, as an action, we employ allocation g_{ijt} instead of $\mathbf{b}_{it}$ and q_{it}. The overall algorithm is shown in Algorithm 1.

Algorithm 1 Envy-free auction for NaaA

for $t = 1$ **to** T **do**
  Compute a bidding price for every edge: **for** $(v_j, v_i) \in \mathcal{E}$ **do** $b_{ijt} \leftarrow Q^{\pi_i}(\mathbf{s}_{it}, \mathbf{e}_j) - Q^{\pi_i}(\mathbf{s}_{it}, \mathbf{0})$
  Compute an asking price for every node: **for** $v_i \in \mathcal{V}$ **do** $\hat{q}_{it} \leftarrow \underset{q \in [0,\infty)}{\operatorname{argmax}}\, q d_{it}(q)$.
  for $(v_i, v_j) \in \mathcal{E}$ **do**
    Compute allocation: $g_{jit} \leftarrow H(b_{jit} - \hat{q}_{it})$
    Compute the price the agent should pay: $\rho_{jit} \leftarrow g_{jit}\hat{q}_{it}$
  end for
  Make a payment: **for** $v_i \in \mathcal{V}$ **do** $R_{it} \leftarrow \sum_{j \in N_i^{\mathrm{out}}} \rho_{jit} - \sum_{j \in N_i^{\mathrm{in}}} \rho_{ijt}$,
  Make a shipment: **for** $v_i \in \mathcal{V}$ **do** $\tilde{x}_{ijt} = g_{ijt}x_{ijt} + (1 - g_{ijt})\bar{x}_{ijt}$
  for $v_i \in \mathcal{V}$ **do**
    Observe external state $\mathbf{s}_{it}^{ex}$
    $\mathbf{s}_{it} \leftarrow (\mathbf{s}_{it}^{ex}, \tilde{\mathbf{x}}_{it}, \boldsymbol{\theta}_i)$, where $\tilde{\mathbf{x}}_{it} = (\tilde{x}_{i1t}, \ldots, \tilde{x}_{int})^{\mathrm{T}}$ and $\boldsymbol{\theta}_i$ is v_i's parameter.
    Sample action $a_{it}^{ex} \sim \pi_i^{ex}(\mathbf{s}_{it})$
    Receive external reward $R_{it} \leftarrow R_{it} + R_{it}^{\mathrm{ex}}(a_{it}^{\mathrm{ex}})$
    Update Q^{π_i} under the manner of Q-learning by calculating the time difference (TD)-error
  end for
end for

4 Related Work

NaaA belongs to a class of partially observable stochastic game (POSG) (Hansen et al., 2004) because it processes multiple units as agents. POSG, a class of reinforcement learning with multiple agents in a POMDP environment, presents several research issues, one of which is communication. CommNet (Sukhbaatar et al., 2016), which exploits the characteristics of a unit that is agnostic to the topology of other units, employs backpropagation to train multi-agent communication. Another one is credit assignment. Instead of reward $R(a_t)$ of an agent i for actions at t a_t, QUICR-learning (Agogino & Tumer, 2006) maximizes counterfactual reward $R(a_t) - R(a_t - a_{it})$, the difference in the case of the agent i takes an action a_{it} (a_t) and not ($a_t - a_{it}$). COMA (Foerster et al., 2017) also maximizes counterfactual rewards in an actor–critic setting. In the setting, all actors have common critics, which improves both actors and critics with time difference (TD)-error of a counterfactual reward. This paper unifies both issues: communication and credit assignment. The main proposal is a framework to manage the agents to maximize the *counterfactual return*, the extended counterfactual reward along the time axis.

Training a neural network with a multi-agent game is an emerging methodology. Nash equilibrium is not guaranteed to maximizing overall reward, and the difference is designated as a social dilemma. Because the existence of a dilemma depends on the reward design, methods to resolve dilemmas with good reward design are being investigated: mechanism design (Myerson, 1983) is also known as inverse game theory. Mechanism design is applied to auctions (Vickrey, 1961) and matching (Gale & Shapley, 1962). NaaA is outcomes from mechanism design. NaaA applies a digital goods auction (Guruswami et al., 2005) to reinforcement learning with a multi-agent neural network, to obtain a maximized return by units as a Nash equilibrium.

Adaptive DropConnect (ADC), which we propose in a later part of this paper, extends DropConnect (Wan et al., 2013), a regularization technique. The idea of ADC (instead of dropping each connection between units in constant probability, using skew probability correlated to the absolute value of weights) is eventually closer to Adaptive DropOut (Ba & Frey, 2013), although the derivation differs. The adjective "adaptive" is added with respect to the method. Optimizing the neural network with RL was investigated by Andrychowicz et al. (2016). In contrast to their methods, which use recurrent neural network (RNN) and which therefore have difficult implementation, our method is RNN-free and forms as a layer. For those reasons, its implementation is simple and fast. Moreover, it has a wide area of applicability.

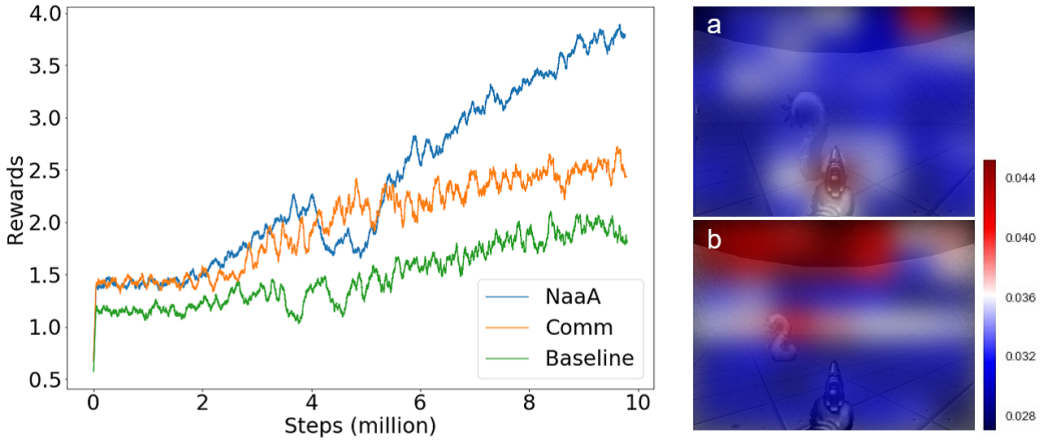

Figure 2: **Left:** Learning curve for the multi-agent task of VizDoom. Our method based on NaaA outperforms the other two methods: baseline and Comm DQN. **Right:** Reward visualization shows us what the cameraman sees: (a) The cameraman sees the pistol. (b) The cameraman sees the point which enemy appear and come closer.

5 EXPERIMENT

We confirmed that additional agents complement the main player using ViZDoom, an environment for Doom. A player in Doom environment should seek the enemy in the map, and then defeat the enemy. Because ViZDoom provides several maps, we used ViZDoom.

5.1 SETUP

We used a scenario based on Defend the Center (DtC), provided by ViZDoom platform. In DtC, players are placed in the center of a field of circle. They attack enemies that come from the wall. The game has two players: a main player and a cameraman. Alhough the main player can attack the enemy with bullets, the cameraman has no way to attack, and only scouts for the enemy. The action space for the main player is the combination of { attack, turn left, turn right }. Therefore, the total number of actions is $2^3 = 8$. The cameraman has two possible actions: { turn left, turn right }. Although the players can only change direction, they cannot move on the field. The enemy will die if have the attack (bullet) from the main player once, then player receives +1. As a default on an episode, the ammunition amount is 26. The main player will die if under attack from the enemy to the extent that health becomes 0, then the player receives -1. The cameraman will not die if attacked by the enemy. The episode will terminate when the maim player dies, or after 525 steps have elapsed.

5.2 MODEL

We compared three models: the proposed method and two comparison targets.

Baseline DQN without communication. The main player learns standard DQN with the perspective that the player is viewing. Because the cameraman does not learn, the player continues to move randomly.

Comm DQN with communication. The main player learns DQN with two perspectives: the player's own and the cameraman's. The communication vector is learned with a feed-forward neural network. The method is inspired by Commnet.

NaaA The proposed method. The main player learns DQN with two perspectives: the player's own and the cameraman's. The transmission of reward and communication are performed using the proposed method.

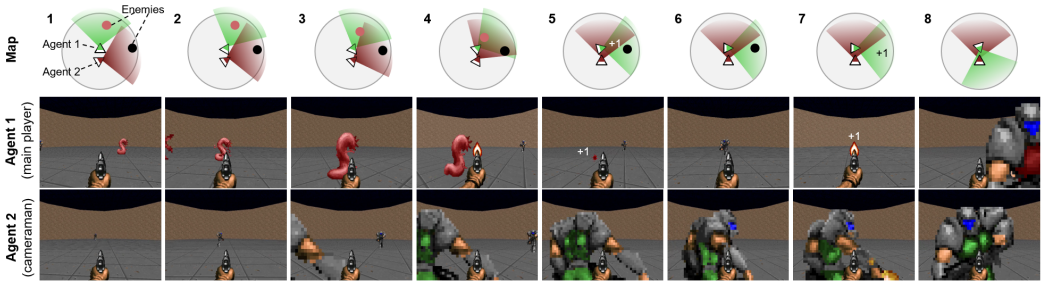

Figure 3: NaaA leads the agents to obtain cooperative relationship. First, the two agents are facing in different directions, and the cameraman sells its information to the main player (**1**). The main player who bought the information starts to turn right to find the enemy. The cameraman who sold the information starts to turn left to seek new information by finding the blind area of the main player (**2** and **3**). With turning, the main player attacks the first enemy which he already saw (**4** and **5**). After the main player finds out the enemy, he attacks the enemy, and obtain the reward (**6** and **7**). Until the next enemy appears, the agents watch their dead area each other (**8**).

5.3 Results

Training is performed in 10 million steps. Figure 2 Left presents that our model NaaA outperforms two methods. Improvement is achieved by Adaptive DropConnect. We confirmed that the cameraman sees the enemy through an episode. This can be interpreted as the cameraman reporting the enemy position. In addition to seeing the enemy, the cameraman sees the area behind of main player several times. This action enables the cameraman to observe attacks from the enemy while seizing a better relative position.

For further interpretation of the result, we present visualization of the revenue that the agent earned in Figure 2 Right as a heatmap. The background picture is a screen in Doom taken at the moment when the filter in CNN is mostly activated. Figure 3 shows an example of learnt sequence of actions by our method.

6 Further Application

Actually, NaaA is useful not only for multi-agent RL, but also for training of the network. Typical training algorithms of a neural network such as those of RMSProp (Tieleman & Hinton, 2012) and Adam (Kingma & Ba, 2014) are based on a sequential algorithm such as stochastic gradient descent (SGD). Therefore, the problem can be interpreted as a problem to update the state (i.e., weight) to the goal, which is minimization of the expected likelihood.

The learning can be accelerated by application of NaaA to the optimizer. We designate the application of NaaA to SGD as *Adaptive DropConnect* (ADC), which is eventually a combination of DropConnect (Wan et al., 2013) and Adaptive DropOut (Ba & Frey, 2013). We introduce ADC herein as one application of NaaA.

ADC uses NaaA for supervised optimization problem with several revisions. First, an environment has an input state such as an image. The agent is expected to update its parameters to maximize its reward obtained from the criterion calculator. The criterion calculator gives batch-likelihood as the reward to the agent. The agent is a classifier which updates its weights to maximize the reward from the criterion calculator. The weights are recorded as an internal state. As a counterfactual return o_{ijt}, we used a heuristic that uses the absolute value of weight $|w_{ijt}|$, which is the same technique as that used by Adaptive DropOut. We use the absolute value of weights because it is the update amount for which the magnitude of error of the output of units is proportional to $|w_{ijt}|$.

The algorithm is presented as Algorithm 2. Because the algorithm is quite simple, its implementation can be performed easily. For that reason, it can be widely applied for most general deep learning problems such as image recognition, sound recognition, and even for deep reinforcement learning.

Algorithm 2 Adaptive DropConnect

for $t = 1$ **to** T **do**
  Compute a bidding price for every edge: **for** $(v_j, v_i) \in \mathcal{E}$ **do** $b_{ijt} \leftarrow |w_{ijt}|$
  Compute an asking price for every node: **for** $v_i \in \mathcal{V}$ **do** $\hat{q}_{it} \leftarrow \underset{q \in [0,\infty)}{\operatorname{argmax}}\, q d_{it}(q)$.
  for $(v_i, v_j) \in \mathcal{E}$ **do**
    Compute allocation: $g_{jit} \leftarrow H(b_{jit} - \hat{q}_{it})$
  end for
  Sample a switching matrix U_t from a Bernoulli distribution: $U_t \sim \mathrm{Bernoulli}(\varepsilon)$
  Sample the random mask M_t from a Bernoulli distribution: $M_t \sim \mathrm{Bernoulli}(1/2)$
  Generate the adaptive mask: $M'_t \leftarrow U_t \circ M_t + (1 - U_t) \circ G_{ijt}$
  Compute $\mathbf{h}_t$ for making a shipment: $\mathbf{h}_t \leftarrow (M'_t \circ W_t)\mathbf{x}_t + \mathbf{b}_t$
  Update W_t and $\mathbf{b}_t$ by backpropagation.
end for

6.1 EXPERIMENT

6.1.1 SETUP

In this experiment, we confirm two tasks of classification and single-agent reinforcement learning.

For classification task, we used three types of datasets, MNIST, CIFAR-10 and STL-10. The task given here is to predict the label for each image. The number of class is 10 in those three datasets. The first dataset, MNIST, is a collection of black and white images of handwritten digits whose size is 28x28. The training set and test set are composed of 60,000 examples and 10,000 examples respectively. The images in CIFAR-10 dataset are colored and the size of each image is 32x32. The task is to predict what is shown in each picture. This dataset contains 6,000 images per class (5,000 for training and 1,000 for test). STL-10 is a dataset for image recognition, the number of which is 1,300 for each class (500 for training and 800 for test). The size of each image is 96x96. In this experiment, however, images were resized into 48x48, since the resolution is large compared to the datasets shown above and this dataset requires far more time and resource to compute.

Next, we set the single-agent reinforcement learning task. We used the CartPole task from OpenAI gym with visual input. In this setting, the agent must balance a pole while moving a cart. There is much non-useful information related to the image. For that reason, pruning the pixels is important.

6.1.2 MODEL

In this experiment, we compared two models, DropConnect and Adaptive DropConnect (proposed model in this paper). The baseline model is composed of two convolutional layers and two fully connected layers whose outputs are dropped out (we set the possibility as 0.5). The labels of input data are predicted using log-softmaxed value of last fully connected layer. In DropConnect model and Adaptive DropConnect model, first fully connected layer is replaced by DropConnected layer and Adaptive DropConnected layer respectively. Note that DropConnect model corresponds to the our method with ε = 1.0 and this means agents do not perform their auctions, and randomly mask the weights.

6.1.3 RESULTS

For the MNIST datasets, the models are trained for 10 epochs and then evaluated with the test data. The numbers of epochs for CIFAR-10 and STL-10 are 20 and 40 respectively. Experiments are repeated 20 times for each condition, and the average and standard deviation of error rate was calculated. The results is shown in Table 1. As expected, with the model using Adaptive DropConnect, the classification error rate was lower than both the baseline and DropConnect regardless of the datasets given in this experiment.

Table 1: Experimental result for image classification tasks and single-agent RL

	MNIST	CIFAR-10	STL-10	CartPole
. DropConnect (Wan et al., 2013)	1.72 ± 0.160	43.14 ± 1.335	50.92 ± 1.322	285
Adaptive DropConnect	**1.36** ± 0.132	**39.84** ± 1.035	**42.17** ± 2.329	**347**

7 DISCUSSION

Regarding the optimization method, although envy-free auction guarantees truthfulness if the buyer prices are sealed, in cases where buyers can mutually communicate and share price information, the buyer can fake the price with lower demand in a process of collusion. To address the issue, several solutions such as random sample auction Goldberg et al. (2006) are proposed.

NaaA is applicable to learning distributed environments on a computer network such as a peer-to-peer network, and controlling the sub-modules of robots such as multiple cameras. Specifically, it is applicable to various methods as described below.

- Hyperparameter tuning. Several algorithms have been proposed such as neuroevolution using genetic algorithms. In the case, profit or counterfactual return is useful for a fitness function.
- Pruning. Computing costs can be reduced by downsizing a neural network.
- Attention control. Research of attention is using reinforcement learning to control attention.
- Ensemble. Our method is applicable to mixed multiple models.

These applications illustrate the direction of our research.

8 CONCLUSION AND FUTURE WORKS

This paper proposed NaaA, a reinforcement learning framework that treats each unit on a neural network as an agent. First, we pointed out there are dilemma problems if we naively optimize NaaA. We proposed an optimization method with auction. Consequently, an action by which units evaluate the counterfactual return of other units is obtained as a Nash equilibrium. Furthermore, we proposed Q-learning based algorithm, adaptive dropconnect, to optimize the neural network topology dynamically with evaluation of counterfactual return. For the evaluation, we performed experiments based on single-agent and multi-agent platforms, demonstrating that our experimentally obtained results improve existing methods.

As a direction of future research, we use on-policy methods to perform adaptive dropconnect, and consider applications combining genetic algorithms.

REFERENCES

A. K. Agogino and K. Tumer. QUICR-learning for multi-agent coordination. AAAI'06, 2006.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Jimmy Ba and Brendan Frey. Adaptive dropout for training deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 3084–3092, 2013.

J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. *arXiv:1705.08926*, 2017.

Jakob Foerster, Yannis Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2137–2145, 2016.

David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

Andrew V Goldberg, Jason D Hartline, Anna R Karlin, Michael Saks, and Andrew Wright. Competitive auctions. *Games and Economic Behavior*, 55(2):242–269, 2006.

Venkatesan Guruswami, Jason D Hartline, Anna R Karlin, David Kempe, Claire Kenyon, and Frank McSherry. On profit-maximizing envy-free pricing. In *ACM-SIAM symposium on Discrete algorithms*, 2005.

Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pp. 709–715, 2004.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pp. 1–8. IEEE, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

William Forster Lloyd. *Two lectures on the checks to population*. 1833.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Roger B Myerson. Mechanism design by an informed principal. *Econometrica: Journal of the Econometric Society*, pp. 1767–1797, 1983.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

S. Sukhbaatar, R. Fergus, et al. Learning multiagent communication with backpropagation. In *NIPS'16*, 2016.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3): 58–68, 1995.

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

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pp. 1058–1066, 2013.

APPENDIX

A.1 PROOF OF THEOREM 3.1

As for a buyer, the asking price q for a seller is unknown, we address q which has support $[0, \infty)$, and consideration to maximize $\mathbb{E}_q[G(b, q)]$, In this case, the following equation holds.

$$\begin{aligned}
\frac{\partial}{\partial b}\mathbb{E}_q[G(b,q)] &= \frac{\partial}{\partial b}\int_0^\infty (H(b-q)\cdot(v-q)+G_0)p(q)dq \\
&= \frac{\partial}{\partial b}\left[\int_0^b (v-q)p(q)dq + G_0\int_0^\infty p(q)dq\right] \\
&= \frac{\partial}{\partial b}\int_0^b (v-q)p(q)dq \\
&= (v-b)p(q=b),
\end{aligned}$$

Therefore, the condition to maximize $\mathbb{E}_q[G(b, q)]$ is $b = v$.