Detecting Switches Against Non-Stationary Opponents (JAAMAS Extended Abstract)

Pablo Hernandez-Leal
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
Pablo.Hernandez@cwi.nl

Yusen Zhan
Washington State University
Pullman, Washington, USA
yzhan@eecs.wsu.edu

Matthew E. Taylor
Washington State University
Pullman, Washington, USA
taylorm@eecs.wsu.edu

L. Enrique Sucar
INAOE
Puebla, México
esucar@inaoep.mx

Enrique Munoz de Cote
INAOE & PROWLER.io Ltd.
Puebla, México & Cambridge, UK
jemc@inaoep.mx

ABSTRACT

Interactions in multiagent systems are generally more complicated than single agent ones. Game theory provides solutions on how to act in multiple agent scenarios; however, it assumes that all agents will act rationally. Moreover, some works also assume the opponent will use a stationary strategy. These assumptions usually do not hold in real world scenarios where agents have limited capacities and may deviate from a perfect rational response. Our goal is still to act optimally in this cases by learning the appropriate response and without any prior policies on how to act. Thus, we focus on the problem when another agent in the environment uses different stationary strategies over time. This paper introduces DriftER, an algorithm that 1) learns a model of the opponent, 2) uses that to obtain an optimal policy and then 3) determines when it must re-learn due to an opponent strategy change. We provide theoretical results showing that DriftER guarantees to detect switches with high probability. Also, we provide empirical results in normal form games and then in a more realistic scenario, the Power TAC simulator.

Keywords

Non-stationary opponents; repeated games; Markov decision processes

1. INTRODUCTION

When different agents interact in real world scenarios they may use different behaviors depending on the context they encounter. For example, in domains such as poker playing agents may use different strategies depending on the opponent's behavior, in trading and negotiation scenarios, opponents use different strategies and change among them. One example is the Power TAC simulator [5] where competing brokers (agents) are challenged to maximize their profits by buying energy from a wholesale market and then offering energy services to customers. A champion agent from a previous competition was TacTex [6], which uses an approach based on reinforcement learning and prediction methods. Even though TacTex learns to bid efficiently (in terms of profit), it is not capable of adapting quickly to non-stationary opponents (that change suddenly to a different strategy). In general, when agents can change among several stationary strategies, they turn the environment into a non-stationary one. This is especially problematic for most learning algorithms which assume a stationary environment and most algorithms will not to react rapidly to sudden changes, causing sub-optimal performance.

Works from machine learning have studied detection of changes mostly in supervised learning settings, this area is commonly known a *concept drift*. However, this is only a partial representation of our problem. In the area of reinforcement learning some approaches have studied how agents should act against non-stationary agents in order to converge to an equilibrium.

Against this background, this paper's main contribution is to introduce DriftER, Drift (based on) Error Rate, which leverages the idea of concept drift to detect when the opponent has changed strategies based on a measure on predictive error. In order to learn how to act, DriftER assumes no prior information of the opponents instead assumes to know the set of attributes the opponent uses to define its strategy[1] and starts with an exploratory policy. DriftER treats the opponent as part of a stationary environment using a Markov decision process to model its behavior [1] and keeps track of the quality of the learned MDP model.

2. DRIFTER

DriftER [4] learns a model of the opponent which is used to compute a policy to act against it. DriftER's interaction with a stationary opponent generates Markovian observations which can be used to learn an MDP that represents the opponent's strategy assuming to know the representation (attributes) used by the opponent. This is because the history of interactions define the transition among states and the learning agent can induce an MDP [1] that models the opponent strategy and can compute an optimal policy against it π^* (assuming the opponent will remain fixed). In all settings, after interacting with the opponent for w timesteps, the environment is learned using the R-max exploration [2] and later we value iteration to solve the MDP.

[1]These can be, for example, previous actions of the agents.

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

Many learning techniques decrease their exploration rate over time. However, when facing non-stationary opponents whose model has been learned, an agent must balance exploitation (to perform optimally against that strategy) and exploration (to attempt to detect switches in the opponent). Opponent strategy switches can be particularly hard to detect producing a "shadowing" effect in the agents perception and the agent will not detect something has changed (unless some exploration occurs). DriftER uses "drift exploration" that solves this shadowing effect by continuously exploring the state space even after an optimal policy has been learned [3].

After learning a model of the opponent, DriftER must decide on each timestep if the model is consistent (the predictions using that model are correct) or the opponent has changed to a different strategy (the model has consistently shown errors). Using the current MDP that represents the opponent strategy, DriftER predicts the next state of the MDP (which for example can correspond to the opponent next action). In the next timestep DriftER compares the predicted and the experienced true state. This comparison can be binarized with *correct/incorrect* values. A Bernoulli process $S_1, S_2, \dots, S_T$ will be produced, assuming a sequence of independent identically distributed events where $S_i \in \{0,1\}$ and T is the last timestep. Let $\hat{p}_i$ be the estimated error (probability of observing *incorrect*) from S_1 to S_i, $i = 1, \dots, T$. Then, the 95% confidence interval $[f_{lower}(\hat{p}_i), f_{upper}(\hat{p}_i)]$ over $S_1, S_2, \dots, S_i$ is calculated for each timestep i using the Wilson score [7] such that the confidence interval will improve as the amount of data grows, where $f_{lower}(\hat{p}_i)$ and $f_{upper}(\hat{p}_i)$ denote the lower bound and upper bound of the confidence interval, respectively.

The estimated error, can increase for two reasons: the opponent is exploring (or make mistakes) or a switch has occurred in the opponent's strategy. To detect the latter, DriftER tracks the finite difference of the confidence interval using the upper bound $f_{upper}(\hat{p}_i)$ at each timestep i. The finite difference is defined by

$$\Delta_i = f_{upper}(\hat{p}_i) - f_{upper}(\hat{p}_{i-1}), \quad i = 1, \dots, T.$$

If $\Delta_i > 0$, $\Delta_{i-1} > 0$,...,$\Delta_{i-n+1} > 0$, (where $n = 1, 2, \dots$ is a parameter determined by the domain), DriftER detects the confidence interval is increasing in the last n steps, then it decides to restart the learning phase.

DriftER provides a theoretical result to justify this method is capable of detecting opponent switches with high probability making the following assumptions: (i) The opponent does not switch strategies while DriftER is in the learning phase. (ii) The probability of exploration or mistake of the opponent is at most ϵ for each timestep.

THEOREM 1. *Let $\epsilon > 0$ and $\delta > 0$ be small constants. If $\Delta_i > 0$, $\Delta_{i-1} > 0$,...,$\Delta_{i-n+1} > 0$ and we set $n = O(\log \delta / \log \epsilon)$, then DriftER detects the opponent switch with probability $1 - \delta$.*

We contrast the behavior of DriftER (black thick line) and a learning agent that does *not* include a switch detection mechanism (blue line) against the same non-stationary opponent in Figure 1 which depicts the upper value of the confidence over the error (f_{upper}). Initially the opponent uses a stochastic policy (showing an f_{upper} value close to 0.2). At round 750 (vertical red line) the opponent changes to a different strategy resulting in suboptimal performance for the agent without switch detection. In contrast, DriftER

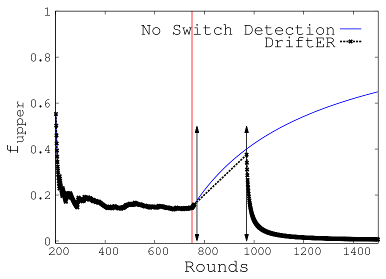

Figure 1: Error probabilities of a learning algorithm without switch detection and DriftER against an opponent that changes between two strategies in the middle of the interaction (vertical bar), small arrows represent DriftER learning phase after detecting the switch.

detects the switch and starts a learning phase (between arrows) after which DriftER produces a new opponent model, therefore its error will decrease.

3. CONCLUSIONS

In real world scenarios different agents interact witch each other, so it is reasonable to expect that they have different behaviors. We focus on the problem when other agent in the environment use different stationary strategies over time. This paper introduced DriftER, an algorithm that models an opponent as an MDP in order to compute an optimal policy to behave against it. Then, it uses the learned model to estimate the opponent's behavior and tracks its error rate to detect opponent switches. When the opponent changes its strategy, the error rate increases and DriftER must learn a new model. Theoretical results provide a guarantee of detecting switches with high probability. Empirical results in repeated games and the Power TAC simulator show that DriftER can be adapted to more realistic scenarios.

Acknowledgments

This research has received funding through the ERA-Net Smart Grids Plus project Grid-Friends, with support from the European Union's Horizon 2020 research and innovation programme.

REFERENCES

[1] B. Banerjee and J. Peng. Efficient learning of multi-step best response. In *Proceedings of the 4th AAMAS Conference*, pages 60–66, Utretch, Netherlands, 2005. ACM.

[2] R. I. Brafman and M. Tennenholtz. R-MAX a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, 2003.

[3] P. Hernandez-Leal, Y. Zhan, M. E. Taylor, L. E. Sucar, and E. Munoz de Cote. An exploration strategy for non-stationary opponents. *Autonomous Agents and Multi-Agent Systems*, Oct. 2016.

[4] P. Hernandez-Leal, Y. Zhan, M. E. Taylor, L. E. Sucar, and E. Munoz de Cote. Efficiently detecting switches against non-stationary opponents. *Autonomous Agents and Multi-Agent Systems*, Nov. 2016.

[5] W. Ketter, J. Collins, and P. P. Reddy. Power TAC: A competitive economic simulation of the smart grid. *Energy Economics*, 39:262–270, Sept. 2013.

[6] D. Urieli and P. Stone. TacTex'13: A Champion Adaptive Power Trading Agent. In *Twenty-Eighth AAAI Conference*, pages 465–471, Quebec, Canada, May 2014.

[7] E. B. Wilson. Probable Inference, the Law of Succesion, and Statistical Inference. *Journal of the American Statistical Association*, 22(158):209–212, Jan. 1927.