Learning from Actions Not Taken: A Multiagent Learning Algorithm

(Extended Abstract)

Newsha Khani
Oregon State University
khanin@engr.oregonstate.edu

Kagan Tumer
Oregon State University
kagan.tumer@oregonstate.edu

ABSTRACT

Learning in multiagent systems is generally slow because the agent has to extract its correct policy through not only through its interaction with the environment, but also from its interactions with other learning agents. In this paper, we present an approach that significantly improves the learning speed in multiagent systems by allowing an agent to update its estimate of the rewards for all its available actions, not just the action that was taken. Our results show that the rewards on such "actions not taken" are beneficial early in training, particularly when agent teams are leveraged to estimate those rewards.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Algorithms, Performance

Keywords

Multiagent Systems, Multiagent Learning

1. LEARNING

Learning in large multiagent systems is a critical area of research with applications ranging from robocup soccer, to rover coordination, to air traffic management [3, 4]. What makes this problem particularly challenging is that the agents in the system provide a constantly changing background in which each agent needs to learn its task. In this paper, we explore the concept of agents learning from actions they do not take by estimating the rewards they would have received had they taken those actions. These counterfactual rewards are estimated using the theory developed for structural credit assignment, and prove effective in the congestion games. We use congestion games to test our algorithm as such games provide an environment where agents need to coordinate their actions, rather than learn to take particular actions. This type of problem is ubiquitous in routing domains (e.g., on a highway, a particular lane is not preferable to any other lane, but what matters is how many others are using a particular lane). The system performance is quantified by a **full system reward** function G. This reward is a function of the full system state z (e.g., the joint action of all agents in the system), and is given by:

$$G(z) = \sum_{k=1}^{n} x_k e^{\frac{-x_k}{C}} \quad (1)$$

where: n is the number of possible actions, x_k is the number of agents taking action k, and C is a real-valued parameter that represents the capacity of the resource.

The agent actions in this problem is to select a resource. The learning algorithm for each agent is a simple reinforcement learner (action value). Each agent keeps an n-dimensional vector providing its estimates of the reward it would receive for taking each possible action, and picks an action probabilistically based on those values [2]. Having each agent receive the full system reward leads to slow learning [1]. As a consequence, in this work, we use the **difference reward** as a starting point [1]:

$$D^i(z) = G(z) - G(z - z_i) \quad (2)$$

where $z - z_i$ specifies the state of the system without agent i (we use zero padded vector addition on states in this paper).

2. ACTION NOT TAKEN (ANT) REWARDS

Though the difference reward given in Equation 2, provides a reward tuned to an agent's actions, it is still based on an agent sampling each of its actions a (potentially large) number of times. In this work, in order to increase the learning speed, we introduce the concept of action-not-taken rewards, or ANT rewards. The goal with ANT rewards is to provide estimates of how the system would have turned out had an agent taken a particular action. The mathematics that allow the computation of the difference reward can be used to compute this type of reward. For an agent i who selected action a at this step, the counterfactual reward for action b is given by:

$$D^{i \to b}(z) = G(z - z_i^a + z_i^b) - G(z - z_i^a) \quad (3)$$

where $D^{i \to b}$ is the reward for agent i taking action b; z_i^a is the state component where agent i has taken action a; z_i^b is the state component where agent i has taken action b.

For an agent taking action a, the second term of Equation 3 ($G(z - z_i^a)$) is the same as the second term of Equation 2. Namely the reward for the state where agent i has not taken the particular action that it took. The first term though is the key to the ANT-reward, and gives the reward

Cite as: Learning from Actions Not Taken: A Multiagent Learning Algorithm, (Extended Abstract), Newsha Khani, Kagan Tumer, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 1277–1278

that would have resulted had agent i taken action b rather than action a. Utilizing this structure, D^i_{ANT} becomes:

$$D^i_{ANT} = \begin{cases} G(z) - G(z - z^a_i) & \text{for } i \to a \\ G(z - z^a_i + z^b_i) - G(z - z^a_i) & \text{for } i \to b \neq a \end{cases} \quad (4)$$

where $i \to a$ means that agent i has taken action a. Note, the removal of the state in which agent i has taken action a in the second term represents the system state without agent i. Because agent i had taken action a, this removal results in a state where agent i has taken neither action a nor action b (which it has never taken). Hence the second term is the same for both conditions conditions of Equation 4. Figure 1 shows that agents using ANT-rewards early significantly speeds up the learning process, though does not result in agents reaching higher performance (120 agents, C=6, n=5, averaged over 20 runs).

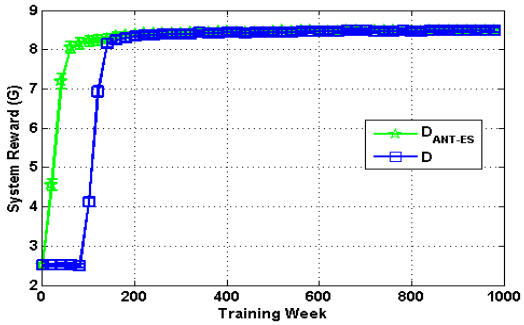

Figure 1: System performance for ANT with early stopping (ES). Stopping occurs at week 6.

3. TEAM ESTIMATES FOR ANT

The computation of the second term of Equation 4 can be improved if agents use information from "team members" to update their reward estimates. Extending this further, instead of using the team members as information sources, all team members taking a particular answer can receive the same reward. The learning strategy is to use team information only during the first weeks (three in the reported results, but the performance is similar for minor changes to this parameter) of learning and switch to the regular difference reward (Equation 2) for the rest of the training period. The key aspect of this approach is that the team members measure the impact of a team not taking a particular action, rather than an individual agent. As a result, agents learn with their team in a smaller state, leading to:

$$D^i_{Team} = \begin{cases} G(z) - G(z - z^a_{T_i}) & \text{for } T_i \to a \\ G(z - z^a_i + z^b_j) - G(z - z^b_{T_i} - z^a_i) & \text{for } i \to b \in T^i \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where $z^a_{T_i}$ is the state component of team members of agent i taking action a. In this formulation, the impact of all of agent's i teammates are removed before the reward is calculated. Note in this case, unlike in Equation 4, the second term is different for the two actions. This is because this term estimates the impact of removing all team members of i that had taken a particular action. When agent i changes its action, this also changes the team members taking the same action as i. For the action a selected by agent i, we only need to remove all its team members who took that action. But to find the counterfactual reward for action b, we need to remove the actual action of agent i (action a) and then remove the team members who had taken action b. Though conceptually similar to previous rewards, the presence of team members leads to this subtle difference in the computation of the team action-not-taken reward.

Providing a weighting factor for the second term of the counterfactual reward further improves this process, by making the reward estimate more accurate. This leads to a weighted team reward for agent i and action b by replacing the condition $(i \to b \in T^i)$ in Equation 5 by:

$$D^{i \to b}_{WT} = G(z - z^a_i + z^b_i) - \mu_{|T^i_{k_{i \to b}}|} \cdot G(z - z^b_{T_i} - z^a_i) \quad (6)$$

where $\mu_{|T^i_{k_{i \to b}}|}$ is the average number of team members taking action b.

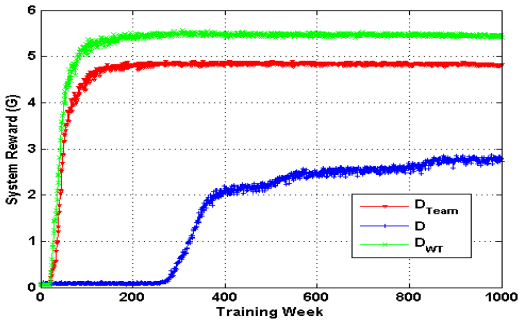

Figure 2: System performance for the weighted team rewards.

Figure 2 explores this idea for 460 agents in a system with 7 actions and a capacity of 4. Because the optimal capacity in this case is $7 \times 4 = 28$, this creates significant congestion. The results show that traditional D starts to suffer in this case, and that the weighted D_{WT} outperforms D_{TEAM}.

In summary, the use of estimated rewards for actions not taken by agents provides a significant speedup in learning, as well as improved system performance, particularly when agent use teams to provide accurate estimates for those rewards. The extensions of this work include explicit communication structures among the team members, agents adopting particular roles within a team, and better estimates of counterfactuals to improve the reward estimates. We are currently investigating all three extensions of this work.

Acknowledgements: This work was partially supported by AFOSR grant no. FA9550-08-1-0187.

4. REFERENCES

[1] A. K. Agogino and K. Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic environments. *Journal of Autonomous Agents and Multi Agent Systems*, 17(2):320–338, 2008.

[2] N. Khani and K. Tumer. Fast Multiagent Learning: Cashing in on Team Knowledge. In *Intel. Engr. Systems Though Artificial Neural Nets* 18:3–11, 2008.

[3] P. Stone. *Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer*. MIT Press, Cambridge, MA, 2000.

[4] K. Tumer and A. Agogino. Distributed agent-based air traffic flow management. In *Proc. of the 6th Intl. Jt. Conf. on Autonomous Agents and Multi-Agent Systems*, pp 330–337, Honolulu, May 2007. **Best Paper Award**.