Evaluating concurrent reinforcement learners

Manisha Mundhe & Sandip Sen
Department of Mathematical & Computer Sciences,
University of Tulsa
e-mail:sandip@kolkata.mcs.utulsa.edu

Abstract

Assumptions underlying the convergence proofs of reinforcement learning (RL) algorithms like Q-learning are violated when multiple interacting agents adapt their strategies on-line as a result of learning. Empirical investigations in several domains, however, have produced encouraging results. We evaluate the convergence behavior of concurrent reinforcement learning agents using game matrices as studied by Claus and Boutilier [1]. Variants of simple RL algorithms are evaluated for convergence under increasing number of agents per group, scale up of game matrix size, delayed feedback and game matrix characteristics. Our results show surprising departures from that observed by Claus and Boutilier, particular for larger problem sizes.

1 Concurrent Reinforcement Learning

In this paper, we evaluate the effectiveness of standard reinforcement learning techniques for learning coordination strategies from feedback received due to repeated interaction with other agents in the environment. No formal guarantees exist for the convergence of concurrent reinforcement learners to mutually coherent policies. Researchers, however, have demonstrated partially successful applications of standard RL methods for a number of synthetic multiagent environments [2, 3].

In RL problems, the learner's environment can be modeled by a discrete time, finite state, Markov decision process represented by a 4-tuple $\langle S, A, P, r \rangle$ where $P : S \times S \times A \mapsto [0, 1]$ gives the probability of moving from state s_1 to s_2 on performing action a, and $r : S \times A \mapsto \Re$ is a scalar reward function. The Q-learning algorithm is designed to find a policy π^* that maximizes reward for all states $s \in S$. The decision policy is represented by a function, $Q : S \times A \mapsto \Re$. If an action a in state s produces a *reinforcement* of R and a transition to state s', then the corresponding Q value is modified as follows: $Q(s, a) \leftarrow (1-\beta)\ Q(s, a) + \beta\ (R + \gamma\ \max_{a' \in A} Q(s', a'))$.

We present four types of concurrent RL algorithm that have been used in our experimentation. The first two of these are based on Claus and Boutilier's experiments [1], while the third one is a modification of the second algorithm: **Individual reinforcement learners (Is):** These agents simply use the following update rule to update their Q-values: $Q(a) \rightarrow (1-\alpha) * Q(a) + \alpha * r$, where r is the reward obtained from the game matrix, and α is the learning rate. The exploration scheme used is a biased Boltzmann method, with action a being chosen with the probability

$$\frac{exp(\frac{Q(a)}{T})}{\sum_{i \in \mathcal{A}} exp(\frac{Q(i)}{T})},$$

where $\mathcal{A}$ is the set of actions available to agent A. Vidal and Durfee [4] calls this a 0-level agent which does not maintain any explicit model of other agents in its environment.

One-level Expected Utility based Probabilistic reinforcement learner (1EUPs): Q-values are calculated as in the case of Is. Expected value of action a is calculated as $E(a) = \sum_{b \in B} Q(a, b) Pr(b)$, where B is the set of actions available to the other agent, and $Pr(b)$ is the probability that the other agent will choose action b. The probabilities are calculated from the observed frequencies with which the other agent has chosen from its set of actions. The expected values for actions, and not the corresponding Q-values, are used to calculate the Boltzmann exploration probabilities. These agents model other agents as 0-level agents. Vidal and Durfee refer to such agents as 1-level agents.

One-level Policy based Probabilistic reinforcement learner (1PPs): Instead of using frequency based model as 1EUP agent, 1PP agent assume other agents to be of the same level as itself. Expected value of an action is calculated as 1EUP agents, except that the probability that other agent will choose an action is calculated based on its own policy matrix. This value is then used to calculate the Boltzmann probabilities. We call these agents as 1p-level agents.

Two-level Simulated Expected Utility Maximizing reinforcement learner (2SEUMs) or Two-level Phantom agents: A 2SEUM models the other agent as a 1EUP. To calculate the action probability distribution of this other

	b0	b1
a0	x	y
a1	m	n

Table 1. A 2x2 game matrix.

agent, the 2SEUM assumes that the other agent has the same policy matrix as itself. Using its own past history of actions, the 2SEUM first calculates the probability of other agent's taking a particular action. Using this probability distribution it simulates a game to predict other agents action. Based on this, it then greedily chooses an action with maximum expected utility. These agents model other agents as 1-level agents. Vidal and Durfee refer to such agents as 2-level agents.

For the rest of the paper, we will refer to the Is, 1EUMs, and 2SEUMs as 0, 1, and 2-level learners respectively.

2 Experimental results

We use repeated two-agent single stage games with matrices of the form shown in Table 1 to evaluate the learners. The game matrix represents the feedback given to both agents when they select any particular action combination. If two strategies converge to the same average payoffs, we prefer the strategy which allows agents to reach the corresponding policies with smaller number of interactions. Experimental parameters were as follows: $\alpha = 0.1$, $T_0 = 16$, and a temperature discounting factor of 0.99.

We varied the number of actions available to each agent from 2 to 5 to 10 to 20. Payoff matrices used had only two possible values: all diagonal elements had the same optimal value of 1, and the off-diagonal elements had a value of 0.75. In general, for the smaller problem sizes (not shown here), performance of the different agent types are in the following order: 2-2, 1-1, 0-0 . But as we increase the action space, i.e., allowing more actions per agent, the 1-1 scenarios converge at a suboptimal value, even though, the rate of convergence is still better than the 0-0 agents (see Figure 1). The 0-0 and 2-2 scenarios still converge optimally. A plausible explanation of the quick convergence in the case of 1-1 scenarios is that there is insufficient exploration of the state space, resulting in premature convergence.

We have conducted a number of other experiments including features like domain noise, delayed feedback, varying matrix properties, etc. Though the general scale up properties follow the pattern displayed here, much work remains to be done to analyze why simple 0-level learners perform so well for large problems. We believe that our experiments only scratch the surface of a complex, interesting landscape that needs closer scrutiny to identify the scope and limitations of concurrent reinforcement learners.

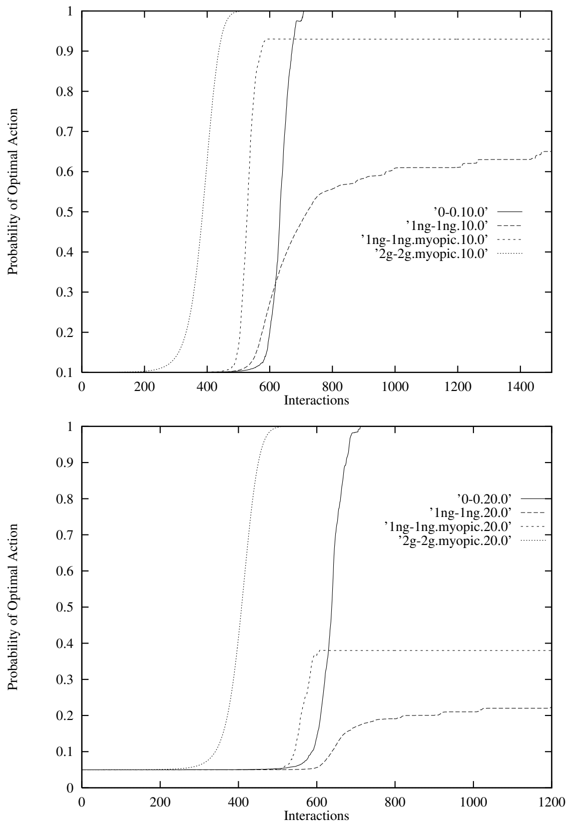

Figure 1. Relative performance of individual I (0-0), 1EUP (1ng-1ng), 1MEUP (1ng-1ng.myopic) and 2SEUM (2g-2g) learner pairs with scale up of problem size. The numbers between the periods refer to the number of actions available to an agent.

Acknowledgments: This work has been supported in part by an NSF CAREER award IIS-9702672.

References

[1] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the AAAI-98*, pages 746–752, Menlo Park, CA, 1998. AAAI Press/MIT Press.

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

[3] S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *Proceedings of AAAI-94*, pages 426–431, Menlo Park, CA, 1994. AAAI Press/MIT Press.

[4] J. M. Vidal and E. H. Durfee. Agents learning about agents: A framework and analysis. In *Woking notes of AAAI-97 workshop on Multiagent Learning*, pages 71–76. AAAI, 1997.