Evolutionary Dynamics of Regret Minimization

Tomas Klos[1], Gerrit Jan van Ahee[2], and Karl Tuyls[3]

[1] Delft University of Technology, Delft, The Netherlands
[2] Yes2web, Rotterdam, The Netherlands
[3] Maastricht University, Maastricht, The Netherlands

Abstract. Learning in multi-agent systems (MAS) is a complex task. Current learning theory for single-agent systems does not extend to multi-agent problems. In a MAS the reinforcement an agent receives may depend on the actions taken by the other agents present in the system. Hence, the Markov property no longer holds and convergence guarantees are lost. Currently there does not exist a general formal theory describing and elucidating the conditions under which algorithms for multi-agent learning (MAL) are successful. Therefore it is important to fully understand the dynamics of multi-agent reinforcement learning, and to be able to analyze learning behavior in terms of stability and resilience of equilibria. Recent work has considered the replicator dynamics of evolutionary game theory for this purpose. In this paper we contribute to this framework. More precisely, we formally derive the evolutionary dynamics of the Regret Minimization polynomial weights learning algorithm, which will be described by a system of differential equations. Using these equations we can easily investigate parameter settings and analyze the dynamics of multiple concurrently learning agents using regret minimization. In this way it is clear why certain attractors are stable and potentially preferred over others, and what the basins of attraction look like. Furthermore, we experimentally show that the dynamics predict the real learning behavior and we test the dynamics also in non-self play, comparing the polynomial weights algorithm against the previously derived dynamics of Q-learning and various Linear Reward algorithms in a set of benchmark normal form games.

1 Introduction

Multi-agent systems (MAS) are a proven solution method for contemporary technological challenges of a distributed nature, such as e.g. load balancing and routing in networks [13,14]. Typical for these new challenges of today is that the environment in which those systems need to operate is dynamic, rather than static, and as such evolves over time, not only due to external environmental changes but also due to agents' interactions. The naive approach of providing all possible situations an agent can encounter along with the optimal behavior in each of them beforehand, is not feasible in this type of system. Therefore to successfully apply MAS, agents should be able to adapt themselves in response to actions of other agents and changes in the environment. For this purpose, researchers have investigated Reinforcement Learning (RL) [15,11].

J.L. Balcázar et al. (Eds.): ECML PKDD 2010, Part II, LNAI 6322, pp. 82–96, 2010.

RL is already an established and profound theoretical framework for learning in a single-agent framework. In this framework a single agent operates in an uncertain environment and must learn to act autonomously and achieve a certain goal. Under these circumstances it has been shown that as long as the environment an agent is experiencing is Markovian, and the agent can try out sufficiently many actions, RL guarantees convergence to the optimal strategy [22,16,12]. This task becomes more complex when multiple agents are concurrently learning and possibly interacting with one another. Furthermore, these agents have potentially different capabilities and goals. Consequently, learning in MAS does not guarantee the same theoretical grounding.

Recently an evolutionary game theoretic approach has been introduced to provide such a theoretical means to analyze the dynamics of multiple concurrently learning agents [19,17,18]. For a number of state of the art MAL algorithms, such as Q-learning and Learning Automata, the evolutionary dynamics have been derived. Using these derived dynamics one can visualize and thoroughly analyze the average learning behavior of the agents and stability of the attractors. For an important class of MAL algorithms, viz. Regret Minimization (RM), these dynamics are still unknown. The central idea of this type of algorithm is that after the agent has taken an action and received a reward in the learning process, he may look back at the history of actions and rewards taken so far, and regret not having played another action—namely the best action in hindsight. Based on this idea a loss function is calculated which is key to the update rule of an RM learning algorithm. To contribute to this EGT backbone for MAL, it is essential that we derive and examine the evolutionary dynamics of Regret Minimization as well, which we undertake in the present paper.

In this paper we follow this recent line of reasoning that captures the dynamics of MAL algorithms and formally derive the evolutionary dynamics of the Polynomial Weights Regret Minimization learning algorithm. Furthermore, we perform an extensive experimental study using these dynamics, illustrating how they predict the behavior of the associated learning algorithm. As such this allows for a quick and thorough analysis of the behavior of the learning agents in terms of learning traces, parameters, and stability and resilience of attractors. The derived dynamics provide theoretical insight in this class of algorithms and as such contribute to a theoretical backbone for MAL. Moreover, we do not only investigate the dynamics in self play but also compare the derived dynamics against the dynamics of Linear Reward-Inaction and Linear Reward-ϵPenalty Learning Automata. It is the first time that these MAL algorithms are compared using their derived dynamical systems instead of performing a time consuming experimental study with the learning algorithms themselves.

The remainder of this paper is structured as follows. In Sect. 2 we introduce the necessary background for the remainder of the paper. More precisely, we introduce Regret Minimization and the Replicator Dynamics of Evolutionary Game Theory. In Sect. 3 we formally derive the dynamics of RM, and we study them experimentally in Sect. 4. Section 5 summarizes related work and we conclude in Sect. 6.

2 Preliminaries

In this section we describe the necessary background for the remainder of the article. We start off by introducing Regret Minimization, the multi-agent learning algorithm of which we want to describe the evolutionary dynamics. Section 2.2 introduces the replicator dynamics of Evolutionary Game Theory.

2.1 Regret Minimization

Regret Minimizing algorithms are learning algorithms relating the history of an agents' play to his current choice of action. After acting, the agent looks back at the history of actions and corresponding rewards, and regrets not having played the best action in hindsight. Playing this action at all stages often results in a better total reward by removing the cost of exploration.

As keeping a history of actions and rewards is very expensive at best, most regret minimizing algorithms use the concept of *loss* l_i to aggregate the history per action i. Using the loss, the action selection probabilities are updated. Several algorithms have been constructed around computing loss.

In order to determine the best action in hindsight the agent needs to know what rewards he could have received, which could be provided by the system. Each action i played, results in a reward r_i and the best reward in hindsight r is determined, with the loss for playing i given by $l_i = r - r_i$: a measure for regret.

The Polynomial Weights algorithm [2] is a member of the Regret Minimization class. It assigns a weight w_i to each action i which is updated using the loss for not playing the best action in hindsight:

$$w_i^{(t+1)} = w_i^{(t)} \left(1 - \lambda l_i^{(t)}\right) \quad , \tag{1}$$

where λ is a learning parameter to control the speed of the weight-change. The weights are now used to derive action selection probabilities by normalization:

$$x_i^{(t)} = \frac{w_i^{(t)}}{\sum_j w_j^{(t)}} \quad . \tag{2}$$

2.2 Replicator Dynamics

The Replicator Dynamics (RD) are a system of differential equations describing how a population of strategies evolves through time [9]. Here we will consider an individual level of analogy between the related concepts of learning and evolution. Each agent has a set of possible strategies at hand. Which strategies are favored over others depends on the experience the agent has previously gathered by interacting with the environment and other agents. The collection of possible strategies can be interpreted as a population in an evolutionary game theory perspective [20]. The dynamical change of preferences within the set of strategies can be seen as the evolution of this population as described by the replicator

dynamics. The continuous time two-population replicator dynamics are defined by the following system of ordinary differential equations:

$$\begin{aligned} \dot{x}_i &= x_i \left((A\mathbf{y})_i - \mathbf{x}^T A \mathbf{y} \right) \\ \dot{y}_i &= y_i \left((B\mathbf{x})_i - \mathbf{y}^T B \mathbf{x} \right) \end{aligned} \quad , \tag{3}$$

where A and B are the payoff matrices for player 1 (population x) and 2 (population y) respectively. For an example see Sect. 4. The probability vector $\mathbf{x}$ (resp. $\mathbf{y}$) describes the frequency of all pure strategies (also called replicators) for player 1 (resp. 2). Success of a replicator i in population x is measured by the difference between its current payoff $(A\mathbf{y})_i$ and the average payoff of the entire population x, i.e. $\mathbf{x}^T A \mathbf{y}$.

3 Modelling Regret Minimization

In this section we will derive a mathematical model, i.e. a system of differential equations, describing the dynamics of the polynomial no regret learning algorithm. Each learning agent will have his own system of differential equations describing the updates to his action selection probabilities. Just as in (3), our models will be using expected rewards to calculate the change in action selection probabilities. Here too, these rewards are determined by the other agents in the system.

The first step in finding a mathematical model for Polynomial Weights is to determine the update $\delta x_i^{(t)}$ at time t to the action selection probability for any action i:

$$\begin{aligned} \delta x_i^{(t)} &= x_i^{(t+1)} - x_i^{(t)} \\ &= \frac{w_i^{(t+1)}}{\sum_j w_j^{(t+1)}} - x_i^{(t)} \quad . \end{aligned}$$

This shows that the update δx_i to the action selection probability depends on the weight as well as the probabilties. If we want the model to consist of a coupled system of differential equations, we need to find an expression for these weights in terms of x_i and y_i. In other words we would like to find an expression of the weights in terms of their corresponding action selection probabilities. Therefore, using (2) we divide any two x_i and x_j:

$$\begin{aligned} \frac{x_i}{x_j} &= \frac{w_i}{\sum_k w_k} \frac{\sum_k w_k}{w_j} \\ w_i &= x_i \frac{w_j}{x_j} \quad . \end{aligned} \tag{4}$$

This allows to represent weights as the corresponding action selection probability multiplied by a common factor. Substituting (4) into (1) and subsequently (2) yields:

$$x_i^{(t+1)} = \frac{\frac{w_j^{(t)}}{x_j^{(t)}} x_i^{(t)} \left(1 - \lambda l_i^{(t)}\right)}{\sum_k \frac{w_j^{(t)}}{x_j^{(t)}} x_k^{(t)} \left(1 - \lambda l_k^{(t)}\right)} = \frac{x_i^{(t)} \left(1 - \lambda l_i^{(t)}\right)}{\sum_k x_k^{(t)} \left(1 - \lambda l_k^{(t)}\right)} \ . \tag{5}$$

The update $\delta x_i^{(t)}$ is found by subtracting $x_i^{(t)}$ from (5):

$$\begin{aligned} \delta x_i^{(t)} &= x_i^{(t+1)} - x_i^{(t)} \\ &= \frac{x_i^{(t)} \left(1 - \lambda l_i^{(t)}\right)}{\sum_j x_j^{(t)} \left(1 - \lambda l_j^{(t)}\right)} - x_i^{(t)} \\ &= \frac{x_i^{(t)} \left(\left(1 - \lambda l_i^{(t)}\right) - \sum_j x_j^{(t)} \left(1 - \lambda l_j^{(t)}\right)\right)}{\sum_j x_j^{(t)} \left(1 - \lambda l_j^{(t)}\right)} \ . \end{aligned} \tag{6}$$

In subsequent formulations, the reference to time will again be dropped, as all expressions reference the same time t.

The next step in the derivation requires the specification of the loss l_i. The best reward may be modeled as the maximum expected reward $r = \max_k (A\mathbf{y})_k$, the actual expected reward is given by $r_i = (A\mathbf{y})_i$. This yields the equation for the loss for action i:

$$l_i = \max_k (A\mathbf{y})_k - (A\mathbf{y})_i \ . \tag{7}$$

After substituting the loss l_i (7) into (6), the derivation of the model is nearly finished. Using the fact that $\sum_j x_j C = C$ for constant C, we may simplify the resulting equation by replacing these terms:

$$\begin{aligned} \mathbb{E}[\delta x_i](\mathbf{x}, \mathbf{y}) &= \frac{x_i \left(1 - \lambda(\max_k (A\mathbf{y})_k - (A\mathbf{y})_i) - \sum_j x_j \left(1 - \lambda(\max_k (A\mathbf{y})_k - (A\mathbf{y})_j)\right)\right)}{\sum_j x_j \left(1 - \lambda(\max_k (A\mathbf{y})_k - (A\mathbf{y})_j)\right)} \\ &= \frac{x_i \left(1 - \lambda \max_k (A\mathbf{y})_k + \lambda (A\mathbf{y})_i - \sum_j x_j + \lambda \sum_j x_j \max_k (A\mathbf{y})_k - \lambda \sum_j x_j (A\mathbf{y})_j\right)}{\sum_j x_j - \lambda \left(\sum_j x_j \max_k (A\mathbf{y})_k - \sum_j x_j (A\mathbf{y})_j\right)} \\ &= \frac{x_i \lambda \left((A\mathbf{y})_i - \sum_j x_j (A\mathbf{y})_j\right)}{1 - \lambda \left(\max_k (A\mathbf{y})_k - \sum_j x_j (A\mathbf{y})_j\right)} \ . \end{aligned} \tag{8}$$

Finally, we recognize $\sum_j x_j (A\mathbf{y})_j = \mathbf{x}^T A\mathbf{y}$ and we arrive at the general model:

$$\dot{x} = \frac{\lambda x_i \left((A\mathbf{y})_i - \mathbf{x}^T A\mathbf{y}\right)}{1 - \lambda \left(\max_k (A\mathbf{y})_k - \mathbf{x}^T A\mathbf{y}\right)} \ . \tag{9}$$

The derivation for $\dot{y}$ is completely analogous and yields:

$$\dot{y} = \frac{\lambda y_i \left((B\mathbf{x})_i - \mathbf{y}^T B\mathbf{x} \right)}{1 - \lambda \left(\max_k (B\mathbf{x})_k - \mathbf{y}^T B\mathbf{x} \right)} \quad . \tag{10}$$

Equations 9 and 10 describe the dynamics of the Polynomial Weights learning algorithm. What is immediately interesting to note is that we recognize in this model the coupled replicator equations, described in (3), in the numerator. This value is then weighted based on the expected loss. At this point we can conclude that this learning algorithm can also be described based on the coupled RD from evolutionary game theory, just as has been shown before for Learning Automata and Q-learning (resulting in different equations that also contain the RD) [17].

4 Experiments

We performed numerical experiments to validate our model, by comparing its predictions with simulations of agents using the PW learning algorithm. In addition, we propose a method to investigate outcomes of interactions among agents using *different* learning algorithms, of which dynamics have been derived. First we present the games and algorithms we used in our experiments.

4.1 Sample Games

We limit ourselves to two-player, two-action, single state games. This class includes many interesting games, such as the Prisoner's Dilemma, and allows us to visualize the learning dynamics by plotting them in 2-dimensional trajectory fields. These plots show the direction of change for the two players' action selection probabilities.

Having two agents with two actions each yields games with action selection probabilities $\mathbf{x} = [x_1 x_2]^T$ and $\mathbf{y} = [y_1 y_2]^T$ and two 2-dimensional payoff matrices A and B for players 1 and 2 respectively:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \quad .$$

Note that these are payoff *matrices*, not payoff *tables* with row players and column players: put in these terms, each player is the *row* player in his own matrix.

This class of games can be partitioned into three subclasses [20]. We experimented with games from all three subclasses. The subclasses are the following.

1. At least one of the players has a dominant strategy when

$$(a_{11} - a_{21})(a_{12} - a_{22}) > 0 \text{ or}$$
$$(b_{11} - b_{21})(b_{12} - b_{22}) > 0 \quad .$$

The Prisoner's Dilemma (PD) falls into this class. The reward matrices used for this class in the simulations and model are

$$A = B = \begin{bmatrix} 1 & 5 \\ 0 & 3 \end{bmatrix}.$$

This game has a single pure Nash equilibrium at $(\mathbf{x}, \mathbf{y}) = ([1,0]^T, [1,0]^T)$.

2. There are two pure equilibria and one mixed when

$$\begin{aligned} (a_{11} - a_{21})(a_{12} - a_{22}) &< 0 \text{ and} \\ (b_{11} - b_{21})(b_{12} - b_{22}) &< 0 \text{ and} \\ (a_{11} - a_{21})(b_{11} - b_{21}) &> 0 \ . \end{aligned}$$

The Battle of the Sexes (BoS) falls into this class. The reward matrices used for this class in the simulations and model are

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \ B = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}.$$

This game has two pure Nash equilibria at $(\mathbf{x}, \mathbf{y}) = ([0,1]^T, [0,1]^T)$ and $([1,0]^T, [1,0]^T)$ and one mixed Nash equilibrium at $([2/3, 1/3]^T, [1/3, 2/3]^T)$.

3. There is just one mixed equilibrium when

$$\begin{aligned} (a_{11} - a_{21})(a_{12} - a_{22}) &< 0 \text{ and} \\ (b_{11} - b_{21})(b_{12} - b_{22}) &< 0 \text{ and} \\ (a_{11} - a_{21})(b_{11} - b_{21}) &< 0. \end{aligned}$$

This class contains Matching Pennies (MP). The reward matrices used for this class in the simulations and model are

$$A = B = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

This game has a single mixed Nash equilibrium at $\mathbf{x} = [1/2, 1/2]^T, \mathbf{y} = [1/2, 1/2]^T$.

4.2 Other Learning Algorithms

The dynamics we derived to model agents using the Polynomial Weights (PW) algorithm can be used to investigate the performance of the PW algorithm in selfplay. This provides an important test for the validity of a learning algorithm: in selfplay it should converge to a Nash equilibrium of the game [6]. Crucially, with our model, we are now also able to model and make predictions about interactions between agents using *different* learning algorithms.

In related work, analytical models for several other algorithms have been derived (see [19,17,7] and Table 1 for an overview). Here we use models for the Linear Reward-Inaction (L_{R-I}) and Linear Reward-ϵPenalty ($L_{R-\epsilon P}$) policy iteration algorithms.

L_{R-I} We study 2 algorithms from the class of linear reward algorithms. These are algorithms that update (increase or decrease) the action selection probability for the action selected by a fraction $0 < \lambda \leq 1$ of the payoff received. The parameter λ is called the *learning rate* of the algorithm. The L_{R-I} algorithm rewards, but does not punish actions for yielding low payoffs: the action selection probability of the selected action is increased whenever rewards $0 \leq r \leq 1$ are received.

$L_{R-\epsilon P}$ The $L_{R-\epsilon P}$ algorithm generalizes both the L_{R-I} algorithm and the L_{R-P} algorithm (which we therefore didn't include). In addition to rewarding high payoffs, the penalty algorithms also punish low payoffs. The $L_{R-\epsilon P}$ algorithm captures both other algorithms through the parameter $\epsilon > 0$ which specifies how severely low rewards are punished: as ϵ goes to 0 (1), there is no (full) punishment and the algorithm behaves like L_{R-I} (L_{R-P}).

The models for these algorithms are represented in Table 1, which also shows how they are all variations on the basic coupled replicator equations.

Table 1. Correspondence between Coupled Replicator Dynamics (CRD) and learning strategies. (Q refers to the Q-learning algorithm.)

Alg. name	Model	Reference
CRD	$x_i\left((A\mathbf{y})_i - \mathbf{x}^T A\mathbf{y}\right)$	[9]
L_{R-I}	$\lambda x_i\left((A\mathbf{y})_i - \mathbf{x}^T A\mathbf{y}\right)$	[17]
$L_{R\text{-}\epsilon P}$	$\lambda x_i\left((A\mathbf{y})_i - \mathbf{x}^T A\mathbf{y}\right) - \epsilon\lambda\left(-x^2(1-(A\mathbf{y})_i) + \frac{1-x_i}{r-1}\sum_{j\neq i} x_j(1-(A\mathbf{y})_j)\right)$	[1]
PW	$\lambda x_i\left((A\mathbf{y})_i - \mathbf{x}^T A\mathbf{y}\right) / \left(1-\lambda\left(\max_k (A\mathbf{y})_k - \mathbf{x}^T A\mathbf{y}\right)\right)$	Sec. 3
Q	$\tau\lambda x_i\left((A\mathbf{y})_i - \mathbf{x}^T A\mathbf{y}\right) + \lambda x_i \sum_j x_j \ln\left(\frac{x_j}{x_i}\right)$	[19,7]

4.3 Results

To visualize learning, all of our plots show the action selection probability for the *first* actions x_1 and y_1 of the two agents on the two axes: the PW agent on the x-axis and the other agent on the y-axis (sometimes this other agent also uses the PW algoritm). Knowing these probabilities, we also know $x_2 = 1 - x_1$ and $y_2 = 1 - y_1$. When playing the repeated game, the learning strategy updates these probabilities after each iteration.

All learning algorithms have been simulated extensively in each of the above games. This has been done in order to validate the models we have derived or taken from the literature. The results show paths starting at the initial action selection probabilities for action 1 for both agents that, as learning progresses, move toward some equilibrium. The simulations depend on: (*i*) the algorithms and their parameters, (*ii*) the game played and (*iii*) the initial action selection probabilities. We simulate and model 3 different algorithms (see Sect 4.2: PW,

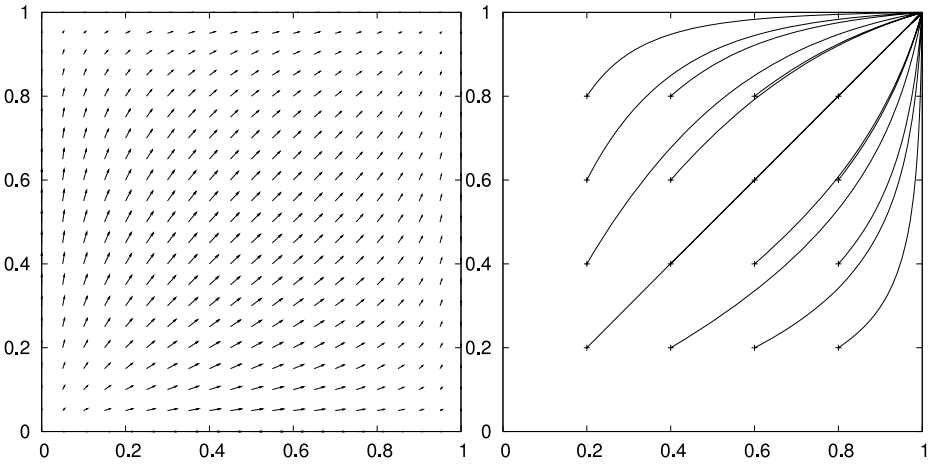

Fig. 1. PW selfplay, Prisoner's Dilemma

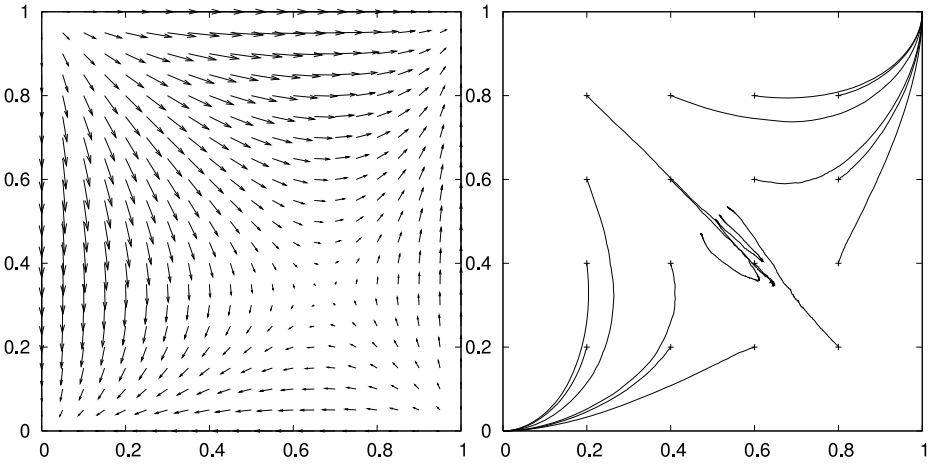

Fig. 2. PW selfplay, Battle of the Sexes

L_{R-I}, and $L_{R-\epsilon P}$), with $\lambda = \epsilon = 1$, in 3 different games (see Sect. 4.1: PD, BoS, and MP). The initial probabilities are taken from the grid $\{.2, .4, .6, .8\} \times \{.2, .4, .6, .8\}$; they are indicated by '+' in the plots. All figures show vector fields on the *left*, and average trajectories over 500 simulations of 1500 iterations of agents using the various learning algorithms on the *right*.

PW Selfplay. In Figures 1, 2, and 3 we show results for the PW algorithm in selfplay in the PD, the BoS, and MP, respectively. In all three figures, the models clearly show direction of motion towards each of the various Nash equilibria in

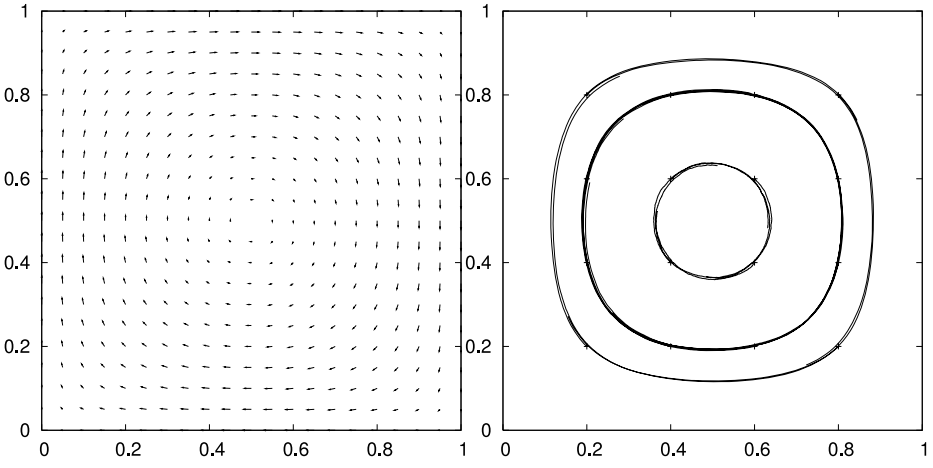

Fig. 3. PW selfplay, Matching Pennies

the respective games: the single pure strategy Nash equilibrium in the PD, the two pure strategy Nash equilibria in the BoS (not the unstable mixed one), and a circular oscillating pattern in the MP game.

The simulation paths (on the right) validate these models (on the left), in that the models are shown to accurately predict the simulation trajectories of interacting PW agents in all three games. The individual simulations in the BoS game (Fig. 2) that start from any one of the 4 initial positions on the diagonal (which is exactly the boundary between the two pure equilibria's basins of attraction) all end up in one of the two pure strategy Nash equilibria, but since we take the average of all 500 simulations, these plots end up in the center, somewhat spread out over the perpendicular (0,0)-(1,1) diagonal, because there is not a perfect 50%/50% division of the trajectories over the 2 Nash equilibria.

PW vs. L_{R-I}. Having established the external validity of our model of PW agents, we now turn to an analysis of interactions of PW agents and agents using other learning algorithms. To this end, in all subsequent figures, we propose to let movement in the x-direction be controlled by the PW dynamics and in the y-direction by the dynamics of one of the other models (see Table 1 and Sect. 4.2). We start with L_{R-I} agents, for which we only show interactions with PW agents in the PD (Fig. 4) and the BoS (Fig. 5). (The vector fields and trajectories in the MP game correspond closely again, and don't differ much from those in Fig. 3.)

This setting already shows that when agents use different learning algorithms, the interaction changes significantly. The direction of motion is still towards the single pure strategy Nash equilibrium in the PD and towards the two pure strategy Nash equilibria in the BoS, albeit along different lines. Again, the simulation paths follow the vector field closely, where the two seemingly anomalous average trajectories starting from $(0.2, 0.6)$ and from $(0.6, 0.4)$ in the BoS game (Fig. 5) can be explained in a similar manner as for Fig. 2. In this setting of PW vs.

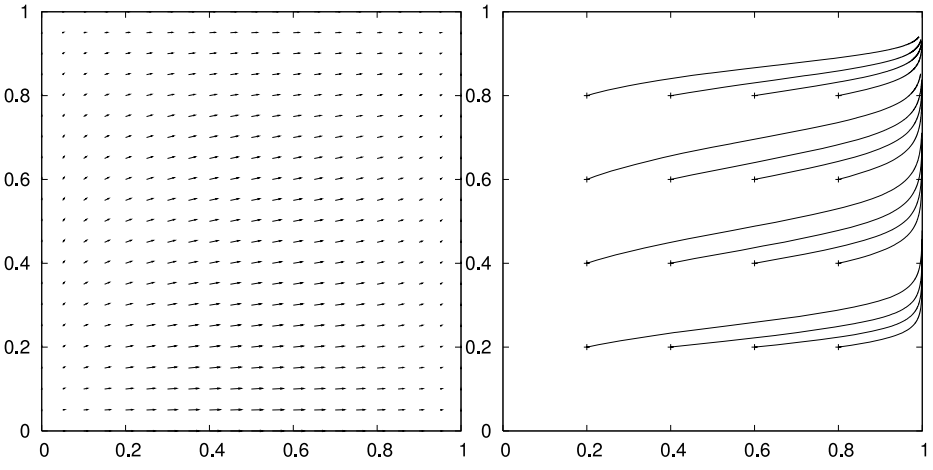

Fig. 4. PW vs. L_{R-I}, Prisoner's Dilemma

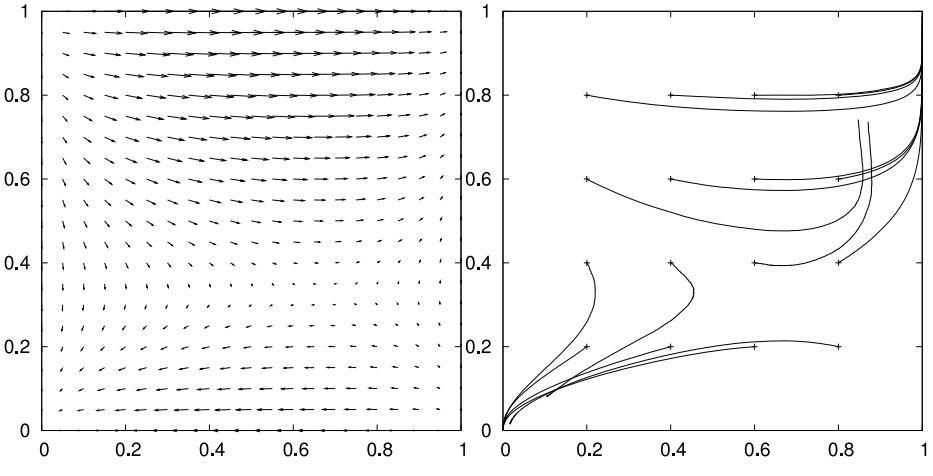

Fig. 5. PW vs. L_{R-I}, Battle of the Sexes

L_{R-I}, these points are now the initial points closest to the border between the basins of attraction of the two pure strategy equilibria, although they are not *on* the border, as the 4 points in Fig. 2 were, which is why these average trajectories end up closer to the equilibrium in whose basin of attraction they started. However, they are close enough to the border with the other basin to let stochasticity in the algorithm take some of the individual runs to the 'wrong' equilibrium.

An important observation we can make based on this novel kind of non-selfplay analysis, is that when agents use different learning algorithms, the outcomes of the game, or at least the trajectories agents may be expected to follow in reaching

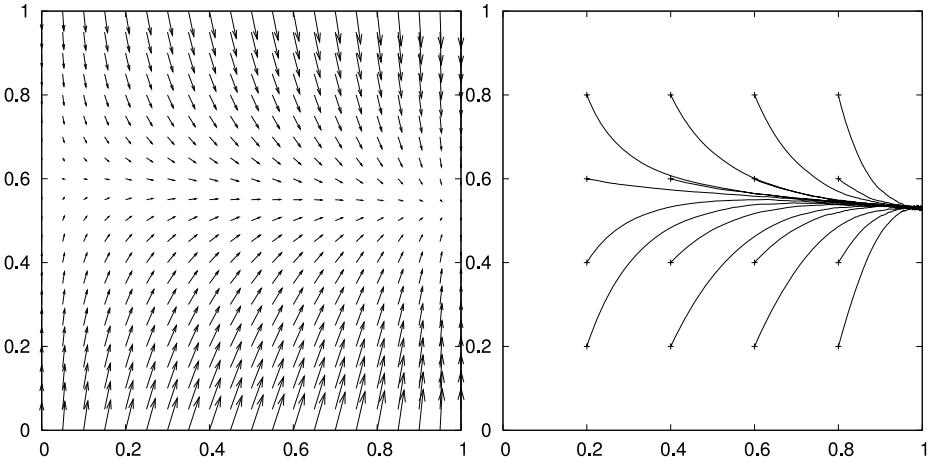

Fig. 6. PW vs. $L_{R-\epsilon P}$, Prisoner's Dilemma

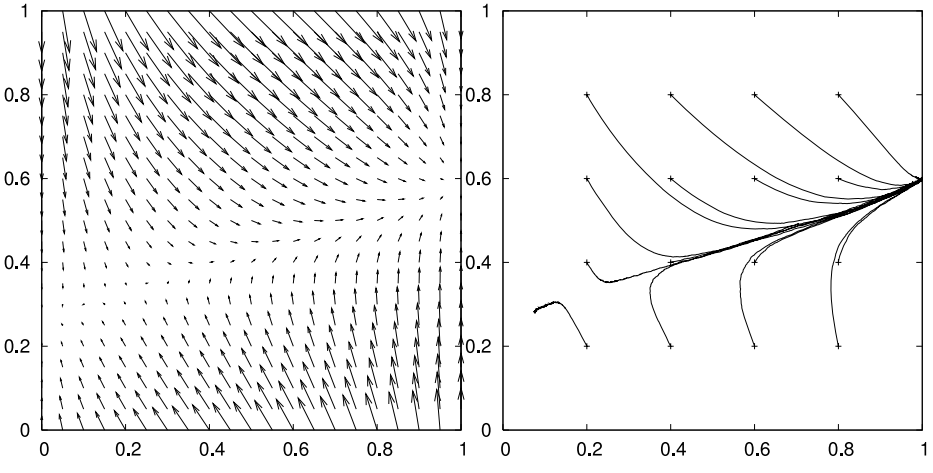

Fig. 7. PW vs. $L_{R-\epsilon P}$, Battle of the Sexes

those outcomes, as well as the basins of attraction of the various equilibria, change as a consequence. This gives insight into the outcomes we may expect from interactions among agents using different learning algorithms.

PW vs. $L_{R-\epsilon P}$. For this interaction, we again show plots for all games (Figures 6–8). The interaction is now changed not just quantitatively, but qualitatively as well. Clearly, the various Nash equilibria are not in reach of the interacting learners anymore: all these games, when played between one agent using the PW algorithm, and one agent using the $L_{R-\epsilon P}$ algorithm, have different equilibria than Nash.

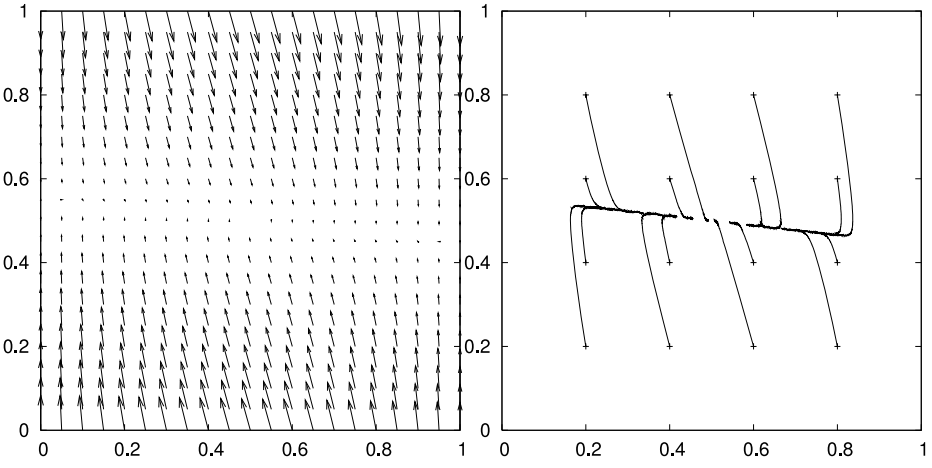

Fig. 8. PW vs. $L_{R-\epsilon P}$, Matching Pennies

What is particularly interesting to observe in Figures 6 and 7 is that the PW player is again playing the strategy prescribed by Nash while the $L_{R-\epsilon P}$ player randomizes over both strategies. (It is hard to tell just by visual inspection whether in Fig. 7 this leads to just a single equilibrium outcome on the right hand side, or whether there is another one on the left. This may be expected given the average trajectory starting at $(0.2, 0.2)$, but should be analyzed more thoroughly, for example using the Amoeba tool [21].) This implies that while the $L_{R-\epsilon P}$ player keeps on exploring its possible strategies the PW player already found the Nash strategy and consequently is regularly able to exploit the other player by always defecting when the $L_{R-\epsilon P}$ occasionally cooperates. Therefore the PW learner will receive, on average, more reward than when playing in self play for instance. While it can be seen from Fig. 4 that the L_{R-I} learner also evolves towards more or less the same point at the right vertical axis as the $L_{R-\epsilon P}$ against the PW learner, it can be observed that the L_{R-I} learner is still able to recover from his mixed strategy and is eventually able to find the Nash strategy as well. Consequently the L_{R-I} performs better against the PW learner than the $L_{R-\epsilon P}$ learner.

In the BoS game (Fig. 7), the equilibria don't just shift, but there even appears to be just a single equilibrium in the game played between a PW player and an $L_{R-\epsilon P}$ player, rather than 3 equilibria, as in the game played by rational agents. In the MP game (Fig. 8), the agents now converge to the mixed strategy equilibrium of the game: the PW player quickly, and the $L_{R-\epsilon P}$ player only after the PW agent has closely approached it's equilibrium strategy, much like in the case of the PW and the L_{R-I} players in the PD in Fig. 4.

5 Related Work

Modelling the learning dynamics of MAS using an evolutionary game theory approach has recently received quite some attention. In [3] Börgers and Sarin

proved that the continuous time limit of Cross Learning converges to the most basic replicator dynamics model considering only selection. This work has been extended to Q-learning and Learning Automata in [19,17] and to multiple state problems in [8]. Based on these results the dynamcs of ϵ-greedy Q-learning have been derived in [7].

Other approaches investigating the dynamics of MAL have also been considered in [5,4,10]. For a survey on Multi-agent learning we refer to [13].

6 Conclusions

We have derived an analytical model describing the dynamics of a learning agent using the Polynomial Weights Regret Minimization algorithm. It is interesting to observe that the model for PW is connected to the Coupled Replicator Dynamics of evolutionary game theory, like other learning algorithms, e.g. Q-learning and L_{R-I}.

We use the newly derived model to describe agents in selfplay in the Prisoner's Dilemma, the Battle of the Sexes, and Matching Pennies. In extensive experiments, we have shown the validity of the model: the modeled behavior shows good resemblance with observations from simulation.

Moreover, this work has shown a way of modeling agent interactions when both agents use *different* learning algorithms. Combining two models in a single game provides much insight into the way the game may be played, as shown in Section 4.3. In this way it is not necessary to run time-consuming experiments to analyze the behavior of different algorithms against each other, but this can be analyzed directly by investigating the involved dynamical systems. We have analyzed the effect on the outcomes in several games when the agents use different combinations of learning algorithms, finding that the games change profoundly. This has significant implications for the analysis of Multi-agent systems, for which we believe our paper provides valuable tools.

In future work, we plan to extend our analysis to other games and learning algorithms and will perform an in-depth analysis of the differences in the mathematical models of the variety of learning algorithms connected to the replicator dynamics. We also plan to systematically analyze the outcomes of interactions among various learning algorithms in different games, by studying the equilibria that arise and their basins of attraction. Also, we need to investigate the sensitivity of the outcomes to changes in the algorithms' parameters.

References

1. Van Ahee, G.J.: Models for Multi-Agent Learning. Master's thesis, Delft University of Technology (2009)
2. Blum, A., Mansour, Y.: Learning, regret minimization and equilibria. In: Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)
3. Börgers, T., Sarin, R.: Learning through reinforcement and replicator dynamics. J. Economic Theory 77 (1997)

4. Bowling, M.: Convergence problems of general-sum multiagent reinforcement learning. In: ICML (2000)
5. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: AAAI (1998)
6. Conitzer, V., Sandholm, T.: AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. Machine Learning 67 (2007)
7. Gomes, E.R., Kowalczyk, R.: Dynamic analysis of multiagent Q-learning with epsilon-greedy exploration. In: ICML (2009)
8. Hennes, D., Tuyls, K.: State-coupled replicator dynamics. In: AAMAS (2009)
9. Hofbauer, J., Sigmund, K.: Evolutionary Games and Population Dynamics. Cambridge University Press, Cambridge (1998)
10. Hu, J., Wellman, M.P.: Multiagent reinforcement learning: Theoretical framework and an algorithm. In: ICML (1998)
11. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. J. Artificial Intelligence Research 4 (1996)
12. Narendra, K., Thathachar, M.: Learning Automata: An Introduction. Prentice-Hall, Englewood Cliffs (1989)
13. Panait, L., Luke, S.: Cooperative multi-agent learning: The state of the art. J. AAMAS 11 (2005)
14. Shoham, Y., Powers, R., Grenager, T.: If multi-agent learning is the answer, what is the question? Artificial Intelligence 171 (2007)
15. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
16. Tsitsiklis, J.: Asynchronous stochastic approximation and Q-learning. Tech. rep., LIDS Research Center, MIT (1993)
17. Tuyls, K., 't Hoen, P.J., Vanschoenwinkel, B.: An evolutionary dynamical analysis of multi-agent learning in iterated games. J. AAMAS 12 (2006)
18. Tuyls, K., Parsons, S.: What evolutionary game theory tells us about multiagent learning. Artificial Intelligence 171 (2007)
19. Tuyls, K., Verbeeck, K., Lenaerts, T.: A selection-mutation model for Q-learning in multi-agent systems. In: AAMAS (2003)
20. Vega-Redondo, F.: Game Theory and Economics. Cambridge University Press, Cambridge (2001)
21. Walsh, W.E., Das, R., Tesauro, G., Kephart, J.O.: Analyzing complex strategic interactions in multi-agent systems. In: Workshop on Game-Theoretic and Decision-Theoretic Agents (2002)
22. Watkins, C., Dayan, P.: Q-learning. Machine Learning 8 (1992)