Nemesis 2D Team Description 2008

Mehrab Norouzitallab, S. M. A Salehizadeh.,Ahmad Pourshoghi

Abstract. In our study, we tried to develop our teams in such a way that machine learning techniques have the main role in improving skills and increasing team performance. We consider soccer simulation platform as an uncertain and dynamic environment, so we develop learning algorithms according to this important feature and agent's partial observability.

I. INTRODUCTION

Nemesis team has commenced its activity since 2004. This team makes of the codes of Mersad team as the base code which are completed by making use of other codes of some other teams.

Our team's implementation has been under continuous development. According to the fact that soccer simulation is a challenging platform for multi-agent research, involving topics such as formation control, coverage control, plan recognition, obstacle avoidance and machine learning we try to apply AI and machine learning techniques wherever possible, some of our goals have been implemented and the others are under development.

We know that the soccer simulation presents an uncertain and dynamic environment for cooperating agents. Thus employment of Reinforcement Learning (RL) methods according to partially observable markov decision process (POMDP) is our main focus which will be described in the following sections. Afterward we apply a formation control based on collecting theory due to partial observability. Another approach we used to develop the performance of the team is applying recurrent fuzzy neural network (RFNN) for the shoot skill for agent to learning to score. At last we will employ coverage control as our future work to enhance our team's capabilities.

The setup of this paper is as follows. In section 2, we will describe POMDP and its application in soccer 2d simulation particularly in kick skill. In section 3 we propose our team formation strategy according to partially observable soccer 2D environment. And in section 4, we will propose RFNN for developing shoot skill. Section 5 presents Nemesis future works. And we will end with a conclusion in section 6.

II. POMDP FRAMEWORK

One important facet of the POMDP approach is that there is no distinction between actions taken to change the state of the world and actions taken to gain information.

For MDPs we can compute the optima policy p and use it to act by simply executing $p(s)$ for current state *s*.

According to the uncertain and dynamic soccer simulation environment what happens if the agent is no longer able to determine the state it is currently in with complete reliability? A naive approach would be for the agent to map the most recent observation directly into an action without remembering anything from the past. Somewhat better results can be obtained by adding randomness to the agent's behavior: a policy can be mapping from observations to probability distributions over actions [1]. Randomness effectively allows the agent to sometimes choose different actions in different locations with the same appearance, increasing the probability that it might choose a good action.

We decompose the problem of controlling a POMDP into two parts [2], as shown in Figure 1. The agent makes observations and generates actions. It keeps an internal *belief state, b* that summarizes its previous experience. The component labeled SE is the *state estimator*: it is responsible for updating the belief state based on the last action, the current observation, and the previous belief state. The component labeled p is the policy: as in MDP, it is responsible for generating actions, but this time as a function of the agent's belief state rather than the state of the world.

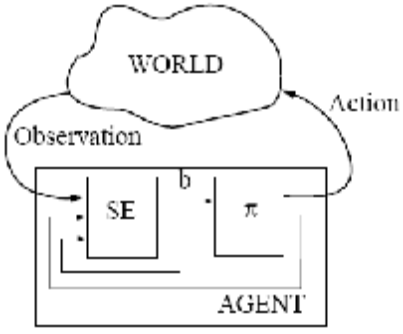

Fig1. A POMDP agent can be decomposed into a state estimator (SE) and a policy (p)

2.1 KICK SKILL

Finding a good kick routine is a very important job. Within our RL framework, this job is done by the agent in a partial observable environment. According to the fact that the agents cannot often directly access actual states of the environment, but can get only observations, which may be partial, from them, serious computational difficulties arise in estimating unobservable states. It is provided with 500 parameterized instances of the kick command (direction discrete in 100 steps, power discrete in 5 steps) together with 36 instances of the turn command. This makes an overall of 536 actions, from which the agent can choose one per cycle according to its observations. The learning agent is controlled based on one-step-ahead prediction using opponent agents' models. It is difficult, however, to apply this method without any approximation because soccer 2D

is a large-scale multi-player environment, and then the utility prediction should involve intractable integrations. To overcome this intractability, sampling is performed over a subspace under the assumption that each opponent player will perform the action which is detrimental to the learning agent [3]. The utility function is given by the following expectation with respect to the predictive distribution:

$$U(a_t, H_t) = \langle R(a_t, u_t) + V(s_{t+1}) \rangle \quad (1)$$

$$\langle f(s_{t+1}) \rangle \equiv \sum_{s_{t+1} \in S_{t+1}} P(s_{t+1} \mid a_t, H_t) f(s_{t+1}) \quad (2)$$

$R(a_t, u_t)$ denotes an immediate reward, which is $R(a_t, u_t) = 0.5 - n$ when the agent gets n penalty points (n may be 0) after the *t-th* cycle. The constant bias 0.5 is attached to make the sum of all agents' rewards zero. In our study, a normalized Gaussian network (NGnet) is used for approximating the value function V we first define a *consistent state* as follows: for a given action sequence, a state which may realize the action sequence is said to be a consistent-state. State can be represented with five elements: the ball's relative position, velocity, the length of the desired ball speed, the player's velocity, the player's body facing.

In our approach, therefore, we first sample pessimistic action sequences $\{\hat{u}_t(k) \mid k = 1, \ldots, K\}$ and then sample consistent-states $\{\hat{s}_t(k,n) \mid n = 1, \ldots, N\}$ for each pessimistic action sequence $\hat{u}_t(k)$.

Using the pessimistic action sequences, equation (2) is approximated as:

$$\langle f(s_{t+1}) \rangle \approx \sum_{\hat{u} \in U_t^-} \sum_{\hat{s}_t \in S_t^-(\hat{u}_t)} P(\hat{u}_t \mid \hat{s}_t, a_t) P(\hat{s}_t \mid H_t) f(\hat{s}_{t+1})$$

$$= \frac{1}{N} \sum_{k=1}^{K} \sum_{n=1}^{N} \prod_{i=1}^{3} P\left(\hat{a}_t^i(k) \mid \hat{o}_t^i(k,n), \hat{f}^i\right) f(\hat{s}_{t+1}(k,n)) \quad (3)$$

$\hat{a}_t^i(k)$ denotes an action of the opponent agent *i*, where $\hat{u}_t(k) = \{\hat{a}_t^1(k), \hat{a}_t^2(k), \hat{a}_t^3(k)\}$.

Here, we assume that every opponent agent, including the rule-based agents selects its action based on the action selection probability $P(a_t^i(k) \mid o_t^i, f^i)$ which depends on an observation o_t^i and the history. $\hat{o}_t^i(k,n)$ and $\hat{s}_{t+1}(k,n)$ denote an observation for agent *i*, which is determined from the state $\hat{s}_t(k,n)$ without any ambiguity, and the next state reached from state $\hat{s}_t(k,n)$ given $\hat{u}_t(k)$ and a_t, respectively.

We then use a sampling-based approximation under the observations of the game theory, and calculate the one-step-ahead utility value using the acquired model. The learning agent selects greedy actions based on the utility; such a strategy is preferable in the sense of Min-Max theory. Computer simulations showed that our model-based RL method is effective for acquiring good strategy for soccer 2D kick skill.

III. Team Formation In Partially Observable Environment

Team formation is important part in soccer simulation system, since it allows team members to focus on a team-goal, which is simpler than a global-goal over an entire system. In addition team formation allows sharing of information [3].

In this system each agent attempts to maximize a utility provided by the team using a learning algorithm such as reinforcement learning. For such a system to work properly, team utilities have to have the following properties:

- Team utilities should be easy for the agents to optimize;
- Agents optimizing their team utilities should result in agents optimizing the global utility; and
- Teams must compute utilities when they cannot fully observe each other.

In this paper we address the first property by using the theory of collectives [4] to create learnable team utilities. We address the second and third property by modifying the theory of collectives for partially observable environments to create team utilities that are "aligned" with the global utility.

There has been extensive research on rule-based agent team formations. But such systems: (i) have to be laboriously modeled; (ii) provide "brittle" global performance; (iii) are not "adaptive" to changing environments; and (iv) generally do not scale well.

To sidestep these problems, yet address the design requirements listed above (i.e., "alignedness" and "learnability") one can use the framework of collectives [4]. Given this framework, the crucial design problem becomes: Assuming the individual agents are able to maximize the team utility function through reinforcement learning, what set of team utilities, when pursued by those agents, result in high global utility?

There are two quantifiable properties that help answer this question. First, the utility functions for the team need to be "aligned" with the global utility, in that an action taken by an agent that improves its team utility also improves the global utility. Second, the utility functions need to be "learnable" in that an agent has to be able to discern the effect of its actions on its team utility and select actions that optimize that utility. The theory of collectives provides utilities for agents that maximize the second property while satisfying the first one.

We have been found that teams can be effective in environments with partial observability, if the proper team utilities are used.

We suppose a system of multi-agent teams (defense, offense teams) that aim to maximize a global utility function $G(z)$, which is a function of the joint move of all agents in the

system, z. In our work each agent in team t will try to maximize a **team utility function** $g_t(z)$.Team utilities have the advantage over agent utilities in partially observable environments in that a team utility may be able to incorporate observations from all the team members. This increase in observational capability will allow team utilities that are more "aligned" with the global utility (Figure 2). In addition team utilities allow for domains where agents are not even capable of computing their own utility, but can still blindly maximize a broadcast team utility.

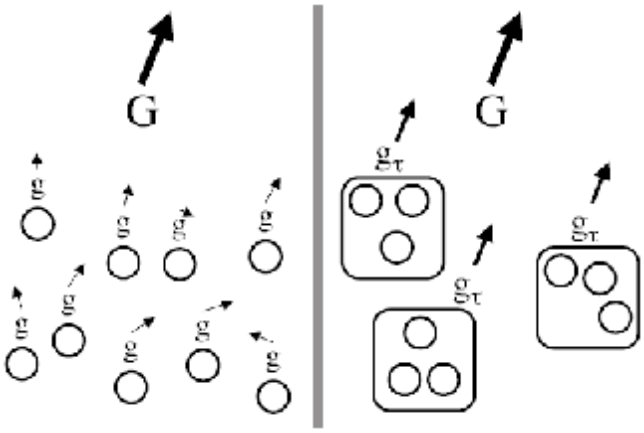

Fig 2.Team vs Agent Utilities. Left figure shows agents following agent utilities that are not fully aligned with the global utility due to partial observability. Right figure shows teams collecting observations from multiple agents allowing them to make team utilities that are more aligned.

We will use the notation z_t to refer to the parts of *z* that are dependent on the actions of team t . The vector z_t is the same size as *z* and is equal to *z* except that all the components that do not depend on team t are set to zero. By subtracting z_t from *z* we produce the vector $z_{-t} = z - z_t$, a vector that is determined by the actions of all the agents other than t .

There are two properties that are crucial to producing systems in which agents acting to optimize their team utilities will also optimize the provided global utility. The first of these concerns "aligning" the team utilities with the global utility.

Formally, a system is **factored** when for each team t [5]:

$$g_t(z) \geq g_t(z') \Leftrightarrow G(z) \geq G(z')$$

$$\forall z, z' \ \ s.t. \ \ z - z_t = z' - z'_t$$

Intuitively, for all pairs of states *z* and z' that differ only for team t , a change in t 's state that increases its team utility cannot decrease the global utility.

The second property, called **learnability**, measures the dependence of a utility on the actions of agents in a particular team as opposed to all the actions of all the other agents. Intuitively, higher learnability means it is easier for a team t to achieve a large values of its team utility.

Consider **difference** utility functions, which are of the form:

$$DU_t \equiv G(z) - G(z_{-t} + c_t) \qquad (1)$$

Where z_{-t} contains all the variable not affected by agents in team t . All the components of *z* that are affected by agents in team t are replaced with the fixed constant c_t . Such difference utilities are factored no matter what the choice of c_t , because the second term does not depend on the actions of agents in team t [4]. Furthermore, they usually have far better learnability than does the global utility, because of the second term of DU, which removes a lot of the effect of other agents (i.e., noise) from t 's evaluation function. In many situations

it is possible to use a c_t that is equivalent to taking team t out of the system. Intuitively this causes the second term of the difference utility to evaluate the global utility of the system without team t and therefore DU evaluates the teams' contribution to the global utility.

For some specific classes of utility such as the DU, this observational demand may be relaxed, since many of the elements of the worldline cancel out and may be ignored. In these cases we must approximate the utility under the constraints of partial observability. We denote the component of *z* that is observable by _ using the vector z^{Ot} and the part of *z* that is not observable by t using the vector z^{ht} . The vector z^{Ot} is the same as *z* except that all the elements that are not observable by t are set to zero. We call z^{Ot} the **observable components** of the worldline. The vector *z* is the sum of these two vectors: $z = z^{Ot} + z^{ht}$. It is assumed that team t can always observe all components of z_t . If the DU depends on any component of z^h then we cannot compute it directly. Instead there are several approximations to the DU that vary in their balance between learnability and factoredness. In this paper we discuss

four approximations[1] :

$$BTU_t(z) = G(z) - G(z^{Ot} - z_t) \qquad (2)$$

$$TTU_t(z) = G(z^{Ot}) - G(z^{Ot} - z_t) \qquad (3)$$

$$BEU_t(z) = G(z) - G(z^{Ot} + E[z^{ht} \mid z^{Ot}] - z_t) \qquad (4)$$

$$TEU_t(z) = G(z^{O_t} - E[z^{h_t} \mid z^{O_t}]) - G(z^{O_t} + E[z^{h_t} \mid z^{O_t}] - z_t) \quad (5)$$

Where $E[.]$ is the expectation operator. Note that *BTU*, as well as *BEU*, assume that the true global utility can be broadcast despite having only partial observability.

We assume that the observational capability of a team goes up with the size of the team. This property of team size can happen for a number of different reasons including, larger teams having more resources and greater coverage of different areas.

IV. Learning to Score using RFNN

Recurrent neural network systems learn and memorize information implicitly with weights embedded in them. In [6] a recurrent fuzzy neural network (RFNN) was proposed based on supervised learning, which is a dynamic mapping network and it is more suitable for describing dynamic systems than the FNN. Of particular interest is that it can deal with time-varying input or output through its own natural temporal operation [3]. Ability of temporarily storing information simplifies the network structure and fewer nodes are required for system identification. Because of complexity in back propagation (BP) learning approach, just diagonal fuzzy rules have been implemented. This limiting feature forces restrictions on users for employing more complete fuzzy rule base. In this paper a novel approach is proposed as a solution to this problem. We combined original BP used in with a particle swarm optimization (PSO) to train the network more easily and without the complexity of taking derivation.

4.1 Network Structure

The key aspects of the RFNN are dynamic mapping capability, temporal information storage, universal approximation, and the fuzzy inference system. The RFNN possess the same advantages as recurrent neural networks and extend the application domain of the FNN to temporal problems [6]. A schematic diagram of the proposed RFNN structure is shown in Figure 3. Signal propagation and the operation functions of the nodes in each layer are described below. In what follows, u_i^k denotes *i*-th input of a node in the *k*-th layer; o_i^k denotes the *i*-th node output in the *k*-th layer.

Output Layer: Each node in this layer is called an output linguistic node. This layer performs the defuzzification operation. The node output is a linear combination of the consequences obtained from each rule. That is

$$y_p = o_p^4 = \sum_{j=1}^{m} u_j^4 w_{jp}^4 \quad (5)$$

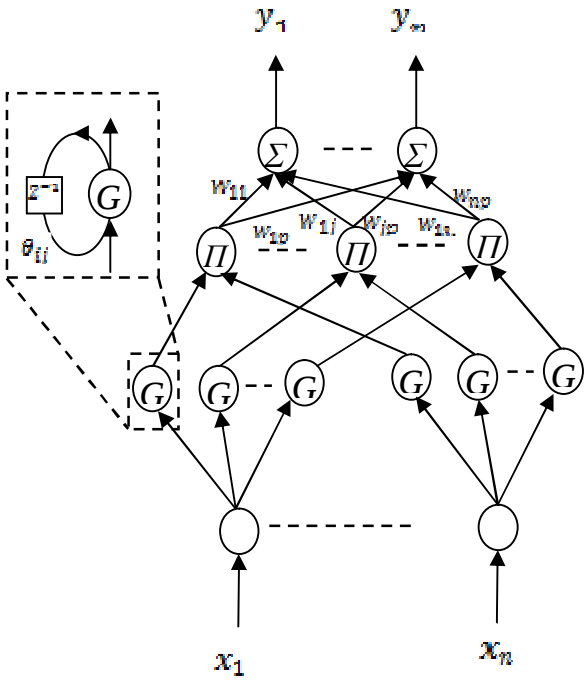

Fig3. Structure of RFNN

where $u_j^4 = o_j^3$ and w_{ji}^4 (the link weight) is the output action strength of the *i*-th output associated with the *j*-th rule. w_{ji}^4 are the tuning factors of this layer. Finally, the overall representation of input x and the *p*-th output is

$$y_p(k) = o_p^4 = \quad (6)$$

$$\sum_{j=1}^{m} w_{jp}^4 \prod_{i=1}^{n} \exp\left[-\frac{(x_i(k) + o_{ij}^2(k-1) \times q_{ij} - m_{ij})^2}{(s_{ij})^2} \right]$$

where m_{ij} , s_{ij} , q_{ij} and w_{jp} are the tuning parameters and

$$o_{ij}^2(k-1) = \exp\left[-\frac{(x_i(k-1) + o_{ij}^2(k-2) \times q_{ij} - m_{ij})^2}{(s_{ij})^2} \right]$$

Obviously, using the RFNN, the same inputs at different times yield different outputs. The proposed RFNN can be shown to be a universal uniform approximator for continuous functions over compact sets if it satisfies a certain condition. Further details about RFNN and its training approach have been described in an accepted paper by Nemesis members.

Here for training RFNN in order to learn how to score we have used following 8 parameters as inputs to RFNN: *Ball Position(x,y), Opponent Goalie Position(x,y) and body direction, relative distance and direction between ball and target points on goal line, kick power (for decreasing noise).* For this purpose we discrete goal line into 30 points and shoot region into 1800 points, and for each episode[1] we keep successful kicks and finally we choose the best kick among them according to two factors, first maximum relative distance between selected goal point and goalie position, and selected point between outline, second, power of the selected kick in regard to noise.

In order to use the trained network in our team we had to consider some situations which RFNN does not cover them. So, we used our previous computational kick when such cases occur. For example some situations in which some none goalie opponents block the way to reach the RFNN selected goal point.

V. Future work

We plan to develop our team performance by considering coverage control in both defense and offense areas.

The coverage control problem refers to the use of a collection of players to provide visual and tactical coverage for a given area of the field. Given a set of *N players*, we wish to allocate each player to a region in which it is responsible for providing visual and tactical information. One decentralized approach is to partition a region *Q* into a set of polytopes: $W = W_1,...,W_N$ that cover *Q*. Each polytope is assigned to a specific player to each region and we let $f_i : R^+ \to R^+$ represent the sensing performance of a player based on its distance from a given point, with *f* small representing good performance. We then form the coverage control problem as choosing the locations of each player such that we minimize:

$$L = \sum_{i=1}^{n} \int_{w^i} f\left(\left\| q - y^i \right\|\right) f(q) dq \quad (7)$$

Where f is a distribution density function that represents the importance of a given area.(For example around the ball has more density than beside the lines)

A key element of this approach is that the only communication required is with the nearest neighbors of the player (See Figure 4).

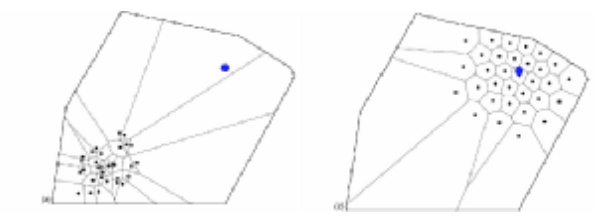

Fig. 4. Agents reform themselves such that they cover around the blue point.

VI. Conclusion

This paper presents our simulated soccer team strategies to improve Nemesis performance. In all of our implementation we consider soccer 2D environment as an uncertain and dynamic one, hence we try to apply machine learning and AI techniques according to partial observability of agents. Our research focuses on partial observable markov decision process and makes use of reinforcement learning algorithms for kick skill and in team formation. Afterward we concentrate on improving the important part of simulation, means “learning to score". In order to achieve this purpose we used a recurrent fuzzy neural network to learn shoot skill. Simulation results have been verified our approaches and show the structure improvement as well. At last we will employ coverage control as our future work to enhance our team’s capabilities.

References

[1] S. P. Singh, T. Jaakkola, and M. I. Jordan., Model-free reinforcement learning for non-Markovian decision process problems. In *Proceeding of the Eleventh International Conference on Machine Learning,* pages 284-292. San Francisco, California, 1994.

[2] L. P. Kaelbling, M.L. Littman, A. R. Cassandra. Planning and acting in partially observalble stochastic domains. Jaunuary, 13, 1997.

[3] A. Stroupe, M. C. Martin, and T. Balch., Distributed sensor fusion for object position estimation by multi-robot systems. In *IEEE International Conference on Robotics and Automation.*, IEEE, May 2001.

[4] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

[5] A. K. Agogino, K. Turner.Team formation in partially observable multi-agent systems.

[6] C.H.Lee, C.C Teng, “Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks,” IEEE Transactions on fuzzy systems, vol. 8, No. 4, 2000.

[7] S. M. A. Salehizadeh, P. Yadmellat, A. Pourshoghi, M. J. Aein. A Hybrid Algorithm for Training Recurrent Fuzzy Neural Network., In Proceeding of the International Conference of Artificial Intelligence and Pattern Recognition, 2008.