A reinforcement learning scheme for a multi-agent card game

Hajime Fujita[†], Yoichiro Matsuno[‡], and Shin Ishii[†]*
[†]Nara Institute of Science and Technology
8916–5 Takayama, Ikoma, 630–0192 JAPAN
{hajime-f,ishii}@is.aist-nara.ac.jp
[‡]Ricoh Co. Ltd. 1–1–17 Koishikawa, Tokyo, 112–0002 JAPAN
yohichiroh.matsuno@nts.ricoh.co.jp
*CREST, Japan Science and Technology Corporation

Abstract – *We formulate an automatic strategy acquisition problem for the multi-agent card game "Hearts" as a reinforcement learning (RL) problem. Since there are often a lot of unobservable cards in this game, RL is approximately dealt with in the framework of a partially observable Markov decision process (POMDP). This article presents a POMDP-RL method based on estimation of unobservable state variables and prediction of actions of the opponent agents. Simulation results show our model-based POMDP-RL method is applicable to a realistic multi-agent problem.*

Keywords: Reinforcement learning, POMDP, multi-agent system, card game

1 Introduction

Many card games are multi-player imperfect-information games; for each game player, there are unobservable state variables, e.g., cards in another player's hand or undealt cards. Since players must contrive a strategy, cooperating or competing with other players, card games are well-defined test-beds for the research on a reinforcement learning (RL) scheme for a multi-agent environment. For example, there have been studies that applied RL to "Black Jack" [1],"Othello" [2], and "Backgammon" [3]. However, these games are defined as complete observable problems, i.e., perfect-information games, in which every information on the game state is presented to players. In order to deal with more general problems such as partially observable problems [5], it is important to estimate missing information [4] and to acquire better strategies that incorporate the prediction of environmental behaviors.

This article deals with the card game "Hearts", which is an n-player($n > 2$) non-cooperative finite-state zero-sum imperfect-information game, and presents an RL scheme based on estimation of unobservable state variables and prediction of actions of the opponent agents. Since there are often a lot of unobservable cards in this game, we formulate this realistic problem as a partially observable Markov decision process (POMDP). Our POMDP-RL method copes with the partial observability by estimating the card distribution in the other agents' hands and by predicting the actions of the other agents. Since the state space of the game Hearts is huge, we use an approximation, in which an opponent agent approximately selects his action based on a mean expected observation state. This approximation successfully avoids manipulating directly the huge state space. In addition, in order to accelerate the learning and the exploration of possible opponent agents' actions, we use function approximators with a reduced state representation scheme.

We carried out experiments using rule-based agents; a rule-based agent has more than 50 rules and is an "experienced" level player of the game Hearts. The learning agent trained by our POMDP-RL method exhibits high performance against rule-based agents; it comes to beat the rule-based agents after 80,000 training games. Owing to the fast learning using efficient function approximators and the POMDP approximation, the learning agent has successfully experienced a large number of training games. As a consequence, our model-based POMDP-RL method is applicable to a realistic multi-agent problem.

2 Preparation

2.1 The card game "Hearts"

The game of Hearts is played by four players and uses the ordinary 52-card deck. There are four suits, i.e., spades(♠), hearts(♡), diamonds(♢), and clubs(♣), and there is an order of strength within each suit (i.e., A,K,Q,...,2). There is no strength order among the suits. Cards are distributed to the four players so that each player has in his hand 13 cards at the beginning

of the game. Thereafter, according to the rules below, each player plays a card clock-wisely in order. When each of the four players has played a card, it is called a trick. Namely, each player plays a card once in one trick. The first card played in a trick is called the leading card and the player who plays the leading card is called the leading player. A single game ends when 13 tricks are carried out.

- Except for the first trick, the winner of the current trick becomes the leading player of the subsequent trick.
- In the first trick, ♣2 is the leading card, denoting that the player holding this card is the leading player.
- Each player must play a card of the same suit as the leading card.
- If a player does not have a card of the same suit as the leading card, he can play any card. When a heart is in such a case played for the first time in a single game, the play is called "breaking hearts".
- Until the breaking hearts occurs, the leading player may not play a heart. If the leading player has only hearts, it is an exceptional case and the player may lead with a heart.
- After a trick, the player that has played the strongest card of the same suit as the leading card becomes the winner of that trick.
- Each heart equals a one-point penalty and the ♠Q equals a 13-points penalty. The winner of a trick receives all of the penalty points of the cards played in the trick.

According to the rules above, a single game is played, and at the end of a single game, the score of each player is determined as the sum of the received points. The lower the score, the better.

2.2 State transition of Hearts

First, we assume there are in the environment a single learning agent and three opponent agents that do not learn.

A single state transition of the game is represented by: (1) real state s that includes every card (observable and unobservable) allocation, (2) observation x for the learning agent, i.e., the cards in the agent's hand and the cards that have already been played in the past tricks and the current trick, (3) the agent's action a, i.e., a single play at his turn, and (4) strategy ϕ of each of the opponent agents. Let t indicate a playing turn of the learning agent. $t = 14$ indicates the end state of the game. At the t-th playing turn, the learning agent does not know the real state s_t, and all he can do is to estimate it by considering the history of observations and actions in the past tricks. In the following descriptions, we assume that there are three opponent agents intervening between the t-th play and the $(t+1)$-th play of the learning agent[1]. Between the t-th play and the $(t+1)$-th play of the learning agent, then, there are three state transitions due to actions by the three opponent agents. These state transitions are indexed by t. Each of the opponent agents is also in a partial observation situation; state, observation, action and strategy at his t-th playing turn are denoted by s_t^i, x_t^i, a_t^i and ϕ_t^i, respectively, where i is the index of an opponent agent.

Let M denote the learning agent, and $M^i (i = 1, 2, 3)$ denote the i-th opponent agent. We assume

- assumption (a)
  Agent M^i probabilistically determines his action a_t^i for his own observation x_t^i at his t-th play.

Under this assumption, the state transition between the t-th play and the $(t+1)$-th play of the learning agent is given by

$$P(s_{t+1}|s_t, a_t, \Phi_t) = \sum_{s_t^1, s_t^2, s_t^3} \sum_{a_t^1, a_t^2, a_t^3} \prod_{j=0}^{3} P(s_t^{j+1}|s_t^j, a_t^j) \times \prod_{i=1}^{3} \sum_{x_t^i} P(a_t^i|x_t^i, \phi_t^i) P(x_t^i|s_t^i), \quad (1)$$

where $s_t^0 = s_t, a_t^0 = a_t, s_t^4 = s_{t+1}, a_t^4 = a_{t+1}, x_t^0 = x_t$ and $x_t^4 = x_{t+1}$. $\Phi \equiv \{\phi_t^i : i = 1, 2, 3\}$, where ϕ_t^i denotes the strategy of opponent agent M^i at his t-th play.

2.3 POMDP Approximation

The game Hearts is approximated as a partially observable Markov decision process (POMDP) [5]; Under this approximation, it is assumed that there is only one learning agent, and the strategies of the other agents are fixed, that is, the other agents constitute the static environment. Due to this POMDP approximation, $\phi^i (i = 1, 2, 3)$ does not depend on the play index t.

Since the game process of Hearts is deterministic, there are two facts:

- New state s_t^{i+1}, which is reached from a previous state s_t^i by an action a_t^i, is uniquely determined. Namely, $P(s_t^{i+1}|s_t^i, a_t^i)$ is 1 for a certain state and 0 for the other states.
- Observation x_t^i is uniquely determined at state s_t^i. Namely, $P(x_t^i|s_t^i)$ is 1 for a certain observation state and 0 for the other observation states.

[1] If the leading player of the t-th trick and that of the $(t+1)$-th trick are different, the number of intervening opponent agents is not three. However, the following explanation can be easily extended to such a case.

Since state s_t is not observable for the learning agent, x_t should be estimated using the history $H_t \equiv \{(x_t, -, -), (x_{t-1}, a_{t-1}, a_{t-1}^{1,2,3}), \ldots, (x_1, a_1, a_1^{1,2,3})\}$, actions $a_t^i (i = 1, 2, 3)$ at the t-th play, and game knowledge (game rules, etc.) K.

The transition probability for the observation state is given by

$$P(x_{t+1}|a_t, \Phi, H_t, K) = \sum_{s_t \in \mathcal{S}_t} \sum_{s_{t+1} \in \mathcal{S}_{t+1}} P(x_{t+1}|s_{t+1}) \times P(s_{t+1}|s_t, a_t, \Phi) P(s_t|H_t, K), \quad (2)$$

where $\mathcal{S}_t$ is the set of possible states at the t-th play of the learning agent.

From the above two facts and equation (1),

$$P(x_{t+1}|a_t, \Phi, H_t, K) = \sum_{s_t \in \mathcal{S}_t} P(s_t|H_t, K) \sum_{(a_t^1, a_t^2, a_t^3) \in \mathcal{S}_{t+1}^+} \prod_{i=1}^{3} P(a_t^i|x_t^i, \phi^i, H_i, K). \quad (3)$$

Here, $\mathcal{S}_{t+1}^+$ denotes the set of possible (a_t^1, a_t^2, a_t^3) by which the previous state-action pair (s_t, a_t, Φ) reaches a new state s_{t+1} whose observation state is x_{t+1}. $P(s_t|H_t, K)$ is called a belief state. Equation (3) provides a model of the environmental dynamics.

However, the calculation in equation (3) has two difficulties. One is the intractability of the belief state; since the state space of the game of Hearts is huge, the rigorous calculation of the summation $\sum_{s_t \in \mathcal{S}_t}$ is difficult. The other is the difficulty in retrieving the game tree; especially when there are a lot of unobservable state variables, i.e., unobservable cards, $\mathcal{S}_{t+1}^+$ is a huge set and then the calculation of the summation $\sum_{(a_t^1, a_t^2, a_t^3) \in \mathcal{S}_{t+1}^+}$ is also difficult.

In order to cope with the former difficulty, we use the following approximation. For history H_t and game knowledge K, we first calculate the mean of estimated observation state for agent M^i as

$$\hat{y}_t^i(a_t, H_t, K) \equiv \sum_{y_t^i} y_t^i P(y_t^i|a_t, H_t, K), \quad (4)$$

where y_t^i is the estimation for the real observation for agent M^i, x_t^i. Using the mean estimated observation $\hat{y}_t^i$, the transition probability (3) is approximated as

$$P(x_{t+1}|a_t, \Phi, H_t, K) \approx \sum_{(a_t^1, a_t^2, a_t^3) \in \mathcal{S}_{t+1}^+} \prod_{i=1}^{3} P(a_t^i|\hat{y}_t^i(a_t, H_t, K), \hat{\phi}^i). \quad (5)$$

From assumption (a), each opponent agent determines its action a_t^i with probability $P(a_t^i|x_t^i, \phi^i, H_t, K)$. However, this action selection probability and the real observation x_t^i are unknown for the learning agent and they should be estimated in some way. Therefore, the learning agent assumes that the action selection process is approximately done by a stochastic process that is dependent on the mean estimated observation $\hat{y}_t^i(a_t, H_t, K)$. It should be noted that the approximated strategy $\hat{\phi}^i$ in equation (5) is different from the real strategy ϕ^i in equation (3).

Strategy ϕ^i represents the policy that determines actions of agent M^i. The approximated policy $\hat{\phi}^i$ is represented and learned by a function approximator. For a game finished in the past, an observation state and an action taken by an opponent agent at that state can be reproduced by replaying the game from the end to the start. In order to train the function approximator for $\hat{\phi}^i$, the input and the target output are given by $\hat{y}_t^i(a_t, H_t, K)$ and the action a_t^i actually taken by agent M^i at that turn, respectively.

2.4 Action control

According to our RL method, an action is selected based on the expected TD error, which is defined by

$$\langle \delta_t \rangle(a_t) = \langle R(x_{t+1}) \rangle(a_t) + \gamma \langle V(x_{t+1}) \rangle(a_t) - V(x_t), \quad (6)$$

where

$$\langle f(x_{t+1}) \rangle(a_t) \equiv \sum_{x_{t+1}} P(x_{t+1}|a_t, \Phi, H_t, K) f(x_{t+1}) \quad (7)$$

and $P(x_{t+1}|a_t, \Phi, H_t, K)$ is given by equation (5). The expected TD error considers the estimation of the unobservable states and the strategies of the other agents.

Using the expected TD error, the action selection probability is determined as

$$P(a_t|x_t) = \frac{\exp(\langle \delta_t \rangle(a_t)/T_m)}{\sum_{a_t \in \mathcal{A}} \exp(\langle \delta_t \rangle(a_t)/T_m)}. \quad (8)$$

T_m is a parameter controlling the action randomness.

Our RL method uses the TD error expected with respect to the estimated transition probability for the observation state. An action is then determined based on the estimated environmental model. Such an RL method is often called a model-based RL method [8].

2.5 Actor-critic algorithm

Although the actor-critic algorithm [7] is *not* used in our RL method, it is briefly introduced here for the convenience of explanation. According to the actor-critic algorithm, the critic maintains the value function $V(x_t)$ that evaluates state x_t at the t-th turn of the learning agent, and the actor determines its action a_t based on a merit function $U(x_t, a_t)$.

The critic calculates the TD error for a given state transition for observable states:

$$\delta_t = R(x_{t+1}) + \gamma V(x_{t+1}) - V(x_t), \quad (9)$$

where $R(x_{t+1})$ is the reward function that is assumed to be dependent only on the observation state x_{t+1}. In the case of Hearts, the reward function represents the (negative) penalty points that the learning agent receives at the t-th trick.

Using the TD error, the critic updates the value function and the actor updates the merit function as

$$V(x_t) \leftarrow V(x_t) + \eta_c \delta_t \tag{10}$$
$$U(x_t, a_t) \leftarrow U(x_t, a_t) + \eta_a \delta_t, \tag{11}$$

where η_c and η_a are the learning rates for the critic and the actor, respectively.
Using the merit function, the actor selects an action according to the Boltzmann policy

$$P(a_t|x_t) = \frac{\exp(U(x_t, a_t)/T_e)}{\sum_{a_t \in \mathcal{A}} \exp(U(x_t, a)/T_e)}, \tag{12}$$

where T_e is a parameter controlling the action randomness and $\mathcal{A}$ denotes the set of possible actions.

3 Proposed method

Our RL architecture roughly consists of two modules: a state evaluation module and an action control module. The action control module consists of three action predictors each corresponding to each of the three opponent agents and one action selector.

3.1 State evaluation module

The state evaluation module has the same role as the critic in the actor-critic algorithm, i.e., it maintains the value function $V(x_t)$ that evaluates the current observation state x_t of the learning agent. In order to accelerate the learning of the state evaluation module, we use a feature extraction technique for its input and output. An input to the function approximator, p_t, is given mainly by the transformation from an observation state x_t as follows.

- $p_t(1)$: the number of club cards that have been played in the current game, or are held by the learning agent.
- $p_t(2)$: the number of diamond cards that have been played in the current game, or are held by the learning agent.
- $p_t(3)$: the number of spade cards (♠2,$\cdots$,♠J) that have been played in the current game, or are held by he learning agent.
- $p_t(4)$, $p_t(5)$, $p_t(6)$: the probability that agent M^1, M^2 and M^3 have the ♠Q, respectively.
- $p_t(7)$: the status of the ♠K.
- $p_t(8)$: the status of the ♠A.
- $p_t(9)$ to $p_t(21)$: the status of each of the heart cards.
- $p_t(22)$ to $p_t(25)$: a bit sequence representing who is the leading player in the current trick.

Since the most important card is the ♠Q in the game of Hearts, we use three dimensions to represent its predictive allocation. The game rules tell us the following facts.

1. If agent M^i did not play a spade card when the leading card was a spade card in a past trick of the current game, $p_t(i+3)$ is zero.
2. $p_t(4) + p_t(5) + p_t(6) = 1$

Under the limitation from these two facts, the probability that agent M^i has the ♠Q, $p_t(i+3)$, is calculated as a uniform probability. The status of the ♠K, the ♠A, or a heart card is represented by one of three values, $-1, 0$ or 1, corresponding to the cases when the card has already been played in the current game, when it is held by the opponent agents, or when it is held by the learning agent, respectively. The bit sequence represents the playing order in the current trick.

In this study, the state evaluation module is trained so as to approximate $V(p_t)$ for an input p_t. This learning is done by equations (9) and (10) where x_t and x_{t+1} are replaced by p_t and p_{t+1}, respectively.

3.2 Action predictor

In the action selection module, there are three action predictors. The action predictor for agent M^i predicts a card played by that agent, in a similar manner to the action selection by the actor in the actor-critic algorithm. In order to predict an action by agent M^i at his t-th turn, the i-th action predictor calculates a merit function value $U^i(\hat{y}_t^i(a_t, H_t, K), a_t^i)$ for the mean estimated observation $\hat{y}_t^i(a_t, H_t, K)$ and a possible action a_t^i. After calculating the merit value for every possible action, an action a_t^i is selected with the predicted probability

$$P(a_t^i|\hat{y}_t^i(a_t, H_t, K), \phi^i) = \frac{\exp(U^i(\hat{y}_t^i(a_t, H_t, K), a_t^i)/T^i)}{\sum_{a_t^i \in \mathcal{A}} \exp(U^i(\hat{y}_t^i(a_t, H_t, K), a_t^i)/T^i)} \tag{13}$$

Here, $\mathcal{A}$ denotes the set of possible actions for agent M^i, and T^i is a constant parameter that denotes the assumed randomness of the action selection of agent M^i.

When training the action predictor for agent M^i, the merit function, $U^i(\hat{y}_t^i(a_t, H_t, K), a_t^i)$ is updated similarly to the actor learning. $\hat{y}_t^i(a_t, H_t, K)$ is reproduced by replaying a past game, and a_t^i is the action actually taken by agent M^i at his t-th play in the past game.

We use a function approximator for representing the merit function and use a feature extraction technique as well as in the state evaluation module.

An input to the function approximator, q_t^i, is given by the transformation from the mean estimated observation $\hat{y}_t^i$ as follows.

- $q_t^i(1)$: if the leading card is a club card, the expected number of club cards held by agent M^i, which are weaker than the strongest card already played in the current trick, otherwise zero.

- $q_t^i(2)$: if the leading card is a club card, the expected number of club cards held by agent M^i, which are stronger than the strongest card already played in the current trick, otherwise the expected number of club cards held by the agent.

- $q_t^i(3)$: similar to $q_t^i(1)$, but the suit is diamond.

- $q_t^i(4)$: similar to $q_t^i(2)$, but the suit is diamond.

- $q_t^i(5)$: if the leading card is a spade card, the expected number of spade cards (♠2,$\cdots$,♠J) held by agent M^i, which are weaker than the strongest card already played in the current trick, otherwise zero.

- $q_t^i(6)$: if the leading card is a spade card, the expected number of spade cards (♠2,$\cdots$,♠J) held by agent M^i, which are weaker than the strongest card already played in the current trick, otherwise the expected number of spade cards (♠2,$\cdots$,♠J) held by the agent.

- $q_t^i(7)$: the expectation value for that agent M^i has the ♠Q.

- $q_t^i(8)$: the expectation value for that agent M^i has the ♠K.

- $q_t^i(9)$: the expectation value for that agent M^i has the ♠A.

- $q_t^i(10)$ to $q_t^i(22)$: the expectation value for that agent M^i has each of the heart cards.

- $q_t^i(23)$ to $q_t^i(26)$: a bit sequence representing who is the leading player in the current trick.

Let $C_t^i(\spadesuit Q)$ be 1 or 0 when agent M^i has or does not have, respectively, the ♠Q in his hand just before his t-th turn, for example. The expectation value of the binomial variable $C_t^i(\spadesuit Q)$ is equivalent to the probability that agent M^i has the ♠Q in his hand:

$$\hat{C}_t^i(\spadesuit Q|a_t, H_t, K) = P(C_t^i(\spadesuit Q) = 1|a_t, H_t, K). \quad (14)$$

The game rules tell us the following facts.

1. If agent M^i did not play a card of the same suit as the leading card in a past trick of the current game, M^i does not have at present any card of this suit.

2. The cards, except for those held by the learning agent and those that have already been played in the current game, may exist in the hand of agent M^i.

Under the limitation from these two facts, the card existence probability in the hand of agent M^i is assumed to be uniform. $\hat{C}_t^i(\text{a-card}|a_t, H_t, K) \in [0, 1]$, which represents the expectation value for that agent M^i has 'a-card' in his hand, is then calculated with respect to the distribution. $q_t^i(7), ..., q_t^i(22)$ correspond to $\hat{C}_t^i(\spadesuit Q|a_t, H_t, K), ..., \hat{C}_t^i(\heartsuit A|a_t, H_t, K)$, respectively. $q_t^i(1), ..., q_t^i(6)$ are calculated by using $\hat{C}_t^i(\clubsuit 2|a_t, H_t, K), ..., \hat{C}_t^i(\heartsuit J|a_t, H_t, K)$. Namely, q_t^i is given by the transformation from the estimated card existence probability $\hat{C}_t^i$. It should be noted that $\hat{C}_t^i$ is similar to the mean expected observation $\hat{y}_t^i$. An input to the function approximator is thus given by the transformation from the mean estimated observation $\hat{y}_t^i$.

The action predictor is trained so as to output the following 26-dimensional vector:

- $r_t^i(1)$: if the leading card is a club card, the merit value for that agent M^i plays a weaker club card than the strongest card already played in the current trick.

- $r_t^i(2)$: if the leading card is a club card, the merit value for that agent M^i plays a club card that is stronger than the strongest card already played in the current trick, and the weakest one in the hand of M^i.

- $r_t^i(3)$: if the leading card is a club card, the merit value for that agent M^i plays a club card that is stronger than the strongest card already played in the current trick, and neither the weakest nor the strongest in the hand of M^i.

- $r_t^i(4)$: if the leading card is a club card, the merit value for that agent M^i plays a club card that is stronger than the strongest card already played in the current trick, and the strongest one in the hand of M^i.

- $r_t^i(5)$: similar to $r_t^i(1)$, but the suit is diamond.

- $r_t^i(6)$: similar to $r_t^i(2)$, but the suit is diamond.

- $r_t^i(7)$: similar to $r_t^i(3)$, but the suit is diamond.

- $r_t^i(8)$: similar to $r_t^i(4)$, but the suit is diamond.

- $r_t^i(9)$: if the leading card is a spade card, the merit value for that agent M^i plays a weaker spade card among $\spadesuit 2, ..., \spadesuit J$ than the strongest card already played in the current trick.

- $r_t^i(10)$: if the leading card is a spade card, the merit value for that agent M^i plays a stronger spade card among $\spadesuit 2, ..., \spadesuit J$ than the strongest card already played in the current trick.

- $r_t^i(11)$: the merit value for that agent M^i plays the ♠Q.

- $r_t^i(12)$: the merit value for that agent M^i plays the ♠K.

- $r_t^i(13)$: the merit value for that agent M^i plays the ♠A.

- $r_t^i(14)$ to $r_t^i(26)$: the merit value for that agent M^i plays each of the heart cards.

Both of the input dimension and the output dimension of the function approximator are 26. The action predictor for agent M^i calculates the estimated card existence probability $\hat{C}_t^i$, and then the input to the function approximator, q_t^i, is calculated from $\hat{C}_t^i$. This calculation corresponds to the process expressed by equation (4). Then, the function approximator of the action predictor outputs the merit function r_t^i for the input q_t^i. After that, r_t^i is transformed into the merit function $U^i(q_t^i, a_t^i)$, and then a possible action is selected by equation (13), in which $U^i(\hat{y}_t^i, a_t^i)$ is replaced by $U^i(q_t^i, a_t^i)$.

3.3 Action selector

The action selector determines an action based on the action selection rule (8). In order to obtain the expected TD error (6), it is necessary to estimate the transition probability $P(x_{t+1}|a_t, \Phi, H_t, K)$ (equation (5)). In order to calculate equation (5), it is necessary to estimate $\hat{y}_t^i(a_t, H_t, K)$ (equation (4)) and then to calculate $P(a_t^i|\hat{y}_t^i(a_t, H_t, K), \phi^i)$. The estimation of $\hat{y}_t^i(a_t, H_t, K)$ is replaced by the estimation of $q_t^i(a_t, H_t, K)$, and the calculation of $P(a_t^i|\hat{y}_t^i(a_t, H_t, K), \phi^i)$ is approximately done by equation (13). By producing every possible combination of actions, (a_t^1, a_t^2, a_t^3), equation (5) is calculated, and then the expected TD error is obtained using the probability (5) for every possible new observation state x_{t+1}.

Especially when there are a lot of cards that can be played in the t-th trick, however, the complete retrieval for every possible combination of cards played in the trick and for every possible new observation state is difficult. This difficulty partly corresponds to the difficulty of the calculation of the summation $\sum_{(a_t^1,a_t^2,a_t^3)\in\mathcal{S}_{t+1}^+}$ in equation (3). In order to overcome this difficulty, we use a pruning technique. For each possible action for agent M^i at his t-th play, a_t^i, the action predictor calculates a merit value $U^i(q_t^i, a_t^i)$ for a pair of the reduced mean estimated observation $q_t^i(a_t, H_t, K)$ and action a_t^i. Then, a state transition due to an action whose merit value is fairly small is dropped in the further evaluation; this introduces pruning within the game tree, in order to obtain efficiently the summation in equations (5) and (7).

For the remaining actions, the action probability is determined as

$$P(a_t^i|\hat{y}_t^i(a_t, H_t, K), \phi^i) \approx \frac{\exp(U^i(q_t^i(a_t, H_t, K), a_t^i)/T^i)}{\sum_{a_t^i\in\mathcal{A}^+}\exp(U^i(q_t^i(a_t, H_t, K), a_t^i)/T^i)} \quad (15)$$

instead of equation (13), where $\mathcal{A}^+$ denotes the set of actions that are not dropped.

3.4 Function approximator

Since the state space of a realistic problem like that of the game of Hearts is huge and it is difficult for the learning system to experience every possible state, the generalization ability of function approximators is very important.

In this study, we use normalized Gaussian networks (NGnet's) as function approximators. An NGnet is defined as

$$O = \sum_{i=1}^{m}\frac{G_i(I)}{\sum_{j=1}^{m}G_j(I)}(W_i I + b_i) \quad (16)$$

$$G_i(I) = (2\pi)^{-\frac{N}{2}}|\Sigma_i|^{-\frac{1}{2}} \times \exp[-\frac{1}{2}(I-\mu_i)'\Sigma_i^{-1}(I-\mu_i)], \quad (17)$$

where I denotes an N-dimensional input vector and O denotes an N_a-dimensional output vector. m denotes the number of units. Σ_i is an $N \times N$ covariance matrix, μ_i is an N-dimensional center vector, W_i is an $N_a \times N$ weight matrix, and b_i is an N_a-dimensional bias vector, for the i-th unit. Prime ($'$) denotes a transpose.

An NGnet can be defined as a probabilistic model, and its maximum likelihood inference is done by an on-line expectation-maximization (EM) algorithm [10]. The on-line EM algorithm is based on a stochastic gradient method, and it is faster than gradient methods. Therefore, the learning of the action predictors is so fast that our RL method can be applicable to a situation where the strategies of the opponent agents may change with time[2].

4 Computer simulations

4.1 Single agent learning in a stationary environment

We carried out experiments using one learning agent and three rule-based opponent agents.

A rule-based agent is an "experienced" level player of the game of Hearts. It has more than 50 rules, and the usage of them has been tuned by playing a lot of games with humans. The acquired penalty ratio was 0.41 when an agent who only took out permitted cards

[2]Although the approximation accuracy is dependent on the number of units, m in equation (16) and (17), its automatic determination method based on the probabilistic interpretation is implemented in the on-line EM algorithm.

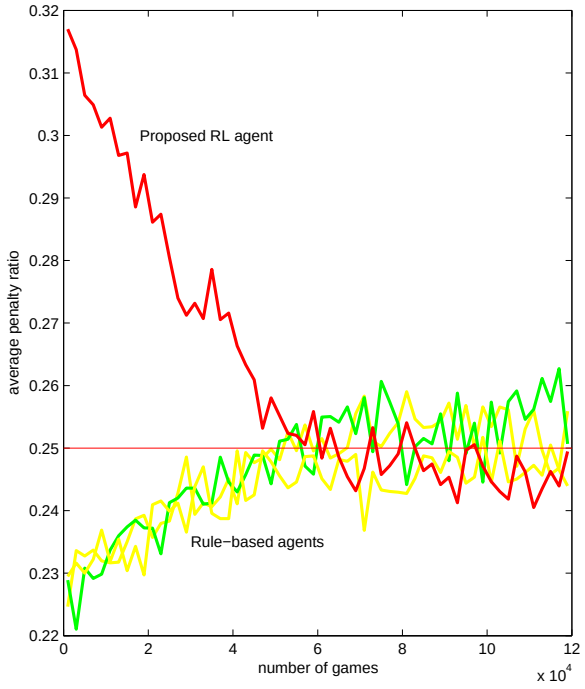

Figure 1: Penalty ratio is smoothed by using 2000 games just before that number of training games.

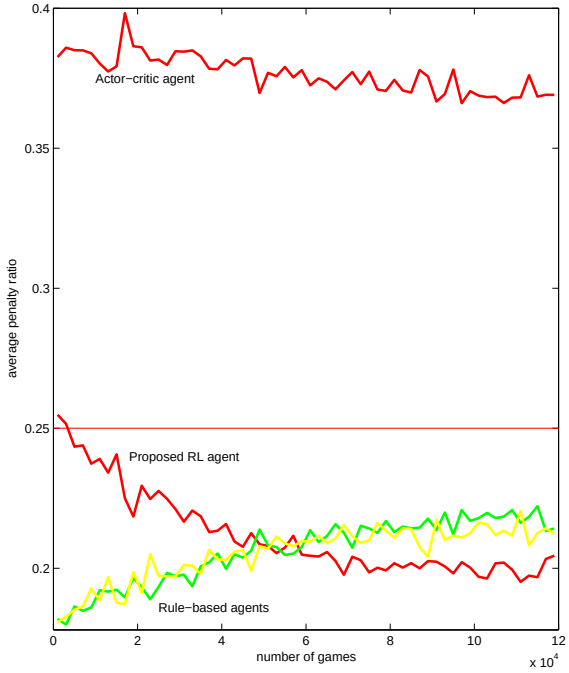

Figure 2: Penalty ratio is smoothed by using 2000 games just before that number of training games.

at random from its hand challenged the three rule-based agents. The acquired penalty ratio is the ratio of the acquired penalty points of the learning agent to the total penalty points of the four agents. That is, a random agent acquired about 2.1-fold points of rule-based agents on average.

Figure 1 shows the learning curve of an agent trained by our RL method when it challenged the three rule-based agents. This learning curve is an average over three learning runs, each of which consists of 120,000 training games. After about 80,000 games playing with the three rule-based agents, our RL agent comes to acquire a smaller penalty ratio than the rule-based agents. Namely, the learning agent gets stronger than the rule-based agents.

4.2 Learning of multiple agents in a multi-agent environment

So far, our RL method has been based on the POMDP approximation, namely, it is assumed that there is only one learning agent in the environment. We try here to apply our RL method directly to multi-agent environments, in which there are multiple learning and hence dynamic agents.

Figure 2 shows the result when one learning agent trained by our RL method, one learning agent based on the actor-critic algorithm, and two rule-based agents played against each other. The actor-critic agent is modified into such as to use feature extraction techniques for its actor and critic, which are similar to those used in our RL method. Although the average penalty ratio of our RL agent became smaller than those of the rule-based agents after about 120,000 training games, the learning agent trained by the actor-critic algorithm was not improved much. This result implies that our model-based approach within the POMDP formulation is more efficient than a model-free approach, i.e., the actor-critic algorithm.

Figure 3 shows the result when two learning agents trained by our RL method and two rule-based agents played with each other. After about 40,000 games, our RL agents came to acquire a smaller penalty ratio than the rule-based agents, and then after about 55,000 games, two learning agents could beat the rule-based agents.

These two simulation results, figures 2 and 3, show that our RL method can be applied to the concurrent learning of multiple agents in a multi-agent environment. This applicability is partly attributed to the fast learning by efficient function approximators. In our RL method, we individually prepare an action predictor that approximates the policy of each opponent agent. We consider this implementation is suitable for application to multi-agent environments. Each action evaluator is able to deal with the characteristics of the corresponding opponent agent independently of the other opponent agent. In addition, if the strategy of one agent changes, it is enough for a single function approximator to adapt to the change independently. It is then expected that

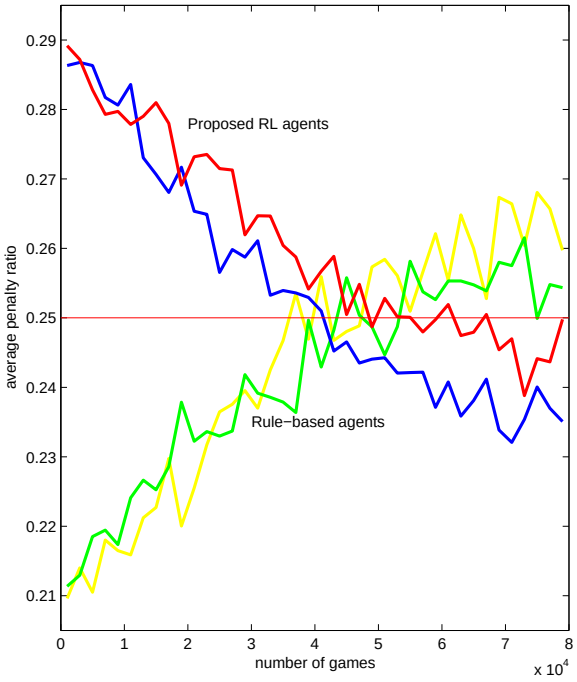

Figure 3: Penalty ratio is smoothed by using 2000 games just before that number of training games.

the RL process is stable even in a concurrent learning setting in a multi-agent environment.

5 Summary

In this article, we proposed an RL scheme for making an autonomous learning agent that plays a multi-player card game "Hearts". Since the game of Hearts is an imperfect-information game, the RL scheme is formulated as a POMDP. The learning agent estimates unobservable state variables and predicts actions of the opponent agents. Then, it copes with the partial observability in a multi-agent environment. In the state representation used by function approxmators, feature extraction techniques are used in order to accelerate their learning. Our model-based POMDP-RL method with efficient state representations has successfully been applied to a realistic multi-agent problem.

References

[1] A.Pèrez-Uribe and A. Sanchez. Blackjack as a test bed for learning strategies in neural networks. Proc. IEEE International Joint Conference Neural Networks, vol3, pp.2022-2027, 1998.

[2] T.Yoshioka, S.Ishii, and M.Ito. Strategy acquisition for game Othello based on min-max reinforcement learning. IEICE Trans. Inf. & Syst., vol.E82-D, no.12, pp.1618-1626, Dec. 1999.

[3] G.Tesauro. TD-Gammon, a self-teaching Backgammon program, achieves master-level play. Neural Computation, vol.6, no.2, pp.215-219, 1994.

[4] M.Ginsberg. Imperfect information in a computationally challenging fame. *Journal of Artificial Intelligence Research*, 14, pp.303-358, 2001.

[5] L.P.Kaelbling, M.L.Littman, and A.R.Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, Vol.101, 1998.

[6] M.L.Littman. Markov games as a framework for multi-agent reinforcement learning. *Proceedings of the 11th International Conference on Machine Learning*, pp.157-163, 1994.

[7] A.G.Barto, R.S.Sutton, and C.W.Anderson. Neuronlike adaptive elements that can solve difficult learning control problems IEEE Trans. Syst., Man. & Cybern., vol.13, pp.834-846, 1983.

[8] A.W.Moore and C.G. Atkeson. Prioritized sweeping: reinforcement learning with less data and less real time. *Machine Learning*, 13, pp.103-130, 1993.

[9] Y.Matsuno, T.Yamazaki, J.Matsuda, and S.Ishii. A multi-agent reinforcement learning method for a partailly-observable competitive game. Fifth International Conference on Autonomous Agents, pp39-40, 2001.

[10] M.Sato and S.Ishii. On-line EM algorithm for the normalized Gaussian network. Neural Computation, vol.12, no.2, pp.407-432, 2000.