Identifying and Tracking Switching, Non-stationary Opponents: a Bayesian Approach

Pablo Hernandez-Leal[1], Matthew E. Taylor[2], Benjamin Rosman[3], L. Enrique Sucar[1] and Enrique Munoz de Cote[1]
[1]Instituto Nacional de Astrofísica, Óptica y Electrónica
Sta. María Tonantzintla, Puebla, Mexico
[2]Washington State University
Pullman, Washington, USA
[3]Council for Scientific and Industrial Research, and the University of the Witwatersrand
South Africa

Abstract

In many situations, agents are required to use a set of strategies (behaviors) and switch among them during the course of an interaction. This work focuses on the problem of recognizing the strategy used by an agent within a small number of interactions. We propose using a Bayesian framework to address this problem. Bayesian policy reuse (BPR) has been empirically shown to be efficient at correctly detecting the best policy to use from a library in sequential decision tasks. In this paper we extend BPR to adversarial settings, in particular, to opponents that switch from one stationary strategy to another. Our proposed extension enables learning new models in an online fashion when the learning agent detects that the current policies are not performing optimally. Experiments presented in repeated games show that our approach is capable of efficiently detecting opponent strategies and reacting quickly to behavior switches, thereby yielding better performance than state-of-the-art approaches in terms of average rewards.

Introduction

A core problem in multiagent systems and human-computer interaction is being able to identify the behaviors of other (target) agents. When such a problem is tackled correctly, it allows an agent to derive an optimal policy against such behavior (Banerjee and Peng 2005). Consider, for example, a poker player whose behavior has been identified by the opponent player (Bard et al. 2013). Then, the opponent can act optimally against such player, anticipating every move. Another example are service robots. In this domain, robots need to interact with persons that have not one but a set of different behaviors (strategies) depending, for example, on factors such as personality or physical capabilities. For each of these behaviors, the robot needs to use an appropriate policy to achieve maximal performance on the task. Since people change of behaviors the robot needs adapt quickly to changes. Moreover, in this domain it is desirable that robots engage in various forms of social learning to learn new tasks (Breazeal 2004). When a new task happens it needs to be learned and then optimized.

In general, behavioral changes can be intentional (e.g. to keep the opponent guessing, or to change to a more profitable behavior) or unintentional (e.g. a human that loses interest in a task or a robot that looses some actuator). In either case, the problem faced by the learning agent is threefold. First, it needs to rapidly detect and identify which strategy is being used by the target agent. Second, as the target agent may change behavior during the interaction, the system needs to be able to track and adapt to these switches, with an appropriate response policy. Third, in the case that the target agent uses a new strategy unknown to the learning agent, which may result in suboptimal behavior using the current policies, the agent must detect this and learn how to optimize against the new opponent strategy.

The target opponent agents that we focus on use a fix (stationary) strategy for an unknown number of interactions, switching to another fixed strategy and repeating this switching to different strategies. There has been some literature addressing this problem (Bowling and Veloso 2002; Da Silva et al. 2006; Elidrisi et al. 2014; Hernandez-Leal, Munoz de Cote, and Sucar 2014a; 2014b), however, no work has focused on the problem of reusing previously seen strategies to boost learning speeds.

The Bayesian policy reuse (BPR) framework (Rosman, Hawasly, and Ramamoorthy 2015) has been proposed to determine *quickly* the best policy to select when faced with an unlabeled (but previously seen) task. However, standard BPR assumes knowledge of all possible tasks and optimal policies from the start. Our contribution, BPR+, is an extension to BPR for adversarial settings against non-stationary opponents. In particular, we relax the assumption of knowing all opponent strategies *a priori* by providing an online learning approach for incorporating models of new strategies when current policies perform suboptimally.

Related Work

Our work relates to learning in non-stationary environments, since switches between stationary environments are a special case of non-stationarity. The win or learn fast (WOLF) idea was introduced to adapt to learning algorithms. WOLF-PHC (Bowling and Veloso 2002) is an extension of Q-learning that performs hill-climbing in the space of mixed policies. This algorithm was designed for converging with opponents that slowly change behavior, without sudden strategy changes. A related approach is reinforcement learning with context detection (RL-CD) (Da Silva et al. 2006), which learns several partial models and selects one to use depending on the context of the environment. However, this has not been tested in multiagent scenarios. Hidden-mode Markov decision processes (HM-MDPs) (Choi, Yeung, and Zhang 1999) assume the environment —in our case an opponent behavior— can be represented using a small number of *modes* (each representing a stationary environment), which have different dynamics and therefore need different policies. At each time step only one mode is active, and modes cannot be directly observed. HM-MDPs require the solution of a POMDP to obtain a policy, they are not capable of learning new models online and are hard to solve in general (PSPACE-complete) (Papadimitriou and Tsitsiklis 1987). The fast adaptive learner (FAL) approach (Elidrisi et al. 2014) has focused on fast learning in two-person stochastic games. It maintains a set of hypotheses according to the history of observations in order to predict the opponent's action. To obtain a strategy against the opponent, they use a modified version of the Godfather strategy (Littman and Stone 2001) which restricts its use. Additionally, FAL experiences an exponential increase in the number of hypotheses (in the size of the observation history), which may limit its use in larger domains. The MDP-CL framework (Hernandez-Leal, Munoz de Cote, and Sucar 2014a) is a model-based approach to handle switching opponents. It learns an opponent strategy as a stationary environment, inducing an MDP based on the interaction history. Solving that MDP provides an optimal policy to act. To detect opponent switches MDP-CL learns a different model periodically and compares it to the current model to evaluate their similarity. If the distance between models is greater than a threshold, it is considered that the opponent has changed strategy and it must restart the learning phase. *A priori* MDP-CL (Hernandez-Leal, Munoz de Cote, and Sucar 2014b) assumes the set of possible strategies $\mathcal{M}_i$ (represented by MDPs) used by the opponent is known. The problem, similar to our scenario, is to detect quickly the strategy used by the opponent and adapt to possible switches. In *a priori* MDP-CL, the correct model will have a perfect similarity with at least one of the $\mathcal{M}_i$ models with enough experience tuples. When this happens, it stops the random exploration phase, and computes a policy against it. A limitation of *a priori* MDP-CL is that it explores randomly until reaching a perfect similarity with one of the $\mathcal{M}_i$ models.

In contrast to previous approaches, BPR+ uses the information of how the policies behave to guide the exploration, potentially reducing the time to detect the model.

Preliminaries

Our approach is developed in the context of repeated games. Consider two players (A and B) that face each other and repeatedly play a *bimatrix game*. A bimatrix game is a two player simultaneous-move game defined by the tuple $\langle \mathcal{A}, \mathcal{B}, R_A, R_B \rangle$, where $\mathcal{A}$ and $\mathcal{B}$ are the set of possible actions for player A and B respectively, and R_i is the reward matrix of size $|\mathcal{A}| \times |\mathcal{B}|$ for each agent $i \in \{A, B\}$, where the payoff to the ith agent for the joint action $(a, b) \in \mathcal{A} \times \mathcal{B}$ is

Table 1: Bimatrix representing the prisoner's dilemma: two agents chose between two actions, cooperate and defect.

	Cooperate	*Defect*
Cooperate	c, c	s, d
Defect	d, s	p, p

given by the entry $R_i(a, b), \forall (a, b) \in \mathcal{A} \times \mathcal{B}, \forall i \in \{\text{A,B}\}$. A *stage game* is a single bimatrix game and a series of rounds of the same stage game form a *repeated game*. A *strategy* specifies a method for choosing an action. A *best response* for an agent is the strategy (or strategies) that produce the most favorable outcome for that player, taking other players' strategies as given.

The prisoner's dilemma is a well-studied two player game where the interactions can be modeled as a symmetric two-player game defined by the payoff matrix in Table 1 where the following two conditions must hold $d > c > p > s$ and $2c > d + s$. When both players cooperate they both obtain the reward c. When both defect, they receive a punishment reward p. A player choosing to cooperate with someone who defects receives the sucker's payoff s, whereas the defecting player gains the temptation to defect, d. The iterated version (iPD) has been subject to many studies, including human trials. One well-known strategy for iPD is called Tit-for-Tat (TFT) (Axelrod and Hamilton 1981), which starts by cooperating, and thereafter chooses the action taken by the opponent in the previous round. Another successful strategy is called Pavlov, which cooperates if both players previously coordinated with the same action and defects whenever they did not. Bully (Littman and Stone 2001) is another well-known strategy involving always behaving as a defecting player.[1] We use this game for experimentation, as we can use the documented strategies to build switching opponents.

Bayesian Policy Reuse

Bayesian Policy Reuse (Rosman, Hawasly, and Ramamoorthy 2015) has been proposed as a framework to determine quickly the best policy to select when faced with an unknown task. Formally, a task is defined as an MDP, $\mathcal{M} = \langle S, \mathcal{A}, T, R \rangle$, where S is the set of states, $\mathcal{A}$ is the set of actions, T is the transition function and R is the reward function. A *policy* is a function $\pi(s)$ that specifies an appropriate action a for each state s. The return, or utility, generated from running the policy π in an interaction of a task instance is the accumulated reward, $U^\pi = \sum_{i=0}^{k} r_i$, with k being the length of the interaction and r_i being the reward received at step i. Solving an MDP $\mathcal{M}$ is to acquire an optimal policy $\pi^\star = \arg\max U_\pi$ which maximizes the total expected return of $\mathcal{M}$.

Let an agent be a decision making entity in some domain, and let it be equipped with a policy library Π for tasks in that domain. The agent is presented with an unknown task which must be solved within a limited and small number of trials. At the beginning of each trial episode, the agent can select one policy from $\pi \in \Pi$ to execute. The goal of the agent is

[1]These strategies can be defined only by the current state and do not depend on the time index; they are *stationary* strategies.

thus to select policies to minimize the total regret incurred in the limited task duration with respect to the performance of the best alternative from Π in hindsight.

In order to select which policy will be used, different mechanisms can be used. A heuristic for policy selection $\mathcal{V}$ is a function that estimate a value for each policy which measures the extent to which it balances exploitation with a limited degree of look-ahead for exploration.

The *probability of improvement* heuristic for policy selection is through the probability with which a specific policy can achieve a hypothesized increase in performance over the current best estimate, $\hat{U}^t(\pi) = \sum_{\tau \in \mathcal{O}} \beta^t(\tau) E[U|\tau, \pi]$. thus, the heuristic chooses the policy that maximizes $\arg\max_{\pi \in \Pi} \sum_{\tau \in \mathcal{O}} \beta(\tau) E[U^+|\tau, \pi]$ where $U^+ > \hat{U}^t(\pi)$.

BPR assumes knowledge of performance models describing how policies behave on different tasks. A *performance model* $P(U|\tau, \pi)$ is a probability distribution over the utility using π on a task τ. A signal σ is any information that is correlated with the performance of a policy and that is provided to the agent in an online execution of the policy on a task (e.g., the immediate reward). For a set of tasks $\mathcal{O}$ and a new instance $\tau^\star$ the *belief* β is a probability distribution over $\mathcal{O}$ that measures to what extent $\tau^\star$ matches the known tasks in their observation signals σ. The belief is initialized with a prior probability. After each execution on the unknown task the environment provides an observation signal to the agent, which is used to update beliefs according to Bayes' rule:

$$\beta^t(\tau) = \frac{P(\sigma^t|\tau, \pi^t)\beta^{t-1}(\tau)}{\sum_{\tau' \in \mathcal{O}} P(\sigma^t|\tau, \pi^t)\beta^{t-1}(\tau).}$$

In summary, BPR starts with a set of policies Π and faces different tasks $\mathcal{O}$. BPR is given performance models $P(U|\mathcal{O}, \Pi)$ for how each policy behaves over each task. BPR initializes the belief over tasks, and for each episode of the interaction BPR selects a policy to execute (according to the belief and exploration heuristic) and receives an observation signal σ which is used to update the belief.

Other policy reuse algorithms exist, e.g. in the work of (Fernández and Veloso 2006), but their approach uses policy reuse to seed learning of new, similar behaviors, which is slower, whereas we assume that these will be directly reused; neither have been used against non-stationary opponents. Multi-armed bandits (Gittins 1979) are another related approach. However, one major difference is that the performance and signal models allow us to keep more detailed information about how the different arms (policies) in bandits relate to each other leading to faster convergence.

BPR+

This section describes BPR+, which handles non-stationary opponents and learns new models in an online manner. BPR was presented in a single agent environment facing different tasks (represented by MDPs). BPR+ extends to a multiagent setting. Now the tasks correspond to opponent strategies and the policies correspond to optimal policies against those stationary strategies. To cope with these new type of environment, some considerations need to be taken, (i) an exploration for switch detection is added: drift exploration. (ii) A method for detecting when an unknown opponent strategy appears is needed. (iii) A method for learning the new opponent strategy (and then an new optimal policy). (iv) An algorithm for updating the performance models to add the information from the recently learned model and policy. These points are presented in more detail in the following sections.

Drift exploration and switch detection

Against non-stationary opponents, exploration cannot be terminated, as is often the case (Munoz de Cote et al. 2010). In fact it is critical since strategy switches can be difficult to detect, particularly if the strategies in question are very similar. Such similarities can produce a "shadowing" effect (Fulda and Ventura 2006) in the agent's perception — an agent's optimal policy $\pi^\star$ will produce an ergodic set of states against some opponent strategy, but if the opponent's switching strategy induces a similar MDP where the policy $\pi^\star$ produces the same ergodic set, the agent will not detect something has changed (unless some new kind of exploration occurs).

Definition 1 *Drift exploration is a type of exploration tailored for non-stationary environments that guarantees that opponent switches will not pass unnoticed.*

BPR+ uses "drift exploration" that solves this shadowing effect by exploring the state space even after an optimal policy has been learned. ϵ-greedy or softmax exploration are useful approaches to drift exploration.

Recall that BPR obtains information from every interaction to update its belief. Against stationary opponents, the belief may converge to a perfect score and the probability of alternative opponent models will drop to zero. However, with non-stationary opponents, we require that the belief may be able to increase to handle opponent switches. To solve this, BPR+ restricts all beliefs to not drop to zero by adding a small $\gamma > 0$ value.

New model detection

We additionally consider the case where the opponent is employing a novel strategy, necessitating model learning. This scenario is identified through receiving a sequence of rewards which are unlikely given the known performance models.

Let r^i_π be the immediate reward obtained in round i using policy π. The probability of obtaining r^i_π given an opponent strategy τ using policy π is (given by the performance models): $p^\tau_i = P(r^i_\pi|\tau, \pi)$. Let $\hat{p}^\tau_i$ be the moving average of rewards received during rounds $i - n, \ldots, i$: $\hat{p}^\tau_i = \sum_{j=1,\ldots,n} \frac{p^\tau_{i-j}}{n}$.

If, for all opponent strategies τ, $\hat{p}^\tau_i < \rho$ in the last n steps, then BPR+ begins to learn a new model. Both ρ and n are parameters of the algorithm, the former controls how different the probabilities need to be (compared to the known opponent strategies) and the latter controls the number of rounds needed before consider learning a new opponent strategy. In summary, BPR+ computes the probability of the rewards under the known models, and if this probability is lower than a threshold (ρ) for some number of rounds (n), then a new model will be learned.

Learning opponent strategies

After BPR+ detects a new strategy it needs to learn an optimal policy to be used. The opponent behavior is modeled trough an MDP, $\mathcal{M}_{new}$, composed of $\langle S, \mathcal{A}, R, T \rangle$ where S is the set of states $S = \times_{z_i \in Z} z_i$, i.e. each state is formed by the cross product of a set of attributes Z. We assume to know these attributes, for example in the iPD these can be the last action of each agent. $\mathcal{A}$ is the action set of the learning agent. The transition function $T : S \times \mathcal{A} \rightarrow S$ is learned using counts $\hat{T}(s,a,s') = \frac{n(s,a,s')}{m(s,a)}$ with $n(s,a,s')$ the number of times the agent was in state s, used action a and arrived at state s', $m(s,a)$ is defined as the number of times the agent was in state s and used action a. The reward function R is learned similarly: $\hat{R}(s,a) = \frac{\sum r(s,a)}{m(s,a)}$ with $r(s,a)$ the reward obtained by the agent when being in state s and performing action a. In Figure 1, a learned MDP with four states is depicted, it represents the model of an opponent that uses the Pavlov strategy in the iPD game (using rewards shown in Table 1). Each state is formed by the last play of both agents, C for cooperate, D for defect, subindices *opp* and *learn* refer to the opponent and the learning agent respectively (other attributes can be used for different domains), with arrows indicating the triplet: action, transition probability and immediate reward for the learning agent. Note that the last play defines the *state* of the learning agent.

To learn the parameters of the MDP an exploratory phase is needed (for example, with random actions). We assume the opponent will not change of strategies during the learning phase (a number of rounds) after which an MDP that represents the opponent strategy is obtained. Since the rewards are deterministic, solving the MDP induced by the opponent is done through value iteration (Bellman 1957), obtaining an optimal policy $\pi^{\star}_{new}$ (Puterman 1994). If the model correctly represents the opponent's strategy, the policy against that opponent will result in a maximization of total rewards against the opponent strategy.

Once $\mathcal{M}_{new}$ is obtained, a performance model can be computed: a policy π can be simulated on $\mathcal{M}_{new}$, representing the opponent $\mathcal{O}$ to produce a list of rewards $r_0, r_1, \ldots, r_n$. These values are modeled as a probability distribution (we used a Gaussian distribution) to obtain the performance model $P(U|\mathcal{O}, \pi)$.

Updating performance models

Consider the following scenario. BPR+ is given an initial set of performance models $P(U|\mathcal{O}, \Pi)$, its associated MDPs ($\mathcal{M}$) and optimal policies Π against those opponent strategies. BPR+ is also capable of detecting a switch to a strategy which is not in $\mathcal{O}$. The problem is then how to add the information from this new strategy O_{new} (and the obtained optimal policy, π_{new}) to the current set of performance models.

This requires a series of steps in the set of performance models. These are divided into different groups: (1) the known performance models $P(U|\mathcal{O}, \Pi)$ (could be $\emptyset$), (2) the performance models using initial policies against the new opponent strategy $P(U|\mathcal{O}_{new}, \Pi)$, (3) the performance

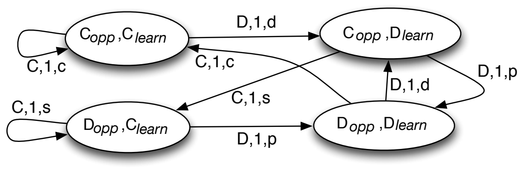


Figure 1: An induced MDP representing the opponent's dynamics. An arrow represents transition, action and reward of the learning agent.

model of the new strategy using the new optimal policy $P(U|\mathcal{O}_{new}, \pi_{new})$, and (4) the information from the new optimal policy against the previously known opponent strategies $P(U|\mathcal{O}, \pi_{new})$.

After detecting a new opponent strategy, BPR+ starts an exploratory phase to learn it (the MDP). This MDP is used to: i) obtain the performance models using the initial policies (group 2), ii) obtain a new optimal policy π_{new} which is added to the set Π and a new performance model is obtained (group 3) and iii) obtain the performance models of the initial opponent strategies with the newly computed policy (group 4). Now BPR+ can optimize its policy against the new opponent and at the same time its performance models are updated accordingly to be able to detect switches to either a new or a previously known strategy. A summary of this process is presented in Algorithm 1 and the complete BPR+ algorithm is shown in Algorithm 2.

Algorithm 1: Learning of performance models in BPR+

```
Input: Known performance models $P(U|\mathcal{O}, \Pi)$ (group 1), opponent models
       $\mathcal{M}$ and optimal policies $\Pi$
1 if new opponent strategy detected, $O_{new} \notin \mathcal{O}$ then
2     Learn opponent strategy $\mathcal{M}_{new}$ and solve to get optimal policy $\pi_{new}$.
3     Compute $P(U|\mathcal{O}_{new}, \Pi)$ with old policies (models in group 2).
4     Compute $P(U|\mathcal{O}_{new}, \pi_{new})$ (models in group 3).
5     Compute $P(U|O, \pi_{new})$ using $\pi_{new}$ on models $\in \mathcal{M}$ (models in
      group 4).
6     $\Pi = \Pi \cup \pi_{new}$
7     $\mathcal{M} = \mathcal{M} \cup \mathcal{M}_{new}$
```

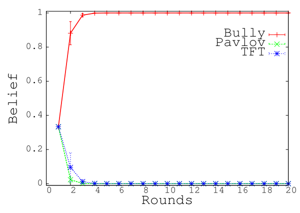
Example 1 *Assume that BPR+ agent is playing the iPD against a non-stationary opponent which uses TFT, Pavlov and Bully strategies ($\mathcal{O}$). BPR+ has faced the opponent strategies TFT and Pavlov. Thus, BPR+ has the models of those strategies, $\mathcal{M}$, its optimal policies against those Π, and the performance models (represented by Gaussians) $P(U|\mathcal{O}, \Pi)$ (as group 1). Then, the opponent changes to a Bully strategy which is detected as new one. The model is learned and its optimal policy is obtained (as described in Section). Finally, the performance models are updated as described in Algorithm 1 (groups 2, 3 and 4).*

Experiments

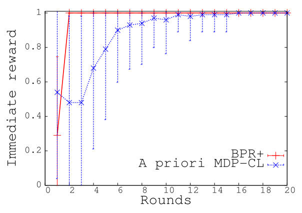
In this section we present experiments in the iPD with values $c = 3, d = 4, s = 0, p = 1$. Opponent strategies

Algorithm 2: BPR+ algorithm

```
Input: Policy library $\Pi$, prior probabilities $P(\mathcal{O})$, performance models
       $P(U|\mathcal{O},\Pi)$, episodes $K$, exploration heuristic $\mathcal{V}$
1 Initialize beliefs $\beta^0(\mathcal{O}) = P(\mathcal{O})$
2 for episodes $t = 1,\dots,K$ do
3     Compute $v_\pi = \mathcal{V}(\pi, \beta^{t-1})$ for all $\pi \in \Pi$
4     $\pi^t = argmax_{\pi\in\Pi} v_\pi$
5     Use $\pi_t$ against the opponent
6     Obtain observation signal $\sigma^t$ from environment
7     Update belief $\beta^t$ using Bayes' rule and restrict to $> 0$
8     Check for new opponent strategy (Algorithm 1)
```



(a) BPR+ belief against Bully



(b) Average rewards when opponent is Bully

Figure 2: Beliefs of BPR+ against the stationary opponent Bully. Immediate rewards of BPR+ and *a priori* MDP-CL against opponent Bully Error bars represent one standard deviation. BPR+ is able to quickly identify a known opponent with high probability and play optimally against it.

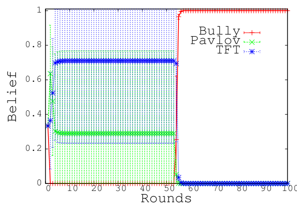
are TFT, Pavlov and Bully. Optimal policies are different against each strategy.[2] Experiments are divided in four parts: i) BPR+ against stationary opponents. ii) BPR+ is then evaluated against non-stationary opponents (showing the importance of drift exploration) and ii) finally we present experiments using the online learning approach.

Model detection against stationary opponents

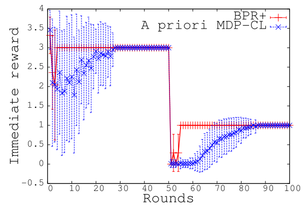
In this setting, BPR+ is given the performance models and optimal policies against Bully, TFT and Pavlov. BPR+ starts with a uniform distribution over the opponent strategies. The objective of BPR+ is to detect with high probability and within a small number of rounds which opponent strategy it is facing, and to use the appropriate response policy.

Each repeated game consisted of 20 rounds and results are averaged over 100 iterations. Figure 2 (a) depicts the belief of BPR+ for an opponent strategy Bully. Against Bully, the belief for Bully reaches the maximum score in 3 steps. When the opponent uses Pavlov, the belief for Pavlov is the maximum of the three strategies, after step 4 although it does not obtain the maximum belief (1.0). Analyzing individual games reveals that BPR+ sometimes converges to TFT. This is because Pavlov will appear as TFT (or vice versa) for histories of cooperation. It is important to note that even when beliefs converge to the incorrect model (in some cases) the rewards are optimal (in terms of best response) and against

[2]Optimal policies are always cooperate, Pavlov and always defect, against opponents TFT, Pablo and Bully, respectively.



(a) BPR+ belief for opponent TFT-Bully



(b) Average rewards for opponent TFT-Bully

Figure 3: Belief and rewards of BPR+ against opponents that switch strategies in middle of the interaction of 100 trials. Error bars represent one standard deviation. BPR+ is able to quickly identify a known opponent that switch strategies during the interaction with high probability and play optimally against it.

all opponent strategies BPR+ obtained the maximum reward in less than 5 steps. In contrast to *a priori* MDP-CL which needs approximately 15 rounds to reach the optimal reward against Bully and more than 20 rounds for Pavlov.

Non-stationary opponents

Now we play against non-stationary opponents that change from one strategy to another during the interaction. The game was increased to 100 rounds and BPR+ starts with the complete set of performance models but it does not know when the opponent switch will occur.

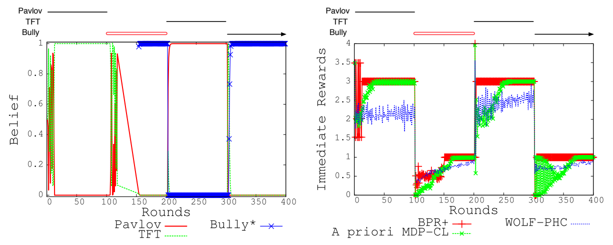
In Figure 3 (a) beliefs and (b) rewards are depicted for an opponent that starts with the TFT strategy and changes to Bully in the middle of the interaction (100 trials). BPR+ is capable of detecting the switch within 3 rounds and thereafter obtains the new optimal reward, in contrast to *a priori* MDP-CL which takes more than 30 rounds to reach the new optimal score.

Table 2 presents results of BPR+ with and without drift exploration in terms of immediate rewards. The drift exploration used explores more when the recent rewards are low and reduces exploration when they are high (in order to detect Bully-TFT switches). Using this drift exploration notably improved the results against Bully-TFT and slightly decreased performance against the other opponents, as expected. However, note that it is behaviorally similar to the omniscient agent, detecting all switches rapidly. As conclusion we note that when an opponent switches between strategies different scenarios may occur: 1) the switch is detected and BPR+ correctly uses a new policy, 2) the switch is not detected but it does not affect the rewards (e.g., switches between TFT and Pavlov), or 3) the switch is not detected, i.e. the Bully-TFT opponent. Without drift exploration, the agent is not capable of detecting the switch because TFT behaves as Bully for histories of defection (shadowing behavior).

Table 2 compares BPR+ (with and without drift exploration), *a priori* MDP-CL, WOLF-PHC and the Omniscient (perfect) agent that best responds immediately after a switch against different non-stationary opponents. Results are the average of 100 trials, bold typeface represents the best score among learning algorithms. Statistical significance of BPR+

Table 2: Average rewards with std. dev. (±) for different non-stationary opponents. Each opponent changes from one strategy to another in the middle of the interaction in a repeated game of 100 rounds. Results are averaged over 100 trials. Statistical significance of BPR+ against BPR+ without drift exploration, *a priori* MDP-CL and WOLF-PHC is represented respectively by ‡, $*$, and †.

Opponent	Omniscient	BPR+	BPR+ (without drift exp.)	*A priori* MDP-CL	WOLF-PHC
Bully-TFT	2.0	**1.67 ± 0.24**‡ $*$ †	0.99 ± 0.01	1.29 ± 0.23	1.09 ± 0.17
Bully-Pavlov	2.0	1.93 ± 0.02 $*$†	**1.98 ± 0.01**	1.75 ± 0.07	1.47 ± 0.19
Pavlov-TFT	3.0	**2.99 ± 0.01**$*$†	**2.99 ± 0.01**	2.81 ± 0.21	2.09 ± 0.44
Pavlov-Bully	2.0	1.92 ± 0.02$*$†	**1.97 ± 0.01**	1.70 ± 0.12	1.51 ± 0.28
TFT-Bully	2.0	1.90 ± 0.04 $*$†	**1.95 ± 0.02**	1.65 ± 0.11	1.34 ± 0.22
TFT-Pavlov	3.0	**2.98 ± 0.03**$*$†	**2.99 ± 0.02**	2.81 ± 0.16	2.08 ± 0.38
Average	2.33	**2.23 ± 0.06**	2.15 ± 0.01	2.00 ± 0.15	1.60 ± 0.28



(a) Average belief of BPR+ agent (b) Average rewards of different algorithms

Figure 4: The upper portion of each figure shows the opponent strategies (thick line means a unknown strategy to BPR+) used at every round. (a) BPR+ beliefs with incomplete performance models (b) BPR+, *a priori* MDP-CL and WOLF-PHC rewards

with a priori MDP-CL, WOLF-PHC and BPR+ (without drift exploration) is indicated with $*$, † and ‡. From the results it can be noticed that BPR+ obtained the best scores (significantly better) among the learning agents. One interesting case happens against the Bully-TFT, where not using drift exploration in BPR+ causes the algorithm to not detect the switch (this is explained in more detail in the next section). As shown previously, *a priori* MDP-CL is able to adapt to the opponents but it requires more rounds to detect and learn the new opponent strategy, obtaining lower rewards. WOLF-PHC obtained the worst results showing slower convergence.

Unknown strategies: online learning

This section evaluates BPR+ against a switching opponent which uses unknown strategies. Now BPR+ must detect a new strategy, learn the associated MDP, optimize against it and update its performance models. Values for parameters were $\rho = 0.8$ and $n = 7$ (experimentally selected).

In the first experiment, BPR+ only has information about TFT and Pavlov strategies. However, the opponent will use the strategies TFT-Bully-Pavlov-Bully (changing every 100 rounds). Figure 4 depicts (a) the beliefs in BPR+ and (b) rewards[3] for BPR+, *a priori* MDP-CL and WOLF-PHC. For the known opponents (TFT and Pavlov) BPR+ takes about 4 rounds to reach the optimal reward. From round 100 the opponent uses Bully which is not known to BPR+. The beliefs and rewards start oscillating, and BPR+ detects this as a new opponent strategy. BPR+ then initiates a learning phase which finishes at approximately round 160, at that point it can compute an optimal policy against Bully. At round 300 the opponent returns to a Bully strategy. However, now the BPR+ agent does not need to relearn it, but rather takes about 3 rounds to select the optimal policy. *A priori* MDP-CL takes longer to obtain the optimal score when a switch happens but it is capable of reaching the best score. WOLF-PHC shows a slower convergence obtaining lower cumulative rewards.

[3]Note that in this domain an immediate reward greater than 3 can be obtained, however, this is not sustainable in the long term.

Table 3: Average rewards with std. dev. (±) of the learning agents (average of 100 trials). The opponent changes strategy randomly every 200 rounds. BPR+ starts without any information $*$, † represent statistical significance with *a priori* MDP-CL and WOLF-PHC respectively.

Learning agent	Avg. rewards ± std.dev
Omniscient	2.32 ± 0.00
BPR+	**2.17 ± 0.27** $*$†
A priori MDP-CL	2.11 ± 0.24
WOLF-PHC	1.77 ± 0.19

Finally, we tested against an opponent that changes its strategy randomly every 200 rounds in a game of 2000 rounds. Table 3, shows immediate rewards for the different approaches (average of 100 trials). BPR+ has no prior information of the opponent (group 1 is $\emptyset$) nor initial policies. Results show that BPR+ adapts faster to switches which results in statistically significant better scores.

Conclusions

We propose a Bayesian approach to cope with non-stationary opponents (that change from one stationary strategy to another) in repeated games. In contrast to previous works, BPR+ is able to exploit information of how the policies behave against different strategies in order to asses the best policy faster than any algorithm in literature. Moreover, BPR+ has an online learning algorithm for adding models to its library to deal with unknown strategies. This line of work is very attractive specially for applications where the early stages are crucial for the rest of the interactions, for example in human-computer interaction. As future work we plan to test other domains and provide theoretical guarantees of switch detection.

References

Axelrod, R., and Hamilton, W. D. 1981. The evolution of cooperation. *Science* 211(27):1390–1396.

Banerjee, B., and Peng, J. 2005. Efficient learning of multi-step best response. In *Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems*, 60–66. Utretch, Netherlands: ACM.

Bard, N.; Johanson, M.; Burch, N.; and Bowling, M. 2013. Online implicit agent modelling. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, 255–262. International Foundation for Autonomous Agents and Multiagent Systems.

Bellman, R. 1957. A Markovian decision process. *Journal of Mathematics and Mechanics* 6(5):679–684.

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

Breazeal, C. 2004. Social interactions in HRI: the robot view. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* 34(2):181–186.

Choi, S. P. M.; Yeung, D.-Y.; and Zhang, N. L. 1999. An Environment Model for Nonstationary Reinforcement Learning. *Neural Information Processing Systems* 987–993.

Da Silva, B. C.; Basso, E. W.; Bazzan, A. L.; and Engel, P. M. 2006. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learnig*, 217–224.

Elidrisi, M.; Johnson, N.; Gini, M.; and Crandall, J. W. 2014. Fast adaptive learning in repeated stochastic games by game abstraction. In *Proceedings of the 13th International Joint Conference on Autonomous Agents and Multiagent Systems*, 1141–1148.

Fernández, F., and Veloso, M. 2006. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 720–727. Hakodata, Hokkaido, Japan: ACM.

Fulda, N., and Ventura, D. 2006. Predicting and Preventing Coordination Problems in Cooperative Q-learning Systems. In *IJCAI-07: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 780–785.

Gittins, J. C. 1979. Bandit Processes and Dynamic Allocation Indices. *Journal of the Royal Statistical Society* 41(2):148–177.

Hernandez-Leal, P.; Munoz de Cote, E.; and Sucar, L. E. 2014a. A framework for learning and planning against switching strategies in repeated games. *Connection Science* 26(2):103–122.

Hernandez-Leal, P.; Munoz de Cote, E.; and Sucar, L. E. 2014b. Using a priori information for fast learning against non-stationary opponents. In *Advances in Artificial Intelligence – IBERAMIA 2014*, 536–547.

Littman, M. L., and Stone, P. 2001. Implicit Negotiation in Repeated Games. *ATAL '01: Revised Papers from the 8th International Workshop on Intelligent Agents VIII.*

Munoz de Cote, E.; Chapman, A. C.; Sykulski, A. M.; and Jennings, N. R. 2010. Automated Planning in Repeated Adversarial Games. *Uncertainty in Artificial Intelligence* 376–383.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of Markov decision processes. *Mathematics of Operations Research* 12(3):441–450.

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

Rosman, B.; Hawasly, M.; and Ramamoorthy, S. 2015. Bayesian Policy Reuse. *arXiv.org*.