Reinforcement Learning for Stochastic Cooperative Multi-Agent-Systems

Martin Lauer
University of Osnabrück

Martin Riedmiller
49069 Osnabrück, Germany

Abstract

We present a distributed variant of Q-learning that allows to learn the optimal cost-to-go function in stochastic cooperative multi-agent domains without communication between the agents.

1. Introduction

Multi-agent reinforcement learning is faced with the problem of several independent agents acting in a shared environment with the objective to learn a strategy to act optimally with respect to a given reward function [1, 2, 6].

We present an algorithm for general stochastic environments for the model-free independet learner (IL) [3] case. No explicit communication between the agents is assumed. In its 'full-lists'- variant, convergence to the optimal joint policy can be shown. In a more efficient 'reduced-lists'-variant, theoretical convergence currently is only known for deterministic environments.

2. A learning algorithm for independent learners (IL)

The framework we consider here is a standard Markov Decision Process (MDP) where actions are vectors of agent individual decisions. The goal is to find an optimal policy π^* that maximizes the sum of discounted rewards.

Independent learners suffer from the lack of information about the actions taken by their teammates. They only know about their own contribution a_i to the joint action. A value estimation based on that information alone would therefore mix the rewards of several different joint action vectors and thus become meaningless. The problem is therefore, that the independent learners must somehow be able to distinguish between the different joint actions. The key idea of the following algorithm is, that the agent can distinguish between different joint actions without knowing, what exactly the joint actions are. This will be realized by the idea of 'implicit coordination'. We will develop our algorithm in four basic steps:

1. Idea of implicit coordination Each agent i maintains a list for every state s, $L_i(s)$, where each entry corresponds to exactly one joint action. A list entry contains a tuple of the form $L_i(s)[l] = \langle Q, a_i, n\rangle$ where a_i is component i of the (implicitly) referenced joint action vector, Q the respective Q-value and n counts the number of occurrences of the pair $(s, \underline{a})$, where $\underline{a} = (a_1, a_2, \ldots a_m)$ is the joint action of the m agents. If we can assure, that at every single point in time, all agents select the same index l to access their individual lists and select the action $L_i(s)[l].a_i$, then the reward can be uniquely assigned. The reward is caused by the joint action implicitly referenced by the list index. Accordingly, the Q-value of that entry, $L_i(s)[l].Q$, can be correctly updated.

2. The Select_Index procedure The Select_Index procedure is realized in every agent and has to assure, that at every point in time all agents get the same index. As a necessary condition this therefore assumes, that all agents are aware of the same time, e.g. by reliably counting the number of decision cycles.

3. Efficient exploration To enforce efficient exploration, the lists are sorted according to decreasing Q-values. This means, that the most promising actions are at the beginning of the list.

4. Reduced lists When the lists contain an entry for every possible joint action, then the approach is called the 'full lists'-variant of the algorithm. However, dealing with full lists will be rather impractical. A more feasible way is to only consider lists with a reduced number of joint actions ('reduced lists'-variant). The size of a list is denoted by l_{max}. Certainly, ignoring some joint actions yields the danger of missing the optimal solution. Therefore, after some training cycles (the 'training interval'), the list is partially reset. Only the l_{keep} best entries are preserved; the rest of the list

is reinitialized and new candidates get a chance. This means, every list entry from $l_{keep}+1$ to l_{max} is assigned a new individual action; the Q and n entries are reset to zero. To enforce a stabilization of the learning process, the length of the training intervals is increased in course of learning.

Convergence: The proof of the theoretical soundness of the 'full lists'-approach is based on exploiting its relationship to standard Q-learning (see full version of this paper).

2.1. Description of the algorithm

Here is a coarse description of an implementation: Every agent i keeps for every state s a list, where every entry in the list, $L^i(s)[l]$ stores three values: the individual action $L^i(s)[l].a \in Actions(i)$, the number $L^i(s)[l].n \in \mathbb{N}$ of times that the index l has been selected up to now, and the current Q-value $L^i(s)[l].Q \in \mathbb{R}$. Each list contains at most l_{max} entries.

A procedure **LearnTransition(state s)** is realized in every agent. It selects an individual action (corresponding to a selected index l_i) which is applied to the environment. The observed successor state is then used to update the Q-value of state s at index l_i. Learning of transitions is repeated until a certain number (num_traincycles) of cycles is done. Then, each agent resets parts of his list : First, the lists are sorted by decreasing Q-values, then all but l_{keep} entries are deleted. The freed list entries are filled again by new candidate actions. The whole process is repeated, until the optimal policy is found or approximated to a sufficient level.

3. Empirical Results

The example we consider is the 'climbing game' taken from [3]:

	a_0	a_1	a_2
b_0	11	-30	0
b_1	-30	7	0
b_2	0	6	5

Claus and Boutilier showed, that their learning scheme for ILs will find an equilibrium, which not necessarily is the optimal solution. In fact, for the climbing game they reported that the agents learned to play (a_1, b_1), which yields a reward of 7, whereas the maximum reward is 11 for joint action (a_0, b_0).

The full-list variant of our algorithm can be theoretically shown to find the optimal solution. For the more efficient reduced-lists variant, where theoretical convergence is not guaranteed at the moment, empirical results demonstrate the success of the algorithm in finding optimal solutions (in the experiments, the list had 2 entries). In extensions of the results of Claus and Boutillier, this can also be observed for different noise levels (see figure 1).

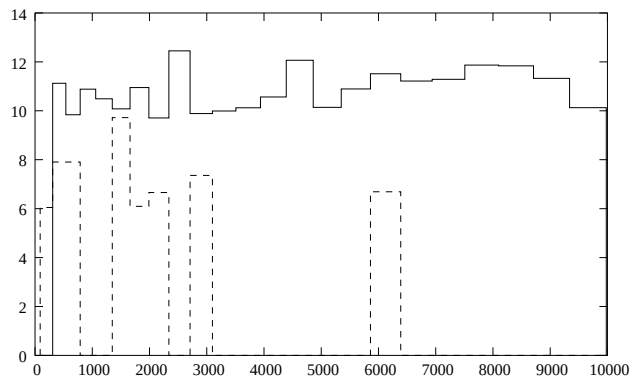

Figure 1. Reduced-lists algorithm for stochastic climbing game with noise variance 1. Learned Q-values of optimal (solid) and second best action (dashed) at the end of training cycle. The optimal action was found after 300 cycles and kept from then on in the (reduced) list.

4. Conclusion

The proposed approach overcomes the elementary problem in distributed learning of distinguishing between random noise and structural influence of other agents. It is based on an implicit agreement at the beginning of learning. A more detailed description and more empirical results can be found in the full version of this paper.

References

[1] J. Hu and M. P. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. *ICML*, 1998.

[2] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. *ICML* 1994.

[3] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *National. Conf. on AI*, 1998.

[4] C. Watkins. *Learning from delayed rewards.* PhD thesis. Cambridge, UK, 1989.

[5] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. *ICML*, 2000.

[6] J. Schneider, W. Wong A. Moore and M. Riedmiller, Distributed value functions *ICML* 1999.

[7] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. *ICML* 2002.