A Heuristic Criterion Algorithm of the Self-learning Fuzzy Controller

Chiy-Ferng Perng and Yung-Yaw Chen

Lab. 202, Dept. of Electrical Engineering, National Taiwan University
E-mail : pcf@ranger.ee.ntu.edu.tw

Abstract

Through the observation of the learning process of the human, we can find punishments and rewards play a somewhat important role. The judgment of whether the action should be punished or rewarded and the decision of the degree of punishments or rewards are two major problems in such a scheme. In this paper, we propose a heuristic criterion algorithm to solve these problems and apply it to a fuzzy controller.

1. Introduction and Motivation

Since Prof. L. A. Zadeh proposed the concept of fuzzy sets in 1965 [1], fuzzy sets theory has become a popular and fast-growing research topic. Fuzzy sets theory is used in many fields to solve practical problems in the real world. The applications of fuzzy sets theory range from consumer electronics to medical diagnosis. One of the most successful applications is fuzzy control. The first fuzzy control steam engine was produced in 1975 [2]. Since then, in literature, one can find many fruitful examples of fuzzy control applications. However, there exists one problem: construction of a workable fuzzy controller.

A fuzzy controller is composed of 4 elements: fuzzification, defuzzification, the inference engine, and the rulebase. Basically speaking, the processes of fuzzification, defuzzification, and inference engine are easily to establish because there are certain formulae to follow. Only the construction of the rulebase could be varied with different usage. The rulebase is usually derived from experts' experiences. Nevertheless, only in few control problems one can find so-called experts. Even though, the transformation of experts' experience into a linguistic rulebase is also a difficult job. Learning schemes seem a good way to solve such a problem.

Michie [3] divided the concerned space into several boxes. The learning machine helped the boxes to adjust their output according to the behaviors of the system. Barto and Sutton [4] used the similar structure as Michie's boxes system. They used associate critic elements (ACE)

and associate search elements (ASE) to evaluate the effect of the default control effort and correct the output of each box. By the method called "reinforcement learning", the controller can learn how to control a complex plant. Anderson [5] used two neural nets to substitute the functions of ASE and ACE. He called them "evaluation network (EN)" and "action network (AN)". C. C. Lee [6] also used the scheme of reinforcement learning. His controller's structure is the same as Barto's controller but he used fuzzy rules to control the plant. His ASE is the combination of a fuzzy controller and an associate learning neuron (ALN). ALN adjusted the membership function of the output of each rule. Berenji [7][8] combined Anderson's work with a fuzzy controller. His fuzzy controller is also a neural network. This neural net learned the behavior of the fuzzy controller. C. T. Lin and G. C.-S. Lee [9] also developed a hybrid fuzzy controller. This controller uses two learning algorithm, backpropagation and self-organization to construct a neural network. It succeeded to create a new rulebase and make a car pass a curve road through learning. Besides, several other methods have been proposed for the construction the controller by self-learning.

These self-learning schemes indeed solve some problems caused by the construction of rulebases. However, the methods often entangle with complex computations or many meaningless weighting values. In fact, the learning process of a human is usually complex but meaningful and purposeful. The easiest learning process is punishments and rewards. Punishments make clear what should not be done. Rewards let one know what he or she could do or even do better. If the concept of punishments and rewards could be applied to the construction of a fuzzy controller's rulebase, it should make the construction of a rulebase reasonable and improve the efficiency of building the rulebase.

In this paper we propose a heuristic algorithm to realize the concept of punishments and rewards. In section II, details about the heuristic criterion algorithm are given. How to add this criterion element to a fuzzy controller is explained in Section III. Section IV contains the simulation results of the inverted pendulum and a second order system control problem. Our conclusions are given in the last section.

2. The Heuristic Criterion Algorithm

Let's consider a plant, whose control purpose is to keep balance in zero state, position zero and velocity zero. When the previous states, velocity and position, are both positive large, the position of the plant's next states will become further from zero without any control signal. Obviously, the control force should let the velocity move towards negative to slow down the increase in position or even make the position moving reversely. If the controller-generating force makes velocity move toward negative, then this action should be encouraged, otherwise, and it should be punished..

Based on the above-mentioned thoughts, we develop a heuristic algorithm. At first, we have to define what the punishment is and what the reward is. The practical method to realize these concepts is to quantify them into small real number. Thus we define them in the domain [-1,1]. Positive number means a reward and negative number means a punishment.

For a second-order system, the concerned area is divided into four blocks: (position, velocity), (+,+), (+,-), (-,+), and (-,-). Through appropriate transformation, the velocity and position could be the difference between the target position and the default position, and its change rate.

They are so called error and error dot. In the following text, we will use error and error dot to replace the position and the velocity

In (+,+) and (-,1-) domains, we have the following criterion functions:

$\alpha = \frac{\dot{e}_{t+1} - \dot{e}_t}{|\dot{e}_t|}$ e =error

In domain (+,+)

$\beta = \exp(3(\alpha - 0.5))$ if $\alpha <= 0$

$\beta = -\exp(3(\alpha - 0.5))$ if $0 < \alpha <= 0.5$

$\beta = -1$ if $\alpha > 0.5$

In domain (-,-)

$\beta = -1$ if $\alpha < -0.5$

$\beta = -\exp(-3(\alpha + 0.5))$ if $-0.5 <= \alpha < 0$

$\beta = \exp(-3(\alpha + 0.5))$ if $\alpha >= 0$

Here we use error (e) to represent position and $\dot{e}$ is the velocity ,change of error or error dot. Error is defined as the difference between the default position and desired position (target). β is the criterion generated by the criterion mechanism. t and t+1 mean the time sequence. The generation of criterion is related to the previous state and current state. The difference between $\dot{e}_t$ and $\dot{e}_{t+1}$ is divided by the absolute value of $\dot{e}_t$. This normalization is used to express the relativity of the criterion. No matter how large or small $\dot{e}$ is, only the amplitude of change in relation to $\dot{e}$ is meaningful. The design principle behind these functions is exactly punishments and rewards.

In (-,+) and (+,-) domains, the criteria are more complex. In the domain(+,-), we have the following criterion functions:

$\alpha = \frac{\dot{e}_{t+1} - \dot{e}_t}{|\dot{e}_t|}$

$|\dot{e}:e| \le 0.1$

$\beta = -1$ if $\alpha \ge 0.2$

$\beta = -0.5 - 2.5 \times \alpha$ if $0 \le \alpha < 0.2$

$\beta = 0.5 + \alpha$ if $-1 \le \alpha < 0$

$\beta = -0.5$ if $\alpha < -1$

$0.1 < |\dot{e}:e| \le 0.5$

$\beta = -1$ if $\alpha \ge 0.6$

$\beta = -1 - 2.5 \times (\alpha - 0.6)$ if $0.2 \le \alpha < 0.6$

$\beta = 0$ if $-1 \le \alpha < 0.2$

$\beta = -0.2 + 0.2 \times (\alpha + 2)$ if $-2 \le \alpha < -1$

$\beta = -0.2$ if $\alpha < -2$

$0.5 < |\dot{e}:e| \le 1$

$\beta = -0.5$ if $\alpha \ge 1$

$\beta = 0.5 - \alpha$ if $0.5 \le \alpha < 1$

$\beta = 0$ if $0 \le \alpha < 0.5$

$\beta = 2 \times \alpha$ if $-0.5 \le \alpha < 0$

$\beta = -1$ if $\alpha < -0.5$

$1 < |\dot{e}:e| \le 2$

$\beta = -0.5$ if $\alpha \ge 1$

$\beta = 0.3 - 0.8 \times \alpha$ if $0 \le \alpha < 1$

$\beta = 2 \times \alpha$ if $-0.2 \le \alpha < 0$

$\beta = -0.4$ if $\alpha < -0.2$

$2 < |\dot{e}:e| \le 10$

$\beta = 0$ if $\alpha \ge 1$

$\beta = 0.6 - 0.6 \times \alpha$ if $0.5 \le \alpha < 1$

$\beta = 0.3$ if $0 \le \alpha < 0.5$

$\beta = -0.4 + \alpha$ if $-0.2 \le \alpha < 0$

$\beta = -0.6$ if $\alpha < -0.2$

$10 < |\dot{e}:e|$

$\beta = 0$ if $\alpha \ge 1$

$\beta = 0.6 - 0.6 \times \alpha$ if $0.5 \le \alpha < 1$

$\beta = 0.3$ if $0 \le \alpha < 0.5$

$\beta = -0.6 + \alpha$ if $-0.4 \le \alpha < 0$

$\beta = -1$ if $\alpha < -0.4$

In area (-,+), the criterion functions are symmetric to the ones in area(+,-). The strategy is to keep the state in this area and not move towards extreme situation. In figure 2, in the first and second domains of the area(+,-), when the error dot of the next state moves towards positive, then the criterion mechanism will give more serious punishments and hope the controller to make the error dot move towards negative. On the contrary, the states in the 5th and 6th parts are hard to keep them in the area. The natural trend of the states is towards area(-,-). The criterion mechanism will not punish the controller because of its action. Moreover, the criterion mechanism will encourage the controller to make error dot move towards positive.

3. The Structure of the Heuristic Criterion Self-learning Fuzzy Controller

We have made some changes in the structure of the fuzzy controller in this paper. Every rule in the rule base will be simplified into one round cell in the concerned

space. And the parameters of a rule are also simplified into center of the cell, radius of the cell, and output of the cell. Figure 1 shows the basic structure of this kind of cellular fuzzy controller.

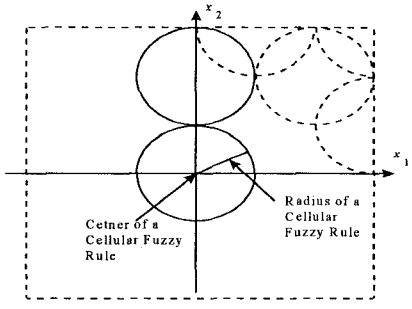

Fig. 1 A cellular fuzzy rule

In the cellular fuzzy controller, the fuzzification process slightly differs from traditional fuzzy controllers. At first the fire strength depends on the distance between the rule center and the states :

$$f\_s = \frac{r - \sqrt{(x_1 - c_1^n)^2 + (x_2 - c_2^n)^2}}{r} \qquad (1)$$

where f_s means fire strength of the rule, and r is the radius of the rule. (x_1, x_2) is the current state. (c_1^n, c_2^n) is the center of the n-th rule. Of course if, the distance between the default states and the rule center is larger than the rule radius, then the fire strength of the rule will be zero. The states will be normalized before they are sent into the controller. The output of one rule is only one crisp number not a fuzzy set. The defuzzification we used is like weighting sum :

$$F = \frac{\sum_{n=1,\ldots,N} f\_s_n \times f_n}{\sum_{n=1,\ldots,N} f\_s_n}, \qquad (2)$$

where F is the final output of the fuzzy controller. f_n means the nth rule output.

At the beginning of the time period (t-1), the system sensor will feed the plant states at the end of time period (t-2) back to the controller and the criterion mechanism. The fuzzy controller will send a control signal to the plant according to the states. The plant acts under this control signal at the time period (t-1). The criterion mechanism will receive states at the end of time period (t-1). Then it will evaluate the fitness of the control action adopted in time period (t-1) according to the plant states at the end of time period (t-2) and (t-1). It sends the β signal as the result of evaluation. The output of rules will be adjusted according to the following formula:

$$f_{t-1}^n = f_{t-2}^n + f\_s_{t-2}^n \times \gamma(t) \times \beta \times |x_1(t-2)| \times sign(x_2(t-1) - x_2(t-2)) \times |F_{t-2}|$$
$$\text{n=1,...,N} \qquad (3)$$

f_t^n means the output of the nth rule after time period (t). $f\_s_t^n$ represents the fire strength of the nth rule at time period (t). $\gamma(t)$ is the learning coefficient. $x_1(t)$ is the error at time t and $x_2(t)$ is the error dot at the beginning of the time period (t). F_t is the output force at the time period (t). The adjusted rule outputs will be the bases of controller output at the time period (t). In (10), the sign function is used to express the moving direction of status' error dot.

4. Simulation Results

In the inverted pendulum problem, we will concentrate on the angle control. The basic data of our cart and pole are as follows: length of the pole is 0.5m; mass of the pole is 0.3kg; mass of the cart is 0.5kg; friction coefficient of pole is 0.000002; friction coefficient of cart is 0.0005; The concerned area: the angle ranges from -12 deg to 12 deg and the angular velocity range from -50 deg/sec to 50 deg/sec.

The learning coefficient is :

$$20 \times \frac{10}{10 + \text{trial}}, \qquad (4)$$

where trial is the number of training process, trial = 1,...,50.

The concerned area is divided into 5*5=25 cells.

In the learning process of the inverted pendulum, the time interval is 0.02 second. The initial rules' outputs are generated at random and the amplitudes are limited between 10 Nts and -10 Nts. The number of learning processes is 50. The learning time steps in each learning process are 1000. The initial point of each learning process is also generated at random. That will reduce the dependence of a constant initial point.

Figure 2 shows the change of the outputs. Figure 3 is the system states' trajectories during the 10th process. Figure 4 shows the control test of four different initial conditions. In this test, we choose four different initial points and take off the learning part of the controller. The result is satisfactory. This controller indeed can achieve the request.

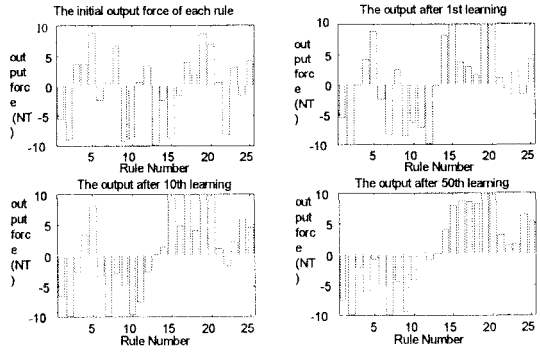

Figure 2. The change of the outputs of each rule

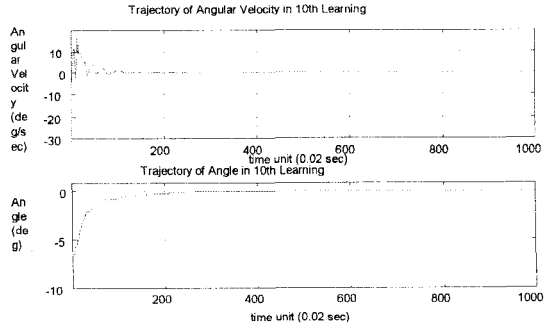

Figure 3. The system trajectory in the 10th learning process

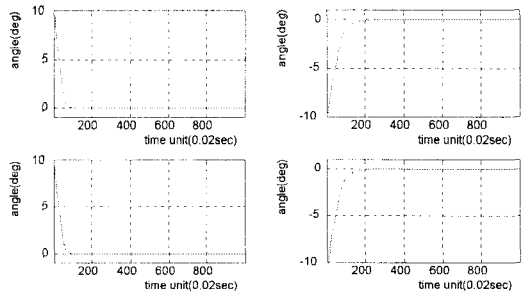

Figure 4. The four test results of the after-learned fuzzy controller

The stable second-order plant we used is :

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -3 & -2 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \qquad (5)$$

The controller structure is similar to the inverted pendulum except that the output forces locate between -40 Nts and 40 Nts. The time interval is 0.05 second. The number of learning processes decreases to 20 but the number of the learning time step increases to 2000. The learning coefficient is :

$$60 \times \frac{3}{3 + \text{trial}}, \qquad (6)$$

where trial is the number of learning processes. Figure5 shows the change of the outputs. Figure 6 shows our control tests of the controller after learning. The performance is also good.

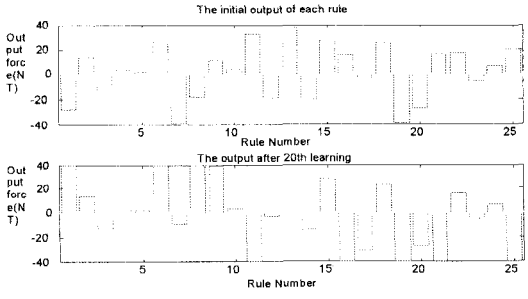

Figure 5. The change of the output of each rule

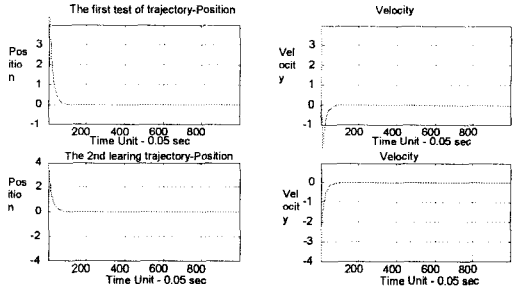

Figure 6. Two tests of the after-learned fuzzy controller

V. Conclusions

The heuristic criterion algorithm indeed helps the self-learning fuzzy controller to establish a usable rulebase. The performance of the fuzzy controller is good. But there are no mechanisms to inform the controller when it should stop learning in the heuristic algorithm. This criterion only provides information of the correctness on the control action but did not supply more information about the learning target. So the oscillation around the equilibrium is possible and the learning efficiency is poor. How to increase the learning effects will be our next challenge.

The learning process in the heuristic algorithm is reasonable and the performance is also good. But too many criterion functions reduce its capability. In fact, these functions should be integrated into a complete mechanism not a separate function. Using a fuzzy system to simulate the action of the heuristic algorithm will be a usable method. Besides, in the heuristic algorithm, the adjustable part of a rule is only the output of a rule. The center of the cell, and the radius of the cell should be also adjustable. Moreover, this algorithm did not cover the consideration of time-delay system. In a time-delay system, the current performance of the system may relate to an action taken several steps before. Maybe the algorithm should take the history of punishments and rewards into account. In the continuous work, we hope to develop a self-generating fuzzy controller through a fuzzy supervisor and add a history-considering mechanism to the controller. That will be our future work.

References

[1] L. A. Zadeh, uzzy sets', *Information and Control*, vol. 8, pp. 338-353, 1965

[2] E. H. Mamdani, pplication of fuzzy algorithms for control of simple dynamic plants', *Proc. IEE* 121(12), pp.1585-1588, 1974

[3] D. Michie and R. A. Chambers, ''Boxes' as a model of pattern-formation', in *Toward a Theoretical Biology,* vol. 1, Prolegomena, C. H. Waddington, ed. Edinburgh Univ. Press, 1968, pp. 206-215

[4] A. G. Barto, R. S. Sutton, and C. W. Anderson, euronlike adaptive elements that can solve difficult learning control problems', *Trans. SMC,* vol. 13, no. 5, 1983, pp. 834-846

[5] C. W. Anderson, trategy learning with multilayer connectionist representation', Tech. Rep. TR87-509.3, GTE Laboratories Inc., May 1988

[6] C. C. Lee, elf-learning rule-based controller employing approximate reasoning and neural-net concepts', *Int. J. Intelligent Systems,* vol. 6, pp. 71-93, 1991

[7] H. R. Berenji, reinforcement learning-based architecture for fuzzy logic control', *Int. J. Approximate Reasoning,* no. 6, 1992, pp.267-299

[8] H. R. Berenji and P. Khedkar, earning and tuning fuzzy logic controllers through reinforcements', *IEEE Trans. Neural Networks,* Sep. 1992, pp. 724-740

[9] C.-T. Lin and C.-S. G. Lee, eural-network-based fuzzy control and decision system', *IEEE Trans. Computer,* Dec. 1991, pp.1320-1336