Journal of Uncertain Systems
Vol.2, No.1, pp.54-66, 2008
Online at: www.jus.org.uk

Reinforcement Learning for Fuzzy Control with Linguistic States

Mohammad Hossein Fazel Zarandi[1,*], Javid Jouzdani[1], Maryam Fazel Zarandi[2]
[1] *Department of Industrial Engineering,*
AmirKabir University of Technology, Tehran, Iran
P.O. Box: 15875-4413
[2] *Department of Computer Science*
10 King's College Road, University of Toronto, Toronto, ON., Canada M5S3G4
zarandi@aut.ac.ir javid@aut.ac.ir mfazel@cs.toronto.edu

Received 9 September 2007; Accepted 12 October 2007

Abstract

This paper presents a generalized reinforcement learning methodology for tuning fuzzy logic controllers with nonlinear dynamic behaviors. To this aim, the Generalized Approximate Reasoning-Based Intelligent Controller (GARIC) model in [3] is modified to handle vagueness in control states. The proposed architecture has a self-tuning capability even when only a weak reinforcement signal such as binary failure signal is available. The controller is tested and validated by the well-know Cart-Pole control problem. Compared to similar models, the proposed controller exhibits a better performance with regards to the learning speed and robustness to changes in controlled system dynamics, even in the presence of uncertainty in the states obtained.

Keywords: fuzzy control, fuzzy reinforcement learning, fuzzy system modeling

1 Introduction

Fuzzy logic is proved to be effective in solving many nonlinear control problems, where the nonlinear behavior of the system makes it difficult, if not impossible, to build an analytical model of the system. Nevertheless, building a fuzzy controller has its own difficulties that should be resolved through the implementation of suitable techniques.

There are two approaches to the development of a fuzzy model. The first approach is based on describing the rules governing the system linguistically using terms of natural language, and then transforming them into fuzzy rules. In this approach, which is referred to as the direct approach to fuzzy modeling, the linguistic descriptions are constructed subjectively according to prior knowledge of the system. This makes the process highly dependent on the expert's knowledge, and if the expert's knowledge about the system is faulty this would result in developing an un-robust model of the system. Thus, the rules should be fine-tuned in order to be used for control purposes.

The second approach uses input-output data in the development of the fuzzy model, and is called the indirect approach to fuzzy modeling. The problem of extracting fuzzy rules from data arose in the early years after the birth of fuzzy modeling concepts. Since in this approach the fuzzy rules are constructed based on data, if the data is faulty, damaged, or noisy, the obtained rules may not be reliable; i.e., crude rules may be obtained that need to be fine-tuned.

When input-output training data are available, supervised learning techniques perform well on the task of tuning the controller. However, when such data are not available, unsupervised methods such as reinforcement learning can be used to solve the problem. In reinforcement learning, it is assumed that the equations describing the system are not well-known to the controller and the only information available are the states of the system and a reinforcement signal evaluating the performance via a failure or success signal. The controller is expected to learn the best policy through a trial-and-error interaction with the dynamic environment.

Reinforcement learning can be used to fine-tune a fuzzy logic controller; either for structure identification (e.g., [1,2]) or for parameter identification (e.g., [3,4]). This paper concentrates on the latter issue. A number of previous studies exist in the literature. In [9], Mustapha and Lachiver present a model called the Generalized Reinforcement

* Corresponding author. Email: zarandi@aut.ac.ir (M. H. F. Zarandi)

Learning Fuzzy Controller (GRLFC), which is similar to the model proposed in this paper. The architecture proposed by Berenji and Khedkar [4], called the Generalized Approximate-Reasoning-Based Intelligent Controller (GARIC) is the main inspiration for generating GRLFC. However, one shortcoming of their model is that it needs a large number of trials to be tuned. Seo *et al.* [10] notice and model vagueness both in states and goals, but they ignore the generalization issue. Feng [11] introduces a model in which the learning rates are variable. However, the vagueness in the states is ignored. In [6], Jouffe proposes a good model which uses eligibility traces to enhance the speed of learning. In his work, he investigates the continuous and discrete actions.

In this paper we extend the GARIC architecture with respect to various aspects. The vagueness in the input states are modeled by adding a new component called *Fuzzifier*. In addition, learning mechanism in *Critic* is different from that of Action Evaluation Network which plays the role of the critic in GARIC. Furthermore, variable learning rates are used for updating the parameters of the *Actor* and the *Critic*. This increases the speed of learning that leads to a satisfactory performance of GRLFC, even when vagueness exists in the input variables. Also, *Explorer* in GRLFC is an extension of the Explorer in GARIC and uses the Episode Information Feedback from the plant to perform a better exploration and exploitation. GRLFC not only captures the vagueness in the input states, but also has a superior performance in comparison with similar GARIC models.

The rest of this paper is organized as follows: In Section 2, the fundamentals of the fuzzy inference system that is used in this paper are reviewed. Section 3 presents a brief review of reinforcement learning and its relation with fuzzy inference systems. The proposed Generalized Reinforcement Learning Fuzzy Controller (GRLFC) is discussed in Section 4. In Section 5, simulation results are presented to show the efficiency and superiority of the proposed model in comparison with similar models like GARIC. Finally, Section 6 concludes the paper with a discussion of the contributions and areas of future work.

2 Fuzzy Inference System

A fuzzy inference system (FIS) can be defined as a set of IF-THEN rules that maps an input space onto an output space. Therefore, it can be considered as a method for generalization and functional approximation.

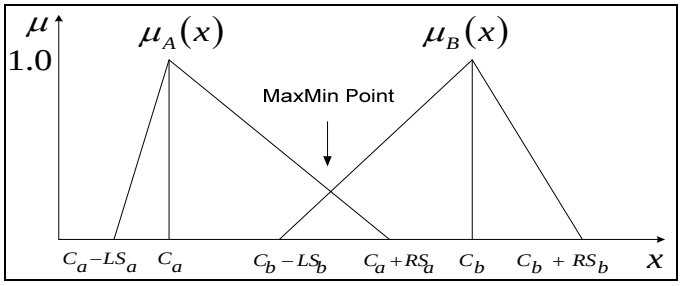


Fig.1. Triangular membership functions, their parameters and the MaxMin point determining the degree of matching.

The input variables of an FIS are usually considered to have crisp values. However, in most real-world situations, input variables (states) are vague. The proposed FIS can capture the vagueness in the input variables by considering them as triangular fuzzy numbers (Fig. 1).

In this research triangular membership functions are used in both antecedent and consequent parts of the FIS, for the sake of simplicity. These membership functions are determined by three parameters: centre (C), right spread (RS), and left spread (LS). We can use any kind of matching operators (t-conorm/t-norm) to calculate the degree of matching of input variables and their corresponding antecedents. However, we use the MaxMin operator since it is more widely used. For future work, we will try other operators to justify and select the most suitable ones for our problem. Of course, if we use parametric t-conorms and t-norms, we should add an optimization module to tune the prameters of t-conorm/t-norm which are used for calculating the degree of matching.

Under the above assumptions, the degree of matching between an input variable and its corresponding antecedent label can be easily calculated by using the MaxMin operator:

$$MaxMin(A,B)=\begin{cases}1, & if\ RS_a=\infty\ or\ LS_b=\infty\\ 0, & if\ RS_a=0\ and\ LS_b=0\\ \psi\left(\dfrac{C_a-C_b+RS_a+LS_b}{RS_a+LS_b}\right), & otherwise\end{cases} \tag{1}$$

where ψ is a function defined as

$$\psi(x)=\begin{cases}0, & if\ x\le 0\\ x, & if\ x>0\end{cases} \tag{2}$$

The degree of applicability of a rule can be determined by applying a t-Norm to the degrees of matching between each input variable of the rule and its corresponding antecedent labels. GRLFC inherits its t-Norm from GARIC in which Softmin, defined in equation (3), is used as the t-Norm. In addition, we assume that $k=10$ [3]. Since we do not tune the antecedent labels and we do not need its differentiability, any other kind of t-Norm would also be appropriate for our system. However, the results should be validated and justified based on the situation of the problem domain. Using parametric t-conorm and t-norm would also need a new optimization module to tune the parameters of the union and intersection operators. We usually use Neural Networks together with an evolutionary algorithm for this purpose. For future work, we will use other kinds of t-conorm and t-norm, and compare the results obtained. We consider this to be very necessary in developing adaptive systems.

$$SoftMin(x_1,x_2,\ldots,x_n)=\frac{\sum_{i=1}^{n}x_i\exp(-kx_i)}{\sum_{i=1}^{n}\exp(-kx_i)}. \tag{3}$$

Thus, using the Softmin operator and degrees of matching, the degree of applicability or the degree of firing of rule R_i can be calculated by

$$w_i = Soft\ Min\left(MaxMin\ (A_1,B_{i1}),\ldots,MaxMin\ (A_n,B_{in})\right) \tag{4}$$

where w_i is the degree of firing of rule R_i, A_j is the j^{th} input variable, and B_{ij} is the j^{th} antecedent label in the i^{th} rule, R_i. By applying the degree of firing of R_i to its consequent part, the output of the rule can be calculated. The defuzzified output is calculated by

$$z_i=\mu_{C_i}^{-1}(w_i) \tag{5}$$

where μ^{-1} is a defuzzification method, C_i is the consequent label of the i^{th} rule, and z_i is the defuzzified output for the i^{th} rule. In this paper, we use the Local Mean of Maximum (LMOM) method for defuzzification [3].

Combining the outputs of all rules, a crisp control action in the form of weighted average is obtained, using the following equation

$$F=\frac{\sum_{i=1}^{m}z_i w_i}{\sum_{i=1}^{m}w_i} \tag{6}$$

where m is the number of the rules. This can also be extended for multiple output variables.

3 Reinforcement Learning

Similar to many other techniques in the field of artificial intelligence, reinforcement learning (RL) has its roots in psychology. The idea behind RL is learning from experience and through trial-and-error interactions with a dynamic environment, similar to what any intelligent creature would do during its lifetime.

In many tasks to which we would like to apply reinforcement learning, most states encountered have never been exactly experienced before. This is almost always the case when the state or action spaces include continuous variables or complex and vague sensations. Therefore, a generalization method is needed; more specifically, the kind of generalization we require is often called *function approximation* because it takes examples from a desired function (e.g., a value function) and attempts to generalize to construct an approximation of the entire function.

Fuzzy Inference Systems (FIS) are appropriate tools for generalization. The use of FIS as opposed to a global function approximator like Neural Networks has two major advantages: 1) the FIS's inherent locality property permits the introduction of human knowledge, and 2) localizes the learning process to only implicated parameters [5]. However, the process of fine-tuning the fuzzy controller still remains a difficult task.

It should be noted that supervised control learning requires training data or a teacher of the subject domain. In most real-world applications, training data is often hard to obtain or may not be available at all. An approach to solving this problem is based on reinforcement learning, a paradigm that stems from the desire to make systems that learn from autonomous interactions with their environments. Therefore, reinforcement learning techniques can be effective for the fine-tuning of fuzzy controllers when no training data is available and only a weak reinforcement can be obtained.

This area of research has attracted many researchers. Some have applied RL techniques to tune a conventional Neural Network as the critic while the actor is an FIS [3,7]. Others have used Fuzzy Inference Systems to increase the knowledge of the critic about the goodness of the states and consequently enhance the performance of the system [4,5]. However, not many researchers have considered the vagueness in input states. In this paper, we propose a model that captures the uncertainty in the state of the system. In addition, the proposed model demonstrates a superior performance in comparison with similar models even in presence of uncertainty. The proposed GRLFC is tested and validated by several test cases.

4 The Architecture of GRLFC

In the proposed model, a fuzzy controller is implemented in the form of an FIS which plays the role of the *Actor*. The Actor implements the knowledge of the expert operator about how to control the system. The *Critic* which evaluates the value of the current state is another FIS, and it incorporates the knowledge about the goodness of the states of the plant. Both of these components simultaneously learn to improve their performance through interaction with a dynamic environment and by receiving a reinforcement signal.

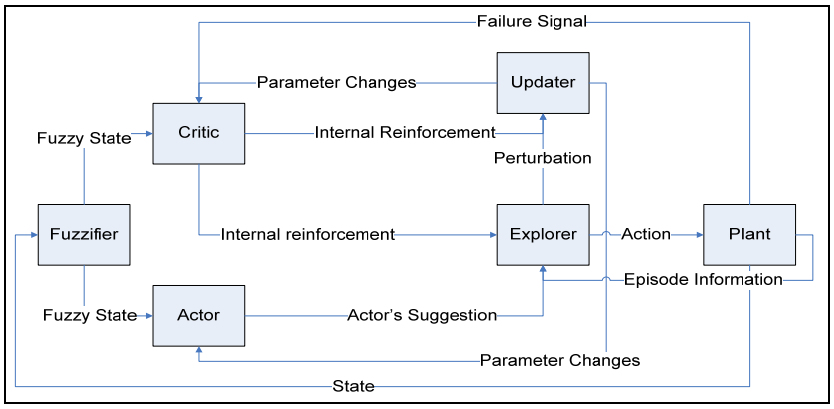


Fig. 1. The architecture of GRLFC.

The architecture of GRLFC is shown in Fig. 2. The system has five components: *Fuzzifier*, *Actor*, *Critic*, *Explorer* and *Updater*. Current state of the plant is fed into the *Fuzzifier* which captures the vagueness of that state. The *Actor* uses this fuzzy state to determine the action and the *Critic* evaluates its value. Combining the value determined by the *Critic* and the reinforcement signal, an internal reinforcement is generated which is used for fine-tuning both the *Actor* and the *Critic*. Learning in both *Actor* and *Critic* is through tuning of parameters of the consequent part labels. *Explorer* perturbs the action suggested by the *Actor* in order to provide a better search of the state space. To accomplish this task, the *Explorer* uses current episode information, the internal reinforcement, and the action suggested by the *Actor*. In what follows, we present each component in more detail.

4.1 Fuzzifier

As already mentioned, in many real-world problems, uncertainty exists in the states of a system. This can be caused by many factors like uncertainty in sensor readings or uncertain nature of the states of the system (linguistic input states).

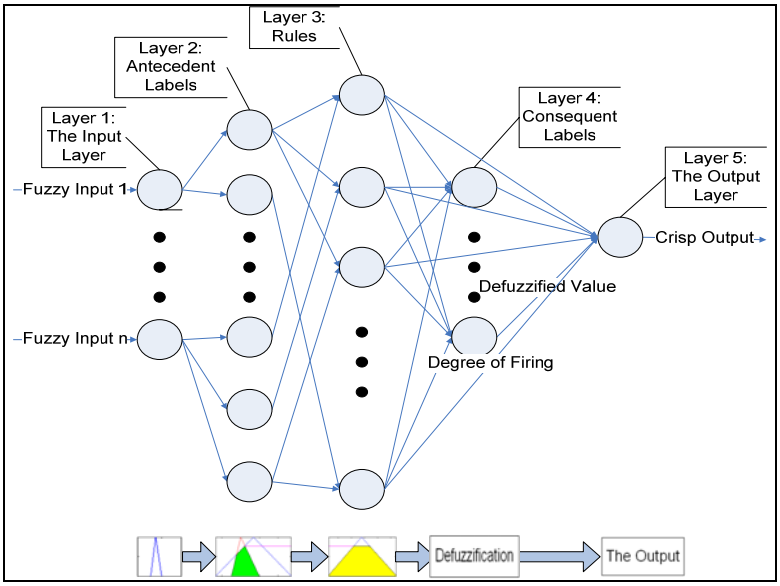


Fig. 2. FIS scheme

The *Fuzzifier* considers the uncertainty in the input variables by constructing a symmetric or asymmetric triangular fuzzy membership function using the crisp input states. In other words, the crisp input makes the centre of the triangular membership function and the spreads are determined by the *Fuzzifier* according to the specifications of the problem. Thus, the shape of the membership function constructed is determined by the nature and amount of uncertainty in the input states.

4.2 Actor and Critic

Both the *Actor* and the *Critic* are fuzzy inference systems described in Section 2, and have similar architectures depicted in Fig. 3. In this scheme, the first layer is the input layer which can accept triangular fuzzy membership functions as fuzzy states. Layer 2 contains the antecedent labels and determines the degree of matching using the MaxMin operator. In layer 3 the degree of firing of the rule is calculated using the Softmin operator as well as the degrees of matching between the fuzzy input variables and their corresponding antecedent labels. Consequent labels are in layer 4, where defuzzification is performed using LMOM defuzzification method [3] and the output of each rule is calculated. Layer 5 is the output layer in which the crisp control action is determined.

4.3 Explorer

This component makes a trade-off between exploration and exploitation of the state space. Since we assume that the knowledge-bases of the *Actor* and the *Critic* may be rough and out of tune, in the early steps of simulation (or the actual run), the state space must be sufficiently explored. When the time passes and the *Actor* learns to suggest more appropriate actions, the *Critic* learns to correctly predict the state values by trial-and-error interactions with the environment. Therefore, exploration is smoothly substituted by exploitation. The *Explorer* accomplishes this task by perturbing the action, F, suggested by the *Actor* using the TD prediction error [8], δ, given by equation (13), and the length of the previous episode, T. This process is done via the following equation

$$F' = F + R\sigma(\delta, T) \tag{7}$$

where R is a standard uniformly distributed random variable defined on $[-a, a]$, and σ is some monotonically decreasing function with respect to T and the magnitude of δ. In this way, when the magnitude of δ is large (small), there will be a large (small) difference between the *Actor's* suggested action, *F*, and what is actually applied to the plant, F'. This provides the exploration of the state space. The *Explorer* also provides the perturbation needed by the *Updater* in updating the parameters of the *Actor* and the *Critic*. The perturbation, *s*, is calculated using the following equation:

$$s = \exp\left(\left((F' - F)\delta\right)^2\right). \tag{8}$$

4.4 Updater

This component tunes the labels in the consequent parts of the *Actor* and the *Critic* using a decaying learning rate, the TD error, and the gradient of each FIS (*Actor* and *Critic*) with respect to parameters of the consequent parts of the corresponding FIS. To be more specific, the centers and the spreads in the antecedents of the labels in the consequent parts of each FIS are tuned.

For the *Actor*, the parameters are tuned in order to reach the objective of maximizing the state value so that the system would end up in a good state and eventually avoid failure. This can be done through equation (9) in which s is the perturbation term calculated by the *Explorer*, p is a parameter of *Actor* to be tuned, v is the value of the state calculated by the *Critic*, δ is the TD error, and $\mathrm{sgn}(\bullet)$ is the sign function. The reason $\mathrm{sgn}(\delta)$ is used is that when this term is negative, it means that the current state is worse than the previous one and therefore a step toward the opposite direction is needed. On the other hand, when this term is positive, it means that the current state is better than the previous one and therefore the current step should be taken in the direction of the previous step.

$$\Delta p = \alpha\,\mathrm{sgn}(\delta)s\frac{\partial v}{\partial p} = \alpha\,\mathrm{sgn}(\delta)s\frac{\partial v}{\partial F}\frac{\partial F}{\partial p}. \tag{9}$$

To calculate equation (9), we need to calculate two derivatives on its right hand side. $\partial v/\partial F$ is approximated using (10):

$$\frac{\partial v}{\partial F} \approx \frac{dv}{dF} \approx \frac{v_t - v_{t-1}}{F_t - F_{t-1}}. \tag{10}$$

Since this approximator ignores the change in a state between successive time steps, it is a very crude estimator of the derivative and thus we consider only the sign of this estimator and not its magnitude. The existence of the derivative is an implicit assumption [4].

Calculation of $\partial F/\partial p$ is not difficult. V, a label in the consequent part of the *Actor*, is parameterized by p_V which may be center, left spread, or right spread and R_V is the rule that its consequent label is V. In addition, z_{R_V} is the defuzzified output of the rule R_V calculated by (11) using LMOM defuzzification method [4] and w_{R_V} is the degree of firing of rule R_V. Thus,

$$z_{R_V} = C_V + \tfrac{1}{2}\left(RS_V - LS_V\right)\left(1 - w_{R_V}\right). \tag{11}$$

In (11), C_V is the center of the label V and RS_V and LS_V are its right and left spreads, respectively. The derivative needed for tuning of the *Actor* can be calculated using (10) and (12) .

$$\frac{\partial F}{\partial p_V} = \frac{1}{\sum_i w_i}\sum_{V=R_V} w_{R_V}\frac{\partial z_{R_V}}{\partial p_V}. \tag{12}$$

In the above equation, $i \in \{1,2,\ldots,m\}$, where m is the number of the rules in the knowledge-base of the *Actor*.

For the *Critic*, the objective is to minimize the TD prediction error, δ_t, given by the following equation

$$\delta_t = \begin{cases} 0, & starting\ state \\ r_{t+1} - V_t(s_t), & failure\ state \\ r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t), & otherwise \end{cases} \tag{13}$$

where $V_i(s_j)$ is the *Critic* estimation for the value of state s_j in time step i. The term $r_{t+1} + \gamma V_t(s_{t+1})$ is actually an estimation of $V_t(s_t)$ and, therefore, δ_t is the error of that estimation. γ is the reward discount rate which determines the importance of the future time steps in the current learning cycle. If γ is set to 0, only the immediate reward is considered and if it is set to 1, the value of all the future states and the effect of the applied actions are taken into account in the learning process. In our experiments, we use $\gamma = 0.95$, since this value is usually used by researchers and yields better results. Therefore, the learning rule for the *Critic* is similar to that of the *Actor* and is given by (14) in which β is the learning rate and, like α, it is a small positive variable.

An episode begins when the simulation (or the run of the actual system) starts and it ends when a failure occurs. In the beginning of each episode, the learning rates are set to a relatively large value. Then, during an episode the learning rates decay quickly and reach to small values and from that point forward, the learning rates decay after each N time steps to provide more exploitation of the good policy found by the controller.

$$\Delta p = -\beta\,\mathrm{sgn}(\delta)\,s\,\frac{\partial \delta}{\partial p} = -\beta\,\mathrm{sgn}(\delta)\,s\,\frac{\partial \delta}{\partial v}\frac{\partial v}{\partial p}. \tag{14}$$

In (14), the term $\partial v/\partial p$ can easily be calculated similar to the calculation of $\partial F/\partial p_V$ in (12). This is because the *Critic*, like the *Actor*, is an FIS. The other term, $\partial \delta/\partial v$ is approximated using (15), assuming that the derivative does not depend on r.

$$\frac{\partial \delta}{\partial v} \approx \frac{d\delta}{dv} = (1-\gamma) + \gamma\left(d^2 v\right) \tag{15}$$

where, similar to the *Actor* learning rules, only the sign of $\partial \delta/\partial v$ is used in the calculations. Furthermore, $d^2 v$ is given by the finite difference $v_t - 2v_{t-1} + v_{t-2}$.

5 Experiments

To show the efficiency and superiority of the proposed system, we applied the GRLFC to a well-known control problem called Cart-Pole. In this problem, a pole is hinged to a cart which moves along a track. The control objective is to apply an appropriate force, *F*, to keep the pole in a vertical position and the cart within track boundaries. The state of the system is determined by $(x, \dot{x}, \theta, \dot{\theta})$ in which x and $\dot{x}$ are the displacement and velocity of the cart, and θ and $\dot{\theta}$ are the angular displacement and angular velocity of the pole, respectively. A failure occurs when either $|\theta| > 12^\circ$ or $|x| > 2.4m$ or $|F| > 10N$, whereas a success is when the pole stays balanced for $100{,}000$ time steps. In our experiment, δ is calculated using $\gamma = 0.95$. In addition, half-pole length, $l_p = 0.5m$, pole mass, $m_p = 0.1kg$ and cart mass, $m_c = 1kg$. The dynamics of the cart-pole system are modeled by the following nonlinear differential equations [3]

$$\ddot{\theta} = \frac{g\sin(\theta) + \cos(\theta)\left[\frac{-F - m_p l_p \dot{\theta}^2 \sin(\theta) + \mu_c \operatorname{sgn}(\dot{x})}{m_c + m_p}\right] - \frac{\mu_p \dot{\theta}}{m_p l_p}}{l_p\left[\frac{4}{3} - \frac{m_p \cos^2(\theta)}{m_c + m_p}\right]},$$
$$\ddot{x} = \frac{F + m_p l_p\left[\dot{\theta}^2 \sin(\theta) - \ddot{\theta}\cos(\theta)\right] - \mu_c \operatorname{sgn}(\dot{x})}{m_c + m_p}. \tag{16}$$

One of the important strengths of the proposed model is its capability of capturing the uncertainty in the state of the system. In our particular experiment with the cart-pole problem, vagueness may be caused by uncertainty in sensor readings. The *Fuzzifier* captures the uncertainty through constructing a triangular fuzzy membership function that has the crisp state as its center, C, and $C - LS$ and $C + RS$ as its left spread and right spread, respectively. In this experiment, we consider symmetric membership functions with $RS = LS = 0.001$.

Moreover, the *Actor* has the same 9+4 rules of the Action Selection Network (ASN) and the Critic has same 5+5 rules of the Action Evaluation Network (AEN) in GARIC [4]. The *Actor* rules are as follows.

1. If *Theta* is POSITIVE and *ThetaDot* is POSITIVE then *Force* is POSITIVE LARGE.
2. If *Theta* is POSITIVE and *ThetaDot* is ZERO then *Force* is POSITIVE MEDIUM.
3. If *Theta* is POSITIVE and *ThetaDot* is NEGATIVE then *Force* is ZERO.
4. If *Theta* is ZERO and *ThetaDot* is POSITIVE then *Force* is POSITIVE SMALL.
5. If *Theta* is ZERO and *ThetaDot* is ZERO then *Force* is ZERO.
6. If *Theta* is ZERO and *ThetaDot* is NEGATIVE then *Force* is NEGATIVE SMALL.
7. If *Theta* is NEGATIVE and *ThetaDot* is POSITIVE then *Force* is ZERO.
8. If *Theta* is NEGATIVE and *ThetaDot* is ZERO then *Force* is NEGATIVE MEDIUM.
9. If *Theta* is NEGATIVE and *ThetaDot* is NEGATIVE then *Force* is NEGATIVE LARGE.
10. If *Theta* is VERY SMALL and *ThetaDot* is VERY SMALL and *x* is POSITIVE and *xDot* is POSITIVE then *Force* is POSITIVE SMALL.
11. If *Theta* is VERY SMALL and *ThetaDot* is VERY SMALL and *x* is POSITIVE and *xDot* is POSITIVE SMALL then *Force* is POSITIVE VERY SMALL.
12. If *Theta* is VERY SMALL and *ThetaDot* is VERY SMALL and *x* is NEGATIVE and *xDot* is NEGATIVE then *Force* is NEGATIVE SMALL.
13. If *Theta* is VERY SMALL and *ThetaDot* is VERY SMALL and *x* is NEGATIVE and *xDot* is NEGATIVE SMALL then *Force* is NEGATIVE VERY SMALL.

The *Critic* has the following 10 rules:

1. If *Theta* is POSITIVE and *ThetaDot* is POSITIVE then *State* is BAD.
2. If *Theta* is POSITIVE and *ThetaDot* is NEGATIVE then *State* is OK.
3. If *Theta* is ZERO and *ThetaDot* is ZERO then *State* is GOOD.
4. If *Theta* is NEGATIVE and *ThetaDot* is POSITIVE then *State* is OK.
5. If *Theta* is NEGATIVE and *ThetaDot* is NEGATIVE then *State* is BAD.
6. If *x* is POSITIVE and *xDot* is POSITIVE then *State* is BAD.
7. If *x* is POSITIVE and *xDot* is NEGATIVE then *State* is OK.
8. If *x* is ZERO and *xDot* is ZERO then *State* is GOOD.
9. If *x* is NEGATIVE and *xDot* is POSITIVE then *State* is OK.
10. If *x* is NEGATIVE and *xDot* is NEGATIVE then *State* is BAD.

The *Actor* and the *Critic* knowledge-bases are depicted in Fig.4 and Fig.5, respectively. Fig.6 depicts the 11^{th} rules of the *Actor* as an example. Fig.7 and Fig.8 summarize the rules of the *Actor* and the *Critic*, respectively. Figures 9-11 show the simulation results of the experiments in which various labels of the *Actor* or the *Critic* are damaged and the system has managed to tune those parameters to their appropriate values. Particularly, Fig.12 depicts the success of GRLFC in accomplishing a difficult task in which two of the most important labels of both the *Actor* and the *Critic* are damaged and the spreads are set to 0.1, that is 100 times more vagueness in the input variables; i.e., GRLFC is capable of learning in a few trials even with the existence of vagueness in the input states which are provided by slower sensor signals. In addition, Fig.13 depicts the robustness of GRLFC in the case of changes in the dynamics of the plant. Fig.13 illustrates only the early stages of learning for this task; i.e., learning took more time steps to complete.

In Fig.14, various labels of the *Actor* are damaged. Furthermore, the simulation time step is increased to $0.06ms$ (from $0.02ms$; i.e., the sensors respond 3 time slower) and the starting states are random. Fig.14 also shows that

GRLFC has been successful in avoiding failure for 33 minutes, i.e., 33000 simulation time steps in this experiment. In both of these difficult tasks (i.e. the one whose results are shown in Fig.12 and Fig.14) GRLFC managed to adapt to the new situation only after a few trials.

In Fig.15, the tolerance of failure conditions is confined; i.e., failure conditions are $|\theta| > 6^\circ$, $|x| > 0.4m$ and $|F| > 10N$. In addition, the half-length of the pole is decreased to 0.4 from 0.5. GRLFC adapts itself to the new situation after only 2 trials.

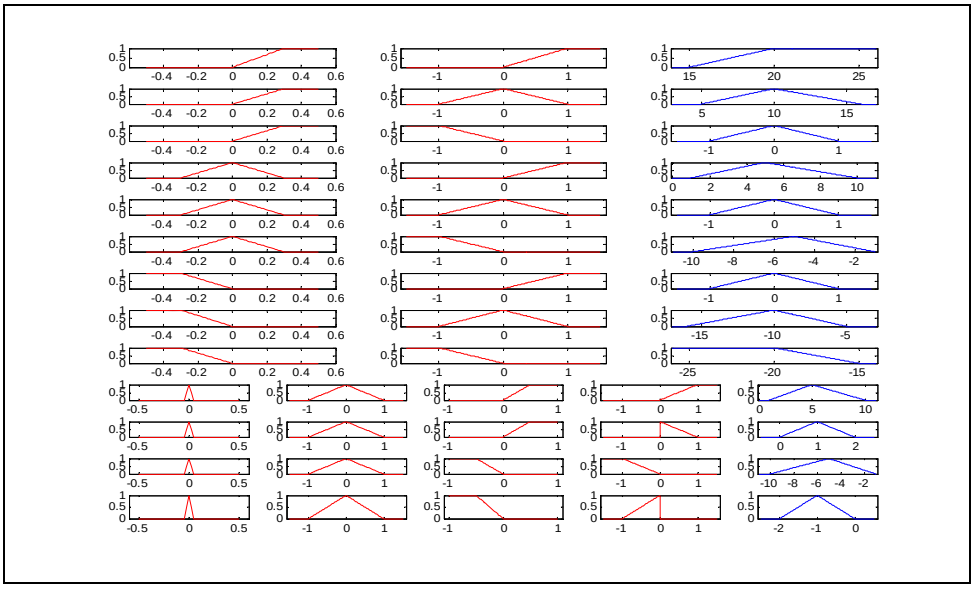


Fig. 3. The *Actor's* Knowledge base.

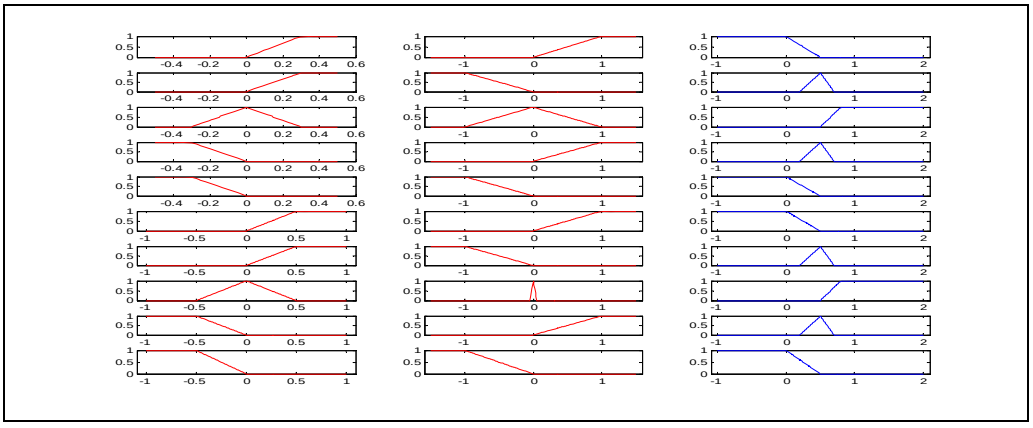


Fig. 4. The Critic's Knowledge base.

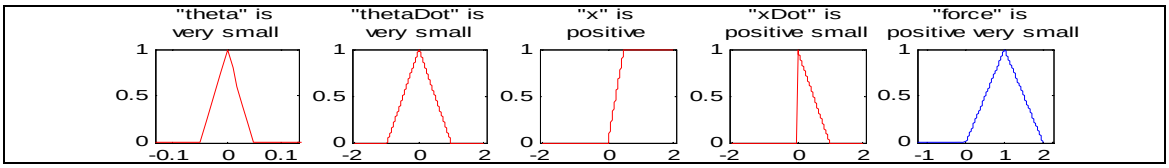


Fig. 5. The 11th rule of the *Actor*

	NE	ZE	PO	VS
NE	NL	NS		ZE
ZE	NM	ZE		PM
VS				
PO	ZE	PS		PL

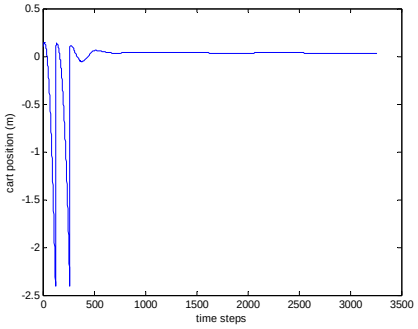
	NE	ZE	PO	VS
NE	NS			
ZE				
VS	NVS			PVS
PO				PS

Fig. 7. 9+4 rules used in Actor.

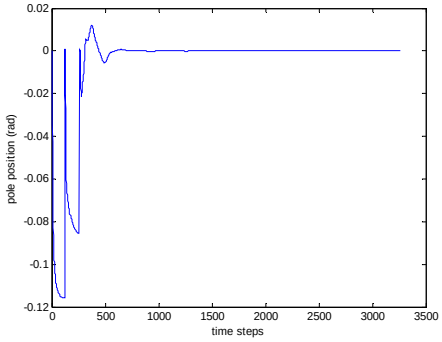
	PO	ZE	NE
PO	BAD		OK
ZE		GOOD	
NE	OK		BAD

	PO	ZE	NE
PO	BAD		OK
ZE		GOOD	
NE	OK		BAD

Fig. 8. 5+5 rules used in Critic.

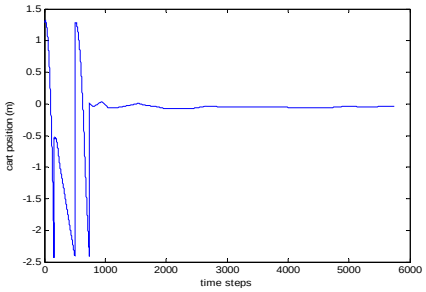


(a) Cart position

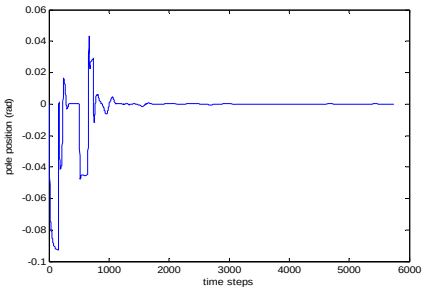


(b) Pole position

Fig. 9. The center of ZE force label was shifted to +5. The system shifted it back to about 0. Each episode started in a non-random state. The learning took only 3 trials.

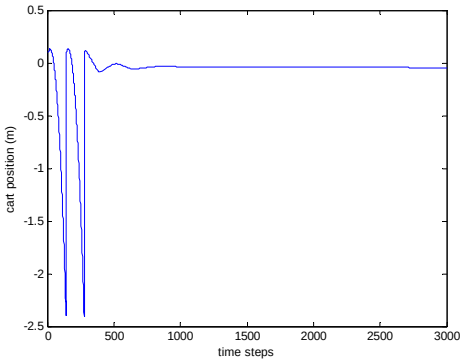


(a) Cart position

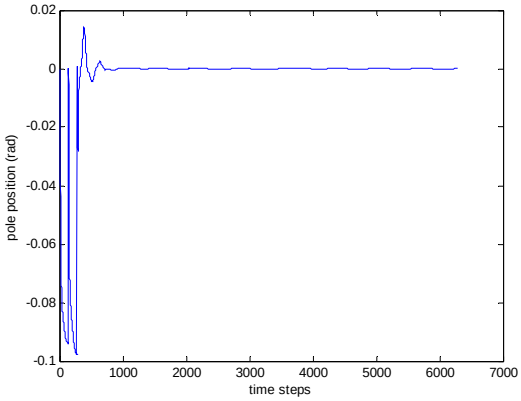


(b) Pole position

Fig. 10. The center of ZE force label was shifted to +3. The system shifted it back to about 0. Each episode started in a random state. The learning took only 4 trials.

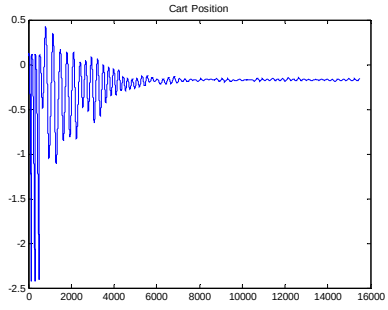


(a) Cart position

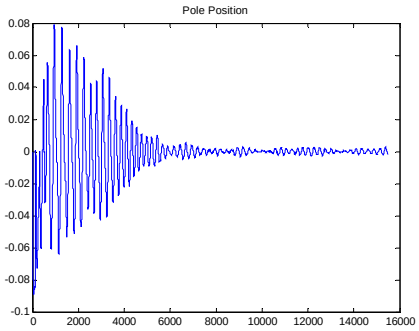


(b) Pole position

Fig. 11. The center of GOOD label was shifted to 0.5. The system shifted it back to about 1. Each episode started in a non-random state. The learning took only 3 trials.

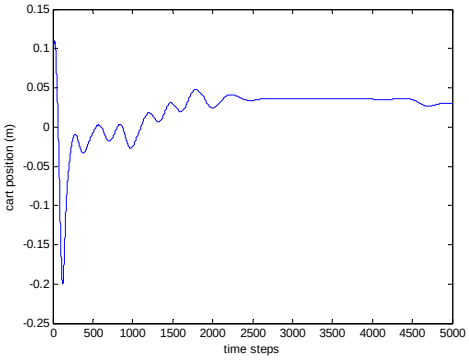


(a) Cart position

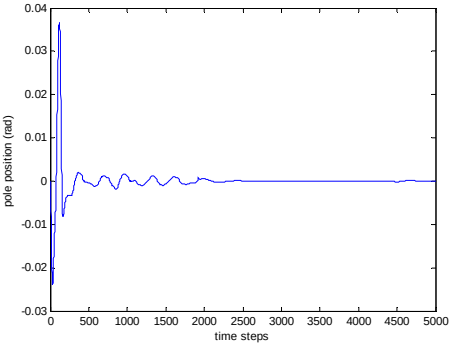


(b) Pole position

Fig. 12. The center of GOOD label and the center of ZE label were damaged (GOOD-0.5 and ZE+3). In addition spreads are set to 0.1. The learning took only 4 trials.

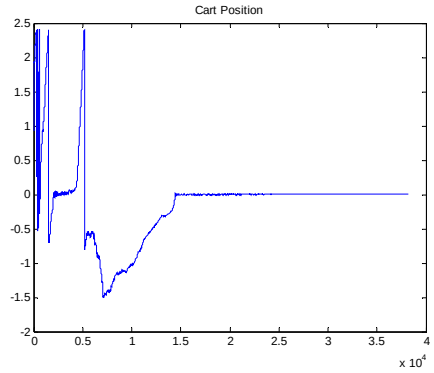


(a) Cart position

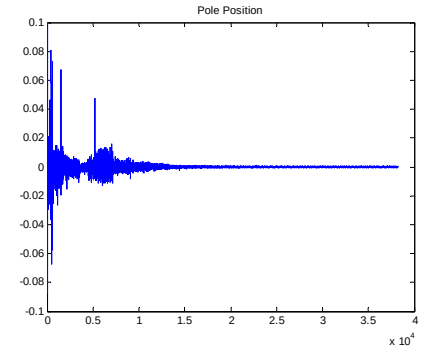


(b) Pole position

Fig. 13. The mass of the cart is changed from 1kg to 2kg. The system is adapted to new situation without a failure.

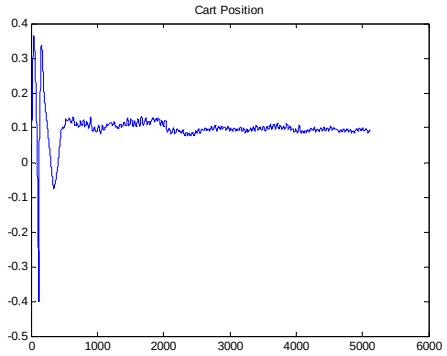


(a) Cart position

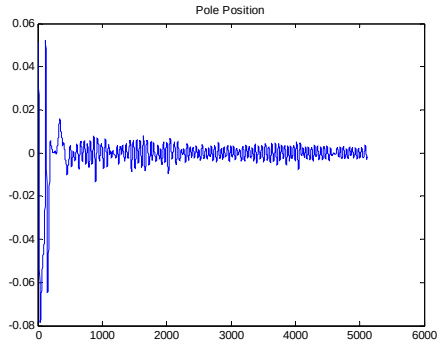


(b) Pole position

Fig. 14. Various force labels were damaged: ZE-2,PL+5,PM+3,PS+2,NM+1. In addition, simulation time step was increased to 0.06 from 0.02. Each episode started in a random state. The learning took 6 trials. GRLFC avoided failure for 33 minutes.



(a) Cart position



(b) Pole position

Fig. 15. Tolerance for the failure conditions is changed and the half length of the pole is decreased to 0.4 from 0.5. The system is adapted to new situation with only 1 failure (2 trials).

Table 1. Learning method comparison on cart-pole problem

System	Proposed by	# of Trials	Vagueness in Input States
--	O. Esogbue et al.	100	No
--	S. M. Mustapha et al.	50	No
GARIC	H. R. Berenji et al.	4	No
FACL	L. Jouffe	3.70	No
GRLFC	M. H. Fazel et al.	3.35	Yes

6 Conclusions and Future Works

This research has modified the GARIC model [3]. The proposed model extends GARIC in several ways. The model is capable of capturing the vagueness in input variables. Furthermore, the learning speed is increased and the number of failures that occur before a success is decreased using variable learning rates. Learning strategy for the *Critic* is different from that of Action Evaluation Network (AEN). This new strategy stems from viewing what is the so called *internal reinforcement* from another point of view; more specifically, considering the *internal reinforcement* as the *Temporal Difference error (TD error)*. The *Explorer* component also extends the Stochastic Action Modifier (SAM) in GARIC to provide better exploration of the state space. Simulation results show the superiority and efficiency of the proposed GRLFC in comparison with other models. Table 1 summarizes the comparison between the learning models, regarding the speed of learning and the features of each.

An interesting topic of future work is to use a linguistic output instead of using a crisp one. This would capture the vagueness in the output variables. In this way, instead of applying the TSK-like method which we used in our model, approaches like Mamdani or Logical, which are able to generate and process linguistic outputs, should be incorporated. By using these methods, the performance of each of these techniques and their combinations can be investigated.

Considering different spreads for the triangular fuzzy membership function of each input element in the input vector can also be considered as another topic of future work. Investigating the effect of the shape of these membership functions, including the symmetry and spreads, can be an interesting study.

Moreover, instead of using the gradient descent method, applying a more efficient technique may contribute to the quality of the solution and further decrease the number of time steps for reaching a solution. Furthermore, reaching such a solution may need less number of time steps.

References

[1] Lin, C. and Y. Xu, A novel genetic reinforcement learning for nonlinear fuzzy control problem, *Neurocomputing*, vol.69, nos. 16-18, pp.2078-2089, 2006.

[2] Lin, C. and C. Lee, Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems, *IEEE Trans. Fuzzy Syst*, vol.2, no.1, pp. 46–63, 1994.

[3] Berenji, H. R. and P. S. Khedkar, Learning and tuning fuzzy logic controllers through reinforcements, *IEEE Trans. on Neural Networks*, vol.3, no.5, pp.724-740, 1992.

[4] Berenji, H. R. and P. S. Khedkar, Using fuzzy logic for performance evaluation in reinforcement learning, *Intl. J. Approximate Reasoning*, pp.131-144, 1997.

[5] Jouffe, L., Actor-critic learning based on fuzzy inference system, *Proc. of IEEE International Conference on Systems, Man, and Cybernetics*, vol.1, pp.339-344, 1996.

[6] Jouffe, L., Fuzzy inference system learning by reinforcement methods, *IEEE Trans. Syst., Man, Cybernetics, Part C: Applications and Reviews*, vol.28, no.3, pp.338-355, 1998.

[7] Chiang, C., H. Chung and J. Lin, A self-learning fuzzy logic controller using genetic algorithms with reinforcements, *IEEE Trans. Fuzzy Syst.*, vol.5, no.3, pp.460-467, 1997.

[8] Sutton, R. S., *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[9] Mustapha, S. M. and G. Lachiver, A modified actor-critic reinforcement learning algorithm, *Proc. of Canadian Conference on Electrical and Computer Engineering*, Halifax, NS, Canada, vol.2, pp.606-609, July 7-8, 2000.

[10] Seo, H. S., S. J. Youn, K. W. Oh, A fuzzy reinforcement function for the intelligent agent to process vague goals, *Proc. of 19th International Conference of NAFIPS*, Atlanta, GA, USA, pp.29-33, July 13-15, 2000.

[11] Feng, H. M., A self-tuning fuzzy control system design, *Proc. of Joint Conference of IFSA and NAFIPS*, Vancouver, BC, Canada, vol.1, pp.209-214. July 25-28, 2001.