Kobe University Repository : Kernel

PDF issue: 2025-12-05

Fuzzy function approximators with ellipsoidal regions

Abe, Shigeo

(Citation)
IEEE transactions on systems, man and cybernetics. Part B, Cybernetics,29(5):654-661

(Issue Date)
1999-10

(Resource Type)
journal article

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/90000218

KOBE UNIVERSITY

Fuzzy Function Approximators with Ellipsoidal Regions

Shigeo Abe

Abstract— **This paper discusses two types of fuzzy function approximators that dynamically generate fuzzy rules with ellipsoidal regions: a function approximator based on Takagi–Sugeno type model with the center-of-gravity defuzzification and a function approximator based on a radial basis function network. Hereafter the former is called FACG and the latter is called FALC.**

In FACG, for each training datum the number of the training data that are within the specified distance is calculated and the training datum which has the maximum number of the training data is selected as the center of a fuzzy rule and the covariance matrix is calculated using the training data around the center. Then the parameters of the linear equation that defines the output value of the fuzzy rule are determined by the least-squares method using the training data around the center.

In FALC, the training datum with the maximum approximation error is selected as the center of a fuzzy rule. Then using the training data around the center, the covariance matrix is calculated, and the parameters of a linear equation that determines the output value are calculated by the least-squares method.

Performance of FACG and FALC is compared with that of multilayered neural networks and other fuzzy function approximators for the data generated by the Mackey–Glass differential equation and the data from a water purification plant.

Index Terms— **Center of gravity, ellipsoidal region, function approximation, fuzzy inference, radial basis function networks.**

I. Introduction

Function approximation is one of the major applications of fuzzy systems. Conventional fuzzy systems have no training capability but recently many fuzzy systems with training capabilities have been proposed [1]–[11]. In general, these fuzzy systems have faster training capabilities than multilayered neural networks have and comparable generalization ability. The fuzzy rules extracted by these methods are classified into two types according to the input regions: hyperbox regions and ellipsoidal regions. The fuzzy rules with hyperbox regions include conventional fuzzy rules in which input and output spaces are divided into subregions in advance [2]–[4] and fuzzy rules in which input subregions are defined dynamically during fuzzy rule extraction [5]–[7].

Radial basis function networks are considered to be one type of fuzzy function approximator in which the output is synthesized by the linear combination of the degrees of membership of the fuzzy rules with ellipsoidal regions [9], [10]. Fuzzy rules with ellipsoidal regions are generated dynamically [8]–[11]. The center and the covariance matrix of a fuzzy rule need to be determined. The center is determined based on cluster estimation [8], [11] or on the approximation error [9], [10]. The covariance matrix is either fixed [8], [9] or estimated by the steepest descent [10] or unsupervised learning [11].

In this paper, we calculate the covariance matrix using the training data around the center which is determined either by cluster estimation or the approximation error and we discuss two types of fuzzy function approximators that dynamically generate fuzzy rules with ellipsoidal regions: a function approximator based on Takagi–Sugeno type model [12] with the center-of-gravity defuzzification and a function approximator based on a radial basis function network. Hereafter the former is called FACG and the latter is called FALC.

In FACG, for each training datum the number of the training data that are within the specified distance is calculated and the training datum which has the maximum number of the training data is selected as the center of a fuzzy rule and the covariance matrix is calculated using the training data around the center. Then the parameters of the linear equation that defines the output value of the fuzzy rule are determined by the least-squares method using the training data around the center.

In FALC, the training datum with the maximum approximation error is selected as the center of a fuzzy rule. Then using the training data around the center, the covariance matrix is calculated and the parameters of a linear equation that determines the output value are calculated by the least-squares method. The difference between FALC and the method discussed in [10] is that the latter tunes the center and the covariance matrix by the steepest descent method but the former calculates the covariance matrix using the training data around the center.

In the following, first we discuss the form of fuzzy rules and then we discuss how to generate fuzzy rules for FACG and FALC. Finally, we evaluate FACG and FALC with multilayered neural networks and other fuzzy function approximators for the data generated by the Mackey–Glass differential equation and the data from a water purification plant.

II. Fuzzy Rule Inference

A. Fuzzy Rule Representation

We discuss function approximation with the m-dimensional input vector $\boldsymbol{x}$ and the one-dimensional (1-D) output y. (Extension to the multidimensional output is straightforward.) The fuzzy rules with ellipsoidal regions are given by

$$R_i: \quad \text{If } \boldsymbol{x} \text{ is } \boldsymbol{c}_i \text{ then } y = o_i \text{ for } i = 1, \cdots, N \tag{1}$$

where $\boldsymbol{c}_i$ is the center of the ith fuzzy rule, o_i is the corresponding output, and N is the number of fuzzy rules. The degree of membership of the fuzzy rule $R_i, m_i(\boldsymbol{x})$, is given by

$$m_i(\boldsymbol{x}) = \exp(-d_i^2(\boldsymbol{x})) \tag{2}$$

$$d_i^2(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{c}_i)^t Q_i^{-1} (\boldsymbol{x} - \boldsymbol{c}_i) \tag{3}$$

where $d_i(\boldsymbol{x})$ is the weighted distance between $\boldsymbol{x}$ and $\boldsymbol{c}_i = (c_{i1}, \cdots, c_{im})^t$, Q_i is the $m \times m$ covariance matrix, the superscript t denotes the transpose of a matrix and the superscript -1 denotes the inverse of a matrix. The covariance matrix is set in three ways:

1) a constant diagonal matrix with the same diagonal element σ^2 (>0);
2) the diagonal matrix calculated using the set of data around the center $\boldsymbol{c}_i, S_i$, as follows:

$$Q_{i,jj} = \frac{1}{|S_i|} \sum_{\boldsymbol{x} \in S_i} (x_j - c_{ij})^2 \tag{4}$$

where $|S_i|$ is the number of the data in S_i. If the diagonal elements are zero, they are replaced by σ^2,

Manuscript received January 16, 1998.

The author is with the Graduate School of Science and Technology, Kobe University, Kobe, Japan.

Publisher Item Identifier S 1083-4419(99)03540-2.

1083–4419/99$10.00 © 1999 IEEE

3) the nondiagonal matrix given by

$$Q_i = \frac{1}{|S_i|} \sum_{\boldsymbol{x} \in S_i} (\boldsymbol{x} - \boldsymbol{c}_i)(\boldsymbol{x} - \boldsymbol{c}_i)^t. \quad (5)$$

If some diagonal elements of Q_i are zero, they are replaced with the positive σ^2.

Instead of using constant o_i in (1), we can use the linear combination of input variables as follows [12]:

$$o_i = p_{0i} + p_{1i}x_1 + \cdots + p_{mi}x_m \text{ for } i = 1, \cdots, N \quad (6)$$

where $p_{0i}, p_{1i}, \cdots, p_{mi}$ are constants and are determined by the least-squares method. This is called the Takagi–Sugeno type model.

B. Defuzzification Methods

The output of the fuzzy rules $R_i (i = 1, \cdots, N)$, $\hat{y}(\boldsymbol{x})$, for the input $\boldsymbol{x}$ can be synthesized by the center-of-gravity method and the linear combination of the degrees of membership. By the center-of-gravity method $\hat{y}(\boldsymbol{x})$ is given by [1]

$$\hat{y}(\boldsymbol{x}) = \frac{\sum_{i=1}^{N} o_i m_i(\boldsymbol{x})}{\sum_{i=1}^{N} m_i(\boldsymbol{x})} \quad (7)$$

where o_i are constant or calculated by (6).

If we use constant o_i, to improve approximation accuracy, we can introduce additional parameters w_i as follows:

$$\hat{y}(\boldsymbol{x}) = \frac{\sum_{i=1}^{N} o_i w_i m_i(\boldsymbol{x})}{\sum_{i=1}^{N} w_i m_i(\boldsymbol{x})} \quad (8)$$

and tune o_i and w_i by the steepest descent method [3], [5].

By the linear combination of the degrees of membership, the output is synthesized by

$$\hat{y} = q_0 + q_1 m_1(\boldsymbol{x}) + \cdots + q_N m_N(\boldsymbol{x}) \quad (9)$$

where $q_0, q_1, \cdots, q_N$ are constants and determined by the least-squares method. This architecture is the radial basis function network. In this case the outputs o_i are not used for defuzzification.

In this paper we consider the following two fuzzy function approximators.

1) FACG in which the center-of-gravity method given by (7) is used, and the outputs o_i given by (6) are used as long as parameters p_{ji} are determined and the constant o_i are used if p_{ji} are not determined because of singularity of the associated matrix.
2) FALC in which output is synthesized by the linear combination of the degrees of membership given by (9).

III. Fuzzy Rule Generation

A. Concept

In general, there are two methods to generate fuzzy rules. One clusters the training data in advance and generates a fuzzy rule for each cluster. The other generates fuzzy rules dynamically until the approximation error meets the required error limit. In the former method, since clustering is not directly linked to approximation errors, it is difficult to find suitable clustering to realize the required error limit. Therefore, in this paper we consider dynamic rule generation for FACG and FALC.

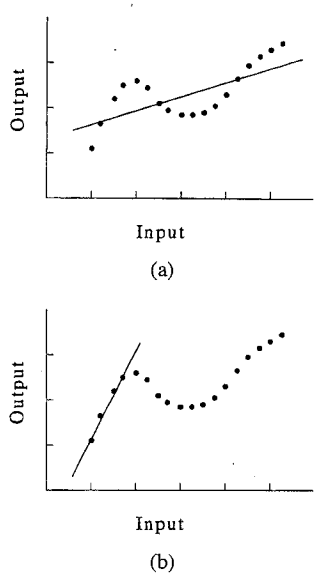

Fig. 1. The first rule generation when the center-of-gravity method is used. (a) When all the training data are used for fitting and (b) when the subset of the training data is used for fitting.

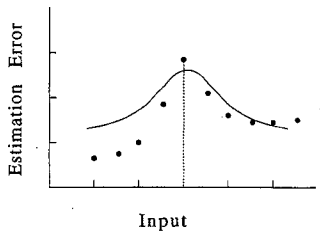

Fig. 2. Rule generation when the output is synthesized by a linear combination of the degrees of membership. A new rule is generated at the point where the approximation error is the maximum.

The selection method of the center of a fuzzy rule needs to be changed according to which of the fuzzy function approximators are used. In FACG, when $N = 1$, (7) reduces to $\hat{y} = o_1$ [see Fig. 1(a)]. Thus, if o_1 is given by (6), the approximation reduces to linear regression. Thus the subsequent rule generation results in compensating the linear regression error. This is not a favorable strategy, since the Takagi–Sugeno type model works to estimate the derivative of the input-output relations and thus can represent the input-output relations with a small number of rules [1]. Thus, for the Takagi–Sugeno type model to work properly, we need to determine the subset of the training data dynamically that should be used to determine the parameters p_{ji} [see Fig. 1(b)]. In Section III-B, we discuss how to determine the subset of the training data.

In FALC, on the other hand, it is effective to add the center of cluster at the point where the approximation error is the maximum [10]. Suppose that N fuzzy rules have been created and a rule needs to be added to reduce the approximation error. Using N fuzzy rules, the output $\hat{y}$ is given by (9). When the $(N + 1)$th fuzzy rule is created the output is given by

$$\hat{y} = q_0 + q_1 m_1(\boldsymbol{x}) + \cdots + q_N m_N(\boldsymbol{x}) + q_{N+1} m_{N+1}(\boldsymbol{x}). \quad (10)$$

Assuming that the values of $q_{0i}, q_{1i}, \cdots, q_{Ni}$, are the same as those determined for the N fuzzy rules, the approximation error is reduced best when the center of the $(N + 1)$th fuzzy rule is located at the point where the approximation error is the maximum (see Fig. 2).

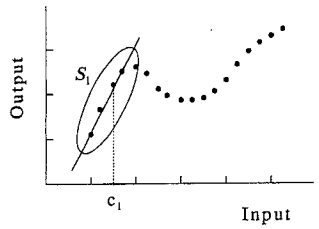

Fig. 3. Concept of clustering. After the center c_1 is selected, the data in S_1 are eliminated from selection of the second center.

B. Rule Generation for FACG

The center of the fuzzy rule needs to be at the center of a cluster where data gather. There are several clustering techniques [13]–[15]. Most of them are iterative methods, that is, they iterate the procedure until the clustering is converged. In [8], for each training datum, the potential is calculated and the training datum with the maximum potential is selected as the cluster center. To avoid consuming the calculation time in clustering, here we use the simplified version of this method. Namely, to generate the first fuzzy rule, for each training input, we count the number of the training inputs that are within the specified distance from the training input in consideration. Then we select the training input that has the maximum number of the training inputs within the specified distance as the center of the first fuzzy rule and determine the parameters $p_{01}, p_{11}, \cdots, p_{m1}$ in (6) by the least-squares method. If the parameters are not determined because the associated matrix is singular, we use the fixed o_i, where the training output corresponding to the selected training input is set to o_i.

The ith $(i > 1)$ fuzzy rule is generated as follows: We delete the training inputs that are within the specified distance from the center of the kth $(k = 1, \cdots, i-1)$ fuzzy rule. For each of the remaining training inputs, we count the number of the training inputs that are within the specified distance from the training input in consideration. Then we select the training input that has the maximum number of the training inputs within the specified distance as the center of the ith fuzzy rule and determine the parameters $p_{0i}, p_{1i}, \cdots, p_{mi}$ by the least-squares method. If the parameters are not determined because the associated matrix is singular, we use the fixed o_i, where the training output corresponding to the selected training input is set to o_i. If there are no training inputs remained but still the approximation error is not within the specified error limit ε, we select the training input whose approximation error is the maximum as the center of the fuzzy rule. By deleting the training inputs that are within the specified distance from the already selected centers, we can avoid selecting the training inputs that are near some of the centers (see Fig. 3).

1) Selection of Centers: Let S denote the set of the initial training inputs. For each training input $\boldsymbol{x}_j$, we count the number of the training inputs $\boldsymbol{x}_k$ that are within the specified, average 1-D distance $R(>0)$

$$\sqrt{\sum_{l=1}^{m} (x_{jl} - x_{kl})^2 / m} < R \tag{11}$$

where R is an application dependent parameter. But since usually the training time is short, we can easily obtain a suitable value as seen from the examples in Section IV. Then we select the training input that has the maximum number of the training inputs within R as the center of the fuzzy rule, $\boldsymbol{c}_1$. Let S_i denote the subset of the training inputs that are within the average 1-D distance of R from $\boldsymbol{c}_i$ and that are included in $S - (S_1 \cup \cdots \cup S_{i-1})$.

To select the ith $(i > 1)$ center, $\boldsymbol{c}_i$, for each of the training inputs in $S - (S_1 \cup \cdots \cup S_{i-1})$, we count the number of the training inputs in $S - (S_1 \cup \cdots \cup S_{i-1})$ whose average 1-D distance is within R. Then we select the training input that has the maximum number of the training inputs within R as the center of the fuzzy rule, $\boldsymbol{c}_i$. If the set $S - (S_1 \cup \cdots \cup S_{i-1})$ is empty, and still the approximation error is not within the specified limit, we select the training input where the approximation error is the maximum as the center $\boldsymbol{c}_i$. For the ith fuzzy rule, we calculate the covariance matrix using (4) or (5).

2) Determination of Parameters $p_{0i}, \cdots, p_{mi}$: Let assume that we have already determined parameters $p_{0k}, \cdots, p_{mk}$ for $k = 1, \cdots, i-1$ and we determine parameters for the ith fuzzy rule, $p_{0i}, \cdots, p_{mi}$, by the least-squares method using the subset of the training inputs $S_1 \cup \cdots \cup S_i$. The reasons why we use the training inputs included in $S_1 \cup \cdots \cup S_{i-1}$ as well as S_i are that the training inputs in $S_1 \cup \cdots \cup S_{i-1}$ may include data that are within the average 1-D distance of R from $\boldsymbol{c}_i$ and to ensure sufficient data to prevent the associated matrix from being singular.

From (7), using i fuzzy rules, $\hat{y}$ for $\boldsymbol{x}$ in $S_1 \cup \cdots \cup S_i$ is given by

$$\hat{y}(\boldsymbol{x}) = a_i(\boldsymbol{x}) o_i + b_i(\boldsymbol{x}) \tag{12}$$

where $a_i(\boldsymbol{x})$ and $b_i(\boldsymbol{x})$ are given by

$$a_i(\boldsymbol{x}) = \frac{m_i(\boldsymbol{x})}{\sum_{k=1}^{i} m_k(\boldsymbol{x})} \tag{13}$$

$$b_i(\boldsymbol{x}) = \frac{\sum_{k=1}^{i-1} o_k m_k(\boldsymbol{x})}{\sum_{k=1}^{i} m_k(\boldsymbol{x})}. \tag{14}$$

Thus by the least-squares method, $p_{0i}, \cdots, p_{mi}$ can be determined using the training data included in $S_1 \cup \cdots \cup S_i$.

C. Rule Generation for FALC

We generate the fuzzy rules successively according to the errors between the training data outputs and the synthesized outputs. In the beginning of rule generation, we consider that the synthesized outputs are zero. We select the training datum that gives the maximum error between the training data outputs and the synthesized outputs. We set the center of the fuzzy rule at the selected datum. We iterate the fuzzy rule generation until the approximation error for the training data is within the specified limit.

Let $(\boldsymbol{x}_j, y_j), j = 1, \cdots, D$ be input-and-output pairs of the training data where D is the number of the training data. We assume that the synthesized outputs $\hat{y}_j$ with no fuzzy rules are zero. In the following we discuss the fuzzy rule generation.

1) Selection of Centers: Initially, we set $\hat{y}_j = 0$ for $j = 1, \cdots, D$. And calculate the maximum approximation error. Let

$$\arg \max_{j=1,\cdots,D} |y_j - \hat{y}_j| \tag{15}$$

be k where arg is the function that returns the subscript of the associated function, in this case max. If

$$|y_k - \hat{y}_k| > \varepsilon \tag{16}$$

where ε is the specified error limit, we generate the first fuzzy rule with the center $\boldsymbol{c}_1 = \boldsymbol{x}_k$. For the ith rule $(i > 1)$ we select the training input that is not previously selected as the center of the fuzzy rule, i.e., $\boldsymbol{c}_i$. Then we calculate the subset S_i of the training inputs that are within the average, 1-D distance of R from $\boldsymbol{c}_i$. We calculate the covariance matrix using (4) or (5).

TABLE I
APPROXIMATION ERRORS FOR THE MACKEY-GLASS TEST DATA

Approximator	NRMSE
NN [16]	0.02
ANFIS [4]	0.007
Cluster Estimation-Based [8]	0.014
FAHB [7]	0.092
FACG[1]	0.006
FALC[2]	0.006

1: constant matrix with $R = 0.05$ and $\sigma = 0.03$

2: diagonal matrix with $R = 0.4$

2) Determination of Parameters for Output Synthesis: We determine the parameters $q_0, q_1, \cdots, q_i$ by the least-squares method using all the training data. We discard the values for $q_0, q_1, \cdots, q_{i-1}$ previously determined by the least-squares method, since, if we use previously determined values, the solution may go into the local minimum. (This actually happened in our simulation studies.)

IV. PERFORMANCE EVALUATION

We evaluated performance of FACG and FALC using two different types of data sets: the noiseless time series data set generated by the Mackey–Glass differential equation [16] and the noisy data set gathered from a water purification plant [17]. We measured the computation time using the SUN Sparcstation 20 model 151.

A. Mackey–Glass Differential Equation

The Mackey–Glass differential equation generates time series data with a chaotic behavior and is given by

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{17}$$

where t and τ denote time and time delay, respectively.

By integrating (17), we can obtain the time series data $x(0), x(1), \cdots, x(t), \cdots$. Using x prior to time t, we predict x after time t. Setting $\tau = 17$, and using four inputs $x(t-18), x(t-12), x(t-6), x(t)$, we estimate $x(t+6)$.

The first 500 data from the time series data $x(118), \cdots, x(1117)$ were used to generate fuzzy rules and the remaining 500 data were used to test performance. This data set is often used as the bench mark data for function approximators and the normalized root-mean-square error (NRMSE), i.e., the root-mean-square error divided by the standard deviation, of the time series data is used to measure the performance. Therefore, we measured the performance by NRMSE.

Table I shows the best performance of the several approximators for the Mackey–Glass test data. The performance of the fuzzy function approximator with hyperbox regions (FAHB) is the worst. Both FACG and FALC give the best performance among them. For FACG a constant covariance matrix with $\sigma = 0.03$ was used and R was set to 0.05 and for FALC a diagonal covariance matrix was used and R was set to 0.4. For both FACG and FALC performance was measured when 100 rules were created. In the following we discuss their performance in detail.

1) Performance of FACG: First we evaluated FACG with a constant covariance matrix. In this case we changed R and σ and calculated the approximation errors when 100 rules were generated. Table II shows the results. For each run the calculation was completed in about 1 min. The NRMSE for the training data and that for the test data are almost the same. This meant that the overfitting was not a problem for the Mackey–Glass data; the NRMSE for the test data decreased as that for the training data decreased. Table III shows the NRMSE's for $R = 0.05$ and $\sigma = 0.03$ for different numbers of fuzzy rules. The NRMSE for the test data decreased monotonically as the rules were generated and reached 0.003 when 200 rules were generated. When 60 rules were generated, the set $S - (S_1 \cup \cdots \cup S_{60})$ became empty and the centers of the subsequent rules were determined according the maximum approximation error.

Instead of setting the diagonal elements of the covariance matrix, we calculated the covariance matrix using the training data. Tables IV and V show the NRMSE's when the diagonal and nondiagonal covariance matrices were used. Since the overfitting occurred during training, we terminated rule generation when the NRMSE for the test data began to increase. Comparing Tables IV and V, FACG with a nondiagonal covariance matrix shows inferior performance while consuming more computation time. Therefore, in the following we do not consider nondiagonal covariance matrices for both FACG and FALC. Comparing Tables II, IV, and V, FACG with a constant covariance matrix shows the best performance. From Tables I and IV, FACG with a diagonal covariance matrix shows comparable performance with that of the cluster estimation-based method [8].

In all the cases, singularity of the matrix in determining parameters in (6) did not occur.

2) Performance of FALC Setting the maximum number of fuzzy rules to be 100, and changing σ for a constant covariance matrix and changing R for a diagonal matrix, fuzzy rules were generated. Tables VI and VII show performance of FALC with constant and diagonal covariance matrices, respectively. From the tables FALC with a diagonal covariance matrix shows a slightly better performance. In Table VI, the number of rules is 81 when $\sigma = 0.20$. This was because the matrix associated with determining parameters $q_0, \cdots, q_{82}$ became singular.

From Tables II, VI, and VII, the smallest approximation errors by FACG and FALC were almost the same but FALC required more computation time since parameters $q_k (k = 0, 1, \cdots, i)$ were determined at the ith rule generation. Table VIII shows the approximation errors of FALC with $R = 0.4$ for different numbers of fuzzy rules. The calculation was terminated when the 125th rule was generated because of the singularity of the matrix associated with determining parameters $q_0, \cdots, q_{126}$. From Tables III and VIII, for the numbers of rules from 40 to 80, the approximation errors of FACB are smaller than those of FALC; this means that FACB can realize the same approximation error that FALC does with a smaller number of fuzzy rules.

B. Water Purification Plant

In a water purification plant, to eliminate small particles floating in the water taken from a river, coagulant is added and the water is stirred while these small particles begin sticking to each other. As more particles stick together they form flocs which fall to the bottom of a holding tank. Potable water is obtained by removing the precipitated flocs and adding chlorine. Careful implementation of the coagulant injection is very important in obtaining high quality water. Usually an operator determines the amount of coagulant needed according to an analysis of the water qualities, observation of floc formation, and prior experience.

To automate this operation, as inputs for water quality, turbidity, temperature, alkalinity, pH, and flow rate were used and to replace the operator's observation of floc properties by image processing, floc diameter, number of flocs, floc volume, floc density, and illumination intensity were used [17].

The 563 input–output data which were gathered over a one-year period were divided into 478 stationary data and 95 nonstationary data according to whether turbidity values were smaller or larger

TABLE II
APPROXIMATION ERRORS OF FACG WITH A CONSTANT COVARIANCE MATRIX FOR THE MACKEY-GLASS DATA

R	σ	NRMSE(train.)	NRMSE(test)	No.Rules	Time(sec)
0.05	0.03	0.006	0.006	100	62
	0.05	0.009	0.008	100	62
	0.07	0.013	0.013	100	62
0.07	0.03	0.006	0.006	100	60
	0.05	0.008	0.008	100	59
	0.07	0.013	0.012	100	59

TABLE III
APPROXIMATION ERRORS OF FACG WITH A CONSTANT COVARIANCE MATRIX FOR THE MACKEY-GLASS DATA FOR DIFFERENT NUMBERS OF FUZZY RULES ($R = 0.05, \sigma = 0.03$)

No.Rules	NRMSE(train.)	NRMSE(test)
20	0.238	0.263
40	0.014	0.013
60	0.010	0.010
80	0.007	0.007
100	0.006	0.006
150	0.004	0.004
200	0.003	0.003

TABLE IV
APPROXIMATION ERRORS OF FACG WITH A DIAGONAL COVARIANCE MATRIX FOR THE MACKEY-GLASS DATA

R	NRMSE(train.)	NRMSE(test)	No.Rules	Time(sec)
0.04	0.019	0.021	64	37
0.05	0.013	0.012	44	20
0.06	0.018	0.017	34	14
0.07	0.022	0.022	27	10
0.08	0.028	0.028	24	9

TABLE V
APPROXIMATION ERRORS OF FACG WITH A NONDIAGONAL COVARIANCE MATRIX FOR THE MACKEY-GLASS DATA

R	NRMSE(train.)	NRMSE(test)	No.Rules	Time(sec)
0.04	0.033	0.039	48	45
0.05	0.015	0.015	38	30
0.06	0.021	0.019	38	28
0.07	0.027	0.027	27	17
0.08	0.031	0.033	25	15

TABLE VI
APPROXIMATION ERRORS OF FALC WITH A CONSTANT COVARIANCE MATRIX FOR THE MACKEY-GLASS DATA

σ	NRMSE(train.)	NRMSE(test)	No.Rules	Time(sec)
0.05	0.038	0.038	100	110
0.10	0.013	0.012	100	107
0.15	0.007	0.007	100	109
0.20	0.009	0.008	81	65

than a specified value. Then each type of data were further divided into two groups to form a training data set and a test data set; division was done in such a way that both sets had similar distributions in the output space. The data sets used in this study were 241 training data and 237 test data for stationary data, and 45 training data and 40 test data for nonstationary data.

TABLE VII
APPROXIMATION ERRORS OF FALC WITH A DIAGONAL COVARIANCE MATRIX FOR THE MACKEY-GLASS DATA

R	NRMSE(train.)	NRMSE(test)	No.Rules	Time(sec)
0.1	0.028	0.028	100	103
0.2	0.009	0.009	100	116
0.3	0.008	0.007	100	104
0.4	0.006	0.006	100	111
0.5	0.007	0.006	100	108

TABLE VIII
APPROXIMATION ERRORS OF FALC WITH A DIAGONAL COVARIANCE MATRIX FOR THE MACKEY-GLASS DATA FOR DIFFERENT NUMBERS OF FUZZY RULES ($R = 0.4$)

No.Rules	NRMSE(train.)	NRMSE(test)
20	0.039	0.038
40	0.024	0.024
60	0.014	0.014
80	0.010	0.009
100	0.006	0.006
125	0.005	0.005

TABLE IX
APPROXIMATION ERRORS (IN MILLIGRAMS PER LITER) FOR THE STATIONARY DATA

Approximator	Training data		Test data	
	Ave.error	Max.error	Ave.error	Max.error
NN	0.84	4.75	0.99	6.95
FAHB	1.07	4.75	1.18	5.57
FACG[1]	1.01	7.35	1.12	4.80
FALC[2]	1.04	3.75	1.31	5.30

1: $R = 0.3$ and $\sigma = 0.15$, 2: $R_1 = 0.1$ and $\sigma = 0.3$

To compare performance of FACG and FALC with that of the multilayered neural network and the fuzzy function approximator with hyperbox regions [5], we calculated the average approximation error and the maximum approximation error. Since both the stationary and nonstationary data are noisy, overfitting occurs; thus we set the approximation error limit ε to 4.0 in the following study.

1) Performance Evaluation for the Stationary Data: Table IX shows the best performance for the stationary data using the multilayered neural network (NN), the function approximator with hyperbox

TABLE X
APPROXIMATION ERRORS (IN MILLIGRAMS PER LITER) OF FACG WITH A DIAGONAL COVARIANCE MATRIX ($\varepsilon = 4.0$) FOR THE STATIONARY DATA

R	σ	Training data		Test data		No.Rules	Time(sec)
		Ave.error	Max.error	Ave.error	Max.error		
0.2	0.1	0.87	9.48	1.21	11.8	14(7)	3
	0.15	0.80	3.94	1.27	14.7	10(2)	3
	0.2	0.86	3.80	1.11	8.39	10(2)	3
0.3	0.1	1.02	7.37	1.14	5.40	7	2
	0.15	1.01	7.35	1.12	4.80	7	2
	0.2	1.01	7.84	1.10	5.46	7	2

(): the number of fuzzy rules with fixed outputs

TABLE XI
APPROXIMATION ERRORS (IN MILLIGRAMS PER LITER) OF FALC WITH A DIAGONAL COVARIANCE MATRIX FOR THE STATIONARY DATA ($\varepsilon = 4.0, R = 0.1$)

σ	Training data		Test data		No.Rules	Time(sec)
	Ave.error	Max.error	Ave.error	Max.error		
0.1	1.30	3.98	1.48	7.85	11	1
0.2	1.04	3.62	1.30	7.42	13	1
0.3	1.04	3.75	1.31	5.30	10	1
0.4	1.19	3.99	1.45	7.30	9	1

regions (FAHB) [5], FACG, and FALC. The four methods are comparable; the maximum errors of FACG and FALC for the test data are smaller than NN and FAHB but the average approximation error of FALC for the test data is largest and the maximum approximation error of FACG for the training data is largest.

In the following, we discuss performance of FACG and FALC more in detail. Table X shows the approximation errors of FACG with a diagonal covariance matrix for different R and σ. The best performance was obtained for $R = 0.3$ and $\sigma = 0.15$. The reason why performance changed for different values of σ is that some of the diagonal elements of the covariance matrix became zero sometimes during iterations and σ^2 needed to be set to the diagonal elements. The numbers in the parentheses are the numbers of fuzzy rules with fixed outputs because of singularity of the matrix associated with determining parameters p_{ij}. The maximum error for the training data exceeded the approximation error limit ε of 4.0. This happened as follows. When a fuzzy rule was generated, the approximation error for the training datum that had been selected as a center of a fuzzy rule exceeded 4.0. But since this datum had been selected as the center, a reduction in the approximation error for this datum was not realized.

Sometimes the sixth to tenth diagonal elements of the covariance matrix became zero. Therefore, we deleted the sixth to the tenth input data and trained FACG for the values of R and σ shown in Table X. The approximation errors did not changed for different σ for $R = 0.3$ but they changed for $R = 0.3$ since the first to fifth diagonal elements became zero. But the approximation errors reduced; For $R = 0.2$ and $\sigma = 0.15$, the average (maximum) approximation error was 0.78 (4.35) mg/l for the training data and the average (maximum) approximation error was 0.97 (5.16) mg/l for the test data. This meant that input data obtained by image processing were not necessary for approximation of the stationary data using FACG.

Table XI shows approximation errors of FALC for $R = 0.1$ and for different values of σ. Unlike FACB the maximum errors of the training data did not exceed the approximation error limit ε of 4.0. In addition, the deletion of the sixth to the tenth inputs did not improve the approximation errors.

TABLE XII
APPROXIMATION ERRORS (IN mg/l) FOR THE NONSTATIONARY DATA

Approximator	Training data		Test data	
	Ave.error	Max.error	Ave.error	Max.error
NN	1.59	6.83	1.74	6.78
FAHB	1.56	7.20	1.46	4.97
FACG[1]	1.91	6.30	1.95	7.18
FALC[2]	1.28	2.85	2.38	8.93

1: $R = 0.22$ and $\sigma = 0.1$, 2: $R = 0.3$ and $\sigma = 0.1$

2) Performance Evaluation for the Nonstationary Data: Table XII shows performance of the multilayered neural network, the function approximator with hyperbox regions (FAHB), FACG, and FALC for the nonstationary data. The maximum approximation errors of FACG and FALC for the test data are comparable to that of the multilayered neural network, but the average approximation errors are worse than the multilayered neural network and FAHB. Table XIII shows performance of FACG fixing $\sigma = 0.1$ and changing R. Performance is sensitive to the value of R and overfitting occurred except for $R = 0.2$ and 0.22. Since the number of the training data was small, the matrix associated with determining parameters p_{ij} became singular except for $R = 0.5$.

Table XIV shows performance of FALC fixing $\sigma = 0.1$ and changing R. The maximum approximation errors for the test data are not good.

We deleted the fifth to the tenth inputs. The approximation errors of FACB and FALC were comparable to those with ten input variables and significant improvement was not obtained.

V. DISCUSSIONS

We evaluated performance of FACG and FALC using the data generated by the Mackey–Glass differential equation and the data gathered from the water purification plant. The former data are

TABLE XIII
APPROXIMATION ERRORS (IN MILLIGRAMS PER LITER) OF FACG WITH A DIAGONAL COVARIANCE MATRIX FOR THE NONSTATIONARY DATA ($\varepsilon = 4.0, \sigma = 0.1$)

R	Training data		Test data		No.Rules	Time(sec)
	Ave.error	Max.error	Ave.error	Max.error		
0.18	0.80	11.9	20.7	455	8(2)	1
0.2	1.72	7.07	2.42	6.63	8(6)	1
0.22	1.91	6.30	1.95	7.18	9(6)	1
0.24	1.69	6.51	153	5760	8(5)	1
0.3	1.65	7.18	116	2370	7(4)	1
0.4	1.53	5.82	2.74	36.3	4(1)	1
0.5	1.43	5.83	3.69	80.4	4	1

(): the number of fuzzy rules with fixed outputs

TABLE XIV
APPROXIMATION ERRORS (IN MILLIGRAMS PER LITER) OF FALC WITH A DIAGONAL COVARIANCE MATRIX FOR THE NONSTATIONARY DATA ($\varepsilon = 4.0, \sigma = 0.1$)

R	Training data		Test data		No.Rules	Time(sec)
	Ave.error	Max.error	Ave.error	Max.error		
0.1	0.96	3.48	2.77	14.2	14	1
0.2	1.32	3.24	2.52	12.6	10	1
0.3	1.28	2.85	2.38	8.93	11	1
0.4	1.10	2.82	2.39	13.1	13	1
0.5	1.35	3.82	2.47	14.3	10	1

noise-free and the training data and the test data are quite similar. Thus performance of approximators is determined by how well the approximators fit the training data. (In practice, 100 or 200 fuzzy rules may be impractical in expressing such a simple system. But here, our function approximators are tested for those numbers of fuzzy rules only to show the approximation power of the approximators.) On the other hand, the data obtained from the water purification plant are very noisy and the training data and the test data are not similar. Thus, we need to avoid overfitting to obtain good performance for the test data. Both FACG and FALC performed well for both data except for the nonstationary data obtained by the water purification plant.

For the Mackey–Glass data, the numbers of fuzzy rules generated by FACG and FALC are larger than those by the cluster estimation based method [8] or ANFIS [4]. Our methods are based on simpler algorithms and thus need more fuzzy rules to realize the same approximation error. The cluster estimation-based method and FACG are similar methods; the major difference is that the former method uses sophisticated pre-clustering while the latter adopts the simpler dynamic clustering. For the Mackey–Glass data the number of fuzzy rules of the cluster estimation-based method to realize the NRMSE of 0.014 is 25 [8]. From Table III, the number of FACG is 40, which is larger.

Unlike pattern classification [18], we need to set the value of the average, 1-D distance from the center, R. In addition, we need to set the value of σ, to avoid the inverse of the covariance matrix from being singular. But since the computation time for generating fuzzy rules is very short, trial and error for determining the optimal values for parameters R and σ is not a problem.

The main reason that the FACB and FALC did not perform well for the nonstationary data is that the number of data is very small: only 45 data points. Thus a fraction of 45 data were used to generate fuzzy rules and overfitting occurred quite easily. To improve the generalization ability of FACB and FALC for the small number of data is one of the subjects of the future study.

VI. CONCLUSIONS

In this paper we discussed two types of fuzzy function approximators that dynamically generate fuzzy rules with ellipsoidal regions: FACG based on Takagi–Sugeno type model with the center-of-gravity defuzzification and FALC based on a radial basis function network.

In FACG, for each training datum the number of the training data that are within the specified distance is calculated and the training datum which has the maximum number of the training data is selected as the center of a fuzzy rule and the covariance matrix is calculated using the training data around the center. Then the parameters of the linear equation that defines the output value of the fuzzy rule is determined by the least-squares method using the training data around the center.

In FALC, the training datum with the maximum approximation error is selected as the center of a fuzzy rule, and using the training data around the center, the covariance matrix is calculated. Then the parameters of a linear equation that determines the output value are calculated by the least-squares method.

Performance of FACG and FALC was compared with that of multilayered neural networks and other fuzzy function approximators for the data generated by the Mackey–Glass differential equation and the data from a water purification plant.

REFERENCES

[1] S. Abe, *Neural Networks and Fuzzy Systems: Theory and Applications*. Boston, MA: Kluwer, 1996.
[2] L.-X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1414–1427, 1992.
[3] C.-T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1320–1336, 1991.
[4] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, 1991.

[5] S. Abe and M.-S. Lan, "Fuzzy rules extraction directly from numerical data for function approximation," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 1, pp. 119–129, 1995.
[6] R. Thawonmas, S. Abe, and A. Cichocki, "A fuzzy function approximator based on partitioned hyperboxes," in *Joint Pacific Asian Conf. Expert Systems/Singapore Int. Conf. Intelligent Systems (PACES/SPICIS '97)*, Singapore, Feb. 1997, pp. 537–544.
[7] J. Mitchell and S. Abe, "Fuzzy clustering networks: Design criteria for approximation and prediction," *Trans. Inst. Electron., Inform., Commun, Eng. Jpn.*, vol. E79-D, no. 1, pp. 63–71, 1996.
[8] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *J. Intell. Fuzzy Syst.*, vol. 2, pp. 267–278, 1994.
[9] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.
[10] R. Katayama, M. Watanabe, K. Kuwata, Y. Kajitani, and Y. Nishida, "Performance evaluation of self generating radial basis function for function approximation," in *Proc. Int. Joint Conf. Neural Networks*, Nagoya, Japan, vol. 1, pp. 471–474, Oct. 1993.
[11] J. A. Dickerson and B. Kosko, "Fuzzy function approximation with ellipsoidal rules," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, no. 4, pp. 542–560, 1996.
[12] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. 15, no. 1, pp. 116–132, 1985.
[13] M. P. Windham, "Cluster validity for the fuzzy c-means clustering algorithm," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 4, no. 4, pp. 357–363, 1982.
[14] J. C. Bezdek and R. J. Hathaway, "Convergence theory for fuzzy c-means: Counterexamples and repairs," *IEEE Trans. Syst., Man, Cybern.*, vol. 17, no. 5, pp. 873–877, 1987.
[15] T. Kohonen, *Self-Organization and Associative Memory*, 2nd ed. Berlin, Germany: Springer-Verlag, 1987.
[16] R. S. Crowder, "Predicting the Mackey–Glass time series with cascade-correlation learning," in *Proc. 1990 Connectionist Models Summer School*, Carnegie Mellon University, Pittsburgh, PA, 1990, pp. 117–123.
[17] K. Baba, I. Enbutsu, and M. Yoda, "Explicit representation of knowledge acquired from plant historical data using neural network," in *Proc. Int. Joint Conf. Neural Networks*, vol. 3, San Diego, CA, June 1990, pp. 155–160.
[18] S. Abe and R. Thawonmas, "A fuzzy classifier with ellipsoidal regions," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 3, pp. 358–368, 1997.

An Adaptive Fuzzy Controller Based on Sliding Mode for Robot Manipulators

F. C. Sun, Z. Q. Sun, and G. Feng

***Abstract*—This paper considers adaptive fuzzy control of robotic manipulators based on sliding mode. It is first shown that an adaptive fuzzy system with the system representative point (RP, or as is often termed, a switching function in variable structure control (VSC) theory) and its derivative as inputs, can approximate the robot nonlinear dynamics in the neighborhood of the switching hyperplane. Then a new method for designing an adaptive fuzzy control system based on sliding mode is proposed for the trajectory tracking control of a robot with unknown nonlinear dynamics. The system stability and tracking error convergence are also proved by Lyapunov techniques.**

***Index Terms*—Adaptive fuzzy systems, adaptive tracking control, robot manipulators, sliding mode control.**

I. Introduction

Sliding mode control and fuzzy control are two attractive methods for robot control. However, Both of them suffer from some problems. The sliding mode control has been confirmed as an effectively robust control approach for nonlinear systems against parameters and load variations during the past decades. However, some bounds on system uncertainties must be estimated in order to guarantee the stability of the closed-loop system, and its implementation in practice will cause a chattering problem, which is undesirable in application [1]. As for the fuzzy control, though it is one of the most effective methods using expert knowledge without knowing the parameters and structure of the nonlinear systems, the major drawback is lack of adequate analysis and design techniques. To overcome their demerits, a novel approach called fuzzy sliding mode control which combines the attractive features of fuzzy control and sliding mode control, has appeared and is receiving more and more attention [2]–[8], [10]–[12]. By introducing the concept of sliding mode to the fuzzy control and fuzzifying the sliding surface, the chattering in a sliding mode control system can be alleviated, and the fuzzy control rules can be determined systematically by the reaching condition of the sliding mode control. These features makes it practical to design an adaptive fuzzy controller based on sliding mode.

There are two main kinds of fuzzy sliding mode control approaches. One is for systems with model partially known where fuzzy control is used to properly adjust the feedback gain in the conventional sliding mode control system. The resulting sliding mode control is so moderate that the RP can reach the switching surface smoothly and do not cross it frequently. As a result, alleviation of chattering and robust performance can be achieved without sacrificing invariant property [2], [3]. The other is for systems with model unknown where fuzzy control is used to reconstruct the system

Manuscript received August 27, 1996; revised September 18, 1998. This work was supported by the National Science Foundation of China under Grant 69685002.

F. C. Sun and Z. Q. Sun are with the Department of Computer Science and Technology, National Laboratory of Intelligence Technology & Systems, Tsinghua University, Beijing 100084, China.

G. Feng is with the School of Electrical Engineering, University of New South Wales, Sydney 2052, NSW, Australia.

Publisher Item Identifier S 1083-4419(99)03539-6.

1083–4419/99$10.00 © 1999 IEEE