A Reinforcement Learning Based Radial-Bassis Function Network Control System*

Jianing Li, Jianqiang Yi, Dongbin Zhao, and Guangcheng Xi

Lab of Complex Systems and Intelligence Science, Institute of Automation,
Chinese Academy of Sciences, P. O. Box 2728, Beijing 100080, China
{jianing.li,jianqiang.yi,dongbin.zhao,guangcheng.xi}
@mail.ia.ac.cn

Abstract. This paper proposes a reinforcement learning based radial-basis function network control system (RL-RBFNCS) to solve non-training data based learning of radial-basis function network controllers (RBFNC). In learning process, a major contribution is by using the critic signal and the stochastic exploration method to estimate the "desired output", reinforcement learning is considered and solved from the point of view of training data based learning. Computer simulations of robot obstacle avoidance in unknown environment are conducted to show the performance of the proposed method.

1 Introduction

According to whether sufficient and consistent training data are available, learning algorithms for radial-basis function network controllers (RBFNCs) can be divided into two kinds: training data based learning such as supervised and unsupervised learning, and non-training data based learning which mainly means reinforcement learning. In general, training data based learning is a fast and effective method when training data are easy to get. But in most of real-world applications when it is expensive to obtain training data, the performance to this approach will degrade greatly. For this reason, reinforcement learning requiring no training data seems to be an attractive alternative. Enlightened by the Sutton and Barto's model presented in [1], this paper proposes a reinforcement learning based radial-basis function network control system (RL-RBFNCS), which consists of a RBFNC and a predictor, to solve non-training data based learning for RBFNCs. Based on our knowledge, there are mainly two kinds of methods for the adjustment of parameters in Sutton and Barto's model-based reinforcement learning: one such as in [2] is originated from Sutton and Barto's model; the other such as in [3] is based on gradient information. This paper is trying to solve reinforcement learning problem from another point of view. The main idea used is by employing the critic signal and the stochastic exploration method to estimate the "desired output", reinforcement learning problem is changed into training data based learning problem.

Section 2 describes the structure of a RBFNC and the RL-RBFNCS. The learning algorithm of the RL-RBFNCS is introduced in Section 3. Section 4 presents an example of robot obstacle avoidance. Finally, conclusions are given in Section 5.

* This work was partly supported by NSFC Projects (Grant No. 60334020, 60440420130, and 60475030), and MOST Projects (Grant No. 2003CB517106 and 2004DFB02100), China.

2 Structure of the RL-RBFNCS

2.1 Radial-Basis Function Network Controller (RBFNC)

This paper employs the most basic form for the construction of RBFNCs, which involves three layers, as shown in Fig. 1. The role of the input layer is for connecting the network to its environment. The hidden layer applies a nonlinear transformation from the input space to the hidden space. The output layer is comprised of linear nodes that supply the response to the network activated by the input vector. For convenience, the RBFNC is defined with two inputs and a single output. Assuming all radial basis functions for the hidden nodes use Gaussian curves, which have the same dimensions as the input vector. Next, the functions of the nodes in the hidden and the output layer shall be described by equations, in which I_i^n and O_i^n are the input and output value of the ith node in Layer n; m_{ij} and σ_{ij} are the center and width of the jth-dimensional Gaussian curve of the ith node in the hidden layer; W_i is the link weight connected with the ith hidden node in the output layer.

The hidden layer:

$$O_i^2 = \exp[-\frac{(I_1^1 - m_{i1})^2}{2\sigma_{i1}^2} - \frac{(I_2^1 - m_{i2})^2}{2\sigma_{i2}^2}] \ , i = 1,2,\cdots N \ . \tag{1}$$

The output layer:

$$I_i^3 = O_i^2 \ and \ O_1^3 = \sum_{i=1}^{N} W_i I_i^3 \ . \tag{2}$$

2.2 Structure of the RL-RBFNCS

Fig. 2 shows the structure of the RL-RBFNCS, which involves the above RBFNC and a predictor. The RBFNC can choose a proper control output according to the current input vector. The predictor performs the single or multi-step prediction of the external reinforcement signal. In this paper, we use a simple three-layer perceptron with one output node to model the predictor, which shares the same input layer with the RBFNC. The functions of the hidden and the output layer are described as follows, in which M is the number of the hidden nodes; V_{ij} and U_i are link weights of the hidden and the output layer, respectively; other symbols are defined as the same as previously.

The hidden layer:

$$I_i^2 = \sum_{j=1}^{2} V_{ij} O_j^1 \ , \ (O_j^1 = I_j^1) \text{ and } O_i^2 = \frac{1-\exp(-I_i^2)}{1+\exp(-I_i^2)} \ i = 1,2\cdots M \tag{3}$$

The output layer:

$$I_1^3 = \sum_{i=1}^{M} U_i O_i^2 \text{ and } O_1^3 = \frac{1-\exp(-I_1^3)}{1+\exp(-I_1^3)} \ . \tag{4}$$

Assuming the predictor works under single-step mode, which means a reinforcement signal is only one step behind its corresponding action. The working process of the RL-RBFNCS is described briefly as follows: at step *t*, the input vector *x*(*t*) supplied by the environment is fed simultaneously into the RBFNC and the predictor. Based on $x(t)$, the predictor produces a signal $p(t+1)$, which is the prediction of the external reinforcement signal $r(t+1)$ but available at step t; and the RBFNC gets an output variable $y(t)$. Then using $p(t+1)$ and $y(t)$, the actual output $\hat{y}(t)$ is chosen by the stochastic exploration method introduced in the next section. Driven by $\hat{y}(t)$, the system evolves to step $t+1$ and gets $r(t+1)$ by interacting with the environment. Finally, link weights of the predictor are updated by the internal reinforcement signal $\hat{r}(t+1)$, which is the prediction error computed by $r(t+1)$ and $p(t+1)$. $\hat{r}(t+1)$, $y(t)$ and $\hat{y}(t)$ are used for the adjustment of link weights and parameters of Gaussian radial-basis functions of the RBFNC. The learning algorithms will be presented in detail in the following section.

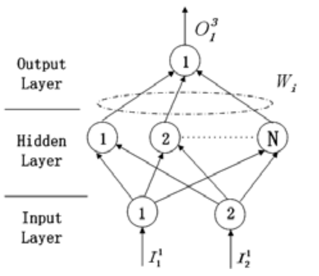

Fig. 1. Structure of the RBFNC.

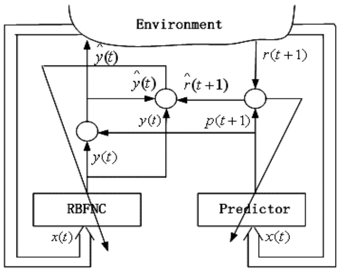

Fig. 2. Structure of the RL-RBFNCS.

3 Learning Algorithm for RL-RBFNCS

3.1 Learning Algorithm for the Predictor

The goal of the predictor learning is to adjust link weights to minimize $\hat{r}$ at each step, which, based on the single-step mode, is computed by $\hat{r}=r-p$. Here, the gradient descent learning is used to update the link weights of the predictor. The error function is defined as $E=(r-p)^2/2$. According to the chain rule, the updated values of the link weights of the predictor are as follows, in which η_1 and η_2 are learning rates.

$$\Delta U_i = \eta_1(-\frac{\partial E}{\partial U_i}) = \eta_1(r-p)\left[\frac{2\exp(-I_1^3)}{(1+\exp(-I_1^3))^2}O_i^2\right]. \tag{5}$$

$$\Delta V_{ij} = \eta_2(-\frac{\partial E}{\partial V_{ij}}) = \eta_2(r-p)\left[\frac{2\exp(-I_1^3)U_i}{(1+\exp(-I_1^3))^2}\frac{2\exp(-I_i^2)O_j^1}{(1+\exp(-I_i^2))^2}\right]. \tag{6}$$

3.2 Stochastic Exploration Method

When $y(t)$ is produced by the RBFNC, the conflict between the desire to use $y(t)$ and the desire to further explore the environment to improve the performance of the

RBFNC has to be considered. This paper uses the stochastic exploration method proposed in [3] to overcome this problem, which is described as follows.

Step 1. The range of stochastic exploration is determined by the following equation, in which K and A are search-range scaling parameters.

$$\sigma(t) = \frac{K}{1+\exp[Ap(t+1)]} \tag{7}$$

Step 2. A Gaussian random variable $\hat{y}(t)$ is chosen by exploring the range $\sigma(t)$ around the mean point $y(t)$.

$\sigma(t)$ can be interpreted as the extent to which the output variable searches for a better action. $p(t+1)$ is the prediction of $r(t+1)$. When $p(t+1)$ is small, $\sigma(t)$ will be large, which means to broaden the search range around the $y(t)$. This can provide a higher probability to choose a $\hat{y}(t)$, which is far from $y(t)$, since $y(t)$ is regarded to be far from the best action possible for the current input vector. The similar analysis is true when $p(t+1)$ is large. By using this method, the RBFNC explores for actions possible, then a better output will be rewarded and a worse one be punished by the learning algorithm of the RBFNC introduced in the next subsection.

3.3 Learning Algorithm for the RBFNC

The goal of the RBFNC learning is to adjust link weights and centers and widths of Gaussian radial-basis functions to maximize r at each step, which means to produce an optimal action for each input vector. Basically, the difference between training data based learning and reinforcement learning is: for each input vector, the former can get the instructive signal, which is described as the desired output; and the later has only the critic signal, which represents a reward or a penalty for the output action. If the "desired output" can be produced by employing the critic signal, then a reinforcement learning problem can be changed into a training data based learning problem. In this paper, the learning of the reinforcement-based RBFNC is considered and solved just based on this idea.

Firstly, using the method presented in [4], the desired output is estimated by

$$y_d(t) \approx y(t) + \rho\frac{\partial r}{\partial y} \text{ and } \frac{\partial r}{\partial y} \approx [r-p]_{t+1}\left[\frac{\hat{y}(t)-y(t)}{\sigma(t)}\right]. \tag{8}$$

where ρ is a real number, $\rho \in \{0,1\}$; $y_d(t)$ is the estimated output. If $r(t+1) > p(t+1)$, which means $\hat{y}(t)$ is better than $y(t)$, $\hat{y}(t)$ should be rewarded. So $y_d(t)$ is moved closer to $\hat{y}(t)$. On the other side, $y_d(t)$ is moved further away from $\hat{y}(t)$. When the desired output is produced, the reinforcement learning can be changed completely into a training data based learning. The gradient descent learning is used again to update all parameters of the RBFNC. The error function is defined as $E = (y_d - y)^2/2$. The updated values of the RBFNC are as follows.

$$\Delta W_i = \beta_1(-\frac{\partial E}{\partial W_i}) = \beta_1(y_d - y)I_i^3 \tag{9}$$

$$\Delta m_{ij} = \beta_2(-\frac{\partial E}{\partial m_{ij}}) = \beta_2(y_d - y)W_i I_i^3 \frac{(I_j^1 - m_{ij})}{\delta_{ij}^2} \tag{10}$$

$$\Delta \delta_{ij} = \beta_3(-\frac{\partial E}{\partial \delta_{ij}}) = \beta_3(y_d - y)W_i I_i^3 \frac{(I_j^1 - m_{ij})^2}{\delta_{ij}^3} \tag{11}$$

4 An Illustrative Example

The proposed RL-RBFNCS is simulated for on-line obstacle avoidance of mobile robot in unknown indoor environment that consist of walls and static obstacles, in which walls also are treated as obstacles. This paper employs the behavior-based control architecture presented in [5], which maps sensor information to control command directly. The model of the mobile robot used is a cylindrical platform and 18 infrared sensors equipped evenly in a ring, which are assumed to work in perfect mode. To reduce input dimension, the sensors around the robot are divided into six groups ($S_1 \sim S_6$), each of which consists of three neighboring sensors, as depicted in Fig. 3. The distance measured by each sensor group is defined as the smallest value. We do not consider the rotation of the robot, that means the robot coordinate system is consistent with the world coordinate all the time. Assuming the robot to move with a constant linear velocity, the control variable is defined as the angle from x-axis. r is defined as related with the distance measure supplied by sensors. As long as there are obstacles in sensor field of view, a r can be obtained at each step, so the reinforcement learning is a single-step prediction problem. Without using a normal two-valued number, the value of r is defined as a real number, $r \in \{0,1\}$, which represents a detailed and continuous degree of success or failure. When r is larger than a set safety threshold, r will keep the same changing trend with the smallest distance measure among sensor groups.

In Fig. 4, driven by the RL-RBFNCS, the robot begins to move from A and B, respectively. The region between the two dotted circles represents the detectable range of sensors. The folded lines are the trajectories of the robot center. At the start, $p(t+1)$, $y(t)$ and $\hat{y}(t)$ are set to zero; $\sigma(t)$ set to unity; all link weights are set to small nonzero values; m_{ij} are set at random in the range of detection distance; σ_{ij} are computed by $\sigma_{ij} = \sqrt{D/m}$, in which D is maximum distance of j chosen centers and m is the number of input variables. Parameters for learning are shown in Table 1. When the smallest distance measure is lower than the given threshold, the robot is backtracked two steps and $p(t+1)$, $y(t)$, $\hat{y}(t)$ and $\sigma(t)$ are set again with initial values. If there are no obstacles in sensor field of view, the robot will go along the former direction. In learning process, by adjusting the search-range scaling parameters, the search range can be broadened to speed up learning. While it should be also noticed that a too large search range will degrade learning in other situations.

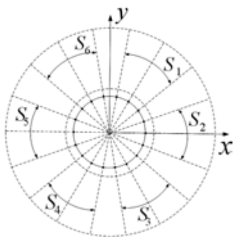

Fig. 3. Mobile robot and sensor arrangement.

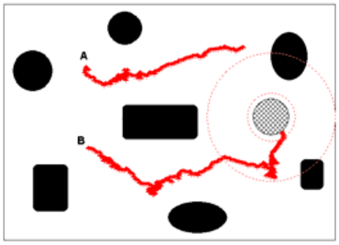

Fig. 4. Simulation of robot obstacle avoidance.

Table 1. Parameters for learning of the RL-RBFNCS.

$\eta_1 = 0.9$	$\eta_2 = 0.9$	$K = 1$	$A = 1$	$\rho = 1$	$\beta_1 = 0.8$	$\beta_2 = 0.1$	$\beta_3 = 0.1$

5 Conclusion

This paper describes a RL-RBFNCS to solve non-training data based learning of RBFNCs, which consists of a RBFNC and a predictor. By using the critic signal and the stochastic exploration method to estimate the "desired output", the reinforcement learning is treated and realized from the point of view of training data based learning. Employing the proposed method, the RBFNC and the predictor will be gradually close to an optimal action controller and an optimal state evaluation unit, respectively. Since our method greatly lessens the quality and quantity requirements of training data for the learning of RBFNCs, the design of RBFNCs will be more practical for real-world applications. Computer simulations show the effectiveness and applicability of the proposed approach. Future work will focus on adaptive adjustment for the structure of RBFNCs and multi-step prediction problem.

References

1. Barto, A.G., Sutton R.S., Anderson C.W.: Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems. IEEE Trans. Syst, Man, Cybern, **13** (1983) 834-846.
2. Ye, C., Yung, N.H.C., Wang D.W.: A Fuzzy Controller with Supervised Learning Assisted Reinforcement Learning Algorithm for Obstacle Avoidance. IEEE Trans. Syst, Man, Cybern, **33** (2003) 17-27
3. Lin, C.T., Lee, G.S.G.: Reinforcement Structure/Parameter Learning for Neural-Network-Based Fuzzy Logic Control Systems. IEEE Trans. Fuzzy Syst, **2** (1994) 46-63
4. Lin, C.T., Lin, C.T.: Reinforcement Learning for an ART-Based Fuzzy Adaptive Learning Control Network. IEEE Trans. Neural Network, **7** (1996) 709-731
5. Brooks, R.A.: Robust Layered Control Systems for a Mobile Robot. IEEE Trans. Robot. Automat., **2** (1986) 14-23