Multiagent Reinforcement Learning in Markov Games: Asymmetric and Symmetric Approaches

Ville Könönen
Neural Networks Research Centre
Helsinki University of Technology
P.O. Box 5400, FI-02015 HUT, FINLAND
ville.kononen@hut.fi

***Abstract*— Modern computing systems are distributed, large and heterogeneous. Computers, other devices and humans are very tightly connected with each other and therefore it would be more preferable to handle these entities more as agents as stand-alone systems. One goal of the artificial intelligence is to understand interactions between entities, whether they are artificial or natural, and how to make good decisions taking other decision makers also into account. In this research project, these interactions between intelligent and rational agents are modeled with Markov games and the emphasis is on the adaptation and learning in multiagent systems.**

I. Introduction

Reinforcement learning methods have attained lots of attention in recent years. Although these methods and procedures were earlier considered to be too ambitious and to lack a firm foundation, they have been established as practical methods for solving *Markov Decision Processes (MDPs)*. However, the requirement for reinforcement learning methods to work is that the problem domain in which these methods are applied satisfies the Markov property. Basically this means that the next state of a process depends only on the current state, not on the history. In many real-world problems this property is not fully satisfied. However, many reinforcement learning methods can still handle these situations relatively well. Especially, in the case of two or more decision makers in the same system the Markov property does not hold and more advanced methods should be used instead. A powerful tool for handling these highly non-Markov domains is the concept of *Markov Game (MG)*. In this project, we have developed efficient, both tabular and numeric, learning methods for MGs. In addition, the developed methods have been tested with several example applications.

II. MGs

With multiple agents in the system, the fundamental problem of single-agent MDPs is that the approach treats other agents as a part of the static environment and thus ignores the fact that the decisions of the other agents may influence the state ($s \in S$) of the environment.

One possible solution is to use competitive multiagent MDPs, i.e. MGs. In a MG, the process changes its state according to the action choices of all agents and can thus be seen as a multicontroller MDP. In Fig. 1, there is an example of a MG with three states (s_1,s_2,s_3) and two agents. The process changes its state according to probability $P(s_i|s_1, a^1, a^2), i = 2, 3$, where a^1, a^2 are actions selected by the agents 1 and 2.

In single-agent MDPs, it suffices to maximize the utility of the agent in each state. In MGs, however, there are multiple decision makers and more elaborated solution concepts are needed. Game theory provides a reasonable theoretical background for solving this interaction problem. In single-agent learning, the goal is to find the utility maximizing rule (policy) that stipulates what action to select in each state. Analogously, in a multiagent setting the goal is to find equilibrium policies between the learning agents.

III. Practical Learning Methods

We have concentrated on the case where the state transition probabilities and utility values are not known to the learning agents. Instead, the agents observe their environment and learn from these observations. In general, we use the update rule in the following form:

$$\begin{aligned} Q^i_{t+1}(s_t, a^1_t, \ldots, a^N_t) = (1-\alpha_t) Q^i_t(s_t, a^1_t, \ldots, a^N_t) \\ + \alpha_t [r^i_{t+1} + \gamma \text{Val}\{(s_{t+1})\}], \end{aligned} \quad (1)$$

where $Q^i_t(s_t, a^1_t, \ldots, a^N_t)$ is the estimated utility value for agent i at the time instance t when the system is in the state s_t and the agents select actions $a^1_t, \ldots, a^N_t$. r^i_{t+1} is the immediate reward for agent i and γ is the discount factor. Val$\{\cdot\}$ is a function used for evaluating values of the games associated with states. Although the general form of the update rule is the same for all agents in the system, the implementation of the operator Val$\{\cdot\}$ may be different for agents.

In this research project, the following choices for Val$\{\cdot\}$ have been considered:

1) *Stackelberg equilibrium*. In this case, the agents in the system are put in some ordering and all the agents are assumed to know and accept this ordering. The evaluation of Val$\{\cdot\}$ is very fast due to the sequential nature of the decision process.
2) *Correlated equilibrium*. In the correlated equilibrium concept, it is assumed that in addition to agents, there exists also a *mediator*, a human or a machine that randomly recommends pure strategies to the agents. If the distribution where the recommendations are drawn satisfies *strategic incentive constraints*, it is also rational

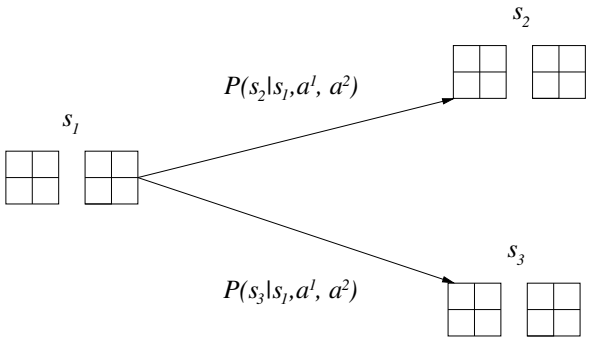

Fig. 1. An example MG with three states.

to obey these recommendations for all the agents. The value Val$\{\cdot\}$ can be evaluated by solving a simple linear program.

3) *Nash equilibrium*. With Nash equilibrium concept, the roles of the agents are symmetric, i.e. all agents make their decisions simultaneously. The resulting equilibrium policy may be stochastic. In general case, the evaluation of Val$\{\cdot\}$ can be computationally demanding; in fact, it is still an open question if there exists computationally efficient methods for determining Nash equilibria of an arbitrary game.
4) *MaxMin solution*. Much of the earlier work in game theory was on two-person zero sum games. In these games, the Nash equilibrium concept reduces to the MaxMin solution that is possible to be determined by using a simple linear program. If this solution concept is applied in general-sum problems, it provides a security level solution: it is a best response against the worst possible action selection of the opponent. If the agent assumes that it faces a zero-sum problem and applies MaxMin solution concept, it can do no worse but possible better against an unknown opponent.
5) *MaxMax solution*. The opposite class of games for zero-sum games is the class of games in which the players share the same utility function. In this case, the problem reduces, in fact, to the decision problem in which there is a meta-agent corresponding all the agents. The actions available to this meta-agent are joint actions of all the agents and the MaxMax solution returns the value corresponding a joint action that maximizes the utility function. The value of the operator Val$\{\cdot\}$ is very fast to evaluate and the resulting equilibrium policy is deterministic.

In multiagent reinforcement learning systems based on MGs, the space and computational requirements grow very quickly with the number of learning agents and the size of the problem instance. Therefore, it is necessary to use function approximators, such as neural networks, to model agents in real-world applications. In this research project, various numeric learning methods are proposed for multiagent learning problem.

IV. Example Applications

The proposed methods are tested with small but non-trivial example problems from different research areas, including artificial robot navigation, simplified soccer game and automated pricing models for intelligent agents. The main properties of the applications are discussed below.

1) *Grid world problem*. The problem is a version of robot navigation problem in which two agents learn to navigate through a grid world without colliding with each other or with other obstacles. The problem is particularly interesting due to its general-sum payoff structure, i.e. the interests of the agents are only partially conflicting.
2) *Simple soccer game*. In this problem, there are three players (agents): one fixed strategy player and two learning agents that constitute a team. The goal of the agents is to find the optimal strategy against the fixed strategy opponent.
3) *Pricing applications*. In pricing applications, there are two or more brokers (agents) that sell identical products and compete with each other on the basis of price. The goal of the agents is to learn to prevent "price wars", i.e. repeated price reductions among the brokers.

References

[1] Ville J. Könönen. Asymmetric multiagent reinforcement learning. In *Proceedings of the 2003 WIC International Conference on Intelligent Agent Technology (IAT-2003)*, pages 336–342, Halifax, Canada, 2003.
[2] Ville J. Könönen. Gradient based method for symmetric and asymmetric multiagent reinforcement learning. In *Proceedings of the Fourth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL-2003)*, pages 68–75, Hong Kong, China, 2003.
[3] Ville J. Könönen. Policy gradient method for multiagent reinforcement learning. In *Proceedings of the second International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS-2003)*, Singapore, 2003. CD-ROM.
[4] Ville J. Könönen. Hybrid model for multiagent reinforcement learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN-2004)*, pages 1793–1798, Budapest, Hungary, 2004.
[5] Ville J. Könönen. Policy gradient method for team Markov games. In *Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL-2004)*, Exeter, UK, 2004. In press.
[6] Ville J. Könönen and Erkki Oja. Asymmetric multiagent reinforcement learning in pricing applications. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN-2004)*, pages 1097–1102, Budapest, Hungary, 2004.