Multiagent Learning Using a Variable Learning Rate

Michael Bowling, Manuela Veloso

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213-3890

Abstract

Learning to act in a multiagent environment is a difficult problem since the normal definition of an optimal policy no longer applies. The optimal policy at any moment depends on the policies of the other agents. This creates a situation of learning a moving target. Previous learning algorithms have one of two shortcomings depending on their approach. They either converge to a policy that may not be optimal against the specific opponents' policies, or they may not converge at all. In this article we examine this learning problem in the framework of stochastic games. We look at a number of previous learning algorithms showing how they fail at one of the above criteria. We then contribute a new reinforcement learning technique using a variable learning rate to overcome these shortcomings. Specifically, we introduce the WoLF principle, "Win or Learn Fast," for varying the learning rate. We examine this technique theoretically, proving convergence in self-play on a restricted class of iterated matrix games. We also present empirical results on a variety of more general stochastic games, in situations of self-play and otherwise, demonstrating the wide applicability of this method.

Key words: Multiagent learning, reinforcement learning, game theory

1 Introduction

Research in multiagent systems includes the investigation of algorithms that select actions for multiple agents coexisting in the same environment. Multiagent systems are becoming increasingly relevant within artificial intelligence, as software and robotic agents become more prevalent. Robotic soccer, disaster

Email addresses: mhb@cs.cmu.edu (Michael Bowling), mmv@cs.cmu.edu (Manuela Veloso).

mitigation and rescue, automated driving, and information and e-commerce agents are examples of challenging multiagent domains. As the automation trend continues, we need robust algorithms for coordinating multiple agents, and for effectively responding to other external agents.

Multiagent domains require determining a course of action for each agent just as in single-agent domains. Machine learning can be a powerful tool for finding a successful course of action and can greatly simplify the task of specifying appropriate behaviors for an agent. In particular, through learning an agent can discover and exploit the dynamics of the environment and also adapt to unforeseen difficulties in the task. These benefits have caused learning to be studied extensively for single-agent problems with a stationary environment. In multiagent environments, learning is both more important and more difficult, since the selection of actions must take place in the presence of other agents.

We consider multiagent domains in which agents are forced to interact with other agents that may have independent goals, assumptions, algorithms, and conventions. We are interested in approaches where the agents can learn and adapt to the other agents' behavior. Since we assume that the other agents also have the ability to adapt their behavior, we face a difficult learning problem with a moving target. The optimal course of action is changing as all the agents adapt. These external adapting agents violate the basic stationary assumption of traditional techniques for behavior learning. New techniques need to be considered to address the multiagent learning problem. Furthermore, multiagent learning has a strong connection to game theory, where players select actions to maximize payoffs in the presence of other payoff maximizing players.

A few efforts have contributed new approaches to the multiagent learning problem, successfully demonstrating algorithms that can learn "optimal" policies under specific assumptions. In this article, we overview some of these algorithms while providing a parallel between game theory and multiagent learning. The analysis of previous algorithms leads us to introduce two desirable properties for multiagent learning algorithms: *rationality* and *convergence*. Interestingly, we note that previous algorithms offer either one of these properties but not both.

In this article, we contribute a new learning technique: a *variable learning rate.* We introduce this concept and provide a specific principle to adjust the learning rate, namely the WoLF principle, standing for "Win or Learn Fast." We successfully develop and apply the WoLF principle within different learning approaches. Given the novelty of the WoLF principle, we face the challenge of determining whether a WoLF-based learning algorithm is rational and convergent according to our own introduced properties of multiagent learning algorithms. We show the rationality property and we contribute a theoretical proof of the convergence of WoLF gradient ascent in a restricted class of iterated matrix games. We then show empirical results suggesting convergence of an extended WoLF algorithm and compare its performance in a variety of game situations used previously by other learning algorithms.

The article is organized as follows. In Section 2 we describe the stochastic game framework as a description of the multiagent learning problem. We also examine how previous techniques have one of two crucial shortcomings. In Section 3 we describe the variable learning rate technique and the WoLF principle. We analyze it theoretically on a restricted class of games, proving it overcomes the shortcomings of previous algorithms. In Section 4 we describe a practical algorithm that extends this technique to the general class of stochastic games. Finally, in Section 5 we show results demonstrating the applicability and effectiveness of the algorithms we introduced.

2 Stochastic Games and Learning

In this section we give an overview of the stochastic game framework. We also introduce two desirable properties of multiagent learning algorithms: rationality and convergence. We then examine previous learning algorithms specifically comparing them with respect to these two properties.

2.1 Stochastic Game Framework

Before presenting the formal definition of a stochastic game we examine other related models. We begin by examining Markov Decision Processes — a single-agent, multiple state framework. We then examine matrix games — a multiple-agent, single state framework. Finally, we introduce the stochastic game framework, which can be seen as the merging of MDPs and matrix games. Due to this background of game theory and agent-based systems, we will use the terms agent and player interchangeably.

2.1.1 Markov Decision Processes

A *Markov decision process* (MDP) (1; 2) is a tuple, $(\mathcal{S}, \mathcal{A}, T, R)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, T is a transition function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, and R is a reward function $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. The transition function defines a probability distribution over next states as a function of the current state and the agent's action. The reward function defines the reward received when selecting an action from the given state. Solving MDPs consists of finding a policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, mapping states to actions so as to maximize discounted future reward with discount factor γ.

MDPs are the focus of much of the reinforcement learning (RL) work (3; 4). The crucial result that forms the basis for this work is the existence of a stationary and deterministic policy that is optimal. It is such a policy that is the target for RL algorithms.

2.1.2 Matrix Games

A *matrix game* or *strategic game* (5; 6) is a tuple $(n, \mathcal{A}_{1\ldots n}, R_{1\ldots n})$, where n is the number of players, $\mathcal{A}_i$ is the set of actions available to player i (and $\mathcal{A}$ is the joint action space $\mathcal{A}_1 \times \ldots \times \mathcal{A}_n$), and R_i is player i's payoff function $\mathcal{A} \rightarrow \mathbb{R}$. The players select actions from their available set and receive a payoff that depends on *all* the players' actions. These are often called matrix games, since the R_i functions can be written as n-dimensional matrices.

In matrix games players are finding strategies to maximize their payoff. A *pure strategy* selects some action deterministically. A *mixed strategy* selects actions according to a probability distribution over the available actions. A strategy for player i is denoted $\sigma_i \in PD(\mathcal{A}_i)$, i.e., a probability distribution over the set of actions available to that player. A pure strategy is one of these distributions that assigns some action a probability of one. We use the notation $\mathcal{A}_{-i}$ to refer to the set of joint actions of all players excluding player i, i.e., $\mathcal{A}_{-i} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_{i-1} \times \mathcal{A}_{i+1} \times \ldots \times \mathcal{A}_n$. And we use σ_{-i} to refer to a joint, possibly mixed, strategy for these players, i.e., $\sigma_{-i} \in PD(\mathcal{A}_{-i})$.

Example matrix games are shown in Table 1. Table 1 (a) shows the matrices for a simple two-player game called matching pennies. In this game each player may select either *Heads* or *Tails*. If the choices are the same, then Player 1 takes a dollar from Player 2. If they are different, then Player 1 gives a dollar to Player 2. The matrices R_1 and R_2 represent the payoffs for players 1 and 2, with the row and columns corresponding to the two actions *Heads* and *Tails*. Table 1 (b) shows the game Rock-Paper-Scissors. In this game players select an action and a winner is determined by the rules: *Paper* beats *Rock*, *Scissors* beats *Paper*, and *Rock* beats *Scissors*. The winner, if there is one, takes a dollar from the loser. Table 1 (c) shows a coordination game with two players, each with two actions. The players only receive a payoff when they select the same action, but the players have different preferences as to which actions they would prefer to agree on.

Unlike MDPs, it is difficult even to define what it means to "solve" a matrix game. A strategy can only be evaluated if the other players' strategies are known. This can be illustrated in the matching pennies game (Table 1 (a)). In this game, if Player 2 is going to play *Heads*, then Player 1's optimal strategy

$$R_1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad R_2 = -R_1$$

(a) Matching Pennies

$$R_1 = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix} \qquad R_2 = -R_1$$

(b) Rock-Paper-Scissors

$$R_1 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \qquad R_2 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

(c) Coordination Game

Table 1
Example matrix games. Games (a) and (b) are zero-sum games, and (c) is a general-sum game.

is to play *Heads*, but if Player 2 is going to play *Tails*, then Player 1's optimal strategy is to play *Tails*. So there is no optimal pure strategy independent of the opponent. Similarly, there is no opponent-independent mixed strategy that is optimal. What does exist is an opponent-*dependent* solution, or set of solutions. This is called a *best-response*.

Definition 1 *For a game, the* best-response function *for player* i, $\mathrm{BR}_i(\sigma_{-i})$, *is the set of all strategies that are optimal given the other player(s) play the joint strategy* σ_{-i}.

The major advancement that has driven much of the development of matrix games and game theory is the notion of a best-response equilibrium or *Nash equilibrium* (7).

Definition 2 *A* Nash equilibrium *is a collection of strategies for all players,* σ_i, *with*

$$\sigma_i \in \mathrm{BR}_i(\sigma_{-i}).$$

So, no player can do better by changing strategies given that the other players continue to follow the equilibrium strategy.

What makes the notion of equilibrium compelling is that all matrix games have a Nash equilibrium, although there may be more than one.

Types of Matrix Games. Matrix games can be usefully classified according to the structure of their payoff functions. Two common classes of games are *strictly collaborative* and *strictly competitive* games. In strictly collaborative games (or team games), all agents have the same payoff function, so an action in the best interest of one agent is in the best interest of all the agents.

In strictly competitive games, there are two agents, where one's payoff function is the negative of the other (i.e., $R_1 = -R_2$). The games in Table 1(a) and Table 1(b) are examples of such a game. Strictly competitive games are called *zero-sum games* since the payoff functions sum to zero or equivalently to some other constant. Other games, including strictly collaborative games, are called *general-sum games*. Table 1(c) is an example of such a game. One appealing feature of zero-sum games is that they contain a unique Nash equilibrium. [1] This equilibrium can be found as the solution to a relatively simple linear program. [2] Finding equilibria in general-sum games requires a more difficult quadratic programming solution (8).

2.1.3 Stochastic Games

We will now present the stochastic game framework, combining MDPs and matrix games. A *stochastic game* is a tuple $(n, \mathcal{S}, \mathcal{A}_{1\ldots n}, T, R_{1\ldots n})$, where n is the number of players, $\mathcal{S}$ is the set of states, $\mathcal{A}_i$ is the set of actions available to player i (and $\mathcal{A}$ is the joint action space $\mathcal{A}_1 \times \ldots \times \mathcal{A}_n$), T is the transition function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, and R_i is the reward function for the ith agent $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$. This looks very similar to the MDP framework except there are multiple players selecting actions and the next state and rewards depend on the joint action of those players. It's also important to notice that each player has its own separate reward function. We are interested in determining a course of action for a player in this environment. Specifically, we want to learn a stationary, though possibly stochastic, policy, $\pi : \mathcal{S} \times \mathcal{A}_i \to [0, 1]$, that maps states to a probability distribution over its actions. The goal is to find such a policy that maximizes the player's discounted future reward with discount factor γ.

A non-trivial result, proven by Shapley (9) for zero-sum games and by Fink (10) for general-sum games, is that there exist equilibria solutions for stochastic games just as they do for matrix games.

Stochastic games are a very natural extension of MDPs to multiple agents. They are also an extension of matrix games to multiple states. Each state in a stochastic game can be viewed as a matrix game with the payoff to player i of joint action a in state s determined by $R_i(s, a)$. After playing the matrix game and receiving the payoffs, the players are transitioned to another state (or matrix game) determined by their joint action. We can see that stochastic games then contain both MDPs ($n = 1$) and matrix games ($|\mathcal{S}| = 1$) as subsets of the framework. There is also a second connection between MDPs and stochastic games. If all but one player in a stochastic game play a fixed, though possibly stochastic, policy, then the problem for the remaining agent reverts back to an MDP. This is because fixing the other agents' policies,

[1] There can actually be multiple equilibria, but they all have equal payoffs and are interchangeable (5).

[2] The value of the equilibrium is the solution to the following linear program: $\max_{\sigma_1 \in PD(\mathcal{A}_1)} \min_{a_2 \in \mathcal{A}_2} \sum_{a_1 \in \mathcal{A}_1} R_1(\langle a_1, a_2 \rangle)\sigma_{a_2}$.

even if stochastic, makes the transitions Markovian, depending only on the remaining player's actions.

Types of Stochastic Games. The same classification for matrix games can be used with stochastic games. Strictly collaborative games, or team games, are ones where all the agents have the same reward function. Strictly competitive games, or zero-sum games, are two-player games where one player's reward is always the negative of the others'. Like matrix games, zero-sum stochastic games have a unique Nash equilibrium, although finding this equilibrium is no longer as trivial. Shapley presented along with his existence proof (9) a value iteration algorithm and others have also examined a variety of techniques (11; 12).

Learning in Stochastic Games. This article explores algorithms for an agent to learn a policy in stochastic games while other agents are learning simultaneously. This problem can be formalized as an on-line reinforcement learning problem, i.e., agents observe the current state and must select an action, which then affects the observed next state and reward. The algorithms we discuss and present vary, though, in what is observable about the other agents. As we will note, our theoretical results presented in Section 3 require the player's complete reward matrix to be known as well as observations of the other players' current stochastic policy. Algorithms described later in this Section require observation of only the other players' immediate actions, and additionally require observations of their immediate rewards. The novel algorithm we introduce in Section 4 requires the least information, neither needing to observe the other players' policies, actions, nor rewards.

One final concept of importance is the idea of *convergence* to a stationary policy.

Definition 3 *A learning algorithm for player i* converges to a stationary policy π *if and only if for any $\epsilon > 0$ there exists a time $T > 0$ such that,*

$$\forall t > T, a_i \in \mathcal{A}_i, s \in \mathcal{S} \quad P(s,t) > 0 \Rightarrow |P(a_i|s,t) - \pi(s,a_i)| < \epsilon,$$

where $P(s,t)$ is the probability that the game is in state s at time t, and $P(a_i|s,t)$ is the probability that the algorithm selects action a_i, given the game is in state s at time t.

2.2 Properties

In this section we present two properties that are desirable for multiagent learning algorithms. We will then examine how well previous algorithms have achieved these properties. Intuitively, the properties formalize the idea that a learner should learn a best-response when possible. Also, the learner should have some guarantee of convergence. We will now define these properties formally.

Property 1 *(Rationality) If the other players' policies converge to stationary policies then the learning algorithm will converge to a policy that is a best-response to the other players' policies.*

This is a fairly basic property requiring the learner to learn and play a best-response policy when the other players play stationary policies, in which case a best-response policy does indeed exist. As we observed earlier, when other players play stationary policies, the stochastic game simply becomes an MDP. So, this property requires that the algorithm finds an optimal policy to this MDP.

Property 2 *(Convergence) The learner will necessarily converge to a stationary policy. This property will usually be conditioned on the other agents using an algorithm from some class of learning algorithms.*

This convergence property requires that, against some class of other players' learning algorithms (ideally a class encompassing most "useful" algorithms), the learner's policy will converge. For example, one might refer to convergence with respect to players with stationary policies, or convergence with respect to rational players.

In this paper, we focus on convergence in the case of self-play. That is, if all the players use the same learning algorithm, do the players' policies converge? This is a crucial and difficult step towards convergence against more general classes of players. In addition, ignoring the possibility of self-play makes the naïve assumption that other players are in some way inferior since they cannot be using an identical algorithm.

Relationship to Equilibria. Although these properties do not explicitly relate to notions of equilibria, there is a strong relationship. If all the players use rational learning algorithms and their policies converge, they must have converged to an equilibrium. This can be seen by considering one of the rational players: since the other players converge to a stationary policy, it will converge to a best-response because it is rational. Since this is true for all the players, then the resulting policies must be an equilibrium (i.e., each player is playing a best response to the other players). So equilibria are fixed points of rational learning algorithms. In addition, if all the players use the same learning algorithm and it's rational and convergent in self-play, then the players are guaranteed to converge to a Nash equilibrium. This is because the convergent property guarantees convergence, and by the previous result, if they converge, it must be to an equilibrium.

Matrix Game Solver	+	MDP Solver	=	Stochastic Game Solver

MG + MDP =		Stochastic Game Solver Game Theory	Reinforcement Learning
LP	TD(0)	Shapley (9)	**Minimax-Q** (14)
LP	TD(1)	Pollatschek and Avi-Itzhak (11)	–
LP	TD(λ)	Van der Wal (11)	–
QP	TD(0)	–	Hu and Wellman (15)
FP	TD(0)	Fictitious Play (16; 11)	**Opponent-Modeling** (17)/- **JALs** (18)

LP/QP: linear/quadratic programming FP: fictitious play TD : temporal differencing

Table 2
Summary of algorithms to solve stochastic games. Each algorithm contains a matrix game solving component and an MDP solving component. "Game Theory" algorithms assume the transition and reward functions are known. "Reinforcement Learning" algorithms only receive observations of the transition and reward functions. The emphasized techniques are further described in Section 2.3.

2.3 Previous Algorithms

Stochastic games have been studied for many years and a number of algorithms have been proposed for "solving" these games. These algorithms have come both out of the game theory community and, more recently, the reinforcement learning community. Although the algorithms differ in their assumptions of what is known and what control can be exerted over the agents, the algorithms still have striking similarity (13). The algorithms consist of an MDP solving mechanism (e.g., some form of temporal differencing), and a matrix game solving mechanism (e.g., linear programming). Table 2 summarizes a number of these algorithms from both game theory and reinforcement learning, categorizing them by these components.

We will examine more closely three specific algorithms from the reinforcement learning community. In addition to presenting the algorithm, we will then examine it in light of the properties presented in Section 2.2. The first algorithm is probably the most commonly applied learning algorithm for multiagent systems. This is just the application of a single agent learning algorithm (e.g., Q-Learning), while completely ignoring the other agents. [3] We will also

[3] Single-agent learning is not included in Table 2 as it is not in a strict sense a stochastic game algorithm. It makes no attempt to address the other agents and therefore has no matrix game solving component.

examine two specifically multiagent learning algorithms: Minimax-Q and Opponent Modelling. These algorithms have been selected since they represent the basic approaches to learning in stochastic games. The algorithm by Hu and Wellman (15) is an extension of Minimax-Q to general-sum games. Their algorithm, though, requires some restrictive assumptions in order to be guaranteed to converge (15; 19). We will not examine the algorithm in this article since it has similar properties to Minimax-Q.

2.3.1 Q-Learning

Q-Learning (20) is a single-agent learning algorithm specifically designed to find optimal policies in MDPs. In spite of its original intent it has been widely used for multiagent learning, and not without success (21; 22; 18). It also has some theoretical merits. As observed, when the other players play a stationary strategy, the stochastic game "becomes" an MDP, and therefore Q-learning will learn to play an optimal response to the other players. So, Q-learning is rational.

On the other hand, Q-learning does not play stochastic policies. This prevents Q-learners from being convergent in self-play. The reason is that if Q-learners converge, since they're rational, they must converge to a Nash equilibrium. In games where the only equilibria are mixed equilibria (e.g., matching pennies and rock-paper-scissors as described in Section 2.1.2), Q-learners could not possibly converge. There are single-agent learning techniques that are capable of playing stochastic policies (23; 24; 25; 26). These techniques mainly address the issues of partial observability and/or function approximation in single-agent decision problems. In general, this does not solve the problem, as we will show two learners capable of playing stochastic policies, which still do not converge in self-play (See Sections 3.1 and 4.1).

2.3.2 Minimax-Q

Littman (14) was the first to examine stochastic games as a framework for multiagent reinforcement learning. He extended the traditional Q-Learning algorithm to zero-sum stochastic games. The algorithm is shown in Table 3. The notion of a Q function is extended to maintain the value of *joint* actions, and the backup operation computes the value of states differently, by replacing the max operator with the Value_i operator. The Value_i operator computes the expected payoff for player i if all the players played the unique Nash equilibrium. It is interesting to note that this is basically the off-policy reinforcement learning equivalent of Shapley's original "value iteration" algorithm for stochastic games (9).

This algorithm uses the game theoretic concept of equilibria in order to es-

(1) Initialize $Q(s \in \mathcal{S}, a \in \mathcal{A})$ arbitrarily, and set α to be the learning rate.

(2) Repeat,

   (a) Given the current state s, find the equilibrium, σ, of the matrix game $[\, Q(s,a)_{a \in \mathcal{A}} \,]$.
   (b) Select action a_i according to the distribution σ_i, with some exploration.
   (c) Observing joint-action a, reward r, and next state s',

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma V(s')),$$

   where,

$$V(s) = \mathrm{Value}_i\left(\left[\, Q(s,a)_{a \in \mathcal{A}} \,\right]\right).$$

Table 3
Algorithm: Minimax-Q for player i. The Value_i operator computes the value of a zero-sum matrix game, i.e., the expected payoff to the player if both players play the Nash equilibrium.

timate the value of a state. This value is then used to update the value of states that transition into the state. Using this computation, the Minimax-Q algorithm learns the player's part of the Nash equilibrium strategy. The only requirement is that all players execute all of their actions infinitely often (i.e., completely explore all states and actions.) This is true even if the other agent does not converge to their part of the Nash equilibrium, and so provides an *opponent-independent* method for learning an equilibrium solution.

This algorithm is guaranteed to converge in self-play. On the other hand the algorithm is not rational. Consider an opponent in rock-paper-scissors playing almost exclusively *Rock*, but playing *Paper* and *Scissors* with some small probability. Minimax-Q will find the equilibrium solution of randomizing between each of its actions equally, but this is not a best-response (playing only *Paper* in this situation is the only best-response).

2.3.3 Opponent Modelling

The final algorithm we examine is Opponent Modelling (17) or Joint-Action Learners (JALs) (18). The algorithm is shown in Table 4. The idea is to learn explicit models of the other players, assuming that they are playing according to a stationary policy. In the algorithm, $C(s, a_{-i})/n(s)$ is the estimate the other players will select joint action a_{-i} based on their past play. The player then plays the optimal response to this estimated distribution. Uther and Veloso (17) investigated this algorithm in zero-sum games and Claus and Boutilier (18) examined it for fully collaborative matrix games.

(1) Initialize Q arbitrarily, and $\forall s \in \mathcal{S}, a_{-i} \in \mathcal{A}_{-i} \quad C(s, a_{-i}) \leftarrow 0$ and $n(s) \leftarrow 0$.

(2) Repeat,

(a) From state s select action a_i that maximizes,

$$\sum_{a_{-i}} \frac{C(s, a_{-i})}{n(s)} Q(s, \langle a_i, a_{-i} \rangle)$$

(b) Observing other agents' actions a_{-i}, reward r, and next state s',

$$\begin{aligned} Q(s,a) &\leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma V(s')) \\ C(s, a_{-i}) &\leftarrow C(s, a_{-i}) + 1 \\ n(s) &\leftarrow n(s) + 1 \end{aligned}$$

where,

$$\begin{aligned} a &= (a_i, a_{-i}) \\ V(s) &= \max_{a_i} \sum_{a_{-i}} \frac{C(s, a_{-i})}{n(s)} Q(s, \langle a_i, a_{-i} \rangle). \end{aligned}$$

Table 4
Algorithm: Opponent Modeling Q-Learning for player i.

The algorithm is essentially fictitious play (16; 11) in a reinforcement learning context. Fictitious play is a game theory algorithm that has been proven to find equilibria in certain types of games. Basically, the fictitious play algorithm has players selecting the action at each iteration that would have received the highest total payoff if it had been played exclusively throughout the past. Fictitious play, when played by all players, has been proven to converge to the Nash equilibrium in games that are iterated dominance solvable. These are games such that iteratively removing dominated actions (i.e., actions whose payoffs are lower than some other strategy regardless of the opponent's play) will leave a single action or set of equivalent actions. In addition, in zero-sum games the players' empirical distribution of actions that are played will converge to the game's Nash equilibrium, even though the actual strategies being played may not. The behavior of the opponent modelling algorithm is very similar, although not all of these results have formal proofs.

Like single-agent learners, opponent modelling is rational. This is because eventually the player's estimates of its opponent's policy will converge to the true policy. Since it finds best-response policies given its estimates eventually it will converge to a best-response policy to the opponent's true policy. Also, like single-agent learning it is not convergent. The reason is identical: it only plays pure policies, and so cannot converge in games with only mixed equilibria.

2.3.4 Discussion

In summary, Single-agent learners and joint-action learners are both rational, but have no guarantee of convergence. Minimax-Q is guaranteed to converge to the equilibrium, but there's no guarantee that this is a best-response to the actual opponent. So Minimax-Q is not rational.

Although the rational and convergent properties do not encompass all that is desirable in a learning technique, it is interesting that simultaneously achieving both properties is very difficult. The rest of this article will look at a new technique to do exactly that, i.e., a rational and convergent learning algorithm. The idea is to use a variable learning rate in rational learning algorithms to make them convergent. In Section 3 we will look at a theoretical analysis of variable learning rates in a restricted class of iterated matrix games. In Section 4 we will further develop the technique into a more general stochastic game learner and show empirical results of this algorithm.

3 Theoretical Analysis

In this section we will begin by examining gradient ascent as a technique for learning in simple two-player, two-action, general-sum repeated matrix games. We will look at a theoretical analysis of this algorithm, which observes that the algorithm fails to converge. We will follow by introducing the concept of a variable learning rate, and prove that this concept, in fact, causes gradient ascent to converge.

3.1 Gradient Ascent

Singh, Kearns, and Mansour (27) examined the dynamics of using gradient ascent in two-player, two-action, iterated matrix games. We can represent this problem as two matrices,

$$R_r = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \qquad R_c = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}.$$

Each player selects an action from the set $\{1, 2\}$ which determines the rewards or payoffs to the players. If the *row* player selects action i and the *column* player selects action j, then the row player receives a payoff r_{ij} and the column player receives the payoff c_{ij}.

Since this is a two-action game, a strategy (i.e., a probability distribution over the two available actions) can be represented as a single value. Let $\alpha \in [0,1]$ be a strategy for the row player, where α corresponds to the probability the player selects the first action and $1-\alpha$ is the probability the player selects the second action. Similarly, let β be a strategy for the column player. We can consider the joint strategy (α, β) as a point in $\mathbb{R}^2$ constrained to the unit square.

For any pair of strategies (α, β), we can write the expected payoffs the row and column player will receive. Let $V_r(\alpha, \beta)$ and $V_c(\alpha, \beta)$ be these expected payoffs, respectively. Then,

$$\begin{aligned} V_r(\alpha,\beta) &= \alpha\beta r_{11} + \alpha(1-\beta) r_{12} + (1-\alpha)\beta r_{21} + (1-\alpha)(1-\beta) r_{22} \\ &= u\alpha\beta + \alpha(r_{12} - r_{22}) + \beta(r_{21} - r_{22}) + r_{22} \end{aligned} \tag{1}$$

$$\begin{aligned} V_c(\alpha,\beta) &= \alpha\beta c_{11} + \alpha(1-\beta) c_{12} + (1-\alpha)\beta c_{21} + (1-\alpha)(1-\beta) c_{22} \\ &= u'\alpha\beta + \alpha(c_{12} - c_{22}) + \beta(c_{21} - c_{22}) + c_{22} \end{aligned} \tag{2}$$

where,

$$\begin{aligned} u &= r_{11} - r_{12} - r_{21} + r_{22} \\ u' &= c_{11} - c_{12} - c_{21} + c_{22}. \end{aligned}$$

A player can now consider the effect of changing its strategy on its expected payoff. This can be computed as just the partial derivative of its expected payoff with respect to its strategy,

$$\frac{\partial V_r(\alpha,\beta)}{\partial \alpha} = \beta u + (r_{12} - r_{22}) \tag{3}$$

$$\frac{\partial V_c(\alpha,\beta)}{\partial \beta} = \alpha u' + (c_{21} - c_{22}). \tag{4}$$

In the gradient ascent algorithm a player will adjust its strategy after each iteration so as to increase its expected payoffs. This means the player will move their strategy in the direction of the current gradient with some step size, η. If (α_k, β_k) are the strategies on the kth iteration, and both players are using gradient ascent then the new strategies will be,

$$\begin{aligned} \alpha_{k+1} &= \alpha_k + \eta \frac{\partial V_r(\alpha_k,\beta_k)}{\partial \alpha_k} \\ \beta_{k+1} &= \beta_k + \eta \frac{\partial V_r(\alpha_k,\beta_k)}{\partial \beta_k}. \end{aligned}$$

If the gradient will move the strategy out of the valid probability space (i.e., the unit square) then the gradient is projected back on to the probability space. This will only occur on the boundaries of the probability space. The question to consider then is what can we expect will happen if both players are using gradient ascent to update their strategies.

Notice that this algorithm is *rational* by the properties defined in Section 2.2. This is because fixing the other player's strategy causes the player's gradient to become constant, and will eventually force the player to converge to the optimal pure strategy response. On the other hand the algorithm is not *convergent*, which is shown in (27). This is despite the fact that the algorithm can and does play mixed strategies.

The analysis, by Singh and colleagues, of gradient ascent examines the dynamics of the learners in the case of an infinitesimal step size ($\lim_{\eta \to 0}$). They call this algorithm Infinitesimal Gradient Ascent (IGA). They observe later that an algorithm with an appropriately decreasing step size will have the same properties as IGA. In the next section we will briefly outline their analysis.

3.2 Analysis of IGA

The main conclusion of Singh, Kearns, and Mansour (27) is the following theorem.

Theorem 1 *If both players follow Infinitesimal Gradient Ascent (IGA), where $\eta \to 0$, then their strategies will converge to a Nash equilibrium* OR *the average payoffs over time will converge in the limit to the expected payoffs of a Nash equilibrium.*

Their proof of this theorem proceeds by examining the dynamics of the strategy pair, (α, β). This is an affine dynamical system in $\mathbb{R}^2$ where the dynamics are defined by the differential equation,

$$\begin{bmatrix} \frac{\partial \alpha}{\partial t} \\ \frac{\partial \beta}{\partial t} \end{bmatrix} = \begin{bmatrix} 0 & u \\ u' & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} (r_{12} - r_{22}) \\ (c_{21} - c_{22}) \end{bmatrix}.$$

If we define U to be the multiplicative matrix term above with off-diagonal values u and u', then we can classify the dynamics of the system based on properties of U. From dynamical systems theory, if U is invertible then there are only two qualitative forms for the dynamics of the system, depending on whether U has purely real or purely imaginary eigenvalues. This results in three cases: U is not invertible, U has purely real eigenvalues, or U has purely imaginary eigenvalues. The qualitative forms of these different cases are shown in Figure 1. Their analysis then proceeded by examining each case geometrically. One important consideration is that the basic forms above are for the unconstrained dynamics not the dynamics that projects the gradient onto the unit square. Basically, this requires considering all possible positions of the unit square relative to the dynamics shown in Figure 1.

One crucial aspect to their analysis were points of zero-gradient in the constrained dynamics, which they show to correspond to Nash equilibria. This is also discussed in Lemma 2. In the unconstrained dynamics, there exist at most one point of zero-gradient, which is called the center and denoted (α^*, β^*). This point can be found mathematically by setting equations 3 and 4 to zero and solving,

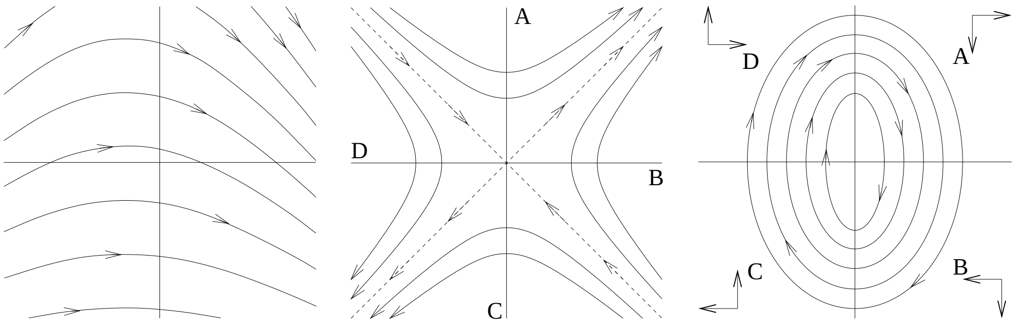
$$(\alpha^*, \beta^*) = \left(\frac{(c_{22} - c_{21})}{u'}, \frac{(r_{22} - r_{12})}{u} \right).$$

Notice that the center may not even be inside the unit square. In addition, if U is not invertible then there is no point of zero gradient in the unconstrained dynamics. But in the constrained dynamics, where gradients on the boundaries of the unit square are projected onto the unit square, additional points of zero gradient may exist. When IGA converges it will be to one of these points with zero gradient.

This theorem is an exciting result since it is one of the first convergence results for a rational multiagent learning algorithm. The notion of convergence, though, is rather weak. In fact, not only may the players' policies not converge when playing gradient ascent but the expected payoffs may not converge either. Furthermore, at any moment in time the expected payoff of a player could be arbitrarily poor.[4] Not only does this make it difficult to evaluate a learner, it also could be potentially disastrous when applied with temporal differencing for multiple state stochastic games, which assumes that expected payoffs in the past predict expected payoffs in the future.

In the next section we will examine a method for addressing this convergence problem. We will then prove that this new method has the stronger notion of convergence, i.e., players will *always* converge to a Nash equilibrium.

[4] The idea that average payoffs converge only means that if there's a period of arbitrarily low payoffs there must be some corresponding period in the past or in the future of arbitrarily high payoffs.



(a) U is not invertible

(b) U has real eigenvalues

(c) U has imaginary eigenvalues

Fig. 1. Qualitative forms of the IGA dynamics.

3.3 Variable Learning Rate

We now introduce the concept and study the impact of a variable learning rate. In the gradient ascent algorithm presented above the steps taken in the direction of the gradient were constant. We will now allow them to vary over time, thus changing the update rules to,

$$\alpha_{k+1} = \alpha_k + \eta \ell_k^r \frac{\partial V_r(\alpha_k, \beta_k)}{\partial \alpha}$$
$$\beta_{k+1} = \beta_k + \eta \ell_k^c \frac{\partial V_r(\alpha_k, \beta_k)}{\partial \beta}$$

where,

$$\ell_k^{r,c} \in [\ell_{\min}, \ell_{\max}] > 0.$$

At the kth iteration the algorithm takes a step of size $\eta \ell_k$ in the direction of the gradient. Notice the restrictions on ℓ_k require that it be strictly positive and bounded, thus bounding the step sizes as well.

The specific method for varying the learning rate that we are contributing is the WoLF ("Win or Learn Fast") principle. The essence of this method is to learn quickly when losing, and cautiously when winning. The intuition is that a learner should adapt quickly when it is doing more poorly than expected. When it is doing better than expected, it should be cautious since the other players are likely to change their policy. The heart of the algorithm is how to determine whether a player is winning or losing. For the analysis in this section each player will select a Nash equilibrium and compare their expected payoff with the payoff they would receive if they played according to the selected equilibrium strategy. Let α^e be the equilibrium strategy selected by the row player, and β^e be the equilibrium strategy selected by the column player. Notice that no requirement is made that the players choose the same equilibrium (i.e., the strategy pair (α^e, β^e) may not be a Nash equilibrium). Formally,

$$\ell_k^r = \begin{cases} \ell_{\min} \text{ if } V_r(\alpha_k, \beta_k) > V_r(\alpha^e, \beta_k) & \textsc{winning} \\ \ell_{\max} \text{ otherwise} & \textsc{losing} \end{cases}$$

$$\ell_k^c = \begin{cases} \ell_{\min} \text{ if } V_c(\alpha_k, \beta_k) > V_c(\alpha_k, \beta^e) & \textsc{winning} \\ \ell_{\max} \text{ otherwise} & \textsc{losing} \end{cases}$$

With a variable learning rate such as this we can still consider the case of an infinitesimal step size ($\lim_{\eta \to 0}$). We will call this algorithm WoLF-IGA and in the next section show that the WoLF adjustment has a very interesting effect on the convergence of the algorithm.

3.4 Analysis of WoLF-IGA

We will prove the following result.

Theorem 2 *If in a two-person, two-action, iterated general-sum game, both players follow the WoLF-IGA algorithm (with $\ell_{\max} > \ell_{\min}$), then their strategies will converge to a Nash equilibrium.*

Notice that this is the more standard notion of convergence and strictly stronger than what is true for basic IGA.

The proof of this theorem will follow closely with the proof of Theorem 1 from Singh and colleagues (27), by examining the possible cases for the dynamics of the learners. First, let us write down the differential equations that define the system with an infinitesimal step size,

$$\begin{bmatrix} \frac{\partial \alpha}{\partial t} \\ \frac{\partial \beta}{\partial t} \end{bmatrix} = \begin{bmatrix} 0 & \ell^r(t)u \\ \ell^c(t)u' & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} \ell^r(t)(r_{12} - r_{22}) \\ \ell^c(t)(c_{21} - c_{22}) \end{bmatrix}.$$

We will call the multiplicative matrix with off-diagonal entries $U(t)$ since it now depends on the learning rates at time t, $\ell^r(t)$ and $\ell^c(t)$. At time t, the qualitative form of the dynamics is determined by the $U(t)$ matrix and can be summarized into three general cases,

- $U(t)$ is not invertible,

- $U(t)$ has purely real eigenvalues, or
- $U(t)$ has purely imaginary eigenvalues.

The first thing to note is that the above cases do not depend on t. The following lemma is even stronger.

Lemma 1 *$U(t)$ is invertible if and only if U (as defined for IGA in Section 3.2) is invertible. $U(t)$ has purely imaginary eigenvalues if and only if U has purely imaginary eigenvalues. $U(t)$ has purely real eigenvalues if and only if U has purely real eigenvalues.*

Proof. Since $\ell^{r,c}(t)$ is positive, it is trivial to see that $(\ell^r(t)u)(\ell^c(t)u') = (\ell^r(t)\ell^c(t))uu'$ is greater-than, less-than, or equal-to zero, if and only if uu' is greater-than, less-than, or equal-to zero, respectively. Since these are the exact conditions of invertibility and purely real/imaginary eigenvalues the lemma is true. $\square$

So $U(t)$ will always satisfy the same case (and therefore have the same general dynamics) as IGA *without* a variable learning rate. In the sections that follow we will be examining each of these cases separately. The proofs of most of the cases will proceed identically to the proof for IGA. In fact most of the proof will not rely on any particular learning rate adjustment at all. Only in the final sub-case of the final case will we be forced to deviate from their arguments. This is due to the fact that variable learning rates in general do not change the overall direction of the gradient (i.e., the sign of the partial derivatives). Since most of the proof of IGA's convergence only depends on the signs of the derivatives, we can use the same arguments. For these cases we will present only an abbreviated proof of convergence to illustrate that the variable learning rate does not affect their arguments. We recommend the IGA analysis (27) for a more thorough examination including helpful diagrams. In the remaining sub-case, where IGA is shown not to converge, we will show that in this case WoLF-IGA will converge to a Nash equilibrium.

We will make liberal use of a crucial lemma from their proof for IGA. This lemma implies that if the algorithms converge then what the strategies converge to must be a Nash equilibrium.

Lemma 2 *If, in following IGA or WoLF-IGA, $\lim_{t\to\infty}(\alpha(t), \beta(t)) = (\alpha_c, \beta_c)$, then (α_c, β_c) is a Nash equilibrium.*

Proof. The proof for IGA is given in (27), and shows that the algorithm converges if and only if the projected gradient is zero, and such strategy pairs must be a Nash equilibrium. For WoLF-IGA notice also that the algorithm converges if and only if the projected gradient is zero, which is true if and only if the projected gradient in IGA is zero. Therefore that point must be a Nash equilibrium. $\square$

Now we will examine the individual cases.

3.4.1 $U(t)$ is Not Invertible

In this case the dynamics of the strategy pair has the qualitative form shown in Figure 1(a).

Lemma 3 *When U(t) is not invertible, IGA with any learning rate adjustment leads the strategy pair to converge to a point on the boundary that is a Nash equilibrium.*

Proof. Notice that $U(t)$ is not invertible if and only if u or u' is zero. Without loss of generality, assume u is zero, then the gradient for the column player is constant. The column player's strategy, β, will converge to either zero or one (depending on whether the gradient was positive or negative). At this point, the row player's gradient becomes constant and therefore must also converge to zero or one, depending on the sign of the gradient. The joint strategy therefore converges to some corner, which by Lemma 2 is a Nash equilibrium. □

3.4.2 $U(t)$ has Real Eigenvalues

In this case the dynamics of the strategy pair has the qualitative form shown in Figure 1(b).

Lemma 4 *When U(t) has real eigenvalues, IGA with any learning rate adjustment leads the strategy pair to converge to a point that is a Nash equilibrium.*

Proof. Without loss of generality, assume that $u, u' > 0$. This is the dynamics show in Figure 1(b). Consider the case where the center is inside the unit square. Notice that if the strategy pair is in quadrant A, the gradient is always up and right. Therefore, any strategy pair in this region will eventually converge to the upper-right corner of the unit square. Likewise, strategies in quadrant C will always converge to the bottom-left corner. Now consider a strategy pair in quadrant B. The gradient is always up and left, and therefore the strategy will eventually exit this quadrant, entering quadrant A or C, or possibly hitting the center. At the center the gradient is zero, and so it has converged. If it enters one of quadrants A or C then we've already shown it will converge to the upper-right or lower-left corner. Therefore, the strategies always converge and by Lemma 2 the point must be a Nash equilibrium. Cases where the center is not within the unit square or is on the boundary of the unit square can also be shown to converge by a similar analysis, and are discussed in (27). □

3.4.3 $U(t)$ has Imaginary Eigenvalues

In this case the dynamics of the strategy pair has the qualitative form shown in Figure 1(c). This case can be further broken down into sub-cases depending where the unit square is in relation to the center.

Center is Not Inside the Unit Square. In this case we still can use the same argument as for IGA.

Lemma 5 *When U(t) has imaginary eigenvalues and the center, (α^*, β^*), is not inside the unit square, IGA with any learning rate adjustment leads the strategy pair to converge to a point on the boundary that is a Nash equilibrium.*

Proof. There are three cases to consider. The first is the unit square lies entirely within a single quadrant. In this case the direction of the gradient will be constant (e.g., down-and-right in quadrant A). Therefore the strategies will converge to the appropriate corner (e.g., bottom-right corner in quadrant A). The second case is the unit square is entirely within two neighboring quadrants. Consider the case that it lies entirely within quadrants A and D. The gradient always points to the right and therefore the strategy will eventually hit the right boundary at which point it will be in quadrant A and the gradient will be pointing downward. Therefore in this case it will converge to the bottom right corner. We can similarly show convergence for other pairs of quadrants. The third and final case is when the center is on the boundary of the unit square. In this case some points along the boundary will have a projected gradient of zero. By similar arguments to those above, any strategy will converge to one of these boundary points. See (27) for a diagram and further explanation. Since in all cases the strategy pairs converge, by Lemma 2 they must have converged to a Nash equilibrium. □

Center is Inside the Unit Square. This is the final sub-case and is the point where the dynamics of IGA and WoLF-IGA qualitatively differ. We will show that, although IGA will not converge in this case, WoLF-IGA will. The proof will identify the areas of the strategy space where the players are "winning" and "losing" and show that the trajectories are actually piecewise elliptical in such a way that they spiral towards the center. All of the lemmas in this subsection implicitly assume that $U(t)$ has imaginary eigenvalues and the center is inside the unit square. We begin with the following lemma that considers the dynamics for fixed learning rates.

Lemma 6 *If the learning rates, ℓ^r and ℓ^c, remain constant, then the trajectory of the strategy pair is an elliptical orbit around the center, (α^*, β^*), and the* *axes of this ellipse are,*

$$\begin{bmatrix} 0 \\ \sqrt{\frac{\ell^c|u|}{\ell^r|u'|}} \end{bmatrix}, \quad \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Proof. This is just a result from dynamical systems theory (28) as mentioned in (27) when $U(t)$ has imaginary eigenvalues. □

We now need the critical lemma that identifies the areas of strategy space where the players are using a constant learning rate. Notice that this corresponds to the areas where the players are "winning" or "losing".

Lemma 7 *The player is "winning" if and only if that player's strategy is moving away from the center.*

Proof. Notice that in this sub-case where $U(t)$ has imaginary eigenvalues and the center is within the unit square, the game has a single Nash equilibrium, which is the center. So, the players' selected equilibrium strategies for the WoLF principle must be the center, i.e., $(\alpha^e, \beta^e) = (\alpha^*, \beta^*)$. Now, consider the row player. The player is "winning" when its current expected payoff is larger than the expected payoffs if it were to play its selected equilibrium. This can be written as,

$$V_r(\alpha, \beta) - V_r(\alpha^e, \beta) > 0. \tag{5}$$

We can rewrite the left hand side of inequality 5 by using equation 1,

$$\begin{aligned} &(\alpha\beta u + \alpha(r_{12} - r_{22}) + \beta(r_{21} - r_{22}) + r_{22}) - \\ &(\alpha^e\beta u + \alpha^e(r_{12} - r_{22}) + \beta(r_{21} - r_{22}) + r_{22}). \end{aligned} \tag{6}$$

Using expression 6 in inequality 5, we substitute the center for the equilibrium strategies, and then simplify making use of equation 3, as presented in Section 3.1.

$$(\alpha - \alpha^*)\beta u + (\alpha - \alpha^*)(r_{12} - r_{22}) > 0 \tag{7}$$
$$(\alpha - \alpha^*)(\beta u + (r_{12} - r_{22})) > 0 \tag{8}$$
$$(\alpha - \alpha^*)\frac{\partial V_r(\alpha, \beta)}{\partial \alpha} > 0. \tag{9}$$

Notice that inequality 9 is satisfied if and only if the two left hand factors have the same sign. This is true if and only if the player's strategy α is greater than the strategy at the center α^* and it is increasing, or it's smaller than the center and decreasing. So the player is winning if and only if its strategy is moving away from the center. The same can be shown for the column player. $\square$

Corollary 1 *Throughout any one quadrant, the learning rate is constant.*

Combining Lemmas 6 and 7, we find that the trajectories will be piece-wise elliptical orbits around the center, where the pieces correspond to the quadrants defined by the center. We can now prove convergence for a limited number of starting strategy pairs. We will then use this lemma to prove convergence for any initial strategy pairs.

Lemma 8 *For any initial strategy pair, $(\alpha^*, \beta^* + \beta_0)$ or $(\alpha^* + \alpha_0, \beta^*)$, that is "sufficiently close" to the center, the strategy pair will converge to the center. "Sufficiently close" here means that the elliptical trajectory from this point defined when both players use 1 as their learning rate lies entirely within the unit square.*

Proof. Without loss of generality assume $u > 0$ and $u' < 0$. This is the case shown in Figure 1(c). Let $l = \sqrt{\frac{\ell_{\min}}{\ell_{\max}}} < 1.0$, and $r = \sqrt{\frac{|u'|}{|u|}}$. Consider an initial strategy $(\alpha^*, \beta^* + \beta_0)$ with $\beta_0 > 0$.

For any fixed learning rates for the players, the trajectory forms an ellipse centered at (α^*, β^*) and with the ratio of its y-radius to its x-radius equal to,

$$\sqrt{\frac{\ell^c}{\ell^r}} r.$$

Since the trajectory is piecewise elliptical we can consider the ellipse that the trajectory follows while in each quadrant. This is shown graphically in Figure 2. As the trajectory travels through quadrant A, by Lemma 7, we can observe that the row player is "winning" and the column player is "losing". Therefore, $\ell^r = \ell_{\min}$ and $\ell^c = \ell_{\max}$, so the ratio of the ellipse's axes will be r/l, and this ellipse will cross into quadrant B at the point $(\alpha^* + \beta_0 \frac{l}{r}, \beta^*)$. Similarly, in quadrant B, the row player is "losing" and the column player is "winning" therefore the ratio of the ellipse's axes will be rl and the ellipse will cross into quadrant C at the point $(\alpha^*, \beta^* - \beta_0 l^2)$.

We can continue this to return to the axis where the trajectory began. The strategy pair at that point will be $(\alpha^*, \beta^* + \beta_0 l^4)$. So, for each orbit around the center we decrease the distance to the center by a factor of $l^4 < 1.0$, and therefore the trajectory will converge to the center. We can reason identically for any other sufficiently close initial strategies on the axes. $\square$

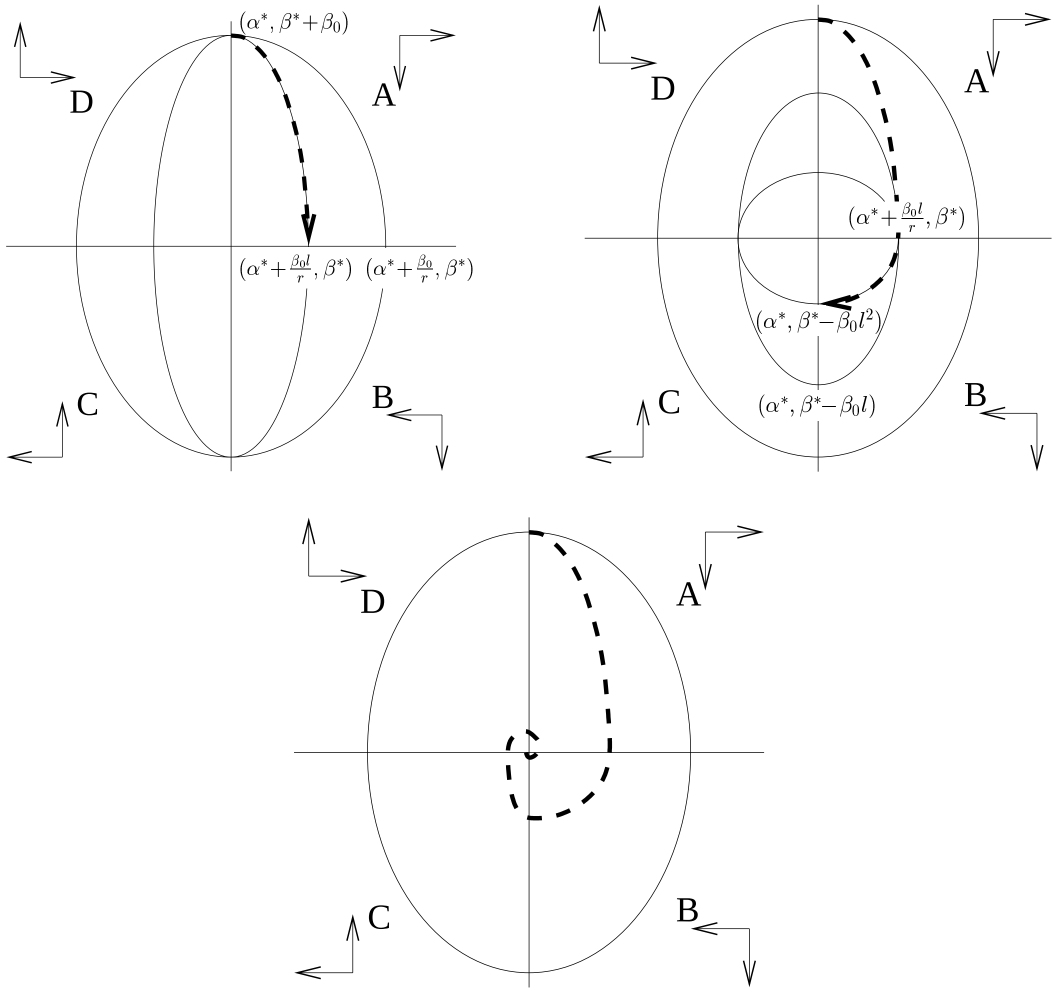


Fig. 2. The trajectory of learning rates using WoLF-IGA when $U(t)$ has imaginary eigenvalues and the center is inside the unit square.

Lemma 9 *When U(t) has imaginary eigenvalues and the center, (α^*, β^*), is inside the unit square, WoLF-IGA leads the strategy pair to converge to the center, and therefore to a Nash equilibrium.*

Proof. The proof just involves the application of Lemma 8. Consider the largest ellipse, when both players' learning rates are one, that fits entirely within the unit square. This ellipse will touch the boundary of the unit square and do so at the boundary of two quadrants. Now consider any initial strategy pair. The strategy pair will follow piecewise elliptical orbits or move along the unit square boundary while "circling" the center, that is travelling through the four quadrants in a clockwise or counter-clockwise fashion. At some point it must cross the boundary between the same two quadrants mentioned above. At this point it is on or inside the largest ellipse defined when players have a learning rate of one. Therefore we can apply Lemma 8 and so the trajectory will converge to the center. So, from any initial strategy pair the trajectory will converge to the center, which is a Nash equilibrium. □

Lemmas 3, 4, 5, and 9 combine to prove Theorem 2. In summary, the WoLF principle strengthens the IGA convergence result. In self-play with WoLF-IGA, players' strategies and their expected payoffs converge to Nash equilibrium strategies and payoffs of the matrix game. This contributes a reinforcement-based algorithm that is provably *rational* and *convergent* (in self-play) for this restricted class of iterated matrix games. This result can be generalized beyond self-play in the following corollary.

Corollary 2 *If in a two-person, two-action, iterated general-sum game, both players follow the WoLF-IGA algorithm but with different $\ell_{\min}$ and $\ell_{\max}$, then their strategies will converge to a Nash equilibrium if,*

$$\frac{\ell^r_{\min}\,\ell^c_{\min}}{\ell^r_{\max}\,\ell^c_{\max}} < 1.$$

Specifically, WoLF-IGA (with $\ell_{\max} > \ell_{\min}$) versus IGA ($\ell_{\max} = \ell_{\min}$) will converge to a Nash equilibrium.

Proof. The proof is almost identical to Theorem 2. The only deviation is for the imaginary eigenvalue case where the center is inside the unit square. In this case the proof of Lemma 8 is amended. Let $l^r = \sqrt{\frac{\ell^r_{\min}}{\ell^c_{\max}}}$ and $l^c = \sqrt{\frac{\ell^c_{\min}}{\ell^r_{\max}}}$. Following the same argument of the trajectory through the quadrants, after a revolution around the center the new position will be $(\alpha^*, \beta^* + \beta_0(l^r l^c)^2)$. Since,

$$(l^r l^c)^2 = \frac{\ell^r_{\min}\,\ell^c_{\min}}{\ell^r_{\max}\,\ell^c_{\max}} < 1,$$

the trajectory converges to the center. The remainder of the proof is identical to that of Theorem 2. □

We will return to examining WoLF outside of self-play in Section 5.5, where we examine this situation empirically in a more complex domain.

3.5 Discussion

There are some final points to be made about this result. First, we will present some further justification for the WoLF principle as it has been used in other learning related problems. Second, we will present a short discussion on determining when a player is "winning". Finally, we will look at the knowledge the WoLF-IGA algorithm requires. These requirements will be later relaxed in a more practical algorithm in Section 4.

3.5.1 Why WoLF?

Apart from this theoretical result the WoLF principle may appear to be just an unfounded heuristic. But actually it has been studied in some form in other areas, notably when considering an adversary. In evolutionary game theory the *adjusted replicator dynamics* (29) scales the individual's growth rate by the inverse of the overall success of the population. This will cause the populations composition to change more quickly when the population as a whole is performing poorly. A form of this also appears as a modification to the *randomized weighted majority* algorithm (30). In this algorithm, when an expert makes a mistake, a portion of its weight loss is redistributed among the other experts. If the algorithm is placing large weights on mistaken experts (i.e., the algorithm is "losing"), then a larger portion of the weights are redistributed (i.e., the algorithm adapts more quickly.)

3.5.2 Defining "Winning"

The WoLF principle for adjusting the learning rate is to learn faster when losing, more slowly when winning. This places a great deal of emphasis on how to determine that a player is winning. In the description of WoLF-IGA above, the row-player was considered winning when,

$$V_r(\alpha_k, \beta_k) > V_r(\alpha^e, \beta_k).$$

Essentially, the player was winning if he'd prefer his current strategy to that of playing some equilibrium strategy against the other player's current strategy.

Another possible choice of determining when a player is winning is if his expected payoff is currently larger than the value of the game's equilibrium (or some equilibrium if multiple exist). If we consider the final subcase in Section 3.4.3, then mathematically this would correspond to,

$$V_r(\alpha_k, \beta_k) > V_r(\alpha^*, \beta^*).$$

It is interesting to note that in zero-sum games with mixed strategy equilibria these two rules are actually identical.

In general-sum games, though, this is not necessarily the case. This situation should bring to light the differences between the two methods. There exist general-sum two-player, two-action games with points in the strategy space where the player is actually receiving a lower expected payoff than the equilibrium value, but not lower than the expected payoff of the equilibrium strategy against that player. Essentially, the player is not doing poorly because his strategy is poor, but rather because of the play of the other player. It is at this point that the gradient is likely to be moving the strategy away from the equilibrium, and so using the latter rule to determine when winning, the player would move away from the equilibrium quickly, and discourage convergence. [5]

3.5.3 Requirements

The gradient ascent and WoLF gradient ascent algorithms make some strict requirements on the knowledge that is available to the player. Specifically, basic gradient ascent requires the following to be known: the player's own payoff matrix (i.e., R_r for the row-player), and the actual distribution of actions the other player is playing (β_k for the row-player). These two requirements are both very strong, particularly the latter. Often the payoffs are not known and rather need to be learned via experience, and even more often only the action selected by the other player is known (if even that) but not the player's distribution over actions.

WoLF gradient ascent proceeds to add a further requirement that a Nash equilibrium must also be known. In this case, this is not really extra knowledge, since it can be computed from the known payoffs. If we want to make this algorithm more general, though, it will need to be addressed how the algorithm determines whether it's winning or losing when an equilibrium is not known because the payoffs are not known. In the next section we will look at an algorithm that removes all of these knowledge constraints and we give empirical results on more general stochastic games, not just two-player, two-action matrix games.

4 A Practical Algorithm

We now will present a more general algorithm using the WoLF principle to vary the learning rate. We will begin by describing a simple rational algorithm, policy hill-climbing (PHC), that does not converge. This algorithm is similar to gradient ascent, but does not require as much knowledge. We will then describe WoLF-PHC, which varies the learning rate according to an approximate notion of winning. In Section 5, these algorithms will then be examined in a number of

---

[5] An example of such a matrix game is, $R_r = \begin{bmatrix} 0 & 3 \\ 1 & 2 \end{bmatrix}$, $R_c = \begin{bmatrix} 3 & 2 \\ 0 & 1 \end{bmatrix}$, with the strategy point, $(\alpha, \beta) = (0.1, 0.9)$. The only Nash equilibrium is $(0.5, 0.5)$. Hence, $V_r(\alpha^*, \beta) = 0.7 < V_r(\alpha, \beta) = 1.02 < V_r(\alpha^*, \beta^*) = 2$, and so the two rules would disagree. Notice the gradient, $\frac{\partial V_r(\alpha,\beta)}{\partial \alpha} = -0.8$, is negative causing α to decrease away from the equilibrium.

(1) Let $\alpha \in (0,1]$ and $\delta \in (0,1]$ be learning rates. Initialize,

$$Q(s,a) \leftarrow 0, \qquad \pi(s,a) \leftarrow \frac{1}{|\mathcal{A}_i|}.$$

(2) Repeat,

(a) From state s select action a according to mixed strategy $\pi(s)$ with suitable exploration.

(b) Observing reward r and next state s',

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') \right).$$

(c) Step π closer to the optimal policy w.r.t. Q,

$$\pi(s,a) \leftarrow \pi(s,a) + \Delta_{sa},$$

while constrained to a legal probability distribution,

$$\Delta_{sa} = \begin{cases} -\delta_{sa} & \text{if } a \neq \operatorname{argmax}_{a'} Q(s,a') \\ \sum_{a' \neq a} \delta_{sa'} & \text{otherwise} \end{cases}$$
$$\delta_{sa} = \min\left(\pi(s,a), \frac{\delta}{|A_i| - 1} \right).$$

Table 5
The policy hill-climbing algorithm (PHC) for player i.

empirical examples from matrix games to zero-sum and general-sum stochastic games.

4.1 Policy Hill-Climbing

We first present a simple rational learning algorithm that is capable of playing mixed strategies. It is a simple extension of Q-learning and is shown in Table 5. The algorithm, in essence, performs hill-climbing in the space of mixed policies. Q-values are maintained just as in normal Q-learning. In addition the algorithm maintains the current mixed policy. The policy is improved by increasing the probability that it selects the highest valued action according to a learning rate $\delta \in (0,1]$. Notice that when $\delta = 1$ the algorithm is equivalent to Q-learning, since with each step the policy moves to the greedy policy executing the highest valued action with probability 1 (modulo exploration).

(1) Let $\alpha \in (0,1]$, $\delta_l > \delta_w \in (0,1]$ be learning rates. Initialize,

$$Q(s,a) \leftarrow 0, \qquad \pi(s,a) \leftarrow \frac{1}{|\mathcal{A}_i|}, \qquad C(s) \leftarrow 0.$$

(2) Repeat,

(a) Same as PHC in Table 5.
(b) Same as PHC in Table 5.
(c) Update estimate of average policy, $\bar{\pi}$,

$$\begin{aligned} C(s) &\leftarrow C(s) + 1 \\ \forall a' \in \mathcal{A}_i \quad \bar{\pi}(s,a') &\leftarrow \bar{\pi}(s,a') + \frac{1}{C(s)} \left(\pi(s,a') - \bar{\pi}(s,a') \right). \end{aligned}$$

(d) Step π closer to the optimal policy w.r.t. Q. Same as PHC in Table 5(c), but with

$$\delta = \begin{cases} \delta_w & \text{if } \sum_{a'} \pi(s,a')Q(s,a') > \sum_{a'} \bar{\pi}(s,a')Q(s,a') \\ \delta_l & \text{otherwise} \end{cases}.$$

Table 6
The WoLF policy-hill climbing algorithm (WoLF-PHC) for player i.

This technique, like Q-learning, is rational and will converge to an optimal policy if the other players are playing stationary strategies. The proof follows from the proof for Q-learning (20), which guarantees the Q values will converge to Q^* with a suitable exploration policy.[6] π will converge to a policy that is greedy according to Q, which is converging to Q^*, and therefore will converge to a best response. Despite the fact that it is rational and can play mixed policies, it still doesn't show any promise of being convergent. We show examples of its convergence failures in Section 5.

4.2 WoLF Policy Hill-Climbing

We now introduce a modification to the naïve policy hill-climbing algorithm that encourages the desired convergence property without sacrificing the rational property. The algorithm uses a variable learning rate, δ, with the WoLF, "Win or Learn Fast", principle. The required changes to policy hill-climbing are shown in Table 6.

[6] For all algorithms in this article we assume that an exploration policy suitable for online learning (31) is used.

As a reminder, the WoLF principle aids in convergence by giving more time for the other players to adapt to changes in the player's strategy that at first appear beneficial, while allowing the player to adapt more quickly to other players' strategy changes when they are harmful. Practically, the algorithm requires two learning parameters $\delta_l > \delta_w$. The parameter that is used to update the policy depends on whether the agent is currently determined to be winning or losing. This determination is done by comparing whether the current expected value is greater than the current expected value of the average policy. If the current expected value is lower (i.e., the agent is "losing"), then the larger learning rate δ_l is used, otherwise δ_w is used. The average policy is intended to take the place of the unknown equilibrium policy. For many games, averaging over greedy policies does in fact approximate the equilibrium, which is the driving mechanism in fictitious play (11). In summary, WoLF-PHC introduces (i) two learning rates, and (ii) the determination of winning and losing using the average policy as an approximation for the equilibrium policy.

WoLF policy hill-climbing is still rational, since only the speed of learning is altered. Its convergence properties, though, are quite different. In the next section we show examples that this technique converges to best-response policies for a number and variety of stochastic games.

5 Results

We now show results of applying policy hill-climbing and WoLF policy hill-climbing to a number of different games. The domains include two matrix games that help to show how the algorithms work and to demonstrate empirically what the theoretical results in Section 3 predicted. The algorithms were also applied to two multi-state stochastic games. One is a general-sum grid world domain used by Hu & Wellman (15). The other is a zero-sum soccer game introduced by Littman (14). We also examine a final matrix game that involves three players. Finally, we briefly look at the performance of WoLF in a few non-self-play experiments.

The first set of experiments involves training the players using the same learning algorithm. Since PHC and WoLF-PHC are rational, we know that if they converge against themselves, then they must converge to a Nash equilibrium (See Section 2.2). For the matrix game experiments $\delta_l/\delta_w = 2$, but for the stochastic game results a more aggressive $\delta_l/\delta_w = 4$ was used. The smaller ratio was used in the matrix games to *slow down* the convergence of the WoLF algorithm to make the affects of WoLF more visible. In all cases both the δ and α learning rates were decreased by a factor inversely proportionate to the iterations through step (2) of the algorithms, although the exact proportion varied between domains and are reported in the appendix. These learning rates were of course set by hand, but with little or no fine tuning. In practical application, setting the α learning rate and decay is identical to setting this learning rate in Q-learning. The δ learning rate and decay requires some amount of care to have it set appropriate for the value of the α learning rate, i.e., we don't want to take larger steps than we can update values for. Also, all experiments used ϵ-greedy exploration was used with a fixed exploration rate of 5%.

When the results show policy trajectories, these correspond to a single training run of the algorithm showing a prototypical trajectory over time. Many training runs have been performed in each domain, all with very similar trajectories to the one shown in this article.

5.1 Matching Pennies and Rock-Paper-Scissors

The algorithm was applied to two zero-sum matrix games, matching-pennies and rock-paper-scissors (see Table 1). In both games, the Nash equilibrium is a mixed policy consisting of executing the actions with equal probability. As mentioned, a small learning rate ratio and a large number of trials was used in order to better visualize how the algorithm learns and converges.

Figure 3 shows the results of applying both policy hill-climbing and WoLF policy hill-climbing to the matching pennies game. WoLF-PHC quickly begins to oscillate around the equilibrium with ever decreasing amplitude. Without the WoLF modification, it oscillates around the equilibrium, but with no appearance of converging. This is even more obvious in the game of rock-paper-scissors. The results in Figure 4 show trajectories of the players' strategies in policy space through one million steps. Policy hill-climbing circles the equilibrium policy without any hint of converging, while WoLF policy hill-climbing very nicely spirals towards the equilibrium. The behavior is nearly identical to the theoretically proven behavior of WoLF-IGA.

5.2 Gridworld

We also examined Hu's gridworld domain (32) shown in Figure 5. The agents start in two corners and are trying to reach the goal square on the opposite wall. The players have the four compass actions (i.e., N, S, E, and W), which are in most cases deterministic. If the two players attempt to move to the same square, both moves fail. To make the game interesting and force the players to interact, from the initial starting position the North action is uncertain, and it moves the player North with probability 0.5. Hence, the optimal path

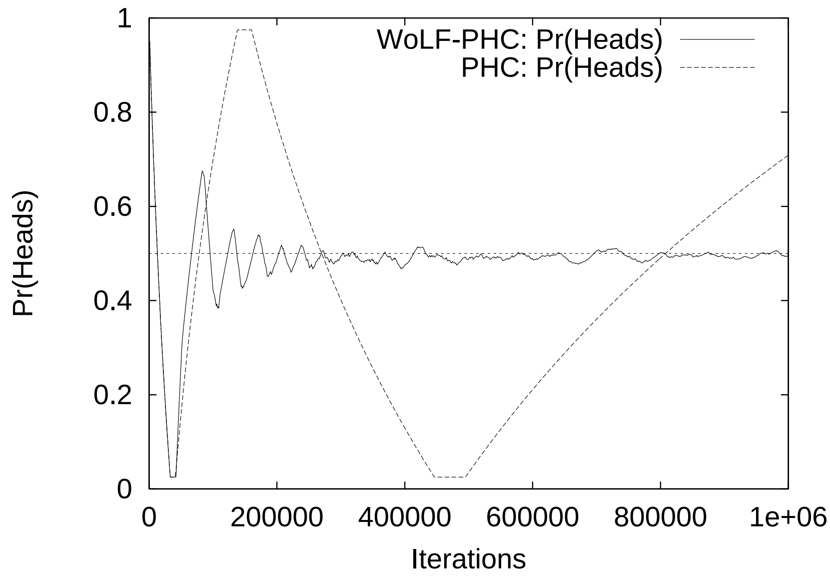


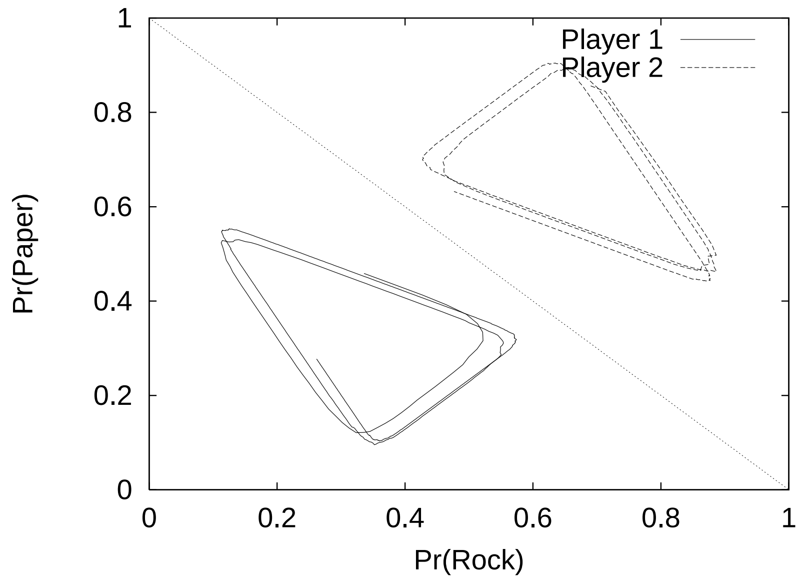
Fig. 3. Results for the matching pennies matrix game: the policy for one of the players as a probability distribution while learning with PHC and WoLF-PHC. The other player's policy looks similar.

for each agent is to move laterally square on the first move and then move North to the goal, but if both players move laterally then the actions will fail. There are two Nash equilibria for this game. They involve one player taking the lateral move and the other trying to move North. So the players must coordinate their actions.

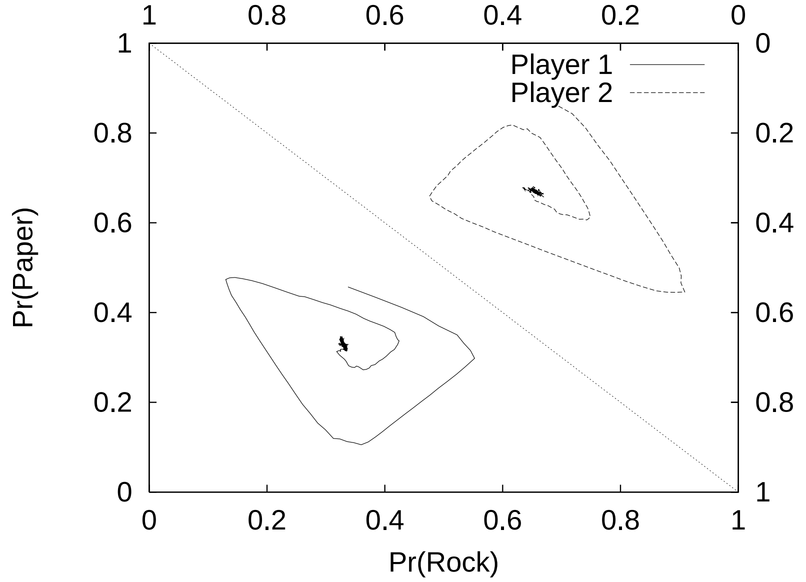
WoLF policy hill-climbing successfully converges to one of these equilibria. Figure 5 shows an example trajectory of the players' strategies for the initial state while learning over 100,000 steps. In this example the players converged to the equilibrium where Player 1 moves East and Player 2 moves North from the initial state. This is evidence that WoLF policy hill-climbing (and in this case unmodified PHC as well) can learn an equilibrium even in a general-sum game with multiple equilibria.

5.3 Soccer

The final domain is a comparatively large zero-sum soccer game introduced by Littman (14) to demonstrate Minimax-Q. Figure 6 shows an example of an initial state in this game, where Player B has possession of the ball. The goal is for the players to carry the ball into the goal on the opposite side of the field. The actions available are the four compass directions as well as not moving. The players select actions simultaneously but they are executed in a random order, which adds non-determinism to their actions. If a player attempts to move to the square occupied by its opponent, the stationary player



(a) Policy Hill-Climbing



(b) WoLF Policy Hill-Climbing

Fig. 4. Results for the rock-paper-scissors matrix game: trajectories of the two players' policies while learning with (a) PHC, and (b) WoLF-PHC through one million iterations.

gets possession of the ball, and the move fails. Unlike the grid world domain, the Nash equilibrium for this game requires a mixed policy. In fact any deterministic policy (therefore anything learned by a single-agent learner or JAL) can always be defeated (14).

Our experimental setup resembles that used by Littman (14). Each player was trained in self-play for one million steps. After training, the player's policy was fixed and a challenger using Q-learning was trained against the player. This

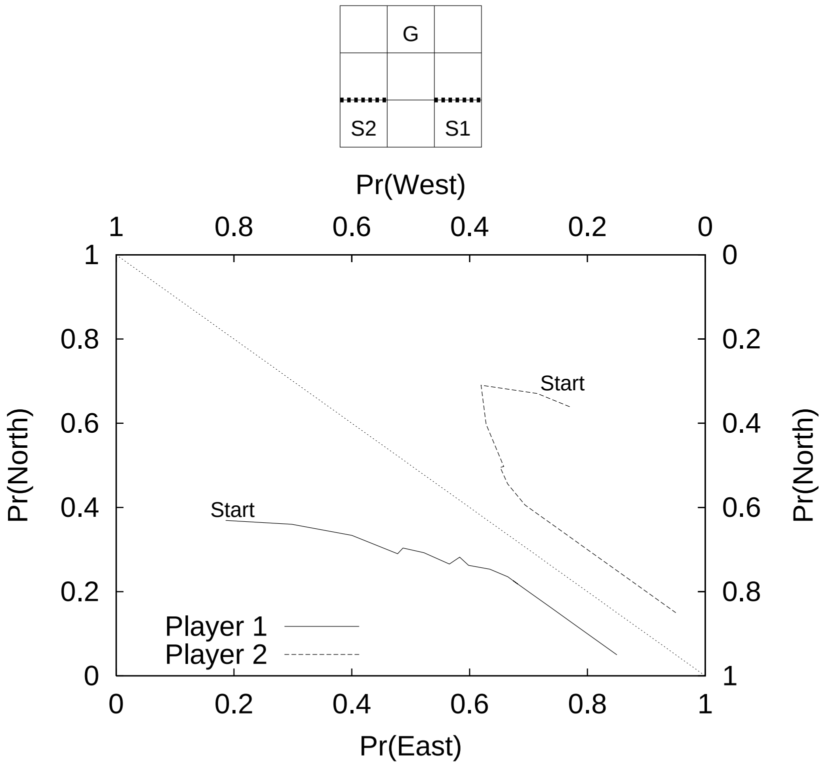


Fig. 5. Gridworld game. The dashed walls represent the actions that are uncertain. The results show trajectories of two players' policies for the initial state while learning with WoLF-PHC.

determines the learned policy's worst-case performance and gives an idea of how close the player was to the equilibrium policy, which would perform no worse than losing half its games to its challenger. Unlike Minimax-Q, WoLF-PHC and PHC generally oscillate around and through the target solution and so at some points in time may be close to the equilibrium but a short time later be very far from it. In order to account for this, training was continued for another 250,000 steps and evaluated after every 50,000 steps. The worst performing policy was then considered the value of the policy for that learning run.

Figure 6 shows the percentage of games won by the different players when playing their challengers. "WoLF" represents WoLF policy hill-climbing trained against itself. "PHC(L)" and "PHC(W)" represent policy hill-climbing with $\delta = \delta_l$ and $\delta = \delta_w$, respectively. "WoLF(2x)" represents WoLF policy hill-climbing trained with twice the training (i.e., two million steps). The performance of the policies were averaged over fifty training runs and the standard deviations are shown by the lines beside the bars. The relative ordering by performance is statistically significant.

WoLF-PHC does very well. Its learned policy is close to the equilibrium and continues to improve with more training. The exact effect of the adjustment

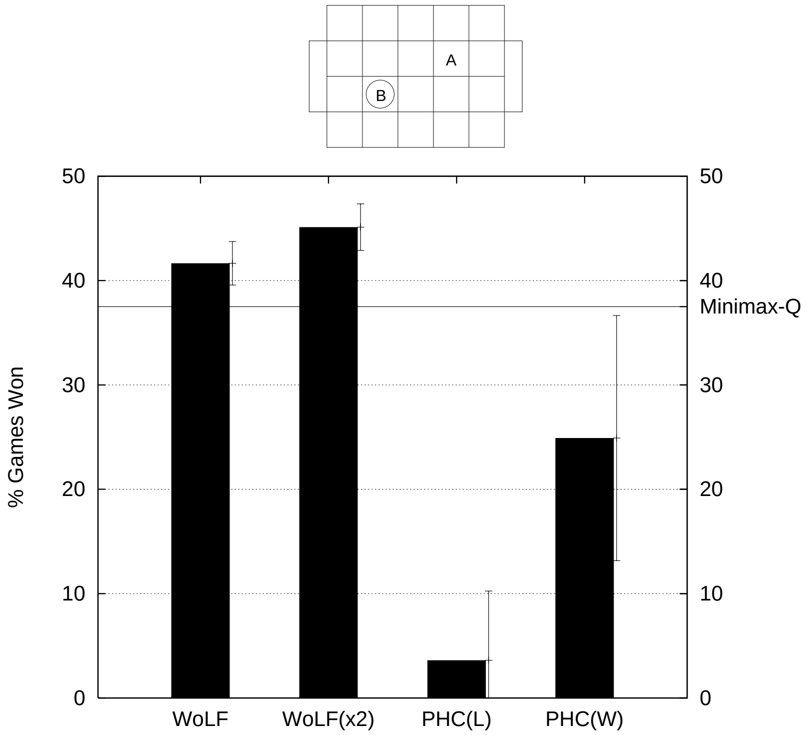


Fig. 6. Soccer game. The results show the percentage of games won against a specifically trained worst-case opponent after one million steps of training. The closer this percentage is to 50% the closer the learned policy is to the equilibrium. Error bars are shown and the relative ordering by performance is statistically significant. The reported performance of Minimax-Q (14) is shown by the solid line.

can be seen by its out-performance of PHC, using either the larger or smaller learning rate. This shows that the success of WoLF is not simply due to changing learning rates, but rather to changing the learning rate at the appropriate time to encourage convergence. Further training for PHC has statistically little effect on either the performance or variance, even up to three million steps of training. This suggests that the learned policies continue to oscillate wildly through the space of possible best-responses.

The speed of convergence of WoLF-PHC is approximately the same as the reported results for Minimax-Q (14). Specifically, Littman reports that in a similarly constructed experiment Minimax-Q with one million steps of self-play training won 37.5% of the games against its challenger. [7] A direct experimental comparison would be needed to make any stronger statements about the rates

[7] The results are not directly comparable due to the use of a different decay of the learning rate. Littman's experiments with Minimax-Q used an exponential decay which decreases too quickly for use with WoLF-PHC. Note that with infinite training Minimax-Q will provably converge to the equilibrium and win half its games versus its challenger. Our results show that WoLF also seems to be moving closer to the equilibrium with more training.

	H	T
H	+1, +1, −1	−1, −1, −1
T	−1, +1, +1	+1, −1, +1

	H	T
H	+1, −1, +1	−1, +1, +1
T	−1, −1, −1	+1, +1, −1

Table 7
Three-Player Matching Pennies. This game has three players selecting either *Heads* or *Tails*. The payoffs to each player are shown in the table, where the first player selects the row, the second selects the column, and the third selects the left or right table.

of convergence. Nevertheless it is compelling that WoLF-PHC is converging to the equilibrium with a speed at all comparable to a learner designed explicitly to learn this equilibrium (see Section 2.3.2).

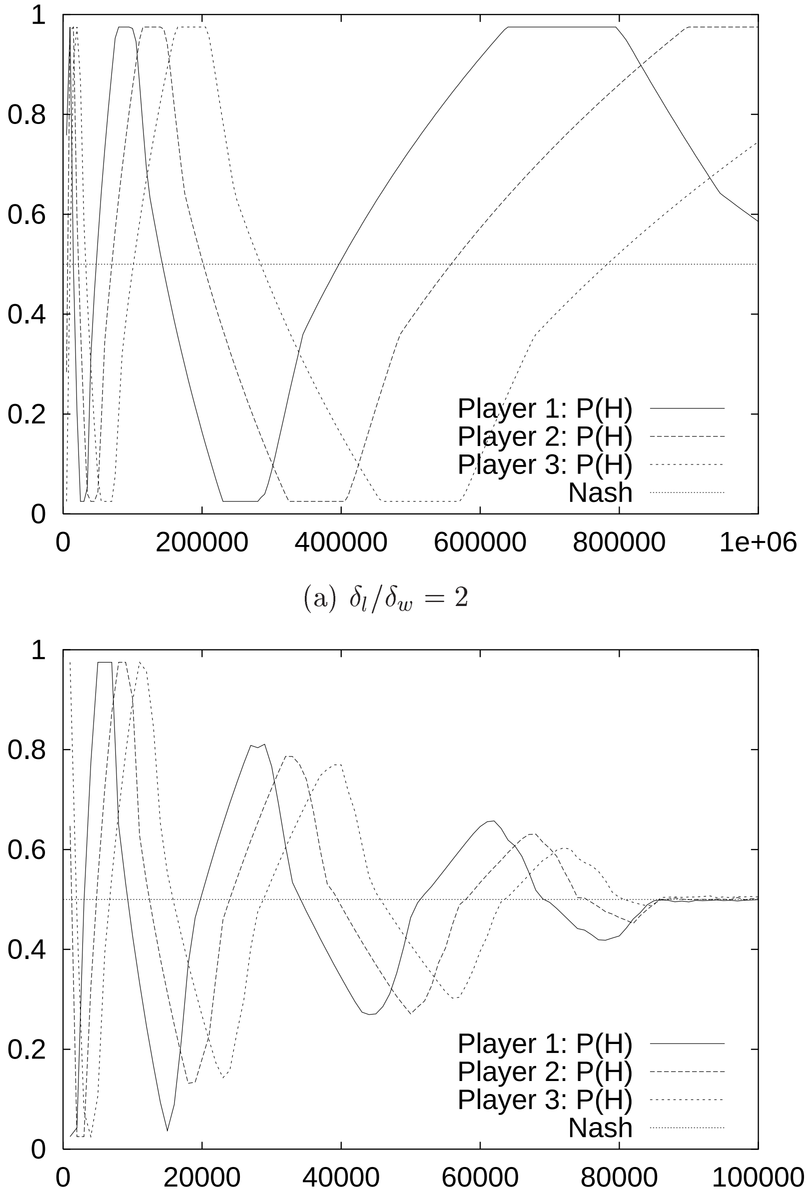
5.4 Three-Player Matching Pennies

The previous domains only involve two players. We have also examined WoLF in a multiple player matrix game, three-player matching pennies (33). In this game, each of the three players has two actions, *Heads* and *Tails*. Player 1, receives a payoff of 1 if his action matches Player 2, otherwise -1. Similarly Player 2 is reward for matching Player 3, and Player 3 is rewarded for selecting an action different from Player 1. Table 7 shows the complete set of payoffs to the players. This game has a single Nash equilibrium corresponding to the players randomizing equally between both actions.

Figure 7 shows the results of all three players using WoLF-PHC in this domain. The results are shown for two different learning rate ratios: $\delta_l/\delta_w = 2$ and $\delta_l/\delta_w = 3$. Notice that with the large ratio WoLF converges to the equilibrium just as in the other experiments, but with a lower ratio it does not converge. This is the first domain we have examined where the ratio of the two learning rates seems to be important to WoLF. This is likely due to the fact that it takes extra time for one player's strategy change to propagate through the extra player before affecting the player of interest's strategy. It is still very interesting that WoLF converges at all since another sophisticated technique, smooth fictitious play, fails to converge in this game (33).

5.5 Results Beyond Self-Play

The previous experiments have been examining WoLF-PHC in self-play, i.e., with all the players using the same algorithm. Corollary 2 (Section 3.4.3) gives a sliver of theoretical evidence that WoLF may learn well against other learners. In this section we present some results examining this empirically.



(a) $\delta_l/\delta_w = 2$

(b) $\delta_l/\delta_w = 3$

Fig. 7. Results of WoLF-PHC in self-play for the three-player matching pennies game. Each line corresponds to one player's strategy over time. (Notice the axes have different scales.)

5.5.1 Matrix Game

The first experiment examining WoLF in situations outside of self-play is the situation described in Corollary 2. We examine the learning of WoLF-PHC against unmodified PHC in rock-paper-scissors. Figure 8 shows the trajectories of the players. The corollary predicted convergence, and the proof gave the intuition that the convergence rate would depend on the ratio of the learning rates of both players. In this experiment, we have convergence to the Nash equilibrium, and the convergence is slower than with two WoLF learners (See Figure 4(b)) as the analysis predicted.

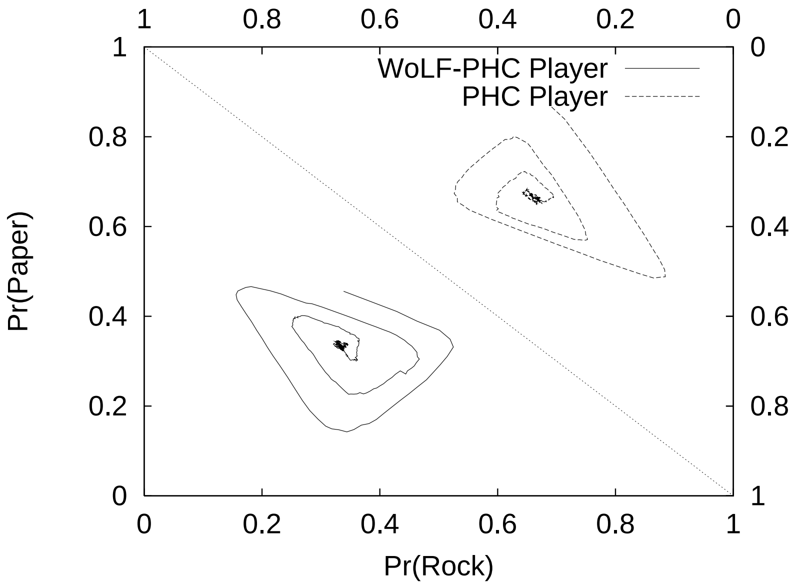


Fig. 8. Results of WoLF-PHC versus PHC in rock-paper-scissors: policy trajectories of the two players in a prototypical run with one million iterations.

5.5.2 Soccer

Since Littman's soccer game (see Figure 6) is the most complex game we have examined this far, we now consider whether WoLF might converge in this domain against different opponents. The first opponent is unmodified policy hill-climbing as above, and the second opponent is Q-Learning. Neither opponents are convergent in self-play, but are rational. So, if play is going to converge, it must be to a Nash equilibrium. The experimental setup is exactly as in Section 5.3. Training is performed for one million steps, and then the resulting policy is frozen and a challenger is trained to find that policy's worst case performance. As before, the closer the policy is to winning half of its games with its challenger, the closer the policy is to the equilibrium.

Figure 9 shows the results. As before we show the percentage of games won against its challenger as a measure of the distance from the equilibrium policy. The results are shown for both one million steps of training and twice that amount, both when training against unmodified PHC and against Q-learning. There are two important observations. The first is that the learned policy is comparatively close to the equilibrium (the result of PHC in self-play is also shown for comparison.) Second, it appears that more training does move the policy closer to the equilibrium giving evidence of convergence.

Of course none of these results proves convergence either in situations of self-

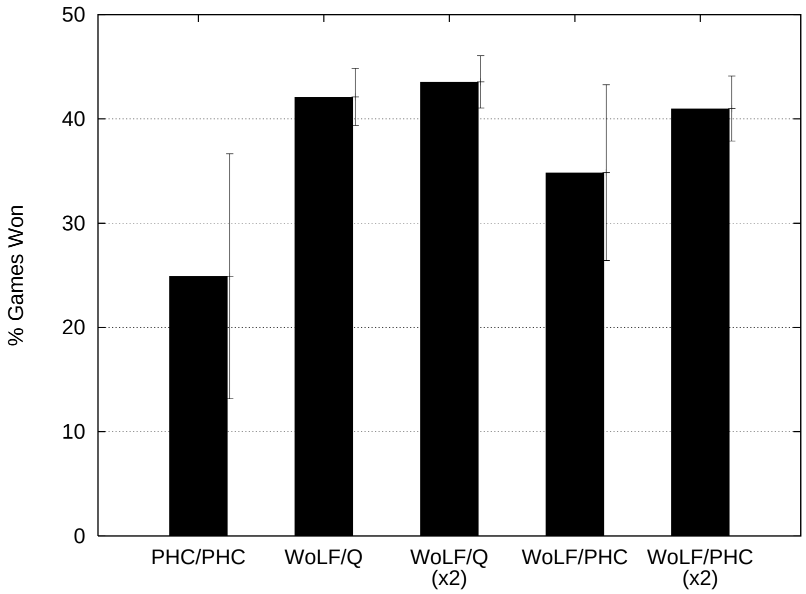


Fig. 9. Soccer game. The results show the percentage of games won against a specifically trained worst-case opponent non-identical opponent after one million steps of training. The closer this percentage is to 50% the closer the learned policy is to the equilibrium. Notice the added improvement with more training. PHC in self-play is included for comparison purposes.

play or otherwise. The results do give evidence that a variable learning rate and the WoLF principle can encourage convergence in an otherwise non-convergent rational learning algorithm. Specifically, WoLF-PHC has been shown in a variety of domains to effectively converge to best-response policies, despite the wild non-stationarity of other learning agents using same or different learning algorithms.

6 Conclusion

Multiagent learning is a powerful and needed technique for building multiagent systems. The framework of stochastic games helps provide a model for this learning problem and also illucidates the difficulties of attempting to learn an essentially moving target. Previous techniques do not adequately address these difficulties, and have one of two shortcomings: they are not rational, i.e., do not always play best-responses when they exist, or they do not necessarily converge at all.

We contribute the concept of learning with a variable learning rate that is shown to be able to overcome these shortcomings. We further contribute the WoLF principle to define how to vary the learning rate. By altering the learning rate according to the WoLF principle, a rational algorithm can be made convergent. This was proven for a restricted class of iterated matrix games, by modifying gradient ascent with a WoLF learning rate. It can also be used to modify more general rational learners. We presented a WoLF modification to policy hill-climbing, a rational stochastic game learning algorithm. We then demonstrated this algorithm empirically on a number of single-state, multiple-state, zero-sum, general-sum, two-player and multi-player stochastic games.

There are two interesting future directions of this research. The first is continuing to explore learning outside of self-play. We presented encouraging experimental evidence, but a number of issues remain. Specifically, whether WoLF techniques can be exploited by a malicious, probably not rational, "learner." This touches on the static optimality (or regret minimizing) property of online learning. A theoretical examination of other non-self-play situations would help to better understand these issues.

The second interesting direction is the common problem in reinforcement learning of making algorithms scale to large problems. The experiments presented in this article used explicit table representations of both value functions and policies. In problems with large state spaces, explicit table representations become intractable. It would be very interesting to combine single-agent scaling solutions, e.g., function approximators and parameterized policies, with the concepts of a variable learning rate and WoLF.

Appendix

The following were the actual learning and decay rates used for the results presented in this article. Here t is the number of the current iteration of step (2) of the algorithms. See the beginning of Section 5 for further explanation of these parameters. With the detailed algorithmic descriptions in Tables 5 and 6 and these parameter schedules, all of the presented results are reproducible.

- Matching Pennies (Figure 3)

  $\alpha(t) = \frac{1}{100+\frac{t}{10000}}$ $\qquad \delta = \delta_w(t) = \frac{1}{20000+t}$ $\qquad \delta_l(t) = 2\delta_w(t)$

- Rock-Paper-Scissors (Figures 4 and 8)

  $\alpha(t) = \frac{1}{10+\frac{t}{10000}}$ $\qquad \delta = \delta_w(t) = \frac{1}{20000+t}$ $\qquad \delta_l(t) = 2\delta_w(t)$

- Gridworld (Figure 5)

  $\alpha(t) = \frac{1}{1+\frac{t}{500}}$ $\qquad \delta = \delta_w(t) = \frac{1}{1000+\frac{t}{10}}$ $\qquad \delta_l(t) = 4\delta_w(t)$

- Soccer (Figures 6 and 9)

$\alpha(t) = \frac{1}{1+\frac{t}{500}}$ $\qquad \delta = \delta_w(t) = \frac{1}{1+\frac{t}{10}}$ $\qquad \delta_l(t) = 4\delta_w(t)$

- Three-Player Matching Pennies (Figure 7)

  $\alpha(t) = \frac{1}{10+\frac{t}{10000}}$ $\qquad \delta = \delta_w(t) = \frac{1}{100+t}$ $\qquad \delta_l(t) = 2\delta_w(t)$

References

[1] R. Bellman, Dynamic Programming, Princeton University Press, 1957.

[2] R. A. Howard, Dynamic Programming and Markov Processes, MIT Press, 1960.

[3] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: A survey, Journal of Artificial Intelligence Research 4 (1996) 237–285.

[4] R. S. Sutton, A. G. Barto, Reinforcement Learning, MIT Press, 1998.

[5] M. J. Osborne, A. Rubinstein, A Course in Game Theory, The MIT Press, 1994.

[6] G. Owen, Game Theory, Academic Press, 1995.

[7] J. F. Nash, Jr., Equilibrium points in n-person games, PNAS 36 (1950) 48–49, reprinted in (36).

[8] O. L. Mangasarian, H. Stone, Two-person nonzero-sum games and quadratic programming, Journal of Mathematical Analysis and Applications 9 (1964) 348–355.

[9] L. S. Shapley, Stochastic games, PNAS 39 (1953) 1095–1100, reprinted in (36).

[10] A. M. Fink, Equilibrium in a stochastic n-person game, Journal of Science in Hiroshima University, Series A-I 28 (1964) 89–93.

[11] O. J. Vrieze, Stochastic Games with Finite State and Action Spaces, no. 33, CWI Tracts, 1987.

[12] J. Filar, K. Vrieze, Competitive Markov Decision Processes, Springer Verlag, New York, 1997.

[13] M. Bowling, M. M. Veloso, An analysis of stochastic game theory for multiagent reinforcement learning, Technical report CMU-CS-00-165, Computer Science Department, Carnegie Mellon University (2000).

[14] M. L. Littman, Markov games as a framework for multi-agent reinforcement learning, in: Proceedings of the Eleventh International Conference on Machine Learning, Morgan Kaufman, 1994, pp. 157–163.

[15] J. Hu, M. P. Wellman, Multiagent reinforcement learning: Theoretical framework and an algorithm, in: Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufman, San Francisco, 1998, pp. 242–250.

[16] J. Robinson, An iterative method of solving a game, Annals of Mathematics 54 (1951) 296–301, reprinted in (36).

[17] W. Uther, M. Veloso, Adversarial reinforcement learning, Tech. rep., Carnegie Mellon University, unpublished (1997).

[18] C. Claus, C. Boutilier, The dyanmics of reinforcement learning in cooperative multiagent systems, in: Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI Press, Menlo Park, CA, 1998.

[19] M. Bowling, Convergence problems of general-sum multiagent reinforcement learning, in: Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufman, Stanford University, 2000, pp. 89–94.

[20] C. J. C. H. Watkins, Learning from delayed rewards, Ph.D. thesis, King's College, Cambridge, UK (1989).

[21] M. Tan, Multi-agent reinforcement learning: Independent vs. cooperative agents, in: Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, 1993, pp. 330–337.

[22] S. Sen, M. Sekaran, J. Hale, Learning to coordinate without sharing information, 1994.

[23] T. Jaakkola, S. P. Singh, M. I. Jordan, Reinforcement learning algorithm for partially observable markov decision problems, in: Advances in Neural Information Processing Systems 6, MIT Press, 1994.

[24] L. C. Baird, A. W. Moore, Gradient descent for general reinforcement learning, in: Advances in Neural Information Processing Systems 11, MIT Press, 1999.

[25] R. S. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in Neural Information Processing Systems 12, MIT Press, 2000.

[26] J. Baxter, P. L. Bartlett, Reinforcement learning in pomdp's via direct gradient ascent, in: Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufman, Stanford University, 2000, pp. 41–48.

[27] S. Singh, M. Kearns, Y. Mansour, Nash convergence of gradient dynamics in general-sum games, in: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufman, 2000, pp. 541–548.

[28] H. Reinhard, Differential Equations: Foundations and Applications, McGraw Hill Text, 1987.

[29] J. W. Weibull, Evolutionary Game Theory, The MIT Press, 1995.

[30] A. Blum, C. Burch, On-line learning and the metrical task system problem, in: Tenth Annual Conference on Computational Learning Theory, Nashville, TN, 1997.

[31] S. Singh, T. Jaakkola, M. L. Littman, C. Szepesvári, Convergence results for single-step on-policy reinforcement-learning algorithms, Machine Learning .

[32] J. Hu, Learning in dynamic noncooperative multiagent systems, Ph.D. thesis, Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI (1999).

[33] D. Fudenberg, D. K. Levine, The Theory of Learning in Games, The MIT Press, 1999.
[34] M. Bowling, M. Veloso, Rational and convergent learning in stochastic games, in: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, Seattle, WA, 2001, pp. 1021–1026.
[35] M. Bowling, M. Veloso, Variable learning rate and the convergence of gradient dynamics, in: Proceedings of the Eighteenth International Conference on Machine Learning, Williams College, 2001, pp. 27–34.
[36] H. W. Kuhn (Ed.), Classics in Game Theory, Princeton University Press, 1997.