A Self-Learning Fuzzy Controller*

Yung-Yaw Chen† , Kao-Zong Lin‡ , Shun-Tang Hsu‡

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

abstract

A learning scheme for the fuzzy control systems is proposed in this paper. The proposed scheme is able to find proper control actions in the rule-base of a fuzzy controller for a number of different dynamic systems, including the well-known cart-pole system. Most of the related pieces of work achieve the self-learning behavior with the learning mechanism attached, and do not come up with a trained controller which can function independently. Our learning scheme which involves the ideas of temporal difference and fuzzy control is able to construct a fuzzy controller which can serve the control purpose on its own after the learning process. Moreover, a number of different plant dynamics have also been tested and the results suggest that the method is universal enough to control processes other than the commonly seen inverted pendulum.

1 Introduction and Motivation

Fuzzy control has found many applications and caught the eyes of the world in recent years. A number of commercial implementations by Japan, e.g. the detection of load and control of the washing cycle of a washing machine, the focusing of the video camera, etc, have promoted fuzzy theory from academic research to production lines. In Japan, the word "fuzzy" has even become a point of selling with great popularity in the market.

The basic idea of fuzzy control is to make use of the knowledge and experience from the experts to form a rule-base with linguistic *if-then* rules. Proper control actions are then derived from the rule-base which can be considered as an emulation of the behavior of the human operators. Different from other control theories, fuzzy control does not involve complex mathematical operations and models of the plants. From one point of view, fuzzy control is actually a form of control rather than a control algorithm. Its design theory does not explicitly suggest a solution for a particular control problem, which often offered by most control schemes, e.g. pole-placement, adaptive, or robust controls. The question of *how* to solve a control problem in fuzzy control is assumed to be the responsibility of the *experts*. Consequently, the design of a fuzzy controller depends entirely on the knowledge and experience of the experts, or the physical sense and intuition of the designer that is far from systematic and reliable.

The issue of learning is therefore of particular interests in fuzzy control. Instead of designing a fuzzy controller, the goal is to allow the controller to learn proper control actions through numbers of trials. Mamdani[5] proposed a performance index table for updating the control actions with some success. Shao[6] applied the method to the process controls. However, Mamdani's method is incapable of producing a stand-alone fuzzy controller with the commonly seen if-then rules. Barto, Sutton and Anderson[1] used two neuron-like elements, ASE and ACE, which are related to the theory of temporal difference and were successful in solving a difficult control problem, the cart-pole system. Lee[4] extended and incorporated the idea with fuzzy control and incorrectly claimed a better result since fewer state variables are considered in his system. Berenji[3] extended further to incorporate neural network architecture in his work with similar

*Supported by NSC Grant 79-0404-E002-51, Taiwan, R.O.C.
†Associate Professor
‡Graduate Student

0-7803-0236-2 /92 $3.00

results as in [4]. However, the goal of deriving a working and stand-alone controller was still not achieved. Lee's fuzzy controller can function only with the learning mechanism attached.

In this paper, a learning scheme based on Barto and Sutton's approach for fuzzy control systems is proposed. The basic discipline is the concept of temporal difference and the so called, ASE and ACE for the determination of the proper reinforcement and control. The reinforcement r is modified and the continuous control force is adopted. Most important of all, our scheme is developed to handle a general class of dynamic systems, not just the too-commonly-seen cart-pole system. Simulation results show that the scheme is functional at least for most, if not all, second- and third-order systems as well as some nonlinear examples.

The organization of the paper is as follows. Some basic operations of the fuzzy controllers are introduced in section 2. The proposed learning algorithm is described in section 3. Following is the simulation result in section 4, which shows that the algorithm is able to handle more than a specific example. Finally, the conclusion is summarized in the last section.

2 Fuzzy Control Systems

Fuzzy sets theory has found its applications in many fields, especially in fuzzy knowledge-based systems, such as fuzzy logic control and approximate reasoning. An introduction to fuzzy logic control will be given in the following to show the general structure of the fuzzy knowledge-based systems. To simplify the notation, we shall discuss a rule-base with only two rules and each rule has only two state variables and one action variable. (It can be easily extended to systems with more state and action variables.) The basic operating procedures of a fuzzy logic controller can be summarized as:

Fuzzification

Suppose that the premise part of the control rules takes inputs from the sensor readings which are usually real numbers. These real-valued sensor measurements, e.g. ${x_1}^0$ and ${x_2}^0$, are matched to their corresponding fuzzy variables by finding the matching membership values, such as:

$$A_1({x_1}^0), B_1({x_2}^0), A_2({x_1}^0), B_2({x_2}^0) \tag{1}$$

Fuzzy Reasoning

For all the control rules in the rule base, we derive the truth values(or *strengths*) of each rule in the premise by forming the conjunctions of the matching membership values:

$$\mu_1 = \mu_{A_1}({x_1}^0) \wedge \mu_{B_1}({x_2}^0) \qquad (\text{rule} - 1) \tag{2}$$

$$\mu_2 = \mu_{A_2}({x_1}^0) \wedge \mu_{B_2}({x_2}^0) \qquad (\text{rule} - 2) \tag{3}$$

The output of each control rule from its action part is represented by the fuzzy sets C_1' and C_2' and is calculated by:

$$C_1' = \mu_1 C_1(u) = \mu_1 \times C_1(u) \qquad (\text{rule} - 1) \tag{4}$$

$$C_2' = \mu_2 C_2(u) = \mu_2 \times C_2(u) \qquad (\text{rule} - 2) \tag{5}$$

Then the combined result of the inferences forms a fuzzy set C' as:

$$C' = C_1' \cup C_2' \tag{6}$$

where the $\cup$ operation is generally the "max" function.

Defuzzification Procedure

The purpose of the defuzzification process is to transform the output of the inferences, which is a fuzzy set, to a real number so that it could be used to control a process, i.e. we need to find a real number

to represent a fuzzy set. Many algorithms were proposed and the most commonly adopted method(called the Center-of-Area operator) is to find the value corresponding to the center of area of the membership function of C' as follows:

$$u' = \frac{\int C'(u)udu}{\int C'(u)du} \tag{7}$$

where u' is the real-valued output.

To simplify the computations involved in our learning algorithm, we adopted an efficient scheme as described in the following.

$$u' = \frac{\sum u_i \mu_i}{\sum \mu_i} \tag{8}$$

where u_i is the center point of the membership function in the action part for each rule and μ_i is the firing strength for that particular rule.

3 Self-Learning Scheme

3.1 Problem Formulation

With the format of the fuzzy controller discussed in the previous section, we intend to develop a learning algorithm which will be able to properly adjust the center points, u_i, $i = 1, ..., n$ of the membership functions in the action part of the fuzzy rule-base such that the task of set-point control can be achieved. To simplify the problem, the pre-condition part is arbitrarily partitioned into equal terms for each variable, e.g. 11 linguistic terms for x_1, 11 linguistic terms for x_2, and 121 rules in the rule-base in our case.

For the learning scheme to operate, some basic assumptions on the plant are assumed. Although there is not yet any mathematical analysis regarding the range of dynamic systems such that the scheme can work, the systems we investigated were basically SISO Time-Invariant systems.

3.2 System Description

The block diagram of the learning system consist of three basic blocks: the Plant dynamics, Fuzzy Controller(FC), and Neuron Adaptive Elements(NAE).

3.2.1 Fuzzy Controller

Following the format described in section 2, the rule-base of the fuzzy controller is of the form:

$$\text{if } E \text{ is } A_i, \text{ and } CE \text{ is } B_i, \text{ then } U \text{ is } U_i$$

where E is the error of the system output, CE is the change rate of the error of the system output, U is the control action, and A_i, B_i, U_i are corresponding linguistic terms.

The linguistic terms for the variables in the pre-condition part, E and CE, are arbitrarily defined because there is no knowledge of the *proper* settings, if any, of the terms. The membership functions for E and CE are partitioned into 11 terms each and equally positioned in the domain of interest. The membership functions for the control action U are defined to be of the triangular shape with bases of length 1 and their center points u_i, $i = 1, ..., 121$ are randomly assigned by a Gaussian distribution with zero mean and 0.01 standard deviation before the training of the rule-base. To be noted is that the length of the bases does not have any effect on the result of the rule-base due to our simplified defuzzification process. (The problem of involving more parameters in the training is currently under investigation.)

3.2.2 Neuron Adaptive Elements

In this section, an approach for the self-learning of a fuzzy controller is described. At the beginning, a number of parameters used in the scheme will be introduced so that a clearer view may be provided.

- u_i: Center point of the membership function for the control of rule-i,
- μ_i: Firing strength of rule-i,
- w_i: Weights of the ASE for each rule-i,
- e_i: Eligibility trace for rule-i,
- $\hat{r}$: Internal reinforcement given by the ACE,
- p: Prediction by the ACE,
- $\bar{\mu}_i$: Trace of the firing strength of rule-i,
- v_i: Weights of the ACE, and
- r: Output reinforcement which is defined as:
  1. $r = 1$ when the system fails, i.e. the system output is out of a prescribed bound, $[a, b]$,
  2. $r = 0$ when the system output is still within the bound,
  3. $r = -\frac{1}{|a-b|}(\frac{1}{N}\sum_{k=1}^{N}|E(kT)|)$, where T is the sampling period and NT is the learning period.

Specifically, the reinforcement r is the indication of the performance of the system which will be feedback to the ACE so that ACE can derive the prediction of the system status and generate an internal reinforcement for the ASE.

Associative Search Element(ASE)

The function of the ASE is to associate the firing strengths of the rule-base and the internal reinforcement supplied by the prediction of the ACE with the weights of the ASE which are directly related to the control actions of the fuzzy controller.

The control actions of the fuzzy controller are determined by u_i, $i = 1, ..., 121$ through the defuzzification process and are related to the weights of ASE by

$$u_i = FL \times tanh(kw_i) \tag{9}$$

where k is a real coefficient, 0.01, and FL is the force limit which is the saturation limit of the actuator. Many other functions, such as the sigmoidal function, can also be used as the output function with similar results.

Unlike Barto's work which has only one cell operating at any time instant t, there are usually 4 fuzzy cells or rules fired with non-zero strengths in our case. The corresponding weights of the ASE are updated by:

$$w_i(t+1) = w_i(t) + \alpha\, sign(E)\, |e_i(t)\, \hat{r}| \tag{10}$$

where α is the learning rate. The eligibility trace of rule-i, e_i, is updated by:

$$e_i(t) = \delta\, e_i(t-1) + (1-\delta)\, \mu_i(t)\, u_i(t) \tag{11}$$

where δ is the rate of decay, $0 \leq \delta < 1$.

Adaptive Critic Element(ACE)

The ACE receives the externally feedback reinforcement signal r and bases on the current system status to determine an internal reinforcement signal, $\hat{r}$, for the ASE. The internal reinforcement is derived through the prediction $p(t)$ as:

$$\hat{r} = \gamma\, p(t) - p(t-1) \tag{12}$$

where $0 \leq \gamma < 1$. $p(t)$ is the prediction defined by:

$$p(t) = \sum_{i=1}^{n} v_i(t)\, \mu_i(t) \tag{13}$$

The way the weights of the ACE, v_i, are updated so that $p(t)$ can converge to the accurate prediction is given by:

$$v_i(t+1) = v_i(t) + \beta\, \hat{r}\, \bar{\mu}_i(t) \tag{14}$$

where $0 \leq \beta < 1$. $\bar{\mu}_i$ acts like the trace of μ_i and is derived by:

$$\bar{\mu}_i(t) = \lambda\, \bar{\mu}_i(t-1) + (1-\lambda)\, \mu_i(t) \tag{15}$$

where $0 \leq \lambda < 1$.

It is beyond the scope of our paper to fully explain the derivations of the learning rules above. Please refer to [1-2] and [7] for more details of the general principles of the method.

4 Results

A computer simulation program is set up on a SUN SPARC workstation in the software simulation package MATLAB. The sampling time is $1ms$ for the first two cases and $20ms$ for the last one. The time of learning for each trial is set to be 1000 time steps, i.e. 1 sec for the first two cases and $20sec$ for the last one..

Three different dynamic systems are tested by the scheme with minor parameter variations.

- $G_1(s) = \frac{100}{s^2+10s+100}$, stable LTI system,
- $G_2(s) = \frac{100}{s^2-10s-100}$, unstable LTI system,
- Cart-Pole system(Detailed dynamics can be found in [1]) with
  1. $m_c = 1.0kg$, mass of the cart,
  2. $m = 0.1kg$, mass of pole,
  3. $l = 0.5m$, half-pole length.

CASE 1: Figure 1 shows the simulation results of our learning scheme for the stable transfer function $G_1(s)$. The "untrained" curve is the system response without any learning, where the output is basically uncontrolled. Curve-1 is the response of learning after 3 trials; curve-2 is the response for 6 trials; curve-3 is for 9 trials; and curve-4 for 12 trials. As we can see, the system learns to approach the set-point almost from the beginning of the trials, though not without overshoots. It quickly converges to curve-4 in the figure with little overshoot after 12 trials. To be noted is that our system learns to drive the output response to a prescribed set-point in a few trials and does not just stay around a certain bound as described in [1] and [4]. To demonstrate its learning capability, we also trained the system with 16 initial conditions so that a rule-base with f_i's is derived. An arbitrary initial condition $(0.3, 10)$ is then chosen and its corresponding response is shown in Figure 2 with very little overshoot and very small rise time.

CASE 2: For the unstable transfer function $G_2(s)$, the untrained system quickly diverges before learning as shown in Figure 3. It takes a little longer for the system to learn to drive the output to the set-point. (It still diverges after 3 trials.) However, after 9 trials (Curve 3), the output response achieves a much better

performance. A 16-point trial set is also applied to the system and an arbitrary initial condition $(1.5, 20)$ is tested with its response in Figure 4.

CASE 3: For the cart-pole system, the results of the learning scheme are very similar to the previous two cases as shown in Figure 5 and Figure 6, with the trained control actions.

The simulation results for the three cases, which belong to three different classes of dynamic systems, suggest that our learning scheme is general and not specifically for one particular system.

5 Conclusions

A self-learning algorithm for the set-point control of a fuzzy control system is proposed. Several examples were tested and the results suggested that the scheme can work for many different classes of dynamic systems. Unlike many previous works which were designed for a specific dynamic system, such as the cart-pole system, our approach is developed aiming at solving a more general learning problem. Moreover, we have succeeded in constructing a fuzzy controller which can function independently after the training without the learning mechanism, which means we will be able to design a fuzzy controller through learning and will be free of the requirement of an expert who is, quite often, not available. From the view point of fine-tuning a fuzzy controller, our method will also provide a systematic approach for accomplishing the task, e.g. if the system performance is not satisfied, the expert-generated fuzzy controller can be used as the initial settings of the learning process. The proposed learning scheme is extremely successful at this preliminary stage. Much more work will be done to make the method complete and efficient in the future.

6 Acknowledgement

The authors would like to thank Dr. Hamid Berenji for his useful comments on this paper.

References

[1] Barto, A., Sutton, R., and Anderson, C., "Neuronlike Adaptive Elements That Can Solve Difficult Learing Control Problems", *IEEE Trans. Syst., Man, and Cybern.*, Vol. SMC-13, No. 5, pp. 834-846, 1983.

[2] Barto, A. and Sutton, R., "Simulation of Anticipatory Responses in Classical Conditioning by a Neuron-like Adaptive Element", *Behavioral Brain Res.*, Vol. 4, pp. 221-235, 1982.

[3] Berenji, R. H., "A Reinforcement Learning-Based Architecture for Fuzzy Logic Control", to appear in the *International Journal of Approximate Reasoning.*

[4] Lee, C.-C., "A Self-Learning Rule-Based Controller Employing Approximate Reasoning and Neural Net Concepts", *International Journal of Intelligent Systems*, pp. 71-93, Jan. 1991.

[5] Procyk. T. and Mamdani, E., "A Linguistic Self-Organizing Process Controller", *Automatica*, Vol. 15, pp. 15-30, 1979.

[6] Shao, S., "Fuzzy Self-Organizing Controller and its Application for Dynamic Processes", *Fuzzy Sets and Systems* Vol. 26, pp. 151-164, 1988.

[7] Sutton, R. and Barto, A., "An Adaptive Network that Constructs and Uses an Internal Model of its World", *Cognition and Brain Theory*, Vol. 4, pp. 213-246, 1981.

[8] Zadeh, L.A., "Fuzzy Sets", *Information and Control*, pp. 338-353, 1965.

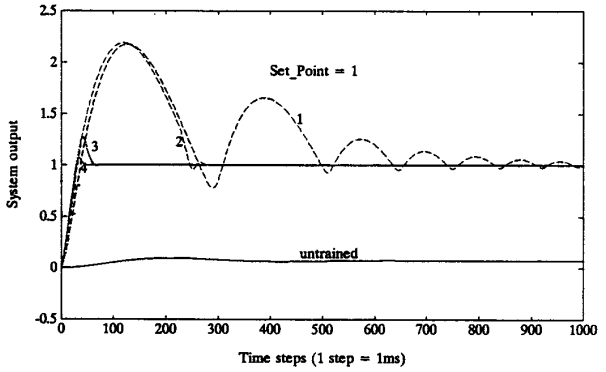

Figure 1: The system response for $G_1(s)$

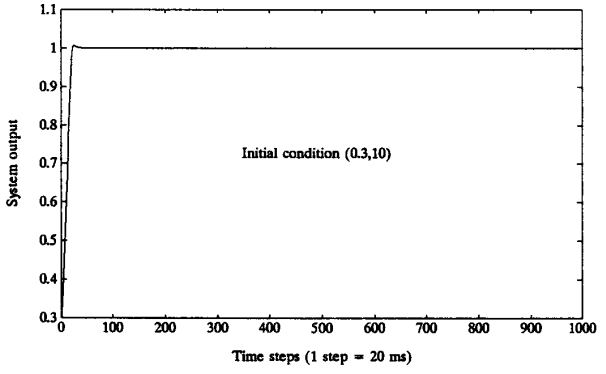

Figure 2: The system response for $G_1(s)$ with the trained controller and the initial condition $(0.3, 10)$

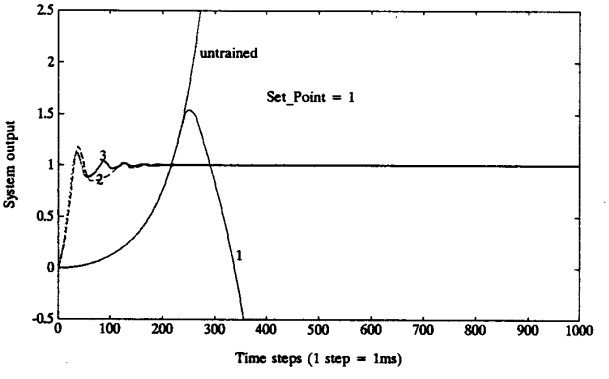

Figure 3: The system response for $G_2(s)$

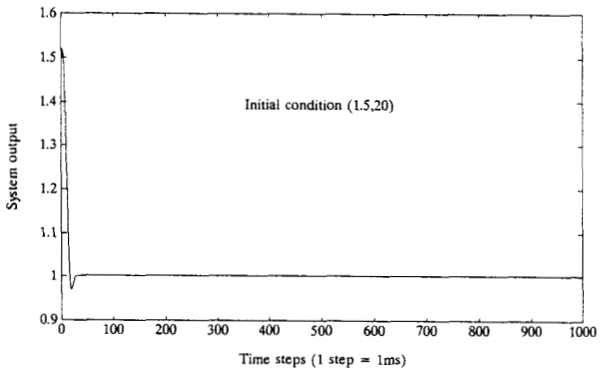

Figure 4: The system response for $G_2(s)$ with the trained controller and the initial condition $(1.5, 20)$

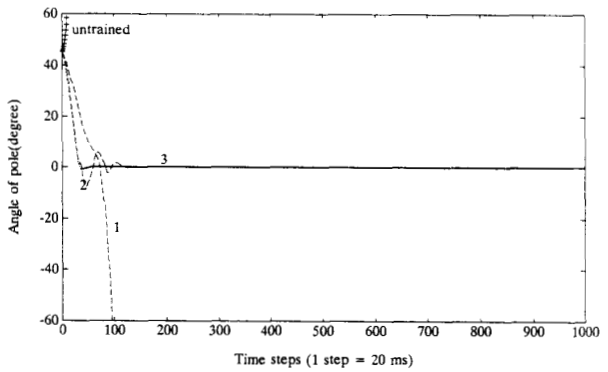

Figure 5: The system response for $G_3(s)$

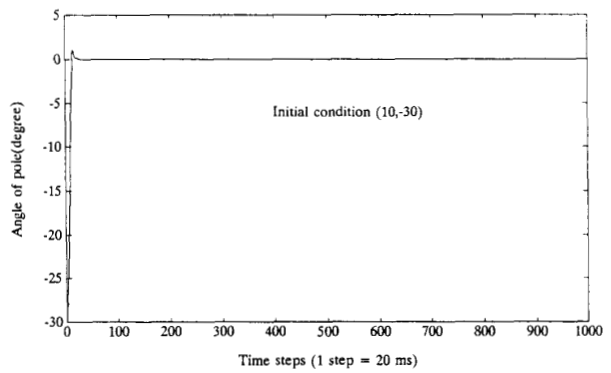

Figure 6: The system response for $G_3(s)$ with the trained controller and the initial condition $(10, -30)$