Opponent Modeling with Incremental Active Learning: A Case Study of Iterative Prisoner's Dilemma

Hyunsoo Park
Dept. of Computer Science and Engineering
Sejong University
Seoul, South Korea
hspark@sju.ac.kr

Kyung-Joong Kim*
Dept. of Computer Science and Engineering
Sejong University
Seoul, South Korea
kimkj@sejong.ac.kr

***Abstract*—What's the most important sources of information to guess the internal strategy of your opponents? The best way is to play games against them and infer their strategy from the experience. For novice players, they should play a lot of games to identify other's strategy successfully. However, experienced players usually play a small number of games to model other's strategy. The secret is that they intelligently design their plays to maximize the chance of discovering the most uncertain parts. Similarly, in this paper, we propose to use an incremental active learning for modeling opponents. It refines the other's models incrementally by cycling "estimation (inference)" and "exploration (playing games)" steps. Experimental results with Iterative Prisoner's Dilemma games show that the proposed method can reveal other's strategy successfully.**

Keywords—iterative prisoner's dilemma; estimation-exploration algorithm; theory of mind; game theory

I. INTRODUCTION

For some games, opponent modeling is one of the most important parts of playing. For example, in the game of poker, there is a bluffer who pretends to have a good hand with bad cards. Players should recognize the opponent's style (bluffer or not) as early as possible to make reasonable decision. In real-time strategy games, experienced players use their resource to identify other players' strategy in the early stage of games from scouting. Recently, there has been interest in mimicking human's scouting in the real-time strategy games [1].

The modeling of an opponent can be formulated as a kind of "reverse engineering" problem. The opponent has a "hidden" model to make decisions and the player can observe only outcomes (plays) of the model. To infer the "hidden" model, the player can interact with the opponent by playing games but the number of interaction should be limited. For example, you are not allowed to play a huge number of games against your opponent to recognize his strategy. It is necessary to play intelligently extracting the most useful sequence of plays from the opponent in a limited number of interactions.

Kim *et al*. propose to use "reverse engineering" algorithm to infer the internal control mechanism of robotic agents [2]. Initially, the Observer robot records the behaviors (trajectory) of the Actor robot from a random starting position. With the first trajectory, the "estimation" algorithm searches for multiple candidate models on the Actor's internal model. Although there are good initial guess on the target, there is a chance to improve the quality of models by using additional information (trajectories) of the Actor. In "exploration" stage, the Observer calculates the disagreement of the predictions of the multiple candidates on all conditions (starting points). The algorithm attempts to manipulate the Actor at the point where the predictions disagree with the maximum. In this way, the Observer adds a new trajectory into his memory and runs again the "estimation" algorithm to improve the quality of models incrementally.

In this work, we formulate the opponent modeling in games as a kind of "reverse engineering" problems and apply the estimation-exploration algorithm (EEA) to reveal the internal "hidden" model with limited number of interactions. Unlike the robotic scenarios, in the games, it is more difficult to manipulate the others to show some desirable behaviors (a sequence of actions). The opponents can play with some noise (random change of his/her behavior) to hide their strategy. Furthermore, the some experienced players can control the games to minimize the exposure of his strategy in the early stage of the games.

Iterative prisoner's dilemma (IPD) games have been widely used to model real-world conflicts with a simplified decisions (defect or cooperate) [3]. In the game, two players select one of the two choices and get payoff from their decisions. In short term, you can get high score by defecting others but your opponent should react with "defect" minimizing your gain. Also, there are a lot of different strategies which make difficult to design a golden rule that maximizes gain against most of the opponents. In this work, we propose to use the EEA (incremental active learning) to reveal the strategy of IPD games. It shows that the EEA can model the opponent in the existence of the noisy actions.

II. APPLYING ESTIMATION-EXPLORATION ALGORITHM TO IPD GAMES

A. *Proposed Method*

In IPD, it is assumed that the player's memory length is $2N$. Each memory unit (length = 2) stores a pair of decisions of two players. They are one of (C, C), (C, D), (D, C) and (D, D) (C: cooperate, D: defect). In other words, the player exploits his memory on the decision of two players in the past

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (2010-06589, 2010-0018950).

*Corresponding author

978-1-4673-5311-3/13/$31.00

N games to determine the next action. The player's strategy can be represented as a decision table returns the decision ('C' or 'D') to the current memory configuration (length = $2N$). The decision table has 2^{2N} rows (all possible memory configuration) and $2N$+1 columns. The last column in the table stores the action of the player with the specific memory configuration in the row.

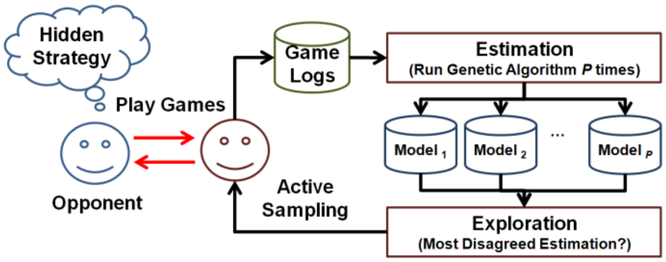


Fig. 1. Overview of proposed method

The decision table of the opponent is defined as DT_{target}. Initially, the player has no game logs with the opponent. At first, the player selects a sequence of his actions randomly and plays games with the opponent. The action logs are used in the genetic algorithm (GA) to calculate fitness values (the similarity between the action logs and the predictions by the candidate decision table). Each individual in the population represents a decision table (the number of parameters is 2^{2N}). Each parameter specifies the action ('C' or 'D') of the opponent to the corresponding memory configuration. Because we run the GA multiple times (with different random seed), it outputs multiple decision tables DT_1, DT_2, $\cdots$, DT_P (the best one from each GA run).

In the "exploration stage," the algorithm calculates the disagreement of the estimation of the multiple decision tables. For each row, it calculates the agreement of the last column (the action of opponent) among the candidates. It selects the most disagreed (necessary to be probed (or tested) in the next games) memory configuration. In the next game, the player intentionally plays with the actions specified in the most disagreed configuration. After the games, the player stores the new log into the game logs, the GA runs again multiple times with the logs (old logs and the new one). The algorithm repeats the cycle (estimation-exploration).

Although the decision tables are a good representation for the IPD strategy, the number of parameters is exponentially increasing with the memory length. Also, it requires huge memory space to store them. In addition to the decision table models, we also train a multi-layer neural network which emulates the decision tables with the game logs.

B. Experimental Results

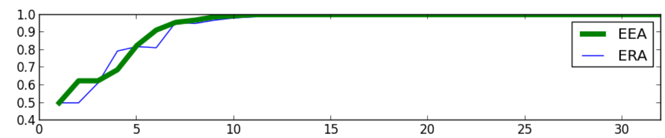
Table 1 summarizes the parameters used in the experiments. In the experiments, we use the EEA (with the active sampling mechanism), ERA (Estimation-Random Algorithm, it is similar to the EEA but the sampling is done randomly). In real games, the opponent could play in different actions with his decision table to hide his strategy. To simulate this behavior, we add 10% random flipping of actions.

- Opponent A : Cooperate when the opponent cooperates and vice versa (also known as "Tit-for-tat")
- Opponent B: Opponent A except that it defects after two cooperation.
- Opponent C: Opponent A with 10% random actions

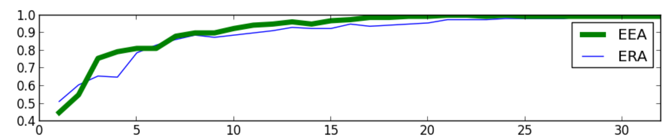
TABLE I. PARAMETERS USED IN EXPERIMENTS

Parameters	Value
The number of models in the estimation stage (P)	3
The number of games played for each game log	32
Memory length	4
Population size of the GA	20
Maximum number of generations in the GA	50

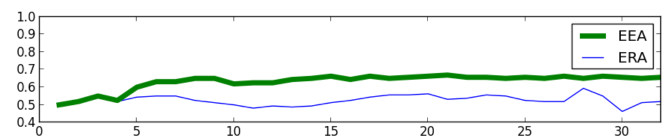
Fig. 2 shows that the increase of accuracy over the EEA cycles. It shows that the EEA works well with the small number of interactions to infer the other's models. In case of 10% random actions, the EEA outperforms the ERA.



(a) Opponent A



(b) Opponent B



(c) Opponent C

Fig. 2. Experimental results (x-axis: # of samples used in the estimation stage, y-axis: mean accuracy of predictions of 10 experiments)

III. CONCLUSIONS

In this paper, we propose to use the incremental active learning to infer the strategy of IPD games. It shows that the EEA can predict the opponent's actions with only a small number of interactions. Introducing random actions to the opponent significantly reduces the accuracy but the EEA are more robust than the passive sampling algorithm.

REFERENCES

[1] H.-S. Park, H.-C. Cho, K.-Y. Lee, and K.-J. Kim, "Prediction of early stage opponent strategy for StarCraft AI using scouting and machine learning," *Workshop at SIGGRAPH ASIA (Computer Gaming Track)*, pp. 7-12, 2012.

[2] K.-J. Kim, K.-Y. Eo, Y.-R. Jung, S.-O. Kim, and S.-B. Cho, "Evolutionary conditions for the emergence of robotic theory of mind with multiple goals," *IEEE Workshop on Robotic Intelligence in Informationally Structured Space (RiiSS)*, pp. 48-54, 2013

[3] R. M. Axelrod, *The Evolution of Cooperation*, NY, Basic Books, 2006.