The Study on Inconsistent Rule Based Fuzzy Logic Control using Neural Network

Jae-Soo Cho, Dong-Jo Park and Z. Bien

Department of Electrical Engineering
Korea Advanced Institute of Science & Technology (KAIST)
373-1 Kusong-dong, Yusong-gu, Taejon 305-701, Republic of Korea
Tel : +82-42-869-8038, Fax : +82-42-869-8038
E-Mail : jaesoo@tercel.kaist.ac.kr

Key Word : Inconsistent rules, Adaptive rule weights, Neural network

Abstract

In this paper is studied a method of fuzzy logic control based on possibly inconsistent if-then rules representing uncertain knowledge or imprecise data. In most cases of practical applications adopting fuzzy if-then rule bases, inconsistent rules have been considered as ill-defined rules and, thus, not allowed to be in the same rule base. Note, however, that, in representing uncertain knowledge by using fuzzy if-then rules, the knowledge sometimes can not be represented in literally consistent if-then rules.

In this regard, when it is hard to obtain consistent rule base, we propose the weighted rule base fuzzy logic control depending on output performance using neural network and we will derive the weight update algorithm. Computer simulations show the proposed method has good performance to deal with the inconsistent rule base fuzzy logic control. And we discuss the real application problems.

1 Introduction

Previous researches revealed that fuzzy control has, in many cases, a superiority over the conventional controller for automating very complex and/or poorly-defined processes. In most of applications adopting fuzzy if-then rule bases, inconsistent rules are often considered as ill-defined rules and are not allowed to form the same rule base. It has been believed that we should resolve inconsistency in the rule base before the rules are processed for inference and that the obtained inconsistent rules should be discarded or modified to make the rule base consistent. However, we know that, though inconsistent, as far as they are heuristically meaningful, each rule in the rule base may contain some form of informative knowledge so that discretion of good rules from bad rules in the rule base may not be an easy task. As the approaches to resolve inconsistency in the obtained inconsistent if-then rules, one may discard inconsistent rules[1] examining the rules according to their contribution to inference. However, since each rule in the rule base may contain informative knowledge, measuring the contribution of the rules in the rule base and discarding bad rules from good rules may not be an easy task. Also, we can use a trading off concept to make the given inconsistent rules. Trading off the consequents of the inconsistent rules can be considered by using multi attributive decision making technique[2]. The new approach for resolving inconsistency in the rule base was proposed[3]. This new approach can be useful when there involves much uncertainty in extracting control rules and it is very difficult to obtain totally consistent rule base. Now we regard inconsistent rules as the ones which contain informative knowledge. In this paper, based on the fact that our knowledge can not always be represented in literally consistent way, it is claimed that inconsistent rules can be processed for inference at the same time. In this regard, when it is hard to obtain consistent rule base, it can be a possible approach that we use the weighted rule bases depending on output performance. And we can change each weights adaptively using the neural network simulator.

2 Problem Statement

2.1 Inconsistent Rules

The fuzzy rule base can be obtained by the qualitative knowledge about the plant or by obserbing the actions of the human operator. Based on the qualitative knowledge, we first divide the input and output spaces into fuzzy regions, and assign each region a fuzzy membership function and then map input regions into output regions by using if-then rules. Each input fuzzy region is mapped into only one output fuzzy region and, therefore, we may say that the rule base made in this manner is a consistent rule base. Many successful results have been obtained by using such a normal rule base. However, there can be many situations where the normal rule base is not enough to express our knowledge specially when the knowledge is uncertain. In this section, we deal with the rule base including inconsistent rules.

Examining fuzzy rule sets, one may imagine that various forms of inconsistencies can exist in a group of rules. In fuzzy logic , we say that a set of two or more *"If-then"* rules are inconsistent if, for the same antecedents(If part), the corresponding consequents(then part) are not the same[4]

Two or more rules are said to be explicitly inconsistent if they have identical antecedents(If-part) but have different consequents(then-part). For example, the following rules R1 and R2 are explicitly inconsistent:

R1 : If (error is big) and (change of error is small), then (output is big)

R2 : If (error is big) and (change of error is small), then (output is small).

Other than the explicitly inconsistent rules, some implicit inconsistency may exit even in the normal rule base. The overlapping of the antecedents in fuzzy sets imply some inconsistencies to some extent. When there are many variables to refer to, if we make two control rules with partial variables and the two rules use different partial variables in their conditional parts, then the two rules can be considered to be inconsistent. For a simple example , consider two rules, each with only one antecedent condition even though there are two variables to use for the rules:

R3 : If (error is small) then (output is big)

R4 : If (change of error is big) then (output is small).

It seems that the rules R3, R4 are not inconsistent with each other. However, the two rules R3 and R4 can be made inconsistent by appending fuzzy description for missing antecedent variables. Specifically, suppose there are another rules in the rule base which read as:

R5 : If (error is small) and (change of error is big) then (output is big)

R6 : If (error is small) and (change of error is big) then (output is small).

Note that R5 and R6 are more specific versions of the rules R3 and R4, respectively. But R5 and R6 are explicitly inconsistent. Also, if we have R6 instead of R4, then R3 and R6 are explicitly inconsistent. If we look at this form in the view point of overlapping, the antecedents of the rules R3 and R4 are also overlapped with each other. Figure 1 shows the overlapping of the two rules. There may be other kinds of inconsistent rules when we use the rules which are independently made by multiple sources or from different view points.

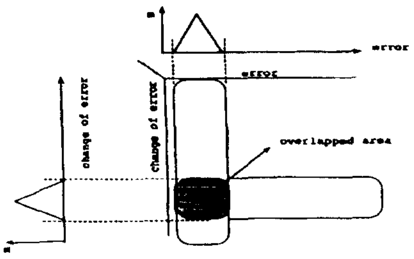


Figure 1: Overlapping of the conditional parts of the rules R3 and R4

2.2 Example simulations

The uncertainties and the imprecision in extracting control rules for fuzzy control can make the rules inconsistent and conflicting with each other, but it would be a very difficult and tedious task for a rule designer to discard or modify one rule or another in consideration of the confidence level to resolve inconsistencies of the rules under consideration. At first, we show the bad performance when you use the inconsistent rule based fuzzy controller to control a simple well-known plant. Through this simple simulation, we can easily know the effects of inconsistent rules to the plant output performance.

Example : Plant Model $H(s) = \frac{1}{S(S+1)}$.

Figure 2 shows the fuzzy rules and their membership functions. We use 13 rules and additional 3 inconsistent rules purposely as follows:

Original Rule 1 : If E is ZO and ΔE is NM, then OUT is NM

Inconsistent Rule 1 : If E is ZO and ΔE is NM, then OUT is NS

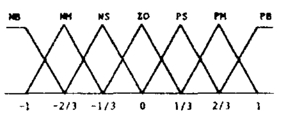
Original Rule 2 : If E is NS and ΔE is ZO, then OUT is NS

Inconsistent Rule 2 : If E is NS and ΔE is ZO, then OUT is PS

Original Rule 3 : If E is PM and ΔE is ZO, then OUT is PM

Inconsistent Rule 3 : If E is PM and ΔE is ZO, then OUT is NM

		ΔE						
		NB	NM	NS	ZO	PS	PM	PB
	NB				NB			
	NM				NM			
	NS				NS			
E	ZO	NB	NM	NS	ZO	PS	PM	PB
	PS				PS			
	PM				PM			
	PB				PB			



E : Error ΔE : Change of Error ZO : Zero
NS : Negative Small NM : Negative Medium NB : Negative Big
PS : Positive Small PM : Positive Medium PB : Positive Big

Figure 2: Basic 13 rules and membership function

Figure 3 shows a comparison between original 13 rule based control and 16 rule based control. We use a step input as the reference and simulate under the same conditions except the rule number. We can see a little bit difference between them and how important it is to get consistent rules. Through the above simulation results, we

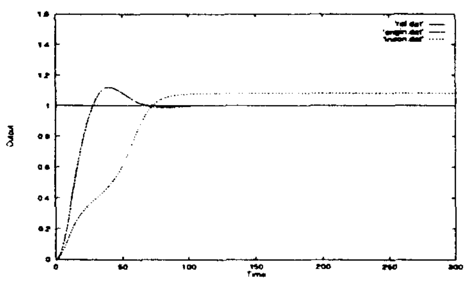


Figure 3: The comparison between correct rules and inconsistent rulse(1)

can check that the inconsisten rule base fuzzy logic controller has a bad output performance. We will use the above plant model and rule bases to test following proposed method.

3 Weighted Rule Base Fuzzy Logic Control

3.1 Basic Structure

One of the possible approaches to design a controller for a very complex plant controlled by humans is to design a controller emulating the control actions of the human by using sampled I/O data from the actions of the human. However it is usually very difficult or tedious to obtain the sampled I/O data of the human actions and, even if we manage to obtain the data, they can be imprecise or noisy. In this case, the design of the controller is basically a function approximation problem by using noisy or imprecise sampled I/O data. It is well known that the artificial neural networks can be chosen to approximate the complex functions. In this weighted rule base fuzzy logic control, we use the neural network to emulate the plant. And this emulating neural network will be used for weight update algorithm.

Figure 4 shows the basic structure of proposed weighted rule base fuzzy logic control. This structure have additional two blocks to the original fuzzy logic control. One block is a neural emulating part and the other is rule weight update algorithm part. And the fuzzy inference block and defuzzifier block are changed to the weighted fuzzy inference and weighted defuzzifier.

Our objective is very simple. We want to impose weights on all rules depending on the performance measures adaptively and update the weights using neural network emulator using backpropagation algorithm. Therefore, the inconsistent rule weights will be adaptively changed to increase the performance measure.

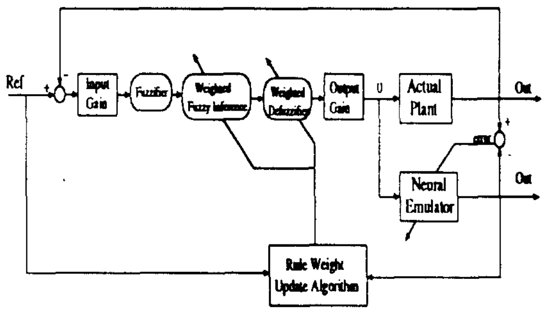


Figure 4: The proposed basic structure

The overall procedure to the proposed method is as follows.

Step 1 : Identification of the actual plant using neural network using off line I/O data.

Step 2 : Initial rough fuzzy rules generation.

Step 3 : Put the initial rules weights one.

Step 4 : Apply control input and obtain outputs.

Step 5 : Calculate the performance measure.

Step 6 : Weight update using neural emulator.

Now, we will explain the weighted rule based fuzzy logic control method. The following if-then rules can characterize the proposed weighted rule base.

Rule 1 : If (x_1 is $\tilde{A}_1$) and (x_2 is $\tilde{B}_1$),$\cdots$, then (u is $W_1 * \tilde{U}_1$)

Rule 2 : If (x_1 is $\tilde{A}_2$) and (x_2 is $\tilde{B}_2$), $\cdots$, then (u is $W_2 * \tilde{U}_2$)

$\vdots$

Rule i : If (x_1 is $\tilde{A}_i$) and (x_2 is $\tilde{B}_i$), $\cdots$, then (u is $W_i * \tilde{U}_i$)

So, if we use the center of area/gravity deffuzzification method, the defuzzified output is as follows:

$$u^* = \frac{\sum_{i=1}^{i=l}(W_i \cdot u_i)\cdot m_U(u_i)}{\sum_{i=1}^{i=l} m_U(u_i)}. \quad (1)$$

3.2 Rule Weight Update Algorithm

The mathematical model for the neural emulator in Figure 4 is shown below:

$$O_N(k) = \sum_j V_j^O X_j(k), \quad X_j(k) = f(S_j(k)), \quad (2)$$

$$S_j(k) = \sum_i V_{ij}^I I_i(k), \quad f(S_j(k)) = \frac{1}{1+e^{-2*S_j(k)}}, \quad (3)$$

where $O_N(k)$, $V^O(k)$, V_{ij}^I, and $I_i(k)$ are respectively neural network output, output weight, input weight and input. f($\cdot$) is a sigmoid function. Let Ref(k) and out(k) be the reference and actual responses of the plant, then a performance function for training cycle for the weight update can be defined as

$$J(k) = \frac{1}{2}(Ref(k) - out(k))^2. \quad (4)$$

The gradient of error in performance function with respect to the weight vector W is represented by

$$\begin{aligned}\frac{\partial J(k)}{\partial W} &= -(Ref(k) - out(k)) \cdot \frac{\partial out(k)}{\partial W} \\ &\equiv -(Ref(k) - out(k)) \cdot \frac{\partial O_N(k)}{\partial W}, \quad (5)\end{aligned}$$

where $out(k) \equiv O_N(k)$. The output gradient $\frac{\partial O_N(k)}{\partial W}$ needs to be computed. The output gradients with respect to the weight are given by chain rule,

$$\frac{\partial O_N(k)}{\partial W(k)} = \frac{\partial O_N(k)}{\partial u^*(k)} \cdot \frac{\partial u^*(k)}{\partial W}, \quad (6)$$

$$\begin{aligned}\frac{\partial O_N(k)}{\partial u^*(k)} &= \sum_{i=1}^{i=NH} \frac{\partial O_N(k)}{\partial X_i(k)} \cdot \frac{\partial X_i(k)}{\partial u^*(k)} \\ &= \sum_{i=1}^{i=NH} V_i^O \cdot \frac{\partial X_i(k)}{\partial u^*(k)},\end{aligned}$$

$$\frac{\partial X_i(k)}{\partial u^*(k)} = \dot{f}(S_j(k)) \cdot \frac{\partial S_j(k)}{\partial u^*(k)} = \dot{f}(S_j(k)) \cdot V_{ij}^I$$

$$\frac{\partial O_N(k)}{\partial u^*(k)} = \sum_{i=1}^{i=NH} V_i^O \cdot \dot{f}(S_j(k)) \cdot V_{ij}^I, \quad (7)$$

$$\frac{\partial u^*(k)}{\partial W} = \frac{\sum_{i=1}^{i=l} u_i \cdot m_U(u_i)}{\sum_{i=1}^{i=l} m_U(u_i)}. \quad (8)$$

The weights can now be adjusted by the following gradient method, i.e., the update rule of the weights becomes

$$W(k+1) = W(k) - \eta \cdot \frac{\partial J(k)}{\partial W(k)}, \quad (9)$$

where η is a learning rate.

4 Simulation Results

To check the performance of the proposed method and learning algorithm, we have performed a lot of simulations. However, for the sake of convenience, we present here only illustrative example. We have compared inconsistent rule based fuzzy logic controller and the proposed weighted rule based fuzzy logic controller.

Figure 5 and Figure 6 show the simulation results when we simulate *Example*. We simulated with all the same conditions to compare the output results. As shown in Figure 5, 6, the proposed method gives better output performance in case of inconsistent rule base fuzzy control. And Figure 7 show the simulation result when we fixed the rule weights at Time = 1200 after learning.

Let's check the weight changes. At first, in the *Example* simulation, Figure 8, 9, 10 show respectively original rule 1 weight and inconsistent rule 1 weight, original rule 2 weight and inconsistent

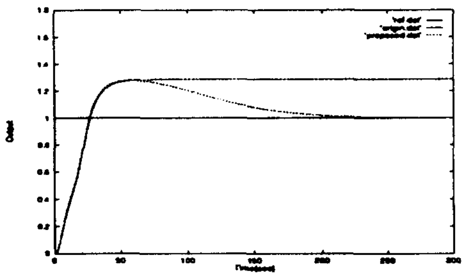


Figure 5: Step reference input simulation result

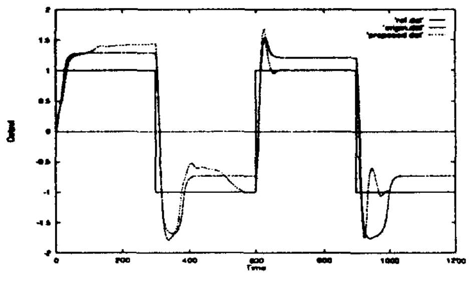


Figure 6: Square reference input simulation result

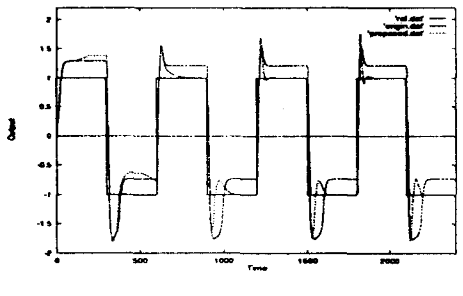


Figure 7: Fixed rule weight after learning until Time = 1200

rule 2 weight, original rule 3 weight and inconsisten rule 3 weight respectively. Figure 11, 12 also show the changed rule weights and unchanged rule weights respectively.

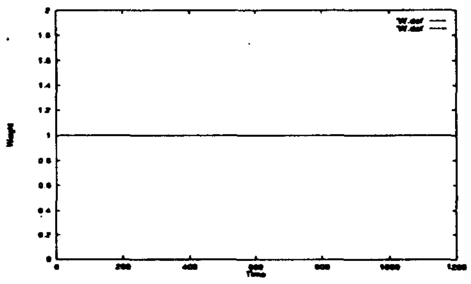


Figure 8: Weight comparison between inconsistent rule 1

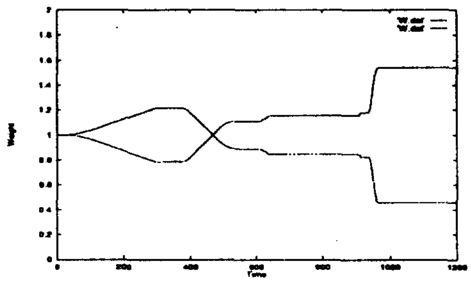


Figure 9: Weight comparison between inconsistent rule 2

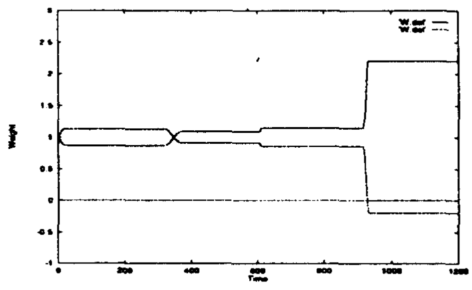


Figure 10: Weight comparison between inconsistent rule 3

Through the weight changes, we can find out two important phenomena. One is inconsistent rule weight generally decrease more than correct rule weights. The other is correct rule weights generally unchanged. This information will be very helpful to make the rough rules in unknown complex plants.

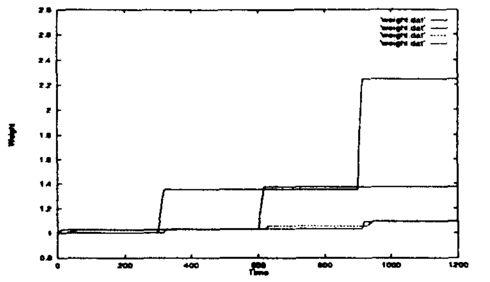


Figure 11: Changed weights(somewhat related inconsistent rules)

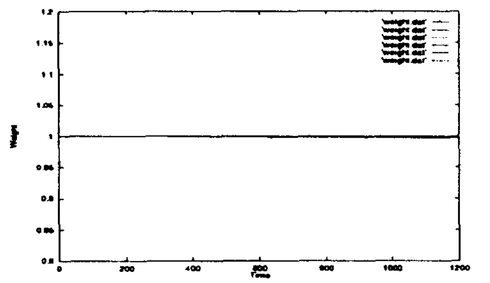


Figure 12: Unchanged Weights(consistent rule)

5 Discussion and Conclusions

Inconsistent rules can be obtained when the control actions of experts are directly and naively represented. However, such inconsistent rules can give trouble for conventional fuzzy logic control because of the fat shape dominance phenomenon. Of course, the fuzzy logic control with the conventional inference method can work well, if the rules and linguistic terms are modified via trial and error so that there are no inconsistent rules. However, it can be a difficult and tedious task for rule designers. In order to overcome such difficulty, we have proposed a method of weighted rule based fuzzy logic control with neural network. And we derived the weight update algorithm with neural network backpropagation algorithm. The computer simulations show that the proposed method is more effective than the conventional method in case of inconsistent rules. But, in real application, we should consider the stability problem when we want to apply the method to the plant in real time. But this method can be very useful to design rough rules for the control of complex unknown plants, namely, It is very helpful in rule generation if we know the weight change information. change information. And this method can be used in the real plant with fixed weight after off-line learning.

References

[1] L.X. Wang and J.M. Mendel, "Generating Fuzzy Rules by Learning from Example," in *IEEE Trans, Syst. Man. Cybern,* vol. 22, No. 6, pp. 1414-1472, 1992.

[2] M.L. Hussein and M.A. Abo-Sina, "Decomposition of multiobjective programming problems by hybrid fuzzy dynamic programming," in *Fuzzy Sets and Systems,* vol. 60, pp. 25 -32, 1993.

[3] W. Yu, and Z. Bien, "Design of fuzzy logic controller with inconsistent rule base," in *Journal of Intelligent and Fuzzy Systems,* vol. 2, pp 147-159, 1994.

[4] R.R. Yager and H.L. Larson, "On discovering potential inconsistencies in validating uncertain knowledge bases by reflecting on the Input," in *IEEE Trans, Syst, Man, Cybern,* vol. 21, pp. 790-801, 1991.

[5] K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamic Systems Using Neural Networks," in *IEEE Trans. Neural Networks,* vol. 1, No. 1, pp. 4-27, Mar. 1990.

[6] Chao-Chee Ku, Kwang Y. Lee, "Diagonal Recurrent Neural Networks for Dynamic Systems Control,"*IEEE Trans. On Neural Networks.* Vol 6. No. 1, January 1995.