Reinforcement Learning in Cooperative Multi–Agent Systems

Hao Ren
haoren@cs.ubc.ca

Abstract

Reinforcement Learning is used in cooperative multi–agent systems differently for various problems. We provide a review on learning algorithms used for repeated common–payoff games, and stochastic general–sum games. Then these learning algorithms is compared with another algorithm for the credit assignment problem that attempts to correctly assign agents the awards that they deserve.

1 Introduction

Cooperative multi-agent systems consists of a number of agents attempting to maximize the joint utility through their interactions [1], and due to these interactions, the complexity can increase drastically with the number of agents and their action space.

1.1 Motive

The main goal of this paper is to provide a review on cooperative multi–agent learning, and provide a relationship between two areas of research within this domain.

Firstly, under the *game theoretic setting*, there is work by Boutilier et al. [2] that examined how to learn to cooperate in repeated common–payoff games. Also, there is Friend–or–Foe Q–learning [3] for general–sum games where the algorithm either finds a coordination equilibrium or an adversarial one depending on if the other agents are cooperative or competitive.

Secondly, there is the *credit assignment problem*, where an algorithm such as QUICR–learning [4] attempts to assign credit to agent i's contribution to the overall performance of the system. Algorithms such as QUICR–learning is more practical, but yet does not provide the strong theoretical properties such as the convergence to Nash equilibrium that comes with the games analyzed in Boutiler et al. [2] or Friend–or–Foe Q–learning. In Section 5, we will model the domain used in QUICR–learning as a general–sum stochastic game.

We will attempt to describe the relationship between these different cooperative multi–agent learning problems which are all based on reinforcement learning. This is an important issue because both of the above research areas contribute to cooperative multi–agent learning, and yet they approach the problem through entirely different models of analysis.

1.2 Domain

There are many types of multi–agent learning. Agents can be communicating or non–communicating [5], and we will consider non-communicating agents. Due to the system dynamics and how the reward function is calculated, communication between the agents does not need to be imposed [6].

Furthermore, as pointed out by Panait et al. [1], for cooperative multi–agent learning, there can be *team learning* or *concurrent learning*. For team learning, only one learner is used for the entire system, and it works out the behavior for all agents in the system. Team learning lacks game–theoretic aspect of multiple learners, but it can still pose an interesting challenge because of the unexpected outcome of the joint behaviors and interactions between the agents.

Team learning is not the focus of this paper though, instead, we will examine concurrent learning. With concurrent learning, each agent learns on its own to modify its behavior.

2 Concurrent Learning

To illustrate one of the main issue of concurrent multi–agent learning, the Stackelberg stage game as done in [7] is shown in Table 1.

	Left	*Right*
Up	1,0	3,2
Down	2,1	4,0

Table 1: Stackelberg Stage Game

Clearly, *Down* is the strict dominant strategy for the row player. In a repeated version of the stage game, if the row player actually plays *Down*, the column player will notice that and respond with $Left$, then there is repeated $(Down, Left)$ play. But suppose the row player instead repeatedly plays *Up*, then the column should notice that and play *Right*, so the two should end up with repeated $(Up, Right)$ play.

This example illustrates that an agent should adopt and learn while other agents teach her, and this would end up benefiting both of the agents.

The problem of interaction gets more complex with a larger number of agents. Agents modify their behavior, which in turn nullifies the assumptions used by other agents and their newly learnt behaviors [1]. This may make the system not to converge for practical problems.

Panait et al. [1] pointed out three areas of research under concurrent learning. Firstly, there is the credit assignment problem of assigning how to distribute the team reward to individual agents. Secondly, *dynamics of learning* analyzes the co-adaptation of different learning agents. Lastly, there is some work done for teammate modeling that studies how to improve interactions between agents. This paper focuses on the first two topics because we want to analyze the relationship between game–theoretic approaches to cooperative learning and the credit assignment problem.

2.1 Credit Assignment

As a group, agents are rewarded based on their joint actions. The most intuitive approach to award individual agents is to equally divide the reward to all agents and it is know as *global reward* [1]. However, some agents may do more work then other agents, and therefore should be awarded more. Global reward does not provide information for individual agents about their specific actions. One extreme solution to solve this problem, called *local reward* [1] [4], is to reward an agent based solely on its own behavior. Yet, this method may encourage selfish behavior, and agents may not have incentives to help each other. In Section 5, QUICR–learning by Agogino et al. [4] will be introduced, and it creates rewards for individual agents that not only have high "alignment" with the overall system learning but it is also very "sensitive" to the actions of the individual agents. Various other older approaches have been proposed, and interested readers can refer to Panait et al. [1] for a review on this field of research.

2.2 The Dynamics of Learning

Agents interact and change each other's learning environments. Most game theoretic methods is concerned with Nash equilibrium since no agent has rational incentive to change its behavior even if such an equilibrium is suboptimal globally. In a "fully-cooperative" [1] setting, the system is modeled as a common–payoff game as done by Boutilier et al. [2], so by increasing the system reward, the reward for all the agents are increased as well. Learning in the general setting, e.g. Friend–or–Foe Q–learning [3] in general–payoff game, corresponds to credit assignment problem as discussed in Section 2.1. This is an important observation, because we may now attempt to combine work for credit assignment problems with work done in the general–payoff games.

3 Preliminary Concepts and Notation

In this section, we will introduce some definitions for Reinforcement Learning (RL) and Stochastic Games.

Reinforcement Learning does not need a model of the environment and can be done online [7]. Therefore, RL is an ideal learning algorithm when an agent has little knowledge about other agents [8].

3.1 Single Agent Reinforcement Learning

Most work for RL has roots from the Bellman equations [9]. The general learning problems that we are considering involve environments that can be modeled as a stochastic process. Readers who are interested in further readings about single agent learning based on *Markov Decision Processes* (MDPs) can use [10], [11], and [12] as excellent introductory references on the topic. For a MDP, we have a set of states $s \in S$, a set actions $a \in A$, and state transition function $T(s, a, s')$ which is the probability of making a transition from state s to state s' using action a. By initializing the value function as $V_0 : S \rightarrow \mathbb{R}$, Bellman [9] shows that the value function can be iteratively updated by computing:

$$V_{t+1} = R(s) + \gamma \max \sum_{s' \in S} T(s, a, s') V_t(s')$$

A well studied method for updating V for single agent learning is Q–learning [13]. For MDPs with unknown reward and transition functions, Q–learning can be shown to converge, and calculate an optimal policy.

$$\begin{aligned} Q(s,a) &\leftarrow (1-\alpha_t)Q(s,a) + \alpha_t \left[R(s,a) + \gamma V(s')\right] \\ V(s) &\leftarrow \max_{a \in A} Q(s,a) \end{aligned}$$

The constrained values for α_t over time make up a learning rate schedule, and we assume that they are assigned values that satisfy conditions for convergence.

Some modifications to the above Q–value function can extend Q–learning to multi–agent settings.

3.2 Stochastic (Markov) Games

Stochastic games (aka Markov Games) are generalizations of repeated games and MDPs[1]. A stochastic game can be represented as a tuple $< N, S, \vec{A}, \vec{R}, T >$ [7]. N denotes a set of agents. S denotes a set of stage games. There are actions $\vec{A} = A_1, \ldots, A_n$ where A_i is the set of actions (or pure strategies) for agent i. $\vec{R} = R_1, \ldots, R_n$, with $R_i : S \times \vec{A} \rightarrow \mathbb{R}$, this gives the reward for each agent i for every stage game S. $T : S \times \vec{A} \rightarrow \Pi(S)$ is a stochastic transition function similar as the transition function defined for single agent RL, it specifies the probability of the next stage game given the previous stage game and the actions played by all the agents during it.

A stochastic game with only one stage game is a *repeated game*, and a stochastic game with only one agent is an MDP.

Just to provide one more definition: when all learning agents adopt the same strategy (e.g. all adopt reinforcement learning), this is called *self play* [7].

[1] MDPs is the setting for most of the work in AI.

4 The Dynamics of Learning

In this section, we introduce some typical work done for multi–agent learning from the game theoretic perspective.

4.1 Common–Payoff Repeated Games

Claus and Boutilier [2] used RL specifically for n–player common–payoff repeated games with arbitrary number of actions. For a stage game, if at each outcome all agents receive the same payoff, then the game is common-payoff . They demonstrated (without proof) that if agents (Independent Learners or ILs) in a game ignores the existence of the others will reach the equilibrium just as if agents (Joint Action Learners or JALs) learn the value of their respective actions by considering those of others. Convergence is based on the conditions of self–play and decreasing exploration. To demonstrate their results, we will use the game as shown in Table 2.

	a0	a1	a2
b0	11	-30	0
b1	-30	7	6
b2	0	0	5

Table 2: Repeated Common–payoff Game [2]

The two learners will start with non–equilibrium strategy profile $< b2, a2 >$. Then with further exploration, the row player will find $b1$ more rewarding, so they will play at non–equilibrium point $< b1, a2 >$. Column player will then notice the move, and perform $a1$. Once this equilibrium $< b1, a2 >$ is reached, the agents remain there. So convergence to global optimum $< b0, a0 >$ is not always achieved in these games even if each agent can immediately perceive the actions of all other agents in the environment. This result is disturbing, because in practical environments, agents do not have such complete information about the team and the environment [1].

4.2 Friend–or–Foe Q–learning for General–Sum Stochastic Games

Friend-or-Foe Q–learning (FFQ) [3] were shown to converge to a Nash equilibrium in a set of games that are a slight generalization on the set of zero-sum and common payoff games. When an opponent is considered to be friend, Q–value function is updated using ordinary Q–learning in the combined action space of the players. If an opponent is considered a foe, minimax–Q for zero–sum game is used. So the algorithm either finds a coordination equilibrium or an adversarial one depending on if the other agents are cooperative or competitive. But FFQ "cannot address the problem of finding equilibria in cases where neither coordination nor adversarial equilibria exist" [3]. Those are more general games with

some compromise between assuming the opponent is helping the agent to achieve maximum value and an opponent trying to minimize its value. So FFQ may still not be suitable for practical cooperative multi–agent learning problems that is addressed by the credit assignment problem.

FFQ is a typical algorithm for general–sum stochastic games, and as shown, the results are less satisfactory compared to common–payoff repeated games [7]. The main issue is that practical problems need to be modeled as general–sum stochastic games where agents may neither be purely coordinating nor purely adversary but a mixture [1].

5 Credit Assignment Problem

In this section, we continue the discussion of the credit assignment problem as described in Section 2.1, and examine one recent approach to the credit assignment problem[2]. Here, we will assume a large number of agents each with a large action space (e.g. 40 agents each with 5 actions).

5.1 Q Updates with Immediate Counterfactual Rewards–learning (QUICR)

QUICR–learning provides each agent i with reward that is both aligned with the system goals and sensitive to the agent's individual actions. It also has the benefit of not needing domain specific knowledge. Similar to the Q–value function as defined in section 3.1, a Q–value $Q(s_t, a_t)$ is calculated as:

$$Q(s_t, a_t) = r_t(s_t) + \max_a Q(s_{t+1}, a)$$

where we have action a_t and immediate reward $r_t(s_t)$ for state s_t.

The QUICR–learning rule for each agent i is specified as below:

$$\textsc{Quicr}(s_t, a_t) = r_t(s_t) - r_t(s_t - s_{t,i} + s_b) + \max_a \textsc{Quicr}(s_{t+1}, a)$$

where the expression $s_t - s_{t,i} + s_b$ denotes replacing agent i's state with an absorbing state s_b, s_t is the state of the system, and $s_{t,i}$ is the state for agent i.

The reward $r_t(s_t) - r_t(s_t - s_{t,i} + s_b)$ in the above equation is more sensitive to an agent's action than r_t because the effects of the other agents are subtracted out.

5.2 Domain Mapping to Stochastic General–Sum Game

The domain for QUICR–learning can be modeled as a stochastic general–sum game as described in Section 3.2. But as shown in the Section 4.2, the convergence results for stochastic general–sum games are restricted to

[2]Credit assignment problems may be either structural (assigning reward to agent i) or temporal (assigning global reward at time t). We focus on structural credit assignment problem, since the temporal credit assignment problem is solved by Q–learning

assumptions that are not applicable to this domain. Convergence guarantees are not especially practical for complex games as used here [2]. Without the convergence guarantees, Agogino et al. [4] uses experiments to show that the learning method do converge.

6 Conclusion

Due to unequal-share credit assignment, increasing the reward of an agent may not necessarily result in increasing the reward of all its teammates. In fact, such credit assignment may create highly non-cooperative scenarios. Therefore, "general sum games are applicable to the cooperative learning paradigm, even though in some situations such games may not be in any way cooperative" [1].

But existing game theoretic analysis (e.g. FFQ) on stochastic general–sum games have restrictions on the game that are unrealistic for practical purposes. Without theoretical analysis on the convergence properties of the learning algorithms, researchers [4] can only use experiments to show the convergence of their (non-game theoretic) algorithms.

The main contribution of this paper is to convert the domain used in Agogino et al. [4] to a model as a stochastic game, and see QUICR–learning's relationship with FFQ.

References

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

[2] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 746–752, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

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

[4] A. Agogino and K. Tumer. Quicr-learning for multi-agent coordination. In *Proceedings of the 21st National Conference on Artificial Intelligence*, Boston, MA, July 2006.

[5] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.

[6] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored MDPs. In *14th Neural Information Processing Systems (NIPS-14)*, pages 1523–1530, Vancouver, Canada, December 2001.

[7] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Journal of Artificial Intelligence*, 2006. to appear.

[8] Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In Jude W. Shavlik, editor, *ICML*, pages 242–250. Morgan Kaufmann, 1998.

[9] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[10] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[11] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[12] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*, chapter 16, pages 411–433. Morgan Kaufmann Publishers, draft edition, June 2003.

[13] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.