Policy Gradient Method for Team Markov Games

Ville Könönen

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

Abstract. The main aim of this paper is to extend the single-agent policy gradient method for multiagent domains where all agents share the same utility function. We formulate these team problems as Markov games endowed with the asymmetric equilibrium concept and based on this formulation, we provide a direct policy gradient learning method. In addition, we test the proposed method with a small example problem.

1 Introduction

Applying multiagent reinforcement learning to large, realworld applications requires the use of function approximators such as neural networks. The recently proposed numeric methods for multiagent reinforcement learning deal mostly with value function approximation. In this paper, we propose an alternative approach by extending the direct policy gradient method proposed by Sutton et al. in [6] for multiagent domains.

The focus in this work is on team problems, i.e. problems in which all agents share the same utility function. We model the interaction between the agents as a Markov game endowed with the asymmetric equilibrium concept. Due to the sequential nature of the decision making and the shared utility function, the multiagent learning problem reduces to a form that is very close to the single-agent reinforcement learning. We provide a convergent learning rule that solves this learning problem and based on this rule, we formulate a direct policy gradient method. In addition, we test the proposed method with a small example problem.

2 Multiagent Reinforcement Learning

This section is mainly concerned with the basic problem settings and definitions of multiagent reinforcement learning based on Markov games. We start with some preliminary information about Markov games and then proceed to their solution concepts which are essential for the rest of the paper.

2.1 Markov Games

With multiple agents in the environment, the fundamental problem of using single-agent Markov decision processes is that the approach treats the other agents as a part of the environment and thus ignores the fact that the decisions of these agents may influence the state of the environment.

One possible solution to this problem is to use *competitive Markov decision processes*, *Markov games.* In a Markov game, the process changes its state according to the action choices of all agents and it can thus be seen as a multicontroller Markov decision process. Formally, we define a Markov game as follows:

Definition 1. *A* Markov game *(*stochastic game*) is defined as a tuple* $(S, A^1, \ldots, A^N, p, r^1, \ldots, r^N)$*, where* N *is the number of agents,* S *is the set of all states,* A^i *is the set of all actions for each agent* $i \in \{1, N\}$*,* $p : S \times A^1 \times \ldots \times A^N \to \Delta(S)$ *is the state transition function,* $r^i : S \times A^1 \times \ldots \times A^N \to \mathbb{R}$ *is the reward function for the agent* i*.* $\Delta(S)$ *is the set of probability distributions over the set* S*.*

As in the case of single-agent Markov decision processes, we need a *policy* π^i, i.e. a rule stating what to do, given the knowledge of the current state of the environment, for each agent i:

$$\pi^i : S \to A^i, \forall i \in \{1, N\}. \tag{1}$$

In Eq. (1), the policy π^i is assumed to be stationary, i.e. there are no time dependents in the policy. The value for each state-actions tuple is

$$\begin{aligned} Q^i_{\pi^1,\ldots,\pi^N}(s, a^1, \ldots, a^N) &= r^i(s, a^1, \ldots, a^N) \\ &\quad + \gamma \sum_{s'} p(s'|s, a^1, \ldots, a^N) V^i_{\pi^1,\ldots,\pi^N}(s'), \end{aligned} \tag{2}$$

where $r^i(s, a^1, \ldots, a^N)$ is the immediate reward for the agent i when actions $a^1, \ldots,$
a^N are selected in the state s and $V^i_{\pi^1,\ldots,\pi^N}(s)$ is the value of the state for the agent i.

2.2 Equilibria in Markov Games

In multiagent reinforcement learning, it is not sufficient to maximize the expected utilities of individual agents. Instead, our goal is to find an equilibrium policy π^i_* for each agent i. A Nash equilibrium policy is defined as follows:

Definition 2. *If* N *is the number of agents and* Π^i *is the policy space for the agent* i*, the policies* $\pi^1_*, \ldots, \pi^N_*$ *constitute a Nash equilibrium solution of the game if in every state* s *the following inequality holds for all* $\pi^i \in \Pi^i$ *and for all* i*:*

$$V^i_{\pi^1_*,\ldots,\pi^i,\ldots,\pi^N_*}(s) \le V^i_{\pi^1_*,\ldots,\pi^N_*}(s).$$

The first reinforcement learning method utilizing the Nash equilibrium directly was proposed by Hu and Wellman in [1] and [2]. The Nash equilibrium is an appropriate solution concept if the roles of the learning agents are symmetric. If the roles are not symmetric, i.e. some agents decide their actions prior the other agents, the action decision process becomes sequential and the Stackelberg equilibrium should be used instead of Nash equilibrium. In this paper, we study sequential decision problems of two agents in which agent 1 makes its decision prior to agent 2. In this case, the policy function of agent 2 is also the function of the action enforcement of agent 1. Formally, the policy function of agent 2 takes the following form:

$$\pi^2 : S \times A^1 \to A^2. \tag{3}$$

The update rules for agents 1 and 2 are as follows [3]:

$$\begin{aligned} Q^1_{t+1}(s_t, a^1_t, a^2_t) &= (1-\alpha_t) Q^1_t(s_t, a^1_t, a^2_t) \\ &\quad + \alpha_t [r^1_{t+1} + \gamma \max_{b \in A^1} Q^1_t(s_{t+1}, b, Tb)] \end{aligned} \tag{4}$$

$$\begin{aligned} Q^2_{t+1}(s_t, a^1_t, a^2_t) &= (1-\alpha_t) Q^2_t(s_t, a^1_t, a^2_t) \\ &\quad + \alpha_t [r^2_{t+1} + \gamma \max_{b \in A^2} Q^2_t(s_{t+1}, g(s_{t+1}), b)], \end{aligned} \tag{5}$$

where T is an operator conducting the response of agent 2 (assumed to be unique) and $g(s_t)$ is the action enforcement of agent 1 in the state s_t. In team games, the rewards are the same for both agents and therefore one Q-function is sufficient for describing the whole system. In this case, the Stackelberg solution reduces to the MaxMax-solution and the corresponding update rule (the same for both agents) is as follows:

$$\begin{aligned} Q_{t+1}(s_t, a^1_t, a^2_t) &= (1-\alpha_t) Q_t(s_t, a^1_t, a^2_t) \\ &\quad + \alpha_t [r_{t+1} + \gamma \max_{b \in A^1} \max_{c \in A^2} Q_t(s_{t+1}, b, c)]. \end{aligned} \tag{6}$$

The application of the asymmetric learning model to team games also gives a new justification for the use of the MaxMax-operator. However, if there exists an ordering among the agents and both agents know and agree on it, the use of Stackelberg solution solves the possible equilibrium selection problem. A thorough discussion on asymmetric multiagent reinforcement learning can be found in [3].

3 Policy Gradient Methods for Multiagent Domains

In this section, we extend the policy gradient method proposed by Sutton et al. in [6] for multiagent domains. We restrict our attention only to the start state formulation of the problem in which the goal of the agent is to maximize his expected discounted utility starting from the specific state. We start the section by introducing the concept of *joint policy function*. For brevity, all mathematics is presented for the case of two agents.

3.1 Joint Policy Function

We extend the stochastic parametrized policy function to multiagent domains by setting the variables of the function to consist of the state and the joint actions performed by the agents. Formally, this can be expressed as follows:

$$\pi(s, a^1, a^2; \boldsymbol{\theta}) = P(a^1, a^2 | s; \boldsymbol{\theta}), \tag{7}$$

where $\boldsymbol{\theta}$ is an arbitrary parameter vector. The distribution defines the probability of selecting the joint action (a^1, a^2) in the state $s \in S$.

3.2 Policy Gradient

The object function of the policy gradient method is the expected utility in the start state s_0:

$$\rho(s_0, \pi) = V_\pi(s_0) = \sum_{b \in A^1} \sum_{c \in A^2} \pi(s_0, b, c; \boldsymbol{\theta}) Q_\pi(s_0, b, c). \tag{8}$$

By differentiating Eq. (8) with respect to an arbitrary parameter θ we get the following equation (derivation of this equation follows the derivation of the single-agent case in [6]):

$$\frac{\partial \rho}{\partial \theta} = \sum_{s \in S} d_\pi(s) \sum_{b \in A^1} \sum_{c \in A^2} \frac{\partial \pi(s, b, c; \boldsymbol{\theta})}{\partial \theta} Q_\pi(s, b, c), \tag{9}$$

where $d_\pi(s)$ is the discounted weight (probability) of reaching state s starting from the initial state s_0 and following π. It is a real number and therefore we can exclude it from (9) and still get an unbiased estimate of the gradient if the state transitions are sampled by following π.

Now the only remaining step is to find a suitable approximator for the normally unknown Q_π function. Let the function $f(s, a^1, a^2; \boldsymbol{\omega})$ be our approximation for Q_π that is parametrized with the vector $\boldsymbol{\omega}$. Moreover, if we set an additional restriction that the function f fulfills the compatibility property [6]:

$$\frac{\partial f(s, a^1, a^2; \boldsymbol{\omega})}{\partial \omega} = \frac{\partial \ln \pi(s, a^1, a^2; \boldsymbol{\theta})}{\partial \theta} \tag{10}$$

it can be shown that the error due to the use of the function f in place of Q is orthogonal to the gradient of the policy function π. Hence we can replace the function Q with the function f in Eq. (9).

3.3 Value Function Approximation

A natural way to update the parameters $\boldsymbol{\omega}$ is to minimize the following error function at each time step t:

$$E_t = \frac{1}{2}[r_{t+1} + \gamma \max_{b \in A^1} \max_{c \in A^2} f(s_{t+1}, b, c; \boldsymbol{\omega}) - f(s_t, a_t^1, a_t^2; \boldsymbol{\omega})]^2. \tag{11}$$

In this paper, we use the Gibbs distribution as a policy function, i.e.:

$$\pi(s, a^1, a^2, \boldsymbol{\theta}) = \frac{e^{\boldsymbol{\theta}^T \phi(s,a^1,a^2)}}{\sum_{b \in A^1} \sum_{c \in A^2} e^{\boldsymbol{\theta}^T \phi(s,b,c)}}, \tag{12}$$

where $\phi(s, a^1, a^2)$ is a unit vector with the element corresponding to state-actions tuple (s, a^1, a^2) set to one. Correspondingly, the compatible function approximator f takes the linear form:

$$f(s, a^1, a^2, \boldsymbol{\omega}) = \boldsymbol{\omega}^T [\phi(s, a^1, a^2) - \sum_{b \in A^1} \sum_{c \in A^2} \phi(s, b, c)\pi(s, b, c)]. \tag{13}$$

In the above equation, the second term containing two sums is, in fact, the value of the state s and does not depend on the action choices a^1 and a^2. Therefore this term does not change the direction of the policy gradient in Eq. (9) and it is possible to learn the parameters $\boldsymbol{\omega}$ by using the standard learning rule presented in Eq. (6).

3.4 Extensions to General-Sum Problems

There are two ways to extend the above policy gradient method to general-sum problems:

1. Agents update the policy distribution π separately. In this case the goal is to find a policy that maximizes the total (summed) utility of the agents.
2. Agents use the conditional distributions, conditioned with the opponent's action
   choice, to update π. A sketch of the method can be found in [4].

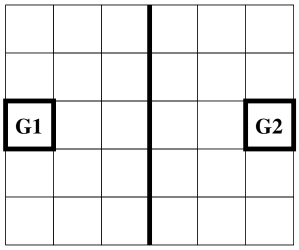
In both cases, the problem is that there is no convergent learning rule for teaching the parameters of the compatible function approximator f and therefore we restrict our attention to team games only.

4 Empirical Tests

In this section, we test the proposed policy gradient method with a simple soccer game that was originally proposed in [5]. In this game, there are three players (agents): one fixed-strategy opponent and two learning agents that constitute a team. The game is played on a 5×6 field illustrated in Fig. 1 (left). In this figure, the cell marked with **G1** is the goal for the learning agents and the cell **G2** for the fixed-strategy agent. The agents are capable of moving to four cardinal directions or staying in the current cell. There can be only one agent in a cell simultaneously. The agent having the ball loses it when colliding with another agent. In addition, each learning agent is capable to pass the ball to its team mate located within 2 cells.

A state consists of the agents' positions and the ball possession information. Initially the fixed strategy agent is located in the left half of the field and the learning agents in the right half of the field. The ball possession is selected randomly. The game ends when the agent possessing the ball reaches a goal cell. The cell **G1** produces the payoff of 1 and **G2** the payoff of -1. When the agent with the ball reaches the goal, players are returned back to random initial positions. After the agents select their actions using the Stackelberg equilibrium concept, the actions are carried out in random order. This induces stochastic state transitions to the Markov game. The fixed strategy player always moves toward the agent possessing the ball and when it gets the ball, it moves directly toward its goal cell.

We taught the model with 50000 games. The learning rate was decayed linearly with time in both value-function and policy function estimation. The discount factor $\gamma = 0.9$ and the maximum length of the game was restricted to 50 moves. During the learning, action selections were sampled from the joint policy function π. Additionally, the model was tested with 1000 games. In Fig. 1 (right), the average number of wins (averaged from 20 test runs) is plotted against the number of training rounds. From this figure, it can be seen that the number of wins increases along the number of training rounds. However, the system learns very fast in the beginning of the learning process, after that the learning continues but is not so dramatic.



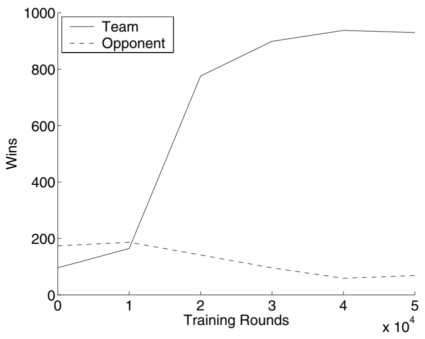


Fig. 1. Left: the game field in the soccer example. The cell **G1** is the goal for the learning agents and the cell **G2** for the fixed strategy opponent. Right: The number of wins plotted against the number of the training rounds.

5 Conclusions

We justified the use of the global maximum value of the state in the value function estimation by using the asymmetric learning model and based on this estimate, we provided the direct policy gradient method. Additionally we tested the policy gradient method with a simple example problem. Although the state-actions space was relatively large, the method learned very fast.

In future research, we will continue the development of the policy gradient method for general-sum games. Additionally, the method will be tested with larger problem instances.

References

1. J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98)*, Madison, WI, 1998. Morgan Kaufmann Publishers.
2. J. Hu and M. P. Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4, 2003.
3. V. J. Könönen. Asymmetric multiagent reinforcement learning. In *Proceedings of the 2003 WIC International Conference on Intelligent Agent Technology (IAT-2003)*, Halifax, Canada, 2003. IEEE Press.
4. V. J. Könönen. Policy gradient method for multiagent reinforcement learning. In *Proceedings of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003)*, Singapore, 2003.
5. L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling. Learning to cooperate via policy-search. In *Proceedings of the Sixteenth Conference on Uncertainty in Artifical Intelligence (UAI-2002)*, Stanford, CA, 2000. Morgan Kaufmann Publishers.
6. R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, Cambridge, MA, 2000. MIT Press.