Reinforcement Learning Approach to Agents' Dynamic Positioning in Robot Soccer Simulation Games

Ki-Duk Kwon , In-Cheol Kim

Department of Computer Science, Kyonggi University, San 94-6, Yiui-Dong, Paldal-Gu, Suwon, 442-760, Korea
(Tel : 82-031-249-9670; Fax : 82-031-249-9673 ; E-mail: kdkwon@hanmail.net*, kic@kyonggi.ac.kr)

Abstract: The robot soccer simulation game is a dynamic multi-agent environment. In this paper we suggest a new reinforcement learning approach to each agent's dynamic positioning in such dynamic environment. Reinforcement learning is the machine learning in which an agent learns from indirect, delayed reward an optimal policy to choose sequences of actions that produce the greatest cumulative reward. Therefore the reinforcement learning is different from supervised learning in the sense that there is no presentation of input-output pairs as training examples. Furthermore, model-free reinforcement learning algorithms like Q-learning do not require defining or learning any models of the surrounding environment. Nevertheless it can learn the optimal policy if the agent can visit every state-action pair infinitely. However, the biggest problem of monolithic reinforcement learning is that its straightforward applications do not successfully scale up to more complex environments due to the intractable large space of states. In order to address this problem, we suggest *Adaptive Mediation-based Modular Q-Learning(AMMQL)* as an improvement of the existing Modular Q-Learning(MQL). While simple modular Q-learning combines the results from each learning module in a fixed way, AMMQL combines them in a more flexible way by assigning different weight to each module according to its contribution to rewards. Therefore in addition to resolving the problem of large state space effectively, AMMQL can show higher adaptability to environmental changes than pure MQL. This paper introduces the concept of AMMQL and presents details of its application into dynamic positioning of robot soccer agents.

Keywords: Robot Soccer Simulation, Multi-agent, Reinforcement Learning, Q-Learning

1. Introduction

Reinforcement learning is the machine learning in which an agent learns from indirect, delayed reward an optimal policy to choose sequences of actions that produce the greatest cumulative reward. Therefore the reinforcement learning is different from supervised learning in the sense that there is no presentation of input-output pairs as training examples[2]. Furthermore, model-free reinforcement learning algorithms like Q-learning do not require defining or learning any models of the surrounding environment. Nevertheless it can learn the optimal policy if the agent can visit every state-action pair infinitely. However, the biggest problem of monolithic reinforcement learning is that its straightforward applications do not successfully scale up to more complex environments due to the intractable large space of states.

One of the most effective solutions for the large state space problem is *Modular Q-Learning(MQL)*, which was suggested by Whitehead[6]. However due to use of both the fixed modules and the fixed mediation strategy, modular Q-learning suffers from difficulty in adapting agent's behavior effectively to drastically changing environment. In order to address this problem, we suggest *Adaptive Mediation-based Modular Q-Learning(AMMQL)* as an improvement of the modular Q-learning. AMMQL combines the results from each learning module in a more flexible way by assigning different weight to each module according to its contribution to rewards. Therefore in addition to resolving the problem of large state space effectively, AMMQL can show higher adaptability to environmental changes than pure MQL. This paper introduces the concept of AMMQL and presents details of its application into dynamic positioning of robot soccer agents.

2. Robot Soccer Simulation

Robot soccer game, especially RoboCup (The World Cup Robotic Soccer) simulation game, is a good test-bed for multi-agent learning techniques including reinforcement

learning. In the RoboCup simulation environment, software agents play a game in a virtual field provided by a soccer server. In other words, each agent receives its perceptual information and executes its actions through the soccer server. Generally this environment has the following difficulties : (1) multi-agent : each team consists of 11 agents, and each agent is required to cooperate with teammates and to compete with the opponents., (2) incomplete perception : it is not possible for each agent to have perfect and complete perception of the state of the environment, (3) real-time : a match is uninterrupted, and each agent should decide its actions in a short period.

When reinforcement learning is applied to this environment, the following difficulties should be overcome : (1) dynamic changes, (2) large state space, (3) distribution of rewards among teammates, (4) partially observable Markov decision process(POMDP), (5) trade-off between exploration and exploitation, etc.

There are several researches applying the reinforcement learning in robot soccer domain. Each agent in CMUnited-99 has a 3-layered learning architecture. At the lowest layer basic skills like ball-interception are learned, but at the highest layer more abstract behaviors like ball-passing are learned. Reinforcement learning is used to learn to whom the ball should be passed. With Andhill team agent[1], reinforcement learning is used to refine its position when it does not have ball. Andhill agents adopt the SGA algorithm, which is a variant of Q-learning method applicable to POMDP. The algorithm addressed some part of the large state space problem by combining Q-learning and the neural network learning, which is one of the inductive learning methods. In NaroSot team agent[3], which is developed in KAIST, modular Q-learning is employed to improve cooperation among the teammates so as to carry out zone defense strategies.

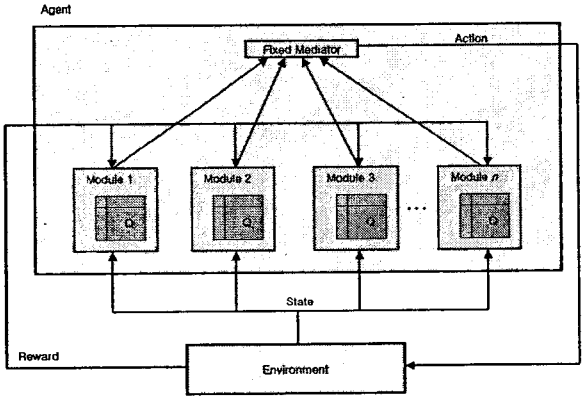

Fig.1 MQL Architecture

3. Reinforcement Learning

Reinforcement learning is the machine learning in which an agent learns from indirect, delayed reward an optimal policy to choose sequences of actions that produce the greatest cumulative reward. Therefore the reinforcement learning is different from supervised learning in the sense that there is no presentation of input-output pairs as training examples. Furthermore, model-free reinforcement learning algorithms like Q-learning do not require defining or learning any models of the surrounding environment. Nevertheless it can learn the optimal policy if the agent can visit every state-action pair infinitely. The updating rule for general Q-learning is as the following equation (1).

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

In the equation (1), s is the current state, a is an action, and R is the reward value. However, the biggest problem of monolithic reinforcement learning is that its straightforward applications do not successfully scale up to more complex environments due to the intractable large space of states.

In order to overcome this problem, various approaches have been considered through previous research. Model-based reinforcement learning algorithms were suggested to increase the speed of Q convergence by computing some more Q-updating calculations based on a certain model of the environment. Generalization techniques were also developed in order to reduce the state space. The generalization techniques allow compact storage of learned information and transfer of knowledge between similar states and actions. But one of the most effective solutions for the large state space problem is *Modular Q-Learning(MQL)*, which was suggested by Whitehead. A typical modular Q-learning architecture is shown in Fig. 1. It contains a number of modules, each of which has its own subgoal and decides its action policy through its own Q-learning. A mediator module is used to mediate global action with the greatest mass(GM) strategy[5].

$$a^* \leftarrow \arg\max_{a \in A} \sum_{i=1}^{n} Q_i(s,a) \quad (2)$$

The equation (2) represents that the Q-values of n modules are summed up for all possible actions, and the action that has the largest sum is selected. This architecture makes the Q-table smaller and improves learning time for problems with multiple goals. However due to use of both the fixed modules and the fixed mediation strategy, modular Q-learning suffers from difficulty in adapting agent's behavior effectively to drastically changing environment. On the other hand, *Automatic Modular Q-Learning(AMQL)*

proposed by Kohri[4] enables agents to obtain a suitable set of modules by themselves through a module selection process like a genetic algorithm. The AMQL architecture has three phases when it obtains the set of modules. Firstly, modules in agents select elements of the environment, and make a Q-table. While agents learn, the AMQL mechanism provides fitness estimates for modules that contribute a reward acquisition action. At selection time, modules are evaluated and selected according to fitness. After repeating these three phases, agents obtain a suitable set of modules. Through reconstructing suitable modules, the AMQL architecture can provide the learning agent higher adaptability to dynamic environment. The AMQL, however, requires a lot of computational time and cost to learn to find suitable modules automatically. Thus it is not effective in short-time environments like robot soccer simulation game.

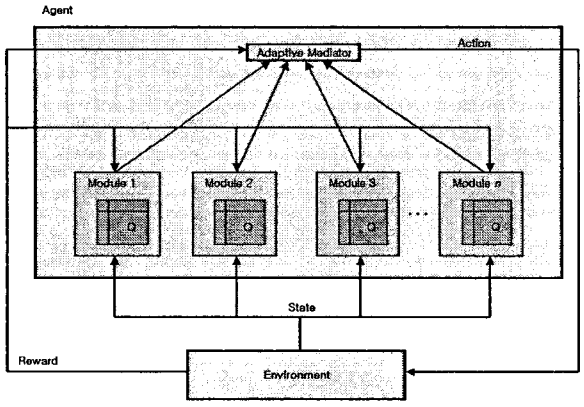

Fig. 2 AMMQL Architecture

4. Adaptive Mediation-based Modular Q-Learning

We suggest a modular approach similar to MQL and AMQL in order to address the large state space problem of monolithic Q-learning. However, unlike AMQL in which agents constructs its own modules for themselves, this new approach uses the fixed modules well designed by the user. Instead it adopts an adaptive mediation strategy, and can adapt the agent's behavior flexibly to radical environmental changes. Therefore this new Q-learning architecture called *Adaptive Mediation-based Modular Q-Learning(AMMQL)* provides higher adaptability than pure MQL by consuming a reasonable amount of additional effort. Furthermore, the AMMQL is more effective than AMQL in short-time environments since it does not require long-time learning for constructing modules.

Fig. 2 shows the architecture of AMMQL consisting of n independent modules. Reward/reinforcement signal from environment is given not only to every Q-learning module but also to the mediation module. The mediation module decides a global action a^* by linearly combining Q-values of each module according to its corresponding weight ω_i. In the equation (3), the sum of all weights is 1, that is, $\omega_1+\omega_2+\ldots+\omega_n=1$.

$$a^* = \arg\max_{a\in A}(\omega_1 Q_1(s_{1k},a)+\ldots+\omega_n Q_n(s_{nk},a)) = \arg\max_{a\in A}\sum_{i=1}^{n}\omega_i Q(s_{ik},a) \quad (3)$$

After executing the selected action, the mediation module adjusts the weight of each module based upon its degree of contribution to the resulting reward. Thus AMMQL takes a double-structured learning architecture, in which the mediation module also learns from reward.

Let R be the reward value provided from environment as a result of executing the selected action a^* at the state s_k. And let r_i be the Q-ratio of the selected action a^*, that is, the proportion of the Q-value of the action a^* to the sum of Q-values of all possible actions at the state s_{ik} in each module M_i as the following equation (4).

$$r_i = \frac{Q_i(s_{ik},a^*)}{\sum_{a_m\in A} Q_i(s_{ik},a_m)} \quad (4)$$

The temporary weight ω_i' of each module M_i is then calculated as the equation (5). The coefficient β in this equation represents a learning rate.

$$\omega_i' \leftarrow \omega_i + \beta \cdot r_i \cdot R \quad (5)$$

And then the new weight ω_i of each module M_i is computed as the equation (6) through normalization process.

$$\omega_i = \frac{\omega_i'}{\omega_1'+\omega_2'+\ldots+\omega_n'} \quad (6)$$

By repeating this weight-adjusting procedure, the AMMQL can adapt the agent's behavior to environmental change without lots of additional cost.

5. Implementation

We apply the AMMQL to agent's dynamic positioning in the RoboCup simulation game environment. What the agent should do when it is far away to the ball is one of the most

difficult tactics to plan. In real world soccer games, a player moves around in the field incessantly because its optimal position is changed dynamically. It must be applicable to the robot soccer simulation agents. In this paper we focus on the dynamic positioning for the purpose of attacking in cooperation with the teammate holding ball. Fig. 3 shows an example of such dynamic positioning.

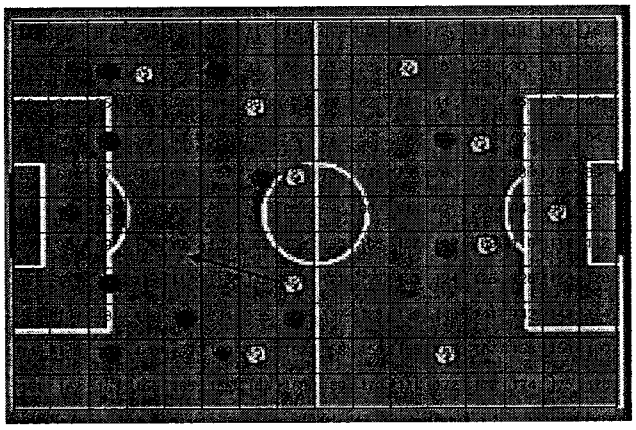

Fig. 3 Example of Dynamic Positioning

We must consider the following environmental elements for deciding the position of an agent :

(1) The relative position of the closest teammate from the ball.

(2) The relative position of the closest opponent from the ball.

(3) The relative position of the agent from the ball.

(4) The relative position of the closest teammate from the destination.

(5) The relative position of the closest opponent from the destination.

(6) The relative position of the agent from the destination.

Suppose the field to be represented by a 16×11 grid. Then the size of state space S for monolithic Q-Learning is $(16 \times 11)^6$=29721861554176. In this paper, we decompose the learning problem into six modules so that each module deals with only one environmental element. So the size of state space of each module is only (16x11)=176. If each possible positioning action taken by an agent can represented as a destination cell within (16 x 11) grid, the size of action space A is (16 x 11)=176. Reward value R is calculated as the equations (7-1) and (7-2), considering the change of the location of the ball. When the ball is inside of the field, the reward R is computed as the equation (7-1). On the other hand, when the ball is outside of the field, the reward R is computed as the equation (7-2).

$$R = \frac{R_0}{1 + (\phi - 1) * t_0 / t_{\lim}} \tag{7-1}$$

$$R = \begin{cases} \phi * \dfrac{x_{avg} - x_t}{x_{og} - x_t} & \text{if } x_{avg} > x_t \\ -\phi * \dfrac{x_t - x_{avg}}{x_t - x_{lg}} & \text{if } x_{avg} < x_t \end{cases} \tag{7-2}$$

The learning rate β for individual Q-learning of each module is 0.1, the discount factor γ is 0.9, and initial Q-values are all 0.1. The learning rate α of the mediation module is 0.1, and the initial weights are all $\omega_i = 1/6 \approx 0.16$.

6. Conclusions

In this paper, we suggested a new modular Q-learning method called AMMQL, and then applied it to dynamic positioning of each agent in robot soccer simulation environment. This learning method provides higher adaptability to environmental changes as well as it addresses the large state space problem of monolithic Q-learning method.

7. References

[1] T. Andou, "Refinement of soccer agents positions using reinforcement learning." Robocup-97: Robot Soccer World Cup I, pp. 373-388, 1998.

[2] L.P. Kaebling, M.L. Littman., A.W. Moore, "Reinforcement learning: A Survey", Journal of AI Research, Vol. 4, pp.237-285, 1996.

[3] J.H. Kim, "Modular Q-learning based multi-agent cooperation for robot soccer", Robotics and Autonomous Systems, Vol.35, pp.109-122, 2001.

[4] T. Kohri et al., "An Adaptive Architecture for Modular Q-Learning", Proceedings of IJCAI-97, 1997.

[5] N. Ono and K. Fukumoto, "Multi-agent Reinforcement Learning: A Modular Approach", Proceedings of ICMAS-96, pp.252-258, 1996.

[6] S. Whitehead, J. Karlsson, and J. Tenenberg, "Learning Multiple Goal Behavior via Task Decomposition and Dynamic Policy Merging", Robot Learning, Kiluwer Academic Press, 1993.