Artif Intell Rev (2012) 38:193–211
DOI 10.1007/s10462-011-9244-8

A multi-agent reinforcement learning approach to robot soccer

Yong Duan · Bao Xia Cui · Xin He Xu

Published online: 21 June 2011

Abstract In this paper, a multi-agent reinforcement learning method based on action prediction of other agent is proposed. In a multi-agent system, action selection of the learning agent is unavoidably impacted by other agents' actions. Therefore, joint-state and joint-action are involved in the multi-agent reinforcement learning system. A novel agent action prediction method based on the probabilistic neural network (PNN) is proposed. PNN is used to predict the actions of other agents. Furthermore, the sharing policy mechanism is used to exchange the learning policy of multiple agents, the aim of which is to speed up the learning. Finally, the application of presented method to robot soccer is studied. Through learning, robot players can master the mapping policy from the state information to the action space. Moreover, multiple robots coordination and cooperation are well realized.

Keywords Multi-agent system · Reinforcement learning · Probabilistic neural network · Robot soccer

1 Introduction

Robot soccer is a good platform for the multi-agent system (MAS) domain research (Kim and Vadakepat 2000; Baghaei and Agah 2007; Reis et al. 2001). In this platform, the subjects of artificial intelligence and robotics have been researched and attention widely.

The decision-making system plays a very important role in robot soccer, and its efficiency and performance will be the key to success in match. In general, the decision-making strategy is designed based on expertise and experience. However, the robot soccer game is a dynamic, uncertain and complex MAS. The robot players have to cooperate with the teammates and compete with the opponents. So the designing method of the decision-making system based

Y. Duan (✉) · B. X. Cui
Shenyang University of Technology, 110870 Shenyang, China
e-mail: duanyong0607@126.com

X. H. Xu
Northeastern University, 11004 Shenyang, China

on expert knowledge and experience, is not completely applicable to all the situations of the robot soccer game. Reinforcement learning (RL) is an effective machine learning algorithm. RL can make the robot players acquire decision-making and inference ability through trial-and-error interaction with the dynamic environment. Consequently, RL is used to design the decision-making strategy of robot soccer will greatly improve flexibility and adaptability of the robot soccer team.

The research of MAS issues focuses on the mutual cooperation and restriction of multiple agents. The action selection strategy of each agent needs to consider other agents' actions. Furthermore, the transformation of the environment state also depends on the performance of all agents. In RL, all state variables of the learning agents compose the join-state. All implementing actions of each learning agent constitute joint-action. So the optimum policy of multi-agent learning represents mapping from the joint-state to joint-action. For complex MAS like robot soccer, according to basic RL methods, it is difficult to obtain a perfect learning result. RL theory and algorithms based on MAS have become research hot spots (Stone and Veloso 2000; Yang and Gu 2004; Taniguchi et al. 2007; Liu and Zeng 2006; Zhong 2003). The followings are the most import studied achievements: the theoretical framework based on Bimatrix Games and Stochastic Games (Yang and Gu 2004), the multi-agent system reinforcement learning (MAS-RL) algorithm based on zero-sum games are proposed by Littman (Littman 1994). Zhong presented the group RL algorithms based on action and state prediction (Zhong 2003). Subsequently, Hu and Wellman extended the presented algorithms to general-sum games and developed a Nash-Q learning algorithm for multi-agent RL (Weinberg and Rosenschein 2004).

In this paper, a MAS-RL algorithm based on the prediction of agent action is proposed. By this method, the probabilistic neural network (PNN) is used to predict the agent selected action and to get the joint-action of all the learning agents. At the same time, the policy shared mechanism is introduced into the MAS-RL process. In order to speed up learning, the optimized policies are mutually exchanged in the learning process. Finally, the application of the proposed MAS-RL method in robot soccer is discussed. By the presented algorithm, the robot players can master the decision-making policy and cooperative ability.

The remainder of this paper is organized as follows: Sect. 2 presents the MAS-RL method based on action prediction. PNN regards to the prediction element of MAS-RL and is used to predict the actions of other agents. Furthermore, using fuzzy inference system (FIS) to implement the MAS-RL algorithm in detail is described. Section 3 describes the action selection of robot soccer based on the MAS-RL algorithm. This section also introduces the policy sharing mechanism and the reward assignment strategy of our MAS-RL. In Sect. 4, we describe a number of experiments illustrating this approach. The last section concludes this paper by briefly summarizing the main results.

2 Multi-agent RL based on action prediction

In Markov decision process (MDP), the agent is able to perceive the state set $S = \{s_i | s_i \in S\}$ of environment, and there is the corresponding action set $A = \{a_i | a_i \in A\}$. At the time step t, agent senses the current state s_t and selects an action a_t. Through implementing the action, the agent can receive the feedback reward r_t and transfer to the new state s_{t+1}. The aim of RL is to achieve an optimum control scheme $\pi : S \rightarrow A$, which makes the sequence $<$ state, action $>$ maximize the cumulative rewards.

In MAS, all agents are in the learning state. For the learning agents, the current environment state can be changed through performing actions themselves. However, other agents

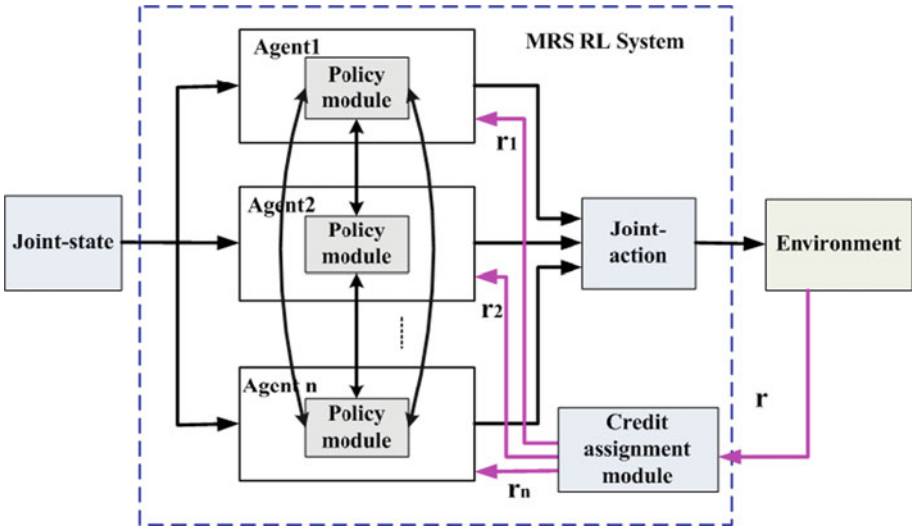

Fig. 1 MAS-RL structure. The joint-state is comprised of all the state information of the learning agents. The joint-state is used to learn policy of each agent. The output actions of each agent make up of joint-action which acts on environment and obtains reward from the environment

are also in the learning phase. The actions of other agents also change the environment state. Hence, the following environment state is unpredictable for the learning agents. The state is related with the actions of all agents. Furthermore, in MAS, cooperation and competition exist among the agents. So the action selection strategy of a learning agent is inevitably influenced and restricted by other agents' actions. In addition, the reinforcement signal of MAS-RL is based on multi-agent, rather than individual agents. For example, in robot soccer, the reward from a goal is the result that all robot players implement actions together. In MAS, the state and action of individual agent are extended to the joint-state and joint-action ($\vec{s}$, $\vec{a}$) of all agents (Weinberg and Rosenschein 2004).

The structure of MAS-RL is shown as Fig. 1. In MAS-RL process, each agent has a respective policy module. The policy describes the mapping method from the state space to the action space of RL. In MAS, multiple agents' behaviors interact with each other. The actions of one agent are influenced inevitably by the states and actions of other agents. So there is interaction among policy modules of agents. The strategies (the mapping from state space to action space) of policy module of one agent also are influenced by other agents'. The joint-action of all agents acts on the environment and changes the environment state. Then the reward is obtained from environment. In order to perform learning of each agent, the reward will be assigned each individual agent according to the credit assignment strategy.

2.1 Multi-agent reinforcement learning

Q-learning is an important algorithm of RL (Sutton and Barto 1998). In Q-learning, the idea is to directly optimize Q-function, which can be calculated recursively. The function of $Q(\mathbf{s}, a)$ is to denote the evaluation of the state-action pair. Q-function is defined as follows (Sutton and Barto 1998):

$$\hat{Q}(\mathbf{s}_t, a_t) = Q(\mathbf{s}_t, a_t) + \alpha_t \cdot [r_t + \gamma \cdot \max_a Q(\mathbf{s}_{t+1}, a_{t+1}) - Q(\mathbf{s}_t, a_t)]. \quad (1)$$

Where $< \mathbf{s}_t, a_t >$ is a state-action pair at time step t. a_t denotes one of the possible selected actions. s_{t+1} expresses the new state when the action a_t has been implemented. r_t represents the reinforcement signal (reward). The value of α_t is a learning rate. γ denotes the discount factor.

Equation 1 updates the current Q-function based on the evaluation value of the next state, which is noted individual agent Q-learning. When the convergence of the Q-function is achieved, the optimal policy can be confirmed. In multi-agent Q-learning, the Q-values depend on the action of the other agents. Agent adopts Nash strategy to update its Q-values (Zhong 2003). $Q^p = (Q^p(\mathbf{s}^1), \ldots, Q^p(\mathbf{s}^m))$ is the Q-table of agent p. $Q^k(\mathbf{s}^j)$ is the Q-value under state $\mathbf{s}^j$. $\overrightarrow{a}_t = \{a^1, \ldots, a^n\}$ describes the joint-action of all agents. Based on the above analysis, the Q-function of MAS-RL is determined by all agents' actions. The update rule of Q-function for agent p is denoted as follows (Zhong 2003; Littman 1994; Weinberg and Rosenschein 2004):

$$Q_t^p(\mathbf{s}_t^p, \overrightarrow{a}_t) = (1-\alpha_t) Q_{t-1}^p(\mathbf{s}_t^p, \overrightarrow{a}_t) + \alpha_t \left[r_t^p + \beta \pi^1(\overrightarrow{s}_{t+1}) \cdots \pi^n(\overrightarrow{s}_{t+1}) Q_{t-1}^p(\overrightarrow{s}_{t+1}) \right] \quad (2)$$

$$\pi^1(\overrightarrow{s}_{t+1}) \cdots \pi^n(\overrightarrow{s}_{t+1}) Q_t^p(\overrightarrow{s}_{t+1}) = \sum_{a^1 \in A} \sum_{a^2 \in A} \cdots \sum_{a^n \in A} P_t^1(\overrightarrow{s}_{t+1}, a^1) \cdots P_t^n(\overrightarrow{s}_{t+1}, a^n) Q_{t-1}^p(\mathbf{s}_{t-1}^p, a^1, \ldots, a^n). \quad (3)$$

Where, $\mathbf{s}_t^p$ is the state variable of agent p. $\overrightarrow{s}_{t+1}$ is the joint-state at the next step. The policy of agent p is expressed by the probability distribution π^p of its action set A^p. $P_t^p(\overrightarrow{s}_{t+1}, a^p)$ denotes the probability of selecting action a^p.

The important difference between the above algorithm and the basic Q-learning algorithm is that the Q-function is redefined based on the learning agent's state and joint-action. Therefore, the key problem of MAS-RL is how to obtain the joint-state and joint-action. During the RL process, each learning individual can sense its state by itself. So with a good communication system, it will be easy to realize the share of joint-state. However, because all agents take actions at the same time, no agent can know what action the other agents will take. So the joint-action cannot be obtained. Usually, the behaviors of agents are not arbitrary. The action strategy can be considered based on some definite probability distribution. So our MAS-RL system is composed of the action predict element and RL element (see Fig. 2). In this MAS-RL, each learning agent has the individual RL element and common action predict element. Where, the action predict element is used to predict other agents' selected action based on the probability method. Furthermore, it supplies the information of the selected actions and their predicted probability of other agents to the RL element. The RL element sends the learning samples to the action predict element to update the predict model.

2.2 Action prediction based on probabilistic neural network

Probabilistic Neural Network (PNN) was proposed by Specht (Specht 1990a,b). It is a kind of neural network model used for classification. In this model the Bayes decision-making analysis of Parzen window estimate is performed by the neural network structure. PNN is developed from the Bayes decision strategy of multiple variables pattern classification. PNN can perform Bayes decision strategy and nonparametric estimators of the probability density function. It is used to classify patterns based on stored samples. PNN has the following merits: (1) PNN performs the forward calculation, but it does not need to perform the error back-propagation calculation. So, it speeds up the learning and is not prone to converging to

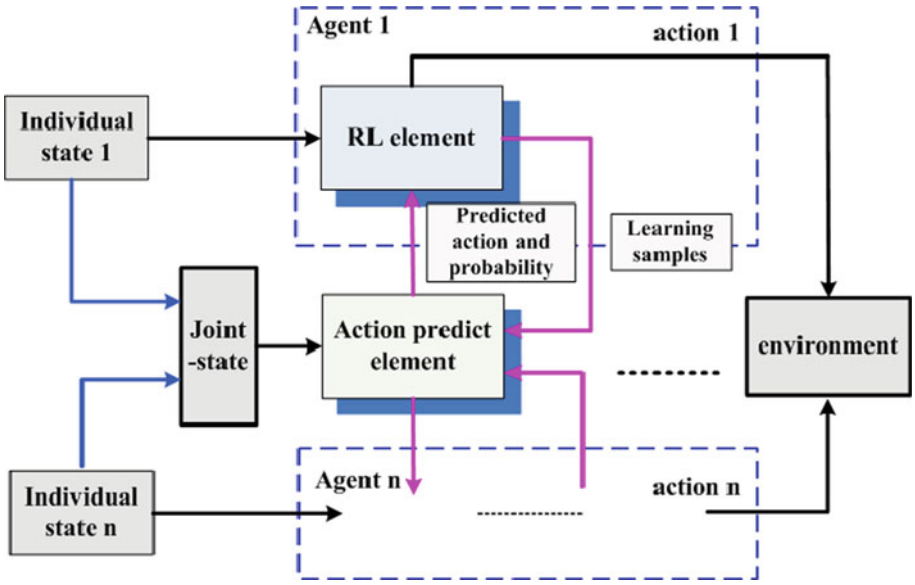

Fig. 2 MAS-RL module. All individual states of agent compose joint-state. Each agent holds respective RL element and uses joint-state to perform learning process. All agents share an action prediction element which is updated by learning samples from the environment. The predicted actions and selected action of agent act on environment

the local minima point. (2) With sufficient training samples, it is guaranteed that PNN can converge to the optimal Bayes classifier. (3) PNN can provide the credit probability based on decision-making. It can also produce the output of Bayes posterior probability.

The network structure of PNN is similar to some forward neural networks. These forward neural networks can be used to build the predictive model (Xie et al. 2007; Lu and Lin 2006). According to the characteristics of PNN, it is used to constitute the action prediction element of multi-agent RL system to predict the actions of other agents. The method of agent action prediction with PNN is that the joint-state $\vec{s} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_i, \ldots, \mathbf{s}_M\}$ of agents regards as the input vector of PNN network and the candidate actions are the output decision categories $\theta_A, \ldots, \theta_K, \ldots, \theta_L$. Then selected action probability can be obtained through PNN network reasoning. The action with maximal probability is the predicted result and serves for selected action of other agent.

PNN is a four-layer networks that can be used to classify patterns. Figure 3 shows a neural network structure for classification of input patterns $\vec{s}$ into multiple categories. The layers are input layer, pattern layer, accumulated layer and output layer (Specht 1990b). PNN respond to an input pattern by processing the input data from one layer to the next with no feedback paths. These types of networks learn pattern statistics from a training set. The input layer directly passes the input $\vec{s}$ to each unit of the pattern layer. The input of each node is the state $\mathbf{s}_i$. $\mathbf{w}_j$ is the weight vector that connected the input layer and pattern layer. In pattern layer, comparison with the input samples and training samples is implemented. There, $\mathbf{w}_j$ is the training sample of the corresponding category. By calculating, the distance vector of the input $\mathbf{s}_i$ and weight $\mathbf{w}_j$ can be obtained. The vector denotes the similarity degree of them. The results are performed the nonlinearity operation and then passed to the accumulated layer.

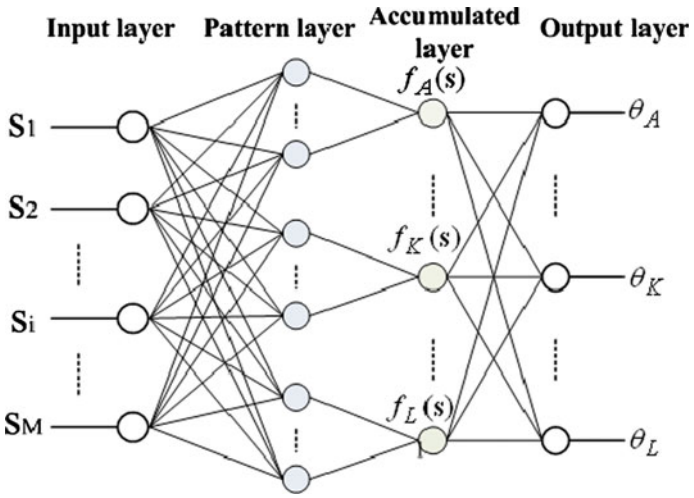

Fig. 3 PNN structure. It is a four layer network which includes input layer, pattern layer, accumulated layer and output layer

The operation is given by:

$$g(\mathbf{z}_j) = \exp\left(-\frac{\mathbf{y}_j}{2\sigma^2}\right) = \exp\left[-\frac{||(\vec{s} - \mathbf{w}_j)||}{2\sigma^2}\right]. \tag{4}$$

Where, $\vec{s}$ and $\mathbf{w}_j$ are normalized to unit length. σ is the smooth coefficient. In Eq. 4, the input and weight of each unit are all vectors. So the Euclidean distance $d_{ij} = |\mathbf{s}_i - w_j|$ is calculated firstly. Then, $\mathbf{y}_j$ is calculated as follows: $\mathbf{y}_j = \mathbf{d}^T \cdot \mathbf{d}$.

The accumulated layer sums the inputs from the pattern layer (Specht 1990b). These inputs correspond to the training samples of the same category. The output of summation node can be given by:

$$f_K\left(\vec{s}\right) = \sum_{j=1}^{n_K} g(\mathbf{z}_j) \tag{5}$$

Where, n_K is the number of training samples belonging to category K.

The output layer can be used to decide the category of the input $\vec{s}$. For the multi-category situation $\theta_A, \theta_B \ldots, \theta_K, \ldots \theta_Q, \ldots$, PNN adapt Bayes decision principle to judge the state of affiliated category $\theta \in \theta_K$, which can be expressed as (Specht 1990b):

$$d\left(\vec{s}\right) \in \theta_K, \quad \text{if } h_K l_K f_K(\vec{s}) > h_Q l_Q f_Q(\vec{s}), \quad Q \neq K \tag{6}$$

Where, $d(\vec{s})$ is Bayes decision of $\vec{s}$. h_K and h_Q respectively express the prior probabilities of $\theta \in \theta_K$ and $\theta \in \theta_Q$. l_K is the loss that have been wrongly classified. $f_K(\vec{s})$ and $f_Q(\vec{s})$ are the probability density functions for categories θ_K and θ_Q. It can be calculated as:

$$f_K(\vec{s}) = \frac{1}{(2\pi)^{M/2}\sigma^M} \cdot \frac{1}{n_K} \sum_{j=1}^{n_K} \exp\left[-\frac{||\vec{s} - \vec{s}_{Kj}||}{2\sigma^2}\right] \tag{7}$$

Where, $\vec{s}_{Kj}$ denotes the jth training sample vector belonging to category θ_K. It is regarded as the weight value $\mathbf{w}_j$ of pattern layer. The samples belonging to category θ_K are updated continuously. So the weight values of the pattern units of PNN are updated accordingly.

PNN can be used to predict the actions which the agents will select. In MAS-RL, the joint-state $\vec{s}$ serve as the input pattern vectors of PNN. The component number of $\vec{s}$ define the unit number of input layer. The action space of agents is regard as the decision categories of PNN. The number of candidate actions defines the unit number in accumulated layer. The problem of agent action prediction equals to classify the input joint-state vector into the corresponding action. During the learning process, the action prediction element and RL element are updated synchronously. Finally, the perfect strategies of action prediction and action selection can be completed. The units' weight values of each pattern denote the training sample vector. In RL process, the learning samples (visited state and selected action) with great reinforcement signal are supplemented continuously to the corresponding group. Moreover, the samples' number belonging to every group is updated synchronously. The samples of a category also can be deleted and replaced. Because the Q-function of state and action is continuously updated, agent maybe obtain greater reward by implemented another action. Then, this sample will move to the units of the category corresponding with another action. According to Eq. 7, we can calculate $f_K(\vec{s})$ of the input joint-state $\vec{s}$ belonging to K class. Finally, for joint-state $\vec{s}$, the condition probability of an agent selecting the action a^K can be expressed as follows:

$$P(a^k|\vec{s}) \propto h_K l_K f_K(\vec{s}) \tag{8}$$

Where, prior probability h_K of selecting action a^k can be estimated from the appearing frequency of action a^K in learning process.

$$h_K = v_K/v \tag{9}$$

Where v_K is the number of samples in the input joint-state set which select the actions belonging to the category a^k.v expresses the total number of training samples. In order to guarantee the probability sum is 1, the condition probability need be normalized

l_K is considered to be related with the reward of the current RL That is,

$$l_K = \begin{cases} \lambda_1 r_t & r_t > 0 \\ \lambda_2/|r_t| & r_t < 0 \end{cases} \tag{10}$$

Equation 10 illustrates that when robots wrongly implement actions, the smaller the reward, the larger the misclassified cost l_K. r_t represents the reward of RL. λ_1 and λ_2 depict scale coefficients.

According to Eq. 8, the condition probability of an agent selecting each action can be calculated. The action with maximal probability is the predicted action of the prediction element. The prediction element can also output the probabilities of each possible action. So the MAS-RL algorithm can be implemented (see Eq. 3).

2.3 Performance method of multi-agent RL

In RL, the classical methods deal with discrete input and output spaces, and the state representation often used is a lookup table. The size of state space of RL grows explosively as the number of state variables increases. This problem is known as the curse of dimensionality. In the case of application, the state and action space of RL are often large, which brings on the search space of training is overly large. Therefore, the agent is difficult to

visit each state-action pair. To cope with this problem, two types of generalization methods are studied widely. One is the function approximation methods. They use approximator to perform the mapping from state space to action space of RL. Sutton proposed CMAC network to implement RL (Sutton and Barto 1998). Touzet (Touzet 1997) also put forward Q-Kohon algorithm, it improved Q-learning method through the use of self-organising map. Other class methods are quantization of state space, which aim is to reduce the complexity of search space. Moore (Moore and Atkeson 1995) proposed Parti-game algorithm, which uses k-d tree partition the state space. Murao and Kitamura put forward an approach noted QLASS, in which the state spaces of RL are constructed as Voronoi diagram (Murao and Kitamura 1997).

For multi-agent RL problem, the policy search space of joint-state and joint-action is far larger than the space of the individual agent RL. In order to deal with it, the fuzzy Q-learning algorithm of Jouffe (Jouffe 1998) is introduced to multi-agent RL. Moreover, the policy share mechanism is proposed. Consequently, the learning speed of multi-agent RL is increased. In fuzzy Q-learning, the state variables or RL are fuzzified. Then FIS is applied to approximate the mapping from the state space to the action space of RL.

According to Eq. 11, the mapping from the state variables to joint-action is expressed by the fuzzy rules. When the antecedent parts of the fuzzy rule are confirmed, all possible action combinations serve as the consequent parts of the fuzzy rule. Through RL the best match consequent part for the antecedent part of the fuzzy rule is confirmed (Jouffe 1998). The fuzzy rule j is shown as follows:

$$\begin{aligned} R_j : \text{If } s^1 \text{ is } F_j \quad & \text{Then } \vec{a} \text{ is } \vec{a}_{j1} \text{ with } q_{j1} \\ & \text{Or } \vec{a} \text{ is } \vec{a}_{jl} \text{ with } q_{jl} \\ & \cdots\cdots \\ & \text{Or } \vec{a} \text{ is } \vec{a}_{jL} \text{ with } q_{jL} \end{aligned} \tag{11}$$

Where, s^1 is the state of agent 1. F_j expresses the fuzzy sets. $\vec{a}_{jl}$ and q_{jl} are the consequent parts of the fuzzy rule. They denote respectively the probable joint-action and corresponding evaluating value of the state s^1. L is the number of the candidate joint-action. The actions $a^2_{jl}, \ldots, a^n_{jl}$ of other agents in joint-action $\vec{a}_{jl} = \{a^1_{jl}, a^2_{jl}, \ldots, a^n_{jl}\}$ can be predicted through PNN mentioned in above section. The action with the most probability is selected. Consequently, the action a^1_{jl} of the learning agent is selected. Firstly, the evaluating value of the state s^1 and the joint-action $\vec{a}$ is calculated as (Zhong 2003):

$$q(s^1, a^1, a^2, \ldots, a^{n-1}) = \sum_{a^n \in A} P^n(\vec{s}, a^n) \cdot q(s^1, a^1, a^2, \ldots, a^n) \tag{12}$$

Through ordinal calculating, we can get:

$$q(s^1, a^1) = \sum_{a^2 \in A} \sum_{a^3 \in A} \cdots \sum_{a^n \in A} P^2(\vec{s}, a^2) P^3(\vec{s}, a^2) \cdots P^n(\vec{s}, a^n) \cdot q(s^1, a^1, a^2, \ldots, a^n) \tag{13}$$

In the Eq. 13, the probability of each action can be obtained from the action prediction element using PNN. Finally, the Boltzman policy is used to select the action a^1 of the learning agent.

$$prob(a^1_k) = \frac{\exp(\beta \cdot q(s^1, a^1_k, \ldots a^n)/T)}{\sum_{a^1_h \in A^1} \exp(\beta \cdot q(s^1, a^1_h, \ldots a^n)/T)} \tag{14}$$

Through the predicted actions of other agents and the selected action of the learning agent from the Eq. 14, the joint-action $\vec{a}_{jl^*}$ can be obtained. Here, the selected joint-action $\vec{a}_{jl^*}$ of each fuzzy rule is fired. And it is regarded as the consequent part of this fuzzy rule in this learning step.

Using the 0-order T-S fuzzy inference system model, we can obtain the output action:

$$a(s) = \sum_{j=1}^{N} \bar{\alpha}_j(s) \times a_{jl^*} \tag{15}$$

The corresponding Q value is:

$$Q(\vec{a}) = \sum_{j=1}^{N} \bar{\alpha}_j(s) \times q_{jl^*} \tag{16}$$

Where, $\bar{\alpha}_j(s)$ express the normalized membership of the fuzzy rule j. The learning agent implements the selected action $a(s)$. Then the state of agent is changed. The reward r_t is received from the environment. The evaluation value q_{jl^*} of the joint-action is updated through the Q-learning algorithm. According to the temporal difference (TD) method, we can get:

$$\Delta Q = \alpha_t \left[r_t^p + \beta \pi^1(\vec{s}_{t+1}) \cdots \pi^n(\vec{s}_{t+1}) Q_{t-1}^p(\vec{s}_{t+1}) - Q_{t-1}^p(s_t^p, \vec{a}_t) \right] \tag{17}$$

On the ground of the action prediction method in Sect. 2.2, the action select probabilities of other agents can be obtained. Then, ΔQ can be calculated through the Eq. 2. The update equation of q value is as follows:

$$q_{jl^*}(t) = q_{jl^*}(t-1) + \Delta Q \cdot \bar{\alpha}_j(s) \cdot e_{jl^*}(t) \tag{18}$$

Where, $e_{jl^*}(t)$ is the eligibility trace. Eligibility traces are used to solve temporal credit assignment problem of reinforcement learning. They simulate the memory parameters associated with the event as eligible for undergoing learning changes (Sutton and Barto 1998). Each state of RL is assigned a trace. When this state is visited in the learning process, its trace is reinforced. Contrarily, the trace is attenuation with the learning process. So eligibility trace can denote that the corresponding state make a contribution to RL. Eligibility trace is defined as follows (Sutton and Barto 1998):

$$e(t) = \begin{cases} \gamma\lambda e(t-1) + 1 & s = s_t \\ \gamma\lambda e(t-1) & otherwise \end{cases} \tag{19}$$

Where, γ is the discount rate. λ expresses the attenuation parameter.

In the reinforcement learning process, the evaluation value q of the joint-action and the action predicted element are updated gradually. After learning, the action of the learning agent in the joint-action with the maximum q value serves as the consequent part. Accordingly, the optimum policy (the mapping from state space to action space) is confirmed.

3 Action selection strategy learning of robot soccer based on multi-agent RL

Based on the proposed MAS-RL method, we study the application to the decision-making strategy learning of robot soccer. Furthermore, the policy sharing mechanism and the reward assignment strategy are introduced. In this paper, we will use MiroSot (Micro-Robot

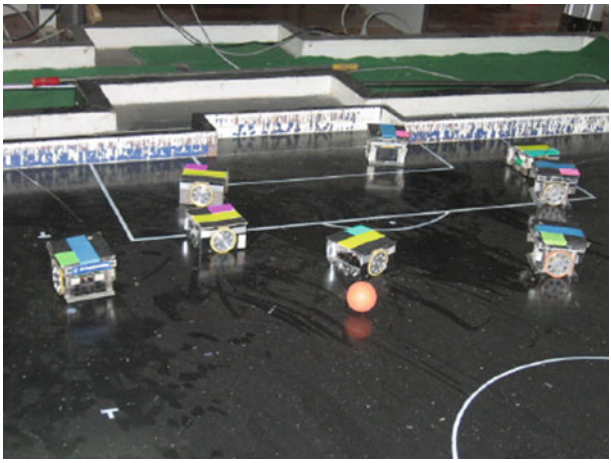

Fig. 4 "NewNeu" robot soccer team

World Cup Soccer Tournament) 5 versus 5 as the research object and experimental platform. We build a real robot soccer team, named NewNeu. The hardwires of robot soccer system include: robot vehicles, image processing system, host computer, and so on. Figure 4 shows the NewNeu robot players on the ground.

NewNeu micro-robot soccer system is composed of five subsystems, which are a host computer control subsystem, the robot vehicle subsystem, the vision subsystem, the decision-making subsystem and the communication subsystem. During game, the host computer subsystem initialized the game states information and performed the general control tasks. The vision subsystem processed the acquired image, and provides the posture information both of the robot players and the ball. The decision-making subsystem implemented reasoning and decision-making based information from vision subsystem. It can decide the behaviors of robot players. The outputs of decision-making subsystem are the velocities of left and right wheel. The communication subsystem performs information exchange by wireless communication.

The robot players of the team compose a MAS. Cooperation and competition behaviors exist in MAS. The action selection (Wu and Lee 2004) is the important part of decision-making system of robot soccer. We will use the presented MAS-RL system to make the robot players learn the action selection strategy. The decision-making system of robot soccer performs the mapping from the state space of match to the action space of robot players. That is, each robot player can select the corresponding action according to the related state information. Obviously, the action selection strategy of each robot player is not only decided by its own state information, but also influenced by the implemented actions of other robot players. MAS-RL of robot soccer is a kind of team learning. Each robot player is an individual learning agent. They learn the common decision-making strategy. All learning agents share the same state space and action space. The different actions are performed based on the different input state variables, and different behaviors are raised. Each learning individual implements MAS-RL algorithm synchronously. They can constitute a distributed multi-agent RL system.

3.1 Definition of state variables, action variables and reinforcement signal function

The decision-making factors of the action selection strategy are described in Fig. 5. d_{B2HG} depicts the distance between the ball and the our goal. d_{B2OG} is the distance from the ball

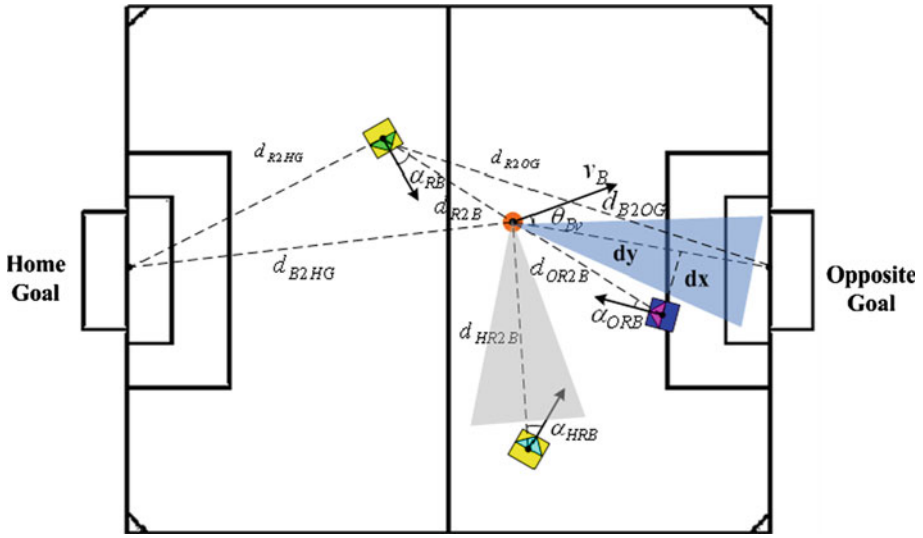

Fig. 5 State information of robot players. The posture relations of robot player and teammate, opponent, ball and goals are described

to the opponent goal. The game ground information (e.g. the locations of home goal and opponent goal, length and width of ground, and so on) are the known parameters. By the image processing of the vision subsystem, the locations of ball and robot players are detected. Hence the distances between the ball and home goal, the ball and opponent goal, and the ball and robots can be computed through the distance between two points—the locations of ball and robot players can be expressed the points. The center points of home goal and opponent describe the locations. d_{R2B} and α_{RB} are the distance and angular between the objective robot and the ball. d_{HR2B} and α_{HRB} are the distance and angular between the ball and the home robot which has the second best posture relative to the ball, after the objective robot. d_{OR2B} and α_{ORB} are the distance and angular between the ball and the opponent robot which has the best posture relative to the ball. θ_{Bv} expresses the angle between the ball's direction of motion and the line connecting the ball and the opponent goal.

The decision-making factors of the action selection strategy include: the ball's position and direction of motion, the competition factor and the cooperation factor, and so on (Weinberg and Rosenschein 2004). Therefore, the state variables of MAS-RL are defined as follows:

$$s_1 = e^{-\tau_1(d_{B2HG}/d_{B2OG})} \tag{20}$$

$$s_2 = \tau_2|\theta_{Bv}|/\pi \tag{21}$$

$$s_3 = \frac{1}{2}\left[e^{-\tau_{31}(d_{R2B}/d_{OR2B})} + e^{-\tau_{32}(\alpha_{RB}/\alpha_{ORB})}\right] \tag{22}$$

$$s_4 = \frac{1}{2}\left[e^{-\tau_{41}(d_{R2B}/d_{HR2B})} + e^{-\tau_{42}(\alpha_{RB}/\alpha_{HRB})}\right] \tag{23}$$

$$s_5 = e^{-\tau_5(d_{R2HG}/d_{R2OG})} \tag{24}$$

$$s_6 = e^{-\tau_6(OFI_R/OFI_G)} \tag{25}$$

In the Eqs. (20) $\sim$ (25), $s_1, s_2, s_3, s_4, s_5, s_6 \in [0, 1]$. $\tau_1\tau_2\tau_{31}$, τ_{32}, τ_{41}, τ_{42}, τ_5 and τ_6 are the positive constants. The state variable s_1 denotes the ball position. From Eq. 20, the value

of s_1 varies between 0 and 1. $s_1 \rightarrow 0$ and $s_1 \rightarrow 1$, respectively, express that the ball is close to opponent goal and home goal. The state variable s_2 describes the mobile direction of ball. It is the normalization result of θ_{Bv}. $s_2 \rightarrow 0$ and $s_2 \rightarrow 1$, respectively, depict that the ball moves toward the opponent goal and the home goal. According to s_1 and s_2, the offensive or defensive situation can be determined roughly. The state variable s_3 shows competition relation of the active robot and the opponent robot. Referring to Eq. 22, s_3 synthesizes the distance factor and the angular factor. $s_3 \rightarrow 1$ indicates that objective robot has a much better chance to control the ball than the opponent robot. $s_3 \rightarrow 0$ expressed the opponent robot will have the better relation with ball than the home robot. The state variable s_4 denotes the cooperation relation of the objective robot and its teammate. $s_4 \rightarrow 1$ expresses robot has an advantage over its teammate. On the other hand, $s_4 \rightarrow 0$ indicates the teammate has better chance to access the ball. The state variable s_5 depicts the position of learning robot in the ground. s_6 can be used to evaluate the kick target. In the Eq. 25, OFI_R and OFI_G determine whether to pass the ball or whether to shoot. The OFI (Obstruction-Free-Indices) (Veloso et al. 1998) reflects how easily an opponent robot could intercept the pass or the shot. The closer the opponent is to the line and the farther it is from the ball, the better chance it has of intercepting the ball (referring to the shadow region in Fig. 5. The OFI (Obstruction-Free-Indices) value is computed by the following algorithm (Veloso et al. 1998): (1) Let $OFI = \kappa$. Initialize the thresholds: mindis and maxdenominator. (2) Compute the distance d_x from the opponent robot to the line connecting the ball and the target (the opponent goal or the teammate). Compute the distance d_y from the ball to the vertical (see Fig. 5). (3) $d_y = \min(d_y, \text{maxdenominator})$. It means that the distance between the opponent robot and the ball is far enough. The opponent robot has enough time to block the ball. So the extra distance is no longer useful. (4) If $d_x \leq \min dis$, then $OFI = d_x/d_y$. (5) Return OFI. $s_6 \rightarrow 0$, easy to shoot. Otherwise, easy to pass. The state variables $s_1 \sim s_6$ serve as the inputs of FIS and are all denoted using three linguistic terms: S (Small), M (Mean) and B (Big). The membership functions of the state variables all use the bell-shaped function (see Fig. 6).

The player's actions are the basic elements of robot soccer match. They are also the foundation and guarantee of the decision-making system. In this paper, the referred candidate actions are the basic skill actions of the robot players. The tactics actions and combination actions can be realized through combining the basic skill actions. According to our experience, the most important skill actions include: **shoot**, **dribble**, **pass**, **intercept**, **clear** and **mark**. Through implementing theses actions, all match tasks of robot soccer can almost be finished. Therefore, the skill actions serve as the action variables of MAS-RL. The selected actions of the learning players can constitute the joint-action. The joint-action and its corresponding evaluation values are the consequent part of the fuzzy rule of learning system.

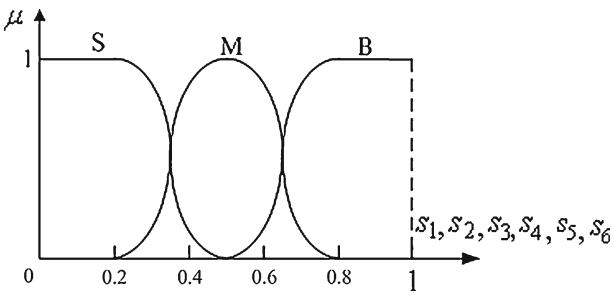

Fig. 6 Membership functions of state variables

Similar to the define method (Wu and Lee 2004), we use $A = \{1, 2, 3, 4, 5, 6\}$ to denote the number of shoot, dribble, pass, intercept, clear and mark. The action variables of RL are the actions' numbers $A_l, l = 1, \ldots, 6$. A_l serves as the consequents of the fuzzy rules. The output action of MAS-RL system is the defuzzification value $\bar{A}$ of the actions' numbers. The proximity degree is defined by: $e_l = |\bar{A} - A_l|$, $1 \leq l \leq 6$. The minimum of the proximity degrees is depicted as: $e_{\min} = \min\{e_1, \ldots, e_6\}$. Hence, the mechanism of the action selection is given by:

$$Act = \begin{cases} Shoot & e_{\min} = e_1 \\ Dribble & e_{\min} = e_2 \\ Pass & e_{\min} = e_3 \\ Intercept & e_{\min} = e_4 \\ Clear & e_{\min} = e_5 \\ Mark & e_{\min} = e_6 \end{cases} \tag{26}$$

Robot soccer is a kind of team behavior. The obtained reward of team isn't dependent on some independent learning individual, but on all learning agents. So the reward (reinforcement signal) of MAS-RL is divided into two parts: one is the team reward function r_t^G, another is an individual reward function r_t^L. The team reward r_t^G is assigned to each learning agent based on the reward assignment module. And the individual reward function r_t^L directly feedback to the corresponding agent. Thus the reward functions are defined as follows:

$$r_t^1 = \begin{cases} 10 & our\ score \\ -10 & opponent\ score \end{cases} \tag{27}$$

$$r_t^2 = \tau \cdot \Delta \bar{d}_{B2G} \tag{28}$$

$$r_t^3 = \begin{cases} +3 & CTR_{T0} < CTR_{T1} \\ -3 & others \end{cases} \tag{29}$$

$$r_t^4 = \begin{cases} 4 & gain\ possession\ of\ the\ ball \\ -4 & lose\ possession\ of\ the\ ball \end{cases} \tag{30}$$

$$r_t^5 = \begin{cases} 2 & perform\ right\ action \\ 0 & otherwise \end{cases} \tag{31}$$

$$r_t^G = \sum_{i=1}^{3} r_t^i \tag{32}$$

$$r_t^L = \sum_{i=4}^{5} r_t^i \tag{33}$$

Obviously, the learning player should obtain the reward when the player can successfully shoot the ball into the goal. However, the score reward is a delayed reward. It is only sparsely given to the player. In order to accelerate the learning speed, we consider other factors to instruct RL. From Eq. 28, $\Delta \bar{d}_{B2G}$ is the average distance difference, that is $\Delta \bar{d}_{B2G} = \bar{d}_{B2HG} - \bar{d}_{B2OG}$. τ is a positive coefficient. If $\Delta \bar{d}_{B2G} > 0$, that is, the ball is located at the opposition area, then a positive reward should be given. $\Delta \bar{d}_{B2G} \leq 0$ expresses that if the ball locates at the home area, then a negative reward should be given. According to Eq. 29, ball possession time severs as the reward. Like the human soccer game, the ball possession time is an important criterion which can be used to evaluate the team's capability. $CTR = CT_H / CT_O$ is ball possession percentage. CT_H and CT_O, respectively, are the

ball possession times of our team and the opponent team respectively. The reward r_t^4 means whether to retain ball possession after the active offense player has executed an action. If our players gain possession of the ball, then a good reward should be given. On the other hand, our players lose the possession of the ball, and then a punishment should be given. r_t^5 is used to judge whether the player can implement the action rightly. The rewards r_t^1, r_t^2 and r_t^3 are the cooperative results of all robot players. They constitute the team reward r_t^G. The rewards r_t^4 and r_t^5 are used to evaluate the validity and effectiveness of the selected actions. They are the results of the individual performing the action. They form the individual reward r_t^L.

3.2 Policy sharing mechanism

The policy sharing mechanism builds the relation among each learning system in distributed multi-agent learning system. It also performs the cooperation of learning agents. And it also can speed up reinforcement learning. The mutual cooperation can be performed through sharing the learned policy in the special learning phase. The robot players can share information through the host computer. Each independent learning system wants to learn the common decision strategy. After completing respective temporary learning, the learning outcomes will incorporate a common FIS. However a frequent exchange policy will go against holding the learning system diversity. Hence, we choose the appropriate frequency of policy sharing through experiment and experience.

The sharing policy method is to modify the q-values of all independent agents. The q-value corresponding to the action l of the learning agent k's fuzzy rule j is defined as q_{jl}^k. The calculation method is:

$$q_{jl}^k = \frac{1}{4} \cdot \sum_{m=1}^{4} q^m(j, l) \tag{34}$$

3.3 Reward assignment strategy

Each learning agent obtains not only the respective reward but also the assignment reward from team reward. The reward comes from all learning agents implement the actions, so it needs to assign to each independent learning system. A reward assignment function is defined as: $\delta_k = w_r + w_a + w_p, k = 1, \ldots, 4$. It can mark the four robot players besides the goalie. Where w_r is the role factor. The active player to be more important than the auxiliary player. If the robot implements the selected action successfully, a bigger w_a is obtained. Otherwise, which robot possesses the ball, then a bigger w_p is given. So the reward r^k of the kth independent learning agent is obtained as follows:

$$r_k^G = \delta_k \cdot r_t^G / \sum_{i=1}^{4} \delta_i \tag{35}$$

The final reward of learning agent is the sum of the assignment reward and the individual reward, $r_k = r_k^G + r_k^L$.

4 Experimental results

To demonstrate the effectiveness of action selection strategy learning, experiments were performed with the robot soccer simulator and the real micro robot soccer players. In order to decrease the learning exhaustion and increase the speed of learning, we firstly implement the

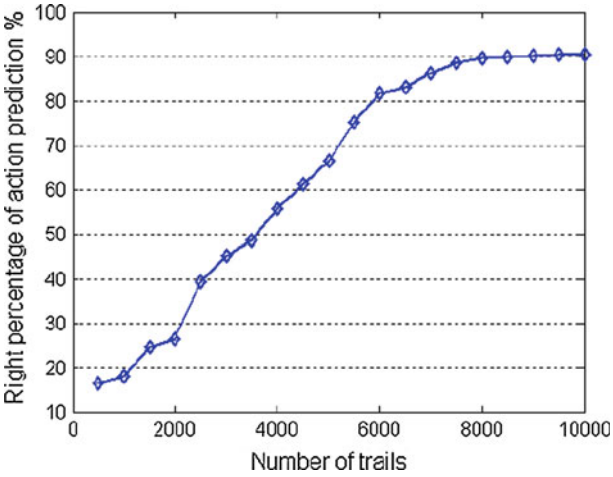

Fig. 7 Correct rate of action prediction

MAS-RL method in the simulation. When the robot player obtains some behavior ability, it further learns the strategy in the real robot soccer.

In the experiment, we choose a weak team as the opponent, because the learning robot team's capability improves gradually. At the early stages of learning, if the opponent team is very powerful, it will be difficult for our team to obtain rewards. Thus the RL algorithm would be unable to perform smoothly and effectively. After implementing the preliminary learning, the opponent ability was improved gradually. Finally, the training opponent was a powerful team. Through our experiments, this learning strategy is effective and can make our system master decision-making ability quickly.

In the experiment, we applied MAS-RL to accomplish the learned tasks. After training, the robot players use learned strategy to select the actions. The number of trails is 10,000 in total. Every 500 trails, we examined the performance of learning. By playing 5 games, we obtain the outcomes of the games. The time of each play is 5 min. The opponent team is our training object. Figure 7 denotes the right action prediction rate of other players in each learning phase. It is shown that the right rate of PNN is increasing gradually. Figure 8 shows the score differences between our scores and the opponent's scores. It is depicted that the offensive and defensive capabilities of our team are both improved by selecting the appropriate actions. Figures 9 and 10 denote respectively the curves of our ball possession percentage and time percentage of ball into opposition area. They can illuminate that comprehensive ability of robot team gradually become competent Fig. 11 depicts the percentage for selecting the right actions. The tested actions include: clear, intercept, shoot and pass. In specified environment, we test whether the robot players can select the right actions under state information. From the curves, we know the percentage of the action selection is increasing. After the training process, we implemented the sufficient experiments with real robot system using the learning results. We provide some experimental results with real robot from exercise games. Now our soccer team has applied the proposed methods and participated in some tournaments. In order to indicate the effectiveness of the presented algorithms, we provide some experiment results with real robot soccer system which include the right rate of prediction and successful rate of action performance from the practical soccer games. Table 1 shows the results of our team playing games with the different opponents.

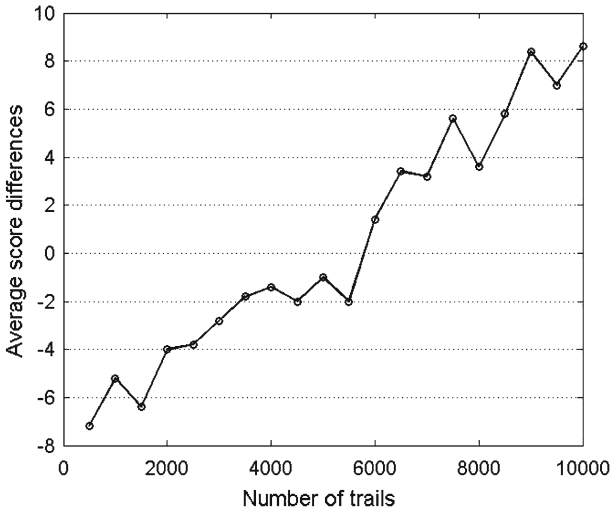

Fig. 8 Score differences of two teams

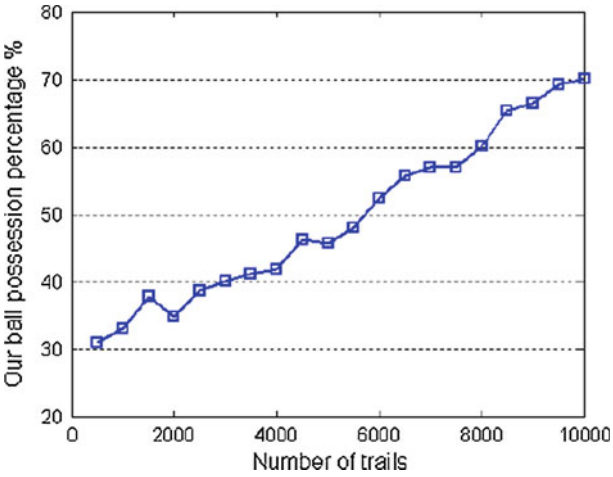

Fig. 9 Ball possession percent of our team

From the above experimental results, it's obvious that the learning speed of decision-making strategy is slower due to the complexity and dynamics of the robot soccer system. So extensive training is needed to build the perfect decision-making system. Moreover, the performance and parameters also influence the learning consequence. Hence improving efficiency and performance of the learning algorithm can enhance the decision-making ability.

Through learning, we find that the learning team shows less flexibility when some special situations occur. The reason is that these situations appear only rarely during the learning and training process. So the robot players cannot learn the perfect ability for dealing with these situations. In order to solve theses problems, we can design some fuzzy rules for the special situations using expert knowledge. And these rules are supplied to the fuzzy rule base

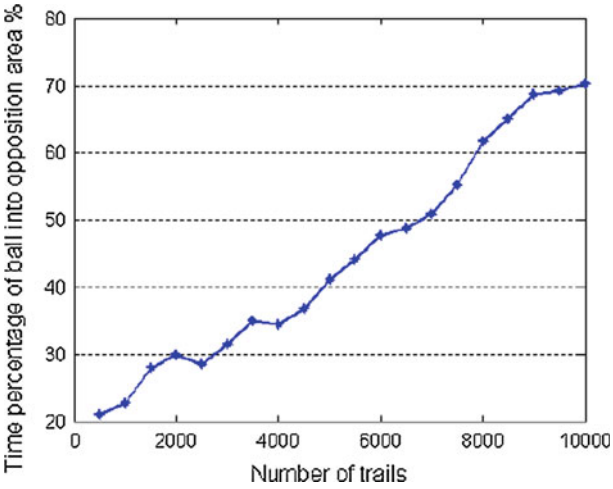

Fig. 10 Percent of ball in opponent area

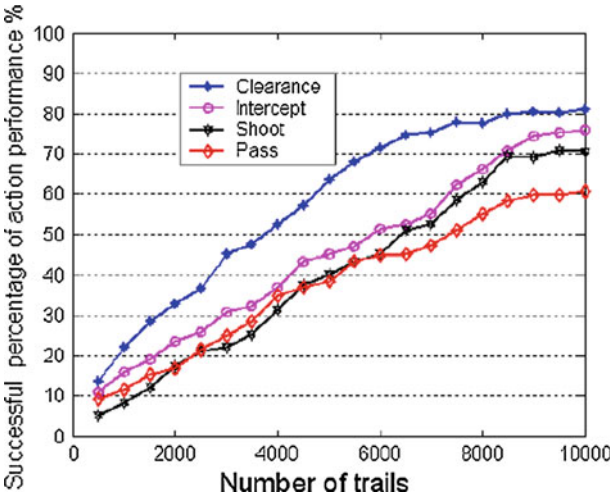

Fig. 11 Success rates of select action

designed by RL. This method can make up the shortage of the learning method and make the designed decision-making system perfect.

5 Conclusions

In this paper, a multi-agent reinforcement learning (MAS-RL) method is studied based on the action prediction and the policy shared mechanism. In a multi-agent team, the selected action of each agent is not only decided by its own state information, but is also unavoidably influenced by the implemented actions of other agents. Hence the PNN is used to constitute the action prediction element and to predict the selected actions of other agents. Further-

Table 1 The experiment results from games with opponents

	Right rate of prediction %	Success rates of action performance %			
		Clearance	Intercept	Shoot	Pass
Opponent 1	86.3	75.6	72.2	66.5	56.4
Opponent 2	90.1	77.1	73.1	65.9	56.9
Opponent 3	84.6	71.9	70.6	62.3	58.7
Opponent 4	85.2	74.4	70.2	64.3	59.2

It shows the right rate of the action prediction and successful rate of action performance

more, in MAS-RL process, the learned policy is exchanged to improve learning speed. The MAS-RL algorithm is performed by FIS. This method effectively decreases the number of dimensions of the state space and realizes the mapping and approximate from the state space to action space.

The proposed MAS-RL method is used to design the decision-making system of robot soccer. Each robot is regarded as a learning individual. All agents learn the common decision-making strategy and form a distributed multi-agent RL system. After learning, we can obtain a group of fuzzy rules, which serve as the fuzzy logic controller of the decision-making strategy of robot soccer system. The rapid operation of controller can meet the real-time requirements of robot soccer. Furthermore, the expert knowledge is easy to be introduced to FIS, so the fuzzy rules obtained by fuzzy RL can be improved and supplemented. The effectiveness of the proposed method is demonstrated through robot soccer experiments. Through learning, the robot players can select the right actions and the decision-making ability can be improved gradually.

The important potential directions for our future research are as follows: firstly, research MAS-RL relational issues such as theory and algorithm of RL, cooperation and coordination of multiple agents, individual goal and team goal, and so on. Secondly, learning tasks of robot soccer are very complex, the convergence speed of MAS-RL is too slow. So how to improve the learning speed is concerned. Thirdly, robot soccer is particularly good domain for studying MAS-RL. The applications of MAS-RL to robot soccer are researched further. The system will be divided into multiple layers and constituted the hierarchical architecture. We would use MAS-RL to perform each decision-making task. Furthermore, in the future, we would like to apply our method in new fields such as the robot navigation, multi-robot system, and so on.

References

Baghaei KR, Agah A (2007) Multi-agent task allocation for robot soccer. J Intell Syst 16(3):207–240

Jouffe L (1998) Inference system learning by reinforcement methods. IEEE Trans Syst Man Cybern 28(3):338–355

Kim JH, Vadakepat P (2000) Multi-agent systems: a survey from the robot-soccer perspective. Int J Intell Autom Soft Comput 6(1):3–17

Littman ML (1994) Markov games as a framework for multiagent learning. In: Proceedings of the 11th international conference on machine learning. Morgan Kaufmann, San Francisco, pp 157–163

Liu F, Zeng GZ (2006) Multi-agent cooperative learning research based on reinforcement learning. In: 10th international conference on computer supported cooperative work in design. Nanjing, pp 1408–1413

Lu Y, Lin X (2006) Applications of neural network to short-term electric load forecasting. J Shenyang Univ Technol 28(1):41–44

Moore AW, Atkeson CG (1995) The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. Mach Learn 21(3):199–233

Murao H, Kitamura S (1997) Q-learning with adaptive state segmentation (QLASS). In: Proceedings of IEEE international symposium on computational intelligence in robotics and automation. pp 179–184

Reis LP, Lau N, Oliveira EC (2001) Situation based strategic positioning for coordinating a team of homogeneous agents. In: Lecture notes in artificial intelligence, vol 2103. pp 175–197

Specht DF (1990) Probabilistic neural networks. Neural Netw 3(1):109–118

Specht DF (1990) Probabilistic neural networks and the polynomial adaline as complementary techniques for classification. IEEE Trans Neural Netw 1(1):111–121

Stone P, Veloso M (2000) Multiagent systems: a survey from a machine learning perspective. Auton Robots 8(3):345–383

Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press Cambridge, Massachusetts

Taniguchi Y, Mori T, Ishii S (2007) Reinforcement learning for cooperative actions in a partially observable multi-agent system. In: Lecture notes in computer science, vol 4668. pp 229–238

Touzet CF (1997) Neural reinforcement learning for behaviour synthesis. Robotics Auton Syst 22(3):251–281

Veloso M, Stone P, Han K (1998) CMUnited-97: RoboCup-97 small-robot world champion team. AI Magazine 19(3):61–69

Weinberg M, Rosenschein JS (2004) Best-response multiagent learning in non-stationary environments. In: Proceedings of the thrid international joint conference on autonomous agents and multiagent systems, vol 2. pp 506–513

Wu CJ, Lee TL (2004) A Fuzzy Mechanism for action selection of soccer robots. J Intell Robotic Syst 39(1):57–70

Xie SM, Chen C, Ding XY (2007) Endpoint prediction of basic-oxygen furnace based on BP neural network. J Shenyang Univ Technol 29(6):707–710

Yang E, Gu D (2004) Multiagent reinforcement learning for multi-robot systems: a survey. Technical Report CSM-404, Department of Computer Science, University of Essex

Zhong Y (2003) Research on distributed reinforcement learning theory and its application in multi-robot systems. Dissertation for the Degree, Harbin Engineering University