Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 177 (2007) 3764–3781
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

A fuzzy Actor–Critic reinforcement learning network

Xue-Song Wang [a,*], Yu-Hu Cheng [a], Jian-Qiang Yi [b]

[a] School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou, Jiangsu 221008, China
[b] Laboratory of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China

Received 8 April 2005; received in revised form 3 March 2007; accepted 10 March 2007

Abstract

One of the difficulties encountered in the application of reinforcement learning methods to real-world problems is their limited ability to cope with large-scale or continuous spaces. In order to solve the curse of the dimensionality problem, resulting from making continuous state or action spaces discrete, a new fuzzy Actor–Critic reinforcement learning network (FACRLN) based on a fuzzy radial basis function (FRBF) neural network is proposed. The architecture of FACRLN is realized by a four-layer FRBF neural network that is used to approximate both the action value function of the Actor and the state value function of the Critic simultaneously. The Actor and the Critic networks share the input, rule and normalized layers of the FRBF network, which can reduce the demands for storage space from the learning system and avoid repeated computations for the outputs of the rule units. Moreover, the FRBF network is able to adjust its structure and parameters in an adaptive way with a novel self-organizing approach according to the complexity of the task and the progress in learning, which ensures an economic size of the network. Experimental studies concerning a cart–pole balancing control illustrate the performance and applicability of the proposed FACRLN.

Keywords: Reinforcement learning; Actor–Critic learning; Fuzzy inference system; Radial basis function neural network

1. Introduction

Reinforcement learning is a framework for an agent to learn to act optimally based on reinforcement signals. Reinforcement learning methods have been widely applied to a variety of problems, such as autonomous robot navigation and nonlinear control [7,22]. However, most conventional reinforcement learning methods use a discrete state space to describe the interaction between the agent and its environment. They assume that the task is a Markov Decision Process (MDP), in which a transition probability is defined for discrete states given a discrete action. Reinforcement learning will encounter large-scale or continuous spaces when it is applied to real-world problems. A straightforward approach to continuous spaces learning would be to make the entire state space into discrete, equal-sized boxes. However, this will lead inevitably to the curse of the dimensionality problem and cannot guarantee the performance of the reinforcement learning system.

* Corresponding author. Tel./fax: +86 516 83883204.
E-mail address: wangxuesongcumt@163.com (X.-S. Wang).

0020-0255/$ - see front matter
doi:10.1016/j.ins.2007.03.012

Therefore, in order to realize an approximation of the optimal value and policy functions for continuous state or action spaces, reinforcement learning should have a capability of generalization. The essence of generalization is to use a function to approximate unlearnt mapping from a state space to an action space [2]. The design of a function approximator with high capabilities of generalization and computational efficiency has become a major focus in the research field of reinforcement learning.

Value function approximation is widely taken into consideration in the generalization of reinforcement learning. Q learning, Temporal Differences (TD) learning and Sarsa learning are algorithms normally used in reinforcement learning with common characteristics, i.e., they all estimate the value function of MDP. The corresponding behavior selection policies are fully determined by the estimation of the value function. On the other hand, approximating the value and policy functions simultaneously basically adopts the Adaptive Heuristic Critic (AHC) algorithm under the Actor–Critic (AC) architecture proposed by Barto et al. [2]. In the AHC algorithm, the Critic realizes the approximation of the value function and the Actor carries out the approximation of the policy function. Barto et al. [2] designed an AHC algorithm based on two single-layer neural networks to realize the control for an inverted pendulum under a discrete state. Anderson [1] further realized the balancing control for an inverted pendulum under a non-discrete state by an AHC algorithm, based on two two-layer feedforward neural networks. The algorithms proposed in [2,1] divided the state space into finite numbers of sub-spaces without any generalization between sub-spaces. As a consequence, suitable division results for complex and uncertain systems could not be obtained. Correlations between state variables were difficult to be embodied into the algorithm and the control structure was excessively complex.

Wang and Mendel [27] proved that a fuzzy inference system (FIS) is actually a universal approximator which can approximate any input–output data with arbitrary precision on a compact set. In theory, fuzzy systems and neural networks are functionally equivalent under some mild restrictions [9]. But it should be noted that these restrictions do not guarantee the equivalence of the two systems in terms of semantic meaning. The main difference between neural networks and fuzzy inference systems is the ability for interpretation. FISs are suitable for representing fuzzy and uncertain knowledge, which agrees quite well with the process of human thinking [29,12]. But fuzzy systems lack self-learning and self-adaptive abilities. On the other hand, it is known that neural networks have advantages of parallel computation, fault-tolerance and self-learning, but they are not suitable for representing knowledge [6]. Hence, functional equivalence has made it possible to combine the features of these two systems, which has been developed into powerful fuzzy neural network systems (FNNs). In these hybrid systems, the fuzzy techniques are actually used to create neural networks or enhance their performance, while neural networks are used to learn membership functions and fuzzy rules [18,20].

In general, fuzzy systems can be realized in an architecture isomorphic to a neural network, i.e., a multiple-layer perceptron (MLP) network, where each unit performs a function such that the entire network becomes perfectly equivalent to a fuzzy system. Back propagation algorithm will still be possible to train the network, such as ANFIS (adaptive-network-based fuzzy inference system) by Jang [8] or FALCON (fuzzy adaptive learning control network) by Lin et al. [17]. Fuzzy neural networks have good characteristics; therefore, much attention has been focused on utilizing ANFIS or FALCON to realize reinforcement learning. Typical examples of FNN-based reinforcement learning systems include ARIC (approximate reasoning based intelligent control) [3], GARIC (generalized ARIC) [4], RNN-FLCS (reinforcement neural-network-based fuzzy logic control system) [15] and FALCON-RL (reinforcement learning strategy based on fuzzy adaptive learning control network) [16]. Most of these reinforcement learning systems adopted two separate MLP networks to fulfill the functions of Actor and Critic networks. The architecture of the system is complex. ANFIS, ARIC and GARIC belong to a category of off-line training systems, i.e., they require a set of training data available beforehand, first to set up the structure and then to tune the parameters appropriately. Moreover, they do not have a capability of structure learning, i.e., the network structure is determined and fixed in advance. Although the topology structure and the parameters of the networks can be tuned on-line simultaneously in RNN-FLCS [15] and FALCON-RL [16], the number and configuration of membership functions for each input and output variable have to be decided by an expert in advance. Another AHC algorithm based on a decomposition of the adaptive basis function was proposed in [23] where the total state space was divided according to approximation errors. The algorithm has some adaptability, but it could not solve the problem of low learning efficiency with the increased complexity of the network structure. Jouffe [11] designed two fuzzy reinforcement learning methods such as fuzzy AC learning (FACL) and fuzzy Q learning (FQL) based

on dynamic planning theory. These two methods merely tuned the parameters of the consequent part of a FIS by using reinforcement signals received from the environment, while the premise part of the FIS was fixed during the learning process. Therefore, they could not realize the adaptive establishment of a rule base, limiting its learning performance. Kondo and Ito [13] proposed an evolutionary state recruitment strategy for AC learning. The strategy was able to allocate appropriate numbers and sizes of normalized Gaussian network units, but it still required an off-line training phase to decide its model parameters in advance.

Radial basis function (RBF) neural networks, capable of universal approximation, utilize basis functions in the hidden layer. The advantages of a RBF network are twofold: the training procedure is substantially faster than the MLP network and there is no local minimum problem [19,21]. Based on this analysis, we propose a new fuzzy Actor–Critic reinforcement learning network (FACRLN) including a system architecture and a learning algorithm based on a fuzzy RBF (FRBF) network. We use a four-layer FRBF network to approximate both the action function of the Actor and the value function of the Critic simultaneously. The reinforcement learning method can solve the curse of the dimensionality problem resulting from making continuous spaces discrete. As well, a novel self-organizing approach is proposed to add and merge units in the rule layer of the FRBF network dynamically, which makes it possible to obtain a reasonably sized network after the learning process.

The paper consists of five sections. Based on our analysis of the functional equivalence between a RBF network and a fuzzy inference system, the fuzzy Actor–Critic reinforcement learning architecture is proposed in Section 2. Section 3 provides the learning algorithm for tuning the structure and the parameters of the FRBF network in a self-organizing manner. In Section 4, we apply the proposed FACRLN to a cart–pole balancing problem and compare it with other reinforcement learning methods. The last section presents some concluding remarks.

2. FACRLN architecture

2.1. Functional equivalence between a RBF network and a FIS

A RBF network is one of the most important neural networks, which can approximate any nonlinear function with arbitrary precision, without local minimum problem [21]. A RBF network with a single hidden layer and single output can be described as:

$$y = \sum_{j=1}^{h} w_j \varphi_j \left(\frac{\|x - \mu_j\|}{\sigma_j} \right), \tag{1}$$

where $x \in R^n$ denotes a n dimensional input observation, $\varphi_j(\cdot)$ the jth radial basis function or receptive field unit, $\mu_j \in R^n$ and $\sigma_j \in R^n$ the center (mean) and the width (variance) vectors of the jth basis function respectively, w_j the weight or firing strength of the jth basis function, j a basis function index and h the number of basis functions. Eq. (1) can be described as Eq. (2) when the basis function of the RBF network is a Gaussian function and the output of the RBF network is normalized.

$$y = \frac{\sum_{j=1}^{h} w_j \prod_{i=1}^{n} \exp\left[-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2} \right]}{\sum_{j=1}^{h} \prod_{i=1}^{n} \exp\left[-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2} \right]}. \tag{2}$$

On the other hand, a FIS is a system based on knowledge or rules. The most important part of a FIS is its rule base made up of a number of 'if–then' rules. A T–S model is a very important fuzzy inference system that has wide applications in fuzzy modeling and fuzzy control domains [25,26]. The jth 'if–then' fuzzy rule of a T–S model can be described as:

$$R_j: \textbf{\textit{if}}\ x_1 \text{ is } A_1^j \quad \text{and } x_2 \text{ is } A_2^j \quad \text{and } \ldots \text{ and } x_n \text{ is } A_n^j \ \textbf{\textit{then}}\ y^j = a_0^j + a_1^j x_1 + \cdots + a_n^j x_n, \tag{3}$$

where the input variable vector is $x = [x_1, x_2, \cdots, x_n]^{\mathrm{T}}$. A_i^j denotes the jth fuzzy set of the ith input variable, y^j the output of the consequent part of the jth fuzzy rule and i an input variable index, $i = 1, 2, \ldots, n$. The 'if' part

of a T–S model is similar to the premise part of a common fuzzy rule of the Mamdani model, but the consequent part of a T–S model is described by a crisp function of the input variables. A T–S model includes not only the qualitative knowledge represented by the premise part, but also the quantitative knowledge represented by the consequent part. Thus, these two kinds of knowledge are easily formulated into a unified mathematical framework by using a T–S model.

Given the input vector, the overall output of the fuzzy system can be obtained by:

$$y = \frac{\sum_{j=1}^{h}\left[fis_j \overset{n}{\underset{i=1}{\mathrm{T}}} \varphi_{ij}(x_i)\right]}{\sum_{j=1}^{h} \overset{n}{\underset{i=1}{\mathrm{T}}} \varphi_{ij}(x_i)}, \tag{4}$$

where $\varphi_{ij}(x_i)$ denotes the membership function for fuzzy set A_i^j, fis_j a crisp function of x_i and T a conjunction operator in fuzzy set operations. In order to obtain an adaptive fuzzy inference system, $\varphi_{ij}(x_i)$ should be differentiable and normalized. Therefore, the membership function $\varphi(\cdot)$ can be adopted as a multi-dimensional Gaussian function. We can obtain Eq. (2) from Eq. (4) if we substitute the multi-dimensional Gaussian function for $\overset{n}{\underset{i=1}{\mathrm{T}}} \varphi_{ij}(x_i)$ in Eq. (4).

It is clear that a RBF network and a FIS have functional equivalence when certain conditions are satisfied. There are two aspects of equivalent characteristics. First of all, the membership functions of the input vectors in a FIS should be equivalent to the basis functions of RBF hidden units. In addition, the consequent part of each fuzzy rule is equivalent to the connection weight between the hidden layer and the output layer of a RBF network. The functional equivalence between a RBF network and a FIS can be described by the following theorem [9].

Theorem. *A RBF network made up of radial basis functions and a fuzzy inference system made up of some fuzzy rules have the same working mechanism. That is, the radial basis functions of a RBF network and the membership functions of a FIS both can localize the input observation by using local receptive field characteristics to produce a central weighted response. Functional equivalence between a RBF network and a FIS can be established if the following criteria are met.*

1. *The number of RBF hidden units is equal to the number of fuzzy 'if–then' rules.*
2. *The output of each fuzzy 'if–then' rule is a constant (the fuzzy system is a zero-order T–S fuzzy system).*
3. *The radial basis function of each RBF hidden unit is the same as the membership function within each fuzzy rule.*
4. *The multiplication operator is used to compute the firing strength of each rule.*
5. *Both the RBF network and the fuzzy inference system under consideration use the same method (i.e., either weighted average or weighted sum) to derive their overall outputs.*

We will concentrate on the zero-order T–S fuzzy system since it has the interesting property of being equivalent to a RBF network as described above.

2.2. FACRLN architecture

Although the restrictions, imposed by Jang and Sun [9], do result in the functional equivalence between a RBF network and a fuzzy system, they do not guarantee the equivalence of the two systems in terms of their semantic meaning. The main difference between a RBF network and a fuzzy inference system is the ability for interpretation. From the point of knowledge expression, a RBF network can, in fact, be viewed as a kind of network expression form of fuzzy rules. Each hidden unit denotes a fuzzy rule. Therefore, we can combine a FIS with a RBF network to form a fuzzy radial basis function (FRBF) network [10]. A FRBF network makes its inner connecting structure more explicit and has adaptive learning ability. The FRBF network adopted here is a four-layer feedforward network, which can describe the commonly used five steps of a fuzzy inference system. Compared with ANFIS and FALCON, both of which adopt a five-layer MLP neural network to

realize a fuzzy inference system, the FRBF network structure is much simpler. In this way, the learning time can be reduced and the learning efficiency is further improved. During the learning process, the units in the rule layer have self-organizing characteristics, i.e., the FRBF network has an adaptive adjustment mechanism according to the complexity of the task and the progress in learning.

The architecture of FACRLN is schematically shown in Fig. 1. There are two essential components of this associated reinforcement learning architecture. One is the Actor network, which learns the state-to-action mapping and the other is the Critic network, which learns to estimate the value function of the policy followed by the Actor [16,30]. The Actor network can choose a recommended action or decision according to the current input vector. Moreover, a stochastic action modifier (SAM) is used to generate stochastically an actual action according to the recommended action $A_k(x_t)$, suggested by the Actor network and the estimated signal $V(x_t)$ from the Critic network. The Critic network receives an external reinforcement signal (i.e., reward) r_t from the environment and produces a TD error (i.e., internal reinforcement signal) $\delta_{\mathrm{TD}}(t)$ sent to the Actor network. The network learning of the Actor and the Critic can proceed through the TD error signal.

It is to be noted that there is only one FRBF network, as shown in Fig. 2, which is used to implement the Actor–Critic reinforcement learning. The Actor and the Critic networks share the same layers 1, 2 and 3 of the FRBF network. Following are our reasons for adopting this course of action. In the first place, the inputs of the Actor and the Critic are both the same vectors derived from the environment and their differences are largely due to their various outputs. The second reason is that the hidden layer of both Actor and Critic interprets the input space by means of local responses. Therefore, from the viewpoint of space division, both Actor and Critic can share the rule layer of the FRBF network. This routine can reduce the demands for storage space from the learning system and avoid repeated computations for the outputs of the rule units. The definite meaning of each layer is described as follows:

Layer 1 is an input layer. Each unit in this layer denotes a crisp input variable x_i where i is an input index. The input vector $x = (x_1, x_2, \ldots x_n)^{\mathrm{T}} \in R^n$ is transmitted directly to the next layer.

Layer 2 is a rule layer. Each unit in the rule layer represents the premise part of a fuzzy rule and has n membership functions. In order to obtain an adaptive fuzzy inference system, the n membership functions take the following differentiable Gaussian function:

$$\mathrm{MF}_{ij} = \exp\left(-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2}\right), \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, h \tag{5}$$

where MF_{ij} denotes the ith membership function in the jth rule unit. μ_{ij} and σ_{ij} are the center and the width of the membership function MF_{ij}, respectively. At time t, the output of the jth rule unit is computed from a n dimensional input observation x_t as Eq. (6). $\varphi_j(x_t)$ denotes the firing strength of the jth rule unit and is equal to the multiplication of the inner n membership functions.

$$\varphi_j(x_t) = \prod_{i=1}^{n} \mathrm{MF}_{ij}(x_{it}) = \exp\left(-\sum_{i=1}^{n} \frac{(x_{it} - \mu_{ij})^2}{2\sigma_{ij}^2}\right), \tag{6}$$

where x_{it} denotes the ith input variable within the input vector x_t at time t.

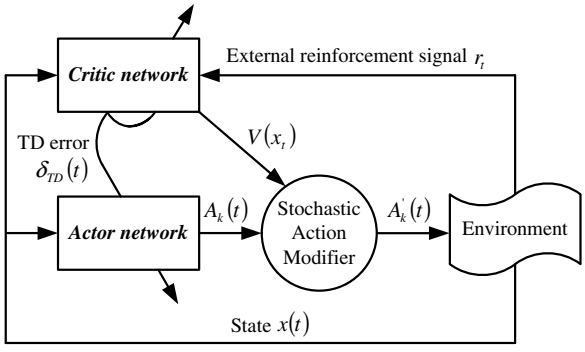

Fig. 1. Architecture of a FACRLN control system.

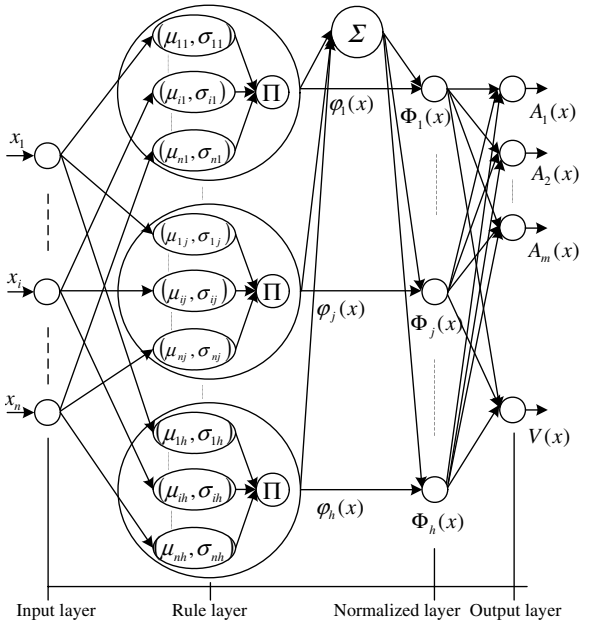

Fig. 2. FRBF-based Actor and Critic networks.

Layer 3 is a normalized layer. The number of units in this layer is equal to that of the rule layer. The effect of the layer is to perform a normalization operation on each rule. Although the normalization operation increases the computing complexity, it can make these points, located in overlapping receptive fields, to have a better interposition performance. The output $\Phi_j(x_t)$ of the jth unit in this layer denotes the normalized firing strength of the jth fuzzy rule.

$$\Phi_j(x_t) = \frac{\varphi_j(x_t)}{\sum_{l=1}^{h}\varphi_l(x_t)} = \frac{\exp\left(-\sum_{i=1}^{n}\frac{(x_{it}-\mu_{ij})^2}{2\sigma_{ij}^2}\right)}{\sum_{l=1}^{h}\exp\left(-\sum_{i=1}^{n}\frac{(x_{it}-\mu_{jl})^2}{2\sigma_{il}^2}\right)}, \quad \left(\forall \mathbf{x}, \sum_{j=1}^{h}\Phi_j(\mathbf{x}) = 1\right). \tag{7}$$

Layer 4 is an Actor–Critic output layer. The layer is made up of an Actor part and a Critic part. The Critic part carries out the estimation of the state value function, i.e., finding a mapping from a state to an expected Critic value $V(x_t) \in R^1$. The Actor part acts as an action selector that realizes a mapping from a n dimensional sensing state space $x_t \in R^n$ to a m dimensional action space $A(x_t) \in R^m$. The kth action output $A_k(x_t)$ of the Actor part and the state value function $V(x_t)$ of the Critic part are calculated as:

$$A_k(x_t) = \sum_{j=1}^{h} w_{kj}\Phi_j(x_t) \tag{8}$$

$$V(x_t) = \sum_{j=1}^{h} v_j\Phi_j(x_t) \tag{9}$$

where w_{kj} denotes the weight between the jth normalized unit and the kth Actor unit and v_j the weight between the jth normalized unit and the single Critic unit.

In order to solve the dilemma of 'exploration' and 'exploitation', the output $A_k(x_t)$ of the Actor part does not directly act on the environment [13]. A noise term n_k that has a normal distribution is added to $A_k(x_t)$. Consequently the actual action $A'_k(x_t)$ is modified as:

$$A'_k(x_t) = A_k(x_t) + n_k(0, \sigma_V(t)) \tag{10}$$

where $\sigma_V(t) = \frac{1}{1+\exp(2V(x_t))}$ denotes the width of the Gaussian noise which can be interpreted as the range where the output unit searches for a proper action. If $V(x_t)$ is small, the exploratory range $\sigma_V(t)$ will be large. This amounts to broadening the search range such that the actual action can have a high probability of being quite different from the recommended action. In contrast, if $V(x_t)$ is large, $\sigma_V(t)$ is small. This amounts to narrowing the search range such that the actual action can have a high probability of being very close to $A_k(x_t)$.

It is to be noted that FACRLN can be easily extended to realize other reinforcement learning methods, for example, fuzzy Q learning. The meaning of the input, rule and normalized layers of a fuzzy Q learning network are identical with that of FACRLN. The major difference between these two reinforcement learning networks lies in the output layer. The output layer of a fuzzy Q learning network is made up actions and Q values.

3. FACRLN learning algorithm

When using a neural network to create a fuzzy inference system, the key problem is the design of a neural network learning algorithm that optimizes the structure and the parameters of the network. The learning process of FACRLN is in fact the FRBF network learning, which includes the following structure learning and parameter learning:

3.1. Structure learning

We can obtain a better generalization performance and much higher convergence speed by keeping the network structure compact during the learning process. Structure learning includes two operations. (1) adding units: whether a new rule unit has to be added or not depends on the TD error and the *if-part* criteria. (2) merging units: whether to merge similar units in the rule layer or not is based on the fuzzy similarity measure criterion.

3.1.1. Adding units

(1) TD error criterion
The TD error criterion is proposed by defining the local errors of each basis function. At first, for the whole Actor–Critic learning system, the exponential moving average (EMA) f of the TD error and the exponential moving average $\hbar$ of the squared TD error can be calculated as follows, based on the definition of EMA [5]:

$$f(t+1) = (1-\xi)f(t) + \xi\delta_{\mathrm{TD}}(t), \tag{11}$$

$$\hbar(t+1) = (1-\xi)\hbar(t) + \xi\delta^2_{\mathrm{TD}}(t), \tag{12}$$

where ξ denotes a moving factor, $\delta_{\mathrm{TD}}(t)$ the TD error and $\delta^2_{\mathrm{TD}}(t)$ the squared TD error.

If the EMAs of the TD error and the squared TD error of the whole Actor–Critic learning system are assigned to each basis function, then we can obtain the following definition of the local errors of each basis function [23,24].

Definition. For each basis function Φ_j, the exponential moving average f_j of the TD error, weighted by $\varphi_j(x_t)$, the firing strength of the basis function, and the exponential moving average $\hbar_j$ of the squared TD error, also weighted by $\varphi_j(x_t)$ are calculated by:

$$f_j(t+1) = (1-\xi_j)f_j(t) + \xi_j\varphi_j(x_t)\delta_{\mathrm{TD}}(t), \tag{13}$$

$$\hbar_j(t+1) = (1-\xi_j)\hbar_j(t) + \xi_j\varphi_j(x_t)\delta^2_{\mathrm{TD}}(t), \tag{14}$$

where ξ_j denotes a moving factor for the jth basis function with $\xi_j = \gamma_c\Phi_j(x_t)$ and γ_c an attenuating coefficient.

From the definition of the TD error it is known that the TD error has the characteristic of approaching zero over time when the Actor–Critic learning converges. When the resolution of the state space is insufficient, the variance of the TD error still remains high even though the learning of the value function of partial observa-

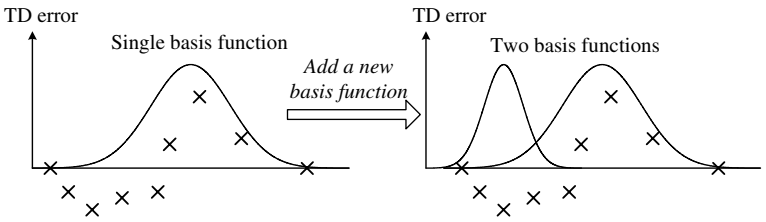

Fig. 3. TD error criterion for adding a new basis function.

tion space converges. Assume this part of the state space is covered by a single basis function and '×' denotes the TD error (see Fig. 3). As seen from the distribution of the TD error in Fig. 3, the average of the TD error approaches zero, while the average squared TD error is still large. That is, this single basis function does not approximate the TD error sufficiently well. Therefore, a new basis function must be added to cover the current input state space in order to obtain a better approximation.

Based on the above analysis, the criterion for adding a new unit is the ratio of $\hbar_j$ and f_j. When $L_j(t) = \frac{\hbar_j(t)}{f_j(t)}$ is larger than a threshold θ_L, a new unit is added. When the environment is stationary, meaning that the state transition probability is constant, the average of the TD error converges at zero and consequently L_j diverges. In order to avoid this problem, we add a rule of arresting the addition of units under the condition that $\hbar_j$ is smaller than a constant θ_g. Thus, the criterion of adding units based on the TD error is:

$$L_j > \theta_L \quad \text{and} \quad \hbar_j > \theta_g. \tag{15}$$

(2) *if-part* criterion

Intuitively, a FIS should always be able to infer an appropriate control action for each state of the control system, which is called the "completeness" property. The completeness property of a fuzzy controller depends on its data base and rule base. From the point of view of the data base, the completeness property largely means that the support of fuzzy sets should cover the related discourse domains with a definite level ε. This property of a FIS is called ε-completeness [14]. Thus, the ε-completeness property ensures that there is always a dominant rule whose belief is greater than ε. However from the point of view of the rule base, the completeness property mainly denotes that an extra rule should be added whenever a fuzzy condition is not included in the rule base, or whenever the match degree (or firing strength) between an input and the predefined fuzzy conditions, is lower than level ε. Otherwise there will be no dominant rule activated in the case of the rule base.

From a statistical point of view, the center and the width of a basis function correspond to the expectation and variance of a distribution. If we assume that the input, i.e., the observed data, obeys a normal distribution, then the probability of the input data located within the range $(\mu_{ij} - 2\sigma_{ij}, \mu_{ij} + 2\sigma_{ij})$ is $P(|x - \mu| > 2\sigma) = 0.4256$. However, the corresponding output of the basis function is only $\varphi(|x - \mu| > 2\sigma) < \exp(-2) = 0.1354$. Thus, the threshold for the output of each unit in the rule layer is obtained as 0.1354 from Eq. (6) and consequently the parameter ε of the corresponding fuzzy rule is set as 0.1354. Therefore, when Eq. (16) is satisfied, meaning that there are no units in the rule layer covering the current input data x_t perfectly, new units should be added to cover the current input data so that the output of the membership function of each input data is larger than 0.1354.

$$\varphi = \arg\max_j \varphi_j(x_t) < 0.1354. \tag{16}$$

If the TD error criterion Eq. (15) and the *if-part* criterion Eq. (16) are established for the current input x_t, the center and the width of the new unit are:

$$\begin{cases} \mu_{\text{new}} = x_t = [x_{1t}, x_{2t}, \ldots, x_{nt}]^{\mathrm{T}} \\ \sigma_{\text{new}} = \tau[\sigma_{1,new}, \sigma_{2,new}, \ldots, \sigma_{n,new}]^{\mathrm{T}} \end{cases} \tag{17}$$

where $\sigma_{i,\text{new}} = \min(|x_{it} - \mu_{ij}|)$ denotes the distance between the ith input state variable and the center of its nearest rule unit within the existing units and τ is an overlapping coefficient that makes the basis functions of adjacent units have proper overlapping regions.

At the initial learning phase, there are no units in the rule layer and the first observed data set is viewed as the first unit. The center of the first rule unit is defined as the first input, while its width can be set as a predefined initial width. For the following input, we can check whether to add units or not according to these two criteria.

3.1.2. Merging units

As the reinforcement learning continues to explore its environment, the number of rule units should increase. It is, therefore, essential to merge any highly similar rule units in order to keep the size of the FRBF network to a minimum without severely degrading its ability to search for the optimal control policy. A fuzzy similarity measure can determine the similarity between two fuzzy sets, but the proposed methods in [15,28] are too complex to compute. Therefore, we propose a similar but more practical method to measure the similarity between two fuzzy sets. The shape and location of the membership function of each rule unit largely depend on the center and the width of the Gaussian function. If the centers and widths of rule units j and p satisfy the relationships in Eq. (18), we can conclude that these two units are similar.

$$\begin{cases} \|\mu_j - \mu_{\mathrm{p}}\| < \Delta\mu_{\min} \\ \|\sigma_j - \sigma_{\mathrm{p}}\| < \Delta\sigma_{\min} \end{cases} \tag{18}$$

where $\Delta\mu_{\min}$ and $\Delta\sigma_{\min}$ denote merging thresholds. For any input variable x_t, the outputs of the rule units j and p meet the condition that

$$\varphi_j(x_t) \approx \varphi_{\mathrm{p}}(x_t). \tag{19}$$

If criterion Eq. (18) is satisfied, the two units should be merged into one and the number of units in both the rule layer and the normalized layer is reduced by one. The weights between the normalized layer and the output layer are set as the following equations after merging:

$$\begin{cases} v = \frac{1}{2}(v_j + v_{\mathrm{p}}), \\ w_k = w_{kj} + w_{kp}. \end{cases} \tag{20}$$

3.2. Parameter learning

The parameter learning of a FACRLN system includes two parts: the Actor parameter learning and the Critic parameter learning. Some parameters, including the connection weights between the normalized layer and the output layer as well as the centers and the widths of the rule units, need to be adjusted on-line.

One feature of the Actor–Critic learning is that the Actor learns the policy function and the Critic learns the value function using the TD method simultaneously. The TD error δ_{TD} is calculated by the temporal difference of the value function between successive states in the state transition.

$$\delta_{\mathrm{TD}}(t) = r_t + \gamma V(x_{t+1}) - V(x_t), \tag{21}$$

where x_t and x_{t+1} denote the states at time t and $(t+1)$, respectively, r_t is the external reinforcement reward signal and γ is the discount factor used to determine the proportion of the delay to the future rewards.

In Eq. (21), the TD error is $\delta_{\mathrm{TD}}(t) = r_t + \gamma V(x_{t+1}) - V(x_t) = r_t - [V(x_t) - \gamma V(x_{t+1})]$. r_t denotes the reinforcement signal with respect to the actual action $A'_k(x_t)$ and $[V(x_t) - \gamma V(x_{t+1})]$ is the reinforcement signal with respect to the recommended action $A_k(x_t)$. The TD error indicates the goodness of the actual action. If $\delta_{\mathrm{TD}}(t) > 0$, the actual action $A'_k(x_t)$ is better than the recommended action $A_k(x_t)$. Therefore, $A_k(x_t)$ should be moved closer to $A'_k(x_t)$. On the other hand, if $\delta_{\mathrm{TD}}(t) < 0$, the actual action $A'_k(x_t)$ is worse than the recommended action $A_k(x_t)$. In that case, $A_k(x_t)$ should be moved further away from $A'_k(x_t)$. The goal of reinforcement learning is to adjust correlated parameters in order to maximize the cumulative sum of the future rewards. The Actor can learn its weights according to Eq. (22) based on the TD approach [15].

$$w_{kj}(t+1) = w_{kj}(t) + \alpha_{\mathrm{A}}\delta_{\mathrm{TD}}(t)\frac{A'_k(x_t) - A_k(x_t)}{\sigma_V(t)}\Phi_j(x_t), \tag{22}$$

where $0 < \alpha_A < 1$ is a learning rate and $\frac{A'_k(x_t)-A_k(x_t)}{\sigma_V(t)}$ the normalized difference between the actual and the recommended actions.

The role of the Critic is to estimate the value function of the policy followed by the Actor, while the TD error is the temporal difference of the value function between successive states in the state transition. Therefore, the goal is to train the Critic to minimize the squared TD error

$$E(t) = \frac{1}{2}\delta_{TD}^2(t). \tag{23}$$

In the following, we will utilize the gradient descent algorithm and the chain rule to update the parameters including the weights v_j, the centers μ_{ij} and the widths σ_{ij} of the rule units of the Critic.

The Critic can learn its weights according to the following equation:

$$v_j(t+1) = v_j(t) - \alpha_C \frac{\partial E(t)}{\partial v_j(t)} = v_j(t) - \alpha_C \frac{\partial E(t)}{\partial \delta_{TD}(t)} \frac{\partial \delta_{TD}(t)}{\partial V(x_t)} \frac{\partial V(x_t)}{\partial v_j(t)} = v_j(t) + \alpha_C \delta_{TD}(t) \Phi_j(x_t) \tag{24}$$

where $0 < \alpha_C < 1$ is a learning rate.

The center updating of the rule unit is:

$$\begin{aligned} \mu_{ij}(t+1) &= \mu_{ij}(t) - \eta_\mu \frac{\partial E(t)}{\partial \mu_{ij}(t)} = \mu_{ij}(t) - \eta_\mu \frac{\partial E(t)}{\partial \delta_{TD}(t)} \frac{\partial \delta_{TD}(t)}{\partial V(x_t)} \frac{\partial V(x_t)}{\partial \Phi_j(x_t)} \frac{\partial \Phi_j(x_t)}{\partial \mu_{ij}(t)} \\ &= \mu_{ij}(t) + \eta_\mu \delta_{TD}(t) v_j(t) \frac{\partial \Phi_j(x_t)}{\partial \mu_{ij}(t)} \end{aligned} \tag{25}$$

where η_μ denotes a learning rate with $0 < \eta_\mu < 1$. $\frac{\partial \Phi_j(x_t)}{\partial \mu_{ij}(t)}$ can be calculated from Eq. (7), to obtain Eq. (26):

$$\frac{\partial \Phi_j(x_t)}{\partial \mu_{ij}(t)} = \left[\frac{\varphi_j(x_t)}{\sum_{j=1}^h \varphi_j(x_t)} - \left(\frac{\varphi_j(x_t)}{\sum_{j=1}^h \varphi_j(x_t)} \right)^2 \right] \left(\frac{x_{it} - \mu_{ij}}{\sigma_{ij}^2} \right) = \Phi_j(x_t)[1 - \Phi_j(x_t)] \left(\frac{x_{it} - \mu_{ij}}{\sigma_{ij}^2} \right). \tag{26}$$

Substituting Eq. (26) into Eq. (25), obtain:

$$\mu_{ij}(t+1) = \mu_{ij}(t) + \eta_\mu \delta_{TD}(t) v_j \Phi_j(x_t)[1 - \Phi_j(x_t)] \left(\frac{x_{it} - \mu_{ij}}{\sigma_{ij}^2} \right). \tag{27}$$

A similar equation is derived for the change in width σ_{ij}.

$$\begin{aligned} \sigma_{ij}(t+1) &= \sigma_{ij}(t) - \eta_\sigma \frac{\partial E(t)}{\partial \sigma_{ij}(t)} = \sigma_{ij}(t) - \eta_\sigma \frac{\partial E(t)}{\partial \delta_{TD}(t)} \frac{\partial \delta_{TD}(t)}{\partial V(x_t)} \frac{\partial V(x_t)}{\partial \Phi_j(x_t)} \frac{\partial \Phi_j(x_t)}{\partial \sigma_{ij}(t)} \\ &= \sigma_{ij}(t) + \eta_\sigma \delta_{TD}(t) v_j(t) \frac{\partial \Phi_j(x_t)}{\partial \sigma_{ij}(t)} = \sigma_{ij}(t) + \eta_\sigma \delta_{TD}(t) v_j(t) \Phi_j(x_t)[1 - \Phi_j(x_t)] \frac{(x_{it} - \mu_{ij})^2}{\sigma_{ij}^3}, \end{aligned} \tag{28}$$

where $0 < \eta_\sigma < 1$ is a learning rate.

3.3. Algorithm description

Based on the above analysis, the entire set of design steps of the fuzzy Actor–Critic reinforcement learning method can be summarized as follows:

1. Obtain the state x_t at time t.
2. Calculate the normalized firing strength of each fuzzy rule from Eq. (7).
3. Calculate the action output $A_k(x_t)$ and the Critic value function $V(x_t)$ from Eqs. (8) and (9), respectively.
4. Calculate the actual action according to Eq. (10) and execute it, receive the reward r_t and then let the state transit from x_t to x_{t+1}.
5. Calculate the new Critic value function $V(x_{t+1})$ from Eq. (9) and the TD error $\delta_{TD}(t)$ from Eq. (21).

6. Check whether to add a rule unit or not according to Eqs. (15) and (16). If the criteria are met, add a new unit whose parameters are set as Eq. (17), otherwise turn to the next step.
7. Adjust the weights w_{kj} and v_j from Eqs. (22) and (24), respectively.
8. Adjust the centers and the widths according to Eqs. (27) and (28) respectively.
9. Check whether to merge units or not according to Eq. (18). If the criterion is met, merge similar units according to Eq. (20), otherwise turn to the next step.
10. Determine whether the learning process is finished or not. If not, then $t \leftarrow t+1$ and return to step 1 to start the next cycle of the fuzzy Actor–Critic reinforcement learning method.

4. Experimental studies

In order to assess the learning performance of FACRLN and compare it with other reinforcement learning methods, a typical example of the cart–pole balancing problem or the so-called inverted pendulum balancing problem is presented in this section.

As shown in Fig. 4, a rigid pole is freely hinged on the top of a wheeled cart that is free to move on a one-dimensional track of fixed length. Both the cart and pole can move only in the vertical plane, i.e., each has only one degree of freedom [15,16]. The cart–pole balancing problem is a problem of learning how to balance the upright pole.

There are four input state variables to describe the cart–pole system: θ, the angle of the pole with the vertical (in degree); $\dot{\theta}$, the angular velocity of the pole (in degree/s); χ, the horizontal position of the center of the cart (in m) and $\dot{\chi}$, the velocity of the cart (in m/s). The only control action is F, the amount of force (in N) applied to the cart to move it towards the left or right. Therefore, the input variables of the FRBF network to control the system are the four state variables described above, while the outputs of the FRBF network are the control action F and the Critic value V. The system fails when the pole falls past a certain angle or the cart runs into the bounds of its track. The dynamic model of the cart–pole system is described by the following nonlinear equations:

$$\begin{cases} \ddot{\theta}_t = \dfrac{g\sin\theta_t + \cos\theta_t\left[\frac{-F_t - m_p l\dot{\theta}_t^2\sin\theta_t + \mu_c \mathrm{sgn}(\dot{\chi}_t)}{m_c+m_p}\right] - \frac{\mu_p\dot{\theta}_t}{m_p l}}{l\left[\frac{4}{3} - \frac{m_p\cos^2\theta_t}{m_c+m_p}\right]}, \\ \ddot{\chi}_t = \dfrac{F_t + m_p l\left[\dot{\theta}_t^2\sin\theta_t - \ddot{\theta}_t\cos\theta_t\right] - \mu_c\mathrm{sgn}(\chi_t)}{m_c+m_p}. \end{cases} \tag{29}$$

The parameters of the studied cart–pole system are the same as those used in Refs. [2,1,4,15,16,11]. The gravity acceleration $g = -9.8\ \mathrm{m/s^2}$, the mass of the cart $m_c = 1.0$ kg, the mass of the pole $m_p = 0.1$ kg, the half-pole length $l = 0.5$ m, the friction coefficient of cart on the track $\mu_c = 0.0005$, the friction coefficient of pole on the cart $\mu_p = 0.000,002$ and the control force $F_t = [-10, +10]$ N. The sampling period $T_s = 0.02$ s during the experiment. The pole must be kept within $\pm 12°$ from the vertical and the cart must be kept within ± 2.4 m from the center of the track. The only feedback signal received by the FACRLN controller from the external environment is a failure signal. When the angle of the pole exceeds the range of $[-12°, +12°]$ or the cart collides with the end of the track at the position of -2.4 m or $+2.4$ m, the external environment will issue a failure signal. Hence, the reward r_t is defined as follows:

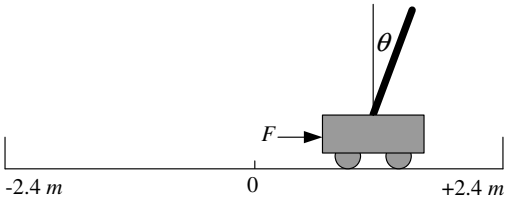

Fig. 4. Cart–pole balancing system.

$$r_t = \begin{cases} 0, & |\theta_t| \leqslant 12^\circ \quad \text{and} \quad |\chi_t| \leqslant 2.4\ \text{m}, \\ -1, & \text{otherwise.} \end{cases} \tag{30}$$

The goal of this control problem was to train the FACRLN such that it could determine a sequence of forces applied to the cart to balance the pole as long as possible without failure. The control strategy was deemed successful if it could balance the pole for 50 000 steps within one trial. In order to acquire control experience from various situations, the FACRLN controller started to learn from a stochastic initial state until a control failure occurred and restarted to learn after each failure.

When we designed the FACRLN controller, some parameters had to be predefined based on the following selection principles. First, by adjusting the discount factor γ, we were able to control the extent to which the learning system is concerned with long-term versus short-term consequences of its own actions. In the limit, when $\gamma = 0$, the system is myopic in the sense that it is only concerned with immediate consequences of its action. As γ approaches 1, future costs become more important in determining optimal actions. Because we were rather concerned with long-term consequences of its actions, the discount factor had to be large, i.e., we set $\gamma = 0.98$. Second, if a learning rate is small, the learning speed is slow. In contrast, if a learning rate is large, the learning speed is fast, but oscillation occurs easily. Therefore, we chose small learning rates to avoid oscillation, such as $\alpha_A = 0.1$, $\alpha_C = 0.01$, $\eta_\mu = 0.055$ and $\eta_\sigma = 0.035$, which, in fact, we used. Third, γ_C, θ_L, θ_g and τ are used to check whether to add a rule unit or not which should meet the following conditions: $0 < \gamma_C < 1$, $\theta_L > 1$, $0 < \theta_g < 1$ and $0 < \tau < 1$. Fourth, in order to ensure that two rule units with high similarity are merged, merging thresholds $\Delta\mu_{\min}$ and $\Delta\sigma_{\min}$ should be rather small; we used $\Delta\mu_{\min} = 0.01$ and $\Delta\sigma_{\min} = 0.01$. Based on these selection principles, the parameters for the FACRLN controller in the experiment are set as shown in Table 1. It is to be noted that the choice of these parameters is not unique. Other choices are possible, as long as the above selection principles are followed. Clearly different parameters will lead to different experimental results. In order to make these parameters optimal, some optimal methods, such as a genetic algorithm or evolutionary computation, may be applied in the future.

Table 1
Parameters for the FACRLN controller

γ	α_A	α_C	γ_C	θ_L	θ_g	τ	$\Delta\mu_{\min}$	$\Delta\sigma_{\min}$	η_μ	η_σ
0.98	0.1	0.01	0.37	3.0	0.01	0.87	0.01	0.01	0.055	0.035

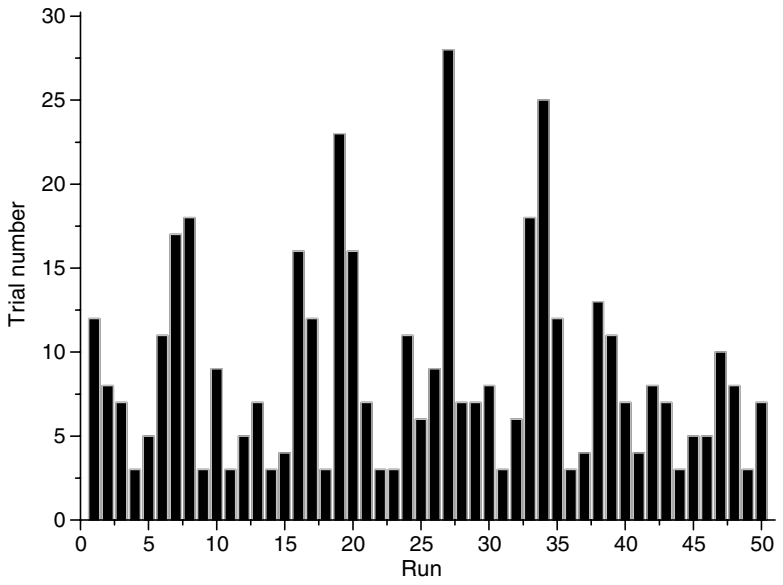

Fig. 5. Trial number required to find a successful FACRLN for each run.

In this example, 50 runs were performed and a run ended when a successful controller was found or a failure run occurred. A failure run is said to occur if no successful controller is found after 500 trials. The number of pole balance trials was measured for each run and their statistical results are shown in Figs. 5 and 6. The minimum and the maximum number of trials over these 50 runs are 3 and 28, while the average number of trials is 8.68. It is clear from Fig. 6 that there are 16 runs whose trials are within the range of 6 and 10. One of the successful results is shown in Figs. 7–10, where the learning performance of the FACRLN on the cart–pole balancing problem pole (Fig. 7), the cart position χ (Fig. 8), the pole angle θ (Fig. 9) and the control force F (Fig. 10) are displayed. The FACRLN could balance the pole for 50 000 steps after four trials.

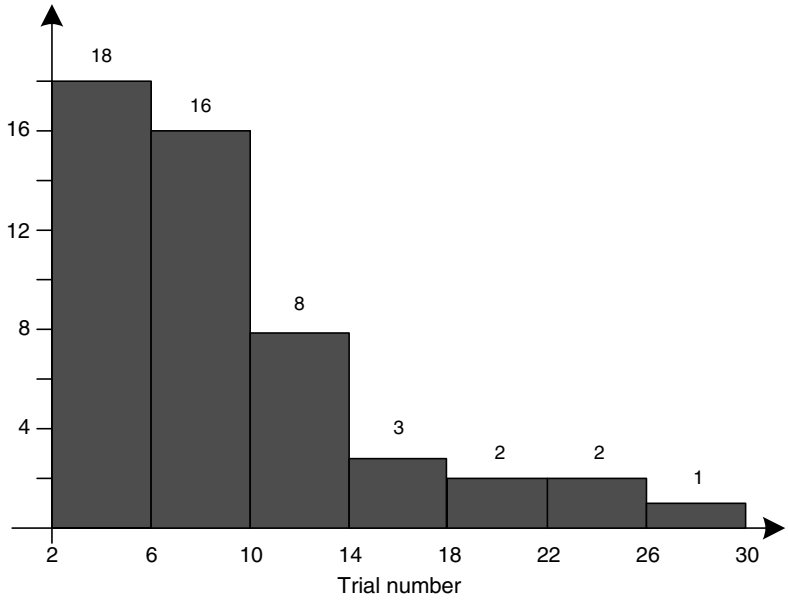

Fig. 6. Statistical result of the trials required to find a successful FACRLN for each run.

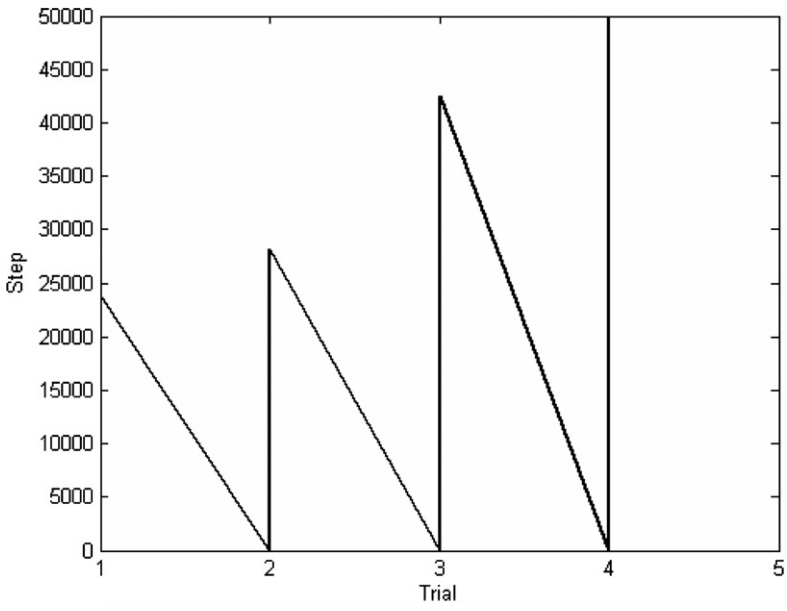

Fig. 7. Performance of the FACRLN on the cart–pole balancing problem.

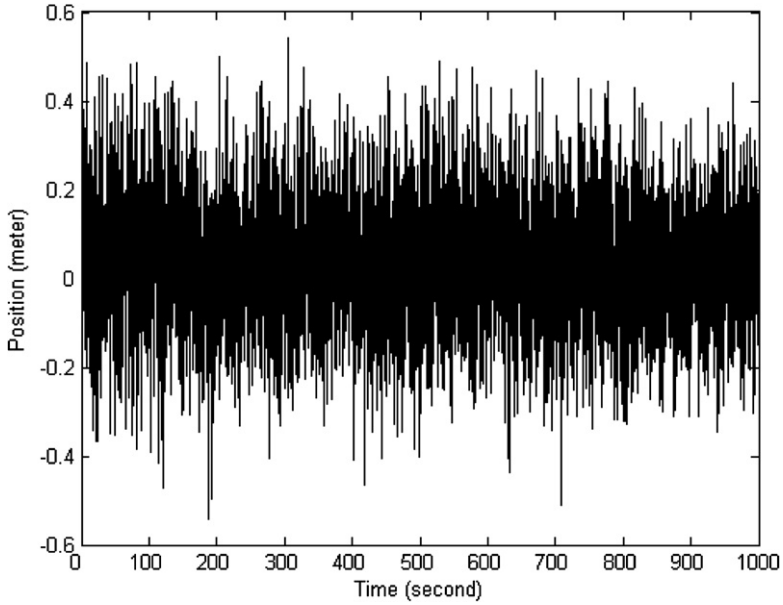

Fig. 8. Position of the cart.

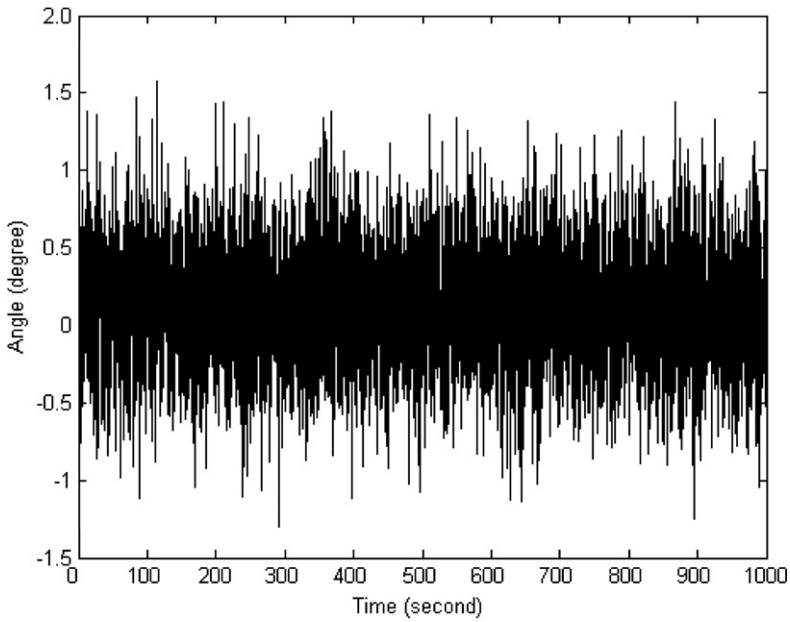

Fig. 9. Angle of the pole.

From the results, we see that only small deviations on both position and angle are left via learning. The average position and angle deviations are 0.6 m and 1.5^0.

The exponential moving average of the weighted squared TD error $\hbar_j$ and the ratio of the moving average of the weighted squared TD error to TD error L_j are shown in Figs. 11 and 12. The growth of the rule units is presented in Fig. 13 based on the TD error and the *if-part* criteria. The abscissa of Figs. 11–13 indicate the time of this particular run. It is clear from Figs. 11 and 12 that $\hbar_j > \theta_g = 0.01$ and $L_j > \theta_L = 3$ at the initial learning phase. Obviously, there are few rule units in the rule layer at the initial learning phase, which means

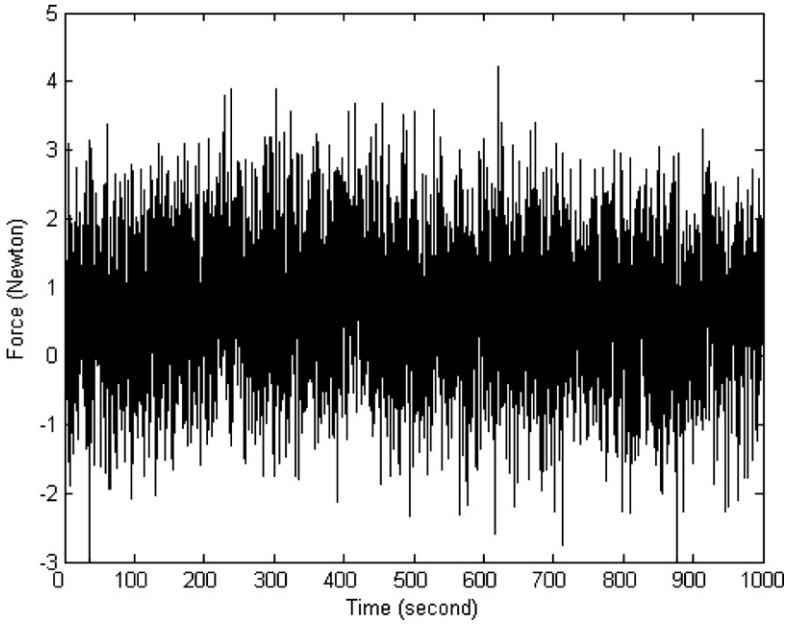

Fig. 10. Control force.

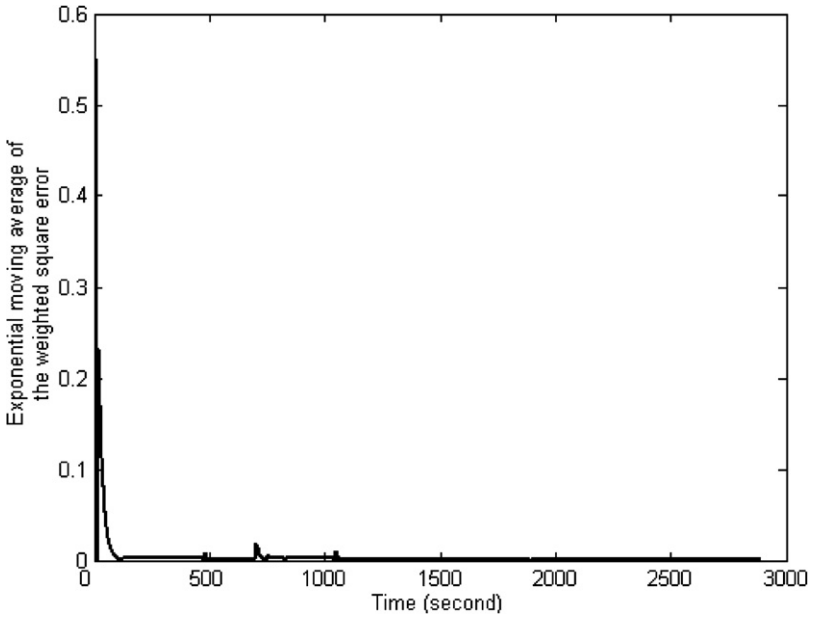

Fig. 11. Exponential moving average of the weighted squared TD error.

that no rule unit can cover the input vectors, i.e., $\varphi = \arg\max_j \varphi_j(x_t) < 0.1354$. Therefore, the growing speed of the rule units is high. The number of rule units quickly increased from 1 to 4 at the initial learning phase. Because the squared TD error converges at zero (i.e., $\hbar_j < \theta_g = 0.01$) along with the progress in learning, the number of rule units is fixed at 11 although $L_j > \theta_L = 3$ after 1650 s.

We now compare the performance of our system with that of other existing AC reinforcement learning systems. The performance indices considered are continuous states and actions, a priori knowledge, number of

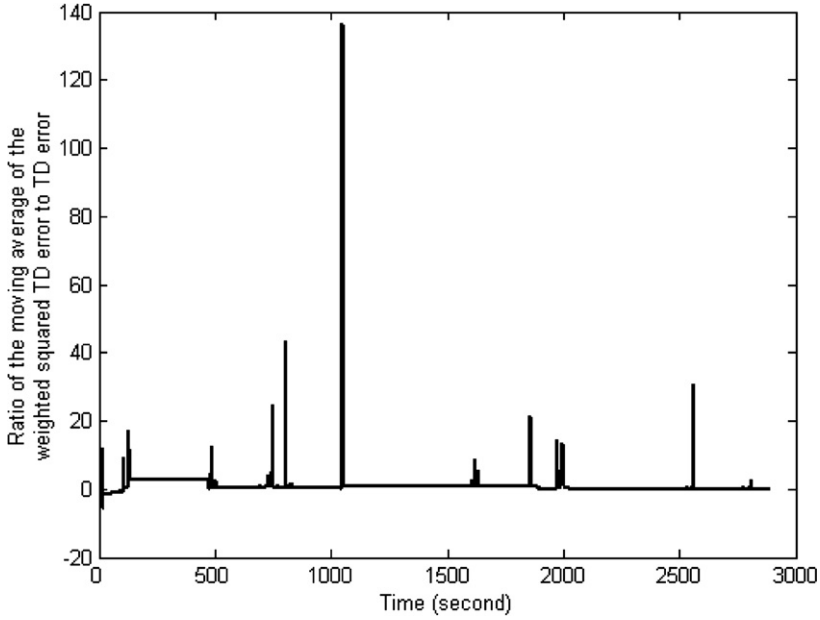

Fig. 12. Ratio of the moving average of the weighted squared TD error to TD error.

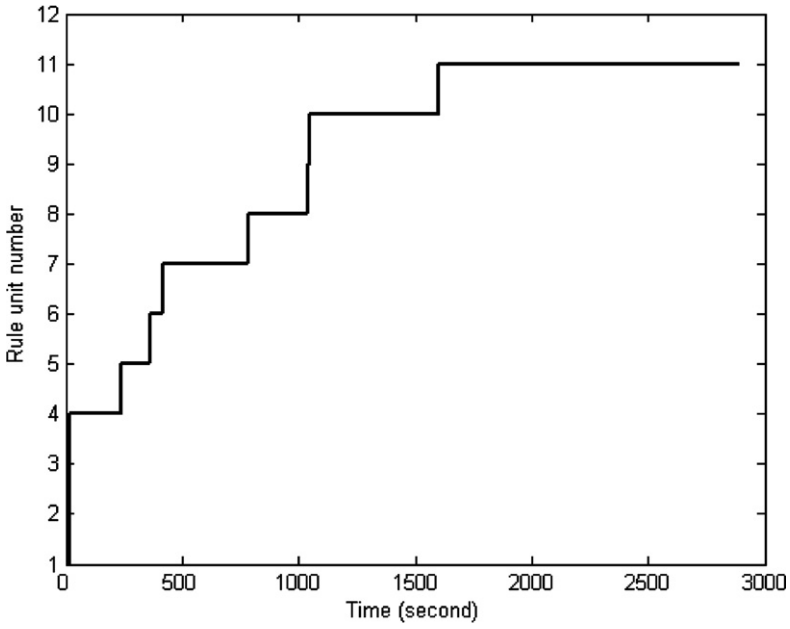

Fig. 13. Growth of the rule units.

fuzzy rules, structure learning and number of trials. The detailed comparison is presented in Table 2, from which we see that our FACRLN system required the smallest number of trials. The number and configuration of membership functions for each input and output variable do not need to be decided by an expert in advance in our FACRLN system, i.e., a priori knowledge is not necessary. Moreover, two additional sets of experiments were performed to verify the adaptation capability of the FACRLN system. In the first set, we varied

Table 2
Performance comparison of various Actor–Critic reinforcement learning systems

	Continuous states	Continuous actions	A priori knowledge	Number of fuzzy rules	Structure learning	Number of trials
AHC [2]	No	No	No		No	35
Anderson [1]	Yes	No	No		No	8000
GARIC [4]	Yes	Yes	Yes	13	No	300
RNN-FLCS [15]	Yes	Yes	Yes	35	Yes	10
FALCON-RL [16]	Yes	Yes	Yes	10	Yes	15
FACL [11]	Yes	Yes	No	54	Yes	13
FACRLN	Yes	Yes	No	11	Yes	8.68

the length of the pole from 0.5 m to 0.25 m. In another set, we doubled the mass of the cart to 2 kg from 1 kg. We found that the proposed FACRLN system was able to complete the control task of balancing the pole.

5. Conclusions

Reinforcement learning has the ability to make use of evaluative feedback from the environment to guide the learning process. It has been widely applied to the application of planning, control and decision making. One of the difficulties encountered in the application of reinforcement learning methods to real-world problems with large-scale or continuous spaces is the well known 'curse of dimensionality'. Thus, in order to apply reinforcement learning to complicated systems, based on the analysis of the functional equivalence between a RBF network and a fuzzy inference system, a new fuzzy Actor–Critic reinforcement learning network based on a four-layer fuzzy RBF neural network has been proposed in our investigation, which takes full advantage of the knowledge representing property of the FIS and the self-learning property of the RBF network.

Because the FRBF neural network is able to approximate both the action function of the Actor and the value function of the Critic simultaneously, this routine can reduce the demand for storage space from the learning system and avoid repeated computations for the outputs of the rule units. Therefore, the Actor–Critic reinforcement learning system based on the four-layer FRBF is much simpler than other reinforcement learning systems realized by two separate neural networks.

Furthermore, the FRBF network has the ability of adding and merging rule units dynamically with a novel self-organizing approach according to the complexity of the task and the progress in learning. At the initial learning phase, there are no units in the rule layer and the first input is viewed as the first unit. For the following input, we can check whether to add units or not according to the TD error and the *if-part* criteria. As the reinforcement learning continues to explore its environment, it is essential to merge any highly similar rule units according to a fuzzy similarity measure criterion in order to keep the size of the FRBF network to a minimum.

Experimental studies concerning the cart–pole balancing control demonstrate the validity and performance of the proposed FACRLN, with its advantages for generalization, a simple control structure and a high learning efficiency. The disadvantage of the FACRLN is that the choice of control parameters should be determined beforehand according to experience and some guiding principles. In order to make these parameters optimal, some optimal methods, such as a genetic algorithm or evolutionary computation, may be applied in the future. As well, future work also will focus on applying the FACRLN to practical problems in the real world, for example to mobile robot navigation.

Acknowledgements

The authors would like to acknowledge the reviewers for their helpful comments and suggestions on earlier drafts of the paper.

This research is financially supported by the Science Foundation for Post-doctoral Scientists of Jiangsu Province (Grant No.: 0601033B), the 'Qing-Lan' Program for Young Excellent Teachers of Jiangsu Province, and the Scientific and Technological Foundation for Youth of the China University of Mining and Technology (Grant Nos.: 0C4466, 0C060093).

References

[1] C.W. Anderson, Learning to control an inverted pendulum using neural networks, IEEE Control System Magazine 9 (3) (1989) 31–37.
[2] A.G. Barto, R.S. Sutton, C.W. Anderson, Neuron like elements that can solve difficult learning control problems, IEEE Transactions on System, Man and Cybernetics 13 (5) (1983) 834–846.
[3] H.R. Berenji, A reinforcement learning-based architecture for fuzzy logic control, International Journal of Approximate Reasoning 6 (2) (1992) 267–292.
[4] H.R. Berenji, P. Khedkar, Learning and tuning fuzzy logic controllers through reinforcements, IEEE Transactions on Neural Networks 3 (5) (1992) 724–740.
[5] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, Time series analysis, forecasting and control, Prentice-Hall Publisher, Inc., 1994.
[6] J.I. Canelon, L.S. Shieh, N.B. Karayiannis, A new approach for neural control of nonlinear discrete dynamic systems, Information Sciences 174 (3–4) (2005) 177–196.
[7] K.S. Hwang, S.W. Tan, M.C. Tsai, Reinforcement learning to adaptive control of nonlinear systems, IEEE Transactions on Systems, Man and Cybernetics 33 (3) (2003) 514–521.
[8] J.S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Transactions on Systems, Man and Cybernetics 23 (3) (1993) 665–685.
[9] J.S.R. Jang, C.T. Sun, Functional equivalence between radial basis function networks and fuzzy inference systems, IEEE Transactions on Neural Networks 4 (1) (1993) 156–159.
[10] Y.C. Jin, B. Sendhoff, Extracting interpretable fuzzy rules from RBF Networks, Neural Processing Letters 17 (2) (2003) 149–164.
[11] L. Jouffe, Fuzzy inference system learning by reinforcement learning, IEEE Transactions on Systems, Man and Cybernetics 28 (3) (1998) 338–355.
[12] H.B. Kazemian, M. Li, A fuzzy control scheme for video transmission in Bluetooth wireless, Information Sciences 176 (9) (2006) 1266–1289.
[13] T. Kondo, K. Ito, A reinforcement learning with evolutionary state recruitment strategy for autonomous mobile robots control, Robotics and Autonomous Systems 46 (2) (2004) 111–124.
[14] C.C. Lee, Fuzzy logic in control systems: fuzzy logic controller – Part I and II, IEEE Transactions Systems, Man and Cybernetics 20 (2) (1990) 404–435.
[15] C.T. Lin, C.S.G. Lee, Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems, IEEE Transactions on Fuzzy Systems 2 (1) (1994) 41–63.
[16] C.J. Lin, C.T. Lin, Reinforcement learning for an ART-based fuzzy adaptive learning control network, IEEE Transactions on Neural Networks 7 (3) (1996) 709–731.
[17] C.T. Lin, C.J. Lin, C.S.G. Lee, Fuzzy adaptive learning control network with on-line neural learning, Fuzzy Sets and Systems 71 (1) (1995) 25–45.
[18] J.B. Mbede, P. Ele, C.M.M. Abia, Y. Toure, V. Graefe, S.G. Ma, Intelligent mobile manipulator navigation using adaptive neuro-fuzzy systems, Information Sciences 171 (4) (2005) 447–474.
[19] G. Meghabghab, A. Kandel, Stochastic simulations of web search engines: RBF versus second-order regression models, Information Sciences 159 (1–2) (2004) 1–28.
[20] P. Melin, O. Castillo, Intelligent control of a stepping motor drive using an adaptive neuro-fuzzy inference system, Information Sciences 170 (2–4) (2005) 133–151.
[21] J. Moody, C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Computation 1 (2) (1989) 281–294.
[22] P. Preux, S. Delepoulle, J.C. Darcheville, A generic architecture for adaptive agents based on reinforcement learning, Information Sciences 161 (1–2) (2004) 37–55.
[23] K. Samejima, T. Omori, Adaptive internal state space construction method for reinforcement learning of a real-world agent, Neural Networks 12 (7–8) (1999) 1143–1155.
[24] B. Ster, A. Dobnikar, Adaptive radial basis function decomposition by learning vector quantization, Neural Processing Letters 18 (1) (2003) 17–27.
[25] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Transactions on Systems, Man and Cybernetics 15 (1) (1985) 116–132.
[26] C.S. Ting, Stability analysis and design of Takagi–Sugeno fuzzy systems, Information Sciences 176 (19) (2006) 2817–2845.
[27] L.X. Wang, J.M. Mendel, Fuzzy basis functions, universal approximation, and orthogonal least squares, IEEE Transactions on Neural Networks 3 (5) (1992) 807–814.
[28] W.J. Wang, New similarity measures on fuzzy sets and on elements, Fuzzy Sets and Systems 85 (3) (1997) 305–309.
[29] J.Q. Yi, N. Yubazaki, K. Hirota, Anti-swing and positioning control of overhead traveling crane, Information Sciences 155 (1–2) (2003) 19–42.
[30] C.J. Zhou, Robot learning with GA-based fuzzy reinforcement learning agents, Information Sciences 145 (1–2) (2002) 45–68.