Fuzzy Logic Control Based on Weighted Rules using Neural Network

Jae-Soo Cho and Dong-Jo Park

Department of Electrical Engineering,
Korea Advanced Institute of Science and Technology(KAIST),
373-1 Kusong-dong, Yusung-gu, Taejon, Republic of Korea 305-701
Tel: +82-42-869-3438; Fax: +82-42-869-3410
e-mail:djpark@eekaist.kaist.ac.kr

ABSTRACT

In this paper a method of fuzzy logic control based on possibly inconsistent if-then rules representing uncertain knowledge or imprecise data is studied. When it is hard to obtain consistent rule bases, we propose the fuzzy logic control based on weighted rules depending on output performances using the neural network and we derive the weight updating algorithm. With the final weight change informations, we can make better decisions by taking into consideration conflicting rules. Computer simulations show the proposed method has good performance to deal with the inconsistent rule base in the fuzzy logic control. And real application problems are also discussed.

1 Introduction

One of the major merit of fuzzy control may be that we can use our heuristic or qualitative knowledge to make the fuzzy if-then rules, which can be obtained by observing humans in the loop or by elaborating the dictations of the humans themselves. In spite of the merit mentioned above, making the rule base is not always an easy task. Since the rule base used in the fuzzy control is often a repository of human knowledge and much of human knowledge is uncertain and imprecise in nature, it is usually the case that a collection of rules and facts representing the knowledge may not be totally consistent.

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 a simple job. We can use a trade-off concept to make the given inconsistent rules consistent. Useful trading-off of the consequents of the inconsistent rules can be considered by using the multi-attributive decision making technique [2]. Another approach for resolving inconsistency in the rule base was proposed in [3]. This approach can also be useful when there involves much uncertainty in extracting control rules but it is very difficult to obtain a totally consistent rule base.

In this paper, when it is hard to obtain a consistent rule base, it can be a possible approach that we use the weighted rule bases depending on the output performance. And then, we can select the consistent rules by using the rule weights.

2 Problem Statements

Examining fuzzy rule sets, one may imagine that various forms of inconsistencies can exist in a group of rules. Two or more rules are said to be explicitly inconsistent if they have the identical antecedents(If part) but have different consequents(then part) [4]. 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 exist even in the normal rule base. The overlapping of the antecedents in fuzzy sets implies inconsistencies to some extent. When we are going to control a plant with many variables to refer to, it is not always easy to consider all the variables at the same time in making each control rule.

We will show the problems of fuzzy logic control based on an inconsistent rule base in a simple example. Through this example, we can easily know the effects of inconsistent rules to the output performance of the plant.

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

Figure 1 shows the original thirteen fuzzy rules and their membership functions. We use the original rules and additional three inconsistent rules purposely which follow:

Original R1: If E is ZO and $\triangle E$ is NM, then OUT is NM.
Inconsistent R1: If E is ZO and $\triangle E$ is NM, then OUT is NS.
Original R2: If E is NS and $\triangle E$ is ZO, then OUT is NS.
Inconsistent R2: If E is NS and $\triangle E$ is ZO, then OUT is PS.
Original R3: If E is PM and $\triangle E$ is ZO, then OUT is PM.

Inconsistent R3: If E is PM and △E is ZO, then OUT is NM.

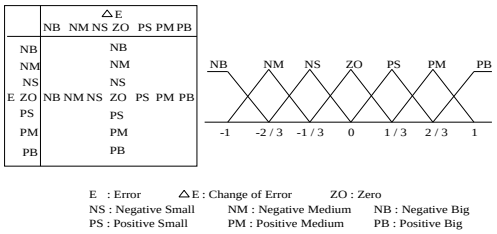


Figure 1: Original thirteen rules and membership functions

Figure 2 shows the comparison between the control performance with the original thirteen rule base and that with the augmented sixteen rule base. We use the step input as the reference and simulate under the same conditions. We can see the performance difference and how important it is to get consistent rules. Through the

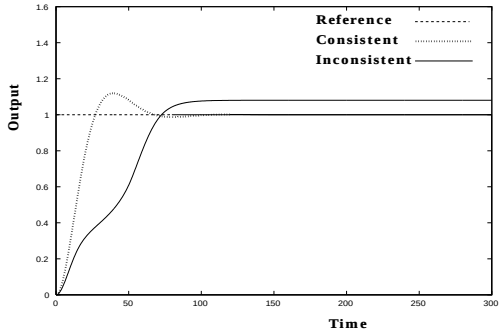


Figure 2: Comparison between correct and inconsistent rule bases

above simulation results, we can show that the fuzzy logic controller with an inconsistent rule base has much deteriorated performance.

3 Fuzzy Logic Control Based on Weighted Rules using Neural Network

3.1 Basic Structure

It is well known that an artificial neural network can be chosen to approximate complex functions. In this fuzzy logic control with a weighted rule base, we will use a neural network to emulate a plant which we want to control. And this emulating neural network will also be used to a weight updating algorithm.

Figure 3 shows the basic configuration of our proposed fuzzy logic control with weighted rule bases. This structure has additional two blocks to the original fuzzy logic control. One block is a neural emulation part and the other is a rule weight updating algorithm part. And the fuzzy inference and the defuzzifier block are changed to the weighted fuzzy inference and the weighted defuzzifier. Figure 4 shows the overall procedure for the

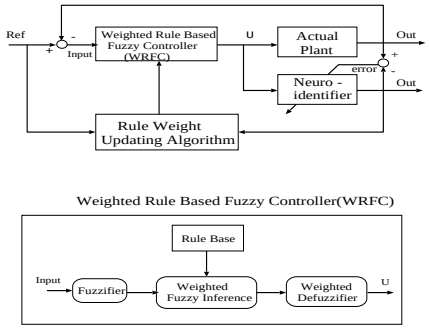


Figure 3: Overall configuration of proposed method

real application of the proposed method.

At first, we need to identify the actual plant using a neural network with off-line I/O data. The problem of identification consists of setting up a suitably parameterized identification model and adjusting the parameters of the model to optimize a performance function based on the error between the plant output and the identification model output. Second, the initial fuzzy rules can be obtained by the qualitative knowledge about the plant or by observing the actions of the human controlling the plant. Examining the initial fuzzy rules, we can imagine that various forms of inconsistencies may exist in a group of rules.

Next, we should apply the test input to the controlled plant to obtain the learning data. After that, calculate the performance measure and update the rule weights using the neural emulator. If the system performance is somewhat good after enough learning, we have two choices. One is to sort out the initial rules into two groups of rules such as consistent(good) rules and inconsistent(bad) rules. The other is to fix the rule weights in order to obtain the stable system performance. If we check the rule weights after the initial rule learning, we can know the good rules and the bad rules because the good rule weights should be increased and the bad rule weights should be decreased in order to obtain good system performance.

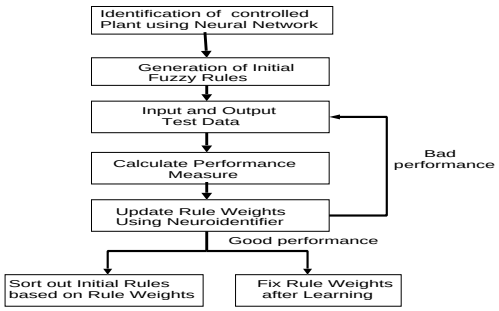


Figure 4: Procedure of proposed method

The following if-then rules can characterize the pro-

posed weighted rule base with m inputs and one output:

Premises(control rules):

Rule 1: If $(x_1$ *is* $\tilde{X_{11}})$ *and* $(x_2$ *is* $\tilde{X_{12}})$, $\cdots$, $(x_m$ *is* $\tilde{X_{1m}})$, *then* $(u$ *is* $W_1 * \tilde{U_1}$ *).*

Rule 2: If $(x_1$ *is* $\tilde{X_{21}})$ *and* $(x_2$ *is* $\tilde{X_{22}})$, $\cdots$, $(x_m$ *is* $\tilde{X_{2m}})$, *then* $(u$ *is* $W_2 * \tilde{U_2}$ *).*

$\vdots$

Rule n: If $(x_1$ *is* $\tilde{X_{n1}})$ *and* $(x_2$ *is* $\tilde{X_{n2}})$, $\cdots$, $(x_m$ *is* $\tilde{X_{nm}})$, *then* $(u$ *is* $W_n * \tilde{U_n}$ *).*

Implication: m inputs are $x_1, x_2, \cdots, x_m$, respectively. W_i, $i = 1, 2, \cdots, n$ are rule weights. Here $\tilde{X_{i1}}$, $\tilde{X_{i2}}$, $\cdots$, $\tilde{X_{im}}$ are fuzzy sets for antecedent linguistic terms such as *big*, *small* or *any* used in the i^{th} rule. The fuzzy sets $\tilde{U_1}$, $\tilde{U_2}$, $\cdots$, $\tilde{U_n}$ represent the consequent linguistic terms.

So, if we use the center of area/gravity defuzzification method, the weighted rule base defuzzification can be obtained like the original one:

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

3.2 Rule Weight Updating Algorithm

At first, an unknown plant is identified by a system identifier, called the neuroidentifier which provides information about the plant to the controller, called the weighted rule based fuzzy controller(WRFC). The mathematical model for the neuroidentifier is shown as:

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

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

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

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

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

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

where $out(k) \equiv O_N(k)$. The output gradient $\frac{\partial O_N(k)}{\partial W}$ is given by the chain rule,

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

where $u^*(k)$ is an input to the neuroidentifier. Applying the chain rule to the term $\frac{\partial O_N(k)}{\partial u^*(k)}$ on the right side of (6),

$$\begin{aligned} \frac{\partial O_N(k)}{\partial u^*(k)} &= \sum_{i=1}^{NH} \frac{\partial O_N(k)}{\partial X_i(k)} \cdot \frac{\partial X_i(k)}{\partial u^*(k)} \\ &= \sum_{i=1}^{NH} V_i^O \cdot \frac{\partial X_i(k)}{\partial u^*(k)}. \end{aligned} \quad (7)$$

Also from (2), (3), and noting that $u^*(k) = I_i(k)$ of (3),

$$\frac{\partial X_i(k)}{\partial u^*(k)} = \dot{f}(S_j(k)) \cdot \frac{\partial S_j(k)}{\partial u^*(k)} = \dot{f}(S_j(k)) \cdot V_{ij}^I. \quad (8)$$

Thus, from (8),

$$\frac{\partial O_N(k)}{\partial u^*(k)} = \sum_{i=1}^{NH} V_i^O \cdot \dot{f}(S_j(k)) \cdot V_{ij}^I. \quad (9)$$

And, from (1),

$$\frac{\partial u^*(k)}{\partial W} = \frac{\sum_{i=1}^{i=l} u_i \cdot m_U(u_i)}{\sum_{i=1}^{i=l} m_U(u_i)}. \quad (10)$$

Finally, we can obtain the output gradient $\frac{\partial O_N(k)}{\partial W}$ from (9), (10). The weights can now be adjusted by a gradient method, i.e., the update rule of the weights becomes

$$W(k+1) = W(k) - \eta \cdot \frac{\partial J(k)}{\partial W(k)}, \quad (11)$$

where η is a learning rate. The equations (2) - (11) define the rule weight updating algorithm of dynamic backpropagation for WRFC.

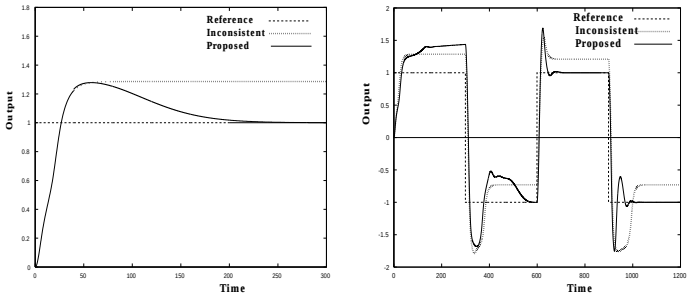
4 Simulation Results

To show the performance of the proposed method, we have performed a number of simulations. However, for the sake of convenience, we present here only an illustrative example. For exact comparison, we will use the controlled plant model which was used for the example plant model in the problem statement section. We have compared the fuzzy logic controller with the inconsistent rule bases and the proposed fuzzy logic controller with the weighted rule bases. Figure 5 (a) and (b) show the simulation results for different reference inputs. We simulated the example under the same conditions to compare the output results. As shown in Figure 5, the proposed method gives better output performance than that of the fuzzy control with the inconsistent rule bases.

Figure 6 shows the simulation result when we fixed the rule weights at time = 1200 after learning. If we examine the changes of the initial rule weights, we will have important informations about the initial rules. We set

the initial rule weights as one. Figure 7(a) and (b) show weight changes of the consistent and the inconsistent rules respectively. As one see the weight changes, the original consistent rule weights increased more than the initial rule weights in order to satisfy the performance function. Additional inconsistent rule weights decreased in comparison with the initial rule weights.

We can find out two important phenomena seeing the above weight changes. One is that the inconsistent rule weights generally decrease, while the consistent rule weights increase. The other is that the changed weight rules are more related to the inconsistency than the unchanged weight rules. And, the rule weights should change in the direction of decreasing the performance function. Those important weight informations will be very helpful to make the consistent rule bases when we have to control the complex unknown plants with the fuzzy logic controller.



(a) Step reference (b) Square reference

Figure 5: Simulation results

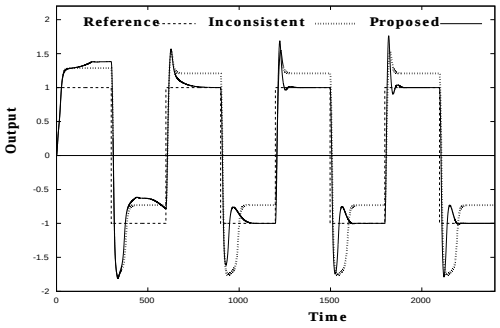
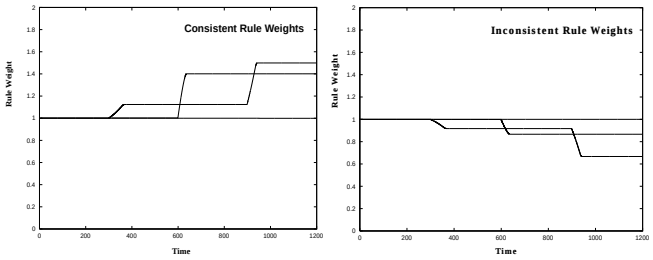


Figure 6: Results of fixed rule weights after learning until time = 1200

5 Discussion and Conclusions

Inconsistent rules can be obtained when the control actions of experts are directly and naively represented. Such inconsistent rules can give trouble for the conventional fuzzy logic control. In order to overcome such an



(a) Consistent rule weights (b) Inconsistent rule weights

Figure 7: Consistent and inconsistent rule weight changes

inconsistency problem, we have proposed a method of fuzzy logic control based on weighted rules with a neural network. And we derived the weight updating algorithm with the backpropagation algorithm. The computer simulations show that the proposed method can handle the fuzzy controller with inconsistent rules and may be very helpful to the rule designer when he can not understand the controlled plant well.

It is very desirable in rule generation if we know the weight change informations. With the final weight change informations, we can make better decisions by taking into consideration conflicting rules. Therefore, this method can be used in the real plant with fixed weight rules or sorting out the inconsistent rules after off-line learning. We will have more simulation studies using other complex plants and analyze the proposed method to compensate deficiencies. We expect this method will have better results to the multiobjective control problems.

References

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

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

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

[4] R.R. Yager and H.L. Larson, (1991). "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.