ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

A Multiagent Architecture for Concurrent Reinforcement Learning [*]

Víctor Uc Cetina

Humboldt-Universität zu Berlin - Institut für Informatik
Unter den Linden 6, 10099 Berlin - Germany

Abstract.
In this paper we propose a multiagent architecture for implementing concurrent reinforcement learning, an approach where several agents, sharing the same environment, perceptions and actions, work towards one only objective: learning a single value function. We present encouraging experimental results derived from the initial phase of our research on the combination of concurrent reinforcement learning and learning from demonstration.

1 Introduction

It is well known that reinforcement learning algorithms usually require a prohibitive amount of time to produce optimal and reliable behavior policies. Focusing on that problem, we present in this paper a multiagent architecture for implementing agents that learn autonomously through reinforcement learning and also from demonstrations generated by one or more available teachers. One key aspect of our approach is the use of multiple agents that concurrently refine a value function. We show that, when several agents are employed to solve a machine learning problem, the state space is explored in a parallel manner, allowing with this an acceleration in the learning process. Moreover, if we enhance this colaborative learning process with the help provided by teachers, the number of training episodes needed to produce a good value function can be reduced remarkably. We provide experimental results that show the viability of our architecture and also that it is sufficiently general to permit the use of any kind of reinforcement learning algorithm.

1.1 Reinforcement Learning

Reinforcement learning (RL) is a machine learning paradigm that addresses the problem of learning a behavior policy through the experience generated by the agent and its interaction with the environment [7]. In the work reported here, we used Q-learning [8], a RL algorithm that under some reasonable conditions is guaranteed to converge and build the optimal value function $Q^*(s, a)$, which assures us that the agent will always choose the best action from every state.

[*]This research work was supported by a PROMEP scholarship and UADY.

1.2 Related work

The idea of enhancing the performance of reinforcement learning algorithms using domain knowledge provided by a supervisor or teacher has been studied in recent years by several researchers. Those works that keep more relation to ours are the following. Lin [2], Rosenstein and Barto [5], Abbeel and Ng [1], Schaal [6], Mataric [4], and Maclin and Shavlik [3].

All of these approaches are similiar to ours in the fact that they all incorporate information generated from teacher's demonstrations to accelerate the construction of a value function, which so far seems to be the most effective way to obtain any significant improvement. Some of them study the way to extract a model from such demonstrations, others attempt to create what they call progress estimators to advice the agent about how close it is to a goal state. However, they do not contemplate in their investigation, how this speed up in learning could be affected by the employment of many agents. In our research, we focus on a multiagent approach to learning from demonstration, analyzing how the number of agents involved can accelerate the whole process.

2 Architecture

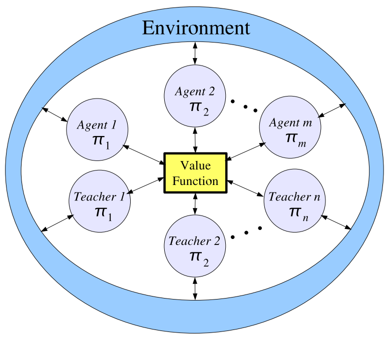

Fig. 1: A multiagent architecture for concurrent reinforcement learning.

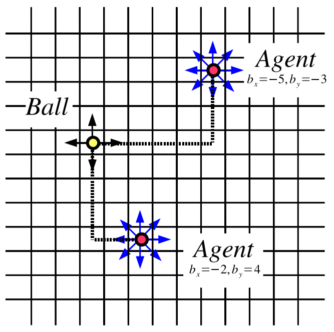

Fig. 2: Dynamic grid world.

The multiagent architecture for concurrent reinforcement learning (see Figure 1) has as main objective to be of practical use for developing reinforcement learning systems that require a minimum number of training episodes in order to safely master a specific target behavior. In other words, we want our systems to learn as fast and reliably as possible. To this end, our approach employs two strategies. First, we use one or more available teachers that provide demonstrations of how the task to be learned should be performed. These demonstrations are used to build an initial value function that can be used by the agents since the beginning of their learning process. We describe this process as constructing a *road map* of the state space that will guide the learning agent to goal states. Second, we use as many agents as possible to further improve the value function through a parallel exploration of the state space. They are aimed to refine the initial road map created by teachers.

The multiagent architecture that we propose in this paper, is intended for machine learning problems where many identical agents are available. By identical we mean that such agents have the same set of sensors and effectors. Consider for example a team of aibo robot dogs learning to perform a specific task in the Robocup context. Using all the robots at the same time to learn the task would certainly provide some benefits in the learning rate. Moreover, we might take advantage from one or more human experts. This is, if we could exploit the domain knowledge of human teachers, the learning process might be further accelerated.

Our reinforcement learning approach is centered in the construction of the *value function*, whether it is a *state value function* or a *state-action value function*. The construction of the value function is accomplished through implicit colaboration of the agents taking part in the learning process, with no communication skills required among agents. All they need to do is to update and consult the same value function everytime. Although the experiments presented here were performed using Q-learning, the use of this architecture is independent of the RL algorithm, so that other kinds of RL based on Markov Decision Process are also applicable.

2.1 Teachers

The purpose of using teachers is that we can obtain optimal and useful trajectories to the goal states. This is really valuable especially during early stages of the learning process when agents without supervision usually waste much time trying fruitless actions. The process for including example demonstrations consists of two phases and it works in the following way. We will consider one teacher to keep the explanation simple.

First, we place the teacher in a complex state s_t, a state where the path to the goal state would be very difficult to find out by an agent. Then we observe the behavior of the teacher through the state space until it reaches the goal state. We record the transitions of the teacher in the environment as a demonstration vector V_d of n tuples $(s_t, a_t, r_{t+1}, s_{t+1})$. Once we have the demonstration vector, we continue with the second phase of building the road map. The vector V_d is used to update the Q-function. The updating step should be performed in backwards order, going from the last element of V_d to the first one. We can also repeat the updating step several times. This method allows us to speed up the refinement of the Q-function [2].

In our experiments, we used only one teacher, and his demonstrations were applied before the agents started to learn by themselves. However, the architecture allows the interactive updating of the value function by agents and teachers, at any time of the learning process. This interaction and its effects in the whole process is subject of another ongoing research.

2.2 Agents

We consider that each agent Ag_i is a standard reinforcement learner, which follows a behavior policy π_i during the training phase and that it is affected by a reward funcion $R(s, a)$. Although each agent might follow a different policy, we decided to use the same for all of them in our experiments, given that in this paper we are only concerned with the improvement that can be produced by many identical agents. In contrast, the reward function must be the same for all the agents, since they will be interacting with the same environment. As we mentioned before, agents and teachers are needed to work colaboratively in order to reduce the amount of time required to refine the value function.

3 Testbed

In a grid of 25×25 cells, we place a moving ball and m agents. The problem addressed by the agents is to learn to intercept the trajectory followed by the ball, before it reaches any of the grid's edges. We consider that an agent has succeeded in intercepting the ball when it moves into the same cell the ball moves in. During each training episode the ball can only follow one direction out of four: north, east, south or west, as shown in Figure 2, and it can move only one cell at each step time. When it reaches a grid's edge, the current training episode is considered to be concluded. The agents in contrast, can change their direction at each time step, by choosing one of the following eight possible movements: north, northeast, east, southeast, south, southwest, west or northwest, see also Figure 2. In order to make their job more challenging, we allow them to move only one cell each time, just like the ball. By doing so, we let the agents move as fast as the ball can move. Therefore, they must perform the minimum number of incorrect movements in order to reach the ball before the current training episode is finished.

To define each state of the world, we made the assumption that at any time, the agent could perceive the position of the ball and its direction. Based on this information, the agent is able to determine its current state as given by (b_x, b_y, d), where b_x and b_y are the distances between the ball and the agent, in the x and y axis respectively, and d is the ball's direction (see Figure 2).

4 Experiments and Results

Initially, we worked with different numbers of agents and without a teacher. In this way, we focused our attention on the improvement obtained in the learning speed, when varying the number m of concurrent learning agents. First, we trained the m agents during 1000 episodes, where each episode consisted in placing the ball and m agents in a randomly selected cell on the grid. Also, the ball was randomly assigned one valid direction. Each episode was concluded when the ball reached the grid's edge. After the training phase, one agent, using the Q-function just generated, was asked to intercept the ball 500 times. Every time the agent handled to intercept the ball was counted as a success.

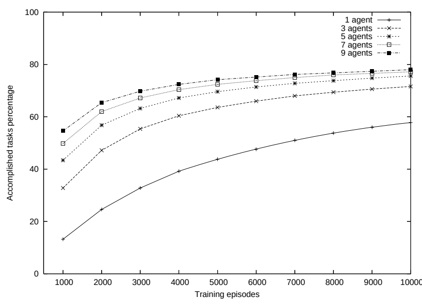

Fig. 3: Performance curves with different number of agents.

We repeated this experiment 1000 times and averaged the resulting number of successes. Next, using the average, we calculated the percentage of tasks successfully accomplished. The same steps were repeated for $2000, 3000, \ldots, 10000$ training episodes and the resulting percentages were plotted in Figure 3, where we can observe how much the percentage of successful tasks is improved when we increase the number of agents involved in the concurrent learning process.

Another experiment carried out was the comparison between learning with a teacher and without it, for 1 and 9 agents. For each number of them, we initially generated 40 example demonstrations of different situations of the world, those that we considered specially difficult to learn. Then, we updated the Q-function in backwards order. The results were plotted in Figures 4 and 5. We used $\alpha = 0.5$, $\gamma = 0.5$, and rewards of 10 in goal states and -1 otherwise.

In Figure 4 we can see that a huge acceleration in the learning process is obtained when we use a teacher and only one agent. However, as it is evident in the plot, once the teacher has provided his advice, the agent is incapable to keep improving the Q-function. The reason for this is that the state space is significantly big, and one only agent is not sufficient to further improve the road map initially created by the teacher, at least not with 10000 training episodes. In contrast, in Figure 5 a different story occurs. After the demonstration by the teacher, the team of 9 agents keeps improving the Q-function. We believe that further improvement can be obtained if different policies were followed by each agent, such that the state space could be explored more efficiently.

5 Conclusions and future work

We have implemented a multiagent system using an architecture for concurrent reinforcement learning where all the agents are used to update the same Q-function at the same time. One teacher provided example demonstrations, in order to build a road map of the state space. Such map was used then by the agents to continue learning, avoiding with this, to start from zero knowledge. In other words, they exploited the state map created by the teacher. The experimental results showed that using more agents to concurrently learn the Q-function can accelerate and improve the learning during early stages of the

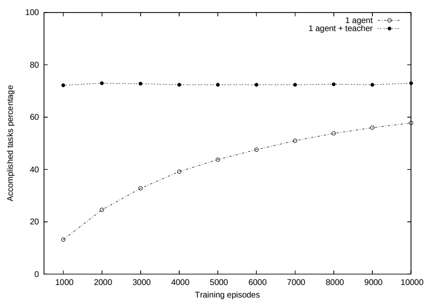

Fig. 4: Performance curves with 1 agent and 1 teacher.

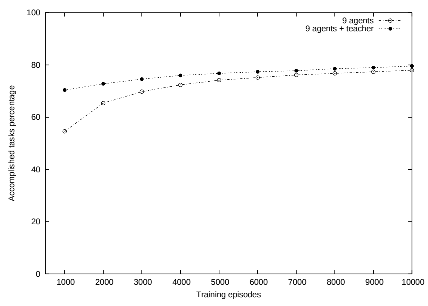

Fig. 5: Performance curves with 9 agents and 1 teacher.

training phase. Also, we showed that using a teacher that provides demonstrations of the task to be learned, is an effective way to reduce the number of training episodes needed to obtain a reliable agent's behavior. We are working on extending this framework to consider a general function approximator instead of a look-up table as the value function, in order to solve machine learning problems with continuous state and action spaces.

References

[1] P. Abbel and A. Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[2] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, (8):293–321, 1992.

[3] R. Maclin and J. W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, (22):251–282, 1996.

[4] M. J. Mataric. Reward functions for accelerated learning. In *Proceedings of the 11th International Conference on Machine Learning*, 1994.

[5] M. T. Rosenstein and A. G. Barto. Supervised actor-critic reinforcement learning. In *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*. John Wiley & Sons, 2004.

[6] S. Schaal. *Learning from Demonstration*, pages 1040–1046. Advances in Neural Information Processing Systems. MIT Press, 1997.

[7] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[8] C. Watkins. *Learning from Delayed Rewards.* PhD thesis, University of Cambridge, 1989.