Multiagent Reinforcement Learning in the Iterated Prisoner's Dilemma

Tuomas W. Sandholm* **and** **Robert H. Crites**†

{sandholm, crites}@cs.umass.edu

University of Massachusetts at Amherst

Computer Science Department

Amherst, MA 01003

Tel. 413-545-0675, Fax. 413-545-1249

Abstract

Reinforcement learning (RL) is based on the idea that the tendency to produce an action should be strengthened (*reinforced*) if it produces favorable results, and weakened if it produces unfavorable results. Q-learning is a recent RL algorithm that does not need a model of its environment and can be used on-line. Therefore it is well-suited for use in repeated games against an unknown opponent. Most RL research has been confined to single agent settings or to multiagent settings where the agents have totally positively correlated payoffs (team problems) or totally negatively correlated payoffs (zero-sum games). This paper is an empirical study of reinforcement learning in the iterated prisoner's dilemma (IPD), where the agents' payoffs are neither totally positively nor totally negatively correlated.

*Supported by ARPA contract N00014-92-J-1698. The content does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred. Also supported by the Finnish Culture Foundation, Honkanen Foundation, Ella and George Ehrnrooth Foundation, Finnish Science Academy, Leo and Regina Wainstein Foundation, Finnish Information Technology Research Foundation, and Jenny and Antti Wihuri Foundation.

†Supported by Air Force Office of Scientific Research Grant F49620-93-1-0269.

RL is considerably more difficult in such a domain. This paper investigates the ability of a variety of Q-learning agents to play the IPD game against an unknown opponent. In some experiments, the opponent is the fixed strategy Tit-for-Tat, while in others it is another Q-learner. All the Q-learners learned to play optimally against Tit-for-Tat. Playing against another learner was more difficult because the adaptation of the other learner created a nonstationary environment, and because the other learner was not endowed with any *a priori* knowledge about the IPD game such as a policy designed to encourage cooperation. The learners that were studied varied along three dimensions: the length of history they received as context, the type of memory they employed (lookup tables based on restricted history windows or recurrent neural networks that can theoretically store features from arbitrarily deep in the past), and the exploration schedule they followed. Although all the learners faced difficulties when playing against other learners, agents with longer history windows, lookup table memories, and longer exploration schedules fared best in the IPD games.

Keywords: Multiagent learning, reinforcement learning, machine learning, prisoner's dilemma, recurrent neural network, exploration.

1 Introduction

Research in machine learning has focused primarily on *supervised learning*, where a "teacher" provides the learning system with a set of training examples in the form of input-output pairs. The usual goal of the learning system is to implement an input-output mapping that generalizes well to inputs outside the training set. *Reinforcement learning* (RL) is applicable in cases where the learning system is not provided with a target output for each input, but instead must select an output for which it receives a scalar evaluation. RL is more difficult than supervised learning since it requires exploration, that is, finding the best output for any given input. It applies naturally to the case of an autonomous agent which receives sensations as inputs from its environment, and selects actions as outputs with the goal of affecting its environment in a way that maximizes utility. This framework is appealing from a biological point of view, since an animal has certain built-in preferences (such as pleasure or pain), but generally has no external signal telling it the best action for any given situation.

RL is based on the idea that the tendency to produce an action should be strengthened (*reinforced*) if it produces favorable results, and weakened if it produces unfavorable results. RL tasks can be divided naturally into two types. In non-sequential tasks, the agent must learn a mapping from situations to actions that maximizes the expected immediate payoff. Sequential tasks are more difficult because the actions selected by the agent may influence its future situations and thus its future payoffs. In this case, the agent interacts with its environment over an extended period of time, and it needs to evaluate its actions on the basis of their long-term consequences. Sequential tasks involve a credit assignment problem: a whole sequence of actions takes place before the long-term consequences are known. Credit for the consequences has to be allocated among the actions in the sequence. This is difficult because actions in the sequence may have different values with respect to the consequences. Furthermore, the value of an action may depend on the other actions in the sequence.

From the perspective of control theory, RL algorithms are techniques for addressing stochastic optimal control problems. The agent is the controller, and the environment is the system to be controlled. The objective is to maximize some performance measure over time. Given a perfect model of the environment, these problems can be solved in principle using dynamic programming (DP) algorithms, although the time required for large problems may make their solution infeasible. Q-learning (Watkins, 1989) is a recent RL algorithm that approximates DP incrementally without requiring a model of the environment. Unlike traditional DP, it can be used to improve performance *on-line* while the agent and the environment interact (Barto et al., 1995).

The Q-learning algorithm works by estimating the values of state-action pairs.

The value $Q(s,a)$ is defined to be the expected discounted sum of future payoffs obtained by taking action a from state s and following an optimal policy thereafter. Once these values have been learned, the optimal action from any state is the one with the highest Q-value. After being initialized to arbitrary numbers, Q-values are estimated on the basis of experience as follows:

1. From the current state s, select an action a. This will cause receipt of an immediate payoff r, and arrival at a next state s'.

2. Update Q(s,a) based on this experience as follows:

$$\Delta Q(s,a) \quad = \quad \alpha[r + \gamma \max_b Q(s',b) - Q(s,a)] \tag{1}$$

where α is the learning rate and $0 \leq \gamma < 1$ is the discount factor.

3. Go to 1.

This algorithm is guaranteed to converge to the correct Q-values with probability one if the environment is stationary and Markovian[1], a lookup table is used to store the Q-values, every state-action pair continues to be visited, and the learning rate is decreased appropriately over time. Q-learning does not specify which action to select at each step. However, no action should be neglected forever. In practice, a method for action selection is usually chosen that will ensure sufficient exploration while still favoring actions with higher value estimates. The Boltzmann distribution provides one such method, where the probability of selecting action a_i in state s is

$$p(a_i) = \frac{e^{Q(s,a_i)/t}}{\sum_a e^{Q(s,a)/t}} \tag{2}$$

where t is a computational temperature parameter that controls the amount of exploration. It is usually annealed, i.e., lowered gradually over time.

[1] An environment is Markovian if the state-transition probabilities from the current state only depend on the current state and the action taken in it, not on the history that led to the current state.

Multiagent RL has been studied since at least the 1950's. The work of Tsetlin (1973) on the collective behavior of learning automata provides an early example. See Narendra and Thathachar (1989) for an excellent introduction to learning automata. In zero-sum games, learning automata converge to the game's solution, and in identical-payoff games, they converge to an equilibrium point which is a local optimum. Traditional learning automata do not use any context in their decision making. Barto and Anandan (1985) introduced *associative* learning automata that do use context. Barto (1985) applied associative learning automata to identical-payoff games with promising results. All of these learning automata were designed for non-sequential tasks.

Algorithms designed for sequential tasks have been studied mainly within a single agent context (Barto et al., 1983), (Sutton, 1988), (Watkins, 1989). Some of the newer work has applied reinforcement learning methods such as Q-learning (Watkins, 1989) to multiagent settings. In many of these studies the agents have had independent or rather easy tasks to learn. On the other hand, the theoretical guarantees about Q-learning do not apply in multiagent settings because the state of the other agents cannot be observed and because the environment is nonstationary due to the other agents' learning. Sen et al. (1994) describe 2-agent block pushing experiments, where the agents try to make the block follow a line by independently applying forces to it. Tan (1993) reports on grid-world predator-prey experiments with multiagent reinforcement learning, focusing on the sharing of sensory information, policies, and experience among the agents. Unfortunately, just slightly harder predator-prey problems have uncovered discouraging results (Sandholm and Nagendraprasad, 1993). On the other hand, Bradtke (1993) describes encouraging results in applying multiagent reinforcement learning to efficiently damp out disturbances of a flexible beam. Crites (1994) proposes applying multiagent RL algorithms to elevator dispatching, where each elevator car would be controlled by a separate agent. Littman and Boyan (1994) describe a distributed RL algorithm (related to the Bellman and Ford algorithm (Bertsekas and Tsitsiklis, 1989)) for packet routing, using a single, centralized Q-function, where each state entry in the Q-function is assigned to a node in the network which is responsible for storing and updating the value of that entry. This differs from the work described in this paper, where an entire Q-function, not just a single entry, is stored by each agent. Littman (1994) experiments with Q-learning agents that try to learn a mixed strategy that is optimal against the worst possible opponent in a zero-sum 2-player game. Markey (1993) uses a team of Q-learning agents to control a vocal tract model with ten degrees of freedom. Weiß(1993) presents Bucket Brigade based sequential reinforcement learning experiments in a simple blocks world problem, where cooperative agents with partial views share a goal but do not know what the goal is. Other multiagent learning research uses purely heuristic algorithms for complex real-world problems such as learning coordination strategies (Sugawara and Lesser, 1993) and communication strategies (Kinney and Tsatsoulis, 1993) with varying success. Shoham and Tennenholtz (1993) describe a simple learning algorithm called Cumulative Best Response that performs well in identical-payoff settings but performs poorly in the IPD. Despite some weak theoretical guarantees of eventual cooperation, in practice, agents using this learning rule usually fail to reach cooperation in hundreds of thousands of iterations. Ashlock et al. (1995) use a criterion filtering algorithm that is closely related to RL to learn the expected payoffs associated with different IPD game partners. Their work applies RL to partner selection while this paper applies RL to action selection within an IPD game. Samuel (1959) pioneered the application of RL to zero-sum games with his checkers playing program. More recently, Tesauro's (1992) RL-based Backgammon program has achieved strong master level play.

Almost all of the research described above investigates settings where the agents have totally positively correlated payoffs (team problems) or totally negatively correlated payoffs (zero-sum games). This paper attempts to fill that gap by studying RL in the IPD, where the agents' payoffs are neither totally positively nor totally negatively correlated. The experiments in this paper confirm that multiagent RL is especially difficult in such a setting. This paper investigates the ability of a variety of Q-learning agents to play the IPD game against an unknown opponent. In some experiments, the opponent is the fixed strategy Tit-for-Tat, while in others it is another Q-learner. All the Q-learners learned to play optimally against Tit-for-Tat. Playing against another learner was more difficult because of the other learner's non-stationary behavior, and because the other learner was not endowed with any *a priori* knowledge about the IPD game such as a policy designed to encourage cooperation. The learners that were studied varied along three dimensions: the length of history they received as context, the type of memory they employed (lookup tables or recurrent neural networks that can theoretically store features from arbitrarily deep in the past), and the exploration schedule they followed. Although all these learners faced difficulties (due to their self-interested nature) when playing against another learner, agents with longer history windows, lookup table memories, and longer exploration schedules fared best in the IPD game.

The remainder of the paper is organized as follows. Section 2 provides an overview of the IPD game. Section 3 describes the learning agents in more detail. The following six sections describe the experiments. The last section contains conclusions and suggestions for future research.

2 Prisoner's dilemma

The 2-agent *prisoner's dilemma* game is an abstraction of social situations where each agent is faced with two alternative actions: *cooperating*, i.e., doing the socially responsible thing, and *defecting*, i.e., acting according to self-interest regardless of how harmful this might be to the other agent. Characteristically, each agent is better off defecting regardless of the opponent's choice, but the sum of the agents' payoffs is maximized if both agents choose to cooperate—thus the dilemma. In game theoretic terms, defecting is a dominant strategy of the game and so the defect-defect action combination is the only *dominant strategy equilibrium* (and therefore also the only *Nash equilibrium*). On the other hand, *social welfare* is maximized at the cooperate-cooperate action combination, if social welfare is defined to be the equiweighted sum of the agents' payoffs. Table 1 shows a payoff matrix for a 2-player game, where each agent has two possible actions. The payoff matrix describes a PD game if the

		column player	
		cooperate (C)	defect (D)
row	cooperate (C)	R (= 0.3)	S (= 0.0)
player	defect (D)	T (= 0.5)	P (= 0.1)

Table 1: *Payoff matrix for the row player. The particular values in parenthesis are the ones used in the experiments of this paper. The column player may have different payoffs as long as they define a PD game. In this paper, both players had the same payoff matrix.*

following inequalities hold:

$$T > R > P > S \tag{3}$$

and

$$2R > T + S > 2P. \tag{4}$$

The PD game is a *noncooperative game*: no pregame negotiation is allowed, the agents cannot bindingly commit to any action, no enforced threats can be made, and no transfer of payoff is possible.

In practical situations, agents often encounter each other more than once. Correspondingly, some social interactions can be modeled by repeated PD games. This *supergame* of the PD game is called the *iterated prisoner's dilemma* (IPD) game. In supergames, an agent's policy (strategy) is a mapping from the entire history (all of its own and its opponent's moves) to an action. In a *pure* strategy, the mapping is deterministic (nonprobabilistic). In a *mixed* strategy, an agent chooses its action stochastically from a distribution that is determined by the history. If an agent uses a pure strategy, its move history is redundant since it can be reconstructed from its strategy and its opponent's move history.

In an IPD game, it may be beneficial even for a selfish agent to cooperate on some iterations in the hope of soliciting cooperation from its opponent. If the number of iterations of the PD game in an IPD game is known, then the last iteration becomes the standalone PD game. So in the last iteration each agent is motivated to defect. Because both agents know that the opponent is going to defect on the last round, they have no motivation to cooperate on the second to last round either. This backward induction can be carried out all the way to the beginning of the interaction. Thus in some sense it is rational to defect throughout the sequence (Luce and Raiffa, 1957) (and thus, paradoxically, some irrational agents will do better than rational ones). Because fixed horizon IPD games have this characteristic, this paper focuses on IPD games with an indefinite horizon, i.e., the agents do not know how many iterations are still to come. The goal of an agent at iteration n is to select actions that will maximize its discounted return:

$$\sum_{i=n}^{\infty} \gamma^{i-n} r_i$$

where r_i is the reward or payoff received on iteration i, and $0 \leq \gamma < 1$ is the discount factor.

Generally, describing an intelligent strategy for a supergame is difficult because arbitrarily long input histories must be considered. There are two main approaches that strategy designers have used to address this problem in practice:

- Use only a fixed number of previous moves as the context upon which the choice of next action is based, and
- Iteratively keep a tally of some (numeric) features that provide an abstract characterization of the entire history.

Two classical examples of the first approach are the pure strategies called Tit-for-Tat (TFT) and PAVLOV. A player using TFT cooperates on the first move and then does whatever its opponent did on the previous move. Despite its simplicity, TFT has proven very successful in open IPD tournaments and evolutionary IPD experiments (Axelrod, 1984). PAVLOV cooperates if and only if the agents chose the same action on the previous move. In evolutionary IPD games with certain random disturbances PAVLOV outperforms TFT (Nowak and Sigmund, 1993). An example of the second approach is to compute at each time step the opponent's discounted cumulative score. A strategy has the desired property of being *collectively stable* if and only if it defects when that score exceeds a threshold (Axelrod, 1984). Both approaches to the problem of growing context suffer from the *hidden state problem*: the first approach ignores the older history, and the second approach can only give an abstraction of the true state—important details may be lost. Secondly, it is a nontrivial task to identify meaningful features.

There is no single best strategy for the IPD game. Which strategy is best depends on the opponent's strategy, which the player obviously does not know. The *folk theorem of repeated games* (Kreps, 1990), (Fudenberg and Tirole, 1991) states that any feasible payoffs that give each agent more than its minimax value can be supported in subgame perfect Nash equilibrium as long as the discount factor γ is sufficiently high[2]. TFT was chosen as an opponent for the learning players not only because it

[2] An agent's minimax payoff is the payoff that the agent gets when it uses its best strategy against

has performed well in IPD tournaments, but also because the optimal way to play against TFT is completely known. There are three different optimal ways to play against TFT depending on the discount factor γ (Axelrod, 1984):

- always cooperate; discounted return $V_c = \frac{R}{1-\gamma}$
- alternate between defecting and cooperating; discounted return $V_a = \frac{T+\gamma S}{1-\gamma^2}$
- always defect; discounted return $V_d = T + \frac{\gamma P}{1-\gamma}$.

For the payoff matrix of the PD game used in this paper ($T = 0.5, R = 0.3, P = 0.1, S = 0.0$), $V_c = \frac{0.3}{1-\gamma}$, $V_a = \frac{0.5}{1-\gamma^2}$, and $V_d = 0.5 + \frac{0.1\gamma}{1-\gamma}$. In this case, the agent playing against TFT should always cooperate if $\gamma \geq \frac{2}{3}$, alternate between defection and cooperation if $\frac{1}{4} \leq \gamma < \frac{2}{3}$, and always defect if $\gamma < \frac{1}{4}$. For the most studied payoff matrix of the PD game ($T = 5, R = 3, P = 1, S = 0$) the thresholds are the same, because that payoff matrix is just a rescaling. Note that each of the three ways of playing can be achieved by a number of strategies. For example, cooperation with TFT is realized by another TFT strategy or by a strategy that always cooperates no matter what the opponent does.

3 Players learning by reinforcement in the IPD

A number of different types of Q-learning agents were generated to challenge TFT and each other in IPD games. The whole learning session was one long trial, i.e., the agent had only one chance to learn and evaluation took place during the last part of the trial. Unless otherwise stated, the experiments were run with learning rate $\alpha = 0.2$ and discount factor $\gamma = 0.95$. The value of α was chosen experimentally to enhance learning of cooperation. The γ in the Q-learning algorithm is by definition

its opponent's strategy that is worst for the agent. In the IPD, this corresponds to receiving a stream of P's (i.e., a stream of 0.1's in the example case with a discounted return of $\frac{0.1}{1-\gamma}$).

the same γ as in the IPD game. Its value was chosen to be so high because higher values tend to promote cooperation (Axelrod, 1984).

The agents differed in the way they stored Q-values and in their exploration policies. The next two sections describe the tested alternatives in detail.

3.1 Storing Q-values: lookup tables vs. recurrent neural nets

If the state is considered to be the entire history, an agent is faced with a stationary environment (the other agent). An agent's learning method (algorithm and parameterization) is part of the agent's strategy in the IPD game. At each decision point, the state increases in dimension with respect to the state at the previous decision point, so each state is visited at most once. Therefore the theoretical convergence results of Q-learning do not apply. Similarly, because each state is visited only once in a single supergame, an agent cannot distinguish whether its opponent is using a pure or a mixed strategy. Thus the agent perceives its environment as deterministic.

If the "state" is viewed as some window of previous moves (call this *sensation*, Figure 1), the agent may be faced with a nonstationary stochastic environment. The environment is *nonstationary* if the opponent has different action probabilities than at a previous time when the same decision context (window of previous moves or some features that capture an abstraction of the entire history) of the agent occurred. Nonstationarity also precludes the convergence proof. *Stochasticity* refers to the case where the opponent's action probabilities are unaltered, but the chosen action may be different. Ignoring exploration, which will be discussed in the next section, there are two possible reasons why the opponent's action probabilities may differ from what they were earlier at the same decision context of the agent. First, the opponent may be using a larger decision context (more previous moves or different features of the history) in which case the agent cannot distinguish between two states that the opponent can distinguish. Once the system has reached stability this is a question of stochasticity with respect to the agent. Secondly, the opponent's learning method may have changed the opponent's mapping from its sensations to its Q-values (a question of nonstationarity).

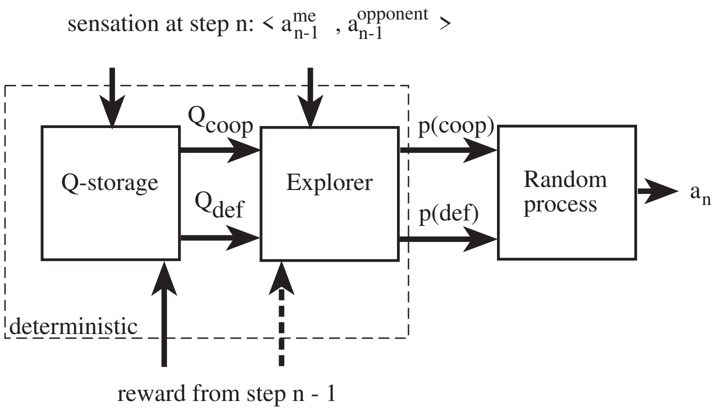

Figure 1: *The architecture of a Q-learning agent for the PD game. Given the current sensation, Q-values for each action are used to determine action probabilities. These probabilities are then used to select an action. Q-values are updated based on the rewards received.*

This hidden state problem was addressed in two different ways. The first set of agents used lookup tables to store their Q-values. The sensations of these agents were limited to w previous moves—i.e. w of the agent's own moves and w of the opponent's moves. For example, given a window of only the last move ($w = 1$), four different sensations would be possible (CC, CD, DC, and DD). For each possible sensation, two Q-values would need to be stored (corresponding to actions C and D). Conceptually, these agents ignore the older history.

The second set of agents had the same input sensations, but they stored the Q-values in a recurrent neural network that can (at least in theory) store information of arbitrarily old actions and automatically learn which history features are important to keep track of, Figure 2. Sensations were presented to the net in four bits because a unary encoding resulted in faster learning than a two bit binary encoding. The first bit was on if the opponent's previous action was cooperate, and the second bit was on if the action was defect. The third bit was on if the agent's own previous action was cooperate, and the fourth bit was on if its action was defect. We had a separate net for both actions, which has been shown empirically to enhance learning speed in certain situations (Lin, 1993). Each net was constructed along the lines of Elman (1990): the net was a normal feedforward net except that the hidden-unit outputs were copied into context units, whose activations were fed back to the normal hidden-units on the next forward sweep. The copy connections from the normal hidden-units to the context units were fixed to one. This allowed the use of the standard backpropagation learning algorithm. In his experiments, Lin (1993) did not fix the copy connections, and was thus forced to do backpropagation through time. He did this by exhaustive unfolding in time in batch mode (i.e., weights were updated after each entire trial), which would have been impossible in the case of only one long trial as opposed to his many short trials of length 30 steps. A real-time version of backpropagation through time exists which does not require the fixing of weights (Williams and Zipser, 1989; Hecht-Nielsen, 1991; Haykin, 1994). It was not used here because it is computationally intensive and its derivation assumes that the new inputs to the net are not a function of the old outputs, which is not true in control tasks such as this.

Each network had four input units, three normal hidden-units, three context (hidden) units and one output unit. The normal hidden-units were logistic units with outputs in the range between 0 and 1. The input units and context units did not do any processing—they simply passed on their input. The output unit (representing the estimated Q-value) was a linear unit. The number of hidden-units was chosen based on common practice and on the experimental results of Lin (1993). The normal hidden-units and the output unit received input from a bias unit (not shown in Figure 2). The net was trained using the error backpropagation algorithm (Rumelhart

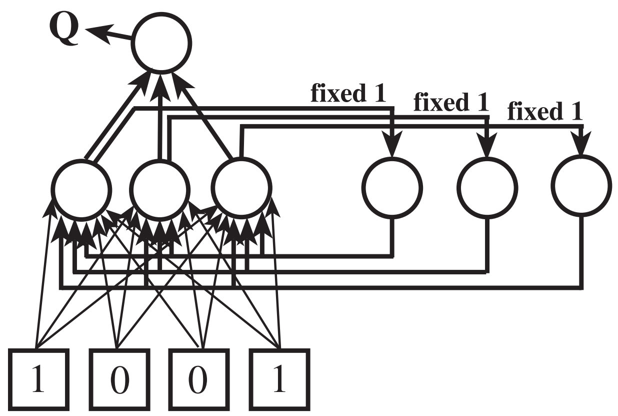

Figure 2: *The recurrent neural net acting as a Q-value storage. Each action had a separate net.*

et al., 1986; Hecht-Nielsen, 1991) with the learning rate for logistic units set to 0.3 and their momentum set to 0.05. The learning rate for the linear output unit was set to 0.01. All of these parameters were chosen experimentally to enhance learning.

The output of a unit in the recurrent net depends not only on the inputs of the net, but also on the outputs of units on the prior forward pass. Therefore, the updating of Q-values and the choice of an action must be done carefully so that both nets (one for each action) get exactly one forward pass per PD game iteration. For the action that was not taken in the previous time step this is simple, because no Q-value backup is required. To compute the new Q-estimate, one forward pass is done with the new sensation as input. For the action that was chosen on the previous time step, the first step is to save the activations. Then a forward pass with the new sensation is performed to determine this action's alternative Q-value to back up (the Q-update rule will choose the highest alternative over all actions). Next the activations are restored so that the net will be in the state that it was before the forward pass. Now the Q-update is done by changing the weights in the net by a backward pass. Last, a forward pass is done with the new weights and the new sensation to give the new

Q-estimate for this action. This Q-estimate is used to choose the next action.

3.2 Exploration methods

Agents must explore the consequences of their actions in order to be able to choose good actions later. It is not sufficient for an agent to always choose the action that it thinks best. It should try other actions as well to identify the environment—especially in potentially nonstationary situations.

Even if the players use the same decision context (fixed number of previous moves or some abstraction of the entire history) and the opponent's Q-values have not changed, due to the opponent's exploration, its action probabilities may be different at a certain decision context of the agent from what they were at the same decision context of the agent before. In general, an agent's exploration policy (mapping from Q-values to action probabilities) can be a function of the entire history (true state). So, with respect to the agent's limited decision context, the opponent's exploration policy may be nonstationary. In this paper, each learning agent's exploration policy was a function of the length of the history so far, not a function of the specific events in the history. The length of the history was used to decrease the temperature for Boltzmann exploration. Specifically, an agent's probability of selecting action a_i from state s was

$$p(a_i) = \frac{e^{Q(s,a_i)/t}}{\sum_{a \in \{C,D\}} e^{Q(s,a)/t}} \tag{5}$$

where the temperature t was a function of the number n of PD games played so far:

$$t = 5 \cdot 0.999^n. \tag{6}$$

If $t < 0.01$ then no exploration was performed, i.e., the action with the highest Q-value estimate was chosen with certainty. The constants for the annealing schedule $(5, 0.999, 0.01)$ were chosen experimentally. 5 and 0.01 are specific to the range of Q-values defined by the payoff matrix. In this domain the true Q-values (discounted payoffs) range between 0 (getting the sucker's payoff all the time) and 10 (defecting on a cooperator all the time). An annealing schedule where t decreased linearly with n was also tested but was not as effective.

In all, there are four distinct reasons why the opponent may act differently at a certain decision context of the agent than it did earlier at the same decision context of the agent:

- The opponent may be using a different decision context than the agent (e.g. longer time window or different history features) in which case the agent cannot distinguish between two states that the opponent can. Once the system has reached stability this is a question of stochasticity with respect to the agent.
- The opponent may have learned, i.e., its mapping from decision contexts to Q-values may have changed. This may make the environment appear nonstationary to the agent.
- The opponent's exploration policy (mapping from Q-values to action probabilities) may have changed. The environment may thus appear nonstationary to the agent.
- The opponent's stochastic action selector may choose a different action even though its action probabilities have not changed. This makes the environment appear stochastic to the agent.

A player cannot distinguish which of these effects is causing the opponent to act differently. If the same decision context occurs multiple times, the agent may be able to differentiate between stochasticity and nonstationarity. The convergence proof of Q-learning applies to stochastic environments, but not to simultaneous learning due to nonstationarity.

4 Experiment 1: Learning to play against Tit-for-Tat

The first experiment was designed to see how the different learning agents performed against a player that used a fixed strategy (TFT). For $\gamma = 0.95$, lookup table based Q-learning agents using the last move as the sensation consistently learned to cooperate with TFT for a variety of parameter settings. The recurrent net based Q-learners also learned to cooperate with TFT as did a learner that had the same neural net architecture as the recurrent net but with no context units and no feedback connections. The lookup table based learners learned to cooperate with TFT in thousands of iterations, while the net based players required tens of thousands of iterations.

The next question was whether an agent could learn the optimal play against TFT for other values of γ as well. The lookup table based Q-learner with the Boltzmann exploration method was selected. At each setting of γ, 100 IPD games were run with 100,000 iterations each. For $\gamma = 0.05$, $\gamma = 0.1$, $\gamma = 0.15$, and $\gamma = 0.2$, the agent consistently learned to defect against TFT. For $\gamma = 0.25$, $\gamma = 0.3$, $\gamma = 0.35$, $\gamma = 0.4$, $\gamma = 0.45$, $\gamma = 0.5$, $\gamma = 0.55$, $\gamma = 0.6$, and $\gamma = 0.65$, the agent learned to alternate between defecting and cooperating. For $\gamma = 0.7$, $\gamma = 0.75$, $\gamma = 0.8$, $\gamma = 0.85$, $\gamma = 0.9$, and $\gamma = 0.95$, the agent learned to cooperate with TFT. Thus the agent learned to play optimally against TFT in every one of the hundred IPD games at each setting of γ.

The Q-learning mechanism, though relatively slow, works extremely well against stationary policies such as TFT, which take into account a short window of previous moves. Playing against an agent with a stationary policy is analogous to single agent learning, because the learning agent perceives a stationary environment. The next sections discuss harder cases where both agents are learning simultaneously.

5 Experiment 2: Both players learning simultaneously

Two types of learning agents were studied in the context of simultaneous learning. They differed in how they stored their Q-value estimates. One type used lookup tables with Boltzmann exploration (LB) and the other type used recurrent networks with Boltzmann exploration (RB). All three pairings of these agents were tested (an agent could play another agent similar to itself). Each pairing consisted of 100 IPD games of 300,000 iterations each. The way Boltzmann exploration was implemented (see Section 3.2), each agent stopped exploring after 6212 iterations. Learning still continued from that point on, however. The system always appeared to reach a stable state within the 300,000 iterations. In the result tables, the first four columns describe alternative final states (only final states that occurred are shown). They show what percentage of the last 100 iterations of an IPD game were of a certain type. CC means that both cooperated, CD means that the first agent cooperated, but the second defected, DC means that the first agent defected, but the second cooperated, and DD means that both defected. The fifth column shows how many times of the 100 IPD games played each of the final states occurred ($\alpha = 0.2$), and column six shows the results for a large α (1.0). The seventh column indicates what happened when the exploration period was extended to 62,143 iterations by increasing the exploration annealing factor from 0.999 to 0.9999 (at $\alpha = 0.2$).

Final states often included loops, e.g., CC, CD, CC, CD, ... The loop length for lookup table learners is bounded above by 2^{m+n}, where m is the number of plays that one agent remembers and n is the number of plays that the other agent remembers. In theory, recurrent net players can have arbitrarily long loops due to their memory, but in practice that did not tend to occur.

When two lookup table players played each other, one never totally took advantage of the other (e.g. CD, CD, CD, ...), but asymmetric loops did occur (e.g. CC, CD, CC, CD, ...). Between recurrent net players neither total advantage taking nor asymmetric loops occurred. When LB and RB played, asymmetric loops only occurred to the advantage of the lookup table player.

Increasing α from 0.2 to 1.0 enhanced cooperation in the games RB-RB, but hindered cooperation in the games LB-LB and LB-RB.

Lookup table, Boltzmann expl. vs. lookup table, Boltzmann expl.						
CC	CD	DC	DD	% trials ($\alpha = 0.2$)	% trials ($\alpha = 1.0$)	% trials (anneal .9999)
0.0	0.0	0.0	100.0	68	75	
100.0	0.0	0.0	0.0	25	1	24
50.0	50.0	0.0	0.0	4	8	14
50.0	0.0	50.0	0.0	2	7	13
0.0	50.0	50.0	0.0	1	2	49
50.0	0.0	0.0	50.0		7	

Lookup table, Boltzmann expl. vs. recurrent net, Boltzmann expl.						
CC	CD	DC	DD	% trials ($\alpha = 0.2$)	% trials ($\alpha = 1.0$)	% trials (anneal .9999)
0.0	0.0	0.0	100.0	86	98	86
50.0	0.0	50.0	0.0	9		4
100.0	0.0	0.0	0.0	5		10
0.0	50.0	50.0	0.0		1	
50.0	0.0	0.0	50.0		1	

Recurrent net, Boltzmann expl. vs. recurrent net, Boltzmann expl.						
CC	CD	DC	DD	% trials ($\alpha = 0.2$)	% trials ($\alpha = 1.0$)	% trials (anneal .9999)
0.0	0.0	0.0	100.0	98	89	88
100.0	0.0	0.0	0.0	2	11	11
50.0	0.0	0.0	50.0			1

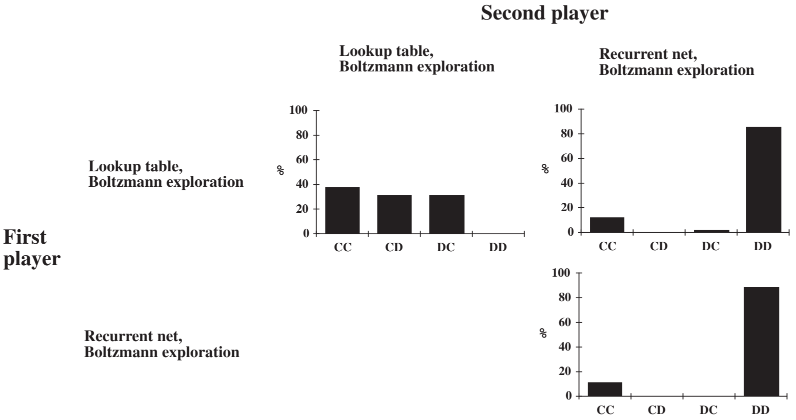

Figure 3: *Percentages of plays at exploration annealing rate 0.9999, ($\alpha = 0.2, \gamma = 0.95$).*

The sum of the agents' payoffs was much higher in the game LB-LB than in LB-RB or RB-RB (see Figure 3). If even one of the players used a recurrent neural net as its Q-value storage, the outcome of the game was significantly less cooperative. This somewhat surprising result may be because the neural net players need more training examples than the lookup table players, because they are performing generalization in addition to the basic Q-learning, and are learning which history features to keep track of. Thus, neural net players may require a longer exploration phase than lookup table players. Naturally, there is also a chance that some other network topology, some other learning algorithm, or some other learning rate and momentum parameters for backpropagation would have been more appropriate, although the parameters for these experiments were experimentally chosen to enhance cooperation.

6 Experiment 3: No exploration in simultaneous learning

The impact of turning off the Boltzmann exploration was also analyzed. A lookup table learner that did not explore (LN) and a recurrent net learner that did not explore (RN) were tested against LB, against RB, against each other and against themselves. The non-exploring agents always picked the action with the highest current Q-value estimate. The results of games where at least one agent does no exploration depend heavily on the initialization of the Q-values. For example, if all Q-values are initialized with the same negative number, a non-exploring agent will always pick the action that it first picked in that state. This is because the Q-value corresponding to that action will be reinforced by a non-negative number (and thus exceed any other action's Q-value in that state), because the payoffs in the game are non-negative. In the experiments, the Q-values were initialized randomly from a uniform distribution from 0 to 1.

In the lookup table games LB-LN and LN-LN, neither player ever totally took advantage of the other, but asymmetric loops occurred to either player's advantage—more often to the the advantage of the exploring LB. The loops were longer (up to 4 plays) and more frequent than among exploring players. On some runs, RB could totally take advantage of RN, but never of another RB. In RN-RN, either player could be totally taken advantage of. Asymmetric loops occurred among recurrent net players when at least one of them did not explore. These loops were always to the advantage of the exploring player. When LN and RN played, asymmetric loops occurred either way, but only the recurrent net player could be totally taken advantage of. LN often had unbeneficial asymmetric loops against RB but few runs showed the reverse. LB often took advantage of RN, but rare asymmetric loops occurred in RN's favor.

The results suggest that exploration is crucial to avoid being taken advantage of by an exploring opponent. If neither agent explores, the initialization of the Q-values determines the outcome. In such cases, cooperation occurs quite frequently. The sum of the agents' payoffs was highest in the game LB-LB, where both agents explored, see Figure 3. The sum was lowest in the games LB-RB and RB-RB, where both agents again explored, Fig. 3. In games where at least one agent did not explore, the sum was between these extremes.

7 Experiment 4: Extending exploration in simultaneous learning

This experiment analyzed the effect of extending the exploration process. The hypothesis is that agents will learn collectively better strategies if they are allowed to explore the system more thoroughly. This is not obvious in multiagent learning, because an agent's exploration introduces nonstationarity and stochasticity in the other agent's learning environment. Section 5 illustrated that changing the annealing factor of the exploration process from 0.999 to 0.9999 often encouraged more cooperative outcomes. This effect was strongest with lookup table learners. This section presents experiments where two such learners play each other, but the annealing factor is increased further to 0.99999, 0.999999 and 0.9999999. With the extended exploration process, learning takes more iterations. The annealing factors 0.999, 0.9999, 0.99999, 0.999999 and 0.9999999 correspond to 6212, 62143, 621458, 6214605, and 62146078 iterations of exploration respectively. To allow for that, the maximum number of iterations was changed from 300,000 to 900,000 (for 0.99999), to 6,500,000 (for 0.999999), and to 62,400,000 (for 0.9999999). This was in order to allow roughly 300,000 iterations of learning after the exploration had ceased in each case.[3] The learning rate α was 0.2, and γ was 0.95. 100 experiments were run for each setting of the exploration annealing rate (except 0.9999999 where only 35 experiments were run due to computational complexity).

Lookup table, Boltzmann expl. vs. lookup table, Boltzmann expl.								
CC	CD	DC	DD	.999	.9999	.99999	.999999	.9999999
0.0	0.0	0.0	100.0	68				
100.0	0.0	0.0	0.0	25	24	6	3	2.9 (1 case)
50.0	50.0	0.0	0.0	4	14	39	45	48.6 (17 cases)
50.0	0.0	50.0	0.0	2	13	33	52	48.6 (17 cases)
0.0	50.0	50.0	0.0	1	49	21		
33.3	33.3	33.3	0.0			1		

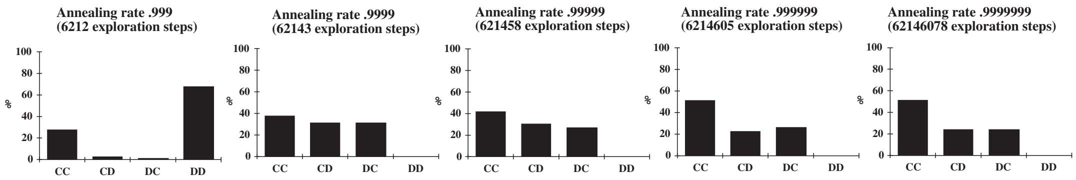

Figure 4: *Percentages of different plays for varying exploration annealing rates, (*$\alpha = 0.2, \gamma = 0.95$*).*

Defect-defect plays disappeared entirely as the exploration was prolonged even slightly, see Figure 4. Surprisingly, however, cooperate-cooperate plays also disappeared and gave way to asymmetric loops of length two. One half of the loop was CC and the other was CD (or symmetrically DC). Another interesting phenomenon was the fact that CD-DC loops occurred more frequently as the annealing factor for exploration was increased to 0.9999, but they became rarer as the annealing factor

[3] To make sure that the results were not overly sensitive to the amount of learning after exploration, 100 tests were run (not shown in the table) with over 500,000 iterations after exploration, with nearly identical results.

was further increased to 0.99999, and they disappeared as the annealing factor was increased to 0.999999 and 0.9999999. It appears that in the limit of extending the annealing schedule, CC plays occur 50% of the time, CD plays 25% of the time, and DC plays 25% of the time. The average payoff in each single stage game increases monotonically—with mostly diminishing returns—with longer exploration schedules and appears to approach 0.275 (Fig. 5) which corresponds to the distribution of plays mentioned above.

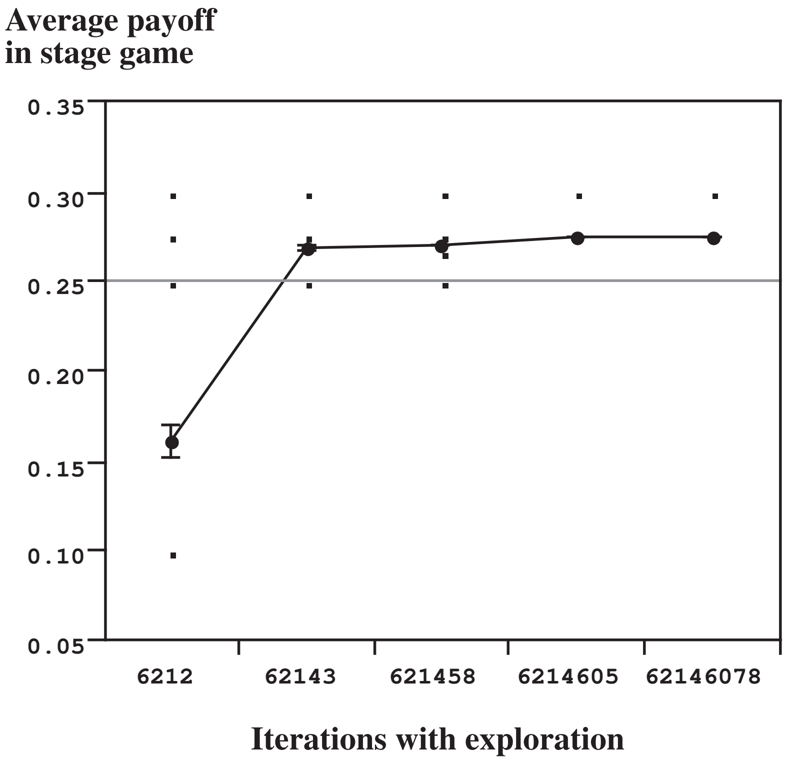

Figure 5: *The average payoff within an iteration as a function of exploration length. Different payoffs are marked by dots. Error bars are presented one standard error above and below each mean.*

8 Experiment 5: Learners without sensations

In all of the experiments described so far, the sensations of each agent consist of one previous move. If this information about the previous move is removed, the agents are reduced to learning the value of each action without regard to context. In this case, the agents have only two Q-values to learn—one for each action. The discount factor γ can be set to zero because there is no longer any discrimination among "states." The goal of this experiment was to determine how these context-insensitive agents would perform against TFT and against each other. In both cases, they learned to always defect. This is not surprising because the defect action dominates the cooperate action in the single shot PD game.

However, with one small modification, the agents learned to cooperate with TFT. (They still always learned to defect against each other). Instead of estimating the average immediate payoff received for an action, the modified agents estimate the average of the next *two* payoffs received for an action. When only the immediate payoff is used, the actions that cause TFT's responses do not receive the proper credit, but when the 2-step returns are used, the actions are more closely tied to their consequences. The average 2-step returns for the cooperate and defect actions against TFT can easily be determined as a function of an agent's probability p of cooperating. There are four cases for both cooperate and defect (keep in mind that TFT must echo the agent's moves):

Cooperating vs. TFT			
step 1	step 2	2-step return	probability
CC	CC	$0.3 + 0.3 = 0.6$	p^2
CC	DC	$0.3 + 0.5 = 0.8$	$p(1-p)$
CD	CC	$0.0 + 0.3 = 0.3$	$p(1-p)$
CD	DC	$0.0 + 0.5 = 0.5$	$(1-p)^2$

Defecting vs. TFT			
step 1	step 2	2-step return	probability
DC	CD	$0.5 + 0.0 = 0.5$	p^2
DC	DD	$0.5 + 0.1 = 0.6$	$p(1-p)$
DD	CD	$0.1 + 0.0 = 0.1$	$p(1-p)$
DD	DD	$0.1 + 0.1 = 0.2$	$(1-p)^2$

From the tables above, it is easy to derive that the expected 2-step return is $0.1p + 0.5$ for cooperate and $0.3p + 0.2$ for defect. By plotting these linear equations over the interval [0,1], it is clear that in terms of the average 2-step return, cooperation dominates defection for all action probabilities. For example, assuming that the learner's current probability of cooperating is 0.5, the average 2-step returns are C: 0.55 and D: 0.35, so cooperation will seem to be the better action.

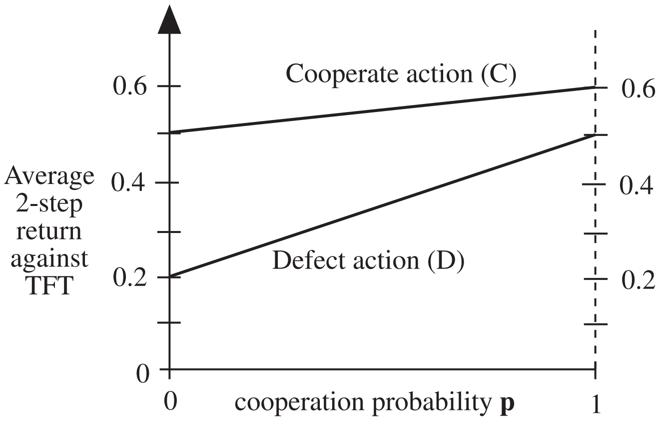

Figure 6: *Average 2-step return for cooperate and defect actions against TFT as a function of the cooperation probability of the agent. Cooperation dominates defection for all values of p.*

Although the 2-step return strategy works well against TFT, it does not work well against another learner. Because the other learner is using a context-free policy, there is no "retaliation" of the type TFT is able to provide. With the punishment for defecting removed, cooperation does not develop.

The results of this experiment illustrate that when a learner is faced with a problem with incomplete state information, a window of future rewards may be useful, as is often the case with a window of past sensations.

9 Experiment 6: Learners with different sensations

The final set of experiments investigated what happens when learning agents with different sensations play against each other in the IPD game. All of the agents tested used lookup tables to store Q-values. They used Boltzmann exploration with annealing factors 0.999 or 0.9999. The joint behavior of the agents was very sensitive to the annealing schedule. Slower annealing tended to produce significantly more cooperation and other semi-cooperative loops and a wider variety of final looping patterns, while the faster annealing schedule increased defection.

Agents with history windows of one, two, and three moves were tested. For example, a history of one means that the agent's sensation includes its own latest move and its opponent's latest move. The history lengths did affect the number and type of looping patterns that developed. With longer histories, a wider variety of patterns developed, and longer patterns developed. The longest looping pattern encountered was of length 8: CD / DC / CD / DC / CC / CC / CD / DD. It developed in the history 1 vs. history 3 game with slow annealing. In the asymmetric contests, the agent with the longer history tended to fare slightly better than the agent with the shorter history (Fig. 7), but not as much as had been expected. Overall, there was clearly more cooperation when both agents had a history of one move, Fig. 4.

History 1 vs. History 2					
CC	CD	DC	DD	% trials (anneal .999)	% trials (anneal .9999)
0.0	0.0	0.0	100.0	85	
0.0	50.0	50.0	0.0	7	41
25.0	25.0	0.0	50.0	4	2
20.0	20.0	40.0	20.0	2	1
0.0	50.0	25.0	25.0	1	26
33.3	0.0	33.3	33.3	1	
100.0	0.0	0.0	0.0		10
25.0	25.0	50.0	0.0		6
50.0	50.0	0.0	0.0		5
50.0	25.0	25.0	0.0		1
25.0	50.0	25.0	0.0		1
50.0	0.0	0.0	50.0		1
33.3	66.7	0.0	0.0		1
66.7	33.3	0.0	0.0		1
33.3	33.3	0.0	33.3		1
33.3	33.3	33.3	0.0		1
20.0	40.0	20.0	20.0		1
16.7	33.3	33.3	16.7		1

History 1 vs. History 3					
CC	CD	DC	DD	% trials (anneal .999)	% trials (anneal .9999)
0.0	0.0	0.0	100.0	93	3
25.0	25.0	0.0	50.0	1	9
0.0	0.0	25.0	75.0	1	
33.3	33.3	0.0	33.3	1	1
25.0	50.0	25.0	0.0	1	1
0.0	0.0	33.3	66.7	1	
0.0	50.0	25.0	25.0	1	44
0.0	50.0	50.0	0.0	1	23
25.0	25.0	50.0	0.0		4
100.0	0.0	0.0	0.0		3
20.0	40.0	20.0	20.0		2
33.3	33.3	16.7	16.7		1
33.3	33.3	33.3	0.0		2
50.0	0.0	0.0	50.0		1
25.0	37.5	25.0	12.5		1
42.9	28.6	14.3	14.3		1
20.0	40.0	40.0	0.0		1
20.0	60.0	20.0	0.0		1
75.0	25.0	0.0	0.0		1
0.0	40.0	20.0	40.0		1

History 2 vs. History 2					
CC	CD	DC	DD	% trials (anneal .999)	% trials (anneal .9999)
0.0	0.0	0.0	100.0	99	
0.0	33.3	66.7	0.0		29
0.0	66.7	33.3	0.0		18
33.3	0.0	33.3	33.3		11
33.3	33.3	0.0	33.3	1	7
0.0	33.3	33.3	33.3		6
0.0	50.0	50.0	0.0		6
25.0	25.0	50.0	0.0		6
25.0	50.0	25.0	0.0		4
100.0	0.0	0.0	0.0		3
16.7	16.7	50.0	16.7		3
33.3	33.3	33.3	0.0		3
20.0	40.0	20.0	20.0		2
50.0	0.0	25.0	25.0		1
16.7	50.0	16.7	16.7		1

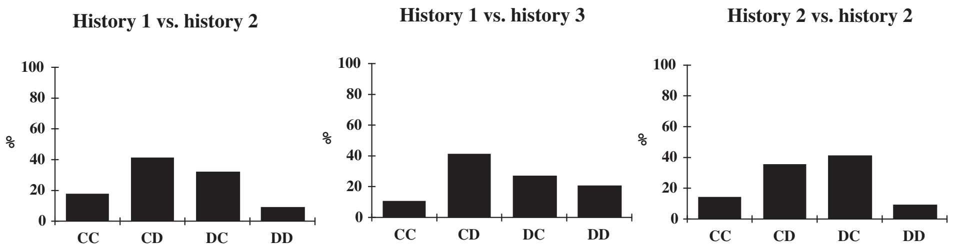

Figure 7: *Percentages of different plays at exploration annealing rate 0.9999,* $(\alpha = 0.2, \gamma = 0.95)$.

10 Conclusions and future research

Computational learning capabilities are an important component of intelligent agents, particularly when their designers cannot anticipate all situations that the agents might encounter and thus cannot preprogram the agents to operate as desired. Even when all situations are known to the designer, the optimal action that the agent should take in any particular situation may be unknown. Reinforcement learning (RL) addresses this problem by promoting actions that lead to high rewards and by demoting others. RL has been studied primarily within a single agent setting. In multiagent domains, RL research has focused on games where the agents' payoffs are either totally positively correlated (team problems) or totally negatively correlated (zero-sum games). This paper addressed this gap by studying an RL algorithm—Q-learning—in the IPD game, where the payoffs are neither totally positively nor totally negatively correlated. While selfish behavior is necessary in zero-sum games of pure competition and in team situations where the goals of each agent are identical to the goals of the team, it is often self-defeating in situations between those two extremes, which makes learning more difficult.

In iterated games, the true state (defined by the entire history of stage game plays) of the environment increases in dimension on every iteration. Thus, an agent can visit a state at most once. This is one reason why the convergence proof of Q-learning does not apply. Two different approaches were taken to handle this problem. In the first, the agent's Q-values were stored in a lookup table that used a fixed length history window as its sensation and ignored the older history. In the second, a recurrent neural net was used for the Q-value storage. In theory, such a net can store information from arbitrarily deep in the history, learn which features are important to keep track of, and generalize from training examples to previously unobserved states.

Both types of agents learned to play optimally against a fixed policy (TFT) for all settings of the discount factor. Playing against TFT is analogous to learning in a single agent environment, because TFT does not learn: its mapping from sensations to actions is fixed. Moreover, TFT uses only one previous move as its sensation, so learning against TFT is equivalent to learning in an environment where the true state does not increase in dimension and can be revisited.

Having multiple agents learning simultaneously makes the learning process considerably more difficult. In a 2-player iterated game, an agent's environment may be nonstationary because its opponent is learning (changing its Q-values) or changing its exploration method (mapping from Q-values to action probabilities). The agent's environment may be stochastic because the opponent may use a different decision context than the agent, and the opponent may choose actions stochastically. According to theory, an agent should learn the optimal Q-values even in stochastic domains, but in nonstationary settings the convergence proof does not apply.

Clear cooperation seldom emerged in experiments with two learners even though the discount factor was set high to stimulate cooperation. Recurrent net learners played non-cooperatively, but the final payoffs were always equal between the players. Lookup table learners played more cooperatively, but asymmetric loops occurred where the payoffs favored one player over another. When a recurrent neural net learner played a lookup table learner, the outcomes were non-cooperative and the asymmetric loops were always to the lookup table learner's advantage. Surprisingly, increasing the learning rate α from 0.2 to 1 enhanced cooperation between recurrent net players. The outcomes of games with non-exploring agents were sensitive to the initialization of their Q-values. An agent stood a higher risk of being exploited if it did not explore. Non-exploring agents also exhibited more looping than agents that explored.

A variety of Boltzmann exploration schedules were tested between lookup table learners. Slowing the annealing of exploration monotonically increased the sum of the agents' payoffs to a level slightly lower than that of total cooperation. DD plays disappeared first, but CC plays also gave way to loops of length two such as CC-CD and CD-DC. With very slow annealing processes, loops of type CC-CD prevailed.

Lookup table learners sensing different lengths of history were also tested. When the learners sensed longer histories, a wider range of interaction patterns occurred, and longer loops developed. When the learners sensed different history lengths, the agent with the longer history window received slightly higher payoffs on average. With both asymmetric and long symmetric memories, the outcomes were less cooperative than when both agents sensed a history of length one.

Future work should examine more closely the effect of exploration strategies on the types of patterns that develop. For example, what would happen between agents using different annealing schedules or strategies other than Boltzmann exploration? In addition, it would be interesting to train agents not just against a single opponent, but against a variety of opponents, as would be the case in tournament situations. This might enable the agents to learn more robust strategies.

In the long run it may be desirable to integrate sound learning methods into more complex agent architectures. Learning could help an agent adapt to the society of other agents and to the tasks at hand. There are numerous potential applications of multi-agent learning. For example, agents could learn pricing, timing and commitment strategies for competitive negotiations (Sandholm, 1993; Sandholm and Lesser, 1995b), deliberation control strategies to reduce computation overhead and to choose the best coalitions for computationally bounded agents (Sandholm and Lesser, 1994; 1995a), variable and value ordering heuristics for cooperative distributed constraint satisfaction, and communication strategies, to name just a few.

Acknowledgment

We thank Andy Barto, Victor Lesser, David Fogel, and two anonymous reviewers for helpful comments.

References

[1] Ashlock, D., Smucker, M. D., Stanley, E. A. and Tesfatsion, L. 1995. Preferential Partner Selection in an Evolutionary Study of Prisoner's Dilemma. This Issue.

[2] Axelrod, R. 1984. *The Evolution of Cooperation.* Basic Books, New York, NY.

[3] Barto, A. G., Sutton, R., and Anderson, C. W. 1983. Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:834–846.

[4] Barto, A. 1989. From chemotaxis to cooperativity: Abstracted exercises in neuronal learning strategies. In Durbin, R., Miall C. and Mitchison, G., eds., *The Computing Neuron*, 73–98. Addison-Wesley.

[5] Barto, A. G., Bradtke, S. J. and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138.

[6] Barto, A. G. and Anandan, P. 1985. Pattern recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:360–375.

[7] Barto, A. G. 1985. Learning by statistical cooperation of self-interested neuronlike adaptive elements. *Human Neurobiology*, 4:229–256.

[8] Bertsekas, D. P. and Tsitsiklis, J. N. 1989. *Parallel and Distributed Computation: Numerical Methods.* Prentice-Hall, Englewood Cliffs, NJ.

[9] Bradtke, S. J. 1993. Distributed Adaptive Optimal Control of Flexible Structures. Computer Science Department, University of Massachusetts at Amherst. Unpublished draft.

[10] Crites, R. 1994. Multi-Agent Reinforcement Learning. PhD dissertation proposal. Computer Science Department, University of Massachusetts at Amherst.

[11] Elman, J. 1990. Finding structure in time. *Cognitive Science*, 14:179-211.

[12] Fudenberg, D. and Tirole, J. 1991. *Game Theory.* MIT Press, Cambridge, MA.

[13] Haykin, S. 1994. *Neural Networks: A Comprehensive Foundation.* Macmillan, New York.

[14] Hecht-Nielsen, R. 1991. *Neurocomputing.* Addison-Wesley, Reading, MA.

[15] Kinney, M. and Tsatsoulis, C. 1993. Learning Communication Strategies in Distributed Agent Environments. Working paper WP-93-4. Intelligent Design Laboratory, University of Kansas.

[16] Kreps, D. 1990. *A Course in Microeconomic Theory.* Princeton University Press, Princeton, NJ.

[17] Lin, L-J. 1993. Reinforcement Learning for Robots Using Neural Networks. Ph.D. dissertation, School of Computer Science, Carnegie Mellon University.

[18] Littman, M. 1993. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning, Proceedings of the Eleventh International Conference*, pp. 157-163, Rutgers University, NJ.

[19] Littman, M. and Boyan, J. 1993. A Distributed Reinforcement Learning Scheme for Network Routing. Technical Report CMU-CS-93-165, Carnegie Mellon University.

[20] Luce, D. and Raiffa, H. 1957. *Games and Decisions.* Reprint: Dover Publications, New York, 1989.

[21] Markey, K. L. 1993. Efficient Learning of Multiple Degree-of-Freedom Control Problems with Quasi-independent Q-agents. In *Proceedings of the 1993 Connectionist Models Summer School.* Erlbaum Associates, Hillsdale, NJ.

[22] Narendra, K. S. and Thathachar, M. A. L. 1989. *Learning Automata: An Introduction.* Prentice-Hall, Englewood Cliffs, NJ.

[23] Nowak, M. and Sigmund, K. 1993. A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner's Dilemma game. *Nature*, 364: 56–58.

[24] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. 1986. Learning internal representations by error propagation. In Rumelhart, D. E. and McClellan, J. L., Eds., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 1, 318–362, MIT Press, Cambridge, MA.

[25] Samuel, A. L. 1959. Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, 210–229. Reprinted in Feigenbaum, E. A. and Feldman, J., Eds., *Computers and Thought*, McGraw-Hill, New York, 1963.

[26] Sandholm, T. 1993. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pp. 256-262, Washington D.C.

[27] Sandholm T. and Lesser, V. 1994. Utility-Based Termination of Anytime Algorithms, In *Proceedings of the European Conference on Artificial Intelligence (ECAI-94) Workshop on Decision Theory for Distributed Artificial Intelligence Applications*, pp. 88-99, Amsterdam, The Netherlands. Extended version: University of Massachusetts at Amherst, Computer Science Technical Report 94-54.

[28] Sandholm T. and Lesser, V. 1995a. Coalition Formation among Bounded Rational Agents. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada.

[29] Sandholm T. and Lesser, V. 1995b. Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework. In *Proceedings of the First International Conference on Multiagent Systems (ICMAS-95)*, San Francisco, CA.

[30] Sandholm, T. and Nagendraprasad, M. 1993. Learning Pursuit Strategies. Class project for CmpSci 689 Machine Learning. Computer Science Department, University of Massachusetts at Amherst, Spring 1993.

[31] Sen, S., Sekaran, M. and Hale, J. 1994. Learning to coordinate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pp. 426-431, Seattle, Washington.

[32] Shoham, Y. and Tennenholtz, M. 1993. Co-Learning and the Evolution of Coordinated Multi-Agent Activity.

[33] Sugawara, T. and Lesser, V. 1993. On-Line Learning of Coordination Plans. Computer Science Technical Report 93-27, University of Massachusetts, Amherst.

[34] Sutton, R. S. 1988. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3:9-44.

[35] Tan, M. 1993. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Machine Learning, Proceedings of the Tenth International Conference*, pp. 330-337, University of Massachusetts, Amherst.

[36] Tesauro, G. J. 1992. Practical issues in temporal difference learning. *Machine Learning*, 8:257–277.

[37] Tsetlin, M. L. 1973. *Automaton Theory and Modeling of Biological Systems.* Academic Press, New York, NY.

[38] Watkins, C. 1989. Learning from delayed rewards. PhD Thesis, University of Cambridge, England.

[39] Weiß, G. 1993. Learning to Coordinate Actions in Multi-Agent Systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pp. 311-316, Chambéry, France.

[40] Williams, R. J. and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280.