Thoughts on Multiagent Learning: From A Reinforcement Learning Perspective

Lisa Jing Yan and Nick Cercone

Technical Report CSE-2010-07

November 2010

Department of Computer Science and Engineering
4700 Keele Street, Toronto, Ontario M3J 1P3 Canada

Thoughts on Multiagent Learning: From A Reinforcement Learning Perspective

Lisa J. YAN

August 30, 2010

Contents

Chapter 1

Artificial Intelligence Meets Game Theory

How to choose an optimal strategy to solve a problem is our interest. Let us start with a classic example: the Prisoner's Dilemma.

> Two suspects are arrested by the police. But, since the police have insufficient evidence for a conviction, prisoners are kept in separate room and are offered the same deal. If one defects from the other for the prosecution against the other and the other remains silent (cooperates with the other), the betrayer goes free and the silent accomplice receives the full 10-year sentence. If both remain silent, both prisoners are sentenced to only six months in jail for a minor charge. If each betrays the other, each receives a five-year sentence. Each prisoner must choose to betray the other or to remain silent. Note that each one is assured that the other would not know about the betrayal before the end of the investigation. How should the prisoners act?

The prisoner's dilemma is a fundamental problem in game theory that demonstrates why two persons might not cooperate even if it is in both of their best interests to do so. Game theory describes such a strategic situation mathematically, and Nash equilibria (Nash (1950)) famously captures this idea of how each individual in the game would behave and adopt a strategy that players are unlikely to change. This strategy is provided through game theory and known as a rational choice. Since how to find an optimal strategy has always been an interest in artificial intelligence (AI), this essense of game theory strongly influences AI researchers in the multiagent learning area.

1.1 Rationale

Multiple agents become increasingly required in various fields for both physical robots and software agents, such as, robot soccer, search and rescue robots, automated driving, auctions and electronic commerce agents, and so on. The merits of game theory influence computer science researchers in non-human-player game playing. An agent, non-human player, observes the environment and chooses an action to perform. Commonly, agents have goals, assumptions, algorithms for learning and reasoning, and conventions. Learning in single agent task has been studied extensively in the reinforcement learning field, in which an agent acts alone in a stationary environment. In multiagent domains, agents interact with others, and coadapt with others and act on the best choice available. Since all the agents are evolving, the environment is no longer stationary, and this brings in a difficult learning problem that violates the basic stationary assumption of traditional techniques for behavior learning. Each agent's choice of policy depends on the others' joint policy which also aims to achieve the best available performance. Our work focuses on the strategic decision making and learning process of agents' behaviors whose target is to select the best strategies and adapt to unforeseen difficulties and changes in the environment.

1.2 Objective

Our intention is to seek an answer to the following question:

> How can an agent efficiently observe other agents behaviors, and learn from its observation in order to act (or adapt) effectively in the complex nonstationary environment? Ultimately, through a learning period and a series of actions, this agent can achieve the top ranked performance, assuming that all agents pursue the same goal.

An agent can learn through experience which is from its own actions and associated effects, while learning from observation of other agents' experience. Note that an agent should effectively associate similar patterns and build knowledge instead of merely keeping a record of the rewards history for all the agents. By using this knowledge, the agent can eliminate part of the search space which might have already been explored by others. Exploration vs. exploitation is a critical choice in the agent learning process.

A complex nonstationary environment provides a dynamic learning domain. This domain is composed of other agents' diverse states. Thus, the complexity of the domain increases with the increase of the number of agents.

1.3 Outline

In this paper, we review seminal research on multiagent learning in general-sum games, from a reinforcement learning perspective. Chapter 2 introduces general research in computer science and game theory, also provides some general ideas of major approaches. Chapter 3 examines the framework of multiagent learning systems and related preliminary concepts. Chapter 4 studies the influential learning techniques in multiagent learning. We end in Chapter 5 with a summary of the main points of in this paper, and examine some open issues as our potential research objectives.

Chapter 2

General Multiagent Learning Approaches

Multiagent learning (MAL) has a long history in the game theory field, as well as in the machine learning community.

In machine learning, learning techniques are classified as supervised learning, unsupervised learning and reward-based learning, utilizing feedback directed to the learners. In multiagent learning, agents are given feedback about their behaviors as rewards or penalties in a given situation. Thus, reward-based methods are widely used in this field, including two major streams: reinforcement learning which estimates value functions, and evolutionary computation which directly learn behaviors using stochastic search methods. The similarities and differences between these two classes of learning methodology have generated a rich literature, and some address both classes, such as the bucket-brigade algorithm (Holland (1985)), the Samuel system (Grefenstette et al. (1990)), and the recent Stochastic Direct Reinforcement policy gradient algorithm (Moody et al. (2004)). In this paper, we focus on examining multiagent learning techniques from a reinforcement learning (RL) perspective.

Reinforcement learning (RL) methods are widely used where rewards and penalties are given after a sequence of actions are performed in the environment. The learning process is through defined formulas to update the expected utilities, and based on the expected utility values to choose the most promising action to explore the state space. Two common RL methods are Q-learning (Watkins (1989)) and Temporal-Difference (TD(λ)) learning (Sutton (1988)). Q-learning learns the utility of performing actions in states, also called an off-policy TD control algorithm; while TD-learning learns the utility of being in the states. The difference between on-policy and off-policy is that: the on-policy is aimed to control; while off-policy can be used in control and predict, and it may learn some tactics which may not be exhibited during the learning phase. (See Section 3.1 for details).

In the MAL literature, the AI community extends Bellman-style single-agent reinforcement learning techniques to a multiagent setting, in particular Q-learning (Watkins and Dayan (1992)). This technique has performed well in: a) zero-sum repeated games (Littman (1994); Littman and Szepesvri (1996)), b) common-pay-off (or team) repeated games (Claus and Boutilier (1998); Kapetanakis and Kudenko (2004); Wang and Sandholm (2002);), but not so well in c) general-sum stochastic games(Hu and Wellman (1998); Littman (2001); Greenwald and Hall (2003)).

In this chapter, we give a general introduction of learning approaches in the field, according to a specific game (known or unknown), or play (observable or unobservable). The literature divides into three major classes of techniques: one is the model-based approach, one is the model-free approach, and the other approach is no-regret learning.

2.1 Model-based Approach

The model-based approach is under the scenario of fully observable games. This approach is, for an agent, firstly to learn from the opponents' strategies as a model, and then accordingly devise a best (or "optimal") response. In other words, the agent learns about the reward function and the state transition function, thereafter, solves for its own optimal policy. The best-known work of this approach is fictitious play (Brown (1951)). The general scheme is as follows:

Step 1 Start with model of the opponent's strategy;

Step 2 Repeat:

- Compute and play the best response;
- Observe opponent's play and update our model of the player's strategy;

2.2 Model-free Approach

Model-free approaches are commonly used in the RL community (Kaelbling et al. (1996)) which avoids building an explicit model of the opponent's strategy. Instead, the agent learns how well its own various possible actions fare over time. Without knowing the reward function R or the transition function T, the agent can directly learn about its optimal policy through observing other agents' actions and immediate payoff. This work has been explored intensively in the reinforcement learning area, and the roots are Bellman equations (Bellman (1957)).

Q-learning (Watkins and Dayan (1992)) is a typical form of model-free learning approach. The learning process is similar to Sutton's (Sutton (1988)) temporal difference (TD): an agent tries an action at a particular state, and evaluates its consequences in terms of the immediate reward or penalty the agent receives and its estimate of the value of the state of which the action is taken. By trying all actions in all states repeatedly, an agent learns which are best overall, based on long-term discounted rewards. We examine the details of Q-learning techniques in the next chapter.

2.3 No-regret Learning

No-regret learning aims to seek to minimize regret, also known as regret minimization approaches. Two important criteria of the learning rules are *safe* and *consistent* (Fudenberg and Levine (1995)). No regret algorithms have been mainly explored in single state games, and little work has been done in application in stochastic games caused by the difficulties of extending this concept to stochastic games (Mannor and Shimkin (2003)). The more detailed disscussion is given in Section 4.4.4.

2.4 Summary

In this chapter, we present general approaches for multiagent learning in the field of reinforcement learning. According to the available game information, three major classes of techniques have been explored: model-based approaches, model-free approaches and no-regret learning. In chapter 3 and chapter 4, we examine details for seminal approaches in multiagent learning on how to find optimal policies.

Chapter 3

Multiagent Learning Framework and Preliminaries

A multiagent system (MAS) has a broad set of definitions, of which each definition leads to different constraints to solve the computational complexity of MAS. We focus on the machine learning field; and the goal of machine learning is to build intelligent programs which can solve problems after a learning and evolving process. This intelligent program is often called a "agent".

An agent is a computational application that is designed to automate certain tasks, with a guiding intelligence to achieve a result. A multiagent environment is one in which more than one agent acts and interacts with one another. Moreover, agents may or may not know everything about the environment. An agent learns by interacting in its environment and by observing the effect of these interactions. This learning, while performing in the environment, is typical. The idea is commonly known as "cause and effect", and this undoubtedly is the key to accumulating experience and forming knowledge through performance.

Before we get into the learning process for multiple agents, we first examine how a single agent learns and evolves in a certain environment. Thereafter, we study the framework for multiagent learning systems.

Reinforcement learning is one approach to expedite the agent learning process, especially in solving two complex problems: game playing and control problems. Each agent learns through the reinforcement (rewards or penalties) from its environment, and this learning process for an agent can be seen as a self-teaching process. On the other hand, each agent learns to perform the best in the environment, this is also an adaptation process.

3.1 Single Agent Learning

One interesting problem arising along with this agent reinforcement learning process is that the trade-off between exploration and exploitation. Once an agent learns a certain action which has performed well, should an agent exploit this action since it is known to receive a decent reward? Or should it explore other possibilities in order to seek a better reward? Obviously, exploring is definitely a good tactic sometimes, but without a balance between exploration and exploitation, agents will not learn successfully. The common way to achieve a good balance is to try a variety of actions while progressively favoring those producing the most reward.

In this section, we examine the most influential work in RL(Sutton and Barto (1998)): temporal difference learning and Q-learning.

3.1.1 Markov Decision Process

An agent learning process can be separated into the following steps:

- Observe the surrounding environment;
- Decide an action (or "strategy") according to certain criteria;
- Perform the action;
- Agent receives feedback, rewards or penalty, from the environment;
- Information about experience is recorded. In details, the experience includes the environment situation, the action chosen, and the feedback received.

Eventually, an agent can learn an optimal decision policy which performs the best in a certain environment, by performing actions and evaluating the results related. Markov decision processes are the foundation for research in single agent learning. A Markov decision process (MDP) (Sutton and Barto (1998);Bellman (1957)) is a tuple, (S, A, T, R), where,

- S is the set of the states;
- A is the set of actions;
- $T : S \times A \times S \to [0, 1]$ is a transition function, which defines a probability distribution over next states as a function of the current state and the agent's action:

$$\forall s \in S, \forall a \in A, \sum_{s' \in S} T(s, a, s') = 1;$$

- $R : S \times A \rightarrow \mathbb{R}$ is a reward function, which defines the reward received when selecting an action from the given state.

At time t, the agent receives the reward $r^t = R(s^t, a^t)$, and the agent observes a new state s^{t+1}, which is drawn from the probability distribution specified by $T(s^t, a^t, s^{t+1})$.

In general, the transition function T and the reward function R are not known in advance. Thus, the goal of a learning agent in an MDP is to learn a policy π to maximize its long-term reward R based on the only samples received. A policy π is defined to map the probability of selecting an action from a particular state. Formally, $\pi \in S \times A \rightarrow [0, 1]$, where $\forall s \in S, \sum_{a \in A} \pi(s, a) = 1$.

Two common ways to formulate the long-term reward are the discounted reward function and the average reward function. Define $V^\pi(s)$ as a policy's state value function, and $E(r^t \mid s^0 = s, \pi)$ as the expected reward received at time t given the initial state s and the agent follows the policy π. The average reward is formed as:

$$V^\pi(s) = \lim_{T \rightarrow \infty} \sum_{t=0}^{T} \frac{1}{T} E(r^t \mid s^0 = s, \pi), \tag{3.1}$$

which is under a common assumption that the MDP is a unichain. The unichain assumption is that the Markov chain induced by every stationary policy (perhaps randomized) has only one ergodic class of states and, perhaps, some transient states.[1]

The discounted reward is described as follows:

$$V^\pi(s) = \sum_{t=0}^{\infty} \gamma^t E(r^t \mid s^0 = s, \pi), \gamma \in [0, 1). \tag{3.2}$$

γ is a discount factor, which accumulates the immediate reward with probability γ instead of a larger future utility. Temporal difference learning describes a class of algorithms that adopt this discounted reward formulation. We discuss more details in Section 3.1.2.

The Markov decision process is under the Markov assumption, which generally, requires that the next state and reward to the agent depend only on the current state and agent's action. Formally, we state this property of MDP as follows.

Definition 1 *A decision process is Markovian if and only if, the sequence of states ($s_t \in S$), actions ($a_t \in A$), and the rewards ($r_i^t \in \mathbb{R}$), satisfies*

$$Pr\{s^t = s, r_i^t = r_i \mid s^{t-1}, a^{t-1}, \ldots, s^0, a^0\} = Pr\{s^t = s, r_i^t = r_i \mid s^{t-1}, a^{t-1}\}.$$

[1]An MDP is unichain if and only if, for all policies, there exists an ergodic set of states (i.e. any state in the set can be reached with non-zero probability from any other state in the set), and all states outside this set are transient (i.e. after some finite point in time it will never be visited again).

An agent's selection of actions is Markovian if and only if,

$$Pr\{a^t = a \mid s^t, s^{t-1}, a^{t-1}, \ldots, s^0, a^0\} = Pr\{a^t = a \mid s^t\};$$

that is, only if the agent's next action depends only on the current state.[2]

We also refer to a Markovian process as stationary, and in the multiagent framework of stochastic games, this property does not hold in a non-stationary environment.

3.1.2 Temporal Difference Learning

Temporal difference (TD) learning is a combination of Monte Carlo ideas and dynamic programming (DP) ideas, and is commonly used for prediction problems and control problems. Both TD and Monte Carlo methods use experience to solve the prediction problem. Given some experience following a policy π, the learning procedure updates its estimate V of V^π. If a nonterminal state s^t is visited at time t, then it is to update the estimate $V(s^t)$ based on what happens after that visit. Once the actual reward R^t at time t is received, a simple every-visit Monte Carlo method suitable for nonstationary environment is

$$V(s^t) \leftarrow V(s^t) + \alpha[R^t - V(s^t)], \tag{3.3}$$

where α is a constant step-size parameter, also known as the learning rate. Monte Carlo methods must wait until the end of the episode to determine the increment to $V(s^t)$, since only then R^t is known. Whereas TD methods need wait only until next time step. At time $t+1$ they immediately form a target and make a useful update using the observed reward r^{t+1} and the estimate $V(s^{t+1})$. The simplest TD method, known as TD(0), is

$$V(s^t) \leftarrow V(s^t) + \alpha[r^{t+1} + \gamma V(s^{t+1}) - V(s^t)]. \tag{3.4}$$

In comparison, the target for the Monte Carlo update is R^t, whereas the update for TD is $r^{t+1} + \gamma V(s^{t+1})$. Here, γ is a discount parameter. The TD method is a bootstrapping method as DP, since the update is based on an existing estimate, not a final reward.

On-policy TD methods learn the value of the policy that is used to make decisions. The value function is updated using results from executing actions determined by some policy. Off-policy learning can learn different policies for behaviors and for

[2]Definition 1, 2, 3, 4, 5, 6, are adopted from the formulation presented in Bowling (2003), and Definition 1, 6 are based on Sutton and Barto (1998), Bellman (1957).

estimation. The update is estimated using hypothetical actions, which have not actually been executed. In contrast to on-policy methods strictly based on experience, off-policy can separate exploration from control, but on-policy algorithms can not.

There are three common policies used for action selection in order to balance the trade-off between exploration and exploitation, and it is not clear which policy produces the best overall results.

- ϵ-greedy: most of the time, with a high probability $1-\epsilon$, the action with the highest estimated reward is chosen, called the greediest action. Every once in a while, say with a small probability ϵ, an action is selected at random. The action is selected uniformly, independent of the action-value estimates. This method ensures that if enough episodes are considered, each action will be tried to ensure optimal actions are discovered.

- ϵ-soft: very similar to ϵ-greedy, the best action is selected with small probability ϵ; while the rest of the time, with high probability $1-\epsilon$, a random action is chosen uniformly.

- softmax: one drawback of ϵ-greedy and ϵ-soft is that they select random actions uniformly. The worst possible action is just as likely to be selected as the second best action. Softmax remedies this case by assigning a random or weight to each of the actions, according to their action-value estimate. A random action is selected with regard to the weight associated with each action, meaning the worst actions are unlikely to be chosen. The most common softmax uses a Gibbs or Boltzmann distribution.

TD(λ)

In TD(λ) algorithms, λ refers to the use of an eligibility trace. Eligibility traces are one of the basic mechanisms of reinforcement learning, and often can accelerate TD learning. When the TD algorithm, described in Section 3.1.2, receives input (y^{t+1}, x^{t+1}), it updates only for the immediately preceding signal x^t. That is, the algorithm modifies only the immediately preceding prediction, here $\lambda = 0$. But since y^{t+1} provides useful information for learning earlier predictions as well, one can extend TD learning so it updates a collection of many earlier predictions at each step, $0 \leq \lambda \leq 1$. Eligibility traces do this by providing a short-term memory of many previous input signals so that each new observation can update the parameters related to these signals. Eligibility traces are usually implemented by an exponentially-decaying memory trace, as decay parameter λ. This generates a family of TD algorithms TD(λ), $0 \leq \lambda \leq 1$, with TD(0) corresponding to updating only the immediately preceding prediction as described in Eq. 3.4, and TD(1) corresponding to equally updating all the preceding predictions. This also applies to non-lookup-table versions of TD learning, where traces of the components of the input vectors are maintained. Eligibility traces do not have to be exponentially-decaying traces, but these are usually used since they are relatively easy to implement and to understand theoretically.

3.1.3 Q-learning

Q-learning is the most significant breakthrough as an off-policy TD control algorithm. The simplest, one-step Q-learning is defined as follows:

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha[r^{t+1} + \gamma \max_a Q(s^{t+1}, a) - Q(s^t, a^t)], \tag{3.5}$$

where α is the learning rate, $0 < \alpha < 1$; When α is set to 0, it means that the Q-value is never updated and nothing is learnt; while α is set to 0.9, it means that learning can occur quickly. $Q(s^t, a^t)$ is the expected value of performing action a in state s; and $\max_a Q(s, a)$ is the maximum reward received and then follows the optimal policy. The Q-learning algorithm is shown in Alg. 1.

Algorithm 1: Q-learning: An off-policy TD control algorithm

```
Initialize $Q(s, a)$ arbitrarily;
repeat for each episode:
    Initialize $s$;
    repeat for each step of episode:
        Choose $a$ from $s$ using policy derived from $Q$;
        Take action $a$, observe $r$, $s'$;
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$;
        $s \leftarrow s'$;
    until $s$ is terminal;
until;
```

3.2 General Framework of MAL

In a multiagent learning framework, agents process three classes of activities: perception, reasoning and action (see Fig. 3.1). First, each agent observes other agents and collect information in the environment, called "perception". Second, agents conduct reasoning according to their own preference and knowledge to decide an optimal strategy; thereafter, agents perform their actions and receive feedback respectively.

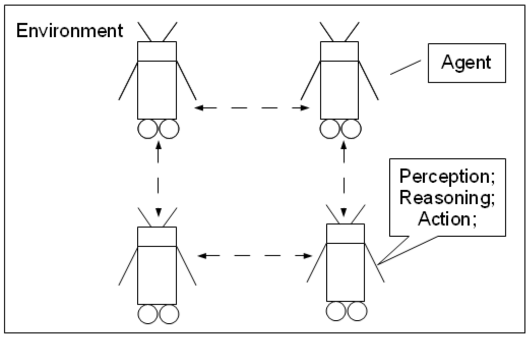

Figure 3.1: Multiagent Framework

Stochastic games are defined as multiple agents with a multiple states framework, which can be viewed as a synthesis of Markov decision processes and matrix games. MDPs model a single agent, multiple states model, which have been explored prominently in the field of reinforcement learning (see Section 3.1.1). On the other hand, matrix games describe a multiagent system with single state model, which are the foundational concepts in the game theory field. Since stochastic games share concepts with these two simpler frameworks, it is useful to consider them independently to analyze the core concepts while addressing the critical issues existing in stochastic games only. Fig. 3.2 illustrates the relations among these three concepts. In Section 3.1, we discuss MDP as a single agent reinforcement learning; then, we examine matrix games, a multiagent, single-state learning process.

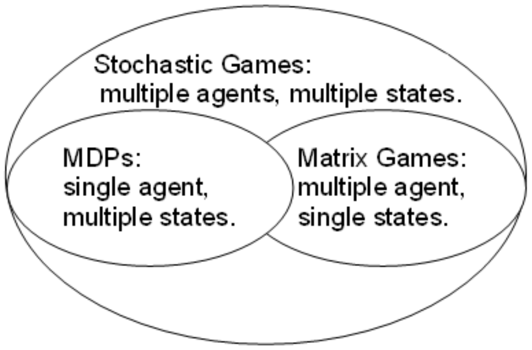

Figure 3.2: Stochastic Games = MDPs + Matrix Games

3.2.1 Matrix Games

Matrix games were first examined in the field of game theory to model strategic interactions of many decision makers (von Neumann and Morgenstern (1944); Osborne and Rubinstein (1994)). Mathematically, a matrix game (or strategic game) is a tuple (A, R), where $A = A_i \times \cdots \times A_n$ is the action space for each player; player i chooses an action A_i, and receives the payoff R_i, $i \in [1, n]$, which depends on all the players' actions. R is normally written as n-dimensional matrices, and each entry in the reward matrices corresponds to the joint actions taken. The learning process in matrix games means that agents repeatedly play the same matrix game, which is also called a repeated game. Agents learn through experience from observation of other agents' behaviors and their rewards, to maximize its own reward.

Examples

As follows, we list several matrix games and the payoff function matrices. Note that R_1 is for player 1 and R_2 is for player 2. In each game matrix, the row represents player 1, and the column represents player 2.

- **(a) Rock-Paper-Scissors**
  Two players with each having three options: "Rock", "Paper" and "Scissors", and the rules are: "Rock" loses to "Paper", "Paper" loses to "Scissors", and "Scissors" loses to "Rock"; otherwise, it is a tie. The winner gains one dollar from the loser, while the loser loses one dollar. For example, player 1 plays P while player 2 plays S, and the reward is -1 for player 1 and 1 for player 2.

$$R_1 = \begin{bmatrix} & R & P & S \\ R & 0 & -1 & 1 \\ P & 1 & 0 & -1 \\ S & -1 & 1 & 0 \end{bmatrix}, \; R_2 = \begin{bmatrix} & R & P & S \\ R & 0 & 1 & -1 \\ P & -1 & 0 & 1 \\ S & 1 & -1 & 0 \end{bmatrix}$$

- **(b) Coordination Game**
  Two players simply both desire to agree on their action choice, but with no preferences between them.

$$R_1 = \begin{bmatrix} & A & B \\ A & 1 & 0 \\ B & 0 & 1 \end{bmatrix}, \; R_2 = \begin{bmatrix} & A & B \\ A & 1 & 0 \\ B & 0 & 1 \end{bmatrix}$$

- **(c) Stackelberg Stage Game**
  The players of this game are a leader and a follower and they compete on re-

ward quantity; the leader moves first and then the follower moves sequentially.

$$R_1 = \begin{bmatrix} & \mathit{Left} & \mathit{Right} \\ \mathit{Up} & 1 & 3 \\ \mathit{Down} & 2 & 4 \end{bmatrix}, \; R_2 = \begin{bmatrix} & \mathit{Left} & \mathit{Right} \\ \mathit{Up} & 0 & 2 \\ \mathit{Down} & 1 & 0 \end{bmatrix}$$

Matrix games can be classified according to their payoff function. If one agent's gain is other agents' loss, we call this type of game as *general-sum games*. For example game (a), the sum of player 1's gain and player 2's loss equals zero, we also call this *zero-sum game*. Another common type of matrix game is *team game*, i.e. game (b), in which all agents have the same payoff function, in other words, one agent's best interest is the best interest of all others. Game (c) looks similar to the general-sum game and team game, but it is neither of them.

What we can learn in game (c) is as follows: imagine a repeated version of this game, and assume that the column player (secondary player: follower) is paying attention to the row player's (first player: leader) strategy and the rewards after each move. The two players will end up in a repeated (Down, Left) play and (Up, Right) play, since this is a way that benefits both. We conclude from this example: that learning and teaching happens at the same time: the row player has taught the column player to play in a way that benefits both most. Or, we can see this as an adaptation rather than a learning process. Note that the concept of strategy is not the same as a move. A move refers to an action taken by a player at the certain point during the game; while a strategy means a complete algorithm for playing the game which then tells a player what to do throughout the game.

Nevertheless, the learning agent's goal is to learn a strategy that maximizes its reward, using either pure strategies or mixed strategies. A pure strategy provides a complete set of how a player plays a game; while a mixed strategy is a probability of each pure strategy. An arbitrary finite matrix game may not have a pure strategy Nash equilibrium, but it always has a mixed strategy Nash equilibrium(Nash (1951)). Therefore, in our research, we focus on mixed strategies, and the definition is given as below.

A *mixed strategy* refers to a joint strategy σ for all n players. One player i's strategy σ_i, specifies a probability distribution over all actions A, and its reward function R_i is defined over mixed strategy as follows:

$$R_i(\sigma) = \sum_{a \in A} R_i(a) \Pi_{i=1}^{n} \sigma_i(a). \tag{3.6}$$

$R_i(a)$ is the reward received by player i when playing action a, and $\sigma_i(a)$ is the probability distribution of playing action a.

In matrix games, one player's optimal strategy can only be evaluated if the other players' strategies are known. So, this is an opponent-dependent solution, also called *best-response*. We use $< \sigma_i, \sigma_{-i} >$ to represent the joint strategy where player i follows σ_i while others follow σ_{-i}. σ_{-i} refers to a joint strategy for all the players except player i.

Definition 2 *For a matrix game, the best-response function for player i, $BR_i(\sigma_{-i})$, is the set of all strategies that are optimal given the other player(s) play the joint strategy σ_{-i}. Formally, $\sigma_i^\star \in BR_i(\sigma_{-i})$, if and only if,*

$$\forall \sigma_i \in PD(A_i) \quad R_i(< \sigma_i^\star, \sigma_{-i} >) \geq R_i(< \sigma_i, \sigma_{-i} >)$$

where $PD(A_i)$ is the set of all probability distributions over the set A_i (the set of all mixed strategies for player i).[3]

One most critical notion in matrix game and game theory is a best-response equilibrium, also called *Nash Equilibrium*(Nash (1950)).

Definition 3 *A Nash equilibrium is a collection of strategies for all players, σ_i, with*

$$\sigma_i \in BR_i(\sigma_{-i}).$$

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

All matrix games have a Nash equilibrium, and there may be more than one. In *zero-sum* games, one appealing feature is that there is a unique Nash equilibrium, and this equilibrium corresponds to the games' *minmax* solution. In other words, this mixed strategy maximizes the worst-case expected reward. This solution can be found in a linear program as illustrated in Eq. 3.7.

$$\begin{array}{rc} \text{Maximize:} & \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \sigma(a_1) R(< a_1, a_2 >), \\ \text{Subject to:} & \sum_{a_1 \in A_1} \sigma(a_1) = 1, \\ & \sigma(a_1) \geq 0, \quad \forall a_1 \in A_1. \end{array} \tag{3.7}$$

This solution is player 1's equilibrium strategy, where this linear program has $\|A_1\|$ parameters. Player 2's strategy can be solved similarly. In Rock-paper-scissors game, there is a unique Nash equilibrium in which each player selects their actions with equal probability (as mixed strategy Nash equilibrium). But, if one player simply adopts this equilibrium strategy, will the player win the competition of a tournament? The answer is no, because a Nash equilibrium provides a rational strategy,

[3]Defnition 2, 3, 4, 5 are based on Nash (1950).

not necessary a best benefit one. Furthermore, in a general matrix game, finding a Nash equilibrium is known to be NP-hard, yet is still an open question(Gilboa and Zemel (1988); Conitzer and Sandholm (2008)).

3.2.2 Stochastic Games

Stochastic games are an extension of a combination of matrix games and MDPs, which include multiple agents with multiple stages. Formally, a stochastic game (Shapley (1953))can be represented as a tuple: (n, S, A, T, R), where

- n is the number of agents;
- S is a set of stages;
- A is a set of actions, $A = A_1, \cdots, A_n$; A_i is player i's action. (We assume that each player has the same strategy space in all games. This is a notational convenience, not a substantive restriction.)
- T is a transition function specifying the probability of the next stage game to be played based on the game just played and the action taken in it: $S \times A \times S \to [0, 1]$, such that,
$$\forall s \in S, \forall a \in A, \quad \sum_{s' \in S} T(s, a, s') = 1.$$
- R is the reward function, $R = R_1, \cdots, R_n$. R_i is the immediate reward function of player i for at the stage S: $S \times A \to R$. Note that each player has its own independent reward function.

When $n = 1$, stochastic games are MDPs; when $\|S\| = 1$, they are matrix games or repeated games. The goal for player i in a stochastic game is to learn a policy that maximizes long-term reward, same as MDPs. A policy for player i, π_i is a mapping that defines the probability of selecting an action from a particular stage. Formally, $\pi_i \in S \times A \to [0, 1]$, where

$$\forall s \in S, \quad \sum_{a \in A} \pi_i(s, a) = 1.$$

We use π to refer to a joint policy for all the players, and Π_i refers to the set of all possible stochastic policies available to player i, while $\Pi = \Pi_1 \times \cdots \times \Pi_n$ refers to the set of joint policies for all the players. π_{-i} refer to a particular joint policy of all the players except player i, and Π_{-i} refers to the set of such joint policies. Finally, the notion $< \pi_i, \pi_{-i} >$ refers to the joint policy where player i follows π_i while the other players follow their policy from π_{-i}.

Next, similar to MDPs, we need to define how to aggregate the set of the immediate rewards received in each stage for each agent in order to quantify the value of a policy. For finitely repeated games, we can simply use the sum or average reward which is the typical approach. For infinitely repeated games, the most common approaches are to use either the limit average or the sum of discounted rewards. The limit average reward function V of player i in stochastic games is defined similarly to MDPs, as follows,

$$V_i^{\pi}(s) = \lim_{T\to\infty} \sum_{t=0}^{T} \frac{1}{T} E(r_i^t \mid s^0 = s, \pi) \tag{3.8}$$

where $E(r_i^t \mid s^0 = s, \pi)$ as the expected reward to player i received at time t given the initial state s and the agents follow the policy π. Similarly, the sum of discounted award function is defined with discount factor $\gamma, \gamma \in [0, 1)$, as,

$$V_i^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t E(r_i^t \mid s^0 = s, \pi). \tag{3.9}$$

Notice that this reward function for each agent i is dependent on the joint policy of the other agents. As in MDPs, we can also define Q-values for a given agent for a particular joint policy. For the discounted reward framework, Q-values can be formulated as,

$$Q_i^{\pi}(s, a) = R_i(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V_i^{\pi}(s').$$

On the other hand, similar to matrix games, there is a best-response in stochastic games. Notice that a policy for a player can only be evaluated in the context of all the players' policies.

Definition 4 *For a stochastic game, the best-response function for player i, $BR_i(\pi_{-i})$, is the set of all policies that are optimal given the other player(s) play the joint policy π_{-i}. Formally, $\pi_i^{\star} \in BR_i(\pi_{-i})$, if and only if,*

$$\forall \pi_i \in \Pi_i, \;\; \forall s \in S, \quad V_i^{<\pi_i^{\star}, \pi_{-i}>}(s) \geq V_i^{<\pi_i, \pi_{-i}>}(s)$$

where $PD(A_i)$ is the set of all probability distributions over the set A_i (the set of all mixed strategies for player i).

We can also define the most critical notion: a best-response equilibrium or *Nash Equilibrium*, similar to matrix games in game theory.

Definition 5 *For a stochastic game, a Nash equilibrium is a collection of policies, one for each player, π_i, such that,*

$$\pi_i \in BR_i(\pi_{-i}).$$

Therefore, no player can do better by changing policies given that the other players continue to follow the equilibrium policy.

Stochastic games can be classified the same way as matrix games. Team games are the ones where all the agents receive the same reward function. General-sum games are the ones where one player's gain means other players' loss. Zero-sum games refer to the sum of total rewards equals to zero. Like matrix games, zero-sum stochastic games have a unique Nash equilibrium, and we examine some seminal learning techniques to solving such stochastic games in the next chapter.

In stochastic games, the Markov assumption still holds, but it has a different form, as follows.

Definition 6 *A multiagent decision problem is Markovian if and only if, the sequence of states ($s^t \in S$), actions ($a^t \in A$), and the rewards ($r_i^t \in R$), satisfies*

$$Pr\{s^t = s, r_i^t = r_i \mid s^{t-1}, a^{t-1}, \dots, s^0, a^0\} = Pr\{s^t = s, r_i^t = r_i \mid s^{t-1}, a^{t-1}\}.$$

that is, if the next state and rewards depend only on the previous state and all of the agents' actions, but not on the history of states and actions.

From the game's perspective, stochastic games are Markovian, but from a single agent's perspective, the process is no longer stationary or Markovian (versus "behavior strategy"[4]). It is because the transition probabilities associated with a single agent's action from a state are not stationary and change over time as the other agents' action choices change. This property is critical to single-agent reinforcement learning research, and this violation of basic assumptions require new techniques to be developed to learn effective policies in stochastic games.

3.3 Summary

In this chapter, we described the single agent learning process and examine most critical techniques TD learning and Q-learning in the reinforcement learning field. Thereafter, we introduce MDPs and matrix games, since stochastic games can be seen as a merging of both. Through detailed analysis of MDPs and matrix games, we

[4]A behavior strategy is defined if $\pi_t = f(h_t)$ where h_t is the history up to time t; a makovian or stationary strategy is a special case of behavior strategy when $h_t = \phi$.

present the general framework for multiagent learning, and some important concepts in stochastic games and in game theory. In the next chapter, we examine seminal learning techniques for finding solutions in stochastic games.

Chapter 4

Seminal Learning Techniques

Autonomous agents are the agents who can sense the environment, act on it, and pursue their own agenda. These autonomous agents include intelligent agents, autonomous robots, even artificial life agents, and many computer viruses. This research involves a spectrum of areas, including reinforcement learning, evolutionary computation, game theory, complex systems, agent modeling and robotics. From the learning task perspective, this research leads to two learning branches: team learning and concurrent learning. Team learning applies to a single learner, to discover joint solutions for multiagents problems; concurrent learning (also called distributed learning) states that multiple agents learn simultaneously. With the process of learning, agents communicate directly or indirectly. Our interest focuses on the dynamic learning process appearing in the multiagent system.

Many dynamic learning studies come from the game theory perspective. An important concept in game theory is the Nash equilibrium which provides a joint overall strategy for learners. As the learners do not normally have control over others, no single agent has any rational incentive to change its strategy away from the equilibrium. Thus, many dynamic learning methods converge to Nash equilibria. This paper analyzes multiagent learning problems starting from stochastic games, which focus on finding a Nash equilibrium. In this chapter, we first present the first algorithm proposed to find equilibria in game theory. Thereafter, we examine seminal learning approaches about how to learn a Nash equilibrium or find the best responses in stochastic games.

4.1 Dynamic Learning

The multiagent learning process is a dynamic learning process, rather than a stationary learning process in which a single-agent explores the static environment and discovers a globally optimal behavior. In a dynamic environment, all agents are constantly alert for environment changes and adapt their own optimal behaviors to each other's learning process. That is, each learner coadapts its behaviors in the context of others; accordingly this co-adaptation brings in a complicated dynamic environment for each learner to act in it with others. Thus, in dynamic learning, since each agent is free to learn and act separately towards achieving its goal; an inherent critical problem is how to assign credit to each agent after reward obtained at a team level. In this section, we discuss credit assignment and dynamic environment.

4.1.1 Credit Assignment

For a task involving multiple agents' joint actions, the simple solution called *global reward* is to divide the total rewards equally, and assign each share to each learner. However, there are two problems associated with this method: one is that in certain situations, the global reward cannot be efficiently computed, particularly in distributed environments. The other problem is that, without sufficient feedback from each learner's action, should it be more helpful if we reward those who perform actions and punish those for laziness? In these two situations, equally dividing the share for each learner is not practical. One approach is called *local rewards* which accesses each agent's performance based merely on its individual behavior. This approach discourages laziness because it only rewards agents who actually act, but, this also may cause greediness.

Balch (Balch (1997, 1999)) states that different credit assignment strategies should be chosen, depending on the problem domain. His experiments explain that the local rewards lead to faster learning rates and fast policy convergence with fully homogeneous behavior, but not necessarily to better performance than global rewards, which lead to greater diversity but with poor convergence policy. For example, in a foraging problem, local rewards produce better results, while in a soccer game, global rewards is better. Balch claims that the more local reinforcement signal increases the homogeneity of the final learned team, which in turn suggest that the credit assignment range can form a certain homogeneity or a certain degree of specialization.

Mataric (Mataric (1994)) specifies in the learning process and promotes agents learning from others as type of local reinforcement, called *social reinforcement*. In this type of reward, agents obtain *observational reinforcement* through observing others and imitating them. This process can improve the overall team behavior by promoting anomalous as well as contributory behaviors. Agents can also receive additional *vicarious reinforcement* whenever others are directly rewarded. A weighted combination of *social reinforcement*, *observational reinforcement* and *vicarious reinforcement* balances local rewards and global rewards, and produces a better result in a foraging application.

There are a number of credit assignment strategies in Panait and Luke (2005) which can favor agents' dynamic learning process, which may also result in dynamics resembling general-sum or even competitive games. Thus, just as in social science, credit assignment policy can complicate the dynamic learning and lead to the co-operator's dilemma (Lichbach (1996)): whether to cooperate or compete. In our research, the credit assignment problems are handled by either the assigning rules which are already known in the stochastic games, or where the reward credit function is defined. Generally, credit assignment problems are more addressed by team cooperative learning communities, rather than those in the reinforcement learning field.

4.1.2 Dynamic Scenarios

Since various credit assignment policies exist, we can divide dynamic learning into two different scenarios: the fully-cooperative scenario and the general-sum game scenario. In a fully-cooperative scenario with only global rewards, if the rewards received by one agent will increase everyone else's reward, it is relatively straightforward to converge to a globally optimal Nash equilibrium in the learning process. Another case is the general-sum, which means that each agent's reward is not less clear, and one agent's gain means the loss of other agent(s). In this situation, the learning process is in a non-global credit assignment policies, and it may include both competing and cooperative scenarios when rewards are involved. We present the influential work on how to find optimal policies in the next part of this chapter, and our research is more focused on general-sum games which involve typically unequal share credit policies.

4.2 Finding Equilibria Algorithms in Game Theory

The algorithms from game theory focus on computing the reward value for the players and a Nash equilibrium, which is used to predict behaviors in stochastic games. The approaches from game theory have some very strong assumptions, which require the game (n, S, A, T, R) to be fully known and observable. Thus, we also call these approaches "model-based". The goal is to compute the value of a Nash equilibrium and the expected reward value for all the players, and in general, without concern for the interaction among players.

There are many approaches for finding algorithms in the game theory field. We mainly discuss the first proposed technique in this paper: fictitious play. Fictitious play was the first proposed technique to find equilibria in matrix games (Brown (1949); Robinson (1951)), and later was extended to stochastic games

(Vrieze (1987)). Note that, fictitious play assumes opponents play stationary or mixed strategies. At each round, each player simulates play of the game and plays the best response to the empirical frequency of the other player's play, $\frac{Q_i(s,a_i)}{t}$. We describe the learning procedure of fictitious play in Alg. 2.

Algorithm 2: Fictitious play for two player, zero-sum stochastic games

```
1. Initialize $Q_i(s, a_i), s \in S, a_i \in A_i$, and $t \leftarrow 0$;
2. Repeat for each sate, $s \in S$,
    (a) Let $a_i = argmax_{a_i \in A_i} Q_i(s, a_i)$;
    (b) Update $Q_i(s, a_i), \forall s \in S, \forall a'_i \in A_i$:
        $Q_i(s, a'_i) \leftarrow Q_i(s, a'_i) + R(s, < a_{-i}, a'_i >) + \gamma \left( \sum_{s' \in S} T(s, a, s') V(s') \right)$,
        where,
        $V(s) = \max_{a_i \in A_i} \frac{Q_i(s, a_i)}{t}$.
    (c) $t \leftarrow t + 1$.
```

4.3 Learning Equilibria Algorithms

In the reinforcement learning communities, learning equilibria presents a different diagram where the goal is to learn through interaction rather than solve an equilibrium. The algorithms, also called as "model-free" approaches, avoid building an explicit model of the opponent's strategy. In general, the agent learns through observation and experience over time and select actions in the environment based on observations of T and R_i, especially T and R_i are not known in advance. Compared with algorithms in game theory field, this model-free approach is more concerned with "play" in stochastic games, to find a solution. The goal of these algorithms is to estimate and converge to a policy in one of the game's Nash equilibria. We review these algorithms of learning equilibrium techniques, as well as determining their conditions for convergence. Note that all the algorithms have a nearly identical structure (see in Alg. 3). These algorithms tend to solve each state as a matrix game, and find equilibrium for the stochastic game. They mainly differ on the *value* operator definition V in the Step2(b).

Algorithm 3: Equilibrium Learning Algorithm: two players a_1, a_2.

```
1. Initialize $Q(s, < a_1, a_2 >), s \in S, a_1, a_2 \in A$, and set $\alpha$ to be the learning rate;
2. Repeat for each state, $s \in S$,
    (a) From state $s$, select actions $a$ that solve the matrix game
        $u[Q(s, < a_1, a_2 >)_{a_1, a_2 \in A}]$, with some exploration;
    (b) Observe joint-action $< a_1, a_2 >$, reward $r$, and next state $s'$, update
        $Q(s, < a_1, a_2 >)$:
            $Q(s, < a_1, a_2 >) \leftarrow (1 - \alpha)Q(s, < a_1, a_2 >) + \alpha\big(r + \gamma V(s')\big),$
        where,
            $V(s) = Value\Big([Q(s, < a_1, a_2 >)_{a_1, a_2 \in A}]\Big).$
```

4.3.1 Minimax-Q

Littman (Littman (1994)) extended the traditional Q-Learning algorithm for MDPs to zero-sum stochastic games. Instead of using the $\max_{a_i \in A}$ in Step2(b) of Alg. 2, the value operator V computes the unique equilibrium value for the zero-sum matrix game defined by the Q value at the current state (see Eq. 4.1). The solution of the zero-sum matrix game is computed using the linear program from Section 3.2.1 which is a minmax function:

$$V_1(s) = \max_{\pi \in \Pi(A_1)} \left(\min_{a_2 \in A2} \sum_{a_1 \in A_1} \pi(a_1) Q_1(s, < a_1, a_2 >) \right) = -V_2(s), \tag{4.1}$$

where, player 1's reward value V_1 is the opposite of player 2's reward value V_2. The idea of *minmax* is to behave to maximize the reward ($\max_{\pi \in \Pi(A_1)}$) under the worst case ($\min_{a_2 \in A2}$), see Alg. 3.

4.3.2 Nash-Q

Hu & Wellman (Hu and Wellman (1998)) extended the Minimax-Q algorithm to two player, general-sum stochastic games. The extension requires that each agent maintain the Q value for all the agents since the reward value is no longer opposite. Similar to the Alg. 3 structure, the Value operator in Step2(b) is the quadratic programming solution for finding a Nash equilibrium in two player general-sum games (see Eq. 4.2), instead of the linear programming solution to find the equilibrium only for zero-sum games. On the other hand, Littman's Minimax-Q learning algorithm assumes that the other agent will always choose a pure Nash equilibrium strategy; instead, this algorithm will choose a mixed strategy.

$$V_i(s) \in Nash\Big(Q_1(s), \cdots, Q_n(s)\Big) \tag{4.2}$$

The Nash-Q learning algorithm is highly general and is guaranteed to converge to the equilibrium, but with restrictive assumptions. The most critical one is that all the intermediate games must have a single equilibrium; and in addition, this equilibrium in all these intermediate games must be a global optimum, which is a joint action that maximizes each agent's payoff.

4.3.3 Friend-or-Foe-Q

Littman's Friend-or-Foe-Q (FFQ) (Littman (2001)) is an equilibrium learner that extends Minimax-Q to include a small class of general-sum games. This algorithm assumes there are two kinds of competing agent in the stochastic games from one agent's perspective: either a *friend* or a *foe*. Knowing the labeling or inferring from the observed rewards, equilibrium policies can be learned in restricted classes of games: e.g. two-player, zero-sum stochastic games, which computes the basic zero-sum linear program of the minimax equilibria (foe-Q) (see Eq. 4.4); e.g., coordination games with uniquely-valued equilibria (friend-Q) (see Eq. 4.3).

$$V_i(s) = \max_{a_1 \in A_1, a_2 \in A_2} \Big(Q(s, < a_1, a_2 >)\Big) \tag{4.3}$$

$$V_1(s) = \max_{\pi \in \Pi(A_1)} \Big(\min_{a_2 \in A2} \sum_{a_1 \in A_1} \pi(a_1) Q_1(s, < a_1, a_2 >)\Big) \tag{4.4}$$

4.3.4 Correlated-Q

The final equilibrium learning technique is Greenwald & Hall's Correlated-Q (CE-Q) (Greenwald and Hall (2003)) in order to generalize both Nash-Q and Friend-and-Foe-Q. CE-Q indicates four variants: utilitarian (uCE-Q), egalitarian (eCE-Q), republican (rCE-Q) and libertarian (lCE-Q), which also demonstrate empirical convergence to equilibrium policies on a testbed of general-sum Markov games. The four correlated equilibrium selection mechanisms are the objective choice of the following functions respectively,

- uCE-Q: maximize the *sum* of the players' rewards:

$$\sigma \in \max_{\sigma \in CE} \sum_i \sum_{a \in A} \sigma(a) Q_i(s, a) \tag{4.5}$$

- eCE-Q: maximize the *minimum* of the players' rewards:

$$\sigma \in \max_{\sigma \in CE} \min_{i} \sum_{a \in A} \sigma(a) Q_i(s,a) \tag{4.6}$$

- rCE-Q: maximize the *maximum* of the players' rewards:

$$\sigma \in \max_{\sigma \in CE} \max_{i} \sum_{a \in A} \sigma(a) Q_i(s,a) \tag{4.7}$$

- lCE-Q: maximize the *maximum* of each player i's rewards:

$$\sigma = \Pi_i \sigma_i, \sigma_i \in \max_{\sigma \in CE} \sum_{a \in A} \sigma(a) Q_i(s,a) \tag{4.8}$$

Using the same algorithm structure in Alg. 3, the Value operator is replaced by Eq. 4.9, where σ satisfies either Eq. 4.5, 4.6, 4.7, 4.8.

$$V_i(s) \in CE_i\Bigg(Q_1(s), \cdots, Q_n(s)\Bigg) = \sum_{a \in A} \sigma(a) Q_i(s,a) \tag{4.9}$$

A correlated equilibrium is a more general concept, and all Nash equilibria are correlated equilibria. A correlated joint policy is an equilibrium if and only if, for each player i, each state s, and each action a_i, the following holds. Let $\sigma_{-i}(a_{-i} \mid a_i)$ be the conditional probability that the other agents select action a_{-i}, given agents are following the correlated joint policy π and agent i is playing action a_i. Then, for all a'_i, the following must be true,

$$\sum_{a_{-i} \in A_{-i}} \sigma_{-i}(a_{-i} \mid a_i) Q^{\pi}(s, < a_i, a_{-i} >) \geq \sum_{a_{-i} \in A_{-i}} \sigma_{-i}(a_{-i} \mid a_i) Q^{\pi}(s, < a'_i, a_{-i} >).$$

From another perspective, given the knowledge about other players' distribution gained from one agent's own prescribed action, that agent gains no increase in expected payoff by playing an action different from its prescribed action. This correlated joint policy is more efficient than Nash-Q, since it does not require the complex quadratic programming Nash equilibrium solver. Instead, according to conditional probability rule: $\pi(a_1, a_2) = \pi(a_1 \mid a_2) * \pi(a_2) = \pi(a_2 \mid a_1) * \pi(a_1)$, correlated equilibruia can be computed via linear programming by treating one player's action as a conditional constraint.

4.4 Learning Best-Response Algorithms

Learning best-response algorithms means directly to learn and play a best-response to other players' policies. Even though these algorithms are not explicitly related to equilibrium, best-response learning algorithms have strong connections to equilibria in terms of the rationality property of the learning algorithms. Two important properties are mentioned by Bowling (Bowling (2003)): *convergence* and *rationality*. We will discuss these two concepts in Section 4.5. Two major kinds best-response learning algorithms in RL field, opponent modeling algorithms and infinitesimal gradient ascent algorithms, are similar to *fictitious play*, which requires observations of the opponent's actions, while the reinforcement learning methods require to maintain different information regarding to reservation over the other players' behaviors. In *fictitious play*, each player maintains a model of the mixed strategy of the other players based on the empirical play so far, and always plays the best response to this model at each iteration.

4.4.1 Q-learning

Q-learning (Watkins (1989)) was originally designed to find optimal policies in MDPs in single-agent learning. However, despite this, it has been widely used for multiagent learning with certain success. Moreover, if the other players play a stationary strategy, the stochastic games can be seen as a MDP; and therefore, Q-learning learns to play an optimal response to the other players. In other words, Q-learning traditionally can not learn or play stochastic policies.

4.4.2 Opponent Modeling

Opponent modeling reinforcement learning algorithms require observations of the opponent's actions, similar to fictitious play. There are two major algorithms: opponent modeling Q-learning (Uther and Veloso (2003)) and joint action learners (JALs) (Claus and Boutilier (1998)), which aim to learn opponents' stationary distributions over their actions but not their individual rewards (different from fictitious play), see Alg. 4. The learned opponents' distribution combined with the joint-action value, are used to select an action. The difference is that Uther & Veloso focused on a zero-sum domain, while Claus & Boutilier investigated team matrix games.

Note that in Alg. 4, $C(s, a_{-i})/n(s)$ denotes the probability that the other players will select joint action a_{-i} based on the past experience. $C(s, a_{-i})$ is the frequency number of playing action a_{-i} at stage s, and $n(s)$ is the totoal number of stage s appears.

Algorithm 4: Opponent modeling Q-learning Algorithm

```
1. Initialize $Q, \forall s \in S, C(s) \leftarrow 0,\ n(s) \leftarrow 0.$;
2. Repeat for each sate, $s \in S$,
    (a) From state $s$, select actions $a_i$ that maximizes,
            $$\sum_{a_{-i}} \frac{C(s, a_{-i})}{n(s)} Q(s, < a_i, a_{-i} >)$$
    (b) Observe other agents' joint-actions $a_{-i}$, reward $r$, and next state $s'$,
        update $Q(s, a)$:
            $$\begin{aligned} Q(s,a) &\leftarrow (1-\alpha)Q(s,a) + \alpha\big(r + \gamma V(s')\big), \\ C(s,a_{-i}) &\leftarrow C(s,a_{-i}) + 1, \\ n(s) &\leftarrow n(s) + 1 \end{aligned}$$
        where,
            $$\begin{aligned} a &= < a_i, a_{-i} > \\ V(s) &= \max_{a_i} \sum_{a_{-i}} \frac{C(s,a_{-i})}{n(s)} Q(s, < a_i, a_{-i} >). \end{aligned}$$
```

4.4.3 Infinitesimal Gradient Ascent

Infinitesimal Gradient Ascent (IGA) (Singh et al. (2000)) is the last best-response learning algorithm. The basic idea is for an agent to adjust its policy in the direction of the gradient of the value function. Agents incrementally adapt their strategy through gradient ascent to an expected payoff. IGA has been proved that, in the simple setting of two-play, two-action, iterated general-sum games, the agents will converge to a Nash equilibrium, or if the strategies may not always converge, their average payoffs will nevertheless converge to the payoffs of a Nash equilibrium. Note that the gradient ascent algorithm assumes a full information game, that is, both players know both game matrices, and can see the mixed strategy of their opponent at the previous step (only if the actual previous move played is visible, a stochastic gradient ascent algorithm can be defined).

Similar in structure to Alg. 4, joint-actions $< a_i, a_{-i} >$ here are replaced by a strategy pair $< \alpha, \beta >$. Assume that two-player, r, c, with two-action, and their payoffs in a general-sum game in matrices as:

$$R = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}, \quad C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

in which the row specifies player 1 and the column indicates the player 2, respectively. The value (or payoff) of the strategy pair $< \alpha, \beta >$ to the row player $V_r(\alpha, \beta)$ and the column player $V_c(\alpha, \beta)$, respectively, are:

$$\begin{aligned} V_r(\alpha,\beta) &= r_{11}\cdot(\alpha\beta) + r_{22}\cdot(1-\alpha)(1-\beta) + r_{12}\cdot\alpha(1-\beta) + r_{21}\cdot(1-\alpha)\beta, \\ V_c(\alpha,\beta) &= c_{11}\cdot(\alpha\beta) + c_{22}\cdot(1-\alpha)(1-\beta) + c_{12}\cdot\alpha(1-\beta) + c_{21}\cdot(1-\alpha)\beta. \end{aligned}$$

In the infinitesimal gradient ascent algorithm, each player repeatedly adjusts their strategy in the direction of their current gradient with some step size η, $\lim_{\eta\to 0}$,

$$\begin{aligned} \alpha_{k+1} &= \alpha_k + \eta\frac{\partial V_r(\alpha_k, \beta_k)}{\partial \alpha} \\ \beta_{k+1} &= \beta_k + \eta\frac{\partial V_c(\alpha_k, \beta_k)}{\partial \beta} \end{aligned} \tag{4.10}$$

Accordingly, the process updates strategy pair $< \alpha, \beta >$ and the Q value, similar to Alg. 4. Here,

$$\begin{aligned} \frac{\partial V_r(\alpha,\beta)}{\partial \alpha} &= \beta u - (r_{22} - r_{12}), \\ \frac{\partial V_c(\alpha,\beta)}{\partial \beta} &= \alpha u' - (c_{22} - c_{21}). \end{aligned} \tag{4.11}$$

in which, letting $u = (r_{11} + r_{22}) - (r_{21} + r_{12})$, and $u' = (c_{11} + c_{22}) - (c_{21} + c_{12})$.

Bowling and Veloso (Bowling and Veloso (2002)) describe two important properties for learning agents: rationality and convergence. They introduce the Win or Learn Fast (WoLF) algorithm, which varies the learning rate from small and cautious values when winning, to large and aggressive values when losing to the others.

4.4.4 Regret Minimization Approaches

Regret minimization approaches seek to minimize regret, not directly learn an equilibrium nor play a best-response, also known as no-regret learning. Described in Eq. 4.12, regret at time T is the difference between the total reward received in T playings and the value of the best stationary strategy over those same playings. Here, r_t is the actual value the player received at time, and $N^T(a_{-i})$ is the number of times the other players played the joint action a_{-i} in the first T trials of a repeated matrix game for player i.

$$Regret_i(T) = \max_{a_i \in A_i} \left(\sum_{a_{-i} \in A_{-i}} N^T(a_{-i}) R_i(< a_i, a_{-i} >) \right) - \left(\sum_{t=1}^{T} r^t \right) \tag{4.12}$$

An algorithm achieves no-regret if and only if,

$$\lim_{T \to \infty} Regret_i(T) \leq 0.$$

In other words, the total amount of reward received by the player must be at least as much as if the player have known the distribution of the other agents' actions ahead of time, but not the order. Notice that the other agents' actions is a fixed strategy, and the algorithm is guaranteed to converge in payoff to the value of the best-response strategy.

No-regret algorithms have been mainly explored in single-state games, such as k-armed bandit problems and matrix games. Two important criteria of the learning rules are *safe* and *consistent* (Fudenberg and Levine (1995)). The first rule, "*safe*", is defined as the requirement that guarantee at least the minimax payoff of the game. The minimax payoff is the maximum expected value a player gained against any opponent. Then the "universal consistency" rule defines that a learning rule do at least as well as the best response to the empirical distribution, regardless of the actual strategy that the opponent is employing. Little work has been done in application of stochastic games. The difficulties of extending this concept to stochastic games are discussed in Mannor and Shimkin (2003). One exception is Manor

(Mannor and Shimkin (2001)) who extends no-regret properties to average-reward stochastic games. More work needs to generalize these approaches to discounted reward stochastic games.

4.5 Properties

In the literature, with respect to the learning algorithms, typical results are evaluated from three properties: convergence of the strategy to equilibria, successfully learning of opponent's strategy, and obtaining optimal payoffs.

First, convergence is the most common one in both game theory and AI literature. Many approaches in AI literature, such as, minimax-Q learning, Nash-Q, FFQ, CE-Q, etc., are respectively proven to converge to a Nash equilibrium in certain types of stochastic games under certain conditions.

Second, rationality is shown as the results of successfully learning opponents' strategies. Since each agent adopts a best response to their beliefs about other agent', the agents will converge to a Nash equilibrium, of the repeated game. However, this result is under the assumption that if the history is observable given the strategies, the agents' belief will only correctly converge .

No-regret learning exemplifies the results of the last property. Two criteria for no-regret learning are *safe* and *consistent*. A large number of algorithms have been shown to satisfy universal consistency (no-regret) requirements. Bowling (Bowling (2005)) combines these criteria in a no-regret learning algorithm, GIGA-WoLF, that provably convergence to a Nash equilibrium, in two-player, two-action stochastic games.

4.6 Summary

In this chapter, we analyze the seminal approaches in multiagent learning research. The multiagent learning process is known as non-stationary, dynamic. First, we illustrate a critical problem along with dynamic learning: credit assignment, and then we present the cooperative scenarios and competing scenarios in the learning problem. Thereafter, we study influential work in MAL on how to learn optimal policies in games. We end this chapter by analyzing important common properties in these proposed learning approaches: convergence, rationality and payoffs. In the next chapter, we will discuss some open issues in the study of MAL and state our research interest.

Chapter 5

Open Issues and Research Interests

As early as 1951, fictitious play as the first learning algorithm was proposed to compute equilibria in games, and there have been numerous proposals regarding learning techniques in stochastic games. The MAL research has produced some inspiring results, yet, it is important to examine the foundations of MAL, and consider some relevant questions. What question exactly is MAL addressing? What is there to learn in stochastic games? What are the yardsticks by which to measure answers to these questions? How can we evaluate the success of learning rules?

Do the agents know the stochastic game, including the stage game and the transition probability? More specifically, the information regarding the following: stochastic stages, transition probabilities, specific actions at each stage, actions available according to the agents, transparent (or not) for all the agents stages, action/strategies, rewards, and so on. These all are rather important factors in the whole process of agents' learning. In general, this learning process can be classified as known or unknown games, observable, partial observable or unobservable play. In broader settings, there is more to learn, not restricted to learning opponents' strategies or the agent' own strategy for proceeding well against opponents.

5.1 Open Issues

In the literature, for the known, fully observable games, there are two aspects to learn in this restricted setting: one is that an agent learns opponents' strategies as a model, so the agent can devise a best (or at least "good") response, (also known as "model-based" learning), for example, fictitious play (Brown (1951)). The other one is that an agent can learn a strategy of its own which does well against the opponents, without explicitly learning the opponents' strategies, (also known as "model-free" learning), for example, Q-learning (Watkins (1989)).

Multiagent learning research still has open issues. Multiple agents act jointly in a common environment to achieve their own agenda, through interaction, either cooperative or in competing with one another. This brings in issues of scalability, adaptive dynamics, and communication. In this chapter, we will discuss them respectively.

5.1.1 Scalability

Scalability is a critical problem for multiagent learning. Multiagent learning involves multiple agents' behaviors in order to solve a common task, thus, the search space can grow exponentially according to the number of agents and the complexity of agent behavior. The evaluating criteria for learning methods should be standardized with respect to their scalability. In a general-sum learning process, especially with partially observed stochastic games, research usually involves studies in two-agent scenarios with two or three actions for each agent. When scaled up to include more agents, current methods are unlikely to work in practice.

5.1.2 Adaptive Dynamics

Due to the small changes caused by agents, multiagent learning can result in an unpredictable global, emergent effect. How does a learning algorithm proceed to discover an optimal solution in a search space with the presence of emergent effect?

5.1.3 Communication

Communication is one means to effectively improve performance and help solve tasks. However, it can increase the learning process search space. The interaction can help solving task through passing or sharing information, but, it can also increase the complexity rapidly, with the number of agents and their sophisticated behaviors.

Still, much research on multiagent communication has been conducted from two perspectives: direct communication and indirect communication. Direct communication is a way for an agent to inform other agents about the past experience which can effectively improve team performance; methods include blackboards (posting and modifying information), messages. Notably, reinforcement learning methods have presumed that the agents have access to a joint policy table to which each agent can contribute. From another perspective, indirect communication uses a third party, such as marking in the environment, to pass information to the others. Most indirect communication is inspired from social insects, such as ants, who utilize pheromones to mark trails to lead others. One agent broadcasts the information in the environment, and the others can exploit it.

Yet, in a multiagent system, (just like any social system), communication is restricted by environment. Researchers claim that unrestricted communication brings the multiagent system back to a single-agent system (Stone and Veloso (2000)). Thus, how to define the communication among agents and allow agents to communicate according to adaptation to the environment is still an open question which needs to be addressed.

5.1.4 Evaluation

In a multiple agents learning process, each agent can constrain, adapt, evolve in the environment of other agents, which are not yet fully understood in game theory, and brings in unknown complexity to computation. How do we set up standard evaluation criteria for learning methods?

5.2 Our Research Interest

In MAL, many aspects have achieved results, such as when all agents adopt the learning procedure under consideration (also called "self play"), the strategy converges to Nash equilibrium of the stage game; agents can learn opponents' strategies (rational learning) successfully. On the other hand, no-regret learning has provided results that the obtained payoffs exceed a specified threshold. However, some observation of the constraints in the literature leads us to some questions:

- While learning procedures apply broadly, the results focus on stochastic games with only two agents. Is this a technical convenience or can we still apply this learning technique to more agents, giving consideration to communication in between?

- With the exception of the work in no-regret learning, the research is mostly focused on investigating convergence to equilibrium play of the stage game. What if the process does not converge to equilibrium play? Should we be concerned, even though better payoffs can be obtained?

- Measuring the performance only against stationary opponents, and not allowing for the possibility of opponents adaptation or learning; this does not seem to be adequate criteria.

- In an infinitely repeated Prisoner's Dilemma game, no-regret dictates the strategy of always defecting, precluding the possibility of cooperation. Should we be concerned?

Seeking answers to these questions will bring us to a new research direction, to learn a robust strategy in certain types of stochastic games, or with minimum adaptation when it comes to different types of stochastic games.

Bibliography

Andre, D. and Teller, A. (1999). Evolving team darwin united. In *RoboCup-98: Robot Soccer World Cup II*, pages 346–351, London, UK. Springer-Verlag.

Axelrod, R. and Hamilton, W. (1981). The evolution of cooperation. *Science*, 211(4489):1390–1396.

Balch, T. (1997). Learning roles: Behavioral diversity in robot teams. pages 7–12. AAAI.

Balch, T. (1999). Reward and diversity in multirobot foraging. In *In IJCAI-99 Workshop on Agents Learning About, From and With other Agents.*

Banerjee, B. and Peng, J. (2005). Efficient no-regret multiagent learning. In *Proceedings of the Twentieth National Conference on Artificial Intelligence.*

Bellman, R. (1957). *Dynamic programming.* Princeton University Press, Princeton.

Bowling, M. (2003). *Multiagent learning in the presence of agents with limitations.* PhD thesis, Pittsburgh, PA, USA. Chair-Veloso, Manuela.

Bowling, M. (2005). Convergence and no-regret in multiagent learning. In *In Advances in Neural Information Processing Systems 17*, pages 209–216. MIT Press.

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

Brown, G. W. (1949). Some notes on computation of games solutions. In *Rand report*, page 78, Santa Monica, California.

Brown, G. W. (1951). Iterative solutions of games by fictitious play. In *Activity Analysis of Production and Allocation*, pages 367–383. Wiley.

Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752. AAAI Press.

Conitzer, V. and Sandholm, T. (2008). New complexity results about nash equilibria. *Games and Economic Behavior*, 63(2):621 – 641. Second World Congress of the Game Theory Society.

Ficici, S. G. and Pollack, J. B. (2000). A game-theoretic approach to the simple coevolutionary algorithm. In *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 467–476, London, UK. Springer-Verlag.

Fudenberg, D. and Levine, D. K. (1995). Consistency and cautious fictitious play. *Journal of Economic Dynamics and Control*, 19(5-7):1065 – 1089.

Gilboa, I. and Zemel, E. (1988). Nash and correlated equilibria: Some complexity considerations. Discussion Papers 777, Northwestern University, Center for Mathematical Studies in Economics and Management Science.

Greenwald, A. and Hall, K. (2003). Correlated-q learning. In *In AAAI Spring Symposium*, pages 242–249. AAAI Press.

Grefenstette, J., Ramsey, C. L., and Schultz, A. C. (1990). Learning sequential decision rules using simulation models and competition.

Hara, A. and Nagao, T. (1999). Emergence of cooperative behavior using adg; automatically defined groups. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99*, pages 1039–1046, San Francisco, CA. Morgan Kaufmann.

Haynes, T. and Sen, S. (1996a). Cooperation of the fittest. Technical Report UTULSA-MCS-96-09, The University of Tulsa.

Haynes, T. and Sen, S. (1996b). Evolving behavioral strategies in predators and prey. In *ADAPTATION AND LEARNING IN MULTIAGENT SYSTEMS*, pages 113–126. Springer Verlag.

Haynes, T. and Sen, S. (1997a). Crossover operators for evolving a team. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 162–167. Morgan Kaufmann Publishers.

Haynes, T. D. and Sen, S. (1997b). Co-adaptation in a team. *INTERNATIONAL JOURNAL OF COMPUTATIONAL INTELLIGENCE AND ORGANIZATIONS*, 1:1–4.

Holland, J. H. (1985). Properties of the bucket brigade. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 1–7, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.

Holland, J. H. and Miller, J. H. (1991). Artificial adaptive agents in economic theory. *American Economic Review*, 81(2):365–71.

Hu, J. and Wellman, M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. In *IN PROCEEDINGS OF THE FIFTEENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING*, pages 242–250. Morgan Kaufmann.

Hu, J. and Wellman, M. P. (2003). Nash Q-learning for general-sum stochastic games. *J. Mach. Learn. Res.*, 4:1039–1069.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH*, 4:237–285.

Kapetanakis, S. and Kudenko, D. (2004). Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1258–1259, Washington, DC, USA. IEEE Computer Society.

Lichbach, M. I. (1996). *The cooperator's dilemma / Mark Irving Lichbach.* University of Michigan Press, Ann Arbor :.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163. Morgan Kaufmann.

Littman, M. L. (2001). Friend-or-foe Q-learning in general-sum games. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322–328, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Littman, M. L. and Szepesvri, C. (1996). A generalized reinforcement-learning model: Convergence and applications. In *In Proceedings of the 13th International Conference on Machine Learning*, pages 310–318. Morgan Kaufmann.

Luke, S. and Spector, L. (1996). Evolving teamwork and coordination with genetic programming. In *GECCO '96: Proceedings of the First Annual Conference on Genetic Programming*, pages 150–156, Cambridge, MA, USA. MIT Press.

Mannor, S. and Shimkin, N. (2001). Adaptive strategies and regret minimization in arbitrarily varying markov environments. In *In Proc. of 14th COLT*, pages 128–142.

Mannor, S. and Shimkin, N. (2003). The empirical bayes envelope and regret minimization in competitive markov decision processes. *Math. Oper. Res.*, 28(2):327–345.

Mataric, M. J. (1994). Interaction and intelligent behavior. Technical report, Cambridge, MA, USA.

Moody, J., Liu, Y., Saffell, M., and Youn, K. (2004). Stochastic direct reinforcement: Application to simple games with recurrence. Technical report, In Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium.

Nash, J. F. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49.

Nash, J. F. (1951). Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295.

Osborne, M. J. and Rubinstein, A. (1994). *A Course in Game Theory*, volume 1 of *MIT Press Books*. The MIT Press.

Panait, L. and Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434.

Panait, L., Wieg, R. P., and Luke, S. (2004a). A sensitivity analysis of a cooperative coevolutionary algorithm biased for optimization. In *Genetic and Evolutionary Computation Conference GECCO 2004, volume 3102 of Lecture Notes in Computer Science*, pages 573–584. Springer.

Panait, L., Wieg, R. P., and Luke, S. (2004b). A visual demonstration of convergence properties of cooperative coevolution. In *In Parallel Problem Solving from Nature PPSN-2004*, pages 892–901. Springer.

Panait, L., Wiegand, R. P., and Luke, S. (2003). Improving coevolutionary search for optimal multiagent behaviors. In *IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence*, pages 653–658, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Potter, M. A. and De Jong, K. A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evol. Comput.*, 8(1):1–29.

Puppala, N., Sen, S., and Gordin, M. (1998). Shared memory based cooperative coevolution. In *Proceedings of the 1998 IEEE World Congress on Computational Intelligence*, pages 570–574, Anchorage, Alaska, USA. IEEE Press.

Quinn, M. (2001). A comparison of approaches to the evolution of homogeneous multi-robot teams. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 128–135 vol. 1.

Quinn, M., Smith, L., Mayley, G., and Husbands, P. (2003). Evolving teamwork and role-allocation with real robots. In *ICAL 2003: Proceedings of the eighth international conference on Artificial life*, pages 302–311, Cambridge, MA, USA. MIT Press.

Rider, R. (1984). The evolution of cooperation : Axelrod, robert, (basic books, inc., 1984) pp. 256. *Journal of Economic Behavior & Organization*, 5(3-4):406–409.

Robinson, J. (1951). An iterative method of solving a game. *The Annals of Mathematics*, 54(2):296–301.

Salustowicz, R. P., Wiering, M. A., and Schmidhuber, J. (1998). Learning team strategies: Soccer case studies. *Mach. Learn.*, 33(2-3):263–282.

Shapley, L. S. (1953). Stochastic Games. *Proceedings of the National Academy of Sciences of the United States of America*, 39(10):1095–1100.

Shoham, Y., Powers, R., and Grenager, T. (2007). If multi-agent learning is the answer, what is the question? *Artif. Intell.*, 171(7):365–377.

Singh, S. P., Kearns, M. J., and Mansour, Y. (2000). Nash convergence of gradient dynamics in general-sum games. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 541–548, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Stone, P. and Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Auton. Robots*, 8(3):345–383.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3(1):9–44.

Sutton, R. S. (1989). Implementation details of the TD(λ) procedure for the case of vector predictions and backpropagation. Technical report.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning.* MIT Press, Cambridge, MA, USA.

t Hoen, P. J. and Tuyls, K. (2004). Analyzing multi-agent reinforcement learning using evolutionary dynamics. In *MACHINE LEARNING: ECML 2004, PROCEEDINGS*, pages 168–179. Springer. LECTURE NOTES IN COMPUTER SCIENCE, 3201.

Tuyls, K., Verbeeck, K., and Lenaerts, T. (2003). A selection-mutation model for q-learning in multi-agent systems. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 693–700, New York, NY, USA. ACM.

Uther, W. T. B. and Veloso, M. M. (2003). Adversarial reinforcement learning. Technical Report CMU-CS-03-107, Carnegie Mellon University.

Vidal, J. and Durfee, E. (1998). The moving target function problem in multi-agent learning. In *ICMAS '98: Proceedings of the 3rd International Conference on Multi Agent Systems*, page 317, Washington, DC, USA. IEEE Computer Society.

Vidal, J. M. and Durfee, E. H. (2003). Predicting the expected behavior of agents that learn about agents: The clri framework. *Autonomous Agents and Multi-Agent Systems*, 6(1):77–107.

von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press.

Vrieze, O. (1987). *Stochastic games with finite state and action spaces*. CWI tracts.

Wang, X. and Sandholm, T. (2002). Reinforcement learning to play an optimal nash equilibrium in team markov games. In *in Advances in Neural Information Processing Systems*, volume 15, pages 1571–1578.

Watkins, C. (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge University, England.

Watkins, C. J. C. H. and Dayan, P. (1992). Technical note: Q-learning. *Mach. Learn.*, 8(3-4):279–292.

Wiegand, R. P. (2004). *An analysis of cooperative coevolutionary algorithms*. PhD thesis, Fairfax, VA, USA. Director-Jong, Kenneth A.

Wiering, M., Salustowicz, R., and Schmidhuber, J. (1999). Reinforcement learning soccer teams with incomplete world models. *Auton. Robots*, 7(1):77–88.