Expertness Based Cooperative Q-Learning

Majid Nili Ahmadabadi and Masoud Asadpour

Abstract—**By using other agents' experiences and knowledge, a learning agent may learn faster, make fewer mistakes, and create some rules for unseen situations. These benefits would be gained if the learning agent can extract proper rules out of the other agents' knowledge for its own requirements. One possible way to do this is to have the learner assign some expertness values (intelligence level values) to the other agents and use their knowledge accordingly.**

In this paper, some criteria to measure the expertness of the reinforcement learning agents are introduced. Also, a new cooperative learning method, called weighted strategy sharing (WSS) is presented. In this method, each agent measures the expertness of its teammates and assigns a weight to their knowledge and learns from them accordingly. The presented methods are tested on two Hunter–Prey systems.

We consider that the agents are all learning from each other and compare them with those who cooperate only with the more expert ones. Also, the effect of the communication noise, as a source of uncertainty, on the cooperative learning method is studied. Moreover, the Q-table of one of the cooperative agents is changed randomly and its effects on the presented methods are examined.

Index Terms—**Cooperative learning, expertness, multi-agent systems, Q-learning.**

I. Introduction

IN HUMAN societies, it can be observed that, the more one learns from another's experiences, a higher chance he has to succeed. In fact, people take advice, consult with each other, receive unprocessed information, and observe others to learn from their activities and experiences. In other words, people cooperate to learn.

In almost all of the present artificial multi-agent teams, agents learn individually and cooperative learning has not been deeply investigated. However, similar to human beings, agents are not required to learn everything from their own experiences (see Fig. 1). In fact, due to having more knowledge and information acquisition resources, cooperation in learning in a multi-agent system may result in a higher efficiency compared to individual learning [17]. Improvements in learning have been shown in different researches even when simple cooperative learning methods are used [30].

As the learner agents are not capable of representing their knowledge properly and observing the other agents requires a high level of sensing and intelligence, the agents cannot advise each other or automatically learn by passively observing the other agents. Therefore, they are required to communicate their experiences and information.

In almost all of the multi-agent learning published papers, cooperation is unidirectional between a fixed trainer agent and a learner. However, all agents may learn something from each other provided that, some proper measures and methods are implemented.

One of the most important issues for a learner agent is the assessment of the behavior and the intelligence level of the other agents. In addition, the learner agent must assign a relative weight to the other agents' knowledge and use it accordingly.

In general, these three issues are very complex and need careful attention. Therefore, in this paper, as well as in [22], attention has been paid to find some solutions for homogeneous, independent, and cooperative Q-learning agents.

In [22], a new cooperative learning strategy, called weighted strategy sharing (WSS) and some expertness measuring methods are introduced. In that paper, it is assumed that the learner agents cooperate only with the more expert agents. Also, it is assumed that, the communication is perfect and all of the agents are reliable. In this paper, it is considered that all of the agents could learn from each other and the obtained results are compared with the results of the algorithm presented in [22]. In addition, effects of the communication noise as a source of uncertainty on the cooperative learning are studied. Moreover, the Q-table of one of the cooperative agents is changed randomly and its effects on the presented method are examined.

Related researches are reviewed in the next section. Then, WSS is briefly introduced and some expertness measures are presented. Also, some weight assigning methods are established. WSS, the effects of implementing the expertness measures, and the role of weight assigning methods are tested in the fourth section. In that section, effects of uncertainty and wrong knowledge are also studied. A conclusion and some directions for future research are given in the last section.

II. Related Researches

Samuel [26] used the competitive Llearning algorithm to train a checker game player. In his method, the cooperator agent acts as an enemy or an evaluator and tries to find the weak points of the learned strategy. Hu and Wellman [12] proposed a framework for multi-agent Q-learning when the competitor agents have incomplete information about other agents' payoff functions and state transition probabilities.

In the ant colony system [6], some ants learn to solve the traveling salesman problem by nonverbal communication through the pheromones on the edges of a graph.

Imitation [16] is one of the cooperative learning methods. In this method, the learners watch the actions of a teacher, learn

Manuscript received September 28, 2000; revised September 9, 2001. This paper was recommended by Associate Editor A. Bensaid and Editor L. O. Hall.

M. N. Ahmadabadi is with the Department of Electrical and Computer Engineering, University of Tehran, Tehran, Iran, and also with the Intelligent Systems Research Center, Institute for Studies on Theoretical Physics and Mathematics (IPM), Tehran, Iran (e-mail: mnili@ut.ac.ir).

M. Asadpour is with the Intelligent Systems Research Center, Institute for Studies on Theoretical Physics and Mathematics (IPM), Tehran, Iran (e-mail: asadpur@karun.ipm.ac.ir).

Publisher Item Identifier S 1083-4419(02)00123-1.

them, and repeat these actions in similar situations. This method does not affect the teacher performance [3] and the learning process is unidirectional. For example, in [16], a robot perceives a human doing a simple assembly task and learns to repeat it in different environments. Hayes and Demiris [10] built a robotic system in which a learner robot imitates a trainer moving in a maze and learns to escape from it.

Yamaguchi and others [33] developed a robotic imitation system to train Q-learning ball-pusher robots. In this system, agents learn individually and imitate each other using simple mimetism, conditional mimetism, and adaptive mimetism methods.

In simple mimetism [35], all agents imitate each other when they are neighbors. In this method, two neighbors may wait for each other forever. This problem is solved by applying conditional mimetism [35]. In conditional mimetism, only the low performance agent (performance is measured based on the sum of the rewards and punishments received in past n actions) imitates the other one. Adaptive mimetism [33], [34] is similar to conditional mimetism, but the imitation rate is adjusted based on the performance difference of two neighbor robots.

The robots cooperate to learn when they share their sensory data and play the role of scout for each other [30]. Episode sharing [14], [30] can be used to communicate the state, action, and reward triples between the reinforcement learners. Tan showed that, sharing episodes with an expert agent could improve the group learning significantly [30]. In [15], the state, action, and value pairs are communicated among the agents. No measure is used to evaluate the received rules by the learners.

In [4], a blackboard is used as a global information system for improving the individual learning and coordination in a multiagent team.

In the collective memory method, learners put learned strategy or experienced episodes on a shared memory [8] or they have a single memory and update it cooperatively [30].

A cooperative ensemble learning system [17] has been developed as a new method in neural network (NN) ensembles [1], [11], [17], [25], [27]. In these studies, a linear combination of the concurrent learning NN's outputs are used as a feedback to add a new penalty term to the error function of each network.

Provost and Hennessy [24] developed a cooperative distributed learning system for systems with huge training sets. The training set is divided into k smaller training subsets and k rule-learning agents learn the local rules. The rules are transmitted to the other agents for evaluation; if the rule satisfies the evaluation criteria, it is accepted as a global one.

High attention is paid to the advice taking method in recent years [9], [13], [19], [21], [23]. Mostow [21] wrote a program that accepts high-level advices to play the card game. Gordon and Subramanian [9] developed a system that translates high-level advices into the concrete actions and evaluates them by genetic algorithm (GA).

Maclin and Shavlik [18] used the advice taking scheme to help a connectionist reinforcement learner. The learner accepts advice in the form of a simple computer program, compiles it, represents the advice in some NNs and adds them to its current network.

In most of the reviewed researches, cooperation is unidirectional from a prespecified trainer to a preselected learner

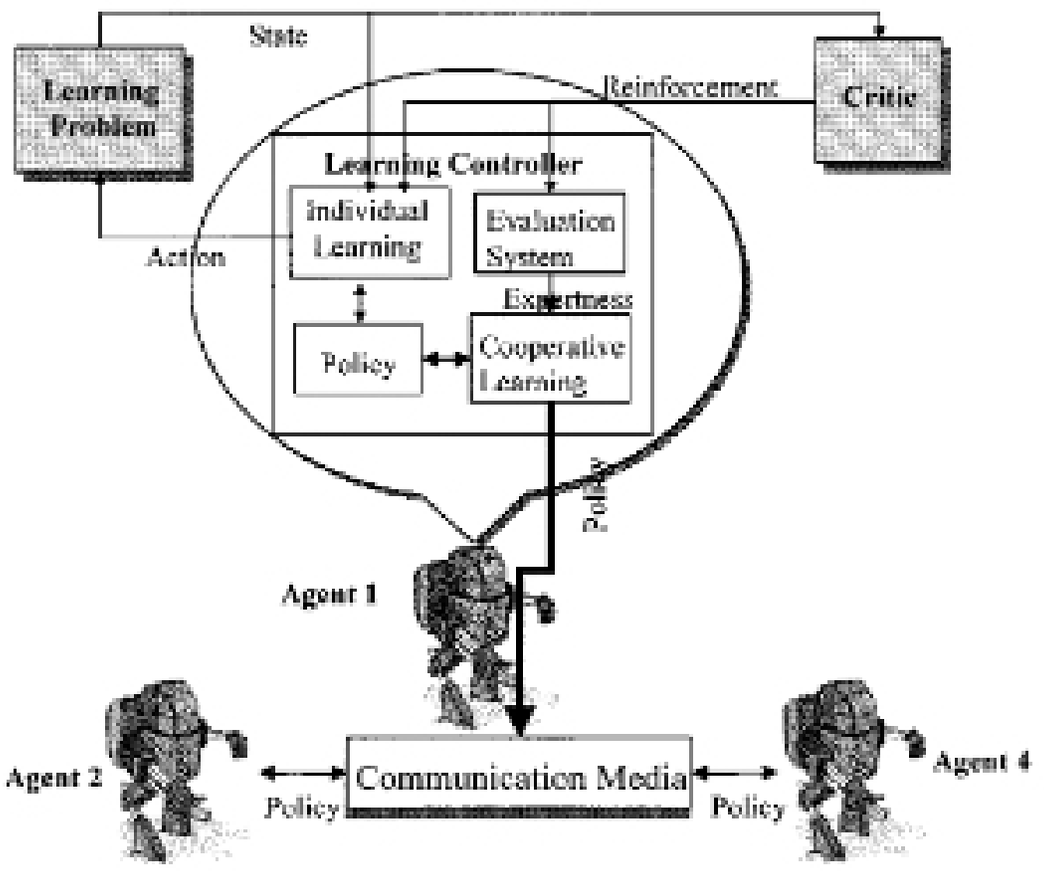

Fig. 1. Weighted strategy sharing (WSS) architecture.

agent. In the real world, cooperative learning is bidirectional and all of the agents learn something from each other (even from the nonexpert ones). In the strategy sharing method [30], each Q-learning agent learns from all of its teammates. The agents learn individually and at some special instants, each agent gathers the Q-tables of the other agents and takes the average of the tables as its own new strategy. In this system, the agents do not have the ability to find good teachers. It seems that, simple averaging of the Q-tables is nonoptimal when the agents have different skills and experiences. Additionally, the Q-tables of the agents become equal after each cooperation step. This decreases the agents adaptability to the environment changes [33].

To overcome the described problems, a new strategy sharing method based on the expertness level of the other agents was proposed in [22].

III. WSS Method

In the WSS method [22] (Fig. 1), it is assumed that n homogeneous one-step Q-learning agents [28], [29], [31], [32] learn in some distinct environments and no hidden state is produced [7].

The agents learn in two modes: individual learning mode and cooperative learning mode (see Algorithm 1). At first, all of the agents are in individual learning mode. Agent i executes t_i learning trials. Each learning trial starts from a random state and ends when the agent reaches the goal. After a specified number of individual trials, all agents switch to cooperative learning mode.

Algorithm 1. Weighted Strategy Sharing
Algorithm for agent A_i
(1) Initialize
(2) while not End Of Learning do
(3) begin
(4) If In Individual Learning Mode then
(5) begin Individual Learning

```
(6) x_i := Find Current State ()
(7) a_i := Select Action ()
(8) Do Action (a_i)
(9) r_i := Get Reward ()
(10) y_i := Go To Next State ()
(11) V(y_i) := Max_{b ∈ action_s} Q(y_i, b)
(12) Q_i^new(x_i, a_i) := (1 − β_i) Q_i^old(x_i, a_i)
                          + β_i (r_i + γ_s V(y_i))
(13) c_i := Update Expertness (r_i)
(14) end
(15) else Cooperative Learning
(16) begin
  (17) for j := 1 to n do
    (18) c_j := Get Expertness (A_j)
  (19) Q_i^new := 0
  (20) for j := 1 to n do
  (21) begin
    (22) W_ij := Compute Weights (i, j, c_1 ··· c_n)
    (23) Q_j^old := Get Q(A_j)
    (24) Q_i^new := Q_i^new + W_ij * Q_j^old
  (25) end
(26) end
(27) end
```

In cooperative learning mode, each learning agent assigns some weights to the other agents' Q-tables with respect to their relative expertness. Then, each agent takes the weighted average of the others' Q-tables and uses the resulted table as its new Q-table[1]

$$Q_i^{new} \leftarrow \sum_{j=1}^{n} [W_{ij} * Q_j^{old}]. \tag{1}$$

A. Some Expertness Criteria

In the WSS method, W_{ij} is a measure of agent i reliance on the knowledge and the experiences of agent j. Here we argue that this weight is a function of the agents relative expertness.

In the strategy sharing method [30], expertness of the agents are assumed to be equal. Nicolas Meuleau [20] used the user judgment for specifying the expert agent. This method requires continuous human supervision.

In [2], different but fixed expertness values are assumed for the agents. However, differences in the expertness values may change during the learning process and are not constant.

Yamaguchi *et al.* [33] specified the expert agents by means of their successes and failures during current n moves and considered the expertness criterion as an algebraic sum of the reinforcement signals in that time interval. This means that more successes and fewer failures are considered a sign of a higher degree of expertness. This expertness measuring method is not optimal in some situations.

For example, the agent that has faced many failures has some useful knowledge to be learned from it. In other words, it is possible that this agent does not know the ways arriving at the goal, but it is aware of those not leading to its target and can avoid them. Also, an agent at the beginning of its learning process is less expert than those learned for a longer time and naturally has confronted more failures.

Considering the discussions, six expertness measures are introduced. These measures include the following.

1) **Normal (Nrm)**: An algebraic sum of the reinforcement signals

$$c_i^{Nrm} = \sum_{t=1}^{now} r_i(t). \tag{2}$$

2) **Absolute (Abs)**: A sum of the absolute value of the reinforcement signals

$$c_i^{Abs} = \sum_{t=1}^{now} |r_i(t)|. \tag{3}$$

3) **Positive (P)**: A sum of the positive reinforcement signals

$$c_i^{P} = \sum_{t=1}^{now} r_i^{+}(t)$$

$$r_i^{-}(t) = \begin{cases} 0, & \text{if } r_i(t) \le 0 \\ r_i(t), & \text{otherwise.} \end{cases} \tag{4}$$

4) **Negative (N)**: A sum of the absolute value of the negative reinforcement signals

$$c_i^{N} = \sum_{t=1}^{now} r_i^{-}(t)$$

$$r_i^{-}(t) = \begin{cases} 0, & \text{if } r_i(t) > 0 \\ -r_i(t), & \text{otherwise.} \end{cases} \tag{5}$$

5) **Gradient (G)**: Changes in the received reinforcement signals since the last cooperation time

$$c_i^{G} = \sum_{t=c}^{now} r_i(t) \tag{6}$$

where c is the start time of the individual learning mode.

6) **Average Move (AM)**: A reverse number of moves each agent does to reach the goal

$$c_i^{AM} = \left(\sum_{trial=1}^{n_{trial}} m_i(\text{trial})/n_{trial} \right)^{-1} \tag{7}$$

where trial is the trial number, n_{trial} is the current trial, and m_i(trial) is the number of moves that each agent has done to reach the goal.

Nrm criterion gives more credit to those who have more successes and fewer failures. Abs considers both rewards and punishments as a sign of being experienced. P disregards experiences not resulted in achieving the goal and considers the successful experiences only. N formula looks at unsuccessful tries only and assigns a higher expertness value to those experiencing more failures. AM is an indirect way to measure the expertness and considers the average number of actions the agent does to reach the goal. G looks at the trend of improvement in the agent performance in its recent actions and does not look directly at its

[1]Multiplication (*) and summation (+) operators must be specified based on the knowledge representation method.