Automatic Generation of Fuzzy Control Rules by Machine Learning Methods

Shih-Chun Hsu   Jane Yung-Jen Hsu*   I-Jen Chiang

Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan 106, R.O.C.

Abstract

This paper presents a multi-strategy learning technique for automatic generation of fuzzy control rules. In order to eliminate irrelevant input variables and to prioritize relevant ones according to their influences on the output value(s), the ID3 algorithm is adopted to classify the given set of training I/O data. The resulting decision tree can be easily converted into IF-THEN rules, which are then fuzzified. The fuzzy rules are further improved by tuning the parameters that define their membership functions using the gradient-descent approach. Experimental results of applying the proposed technique to nonlinear system identification have shown improvements over previous work in the area. In addition, it has been successfully applied to mobile robot control in unknown environments.

1 Introduction

In recent years, fuzzy control has been applied to a wide range of applications, for example, heater temperature control [12], cement kiln control [2], automatic train control [13], vehicle speed control [10], and autonomous mobile robot control [5]. It has been shown that fuzzy control is an effective approach to solving complex ill-defined problems. To emulate the behavior of domain experts, their knowledge and experiences need to be extracted into a set of fuzzy control rules.

Designing a good fuzzy controller is not a trivial task. Given a complex control problem, it is often impossible to completely articulate one's knowledge in the form of linguistic IF-THEN rules. It is also difficult to identify the relevant input to the given system [1]. Moreover, to efficiently characterize the control operations, one should focus on the more important input variables. Machine learning methods can be used to generate automatically fuzzy control rules from input-output data, which are gathered from past successful control operations by skilled operators [3, 8, 11, 4]. A detailed description of the proposed learning technique is presented in the following section. Experimental results in two different application domains are presented in Section 3.

*This research was supported in part by a grant from the National Science Council of ROC under grant number *NSC 82-0408-E-002-429*.

2 Generating Fuzzy Rules

This work combines ID3 [9] with gradient-descent methods. The procedure is outlined in Figure 1. In

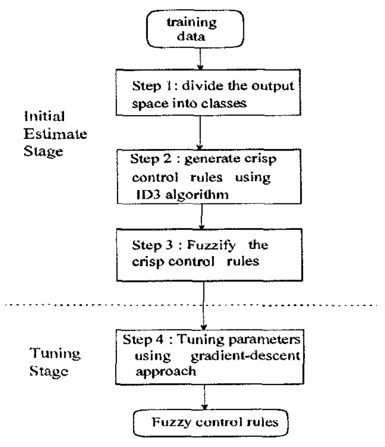
training data
Step 1 : divide the output space into classes
Initial Estimate Stage
Step 2 : generate crisp control rules using ID3 algorithm
Step 3 : Fuzzify the crisp control rules
Tuning Stage
Step 4 : Tuning parameters using gradient-descent approach
Fuzzy control rules

Figure 1: Generation of fuzzy rules

what follows, the individual steps are described in detail.

2.1 Preprocessing

Given input state variables $x_1, \ldots, x_n$ and an output variable y, the i-th training data is represented as:

$$(X_1^i, \ldots, X_n^i, Y^i)$$

where $X_1^i, \ldots, X_n^i$ and Y^i are the values for the corresponding variables. To classify such data by the ID3 algorithm, the output space needs to be divided into distinct (crisp) regions. For simplicity, the output space is divided equally into l regions in our experiments, where l is determined empirically. Therefore, the data is transformed into:

$$(X_1^i, \ldots, X_n^i, C^i)$$

where C^i denotes the output class.

2.2 Generation of crisp control rules

In a complex system with many input variables, the output value may depend only on some of the inputs with a varying degree of influence. A good controller should focus on the important parameters while ignoring irrelevant ones. Therefore, the ID3 algorithm [9, 11], which has been widely used for classification problems, is chosen for its ability to prioritize input variables. The resulting decision tree is in turn transformed into an initial set of crisp control rules. The algorithm is summarized as follows:

1. *Attribute selection* — Select the input variable that best partitions the training data into homogeneous subsets.
2. Apply step 1 recursively to every subset that contains samples with different output classes. The process terminates when every subset is homogeneous, i.e. the data can be classified correctly.
3. *Pruning* — Trim the decision tree.
4. Generate one IF-THEN crisp control rule for each root-to-leaf path in the decision tree.

Attribute selection A number of alternative information measurements can be used in selecting the best attributes[6]. In order to handle multi-valued attributes, Quinlan's *gain ratio* is adopted in this work. Using the following notations,

n_t	the total number of samples in all branches;
n_c	the number of samples of class c;
n_b	the number of samples in branch b;
n_{bc}	the total of samples in branch b of class c.

the procedure for selecting the best variable is to:

1. Calculate the total expected information content:
$$M_e = \sum_c -\frac{n_c}{n_t} \log_2 \frac{n_c}{n_t}$$
2. For each candidate variable x_i, calculate its gain ratio
$$GR = \frac{M_e - M_b}{IV}$$
where M_b is the expected information content if x_i is selected to partition the current subset; the scaling factor IV measures the information value of x_i as defined below:
$$M_b = \sum_b \left(\frac{n_b}{n_t}\right) \cdot \left(\sum_c -\frac{n_{bc}}{n_b} \log_2 \frac{n_{bc}}{n_b}\right)$$
$$IV = -\sum_b \frac{n_b}{n_t} \log_2 \frac{n_b}{n_t}$$
In general, the gain ratio measurement favors variables with an uneven distribution of examples. It discourages over-partitioning and tends to preserve larger clusters of samples. Our implementation partitions the input space of x_i into 3 to 20 uniform regions. The gain ratio for all different ways to partition the space is calculated, and the one that maximizes its value will be used.
3. Select the input variable x_i with the largest gain ratio, and partition the current set into subsets accordingly.

Decision tree pruning As a decision tree grows deeper, the marginal information gain usually decreases. Besides, the training data may contain noise or uncertainty resulting in unnecessarily large decision trees. For more efficient control, decision trees can be pruned to remove the least reliable branches. A number of alternative pruning methods have been studied [7]. Quinlan's *pessimistic pruning* is adopted because it is easy to compute and does not require a separate test data set.

Starting at the root of the tree, each node is evaluated to determine if dividing it into a subtree can meaningfully improve the classification. Suppose that

$N(t)$	=	number of training examples at node t
$e(t)$	=	number of examples mis-classified at node t

1. The *mis-classification rate* with continuity correction is estimated to be
$$\hat{r}(t) = \frac{e(t) + 1/2}{N(t)}$$
2. For the sub-tree T_t rooted at t, the corrected mis-classification rate is
$$\hat{r}(T_t) = \frac{\sum(e(i) + 1/2)}{\sum N(i)} = \frac{\sum e(i) + N_T/2}{\sum N(i)}$$

where i ranges over the leaves of the T_t, and N_T is the total number of leaves in the subtree. We also have $N(t) = \sum N(i)$ as they refer to the same set of examples. Therefore, the rates can be replaced by $\hat{n}(t) = e(t) + 1/2$ and $\hat{n}(T_t) = \sum e(i) + \frac{N_T}{2}$, i.e. the number of mis-classifications.

3. A subtree is pruned unless $\hat{n}(T_t) < \hat{n}(t) - \sigma(\hat{n}(T_t))$, where

$$\sigma(\hat{n}(T_t)) = (\frac{\hat{n}(T_t)(N(t) - \hat{n}(T_t))}{N(t)})^{1/2}$$

is the standard error for $\hat{n}(T_t)$.

Rule generation The i-th rule is generated from the root to the i-th leaf node in the decision tree.

IF $lb_{i1} \leq x_1 < ub_{i1} \wedge \ldots lb_{in} \leq x_n < ub_{in}$ **THEN** y is C_i

where lb_{ij} is the lower bound and ub_{ij} is the upper bound of x_j at the i-th leaf node; C_i is the desired output class.

2.3 Fuzzification

To obtain an initial estimate, each crisp IF-THEN rule is fuzzified by assigning a linguistic variable X_i to each input variable x_i and Y to the output y. The resulting fuzzy rule is of the form:

IF X_1 is $A_{i1} \wedge \ldots \wedge X_n$ is A_{in} **THEN** Y is O_i

where A_{ij} is a fuzzy set of X_j and O_i is a fuzzy singleton denoting the average output value of the corresponding data. Each A_{ij} is defined by a triangle-shaped fuzzy membership function $\mu_{A_{ij}}(X_j) = 1 - \frac{2|X_j - c_{ij}|}{w_{ij}}$, where $c_{ij} = \frac{1}{2}(ub_{ij} + lb_{ij})$ denotes its center and $w_{ij} = 2(ub_{ij} - lb_{ij})$ denotes its width as shown in Figure 2.

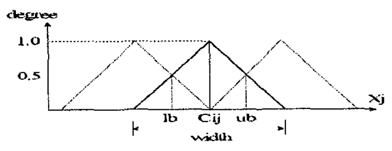

Figure 2: Fuzzify a crisp region

2.4 Parameter tuning

In dynamic environments, the performance of a fuzzy controller can be improved by fine tuning its parameters [3, 4]. The last step of our learning procedure applies the gradient-descent algorithm to the initial set of fuzzy control rules as follows:

$$\begin{aligned} c_{ij}(t+1) &= c_{ij}(t) - \gamma_c \cdot \frac{\partial P}{\partial c_{ij}} \\ w_{ij}(t+1) &= w_{ij}(t) - \gamma_w \cdot \frac{\partial P}{\partial w_{ij}} \\ O_i(t+1) &= O_i(t) - \gamma_O \cdot \frac{\partial P}{\partial O_i} \end{aligned}$$

where γ_c, γ_w and γ_O are the learning rates for parameters c_{ij}, w_{ij} and O_i. The performance measurement is defined by $P = 1/2(y - y^r)^2$, with y^r being the desired output and y being the actual output. The process terminates when P falls below a threshold.

3 Simulation

To demonstrate the utility of the proposed approach, it was applied to the problems of nonlinear system identification and constructing a mobile robot controller to follow walls in unknown environments.

3.1 Nonlinear System Identification

Given a nonlinear system, the objective is to construct a fuzzy rule base that approximates its behavior. We selected the static nonlinear system described by Sugeno [11]:

$$y = (1 + x_1^{-2} + x_2^{-1.5})^2 \text{ where } x_1, x_2 \in [1,5] \quad (1)$$

In [11], a total of 50 input-output data tuples were

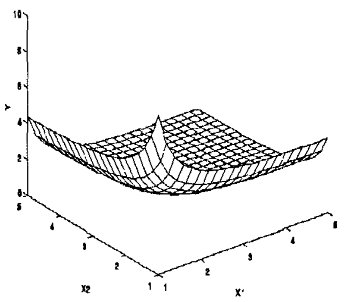

Figure 3: A nonlinear system with two inputs

generated from equation (1) and variables x_3 and x_4 were intentionally added as dummy inputs to check the effectiveness of the system. The performance index was defined as the cumulative mean square error of the output:

$$PI = \sum_{i=1}^{m} (y_i - y_i^r)^2 / m$$

where i ranges over a total number of m data tuples.

First, the output space was partitioned into five equal regions between 1.30 and 5.05, and the training data were classified by ID3. The resulting tree has x_2 as the root and each branch is further divided by x_1. The algorithm successfully eliminated x_3 and x_4 from consideration, and produced rules of the form:

$Rule_i$: **IF** X_2 is A_{i2} and X_1 is A_{i1} **THEN** Y is O_i

The gradient-descent algorithm was then used to fine tune the parameters of all the membership functions. In Figure 4, the dotted and solid curves indicate the changes in the PI values during the tuning phase with a learning rate of 0.004 and 0.002 respectively.

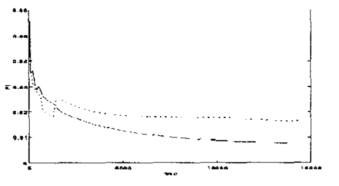
Figure 4: The learning curve

Table 1 summarizes the PI values after structure identification and parameter identification using Sugeno's position-type model (I), Sugeno's position-gradient type model (II), our hybrid approach (I) with $\gamma_c = \gamma_w = \gamma_O = 0.004$ and 0.002 (II). The final values for the fuzzy parameters are in Table 2.

Table 1: A comparison of PI after learning

PI	structure ID	parameter ID
Sugeno (I)	0.318	0.079
Sugeno (II)	0.318	0.010
ID3+GD (I)	0.058	0.016
ID3+GD (II)	0.058	0.007

Table 2: The parameters after tuning

No	c_{i2}	w_{i2}	c_{i1}	w_{i1}	O_i	No	c_{i2}	w_{i2}	c_{i1}	w_{i1}	O_i
1	1.4	3.9	1.2	0.2	5.7	14	1.4	3.9	3.7	0.3	2.1
2	1.4	3.9	1.2	0.2	4.3	15	1.4	3.9	3.8	0.3	3.0
3	1.4	3.9	1.6	0.2	3.9	16	1.4	3.9	4.2	0.3	2.3
4	1.4	3.9	1.7	0.1	5.2	17	1.4	3.9	4.5	0.2	2.7
5	1.3	3.9	1.9	0.2	3.5	18	3.0	2.6	1.1	0.6	4.8
6	1.4	3.9	2.0	0.3	2.6	19	3.0	2.6	1.2	0.4	3.7
7	1.4	3.9	2.3	0.3	3.4	20	3.0	2.6	1.6	0.4	2.6
8	1.4	3.9	2.5	0.3	2.2	21	3.0	2.6	4.6	0.4	1.5
9	1.4	3.9	2.6	0.3	2.3	22	4.8	3.9	0.6	2.1	3.3
10	1.4	3.9	2.8	0.3	2.3	23	4.4	4.0	1.6	1.6	2.0
11	1.4	3.9	2.9	0.2	3.7	24	4.6	3.9	2.8	1.5	1.6
12	1.4	3.9	3.1	0.3	4.1	25	4.7	3.9	3.4	1.5	1.5
13	1.4	3.9	3.4	0.3	2.0	26	4.7	3.9	4.6	2.3	1.3

3.2 Mobile robot control

The second experiment is to generate a fuzzy controller for a mobile robot to move smoothly along arbitrarily shaped walls in unknown environments. A Nomad 200 mobile robot manufactured by Nomadic Technologies Inc. was used. Its drive mechanical system has a zero gyro-radius. A ring of 16 infrared sensors, whose readings range from 0 to 30 inches, are arranged counter-clockwise as shown below. All sen-

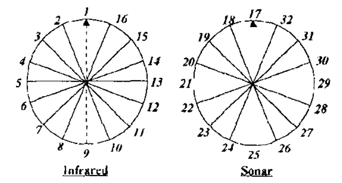

Figure 5: The sensor system

sor readings along with the position and orientation of the robot are transmitted back to the host computer via radio modem.

For simplicity, the robot is assumed to move at a constant forward speed. The controller takes the 16 infrared sensor readings $x_1, \ldots, x_{16}$ as inputs and produces the desired steering angle y, which is limited to $-45^\circ \leq y \leq 45^\circ$, as the single output. Note that a negative value of y represents clockwise rotation of the steering wheel, while a positive value represents counterclockwise rotation.

During the training process, 300 input-output data tuples were collected by recording a skilled expert instructing the mobile robot to follow the wall(s) on its right-hand side. Figures 6 and 7 show the sample training environments and trajectories. A portion of the training data are listed in Table 3.

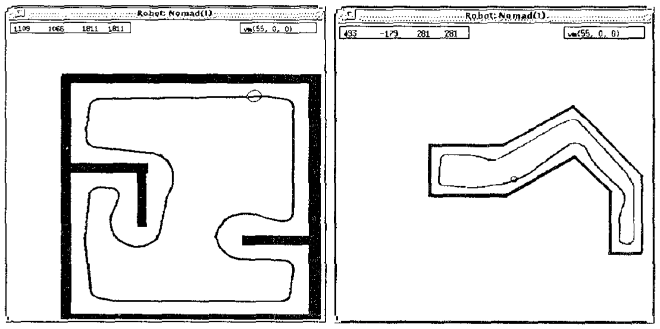

Figure 6: Training environment and trajectory (1)(2)

The output space is partitioned into 4 equal regions between -30.0 and 28.6. Some of the rules generated after the structure identification stage by ID3 are shown in Table 4.

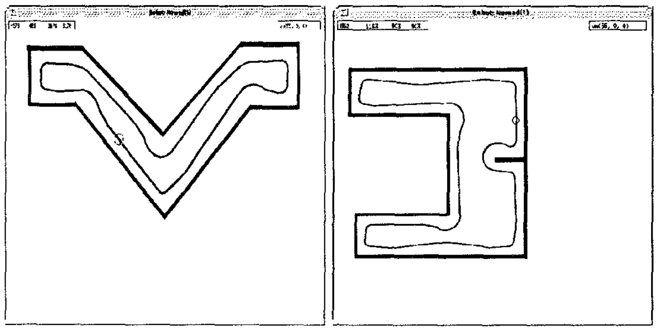
Figure 7: Training environment and trajectory (3)(4)

Table 3: A partial listing of the training data

	1	2	3	4	5
x_1	24	22	20	18	16
x_2	26	24	22	20	20
x_3	30	30	30	30	30
x_4	30	30	30	30	30
x_5	30	30	30	30	30
x_6	30	30	30	30	30
x_7	30	30	30	30	30
x_8	30	30	30	30	30
x_9	30	30	30	30	30
x_{10}	30	30	30	30	30
x_{11}	30	30	30	30	30
x_{12}	30	30	30	30	30
x_{13}	30	30	30	30	30
x_{14}	30	30	30	30	30
x_{15}	30	30	30	30	30
x_{16}	26	24	22	28	24
y	-5.2	1.8	8.6	15.3	21.8

Table 4: Control rules for wall following

IF $X_{16} < 15.33$
THEN Y is 27.1

IF $15.33 \leq X_{16} < 22.67$
THEN Y is 22.6

IF $22.67 \leq X_{16} \wedge X_9 < 25.5$
THEN Y is 21.5

...

IF $22.67 \leq X_{16} \wedge 28.5 \leq X_9 \wedge 21.0 \leq X_1 < 24.0$
THEN Y is 10.5

...

IF $22.67 \leq X_{16} \wedge 28.5 \leq X_9$
$\wedge 27.0 \leq X_1 \wedge 11.3 \leq X_{13} < 20.67$
THEN Y is -1.0

IF $22.67 \leq X_{16} \wedge 28.5 \leq X_9$
$\wedge 27.0 \leq X_1 \wedge 20.67 \leq X_{13}$
$\wedge 12.67 \leq X_{15} < 21.33$
THEN Y is 8.7

...

IF $22.67 \leq X_{16} \wedge 28.5 \leq X_9$
$\wedge 27.0 \leq X_1 \wedge 20.67 \leq X_{13}$
$\wedge 21.33 \leq X_{15} \wedge 20.67 \leq X_{14}$
$\wedge 29.13 \leq X_{11} \wedge 25.33 \leq X_{12}$
THEN Y is -11.5

For example, the first rule states that "If sensor reading 16 is less than 15 inches, i.e. very close to the wall, then the steering angle should be increased by 27.1°, i.e. turn left." It should be pointed out that the resulting rules show that the front and right sensors 16, 9, 1, 13, 15, 14, 11, and 12 were chosen as the most important parameters while following walls on the right. If each input variable can be classified into n regions, the maximum number of rules may be as high as n^{16}. In comparison, the number of fuzzy rules produced by our method is only 21. The utility of prioritizing important input variables while discarding irrelevant ones is quite evident. In addition, such control rules are expressive and intuitive.

Experiments were performed with the learning rate set to be 0.004. Figure 8 shows the trajectory using the generated fuzzy controller in the training environments. Simulation results showed that the mobile robot can successfully follow walls even though the environment is different. Figure 9 shows the trajectory using fuzzy controller in an unknown maze.

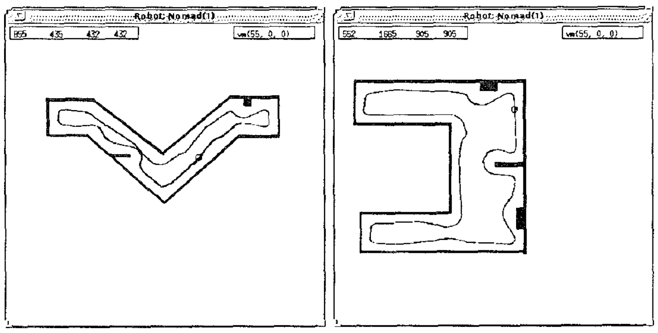

Figure 8: The trajectory using fuzzy controller (3)(4)

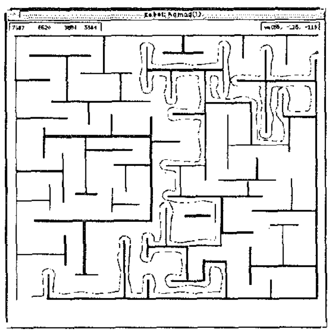

Figure 9: Using the fuzzy controller in a maze

4 Conclusion

This paper presented a hybrid learning approach for automatic generation of fuzzy control rules. In a complex environment, with many input variables, the output may depend only on some of the inputs. Moreover, each input may have a varying degree of influence on the output value. The ID3 algorithm is adopted to produce the initial estimate rules for its ability to select and prioritize the most important input variables. Computer simulation results showed such a method performed well in identifying irrelevant input variables. After fuzzifying the initial estimate rules, the gradient-descent algorithm is applied to fine tune the parameters of membership functions.

The advantages of the proposed approach can be summarized as: 1) It can select the most important input variables while eliminating irrelevant ones for solving problems in a complex environment; 2) It can prioritize input variables according to their influence on the output independent of domain knowledge; 3) The learned fuzzy controller is effective and it outperformed previous work in approximating nonlinear systems; 4) The hybrid approach is both adaptive and expressive.

References

[1] M. Braae and D.A. Rutherford, "Selection of parameters for a fuzzy logic controller," *Fuzzy Sets and Systems,* vol.2, no.3, pp.185-199, 1979.

[2] L.P. Holmblad ,"Control of cement kiln by fuzzy logic," *Fuzzy Information and Decision Process,* pp.389-399, 1982.

[3] C.C. Jou and N.C. Wang, "Training a fuzzy controller to back up an autonomous vehicle," *IEEE International Conference on Robotics and Automation,* pp.923-928, Nice, France, 1993.

[4] C.T. Lin and C.S.G. Lee, "Reinforcement Structure/parameter Learning for an Integrated Fuzzy Neural Network" *Proc. of 1993 IEEE Int'l Conf. on Fuzzy Systems,* pp.88-93, San Francisco, 1993.

[5] Y. Maeda, M. Tanabe, M. Yuta, and T. Takagi, "Hierarchical control for autonomous mobile robots with behavior-decision fuzzy algorithm," *Proceedings of IEEE International Conference on Robotics and Automation,* pp.117-122, Nice, France, 1992.

[6] J. Mingers, "An empirical comparison of selection measures for decision tree induction," *Machine Learning,* vol.3, pp.319-342, 1989.

[7] J. Mingers, "An empirical comparison of pruning methods for decision tree induction," *Machine Learning,* vol.4, pp.227-243, 1989.

[8] H. Nomura, I. Hayashi and N. Wakami , "A learning method of fuzzy inference rules by gradient-descent method," *IEEE International Conference on Fuzzy Systems,* pp.923-930, 1992.

[9] J.R. Quinlan, "Induction of decision trees," *Machine Learning,* vol.1, pp.81-106, 1986.

[10] M. Sugeno and M. Nishida, "Fuzzy control of model car," *Fuzzy Sets and Systems,* vol.16, pp.103-113, 1985.

[11] M. Sugeno and T. Yasukawa, "A fuzzy logic based approach to qualitative modeling," *IEEE Transactions on Fuzzy Systems,* vol.1,no.1, 1993.

[12] T. Tani, M. Sakoda and K. Tanaka "Fuzzy modeling by ID3 algorithm and its application to prediction of heater outlet temperature," *IEEE Int'l Conf. on Fuzzy Systems,* pp.923-930, 1992.

[13] S. Yasunobu and S. Miyamoto, "Automatic train operation system by predictive fuzzy control," *Industrial Appls. of Fuzzy Control,* pp.1-18, 1985.