Modeling Non-Stationary Opponents

(Extended Abstract)

Pablo F. Hernandez-Leal, Enrique Munoz de Cote and L. Enrique Sucar
Instituto Nacional de Astrofísica, Óptica y Electrónica
Sta. Maria Tonantzintla, Puebla, México
{pablohl,jemc,esucar}@ccc.inaoep.mx

ABSTRACT

This paper studies repeated interactions between an agent and an unknown opponent that changes its strategy over time. We propose a framework for learning switching non-stationary strategies. The approach uses decision trees to learn the most up to date opponent's strategy. Then, the agent's strategy is computed by transforming the tree into a Markov Decision Process (MDP), whose solution dictates the optimal way of playing against the learned strategy. The agent's learnt model is continuously re-evaluated to assess strategy switches. Our method detects such strategy switches by measuring tree similarities, and reveals whether the opponent has changed its strategy and a new model has to be learned. We evaluated the proposed approach in the iterated prisoner's dilemma, outperforming common strategies against stationary and non-stationary opponents.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms

Keywords

Opponent modeling; Decision Trees; Markov Decision Process; Iterated Prisoner's Dilemma

1. INTRODUCTION

Robust autonomous agents that operate in open environments should be able to interact effectively with an unknown agent, adapting themselves in response to the opponent. In order to deal with such uncertainty, it is common practice to assume that the counterpart's strategy is rational and stationary [1].

2. MDP4.5 LEARNING STRATEGY

We propose a framework (called MDP4.5) for learning switching non-stationary strategies in repeated games using a novel approach based on decision trees and Markov Decision Processes (MDPs) [3]. In a nutshell, opponent models are constructed via decision trees. These are then converted into a MDP to obtain an optimal strategy against that opponent. Since the opponent might change its strategy, our approach learns different trees throughout the game.

2.1 Decision trees as opponent models

Consider the case where two agents interact several rounds: agent X uses the proposed approach and agent Y continuously switches its strategy during the game. In the beginning of the game it starts with an initial random strategy to promote exploration and learn an accurate model of the opponent. After a number of interactions, agent X learns a decision tree of the opponent using the C4.5 algorithm [4]. The tree will describe the opponent strategy based on the previous interaction between agents.

2.2 From Decision trees to MDPs

Once we have learnt a model of an opponent, we need to be able to play against such strategy. To do so we propose a novel way to transform decision trees into MDPs. By so doing, we can now compute an optimal policy by solving the (now MDP) model of the opponent. In Figure 1 (a) a decision tree with just one decision node and two leaves is depicted. A decision tree that contains as decision nodes only the last n-actions induces a MDP. Solving this MDP gives an optimal policy (strategy) against that opponent.

Each MDP induced by a decision tree D is a tuple $< S, A, T, R >$. The set of states S corresponds to the paths that lead to each leaf of D. The set of actions A is the set of available actions for the agent X. The transition function $T : S \times A \rightarrow S$ is generated using the error percentage of each leaf and the actions in A. Finally the reward function $R : S \times A \rightarrow \mathcal{R}$ is defined by a bimatrix that depends only on the joint actions of the agents. This bimatrix and the decision tree will be used to construct R.

As an example, the tree of Figure 1 (a) and the classic matrix of the prisoner's dilemma with rewards $C = 3, D = 1$, $T = 4, S = 0$ are transformed into the MDP depicted in Figure 1 (c).

Now, it can be the case that an opponent is wired to switch its strategy at specific time and the learnt decision node will contain a feature like *round* that can not be mapped easily to an MDP (see Figure 1 (b)). When this happens we use a hybrid approach. Since each branch of the tree is also a tree, then the branches of this *round* node are the ones converted into MDPs. Thus a general decision tree will induce a set of MDPs, where each leaf will have an associated MDP.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

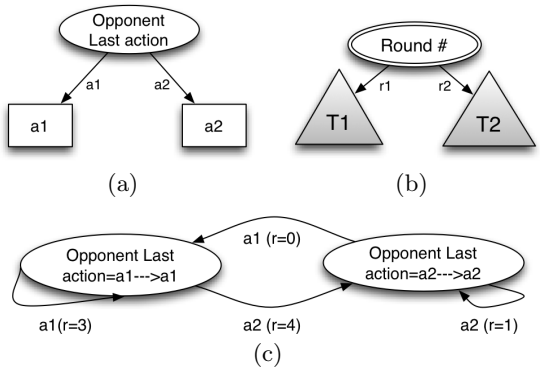

Figure 1: (a) A decision tree that corresponds to a model of an agent. It contains one decision node *Opponent Last action* and two leaves that correspond to the actions a_1 and a_2. (b) A decision tree that has *Round #* as decision node, therefore each branch (T1 and T2) will be transformed into a MDP. (c) This represents a state transition diagram of the MDP with two states (ovals). The arrows represent the transition using action a_x, with the immediate reward in parenthesis.

2.3 Assessing strategy switches

Once the model is converted into a MDP and solved, a strategy is generated. Agent X will start playing with this strategy as soon as it is available. However, the opponent might change its strategy to a different one due to agent X's change or stay with the same. In order to differentiate these two different outcomes we keep two decision trees learning concurrently. One of these (tree a) will learn for w steps and will be reset afterwards, while the other (tree b) is never reset (keeping the entire history). Trees a and b are compared every w steps to evaluate their *similarity*. For this, we use a distance metric presented in [2] which considers the structure and prediction. If the *distance* between these trees is greater than a threshold δ, the opponent has changed strategy and the modeling agent must restart the learning phase, resetting both trees and starting from scratch. Otherwise, it means that the opponent has not switched strategies and k is incremented waiting for a new batch of interactions.

3. EXPERIMENTS AND RESULT

In order to evaluate the proposed framework we apply it in the iterated PD, since it is a largely studied problem. We compare the results of the MDP4.5 agent against the most successful reported strategies in the iPD: Tit for Tat (TFT), Pavlov, Grim and Tit for Two Tats (TFTT). The games have a size of 250 rounds.

To evaluate if the proposed approach can detect the changes in strategy, the opponent will switch between two strategies; it will start with one from {TFT, Pavlov, Grim, TFTT, Random} and in middle of the game it will change to another strategy from the same set. In the upper part of Table 1 we summarize the results for different strategies playing against the non-stationary opponents, each result is the average of 80 games. From the results we can observe that MDP4.5 obtained the best average utility and the second the best result in joint utility (agent utility + opponent utility).

Table 1: Average sum of utilities over 250 rounds. The upper part presents the results against a non-stationary opponent. The lower part summarizes the results against an stationary opponent.

Versus oppponent with non-stationary strategies			
Agent	Agent Util.	Opponent Util.	Joint Util.
MDP4.5	**576.5**	537.2	1113.7
TFT	521.0	522.4	1043.4
Pavlov	531.2	573.7	1105.0
Grim	448.7	299.3	748.1
TFTT	540.7	**625.0**	**1165.7**
Versus opponent with stationary strategy			
Agent	Agent Util.	Opponent Util.	Joint Util.
MDP4.5	**551.8**	521.4	1072.7
TFT	500.0	501.6	1001.6
Pavlov	545.5	606.9	1152.5
Grim	416.5	306.3	722.9
TFTT	550.9	**616.3**	**1167.2**

In the lower part of Table 1 we present the results against an opponent with a stationary strategy (TFT, Pavlov, Grim or TFTT) each result is the average of 20 games. From the results we can observe that out approach obtained the best result in average and this shows that our agent can also be used in simple situations against stationary strategies.

Some conclusions drawn from the experiments are that our agent is capable of detecting changes in the opponent strategy since it obtained high utilities for itself and also for the opponent in most of the cases; for opponents using TFT and Pavlov it converges to the C-C actions. In the case of Grim opponents the results are far form the best because the strategy is wired to switch to action defect (against only one defection from its adversary). The unique best response to such switch is defect and therefore the strategies get caught on the outcome D=1 forever after the first defection.

4. CONCLUSIONS

Our contribution is a novel framework for learning switching non-stationary strategies in repeated games using decision trees to represent the opponent model and transforming those into MDPs to compute an optimal policy against that opponent. As of today we are gathering more experimental evidence to evaluate the robustness of the approach.

5. REFERENCES

[1] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[2] R. Miglio and G. Soffritti. The comparison between classification trees through proximity measures. *Computational Statistics and Data Analysis*, 45(3):577–593, Apr. 2004.

[3] M. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 1994.

[4] J. Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.