FUZZY INFERENCE SYSTEMS OPTIMIZATION BY REINFORCEMENT LEARNING

M. BOUMEHRAZ*, K. BENMAHAMMED, M. L. HADJILI***, V. WERTZ*****

(*) Laboratoire MSE, Département Electronique, Université de Biskra,
medboumehraz@netcourrier.com
(**) Département Electronique, Université de Sétif.
(***) CESAME, Université Catholique de Louvain, Belgique.

ABSTRACT

Fuzzy rules for control can be effectively tuned via reinforcement learning. Reinforcement learning is a weak learning method wich only requires information on the succes or failure of the control application. In this paper a reinforcement learning method is used to tune on line the conclusion part of fuzzy inference system rules. The fuzzy rules are tuned in order to maximize the return function . To illustrate its effectivness, the learning method is applied to the well known Cart-Pole balancing system problem. The results obtained show significant improvements of the speed of learning.

KEYWORDS: reinforcement learning, fuzzy inference systems, Q-learning

1. INTRODUCTION

Reinforcement learning (RL) refers to a class of learning tasks and algorithms in which the learning system learns an associative mapping by maximizing a scalar evaluation (reinforcement) of its performance from the environment [1,2,3]. Compared to supervised learning , RL is more difficult since it has to work with much less information. On the other hand, fuzzy systems are being used successfully in an increasing number of application areas. These rule based systems are more suitable for complex system problems when it is very difficult, if not impossible, to describe the system mathematically. The drawback, however, is that there is no standardized framework regarding the design, optimality and partitioning of fuzzy rule set. Several approaches have been proposed to automatically extract rules from data; gradient descent[4], fuzzy clustering, genetic algorithms [5,6] and reinforcement learning[7,8,9,10,11]. In this paper we use Q-learning to determine the appropriate conclusions for a Mamdani fuzzy inference system. We assume that the structure of the fuzzy system and the membership functions are specified *a priori*.

The rest of this paper is organized as follows: Section 2 and section 3 give the necessary background of reinforcement learning and reinforcement learning methods. Section 4 describes the application of Q-learning for the optimization of fuzzy inference systems. The result of the application of the method for the stabilization of an inverted pendulum are presented in section 5. Section 6 concludes this paper.

2. REINFORCEMENT LEARNING

2.1 Reinforcement learning model

In reinforcement learning an agent learns to optimize an interaction with a dynamic environment through trial and error. The agent receives a scalar value or reward with every action it executes. The goal of the agent is to learn a strategy for selecting actions such that the expected sum of discounted rewards is maximized[1].

In the standard reinforcement learning model, an agent is connected to its environment via percetion and action, as depicted in figure 1. At any given time step t, the agent perceives the state s_t, of the environment and selects an action a_t. The environment responds by giving the agent scalar reinforcement signal, $r(s_t)$ and changing into state s_{t+1}. The agent should choose actions that tend to increase the long run sum of values of the reinforcement signal. It can learn to do this overtime by systematic trial and error, guided by a wide variety of algorithms.

The agent goal is to find an optimal policy, $\pi : S \rightarrow A$, which maps states to actions, that maximize some long-run mesure of reinforcement. In the general case of the reinforcement learning problem, the agent's actions determine not only its immediate rewards, but also the next state of the environment. As a result, when taking actions, the agent has to take the future into account. The reinforcement learning can be summarized In the following steps.

Initialize the learning system
repeat
- With the system in state s, choose an action a according to an exploration policy and apply it to the system
- The system returns a reward r, and also yields next state s'.
- Use the experience, (s,a,r,s') to update the learning system
- $s \leftarrow s'$
until s is terminal

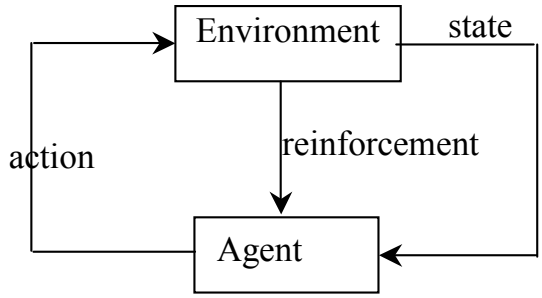

Figure.1 : *Reinforcement*

2.2 The return function

The agent's goal is to maximize the accumulated future rewards. The return function, or the return, R(t), is a long-term measure of rewards. We have to specify how the agent should take future into account in the decisions it makes about how to select an action now. There are three models that have been the subject of the majority of work in this area.

2.2.1 The finite-horizon model

In this case, the horizon corresponds to a finite number of steps in the future. It exists a terminal state and the sequence of actions between the initial state and the terminal one is called a period. The return is given by:

$$R(t) = r_t + r_{t+1} + \cdots + r_{t+K-1} \tag{1}$$

where K is the number of steps before the terminal state.

2.2.2 The discounted return (infinite-horizon model)

In this case the longrun reward is taken into account, but rewards that are received in the future are geometrically discounted according to discount factor γ, $0 < \gamma < 1$ and the criteria becomes.

$$R(t) = \sum_{k=1}^{\infty} \gamma^k r_{t+k} \tag{2}$$

2.2.3 The average-reward model

A third criteria, in which the agent is supposed to take actions that optimize its long-run average reward is also used :

$$R(t) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} r_{t+k} \tag{3}$$

2.3 The state value function or value function

The value function is a mapping from states to states values. The value function $V^{\pi}(s)$ of state s, associated with a given policy $\pi(s)$ is defined as [1] :

$$V^{\pi}(s_t) \equiv E\left[\sum_{k=1}^{\infty} \gamma^k r_{t+k+1}\right] \tag{4}$$

Where s_t is the state at time t, r_{t+k+1} is the reward received for performing action :

$$a_{t+k} = \pi(s_{t+k}) \tag{5}$$

at time $t+k$, and γ is the discount factor ($0<\gamma<1$).

2.4 Action-value function or Q-function

The action-value function measures the expected return of executing action a_t at state s_t, and then following the policy π for selecting actions in subsequent states. The Q-function corresponding to policy $\pi(s)$ is defined as [1]:

$$Q_{t+1}^{\pi}(s_t, a_t) \equiv r_{t+1} + \gamma Q_t^{\pi}(s_{t+1}, \pi(s_{t+1})) \tag{6}$$

The advantage of using Q-function is that the agent is able to perform one-step lookahead search without knowing the one-step reward and dynamics functions.

The disavantage is that the domain of the Q-function increases from the domain of states S to the domain of state-action pairs (s,a).

3. REINFORCEMENT LEARNING METHODS

Reinforcement learning methods can be grouped into two categories: model-based methods and model-free methods. Model based methods have direct links with dynamic programming (DP). Model-free methods can be viewed as appropriate modifications of the model based methods so as to avoid the model requirement.

3.1 Model Based Methods

Dynamic programming (DP) methods [12] are well known classical tools for solving optimization problems. Value iteration and policy iteration are two

widely used DP methods which allow the computation of the optimal value function and optimal policy under the assumption model knowledge.

3.1.1 Value iteration

The basic idea in value iteration is to find the optimal value function. It can be determined by a simple iterative algorithm called value iteration.
Initialize an initial value function V(s) for all s ε ***S***
Repeat
- *Repeat for all s* ε ***S***

$$Q(s,a) \leftarrow r + \gamma \sum_{s' \in S} p_{ss'}(a) V(s')$$

$$V(s) \leftarrow \max_{a \in A} Q(s,a)$$

until a stopping condition.

3.1.2 Policy iteration

The policy iteration algorithm manipulates the policy directly, rather then finding it indirectly via the optimal value function. It operates as follows:
Choose an arbitrary policy π'
Repeat

$$\pi \leftarrow \pi'$$

calculate value function V^{π}

$$V^{\pi}(s) \leftarrow r(s,\pi(s)) + \gamma \sum_{s' \in S} p_{ss'}(\pi(s)) V^{\pi}(s')$$

improve the policy for each state $s \in S$

$$\pi'(s) \leftarrow \arg \max_{a \in A} \left\{ r(s,a) + \sum_{s' \in S} p_{ss'}(a) V^{\pi}(s') \right\} \quad until \ \pi = \pi'$$

3.2 Model free methods

Model-free Reinforcement learning methods are derived by making suitable approximations to the computations in value iteration and policy iteration, so as to eliminate the need for a model. Two important methods result from such approximations: actor-critic methods and Q-learning.

3.2.1 Actor-Critic Method

The Actor-Critic method proposed by Barto, Sutton and Anderson [13], is an adaptive version of policy iteration in which the value function is computed by an algorithm called Temporal difference TD(0). The method can be viewed as a practical approximate way of doing policy iteration : perform one step of an on-line procedure for estimating the value function for a given policy, and at the same time perform one step of an on-line procedure for improving that policy. A bloc diagram of this approach is given in figure 2. It consists of two components: A critic and a reinforcement learning component.

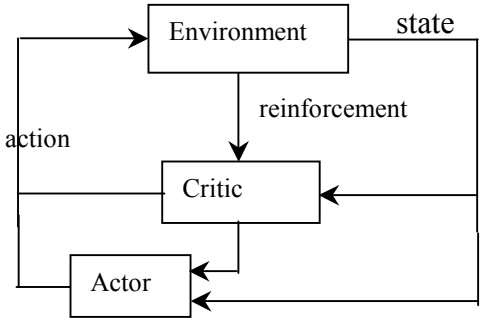

Figure.2 : *Architecture for the actor-critic*

Let (s,a,r,s') an experience tuple summarizing a single transition in the environment. Here s is the state before the transition , a is the applied action, r the instantaneous reward and s' is the resulting state. The value of a policy is learned using TD(0) algorithm which uses the update rule :

$$V(s) = V(s) + \alpha\left(r + \gamma V(s') - V(s)\right) \tag{7}$$

Whenever a state s is visited, its estimate value is updated to be closer to $r + \gamma V(s')$, since r is the instantaneous reward received and $V(s')$ is the estimated value of the actually occurring next state. The TD(0) rule is an instance of a more general class off algorithms called TD(λ), with λ=0. The general TD(λ) rule is given by:

$$V(s) = V(s) + \alpha\left(r + \gamma V(s') - V(s)\right) e(s) \tag{8}$$

TD(λ) rule is similar to the TD(0) rule but it is applied to every state according to its eligibility e(s), rather than just to the immediately previous state s. One version of the eligibility trace is defined to be:

$$e(s) = \sum_{k=1}^{t} (\lambda\gamma)^{t-k} \delta_{s,s_k} \tag{9}$$

where

$$\delta_{s,s_k} = \begin{cases} 1 & if \quad s = s_k \\ 0 & otherwize \end{cases} \tag{10}$$

The eligibility of a state is the degree to which it has been visited in the recent past; when a reinforcement is received, it is used to update all the states that have been recently visited, according to their eligibility. When λ =0 this is equivalent to TD(0). When $\lambda \neq 0$, it is roughly equivalent to updating all the states according to the

number of times they were visited. Note that we can update the eligibility on-line as follows :

$$e(s) = \begin{cases} \lambda\gamma e(s) + 1 & \text{if } s = \text{curent state} \\ \lambda\gamma e(s) & \text{otherwize} \end{cases} \quad (11)$$

3.2.2 Q-Learning

Q-learning is perhaps the more popular of reinforcement learning algorithms. Q-learning is a true learning algorithm. in that it learns the optimal policy function incrementally as it interacts with the environment. The idea of Q-learning is to learn a Q-function that maps the current state s and action a to a utility value Q(s,a) that predicts the total future discounted reward that will be received from current action a. To understand the contest on which Q-learning operates, suppose for the moment that we know the Q-function and that currently the environment is characterized by the state s. The agent chooses an action a so as to maximize Q(s,a). Choosing action a results in an environmental state transition from state s to state s'. The agent then chooses the next action a' so as to maximize $Q(s',a')$. Given that the optimal policy is followed after action a is taken, Q(s,a) is the immediate reward of taking action a from state s, plus the maximum utility possible from the next state s' discounted by the discount factor γ. Therefore Q satisfies the relation:

$$Q(s,a) = r + \gamma \sum_{s' \in S} p_{ss'}(a) \max_{a' \in A} Q(s',a') \quad (12)$$

Temporal difference (TD) methods can be employed to incrementally construct Q as a series of estimates, Q_n. During the learning process equation (12) may not hold and the TD update process is designed to change estimate $Q_n(s,a)$ of $Q(s,a)$ in a direction that reduces the amount of inequality. A specific update formula using temporal difference is given by:

$$Q_{n+1}(s,a) = Q_n(s,a) + \alpha\left(r + \gamma \max_{a' \in A} Q_n(s',a') - Q_n(s,a)\right) \quad (13)$$

Note that equation (13) can be writen as :

$$Q_{n+1}(s,a) = (1-\alpha)Q_n(s,a) + \alpha\left(r + \gamma \max_{a' \in A} Q_n(s',a')\right) \quad (14)$$

where α is a learning rate that is either a small constant or a value that goes to zero as the time step n increases. If each action is executed in each state an infinite number of times on an infinite run and α is decayed appropriately, the Q-values will converge with probability 1 [14]. Q-learning is exploration insensitive: that is, that the Q values will converge to the optimal values independent of how the agent behaves while the data is being collected (as long as all state-action pairs are tried often enough) [14]. This means that although the exploration-exploitation issue must be addressed in Q-learning, the details of the exploration strategy will not affect the convergence of the learning algorithm.

Q-Learning can be summarized in the following algorithm :

Initialize Q(s,a) arbitrarily
Repeat (for each episode)
  Initialize state s
  Repeat (for each step of episode)
    Choose an action a from s using policy derived from Q
    Apply action a and observe reinforcement r and next state s'.
    Update Q :

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max_{a' \in A} Q(s',a')\right)$$

    $s \leftarrow s'$
  until s is terminal
until policy approximation terminated.

4. OPTIMIZATION OF FUZZY INFERENCE SYSTEMES BY Q-LEARNING

Reinforcement learning has been used for optimization of fuzzy inference systems by two types of methods : Methods based on policy iteration, driving to Actor-Critic architectures [7,8]. The others based on value iteration and generalize Q-Learning [9,10,11], in [11] Glorennec uses Q-Learning for the optimization of a zero order Takagi-Sugeno fuzzy inference system, with a constant conclusions. If the action space is continuous the conclusions are equally distributed between lower and upper bounds of the action.

In this paper, we consider a Madani fuzzy inference system and continuous state and action spaces. The FIS structure is fixed *a priori* by the user and the fuzzy sets for the inputs and output are supposed fixed. Our approach, consist in determining the optimal conclusions of the fuzzy inference system.

4.1 Mamdani fuzzy inference system

A Mamdani inference system is described by a set of fuzzy rules of the form:

Rule i : if s is $\boldsymbol{S}^i$ then a is $\boldsymbol{A}^i$

Where s is the fuzzy system input, $\boldsymbol{S}^i$ is a fuzzy label for input in i^{th} rule, a is the output of the fuzzy system and $\boldsymbol{A}^i$ is fuzzy label for the output in i^{th} rule.

The problem is how to choose the appropriate rules in order to optimize system performance (maximize the accumulated future rewards in RL). In this paper we use Q-learning to optimize rule conclusions.

Several competing conclusions are associated to each rule, and a quality value is assigned to each conclusion. The conclusion with the high quality is used by the system to generate actions . The fuzzy rule becomes:

Rule i : if s is $\boldsymbol{S^i}$ then a is $\text{argmax}_{a' \in A} Q(s,a')$

4.2 Learning process

At each rule, several conclusions are associated, and each conclusion has a Q-value:

The fuzzy rule is of the form:

Rule i : if s is $\boldsymbol{S}^i$ then a is $\boldsymbol{A_1}$ with $Q^i(s, \boldsymbol{A_1})$

or a is $\boldsymbol{A_2}$ with $Q^i(s, \boldsymbol{A_2})$

or a is $\boldsymbol{A_3}$ with $Q^i(s, \boldsymbol{A_3})$

or a is $\boldsymbol{A_m}$ with $Q^i(s, \boldsymbol{A_m})$

where $\boldsymbol{A_1}$, $\boldsymbol{A_2}$,….., $\boldsymbol{A_m}$ are the fuzzy sets of the outputs and $Q^i(s, \boldsymbol{A_i})$ is the Q-value of the conclusion a is $\boldsymbol{A_i}$ of the rule j.

During learning the Q-value of each conclusion is updated using Q-learning :

$$Q^i_{t+1}(s_t, A_j) = Q^i_t(s_t, A_j) + \alpha \mu_i(s_t)\{r_{t+1} + \gamma V_t(s_{t+1}) - Q^i_t(s_t, A_j)\} \qquad (15)$$

Where $\mu_i(s_t)$ is the truth value of the i^{th} rule and A_j is the j^{th} conclusion of the i^{th} rule.

With the value of the new state given by:

$$V_t(s_{t+1}) = \sum_{i=1}^{N} \frac{\mu_i(s_{t+1})}{\sum_{j=1}^{N} \mu_j(s_{t+1})} \mu_i(s_{t+1}) \max_{a \in A} Q^i_t(s_{t+1}, a) \qquad (16)$$

if s_{t+1} is a final state then :

$$V_t(s_{t+1}) = 0 \qquad (17)$$

5. RESULTS

The proposed method is applied to a classic problem; the pole balancing problem or inverted pendulum problem. In this problem a pole is hinged to a motor-driven cart which moves on rail tracks to its right or its left. The primary control task is to keep the pole vertically balanced.

The dynamics of the cart-pole system are modeled by the following non linear differential equation [15] :

$$\ddot{\theta} = \frac{g \sin\theta + \cos\theta \left[\frac{-f - ml\dot{\theta}^2 \sin\theta}{m_c + m} \right] - \frac{\mu_p \dot{\theta}}{}}{l\left[\frac{4}{3} - \frac{m\cos^2\theta}{m_c + m} \right]} \qquad (18)$$

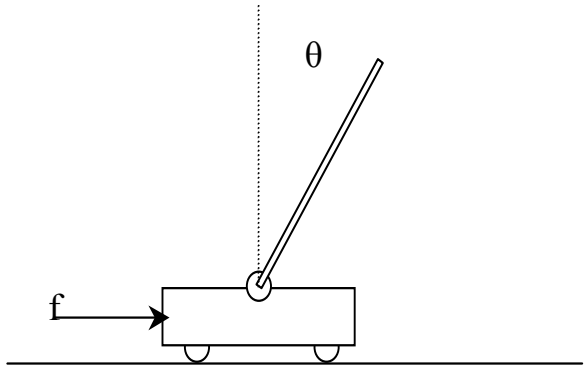

Figure.3 : *The Cart-Pole System*

where θ is the angle of the pole, $\dot{\theta}$ is its angular velocity, g is the gravity, m_c=1.0 Kg is the mass of the cart, m=0.1 Kg is the mass of the pole, l=0.5 m is the half-pole length and μ_p=0.005Ns/m is the coefficient of friction of pole on cart. The sample period is 20 ms.

We assume that a failure happen when |θ|>45°.and the equation of motion is not known to the controller and that only a vector describing the cart-pole system's state at each time step is known.

The inputs of the fuzzy controller are error e and change in error Δe given by:

$$e(k) = \theta(k) \qquad (19)$$

$$\Delta e(k) = e(k) - e(k-1) \qquad (20)$$

The output is the force f and the Q-values of the conclusion of each rule.

The fuzzy partitions of the error, change in error and force are described in figure 4.

The rule base is choosen arbitrary and the Q-values of the conclusions are set initially to zero. We use center of area defuzzification and the min operator to implement the premise and implication.

A trial in our experiments refers to starting with the cart-pole system set to an initial state and ending with the appearance of a failure signal or successful control of the system for an extended period (1000 time steps or 20 seconds). The Q-learning was applied to tune fuzzy rule conclusions. The free constants were γ= 0.95 and α set initially to 0.1 and decreases. Figure 5 shows the average return per trial performance of the controller during the learning process; the average return per trial, figure.6 shows the rule table obtained after learning and figures 7 and 8 show the response of the system for initial angle equal to –50° and 28° respectively.

It is clear that the average return increases during learning until it reaches a sub-optumal value. The obtained fuzzy controller is able to stabilize the pole for angles inferior to 55°.

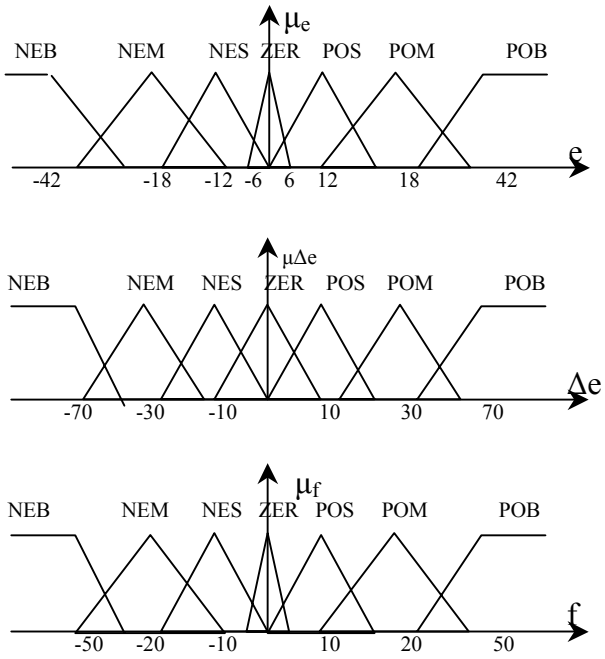

Figure.4 : *Membership functions*

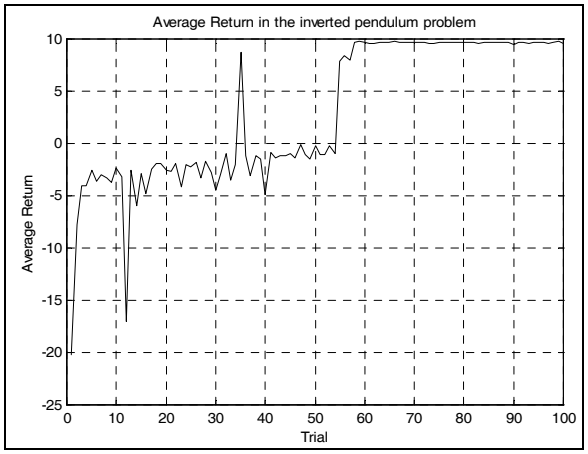

figure.5 : *The Average Return*

Δe

	NEB	NEM	NES	ZER	POS	POM	POB
NEB	NEB	NEM	NEM	NEB	NEP	NEP	ZER
NEM	NEB	NEB	NEB	NEM	ZER	POS	POS
NES	NEM	NEM	NEB	NES	POS	POS	ZER
ZER	NEB	NEM	NES	ZER	POS	POM	POB
POS	NEM	NES	NES	POS	POM	POB	POM
POM	NES	NES	ZER	POM	POB	POM	POB
POB	ZER	POS	POS	POB	POB	POM	POB

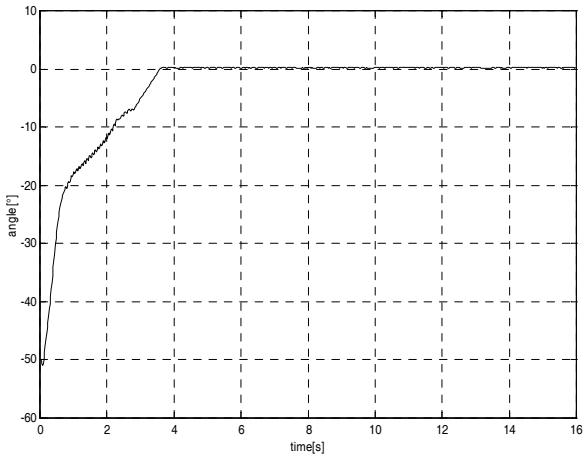

Figure.6 : *Rule table obtained after learning*

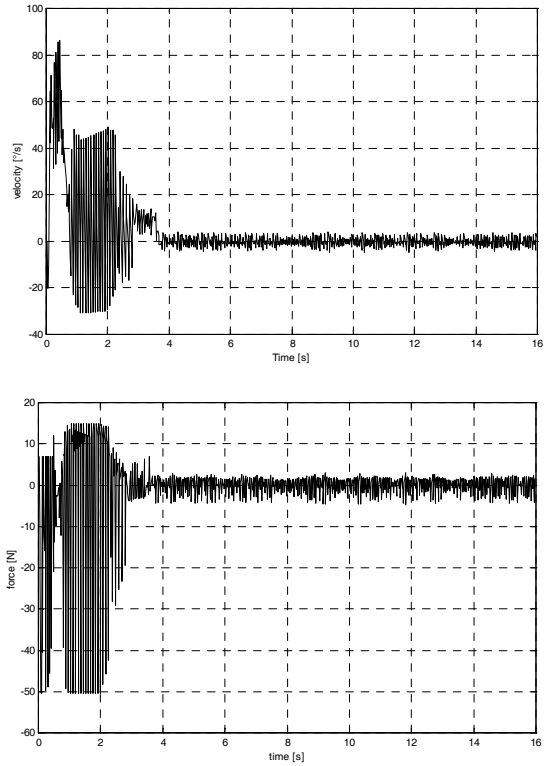

Figure.7 : *Angle, velocity and force for initial angle equal 50°*

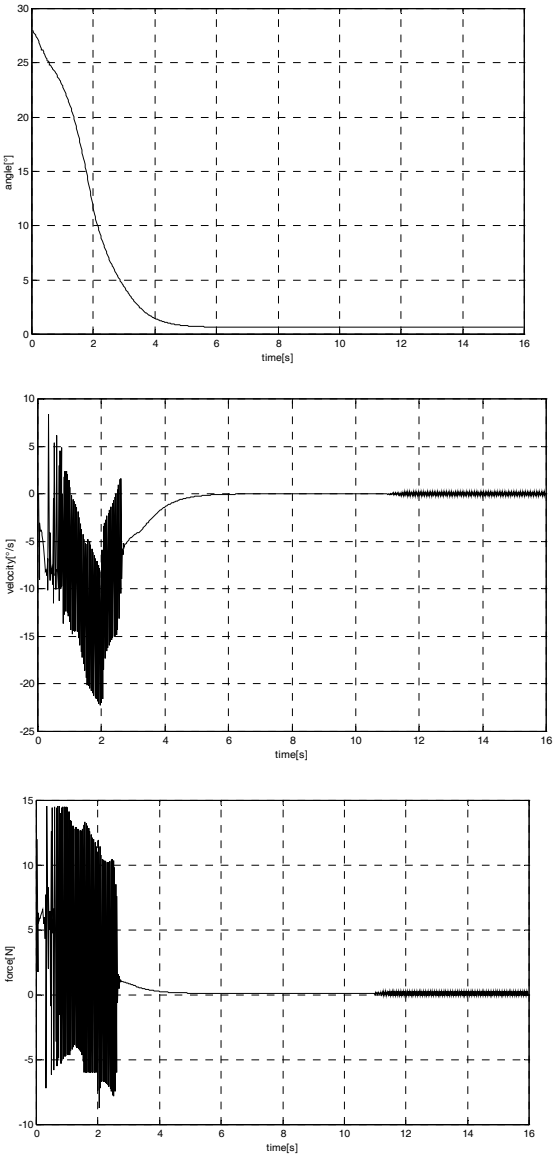

Figure 8 : *Angle, velocity and force for initial angle equal to 28°*

6. CONCLUSIONS

In this paper we have proposed a new method of optimizing fuzzy inference systems based on Q-learning. This method was applied to cart-pole system. After learning , the controller is able to stabilize the pendulum. We assume that the structure of the fuzzy system is fixed *a priori*. The optimization of membership function parameters and number of rules will improve the performance of the proposed method.

REFERENCES

R. S. Sutton, A. G. Barto, Introduction to reinforcement learning, MIT Press/Bradford Books, Cambridge, MA, 1998.

V. Gullapalli, Reinforcement learning and its appication to control, Ph. D. Thesis, University of Massachusetts, Amherst, MA, USA, 1992.

L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: a survey, Journal of Journal Artificial Intelligence Research 4, 1996.

J. R. Jang, Self-Learning Fuzzy Controllers Based on Temporal Back Propagation, IEEE Transactions on Neural Networks, Vol. 3 No. 5, September 1992.

M. G. Cooper, J. J. Vidal, Genetic Design of Fuzzy Controller, Proceedings of Second Inernational Conference on Fuzzy Theory and Technology; Durham, NC, October, 1993.

A. Bonarini, Evolutionary learning of fuzzy rules:competition and cooperation, in Fuzzy modeling : paradigms and practice, Kluwer Academic Publishers, Norwell, MA, 1995.

H. R. Berenji P. Khedkar, Learning and Tuning Fuzzy Logic Controllers Through Reinforcement, IEEE Transactions on Neural Networks, Vol. 3 No. 5, September 1992.

M. V. Buijtenen, G. Schram, R. Babuska, B. Verbruggen, Adaptive Fuzzy Control of Satellite Attitude by Reinforcement Learning, IEEE Transactions on Fuzzy Systems, Vol. 6, No. 2, May 1998.

H. R. Berenji, Fuzzy Q-Learning: a new approach for fuzzy dynamic programming, Proceedings of IEEE international conference on Fuzzy Systems, Nj, 1994.

P. Y. Glorennec, L. Jouffe, Fuzzy Q-Learning, Procedings of FUZZ-IEEE'97, Barcelona, Spain, July 1997.

P. Y. Glorennec, Reinforcement Learning: an Overview, ESIT 2000, Aachen, Germany, 14-15 September 2000.

D. P. Bertsekas, Distributed Dynamic Programming, IEEE transactions on Automatic Control, 27, 1982.

A. G. Barto, R. S. Sutton and C. W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, IEEE Transactions on Systems, Man and Cybernetics, SMC-13(05), 1983.

C. Watkins Learning from Delayed Rewards, PhD. Thesis, University of Cambridge, England, 1989.

K. Passino, S. Yurkovich, Fuzzy Control, Addison Wesley, California, 1998.