Robotica (2000) volume 18, pp. 375–380. Printed in the United Kingdom © 2000 Cambridge University Press

A software development system for fuzzy control

Jie Yang, Yingkai Guo and Xin Huang

Institute of Image Processing and Pattern Recognition, Shanghai Jiao-Tong University, Shanghai 200030 (P.R. of China)

(Received in Final Form: October 25, 1999)

SUMMARY

Fuzzy control has been widely applied in industrial controls and domestic electrical equipment. The automatic learning of fuzzy rules is a key technique in fuzzy control. In this paper, a software development system for fuzzy control is presented. Since the learning of fuzzy rules can be seen as finding the best classifications of fuzzy memberships of input-output variables in a fuzzy controller, it can also be seen as the combination optimization of input-output fuzzy memberships. Multi-layer feedforward network and genetic algorithms (GA) can be used for the automatic learning of fuzzy rules. The algorithms and their characteristics are described. The software development system has been successfully used for the design of some fuzzy controllers.

KEYWORDS: Fuzzy control; Rule generation; Multi-layer feed-forward network; Genetic algorithm.

1. INTRODUCTION

Fuzzy logic can represent and deal with uncertainty and fuzziness of knowledge. For many real-world control systems, state equations are difficult to express or are strongly nonlinear, and the accurate mathematical models and constraints of the systems are not precisely known. Fuzzy control[1] is one of the methods most likely to be used to solve such problems. Fuzzy control simplifies the modeling of a control system, and realizes a smooth transformation of states. In classical approaches, fuzzy membership functions are set in advance and fuzzy rules are adjusted iteratively until expected results of control systems are achieved. The whole procedure is usually handled manually. With the wide application of fuzzy control, the objects to be controlled become more and more complicated, but the development periods of fuzzy controllers are expected to be shorter and shorter. A software development system for fuzzy control is useful and urgently needed. Knowledge acquisition is the bottleneck for implementing fuzzy systems. For a complicated fuzzy system, it is very difficult (even for human experts) to find satisfactory fuzzy rules manually. Automatic learning of fuzzy rules is a key technique in a software development system for fuzzy control. The aim of automatic learning fuzzy rules is to realize a smooth transformation between a pair of expected input-output states. It can also be considered as a pattern classification. A multi-layer perceptron network can be used for the automatic learning of fuzzy rules[2–4] because multi-layer feedforward networks can be used as a tool for function approximation to realize smooth input-output data approximation. Automatic learning of fuzzy rules can also be seen as a problem of an optimization combination of input-output states, while the goal is to find the best combination. As an effective technique of optimization, genetic algorithms can be used for the automatic learning of fuzzy rules.[4–6]

2. SOFTWARE DEVELOPMENT SYSTEM FOR FUZZY CONTROL

In our software development system for fuzzy control, a triangular function is selected as the membership function of fuzzy predicates. A MAX-MIN approach is used for the inference of fuzzy rules.

Defuzzification: fuzzy predicates of output variables are defuzzified into their real values according to pre-defined membership functions. After a fuzzy inference by a MAX-MIN approach, the membership values $\mu(A_i)$ i = 1, . . . , n of fuzzy predicates A_i i = 1, . . . , n of an output variable are calculated. The real value of the output variable is:

$$Z_0 = \sum_{i=1}^{n} \mu(A_i) \times Z_i \Big/ \sum_{i=1}^{n} \mu(A_i)$$

where z_i is the center value of the triangular membership function of fuzzy predicates A_i.

For a fuzzy controller with two input variables and two output variables, each variable is divided into three categories of fuzziness (fuzzy predicates). The structure of the fuzzy neural network is described in Figure 1.

2.1. The function menu of the software development tool for fuzzy control

The function menu of the software development tool for fuzzy control is shown in Figure 2. The system function consists of operations (Edit, Save, Print) for the data file *.fuz of a fuzzy controller which records training data, membership functions, evaluation function, fuzzy rules and codes of rules for a chip.

The membership function consists of three subfunctions: a “Variable definition” subfunction defines input and output variables and their fuzzy predicates; a “Function edit” subfunction shows default membership functions and revises them manually; a “Function generation” subfunction is an automatic generation of membership functions according to the distribution of training data and clustering algorithms.

A Rule-editor function edits fuzzy rules manually or revises the fuzzy rules learnt by neural or GA approaches.

https://doi.org/10.1017/S0263574799002398 Published online by Cambridge University Press

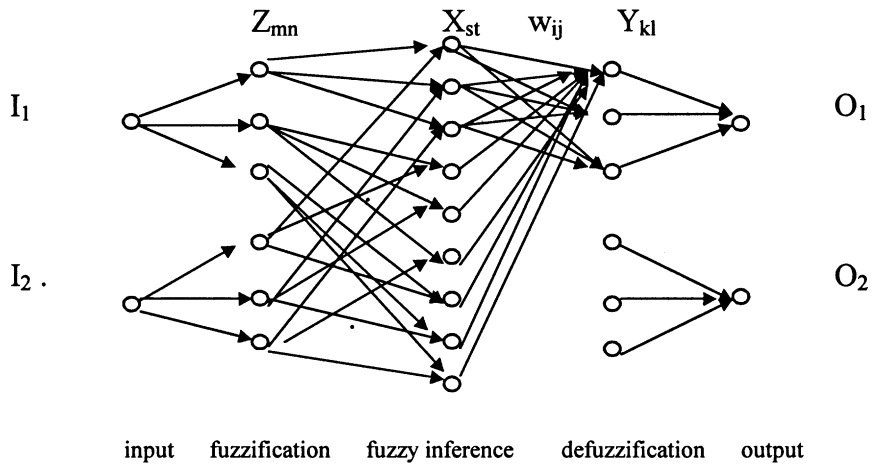

Fig. 1. The structure of a five-layer fuzzy neural network.

A Neural-learning function consists of two subfunctions: a "Parameters definition" subfunction defines the training parameters of a multi-layer feedforward network, as a Rule learning subfunction automatically learns fuzzy rules by a multi-layer feedforward network.

GA-learning function consists of four subfunctions: a "Parameters definition" subfunction defines the training parameters of the genetic algorithm; an "Evaluation" subfunction edits the evaluation function for GA-learning; a "Rule learning" subfunction automatically learns fuzzy rules by GA-learning; a "Rule optimization" subfunction optimizes fuzzy rules that are manually edited or learnt by a neural network.

A Code-generation function consists of two subfunctions: a "Motorola SR3" subfunction generates codes of fuzzy rules for a MCU chip (Motorola SR3); an "Other chips" subfunction generates codes of fuzzy rules for a DSP chip (e.g. TI C240).

2.2. Automatic generation of fuzzy membership functions

The precision and smoothness of a fuzzy controller are affected by the division of the variables' fuzziness categories. In order to reduce the cost of sensors and the fuzzy controllers, researchers usually select more fuzziness categories (more than 4) for variables with a higher demand of precision or select less fuzziness categories (less than 5) for those with a lower demand of precision. After fuzziness categories of each variable have been determined, their fuzzy membership function can be defined automatically. The membership function is usually expected to be denser in the more frequented work area, so that fuzzification or defuzzification is more sensitive to that area. The center values of triangular functions of fuzzy predicates are determined by a clustering algorithm which makes the square mean error of clustering least. For example, the work area of a variable is $[-1, 1]$. Five fuzziness categories [NL, NS, Z, PS, PL] of the variable are divided. The more frequented work area is $[-0.5, 0.5]$; its fuzzy membership function (Figure 3) is automatically generated.

3. LEARNING FUZZY RULES BY A MULTI-LAYER FEEDFORWARD NETWORK

Knowledge acquisition is the bottleneck for implementing fuzzy systems. Neural networks can learn mapping relations between inputs and outputs of a set of training samples. A multi-layer perceptron network with sufficient hidden nodes has been proved to be a universal approximator. Assuming $H(W, X)$ is the approximation of $h(X)$, the learning process is designed to reduce the distance between $H(W, X)$ and $h(X)$.

The goal of designing a fuzzy controller is that the outputs of the fuzzy controller smoothly vary along with the variation of its inputs. Given a set of training samples about inputs and their expected outputs of a fuzzy controller, automatic learning of fuzzy rules realize a smooth transformation among these inputs and their expected outputs.

3.1. Automatically learning fuzzy rules by data fitting

A multi-layer perceptron network can be used for the automatic learning of fuzzy rules because as an approximation tool it can realize data fitting according to the inputs and their expected outputs of the training samples. In the

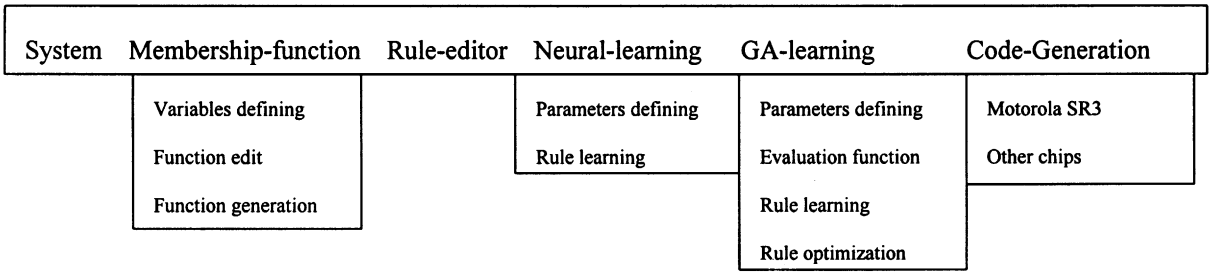

Fig. 2. The menu of the software development tool for fuzzy control.

https://doi.org/10.1017/S0263574799002398 Published online by Cambridge University Press

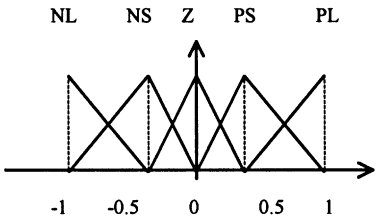

Fig. 3. Membership function of the variable.

multi-layer feedforward network, nodes $Z_1 \ldots Z_M$ in the input layer represent the input variables of a fuzzy controller; nodes $O_1 \ldots O_K$ in the output layer represent the output variables of a fuzzy controller; $d_1 \ldots d_K$ represent the expected outputs of $O_1 \ldots O_K$. A set of training samples about inputs and their expected outputs of the fuzzy controller are used to train weights of the multi-layer feedforward network by an Error Back-Propagation Algorithm until the cycle error is less than the upper bound E_{max}. After the multi-layer perceptron network has been trained, it can be used to realize data fitting so that the expected outputs for arbitrary inputs of a fuzzy controller can be computed by the trained multi-layer perceptron network. Learning a fuzzy rule is to determine the best category of fuzziness of the output variable corresponding to a combination of categories of fuzziness of the input variables. The goal of learning fuzzy rules is that the error between the real outputs of the fuzzy controller to be designed and their expected outputs are reduced as much as possible.

Given the premise of a fuzzy rule (i.e. a combination of categories of fuzziness $A_1, A_2, \ldots, A_n$ of input variables $I_1, I_2, \ldots, I_n$), the center values $X_1, X_2, \ldots, X_n$ of membership functions of $A_1, A_2, \ldots, A_n$ are selected as the values of $I_1, I_2, \ldots, I_n$. Then according to the definition of fuzzy membership functions, the membership of categories of fuzziness of $I_1, I_2, \ldots, I_n$ other than $A_1, A_2, \ldots, A_n$ are all zero, so that the output O_1 of the fuzzy rule is equal to the output of the fuzzy controller. As the expected output of the fuzzy controller can be computed by the multi-layer feedforward network according to the inputs $(X_1, X_2, \ldots, X_n)$, the expected output of the fuzzy rule can be found. The nearest category M of fuzziness (the center value of the membership function of M is nearest to the expected output) is selected as the conclusion of the fuzzy rule, so that the error between the real output of the fuzzy rule and their expected output is minimum. The fuzzy rule learnt by the multi-layer feedforward network is defined as follow: If $I_1 = A_1, \ldots, I_n = A_n$ Then $O_1 = M$.

For example, a fuzzy rule for a fuzzy controller with two inputs and one output under the premise $I_1 = NS$ (negative small), $I_2 = PL$ (positive large) needs to be learnt by the multi-layer feedforward network. The center value of triangular membership function of NS is -0.2; the center value of triangular membership function of PL is 1.8; the expected output of the fuzzy rule is computed by the trained multi-layer feedforward network: $O_1 = 0.3$. If the center value of the triangular membership function of PS in O_1 is nearest to 0.3, PS is selected as a fuzzy conclusion of the fuzzy rule i.e. the fuzzy rule is automatically learnt: If $I_1 = NS$, $I_2 = p\mathrm{L}$ then $O_1 = PS$.

3.2. Automatic learning of fuzzy rules by supervised learning

From the structure of a five-layer fuzzy neural network in Figure 1, it can be seen that the learning of fuzzy rules is to find mapping relations among the combinations of fuzzy predicates of input variables and their corresponding fuzzy predicates of output variables (that is, the weights w_{ij} between fuzzy inference and defuzzification layers).[7] As the fuzzy membership function of each variable and the approaches of fuzzy inference (MAX-MIN) and defuzzification are determined, the weights w_{ij} can be learnt by a back-propagation (BP) algorithm and training examples about inputs and their expected outputs. The following are the definition of weights of each a layer and the computation of outputs of nodes in the five-layer fuzzy neural network (Figure 1).

(i) All weights between input-layer and fuzzification-layer equal to one. The outputs of nodes in the fuzzification-layer are memberships [0, 1] of fuzzy predicates of variables in the input-layer.

$Z_{mn} = \mu_{A_{mn}}(I_m)$, A_{mn} is the nth category of fuzziness of I_m, $m \in \{1, 2\}$, $n \in \{1, 2, 3\}$.

(ii) All weights between fuzzification-layer and fuzzy-inference-layer equal to one. The outputs of nodes in the fuzzy-inference-layer are AND-operation of premises of fuzzy rules (minimum of memberships of input variables). $X_{st} = \min(Z_{1s}, Z_{2t})$, $s, t \in \{1, 2, 3\}$.

(iii) Weights between fuzzy-inference-layer and defuzzification-layer are w_{ij}. The outputs of nodes in the defuzzification-layer are an OR-operation of conclusions of fuzzy rules (maximum of conclusions derived by different fuzzy rules).

$Y_{1l} = \max(\omega_{il} \times X_{st})$, $Y_{2l} = \max(\omega_{i,l+2} \times X_{st})$, $i \in \{1, \ldots, 9\}$, $k \in \{1, 2\}$, $l \in \{1, 2, 3\}$.

(iv) All weights between defuzzification-layer and output-layer are 1. The outputs of nodes in the output-layer are computed by the defuzzification algorithm.

$$O_k = \sum_{l=1}^{3} Y_{kl} \times C_{kl} \Big/ \sum_{l=1}^{3} Y_{kl}, \; k \in \{1, 2\}, \; l \in \{1, 2, 3\},$$

C_{kl} is the center value of the triangular membership function of A_{kl}, A_{kl} is the lth category of fuzziness of O_k.

The weights w_{ij} can be learnt by a BP (Back Propagation) algorithm, so that the square mean error E of the fuzzy neural network is minimum,

$$E = \frac{1}{2N} \sum_{k=1}^{N} (O_d - O_k)^2$$

N is the number of nodes in the output-layer, O_d is the expected output of the dth node in the output-layer. The weights w_{ij} are adjusted according to gradient descent of E.

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj} = w_{kj}(t) - \beta \nabla E_{wkj}, \quad \nabla E_{wkj} = \partial E / \partial w_{kj}$$

As the outputs of nodes in the defuzzification-layer are an OR-operation of conclusions of fuzzy rules (maximum of

https://doi.org/10.1017/S0263574799002398 Published online by Cambridge University Press

conclusions derived by different fuzzy rules), only winning nodes affect the outputs of nodes in the output-layer. Hence only weights w_{ij} of the winning nodes need to be adjusted: $w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj} = w_{kj}(t) - \beta X_{st}\delta$, $\delta = O_k - O_d$, where β is the learning rate. After the weights w_{ij} have been learnt by supervised learning, the weights w_{ij} represent the mapping relations among fuzzy predicates of input variables and fuzzy predicates of output variables, and can be understood as certainty factors of fuzzy rules.

Learning according to a gradient descent in the multi-layer feedforward network involves a problem of a local minimum. Genetic algorithms are an effective technique for global optimization, hence fuzzy rules learnt by neural network can be further optimized by genetic algorithms.

4. LEARNING OF FUZZY RULES BY A GENETIC ALGORITHM

A genetic algorithm is a computational model inspired by Darwin's theory of evolution: superior chromosomes have a greater probability of being selected for inheritance and populations with good fitness are evaluated by chromosome inheritance and chromosome evolution (mutation).

A genetic algorithm is an effective technique for solving complicated problems of optimization. It has been applied for the optimization of combinations. The automatic learning of fuzzy rules can be seen as the problem of the combination optimization of input-output states: a group of fuzzy rules can be seen as a kind of combination, and the automatic learning of fuzzy rules is to find the best combination. As an effective technique of optimization, genetic algorithm can be used for the automatic learning of fuzzy rules.

Key techniques in the automatic learning of fuzzy rules by GA are chromosome coding of fuzzy rules, operators of inheritance (selection, crossover, mutation), and evaluation of chromosomes (fuzzy rules).

4.1. Chromosome coding of fuzzy rules

A group of fuzzy rules can be seen as a kind of combination of fuzzy predicates. As the order of the combinations of fuzzy predicates of input variables can be determined in advance, only fuzzy predicates of output variables (conclusions of the fuzzy rules) under these combinations need to be recorded for the chromosome coding of fuzzy rules. As the number of fuzziness category of output variables is usually not more than 9, numbers between 1 and m ($m \leq 9$) are used to represent the fuzziness category of output variables (m is the number of the fuzziness category of output variables).

For example, chromosome coding of above fuzzy rules with three inputs and one output (Table I) are realized by coding the fuzzy predicates of O_1: 33221233222123321232211232 1, (1,2,3 represent the fuzzy predicates S, M, L, respectively). On the contrary, given a chromosome coding of fuzzy rules 11223322123322123221112232 2, it can be transformed into corresponding fuzzy rules as Table II.

4.2. Operations of chromosome inheritance

(i) **Selection**. The random selection of chromosomes for reproduction is based ona biased roulette wheel algorithm. For a population of size n, a chromosome i is assigned a probability P_i of being selected. P_i is proportional to the fitness f_i, computed as

$$P_i = f_i \Big/ \sum_{k=1}^{n} f_k .$$

After chromosome reproduction, the chromosomes with a higher fitness have a greater probability of being selected for a recombination operation (crossover, mutation).

(ii) **Crossover**. When two chromosomes are randomly selected for the operation of crossover, two points are randomly selected in the length of these two chromosomes. Two new chromosomes are generated by performing a two-point crossover.

(iii) **Mutation**. Mutation is an operation whereby the allele of a gene is altered randomly; it can then introduce new genetic materials into the population. In order to

Table I: Fuzzy rule-table of a fuzzy controller with three inputs and one output

I_1	S	S	S	S	S	S	S	S	S	M	M	M	M	M	M	M	M	M	L	L	L	L	L	L	L	L	L
I_2	S	S	S	M	M	M	L	L	L	S	S	S	M	M	M	L	L	L	S	S	S	M	M	M	L	L	L
I_3	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L
O_1	L	L	M	M	S	M	L	L	M	M	M	S	M	L	L	M	S	M	L	M	M	S	S	M	L	M	S

Table II: Fuzzy rule-table TRANSFORMED FROM A CHROMOSOME 11223322123322123221112232 2

I_1	S	S	S	S	S	S	S	S	S	M	M	M	M	M	M	M	M	M	L	L	L	L	L	L	L	L	L
I_2	S	S	S	M	M	M	L	L	L	S	S	S	M	M	M	L	L	L	S	S	S	M	M	M	L	L	L
I_3	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L
O_1	S	S	M	M	L	L	M	M	S	M	L	L	M	M	S	M	L	M	M	S	S	S	M	M	L	M	M

https://doi.org/10.1017/S0263574799002398 Published online by Cambridge University Press

prevent the loss of potentially useful genetic materials during the search, mutation is usually performed at a very low rate (e.g. 1%). Given a chromosome for the mutation operation, one point P is randomly selected in the length of the chromosome, and the allele of the chromosome at point P is altered randomly. It is necessary that any mutation operation used satisfies the constraint "the allele altered must still represent a category of fuzziness of the output variable". If the number of the categories of fuzziness of the output variable is m ($m \leq 9$) and the allele of the chromosome at point P is n ($n \leq m$), then the allele of the chromosome at point P is altered randomly into n' which satisfies the constraint: $n' \leq m$, $n' \neq n$.

4.3. Evaluation of a chromosome (fuzzy rules)

For each generation, the fitness of each chromosome in the population is evaluated using a fitness function. The choice of a fitness function is usually very specific to the problem under consideration, and often crucial to the success of the GA. The process of learning fuzzy rules by GA terminates when the fitness of the learnt fuzzy rules is accepted by users.

(i) **Evaluating the fitness of the fuzzy rules by user defined fitness function.** According to the goal of the fuzzy controller, the fitness function of fuzzy rules can be defined by users. If fuzzy rules cannot be evaluated by the fitness function directly, simulations of the fuzzy controller based on the learnt fuzzy rules are used for the evaluation. In the well-known "cart-pole balance" example, the goal of fuzzy control is to keep the pole from falling and maintain the pole close to the vertical position. The fuzzy rules are tested with different initial conditions within the range of control. For each of these tests, the cart-pole system is simulated until the pole falls or the prespecified value of T timesteps is reached.

(ii) **Evaluating the fitness of fuzzy rules by computing cumulative errors.** A fitness function specific to the problem of fuzzy control is usually difficult to define. As a group of fuzzy rules describe fuzzy control under different initial conditions, fuzzy rules are usually difficult to be evaluated by the fitness function directly, and simulations for evaluation of fuzzy rules are time-consuming. Given training examples about inputs and their expected outputs of a fuzzy controller, the goal of designing a fuzzy controller is to reduce the errors between expected outputs and real outputs derived by fuzzy rule as much as possible. Thus evaluating the fitness of fuzzy rules can be realized by computing the cumulative errors between expected outputs and real outputs derived by fuzzy rules by training samples with different input situations.

A chromosome operated by crossover and mutation can be transformed into its corresponding fuzzy rules. According to the five-layer fuzzy neural network (Figure 1) and inputs of training samples, real outputs of fuzzy neural network can be derived by predefined algorithms of fuzzification, defuzzification and the fuzzy rules learned by GA. The fitness of fuzzy rules is evaluated by computing cumulative errors between real outputs and expected outputs of the training samples.

$$f = \sum_{i=1}^{n} \sum_{j=1}^{m} (O_j - O_{ej})^2$$ n–the number of training examples,

m–the number of output variables of the fuzzy controller, O_j–the real output derived by fuzzy rules for training sample T_i, Oe_j–the expected output of training example T_i.

The evaluation of the fitness of fuzzy rules by computing cumulative errors is dependent on the training examples. The selected training examples should have good distribution (reflecting different possible input conditions of the fuzzy controller) and good representation (reflect typical input conditions of the fuzzy controller). If existing training examples cannot satisfy these demands, data fitting can be first realized by a multi-layer feedforward network (see Section 3.1). Satisfactory training examples can be selected after the data fitting, and then they can be used for evaluating the fitness of fuzzy rules in GA.

5. CONCLUSIONS

In this paper, an algorithm for the automatic learning of fuzzy rules by a multi-layer feedforward network is presented. It is simple and fast, but has the problem of a local optimum. An algorithm for the automatic learning of fuzzy rules by GA is presented. It can search for a global optimum, but is complicated and time-consuming. The automatic learning of fuzzy rules by combining a multi-layer feedforward network and a genetic algorithm can complement each other.[4]

The following example is used for examining our development tool. A system with three inputs and one output is constructed. Its mathematical model is: $o = 0.5 \times (e^{i_1} + i_2 \times i_3 \times \cos(i_1 \times i_2) + i_1 \times i_3)$, where i_1, i_2, i_3, $o \in [0,1]$. A fuzzy system is taught for the modeling of the system by our developmental tool, where i_1, i_2, i_3 are divided into 5 fuzziness categories. In order to show the performance of the fuzzy system in a graphic form, i_1, i_2, i_3 are represented by functions with the same variable $t \in [0,1]$: $i_1 = t$,

$$i_2 = \begin{cases} 2 \times t & 0 \leq t < 0.5 \\ 2 - 2 \times t & 0.5 \leq t \leq 1 \end{cases}, \quad i_3 = \begin{cases} 0.9 & t \in [0,0.25) \cup [0.5,0.75) \\ 0.1 & t \in [0.25,0.5) \cup [0.75,1] \end{cases}.$$

The following figures are the analysis of the results of our development tool. The curve and the broken line in the figures, respectively, represent the real output and expected output of the fuzzy system. Figure 4 is the result of a fuzzy system taught by a multi-layer perceptron network; Figure 5 is the result of the fuzzy system taught by a genetic algorithm.

The software development system has also been applied to the design of a fuzzy controller for a big steelworks, which has won acclaim for its satisfactory results. The fuzzy controller has four inputs and one output; each variable is divided into five categories of fuzziness. Fuzzy rules are extracted based on 188 training examples. The average error

https://doi.org/10.1017/S0263574799002398 Published online by Cambridge University Press

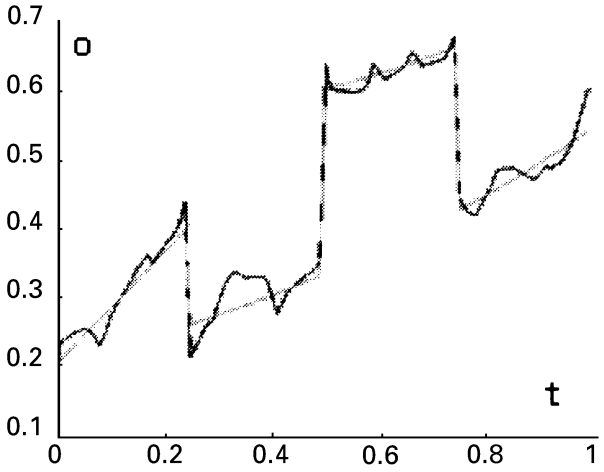

Fig. 4. The result of a fuzzy system taught by a multi-layer perceptron network.

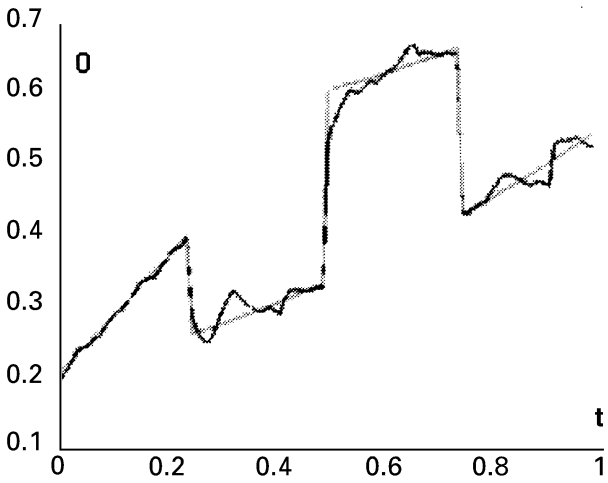

Fig. 5. The result of a fuzzy system taught by a genetic algorithm.

between real output and expected output of the fuzzy controller is less than 10%.

ACKNOWLEDGEMENT

This research is supported by National High Tech-Plan "863-CIMS" in China (No. 863–511–945–005).

References

1. C.T. Lin and C.S. Lee (editors), *Neural Fuzzy Systems* (Prentice-Hall, New Jersey, 1996).
2. C.T. Lin *et al*., "Fuzzy adaptive learning control network with on-line neural learning", *Fuzzy Sets and Systems* **71**, No. 1, 25–45 (1995).
3. M. Mohammadian *et al*., "A Case Study of Knowledge Acquisition: from Connectionist Learning to an Optimized Fuzzy Knowledge Base", *IEEE 2nd International Workshop on Emerging Technologies and Factory Automation* (1993) pp. 106–111.
4. J.J. Shann and H.C. Fu, "A fuzzy neural network for rule acquiring on fuzzy control systems", *Fuzzy Sets and Systems* **71**, No. 3, 345–357 (1995).
5. M.H. Lim *et al*. "A GA paradigm for learning fuzzy rules", *Fuzzy Sets and Systems* **82**, 177–186 (1996).
6. L. Zhang *et al*. "On Rule Checking and Learning in an Acupuncture Diagnosis Fuzzy Expert System by Genetic Algorithm", *1995 IEEE International Conference on Fuzzy Systems* (1995) **Vol. 2**, pp. 455–460.
7. Nashina Takatoshi, "Fuzzy inference neural networks which automatically partition a pattern space and extract fuzzy if-then rules", *3rd IEEE International Conference on Fuzzy Systems* (1995) **Vol. 2**, pp. 1314–1319.

https://doi.org/10.1017/S0263574799002398 Published online by Cambridge University Press