Modeling Difference Rewards for Multiagent Learning

(Extended Abstract)

Scott Proper
Oregon State University
Corvallis, OR 97331, USA
proper@eecs.oregonstate.edu

Kagan Tumer
Oregon State University
Corvallis, OR 97331, USA
kagan.tumer@oregonstate.edu

ABSTRACT

Difference rewards (a particular instance of reward shaping) have been used to allow multiagent domains to scale to large numbers of agents, but they remain difficult to compute in many domains. We present an approach to modeling the global reward using function approximation that allows the quick computation of shaped difference rewards. We demonstrate how this model can result in significant improvements in behavior for two air traffic control problems. We show how the model of the global reward may be either learned on- or off-line using a linear combination of neural networks.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms, Performance, Experimentation

Keywords

Multiagent Coordination, Reward Shaping, Scaling, Air Traffic Control, Function Approximation, Neural Networks

1. INTRODUCTION

Reinforcement learning in large multiagent systems is particularly challenging because the agents in the system provide a constantly changing environment in which each agent needs to learn its task. Difference rewards which encourage good agent behavior by rewarding actions that are closely aligned with the desired overall system behavior. Difference rewards have been shown to perform very well in multiagent domains [1]. However it is not always possible to calculate the value of the difference reward, or even approximate it, due to complex system dynamics.

We mitigate this problem by using function approximation techniques to approximate the global reward signal, which we may then use to calculate an approximate difference reward. We use Tabular Linear Functions [2] to model the value of the global (system) reward. This model is then be used to calculate the difference reward. Our results show that we can greatly improve performance over learning on the system reward directly, and in some cases even outperform the true model of the reward signal.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

2. AIR TRAFFIC SIMULATION

We developed FEATS (Fast Event-based Air Traffic Simulator) to quickly simulate thousands of aircraft of different characteristics taking off from airports, navigating via waypoints and airways to their destination airport, and landing. This simulator is optimized for speed, simulating 26,000 flights/second. Individual simulations require a fraction of a second, allowing efficient experimentation with machine learning techniques. As in [3] we choose to make "meter fixes", rather than aircraft, into learning agents. We manage traffic by controlling aircraft separation distances – called "Miles in Trail" (MIT) separations – at meter fixes surrounding busy airports.

We used a linear combination of terms for measured congestion and delay to calculate the global (system) reward $G(z) = -(B(z) + \alpha C(z))$ where $B(z)$ is the delay penalty for all aircraft in the system, and $C(z)$ is the total congestion penalty. The relative importance of these two penalties is determined by the value $\alpha = 5$. $B(z)$ is the sum of minutes of delay suffered by all aircraft. $C(z)$ is given by $C(z) = \sum_{p \in P} \int_T \Theta(k_{p,t} - c_p)(k_{p,t} - c_p)^2 dt$, where P is the set of airports monitored by the simulation, $k_{p,t}$ is the number of aircraft that have landed in the past 15 minutes (a rolling time window), c_p is the capacity of airport p as defined by the FAA, and $\Theta(\cdot)$ is an indicator function that equals 1 when its argument is greater or equal to zero, and has a value of zero otherwise. Thus $C(z)$ penalizes states where airports become overcapacity. The quadratic penalty provides strong feedback to return the airport to FAA mandated capacities. We use an integral over time due to the fact that our simulation occurs in real time.

3. REWARD MODELING

The learning algorithm for each agent is a simple reinforcement learner using standard TD-update. Following [3], we use the **difference reward** $D_i(z) = G(z) - G(z - z_i)$ to provide the reward signal to each agent, where $z - z_i$ is a modified version of the normal action vector z in which agent i takes a "default" action (in our case, setting its "Miles in Trail" value to zero). As it is not possible to analytically compute this value for the air traffic domain, we learn an approximation $D_i(z) \approx v(z) - v(z - z_i)$, where $v(z) \approx G(z)$. We adapt "Tabular Linear Functions" (TLFs) from previous work [2] to approximate $G(z)$ in this manner.

TLFs have been shown to provide a simple, flexible framework to consider and incorporate different assumptions about the functional form of an approximated function and the set of relevant features. TLFs have previously been used for value function approximation. In this work, we use it to approximate the reward model.

A TLF is a sum over several terms. Each term is given by multiplying a weight and feature value, as with any linear function. Unlike standard linear functions, the weight of each term is given by an arbitrary function of other discretized (or "nominal") features.

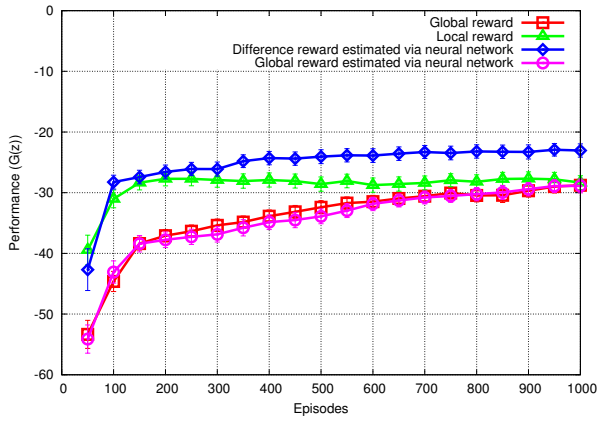

Figure 1: NAS simulation results: The approximated D reward outperforms other approaches, while the approximated global reward show that the approximation used is very accurate.

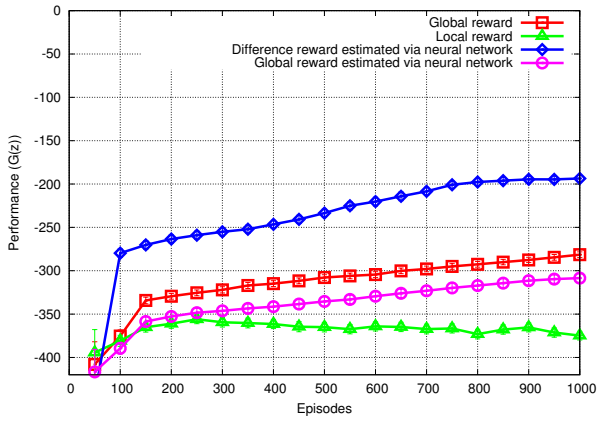

Figure 2: Results for 400 airports and 395 agents in the generic air traffic domain show that D does even better at larger scales (four times the size of the NAS).

More formally, a tabular linear function is represented by Equation 1, which is a sum of n terms. Each term is a product of a linear feature ϕ_i and a weight θ_i. The features ϕ_i need not be distinct from each other. Each weight θ_i is a function of m_i nominal features $f_{i,1}, \ldots, f_{i,m_i}$.

$$v(z) = \sum_{i=1}^{n} \theta_i(f_{i,1}(z), \ldots, f_{i,m_i}(z))\phi_i(z) \quad (1)$$

A TLF using tables to store the value of θ reduces to a linear function when there are no nominal features, i.e. when $\theta_1, \ldots, \theta_n$ are scalar values. However, this is effective only when each table is indexed by just a few nominal features. If this is not the case, we must also approximate the tables themselves. We thus applied TLFs with neural networks to approximate the reward model $G(z)$ for the air traffic simulation, using backpropogation to learn the model:

$$v(z) = \sum_{p \in P} (\theta^p_B(z_p) + \theta^p_C(z_p)) \quad (2)$$

where z_p are the actions for the agents surrounding airport p, $\theta^p_B(\cdot)$ is a neural network approximating $B_p(z) = \sum_{a \in A_p} B_a(z)$, the sum of delays over all aircraft approaching p, and $\theta^p_C(\cdot)$ is a neural network approximating $C_p(z)$, the congestion penalty for a single airport. Each network has an input node for each action taken by the n_p agents (meter fixes) surrounding that airport, $n_p + 1$ hidden units, and 1 output. Given access to the above terms of $G(z)$, we can train each network separately, allowing a more accurate approximation. Note that a meter fix may control incoming traffic to multiple airports.

4. EXPERIMENTAL RESULTS

We performed experiments testing global and difference rewards, as well as a local reward based only on information available to individual agents. Each episode simulated a single traffic "rush" from start to finish. The actions taken by the meter fixes controlled the delay each aircraft suffered as it was routed through that fix. We used TLFs with neural networks approximating each term to estimate $G(z)$ and thus $D_i(z)$. We train each network offline using 10,000 randomly-generated examples.

Figure 1 shows that the estimated $D(z)$ significantly outperforms both local and global rewards. Learning using the estimated $G(z)$ compared to the true $G(z)$ shows that the estimate is very accurate.

In addition, we scaled up our experiments to 400 airports and 14,295 flights using a generic air traffic domain in a space about four times the size of the NAS. Figure 2 shows that the performance of the estimated difference reward greatly outperforms the other methods, particularly at this huge scale. Local reward performs very poorly: it does not allow for coordination between agents, which is critical in this domain. The estimated global reward does well in comparison to the true global reward, but performance does degrade slightly at this large scale. Difference rewards handle the increase in scale far better than any other method, despite the fact that it is using a learned model rather than the "true" difference reward.

5. DISCUSSION

We have shown that although calculating the difference reward for some multiagent domains may be impractical or impossible, it may still be possible to *estimate* D by learning a reward model of $G(z)$ using function approximation. We found that a sufficiently accurate model of $G(z)$ does in fact allow us to estimate D well enough to obtain improved behavior over learning on either the local or global rewards. In the case of air traffic control, a vast database of states and actions already exists, or may be generated via sufficiently sophisticated simulations. This makes learning a model of the reward function offline a practical approach for many domains. Future work includes continued experiments with model learning and the addition of states to our air traffic simulation, allowing agents to learn how to manage and route traffic by dynamically adapting to changing conditions.

Acknowledgements

This work was partially supported by NSF Grant CNS-0931591.

6. 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] S. Proper and P. Tadepalli. Scaling model-based average-reward reinforcement learning for product delivery. In *ECML '06: Proceedings of the 17th European Conference on Machine Learning*, pages 735–742, 2006.

[3] K. Tumer and A. Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 330–337, Honolulu,HI, May 2007.