Concurrently Learning Neural Nets: Encouraging Optimal Behavior in Cooperative Reinforcement Learning Systems

Nancy Fulda and Dan Ventura
Department of Computer Science
Brigham Young University
owens@cs.byu.edu, ventura@cs.byu.edu

***Abstract*—Reinforcement learning agents interacting in a common environment often fail to converge to optimal system behaviors even when the individual goals of the agents are fully compatible. Claus and Boutilier have demonstrated that the use of joint action learning helps to overcome these difficulties for Q-learning systems. This paper studies an application of joint action learning to systems of neural networks. Neural networks are a desirable candidate for such augmentations for two reasons: (1) they may be able to generalize more effectively than Q-learners, and (2) the network topology used may improve the scalability of joint action learning to systems with large numbers of agents. Preliminary results indicate that neural nets benefit from joint action learning in the same way that Q-learners do.**

I. Introduction

Reinforcement learning is a sub-field of machine learning in which agents use numerical rewards provided by the environment to estimate the utility of performing a specific action in a given state. This differs from more general machine learning algorithms such as KNN, the perceptron training rule, and Bayesian learners in that the reinforcement learner is not provided with labeled training sets or error functions which associate a correct output with each training instance. Instead, the agent explores its environment by trial and error and attempts to learn how to maximize its numerical rewards.

This approach to learning is advantageous because the designer is spared the effort of solving (or partially solving) the task in advance in order to obtain training data. Instead, some objective task criteria is used to create a reward function $r(s,a)$ which returns a reward r for each action a executed by the agent in state s. This reward function need not be deterministic. The agent then attempts to learn a policy $\pi(s) = a$ that maximizes future discounted reward.

A gradient descent learning rule can be applied to reinforcement learning by defining the target function $T(s,a)$ to be the expected time-discounted reward attainable by executing action a in state s. This function cannot be calculated without a model of the environment, but it may be incrementally approximated by letting

$$\hat{T}(s,a) = r(s,a) + \gamma argmax_{a'} \hat{T}(s',a') \qquad (1)$$

where $0 = \gamma < 1$ is a discount factor and s' is the state transitioned to by executing action a in state s. When only a single environmental state exists, this approximation can be simplified to $\hat{T}(a) = r(a)$.

Successive approximations of the target function are not guaranteed to converge for neural networks; nevertheless, this and similar approaches have proven fruitful in the past. Reinforcement learning neural networks have been successfully applied to such applications as simulated traffic light control [9], robotic hand reaching [8], planning and landmark navigation [1], and pole balancing tasks [7], [3].

Unfortunately, direct applications of reinforcement learning algorithms to the multiagent domain are not always successful. As each agent updates its utilities (and correspondingly, its behavior) the perceived environmental transitions and rewards experienced by the other agents may change. This creates a more challenging learning environment. Additionally, if the respective tasks assigned to each agent require a high level of inter-agent cooperation to achieve, the agents may settle to sub-optimal equilibrium points in the solution space.

Proposed approaches to overcoming these difficulties include allowing the agents to perceive each others' action selections and/or rewards [2], [4], biasing the agents' exploration of the environment [2], and establishing social conventions [5]. However, most of this research has been performed within the context of dynamic programming and the Q-learning algorithm [10]. Very little attention has been given to reinforcement learning neural networks.

This paper studies an application of joint action learning [2] to neural networks. Preliminary results indicate that neural nets benefit from joint action learning in the same way that Q-learners do. This is encouraging because the neural net topology may scale well to systems with large numbers of agents: for a system of n agents with k possible actions each, the neural network joint action learners require only kn weights, whereas Q-learning joint action learners must store k^n Q-values.

II. Joint Action Learning

Joint action learners are agents that are able to perceive the action selections of other agents in the system, and are thus able to learn utilities for joint actions. Joint action learners can be contrasted with independent learners, who learn utilities only for their own actions, without regard to the action selections of the other agents in the system. Previous research has shown that agents who can perceive the joint action space often perform better than independent learners on coordination tasks [2], [4], [6].

An interesting question that arises with joint action learning is how the agents should use their joint action utilities to determine which individual action should be executed in the next time step. One approach to this problem is to use *fictitious play*, in which each agent maintains a history of the number of times each of the other agents in the system has executed each possible action. This history is then used to estimate the probability that a given agent will execute a specific action in the future based upon its past behavior. The net utility of an individual action can then be calculated as a weighted average of all of the joint actions it contributes to, with the probabilities used as the weighting factor.

An alternative to fictitious play is to introduce an optimistic assumption into the system [5]. In this approach, each agent assumes that all of the agents in the system share the same rewards. Thus, if a joint action is desirable for one agent, it is equally desirable for all other agents in the system. In this case, the net utility of an individual action can be calculated as the utility of the best joint action which the individual action contributes to. In essence, the agent performs a max operation on the joint actions rather than taking a weighted sum. In this paper, this method is referred to as *optimistic action selection*.

III. Joint Action Learning for Neural Networks

The potential benefits of joint action learning are clear. Agents that are capable of perceiving the complete joint action space are potentially able to solve problems that independent learners cannot. Even problems which independent learners are capable of solving may be solved more quickly if joint action learners can utilize their extra information. These potential benefits come with a price, however. The size of the joint action space grows exponentially with the number of agents. In Q-learning systems with large numbers of agents, the joint action space representation for each agent may quickly become intractable.

One way to address this problem is to apply the joint action learning paradigm to a different learning architecture. Neural networks present themselves as a viable option because their representation of the action space is parametric, and thus less susceptible to exponential growth in the face of large state and action spaces. However, neural networks are not guaranteed to converge to the true target function, and the task of convergence frequently becomes a lengthier and less certain process as the complexity of the model increases. Thus, a critical question arises: do the benefits of joint action learning in a system of neural networks outweigh the drawbacks of the necessary increase in model complexity? This paper presents empirical evidence that, at least in some cases, they do.

Figure 1 presents two possible joint action learning topologies for neural networks, depicted for convenience and clarity as a two-layer network in which two agents (A and B) each have two action options (-1 and 1). Only

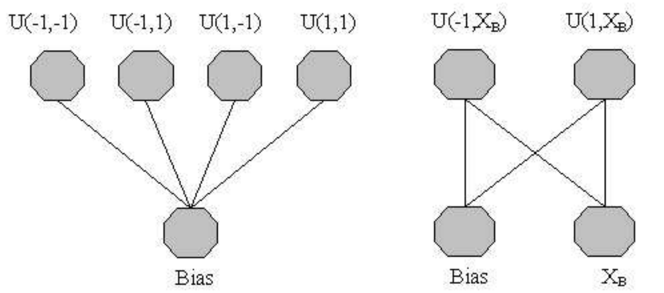

Fig. 1. Two possible topologies for agent A in a system consisting of two agents, A and B, each of which has two possible actions, -1 and 1. For simplicity and clarity, only a single environmental state is assumed and no hidden layer is depicted.

agent A's possible topologies are shown, and joint actions are represented with agent A's action listed first. When agent B's action selection is variable, it is represented as X_B. Thus the term $U(-1, X_B)$ represents the utility of agent A executing action 1 when agent B executes action X_B. Expansions of both topologies to include hidden layers, more than two output actions, more than two interacting agents, or multiple environmental states are not shown here, but are relatively easy to design.

The network topology depicted on the left-hand side of Figure 1 is quite straightforward, but it is more like a nonparametric approximation method than a parametric one: each weight of the network will simply converge towards the utility of the corresponding joint action. This prevents much of the generalization one would hope to achieve by using a neural network. Not surprisingly, this topology also will not scale well to large numbers of agents: the number of weights required (excluding a hidden layer) is k^n, where n is the number of agents in the system and k is the number of actions available to each agent.

The network topology depicted on the right-hand side of Figure 1 has more potential. Here, agent B's action selection is modeled implicitly as an extra input node instead of being explicitly modeled in combination with agent A's action selection. This allows the unique characteristics of neural networks to manifest themselves. This topology also scales more effectively to systems with large numbers of agents, requiring only kn weights. It is this topology which was utilized for the experiments in this paper.

IV. Agent Implementation

This paper presents a comparison of the performance of three types of agents: independent learners, joint action learners using fictitious play, and joint action learners using optimistic action selection. The independent learners are implemented as a two-layer network with a single bias input and two output nodes which represent the estimated utilities of performing actions -1 and 1, respectively. The joint action learning agents are implemented using the network topology shown on the right-hand side of Figure 1. The networks use a gradient descent training rule with sigmoidal activation functions for the outputs.

	a_{-1}	a_1
b_{-1}	1	0
b_1	0	1

Fig. 2. Joint rewards received by agents A and B for performing joint action (a_i, b_j) in a simple coordination task.

Training of the independent learners is fairly elementary. In each interaction, each agent selects an action for execution, and each agent receives a reward r based on the joint action executed. The agent then uses r to calculate the error of the output node corresponding to its executed action. The error of the output node corresponding to the un-executed action is assumed to be 0.

Because the joint action learning networks cannot predict the actions of the other agents in advance, they must make a prediction about the behavior of the other agent based either on the fictitious play algorithm or the optimistic action selection algorithm. This prediction is used to select an action for execution. Once the interaction has taken place, the joint action learner can perceive the action that was actually taken by the other agent. This is fed into the network as an input and the activations of the output nodes are calculated. These activations are then used to calculate errors based on the reward value r, just as was done for the independent learners.

For all agents, action selection was based on a Boltzmann exploration strategy. In Boltzmann exploration, the probability of selecting a given action is proportional to the estimated utility of that action, and the relative probability of selecting the best action increases as the temperature value, T, is decreased. For consistency with the work of other researchers on Q-learning joint action learners, we used an initial Boltzmann temperature of $T = 16$ and decayed the temperature by a multiplicative factor of 0.9 after each interaction [2].

V. A Simple Coordination Task

Figure 2 shows the joint action payoff matrix for two agents learning a simple coordination task. When both agents select the same action index, they both receive a reward of 1. Otherwise, they both receive a reward of 0.

It can be difficult for reinforcement learning agents to learn optimal solutions to this task because there is no clearly dominant action selection for either agent. Rather, the utility of performing a given action is directly dependent on the action selection of the other agent. When a Boltzmann exploration strategy is used, independently learning agents generally settle into one of the two optimal joint actions. The question is whether agents using a joint action learning strategy can settle into an optimal joint action more quickly.

Experimental results are shown in Figure 3. Consistent with the results reported by Claus and Boutilier, joint action learning with fictitious play performs slightly better than independent learning [2]. However, the fictitious play algorithm essentially computes the same utility values as

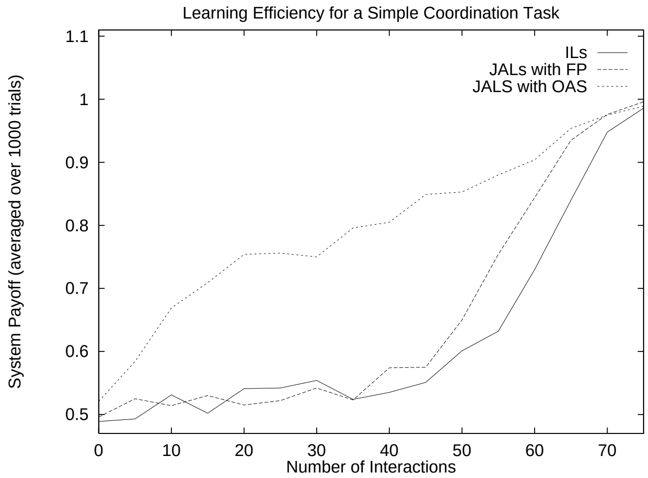

Fig. 3. Learning efficiency of independent learners (ILs), joint action learners with fictitious play (JALs with FP) and joint action learners with optimistic action selection (JALs with OAS) for the coordination task depicted in Figure 2. System payoff was calculated by averaging the individual payoffs received by each agent.

	a_{-1}	a_1
b_{-1}	(1,1)	(.6, 0)
b_1	(0, .6)	(.6, .6)

Fig. 4. Rewards received by agents A and B for performing joint action (a_i, b_j) in a utility distinction task. Agent A's reward is listed first.

the independent learners, thus minimizing the joint action learners' ability to capitalize on their extra knowledge.

Joint action learners using the optimistic action selection algorithm, in contrast, perform significantly better than both other implementations. The difference in this case is caused by the optimistic assumption that all members of the system share the same joint action preferences. Although this assumption can be somewhat limiting, it effectively permits the joint action learners to exploit their additional knowledge.

VI. A Utility Distinction Task

We now consider a task in which the payoffs received by the agents are not always identical. Both agents receive a reward of 0.6 for performing action 1, regardless of the behavior of the other agent. But the reward for performing action -1 is dependent upon the other agent's behavior: if the other agent also chose action -1, then a reward of 1 is received. If not, a reward of 0 is received. This payoff structure is depicted in Figure 4.

This payoff structure is challenging for reinforcement learners because, during initial exploration, each agent receives an average reward of 0.5 for performing action -1, while the average reward for performing action 1 is 0.6. Thus, action 1 appears to be the better option, even though increased rewards could be obtained if both agents selected action -1.

Experimental results for this task are shown in Figure 5.

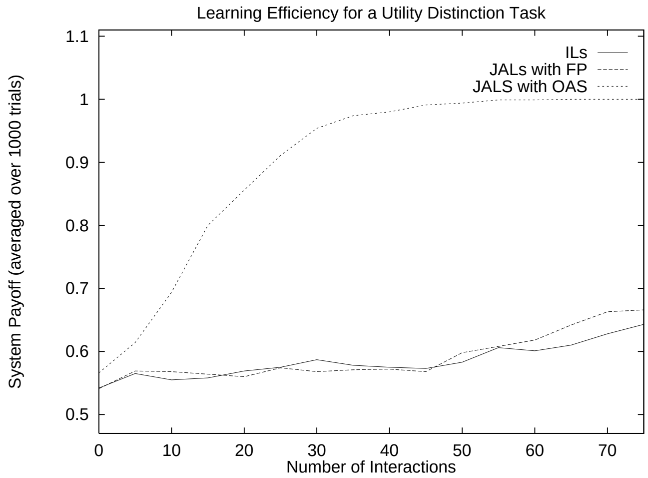

Fig. 5. Learning Efficiency of independent learners (ILs), joint action learners with fictitious play (JALs with FP) and joint action learners with optimistic action selection (JALs with OAS) for the utility distinction task depicted in Figure 4. Payoff for the system was calculated by averaging the individual payoffs received by each agent.

Again, joint action learners using the fictitious play algorithm do not significantly out-perform individual learners. The reason is that fictitious play makes no assumptions about the goals of the other agent. While this approach is highly applicable in adversarial learning situations, it fails to find the optimal solution for the cooperative tasks studied in this paper.

Joint action learning with optimistic action selection again outperforms both other algorithms. In this case, the optimistic assumption is particularly useful because it allows the agents to immediately concentrate on the maximum possible reward provided by the problem structure. This ability to quickly converge to mutually desirable joint actions makes the optimistic action selection algorithm particularly applicable to distributed learning systems in which all agents share the same goals.

VII. Concluding Remarks

In Q-learning systems, joint action learning has shown itself to be a potentially powerful tool for improving the coordination of multiagent systems. However, the theoretical applicability of this method to systems with large numbers of agents is limited by the exponential expansion of the joint action space as the number of agents in the system increases. Joint action learning neural networks may be less susceptible to this problem because they do not explicitly represent the entire joint action space.

The objective of this research was to determine whether neural network architectures benefit from joint action learning in the same way that Q-learners do. Empirical results indicate that this is indeed the case. This is significant because it provides a foundation for the investigation of joint action learning neural networks applied to large distributed systems. In addition, empirical results indicate that if all agents in the system share a common goal, then optimistic action selection is a good method for determining the utility of individual actions based on the joint action utilities.

The next step in this research is to apply the joint action learning neural network implementation to systems with large numbers of agents, thus determining whether the algorithm scales as well in practice as it does in theory. Evaluations of the joint action learning topology's effectiveness for networks with hidden nodes, large numbers of possible agent actions, or multi-state environments would also be desirable.

References

[1] G. Baldassarre. Coarse planning for landmark navigation in a neural-network reinforcement learning robot. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2001.

[2] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI*, pages 746–752, 1998.

[3] D. Hougen, J. Fischer, and D. Johnam. A neural network pole balancer that learns and operates on a real robot in real time. In *Proceedings of the MLC-COLT Workshop on Robot Learning, pages 73–80*, 1994.

[4] J. Hu and M. Wellman. Multiagent refinforcement learning: theoretical framework and an algorithm. In *Proceedings of the 15th International Conference on Machine Learning*, pages 242–250, San Francisco, 1998. Morgan Kaufman.

[5] Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the 17th International Conference on Machine Learning*, pages 535–542, San Francisco, CA, 2000. Morgan Kaufman.

[6] Michael Littman. Markov games as a framework for multiagent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, 1994.

[7] Jürgen H. Schmidhuber. Making the world differentiable: On using self-supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments. Technical Report FKI-126-90(revised), Institut für Informatik, Technische Universität Münschen, 1990.

[8] Katsunari Shibata, Masanori Sugiska, and Koji Ito. Hand reaching movement acquired through reinforcement learning. In *Proceedings of the 2000 Korea Automatic Control Conference, 90rd (CD-ROM)*, 2000.

[9] Thomas Thorpe. Vehicle traffic light control using sarsa, Masters Thesis, Colorado State University, 1997.

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