Multi-Agent State Space Aggregation using Generalized Learning Automata

Yann-Michaël De Hauwere
Peter Vrancx
Ann Nowé
Computational Modeling Lab, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussel, Belgium

Abstract

A key problem in multi-agent reinforcement learning remains dealing with the large state spaces typically associated with realistic distributed agent systems. As the state space grows, agent policies become more and more complex and learning slows. One possible solution for an agent to continue learning in these large-scale systems, is to learn a policy which generalizes over states, rather than trying to map each individual state to an action. In this paper we present a multi-agent learning approach capable of aggregating states, using associative reinforcement learners called generalized learning automata (GLA).

1. Introduction

Reinforcement learning (RL) has already been shown to be a powerful tool for solving single agent Markov Decision Processes (MDPs). Basic RL techniques are not suited for problems with very large state spaces, however, as they mostly rely on a tabular representation for policies and enumerating all possible state-action pairs is not feasible (the so called curse of dimensionality). Because of these issues, several extensions have been proposed to reduce the complexity of learning. Methods for representing the agent’s policy such as neural networks, decision trees and other regression techniques are already widely used. To our understanding, relatively little work has been done on extending RL for large state spaces to MAS, so far.

2. Generalized Learning Automata

A Generalized Learning Automaton (GLA) is an associative reinforcement learning unit. The purpose of a GLA is to learn a mapping from given inputs or contexts to actions. At each time step the GLA receives an input which describes the current system state. Based on this input and its own internal state the unit then selects an action. This action serves as input to the environment, which in turn produces a response for the GLA. Based on this response the GLA then updates its internal state.

Formally a GLA can be represented by a tuple \( (X, A, \beta, u, g, T) \), where \( X \) is the set of possible inputs to the GLA and \( A = \{a_1, \ldots, a_r\} \) is the set of outputs or actions the GLA can produce. \( \beta \in [0,1] \) denotes the feedback the automaton receives for an action. The real vector \( u \) represents the internal state of the unit. It is used in conjunction with a probability generating function \( g \) to determine the action probabilities, given an input \( x \in X \). \( T \) is a learning algorithm which updates \( u \), based on the current value of \( u \), the selected action and response \( \beta \). In this paper we use a modified version of the REINFORCE (WILLIAMS, 1992) update scheme. In (Thathachar & Sastry, 2004) it is shown, that this update mechanism, converges to local maxima of \( f(u) = E[\beta | u] \), showing that the automata find a local maximum over the mappings that can be represented by the internal state in combination with the function \( g \).

We propose to use the GLA described above in Multi-agent Reinforcement learning problems. In such a system each agent internally uses a set of GLA to learn the different regions in the state space where different actions are optimal. We use the following set-up for the GLA. With every action \( a_i \in A \) the automaton can perform, it associates a vector \( u_i \). This results in an internal state vector \( u = [u_1^\tau \ldots u_r^\tau] \) (where \( \tau \) denotes the transpose). With this state vector we use the Boltzmann distribution as probability generating function:
Of course since this function is fixed in advance and the environment in general is not known, we have no guarantee that the GLA can represent the optimal mapping. For instance when using the function given in Equation 1 with a 2-action GLA, the internal state vector represents a hyperplane. This plane separates context vectors which give a higher probability to action 1 from those which action 2. If the sets of context vectors where different actions are optimal, are not linearly separable the GLA cannot learn an optimal mapping.

To allow a learner to better represent the desired mapping from context vectors to actions, we can utilize systems composed of multiple GLA units. For instance the output of multiple 2-action GLAs can be combined to allow learners to build a piecewise linear approximation of regions in the space of context vectors. In general, we can use systems which are composed of feedforward structured networks of GLA. In these networks, automata on one level use actions of the automata on the previous level as inputs. If the feedforward condition is satisfied, meaning that the input of a LA does not depend on its own output, convergence to local optima can still be established (Phansalkar & Thathachar, 1995).

Figure 1 shows the general agent learning set-up. Each agent receives factored state representation as input. GLA decide action to be performed.

Figure 2 shows typical results for approximations that the agents learn for the parabola.