Balancing Multiple Sources of Reward in Reinforcement Learning

Christian Shelton

Artificial Intelligence Laboratory
Massachusetts Institue Of Technology
Cambridge, Massachusetts 02139

http://www.ai.mit.edu

The Problem: Consider a home entertainment system enjoyed by multiple users at the same time (or any computer controlling public resources). We might wish the system to automatically select suitable television shows based on who is present in the room. To train such a system, each user might have some mechanism by which to express reward to the system. This differs from the typical reinforcement learning (RL) framework where there is only a single source of reward. The sources themselves may not all be present at the same time (users may enter or leave the room). We would like the system to adapt to the changing users automatically and not have to be retrained for each combination.

Motivation: This multiple reward framework is natural anytime there are public resources governed by an agent. Simply combining the reward signals (additively for instance) to produce a single reward to train a traditional RL system is not acceptable. The rewards given by different users are not comparable. For example, one user might arbitrarily decide to give twice as much punishment as another for things he does not like; this does not mean that this user should have twice the control over the agent's actions. It also benefits the learning agent to keep track of which rewards came from which sources so that, when the set of sources present changes, the agent can adapt without having to relearn the correct policy for this new set.

Previous Work: This work is related to RL research involving multiple learning agents [1, 2, 3, 4]. Both this work and that of multiple agents adopt ideas and solutions from game theory. However, the basic problems differ. In this work, all of the computation is done by a single agent, thus making the algorithms simpler.

Approach: We combine game theory with reinforcement learning. Each reward source is given a fixed amount of control over the output of the agent by a voting scheme. However, since the optimal votes for a given source depend on the other sources present, we do not learn the votes directly. Such learned values would be meaningless once the set of present sources changed (from our example above, when a new person entered or left the room). Instead, we learn the sources' preferences over the net output of the agent. This induces a game (in the sense of game theory) in which each source would like to pick its vote to cause the output policy of the agent to be optimal according to its preferences. The votes are then chosen by the algorithm to be a Nash equilibrium of this game. This insures that it is in each source's best interest to train the agent according to the source's true preference.

We have developed an algorithm for partially-observable discrete worlds [5].

The left side of figure 1 demonstrates a toy world used to test our algorithm. One source would like the agent to move to state 1 and the second source would like the agent to move to state 9. Once in state 1 or 9, the agent cannot move backwards towards state 5 but must instead continue through the lower states. The right side of figure 1 shows the solution found by our algorithm after learning on simulated runs through the environment. The left column is the learned parameters for the preferences: the top is the importance of the states to each source and the bottom is the preferred direction at each state (high values are preferences to move to the right, low values are preferences to move to the left). The middle column is the votes chosen by the algorithm based on preferences. The top is the weight of the vote for each state. The bottom is the vote for each state (high values are votes for moving right). The right column is the resulting policy.

Notice that both sources agree that for states 10–15, the agent should move towards state 5. They disagree for the other states. Furthermore, the first source cares more about the policy for states near state 1 while the second source cares more about the policy for states near state 9. As shown in the left column, this matched the learned data well. The middle column shows how the Nash equilibrium automatically picks a policy whereby the source's votes collaborate for states 10-15, allow one agent to win for states 1–4 and 6–9, and disagree on state 5. The resulting policy decides

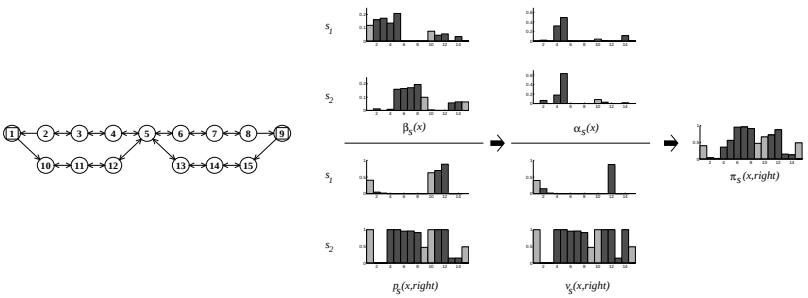

Figure 1: One-way door problem. On the left, the state diagram. At every state there are two actions (left and right) available to the agent. In states 1, 9, 10, and 15 where there are only single outgoing edges, both actions follow the same edge. With probability 0.1, an action will actually follow the other edge. Source 1 rewards entering state 1 whereas source 2 rewards entering state 9. On the right, the solution. From left to right: the source's ideal policies, the votes, and the final agent's policy. Light bars are for states for which both actions lead to the same state.

at state 5 randomly which way to go and then proceeds quickly around the loop back to state 5. This is optimal for pleasing both agents.

Two options exist for trying to solve this problem with traditional RL methods. First, we could have combined the rewards linearly and ignored the multiple sources. This would have resulted in a policy that would have traversed either the left or the right loop but not both. Second, we could have learned two separate optimal policies for each source and then taken their average. This would have resulted in a policy that randomly bounces around in states 2–8 and seldom reaches either goal state. Neither of these methods produces the optimal result demonstrated by our technique.

Difficulty: Finding the votes given a learned preference is hard. For discrete worlds we have to solve an infinite game (a generalization of a matrix game). For continuous worlds, the problem is harder. Whereas for matrix games many techniques for finding Nash equilibria are known, very little is known about the solutions for infinite games and even fewer algorithms exist.

Impact: This work demonstrates, in a solid mathematical framework, how an agent can balance multiple goals, user preferences, or tasks into a single solution. We hope this work will also lead to insights into multiple-agent learning systems.

Future Work: We are currently working to extend the observation space from the discrete world to the continuous world. We are applying this algorithm to financial markets and software internet agents.

Research Support: This research is sponsored by a grant from Office of Naval Research under contract No. N00014-93-1-3085, ONR under contract No. N00014-95-1-0600, National Science Foundation under contract No. IIS-9800032, and NSF under contract No. DMS-9872936. Additional support is provided by: AT&T, Central Research Institute of Electric Power Industry, Eastman Kodak Company, Daimler-Benz AG, Digital Equipment Corporation, Honda R&D Co., Ltd., NEC Fund, Nippon Telegraph & Telephone, and Siemens Corporate Research, Inc.

References:

[1] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. of the 15th Int. Conf. on Machine Learning*, pages 242–250, 1998.

[2] M. Kearns, Y. Mansour, and S. Singh. Fast planning in stochastic games. In *Proc. of the 16th Conference on Uncertainty in AI*, 2000. to appear.

[3] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of the 11th International Conference on Machine Learning*, pages 157–163, 1994.

[4] S. Singh, M. Kearns, and Y. Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proc. of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000. to appear.

[5] C.R. Shelton. Balancing multiple sources of reward in reinforcement learning. in *NIPS 13*, 2000. to appear.