ELSEVIER

Journal of Cognitive Systems Research 2 (2001) 55–66

Cognitive Systems RESEARCH

www.elsevier.com/locate/cogsys

Value-function reinforcement learning in Markov games

Action editor: Ron Sun

Michael L. Littman

AT&T Labs Research, 180 Park Avenue, Florham Park, NJ 07932-0971, USA

Received 1 October 2000; accepted 10 October 2000

Abstract

Markov games are a model of multiagent environments that are convenient for studying multiagent reinforcement learning. This paper describes a set of reinforcement-learning algorithms based on estimating value functions and presents convergence theorems for these algorithms. The main contribution of this paper is that it presents the convergence theorems in a way that makes it easy to reason about the behavior of simultaneous learners in a shared environment.

Keywords: Reinforcement learning; Temporal difference learning; Value functions; Game theory; Markov games; *Q*-learning; Nash equilibria

1. Introduction

Game theory (von Neumann & Morgenstern, 1947) provides a powerful set of conceptual tools for reasoning about behavior in multiagent environments. Markov games (van der Wal, 1981), or stochastic games (Owen, 1982; Shapley, 1953), are a formalization of temporally extended agent interaction.

Reinforcement learning (Kaelbling, Littman & Moore, 1996; Sutton & Barto, 1998) is the problem of an agent learning to behave from experience. One well studied approach to reinforcement learning is building a value function — a mapping from state to expected reward. Several authors have applied value-function reinforcement learning to Markov games to create agents that learn from experience how to best interact with other agents. This paper presents several value-function reinforcement-learning algorithms and what is known about how they behave when learning simultaneously in different types of games.

Section 2 describes single-agent environments and the basic *Q*-learning algorithm, which converges to an optimal value function and optimal behavior in this type of environment. Section 3 examines multi-agent environments and the Nash *Q*-learning algorithm. Section 4 looks at one situation in which Nash *Q*-learning converges — when there are adversarial equilibria; Section 5 examines another — when there

E-mail address: mlittman@research.att.com (M.L. Littman).

1389-0417/01/$ – see front matter
PII: S1389-0417(01)00015-8

are coordination equilibria. Section 6 presents some concluding thoughts.

2. Single-agent environments

Markov decision processes (MDPs) (Bellman, 1957; Howard, 1960) are a descriptive model of single-agent environments. In the MDP framework, it is assumed that, although there may be a great deal of uncertainty about the effects of an agent's actions, there is never any uncertainty about the agent's current state — it has complete and perfect perceptual abilities.

2.1. Markov decision processes

Mathematically, a Markov decision process is a tuple $\langle \mathscr{S}, \mathscr{A}, T, R, \beta \rangle$, where

- $\mathscr{S}$ is a finite set of *states* of the environment;
- $\mathscr{A}$ is a finite set of *actions* available to the agent;
- T: $\mathscr{S} \times \mathscr{A} \to \Pi(\mathscr{S})$ is the *transition function*, giving for each state and agent action, a probability distribution over states ($T(s, a, s')$ is the probability of ending in state s', given that the agent starts in state s and takes action a);
- R: $\mathscr{S} \times \mathscr{A} \to \mathbb{R}$ is the reward function, giving the expected immediate reward gained by the agent for taking each action in each state ($R(s, a)$ is the expected reward for taking action a in state s); and
- $0 \leq \beta < 1$ is a discount factor.

This paper considers only models with finite state and action spaces.

In an MDP, agents should act in such a way as to maximize some measure of their long-run reward received. Under the *discounted objective*, which is the focus of this paper, the discount factor $0 \leq \beta < 1$ controls how much effect future rewards have on the decisions at each moment, with small values of β emphasizing near-term gain and larger values giving significant weight to later situations. Concretely, a reward of r received t steps in the future is worth $\beta^t r$ to the agent now. Mathematically, the discount factor has the desirable property that if all immediate rewards are bounded, then the infinite sum of the discounted rewards is also bounded. From an applications perspective, the discount factor can be thought of as the probability that the agent will be allowed to continue gathering reward after the current step, or, from an economic perspective, as an inverse interest rate on reward (Puterman, 1994).

A policy is a description of the behavior of an agent. A *stationary policy*, $\pi:\mathscr{S} \to \Pi(\mathscr{A})$, specifies, for each state, a probability distribution over actions to be taken. Notationally, $\pi(s, a)$ is the probability assigned to action a in state s. A *deterministic* policy is one that assigns probability 1 to some action in each state. Every MDP has a deterministic stationary optimal policy (Bertsekas, 1987).

A policy π for an agent can be evaluated by computing the long-run value the agent can expect to gain. Let $Q^{\pi}(s, a)$ be the expected discounted future reward to the agent for starting in state s and executing action a for one step, then continuing according to policy π. This can be defined by a set of simultaneous linear equations, one for each state s:

$$Q^{\pi}(s, a) = R(s, a) + \beta \sum_{s' \in \mathscr{S}} T(s, a, s') \times \sum_{a' \in \mathscr{A}} \pi(s', a') Q^{\pi}(s', a').$$

The function Q^{π} is called the *Q-function* for π.

Given an initial state s, the agent should execute a policy π that maximizes $\Sigma_a\ \pi(s, a) Q^{\pi}(s, a)$. Howard (1960) showed that there exists a stationary deterministic policy π^* that is optimal for every starting state. The Q-function for this policy, written Q^*, is defined by the set of equations

$$Q^*(s, a) = R(s, a) + \beta \sum_{s' \in \mathscr{S}} T(s, a, s') \max_{a' \in \mathscr{A}} Q^*(s', a'), \tag{1}$$

and the *greedy policy* that assigns probability one to action $\text{argmax}_a\ Q^*(s, a)$ in state s is optimal (Puterman, 1994).

The presence of the maximization operator in Eq. (1) means the system of equations is not linear. Methods such as value iteration, policy iteration, linear programming, and modified policy iteration can be used to solve the equations (Puterman, 1994).

Barto, Sutton and Watkins (1989) argue that MDPs are an appropriate model for studying reinforcement learning in single-agent environments. The next section describes a reinforcement-learning algorithm with guaranteed performance in MDP environments.

2.2. Q-learning

Q-learning (Watkins, 1989; Watkins & Dayan, 1992) can be viewed as a sampled, asynchronous method for estimating the optimal Q-function for an unknown MDP. Q-learning is a temporal-difference learning method (Sutton, 1988), and its basic version keeps a table of values, $Q[s, a]$, with an entry for each state/action pair. The entry $Q[s, a]$ is an estimate for $Q^*(s, a)$ as defined in Eq. (1). An agent uses its experience to improve its estimate, blending new information into its prior experience according to a *learning rate* $0 < \alpha < 1$.

The Q-function is an ideal data structure for reinforcement learning. There are three fundamental functions in algorithms for solving MDPs: the value function V mapping state to value, the Q-function Q mapping state and action to value, and the policy π mapping state to a probability distribution over actions. Given a model in the form of transition and reward functions, any of the mappings can be computed from any one of the others. Without access to T and R, however, only the Q-function can be used to reconstruct the other two: $V(s) = \max_a Q(s, a)$ and $\pi(s, a) = 1$ if $a = \text{argmax}_{a'} Q(s, a')$, and 0 otherwise. In addition, the Q-function is not difficult to estimate from experience.

The experience available to a reinforcement-learning agent in an MDP environment can be summarized by a sequence of experience tuples $\langle s, a, r, s' \rangle$. An experience tuple is a snapshot of a single transition: the agent starts in state s, takes action a, receives reward r and ends up in state s'.

Given an experience tuple $\langle s, a, r, s' \rangle$, the Q-learning rule is

$$Q[s, a] := (1 - \alpha)Q[s, a] + \alpha\left(r + \beta \max_{a'} Q[s', a']\right). \tag{2}$$

This creates a new estimate of $Q^*(s, a)$ by adding the immediate reward to the current discounted estimate of the discounted reward starting from s'. Because of the way r and s' are chosen, the average value of this new estimate is exactly $R(s, a) + \beta \sum_{s'} T(s, a, s') \max_{a'} Q[s', a']$. In a noise-free environment, this value would be directly assigned to $Q[s, a]$. However, to get an accurate estimate, we need to average together many independent samples. The learning rate α blends our present estimate with our previous estimates to produce a best guess at $Q^*(s, a)$; it needs to be decreased slowly for the estimated Q-function to converge to Q^* (Jaakkola, Jordan & Singh, 1994; Szepesvári & Littmann, 1999; Tsitsiklis, 1994; Watkins & Dayan, 1992).

The greedy policy, which assigns probability one to action $\text{argmax}_a Q[s, a]$ in state s, receives rewards that converge to optimality as Q approaches Q^*. However, if this greedy policy is used to choose actions throughout the learning process, the agent may not explore a sufficient amount to guarantee optimal performance. So, the basic Q-learning algorithm identifies optimal behavior, but cannot adopt it.

Singh, Jaakkola, Littman and Szepesvári (2000) show that the conflict between learning the optimal policy and executing the optimal policy can be overcome by selecting actions that are greedy in the limit with infinite exploration (GLIE). A concrete example of a GLIE policy is decaying ϵ-greedy exploration. Let $n(s)$ be the number of times state s has been encountered so far during learning. Let $\epsilon(s) = c/n(s)$ for $0 < c < 1$. Now, if the agent selects the greedy action in state s with probability $1 - \epsilon(s)$ and a random 'exploratory' action in state s with probability $\epsilon(s)$, the rewards obtained by the learner will converge to optimal.

An agent *converges in behavior* if its action distribution becomes stationary (fixed) in the limit. A GLIE policy need not converge in behavior, since ties in greedy actions are broken arbitrarily. However, if there is a unique optimal policy and the Q-function converges, a Q-learning agent will converge in behavior as well. Convergence in behavior is important in multi-agent settings, since an agent's optimal policy may depend on how other agents behave. If one agent converges in behavior, we can attempt to analyze how other agents in the environment respond.

The following theorem is a consequence of the results of Singh et al. (2000).

Theorem 1. *In a single-agent environment, an agent* *following the Q-learning update rule will converge to the optimal Q-function with probability one. Furthermore, if the agent follows a GLIE policy and the optimal policy is unique, it will converge in behavior with probability one.*

3. Multiagent environments

In the Markov decision process model, a decision-making agent interacts with its environment, represented as a probabilistic transition function. In this view, secondary agents must be fixed in their behavior. The framework of Markov games admits a wider view that includes multiple adaptive agents with interacting or competing goals.

3.1. Markov games

In its general form, an n-player Markov game is defined by a tuple $\langle \mathcal{S}, \mathcal{A}_1, \ldots, \mathcal{A}_n, T, R_1, \ldots, R_n, \beta \rangle$, where

- $\mathcal{S}$ is a finite set of *states* of the environment;
- $\mathcal{A}_1, \ldots, \mathcal{A}_n$ is a collection of finite sets of *actions* available to each agent ($\mathcal{A}_i$ is the set of actions for agent i);
- T: $\mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_n \rightarrow \Pi(\mathcal{S})$ is the *transition function*, giving for each state and one action from each agent, a probability distribution over states ($T(s, a_1, \ldots, a_n, s')$ is the probability of ending in state s', given that the agents start in state s and agent 1 chooses $a_1 \in \mathcal{A}_1$, agent 2 chooses $a_2 \in \mathcal{A}_2$, etc.);
- R_i: $\mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_n \rightarrow \mathbb{R}$ for $1 \leq i \leq n$ is each agent's reward function, giving the expected immediate reward gained by agent i for each set of action choices the group of agents could make in each state ($R_i(s, a_1, \ldots, a_n)$ is the expected reward to agent i in state s when agent 1 chooses a_1, agent 2 chooses a_2, etc.); and
- $0 \leq \beta < 1$ is a discount factor.

Once again, agents attempt to maximize their expected sum of discounted rewards, $E\{\sum_{j=0}^{\infty} \beta^j r^i_{t+j}\}$, where r^i_{t+j} is the reward received j steps into the future by agent i.

If a set of agents adopts stationary policies $\pi_1, \ldots, \pi_n$, we can define a set of Q-functions for agent $1 \leq i \leq n$ much like in the MDP case as follows:

$$Q_i^{\pi}(s, a_1, \ldots, a_n) = R_i(s, a_1, \ldots, a_n) + \beta \sum_{s' \in \mathcal{S}} T(s, a_1, \ldots, a_n, s') \cdot \sum_{a_1' \in \mathcal{A}_1, \ldots, a_n' \in \mathcal{A}_n} \pi_1(s', a_1') \cdots \pi_n(s', a_n') Q_i^{\pi}(s', a_1', \ldots, a_n').$$

Here, Q-functions are defined over joint actions for each of the agents. Each agent receives rewards according to its reward function, with transitions dependent on the actions chosen jointly by the set of agents.

With respect to the same set of stationary policies, we can also define the *best-response* Q-function for each agent i:

$$Q_{\triangle i}^{\pi}(s, a_1, \ldots, a_n) = R_i(s, a_1, \ldots, a_n) + \beta \sum_{s' \in \mathcal{S}} T(s, a_1, \ldots, a_n, s') \cdot \max_{a_i' \in \mathcal{A}_i} \sum_{a_1', \ldots, a_{i-1}'} \sum_{a_{i+1}', \ldots, a_n'} \times \pi_1(s', a_1') \cdots \pi_{i-1}(s', a_{i-1}') \cdot \pi_{i+1} \times (s', a_{i+1}') \cdots \pi_n(s', a_n') Q_{\triangle i}^{\pi}(s', a_1', \ldots, a_n').$$

The idea here is that $Q_{\triangle i}^{\pi}(s, a_1, \ldots, a_n)$ is the Q-function obtained by holding all policies except for π_i fixed, then having agent i choose actions to maximize reward. Note that holding the behavior of all other agents fixed leaves agent i in a single-agent environment — an MDP.

This observation, combined with results of Singh et al. (2000), leads to this theorem.

Theorem 2. *In a multiagent environment, an agent following the Q-learning update rule will converge to the optimal response Q-function with probability one as long as all other agents converge in behavior with probability one. Furthermore, if the agent follows a GLIE policy and its best response policy is unique, it will also converge in behavior with probability one.*

Note that Theorem 2 does not imply that two simultaneous Q-learners will converge to mutual best responses.

As in Section 2, an optimal policy is one that maximizes the expected sum of discounted reward. There are subtleties in applying this objective to Markov games, however. Firstly, consider the parallel scenario in MDPs.

In an MDP, an optimal policy is one that maximizes the expected sum of discounted reward; it is *undominated*, meaning that there is no state from which any other policy can achieve a better expected sum of discounted reward. Every MDP has at least one optimal policy, and of the optimal policies for a given MDP, at least one is stationary and deterministic.

For many Markov games, there is no policy that is undominated because performance depends critically on the behavior of the other agents in the environment. How, then, can we define an optimal policy? An elegant idea from the game-theory literature is to define an agent's optimal behavior as being its behavior at a *Nash equilibrium*. A set of policies $\pi_1, \ldots, \pi_n$ is in Nash equilibrium if each is a best response to the others. That is, if for all $1 \le i \le n$, the value attained by agent i from any state s,

$$\sum_{a_1, \ldots, a_n} \pi_1(s, a_1) \cdots \pi_n(s, a_n) Q_i^{\pi}(s, a_1, \ldots, a_n),$$

is equal to its best response value

$$\max_{a_i \in \mathcal{A}_i} \sum_{a_1, \ldots, a_{i-1}} \sum_{a_{i+1}, \ldots, a_n} \pi_1(s, a_1) \cdots \pi_{i-1}(s, a_{i-1}) \cdot \pi_{i+1}(s, a_{i+1}) \cdots \pi_n(s, a_n) Q^{\pi}_{\triangle i}(s, a_1, \ldots, a_n).$$

At a Nash equilibrium, each agent is maximizing its reward given that all other agents remain fixed.

Filar and Vrieze (1997) show that every Markov game has a Nash equilibrium in stationary policies. However, in contrast to MDPs, these policies are stochastic, in general. A classic example is 'Rock, Paper, Scissors', in which any deterministic policy can be consistently defeated, whereas the optimal stochastic policy always breaks even.

3.2. Nash Q-learning

Define $\text{Nash}_i(s, Q_1, \ldots, Q_n)$ to be a one-stage Nash equilibrium policy for agent i in state s, where the total payoff to agent j is defined by Q-function Q_j in state s. Define $\text{Val}_i(s, Q_1, \ldots, Q_n)$ to be the value obtained by agent i at this Nash equilibrium:

$$\text{Val}_i(s, Q_1, \ldots, Q_n) = \sum_{a_1, \ldots, a_n} \text{Nash}_1(s, Q_1, \ldots, Q_n)[a_1] \cdots \text{Nash}_n(s, Q_1, \ldots, Q_n)[a_n] Q_i[s, a_1, \ldots, a_n].$$

In words, the value received by agent i is the expected value of the agent's future reward (in the Q function). The expected value is taken over all possible joint actions of the n agents, where we expect each agent to select actions according to the Nash equilibrium policy it chooses. Note that Val_i need not be unique, in general, as games can have multiple Nash equilibria with different values. Further, even if a game admits only one value of Val_i, it need not be the case that the policy Nash_i achieves this value, because other agents can choose other Nash equilibrium policies that don't fit with Nash_i. Finally, even if there is a unique Nash equilibrium, Nash_i need not achieve Val_i because other agents might not choose equilibrium strategies at all. Nonetheless, a Nash equilibrium policy is still one sensible choice of policy given the complexity of multiagent environments.

Since the Nash and Val functions extend the notion of greedy action choice with argmax and max in MDPs to Markov games, a natural extension to the update rule from Q-learning (Eq. (2)) is to use the Nash equilibrium value in place of the max to estimate each agent i's Q-function:

$$Q_i[s, a_1, \ldots, a_n] := (1-\alpha) Q_i[s, a_1, \ldots, a_n] + \alpha(r_i + \beta \text{Val}_i(s, Q_1, \ldots, Q_n)), \quad (3)$$

given an experience tuple $\langle s, a_1, \ldots, a_n, r_1, \ldots, r_n, s' \rangle$. This update rule is due to Hu and Wellman (1998).

This learning algorithm is not known to converge in general, even if there is a unique value of the game. Some conditions to guarantee convergence are given in the next two sections. Note, however, that Hu (1999) and Hu and Wellman (2000) found that the rule sometimes converged in simulations even when the strict assumptions needed to guarantee convergence were not satisfied.

4. Adversarial equilibria

Markov games can have a wide variety of reward structures and learning algorithms can display different dynamics depending on this structure. This section examines payoff structures that, to some degree, are in conflict with each other.

Define an *adversarial equilibrium* in an n-player game as one that is a saddle point. This means that if one agent deviates from the equilibrium, it not only hurts the agent, but it helps all other agents. More formally, let π_1, π_2, etc. be policies in equilibrium. Then, it must be the case that for all states s, for all alternative policies π'_1, π'_2, etc.

$$\sum_{a_1,\ldots,a_n} \pi_1(s,a_1)\cdots\pi_n(s,a_n)Q_i[s,a_1,\ldots,a_n] \geq \sum_{a_1,\ldots,a_n} \pi_1(s,a_1)\cdots \pi_{i-1}(s,a_{i-1})\pi'_i(s,a_i)\pi_{i+1}(s,a_{i+1})\cdots \pi_n(s,a_n)Q_i[s,a_1,\ldots,a_n] \quad (4)$$

for all i. Thus, agent i prefers π_i to π'_i; each prefers to stay than switch.

In addition, if π_1, π_2, etc. are in adversarial equilibrium, it must be the case that for all states s, for all alternative policies π'_1, π'_2, etc.

$$\sum_{a_1,\ldots,a_n} \pi_1(s,a_1)\cdots\pi_n(s,a_n)Q_i[s,a_1,\ldots,a_n] \leq \sum_{a_1,\ldots,a_n} \pi'_1(s,a_1)\cdots \pi'_{i-1}(s,a_{i-1})\pi_i(s,a_i)\pi'_{i+1}(s,a_{i+1})\cdots \pi'_n(s,a_n)Q_i[s,a_1,\ldots,a_n]. \quad (5)$$

This means that each agent would prefer that the *other* agents switch. In a sense, what is good for one agent is bad for the others.

The following theorem is a consequence of the results of Hu and Wellman (1998) and Bowling (2000).

Theorem 3. *In a multiagent environment, an agent following the Nash Q-learning update rule will converge to the optimal Q-function with probability one as long as all Q-functions encountered have adversarial equilibria and these are used in the update rule. Furthermore, if the agent follows a GLIE policy and the limit equilibrium is unique, it will converge in behavior with probability one.*

For convergence to the game's optimal Q-function, all combinations of actions must be executed infinitely often. In a sense, this puts a restriction on the behavior of the other agents in the environment. In general, the learning algorithm converges to the optimal Q-function for the game defined by the set of actions that *are* executed infinitely often in combination with all other actions.

Theorem 3 is interesting, but hard to apply in general. The condition that all Q-functions encountered during learning have adversarial equilibria is difficult to verify for an arbitrary Markov game. For example, even if the immediate reward functions R have adversarial equilibria and the Q-functions are initialized to have adversarial equilibria, none of the intermediate Q-functions need have adversarial equilibria.

Also, note that this theorem puts limitations on the Nash Q-learning update itself. In particular, it is not sufficient that the intermediate Q-functions simply *possess* adversarial equilibria, these equilibria must be used in the update rule. This point is important because of the non-uniqueness of equilibria (a game can have both an adversarial *and* a coordination equilibrium, defined later) and seems not to have been noted by Hu and Wellman (1998) or Bowling (2000).

The next section describes a common class of games that are guaranteed to have adversarial equilibria throughout the learning process.

4.1. Zero-sum Markov games

Zero-sum Markov games are a well-studied specialization of Markov games in which two agents have diametrically opposed goals. In particular, for all $a_1 \in \mathcal{A}_1$, $a_2 \in \mathcal{A}_2$, and $s \in \mathcal{S}$, $R_1(s, a_1, a_2) = -R_2(s, a_1, a_2)$. In a sense, therefore, there is only a single reward function R_1, which agent 1 tries to maximize and agent 2 tries to minimize. Zero-sum games can also be called adversarial or fully competitive for this reason.

The notion of a Nash equilibrium takes on special meaning in zero-sum games: each policy is evaluated with respect to the opposing policy that makes it look the worst. This performance measure prefers conservative strategies that can force any opponent to a stalemate over more daring ones that accrue a great deal of reward against some opponents and lose a great deal to others. This is the essence of minimax: behave so as to maximize your reward in the worst case.

Zero-sum Markov games were first studied by Shapley (1953), who showed that each such game has a unique value function and gave an algorithm that converges to this value function. Thus, even though zero-sum Markov games are a strict generalization of MDPs, they predate MDPs by several years.

Because every reward received by agent 1 in a zero-sum game is received with a sign flip by agent 2, we have that $Q_2 = -Q_1$; therefore, only one Q-function needs to be learned. Therefore, the Nash equilibrium conditions (Eq. (4)) can be written simply as:

$$\sum_{a_1, a_2} \pi_1(s, a_1)\pi_2(s, a_2)Q_1[s, a_1, a_2] \geq \sum_{a_1, a_2} \pi'_1(s, a_1)\pi_2(s, a_2)Q_1[s, a_1, a_2]$$

and

$$\sum_{a_1, a_2} \pi_1(s, a_1)\pi_2(s, a_2)Q_1[s, a_1, a_2] \leq \sum_{a_1, a_2} \pi_1(s, a_1)\pi'_2(s, a_2)Q_1[s, a_1, a_2].$$

Note that these instantly satisfy the conditions for being an adversarial equilibrium (Eq. (5)).

The preceding facts imply that if we apply Nash Q-learning to a zero-sum game, all Q-functions encountered during learning will have adversarial equilibria. Thus, Theorem 3 applies and convergence is guaranteed. The next section explains how to simplify the Nash Q-learning algorithm in the context of zero-sum games.

4.2. Minimax Q-learning

Minimax Q-learning is a value-function reinforcement-learning algorithm specifically designed for zero-sum games. It is described in an earlier paper (Littman, 1994), which includes empirical results on a simple zero-sum Markov game version of soccer. Other researchers have carried out similar studies (Uther & Veloso, 1997).

Note that the definition of the value of a game can be simplified in the zero-sum case as:

$$\mathrm{Val}_1(s, Q_1) = \max_{\pi_1(s,\cdot) \in \Pi(\mathcal{A}_1)} \min_{a_2 \in \mathcal{A}_2} \sum_{a_1 \in \mathcal{A}_1} \pi_1(s, a_1)\, Q_1[s, a_1, a_2].$$

This equation identifies the probability distribution that maximizes the expected value in the face of the worst-possible action choice of the opponent. This calculation can be carried out via a small linear program. Note that the min can also be defined over stochastic policies, but, since it is 'inside' the max, the minimum is achieved for a deterministic action choice.

Using this revised definition of game value, the update rule for minimax Q-learning can be written:

$$Q_1[s, a_1, a_2] := (1 - \alpha)Q_1[s, a_1, a_2] + \alpha(r_1 + \beta \mathrm{Val}_1(s, Q_1)). \qquad (6)$$

The convergence of this approach follows from the convergence of the generalized Q-learning algorithm (Littman & Szepesvári, 1996; Szepesvári & Littman, 1999) as well as the results in Section 3.2.

Theorem 4. *In a two-player zero-sum multiagent environment, an agent following the minimax Q-learning update rule will converge to the optimal Q-function with probability one. Furthermore, if the agent follows a GLIE policy and the limit equilibrium is unique, it will converge in behavior with probability one.*

The special structure of zero-sum games makes it possible to provide several additional guarantees about minimax Q-learning. Even if the limit equilibrium is not unique, a minimax Q-learning agent with a GLIE policy in a zero-sum Markov game converges to a policy that always achieves at least its optimal value regardless of its opponent. So, the policy learned by a minimax Q-learning agent is *safe* in that it can be executed in total ignorance of its opponent and still exhibit its intended effects.

Note that adversarial equilibrium policies in general achieve their learned value (or more) regardless of the opponent's action. The policy used in the minimax Q-learning update rule has a stronger guarantee — that it achieves the *largest* value possible in the absence of knowledge of the opponent's policy. This suggests that, even in a non-zero-sum game with adversarial equilibria, minimax Q-learning is preferable to Nash Q-learning — if the agent ignores the opponent's payoffs and assumes it is in a zero-sum game, it will do no worse, and possibly better, in terms of expected reward.

In spite of its strong asymptotic guarantees, minimax Q-learning can be slow to learn. Uther and Veloso (1997) argue that single-agent Q-learning is a sensible alternative in zero-sum games because it appears to learn more quickly. However, because Q-learning's update and action selection are deterministic, it can be 'tricked' into playing suboptimally. Consider the behavior of Q-learning in the game 'Rock, Paper, Scissors'. An opponent algorithm could be written that simulates the updates performed by Q-learning and always selects the optimal response to what Q-learning is about to choose. Thus, Q-learning will score considerably worse than minimax Q-learning (-1 at every stage as opposed to an average score of 0 for minimax Q-learning).

The next section examines the behavior of Nash Q-learning and variants in coordination settings.

5. Coordination equilibria

This section examines payoff structures that make it so that the agents, to some degree, are working toward a common goal.

Define a *coordination equilibrium* in an n-player game as one for which all agents achieve their maximum possible payoff. If π_1, π_2, etc. are in coordination equilibrium, we have that

$$\sum_{a_1,\ldots,a_n} \pi_1(s,a_1)\cdots\pi_n(s,a_n)Q_i[s,a_1,\ldots,a_n] = \max_{a_1,\ldots,a_n} Q_i[s,a_1,\ldots,a_n]$$

for all $1 \leq i \leq n$ and states s. One observation is that if a game has a coordination equilibrium, it has a deterministic coordination equilibrium. This follows from the fact that the value for each agent is a convex combination of the values in the Q-functions.

One consequence of a game possessing a coordination equilibrium is that no agent has any incentive to switch because no other set of policies could result in a higher score. In a sense, what is best for one agent is also best for the others.

The following theorem is a consequence of the results of Hu and Wellman (1998) and Bowling (2000).

Theorem 5. *In a multiagent environment, an agent following the Nash Q-learning update rule will converge to the optimal Q-function with probability one as long as all Q-functions encountered have coordination equilibria and these are used in the update rule. Furthermore, if the agent follows a GLIE policy and the limit equilibrium is unique, it will converge in behavior with probability one.*

Like Theorem 3, this theorem is hard to apply because the conditions are difficult to verify in advance. The next section addresses a special case in which the conditions are easily verifiable — team games.

5.1. Team Markov games

In team Markov games, agents have precisely the same goals. In particular, for all $a_1 \in \mathscr{A}_1$, $a_2 \in \mathscr{A}_2$, etc., and $s \in \mathscr{S}$, $R_1(s, a_1, \ldots, a_n) = R_2(s, a_1, \ldots, a_n) = \ldots$. In a sense, therefore, there is only a single reward function R_1, which all agents try to maximize together. Team games can also be called coordination games or fully cooperative games for this reason. Boutilier (1996) refers to team Markov games as multiagent decision processes, since, acting together, the group of agents is faced with an MDP.

Because every reward received by agent 1 in a team game is received by all agents, we have that

$Q_1 = \cdots = Q_n$; therefore, only one Q-function needs to be learned. Let $a_1^*, \ldots, a_n^* = \text{argmax}_{a_1, \ldots, a_n} Q_1[s, a_1, \ldots, a_n]$. The set of policies in which each agent i assigns probability 1 to a_i^* is a coordination equilibrium.

The above facts imply that if we apply Nash Q-learning to a team game, all Q-functions encountered during learning will have coordination equilibria. Thus, Theorem 5 applies and convergence is guaranteed. The next section explains how to simplify the Nash Q-learning algorithm in the context of team games.

5.2. Team Q-learning

Team Q-learning is a value-function reinforcement-learning algorithm specifically designed for team games. The definition of the value of a game can be simplified in the team case as:

$$\text{Val}_1(s, Q_1) = \max_{a_1, \ldots, a_n} Q_1[s, a_1, \ldots, a_n].$$

This equation simply returns the largest entry in the Q table for the given state.

Using this revised definition of game value, the update rule for team Q-learning can be written:

$$Q_1[s, a_1, \ldots, a_n] := (1 - \alpha) Q_1[s, a_1, \ldots, a_n] + \alpha(r_1 + \beta \text{Val}_1(s, Q_1)). \quad (7)$$

This update is slightly different from the joint-action learner (JAL) algorithm of Claus and Boutilier (1998), since it doesn't use an opponent model.

The convergence of this approach follows from the convergence of the generalized Q-learning algorithm (Littman & Szepesvári, 1996; Szepesvári & Littman, 1999) as well as the results in Section 3.2.

Theorem 6. *In a multiagent environment where agents have identical reward functions, an agent following the team Q-learning update rule will converge to the optimal Q-function with probability one. Furthermore, if the agent follows a GLIE policy and the limit equilibrium is unique, it will converge in behavior with probability one.*

Note that this algorithm learns precisely the same values as Nash Q-learning for games with coordination equilibria, even in a non-team game. The calculation for the team Q-learning update is simpler, though, and more easily applied to n-player games.

An important open problem for team Markov games is finding a robust and general way of selecting an equilibrium when there are multiple coordination equilibria. Boutilier (1996) argues for establishing a tie-breaking scheme (e.g. lexicographic ordering) to pick the equilibrium to play. However, in the presence of noise, there is no guarantee there will ever be ties and the learners could perceive different unique equilibria and therefore not converge on optimal play.

6. Conclusions

This section explores some of the implications of the results described in the paper.

6.1. Value-function reinforcement learning

The theorems described above can be used to reason about the dynamics of simultaneously learning agents.

It follows from Theorem 4 that two independent minimax Q-learning agents will learn Q-functions that converge. In a zero-sum Markov game, the resulting behavior will be optimal for both agents (mutual best responses). This follows from the fact that equilibria in zero-sum games are 'interchangeable'. In other environments, even with more than two agents, all minimax Q-learners behave in a way that maximizes their reward given no assumption whatsoever about the behavior of the other agents.

In a zero-sum Markov game between a minimax Q-learner and a Q-learner, the Q-learner will converge to optimal behavior (Theorem 2) if the equilibrium is unique.

In a sense, team Markov games are harder. A set of team Q-learners will converge to optimal play if the limit Q-functions have a unique maximum for each state. If not, even though all agents will converge on an equilibrium, if they converge on different equilibria, they can score arbitrarily poorly.

This is also true of a Q-learner among team Q-learners.

Claus and Boutilier (1998) examined simultaneous Q-learners in team games. They argue that two such learners converge to a Nash pair, although it need not be the optimal one. As far as I know, no formal proof has been presented. Nonetheless, Sen et al. (1994), Mundhe and Sen (2000) and others have successfully applied this technique to particular games, validating it as a reasonable approach.

In general games, if a Nash Q-learner's behavior converges, a Q-learner will learn a best response. However, as in the case with two team Q-learners, two Nash Q-learners need not converge to compatible equilibria and can score arbitrarily badly. A Nash Q-learner with a minimax Q-learner need not score well, but the minimax Q-learner will at least learn a 'safe' strategy.

6.2. Nash Q-learning

Nash Q-learning is a promising algorithm and has been executed with positive results on some small, but interesting, environments. The results described in this paper, however, undercut its theoretical justifications. As detailed next, any situation in which the use of Nash Q-learning is justified, minimax Q-learning or team Q-learning would appear to be a more sensible choice.

Bowling (2000) pointed out that Nash Q-learning is only guaranteed to converge if the Q-functions at every state either always has an adversarial equilibrium or always has a coordination equilibrium. As mentioned in Section 4, because of the possibility of games having both types of equilibria, the learner must know in advance which type of equilibrium to pick in each state. If a state is of the 'adversarial' type, using minimax Q-learning (Section 4.2) gives a stronger guarantee than Nash Q-learning on the amount of reward the agent will receive. If a state is of the 'coordination' type, using team Q-learning (Section 5.2) results in learning the same values as Nash Q-learning, but with a simpler computation.

Thus, even though Nash Q-learning can be applied in very general settings, in any case in which its use is formally justified — at present — minimax Q-learning or team Q-learning is preferable. This indicates that more theoretical work on Nash Q-learning could be quite beneficial.

6.3. Policy modeling

As is evident from the theorems cited in this paper, multiagent environments with multiple equilibria pose difficult problems for the convergence of learning algorithms. As Bowling and Veloso (2000) point out, if a game has multiple equilibria, the optimal policy must depend on the policies of the other agents. In this type of environment, opponent-independent algorithms like Nash Q-learning, minimax Q-learning, and team Q-learning are not sufficient to identify optimal behavior. Some dependence on the policies of other agents is necessary, and to do this, some assumptions must be made about how other agents will behave.

One reasonable assumption is that past behavior is a strong indicator of future behavior. Q-learning, as described earlier, implicitly uses this idea by choosing actions that maximize payoff as estimated by past interactions. It is also possible to model other agents more directly, for example as probabilistic stationary policies. Claus and Boutilier (1998) defined a joint-action Q-learning algorithm (JAL) that uses this idea to play team games, and Uther and Veloso (1997) used this approach in zero-sum games to good effect. The fictitious play approach in game theory, surveyed by Vrieze and Tijs (1982), also stems from this insight. Agent modeling would seem to be even more important in environments with more complex payoff structures. However, there is also some evidence that opponent modeling can be problematic in some circumstances (Sun & Qi, 2000).

A drawback of straightforward agent modeling is that it is purely reactive. In a sense, each agent is letting the other agents pick the equilibrium and then each learns a best response. It would be beneficial to identify a method that is partly opponent-independent, like Nash Q-learning, and partly opponent-sensitive, like Q-learning. The latter attribute could encourage equilibrium behavior, so the actions chosen fit well with the actions chosen by other agents. The former attribute could help encourage agents to select an equilibrium that results in high payoff. The hope is that this could reduce the probability of selecting 'optimal' actions that score poorly because agents don't agree on an equilibrium or jointly selecting actions that form a highly suboptimal equilibrium.

In sum, reinforcement learning is a powerful framework for studying learning in multiagent scenarios. Adversarial environments are well behaved in that optimal play can be guaranteed against an arbitrary opponent. Coordination environments are less well behaved, as strong assumptions need to be made about other agents to guarantee convergence to optimal behavior. In other types of environments, no value-function reinforcement-learning algorithms with guaranteed convergence properties are known. Nevertheless, the last few years have expanded our understanding of reinforcement-learning approaches and have helped clarify areas in which further study is needed.

Acknowledgements

Craig Boutilier, Michael Bowling, Junling Hu, Satinder Singh, Peter Stone, Ron Sun and Michael Wellman provided pointers to papers and other valuable help.

References

Barto, A. G., Sutton, R. S., & Watkins, C. J. C. H. (1991). Learning and sequential decision making. In: Gabriel, M., & Moore, J. (Eds.), Learning and computational neuroscience: foundations of adaptive networks, MIT Press, Cambridge, MA, Tech. Rep. 89-95, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1989.

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

Bertsekas, D. P. (1987). *Dynamic programming: deterministic and stochastic models*, Prentice-Hall, Englewood Cliffs, NJ.

Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In: Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK-96).

Bowling, M. (2000). Convergence problems of general-sum multiagent reinforcement learning. In: Proceedings of the Seventeenth International Conference on Machine Learning.

Bowling, M., & Veloso, M. M. (2000). An analysis of stochastic game theory for multiagent reinforcement learning. In: Technical Report CMU-CS-00-165, Computer Science Department, Carnegie Mellon University.

Claus, C., & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence.

Filar, J., & Vrieze, K. (1997). *Competitive Markov decision processes*, Springer-Verlag.

Howard, R. A. (1960). *Dynamic programming and Markov processes*, MIT Press, Cambridge, MA.

Hu, J. (1999). *Learning in Dynamic Noncooperative Multiagent Systems*, Department of Computer Science, University of Michigan, Ph.D. thesis.

Hu, J., & Wellman, M. P. (1998). Multiagent reinforcement learning: theoretical framework and an algorithm. In: Shavlik, J. (Ed.), Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann.

Hu, J., & Wellman, M. P. (2000). Experimental results on *Q*-learning for general-sum stochastic games. In: Langley, P. (Ed.), Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann.

Jaakkola, T., Jordan, M. I., & Singh, S. P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Comput*. *6*(6), 1185–1201.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: a survey. *J. Artificial Intell. Res. 4*, 237–285.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the Eleventh International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA.

Littman, M. L., & Szepesvári, C. (1996). A generalized reinforcement-learning model: convergence and applications. In: Saitta, L. (Ed.), Proceedings of the Thirteenth International Conference on Machine Learning.

Mundhe, M., & Sen, S. (2000). Evaluating concurrent reinforcement learners. In: Proceedings of the Fourth International Conference on Multiagent Systems, IEEE Press.

Owen, G. (1982). In: 2nd Edition, Game Theory, Academic Press, Orlando, FL.

Puterman, M. L. (1994). *Markov decision processes — discrete stochastic dynamic programming*, John Wiley, New York.

Sen, S., Sekaran, M., & Hale, J. (1994). Learning to coordinate without sharing information. In: Proceedings of the Twelfth National Conference on Artificial Intelligence.

Shapley, L. (1953). Stochastic games. *Proc. Natl. Acad. Sci. USA 39*, 1095–1100.

Singh, S., Jaakkola, T., Littman, M. L., & Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learn*. *39*, 287–308.

Sun, R., & Qi, D. (2000). Rationality assumptions and optimality of co-learning. In: Proceedings of PRIMA'2000, Lecture Notes in Artificial Intelligence, Springer-Verlag.

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

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: an introduction*, MIT Press.

Szepesvári, C., & Littman, M. L. (1999). A unified analysis of value-function-based reinforcement-learning algorithms. *Neural Comput*. *11*(8), 2017–2059.

Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and *Q*-learning. *Machine Learn*. *16*(3), 185–202.

Uther, W., & Veloso, M. (1997). *Adversarial reinforcement learning*, Unpublished manuscript.

van der Wal, J. (1981). In: Stochastic dynamic programming, Mathematical centre tracts, Vol. 139, Mathematisch Centrum, Amsterdam.

von Neumann, J., & Morgenstern, O. (1947). *Theory of games and economic behavior*, Princeton University Press, Princeton, NJ.

Vrieze, O. J., & Tijs, S. H. (1982). Fictitious play applied to sequences of games and discounted stochastic games. *Int*. *J*. *Game Theory* *11*(2), 71–85.

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

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learn*. *8*(3), 279–292.